PARALLELIZED ARCHITECTURES FOR LOW LATENCY TURBO STRUCTURES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ORHAN GAZİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 2007

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. İsmet Erkmen
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Assist. Prof. Dr. A. Özgür Yılmaz
Supervisor

Examining Committee Members

Prof. Dr. Yalçın Tanık	(METU, EE)	
Assist. Prof. Dr. A. Özgür Yılmaz	(METU, EE)	
Assoc. Prof. Dr. Melek. D. Yücel	(METU, EE)	
Assist. Prof. Dr. Çağatay Candan	(METU, EE)	
Assist. Prof. Dr. Emre Aktaş	(Hacettepe University, EE)	

ABSTRACT

PARALLELIZED ARCHITECTURES FOR LOW LATENCY TURBO STRUCTURES

Gazi, Orhan
Ph. D., Department of Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. A. Özgür Yılmaz

January 2007, 143 pages

In this thesis, we present low latency general concatenated code structures suitable for parallel processing. We propose parallel decodable serially concatenated codes *(PDSCCs)* which is a general structure to construct many variants of serially concatenated codes. Using this most general structure we derive parallel decodable serially concatenated convolutional codes *(PDSCCCs)*. Convolutional product codes which are instances of *PDSCCCs* are studied in detail. *PDSCCCs* have much less decoding latency and show almost the same performance compared to classical serially concatenated convolutional codes. Using the same idea, we propose parallel decodable turbo codes *(PDTCs)* which represent a general structure to construct parallel concatenated codes. *PDTCs* have much less latency compared to classical turbo codes and they both achieve similar performance.

We extend the approach proposed for the construction of parallel decodable concatenated codes to trellis coded modulation, turbo channel equalization, and space time trellis codes and show that low latency systems can be constructed using the same idea. Parallel decoding operation introduces new problems in implementation. One such problem is memory collision which occurs when multiple decoder units attempt accessing the same memory device. We propose novel interleaver structures which prevent the memory collision problem while achieving performance close to other interleavers.

Keywords: iterative decoding, concatenated codes, soft decision decoding algorithms, parallel processing, decoding latency, memory collision, turbo equalization, trellis coded modulation, space time trellis coding

ÖZ

DÜŞÜK GECİKMELİ PARALELLEŞTİRİLMİŞ TURBO YAPILAR

Gazi, Orhan
Doktora, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Assist. Prof. Dr. A. Özgür Yılmaz

Ocak 2007, 143 sayfa

Bu tez calışmasında paralel işlemeye elverişli düşük gecikmeli birleşik kod yapıları öneriyoruz. Paralel çözümlenebilir seri birleşik kodları *(PÇSBK)* ortaya koyuyoruz. *PÇSBK'*lar seri birleşik kodlar icin genel bir yapı ifade etmektedir. Bu genel yapıyı kullanarak paralel çözümlenebilir seri birleşik konvolusyonel kodları *(PÇSBKK)* öneriyoruz. *PÇSBKK'*lar oldukça düşük gecikme sürelerine sahip olmakla birlikte seri birleşik konvolusyonel kodlar ile aynı performansı göstermektedirler. Daha sonra aynı fikirden yola çıkarak paralel çözümlenebilir turbo kodları *(PÇTC)* tanıtıyoruz. *PÇTC'*lar paralel birleşik kodlar için genel bir yapı ifade etmektedirler. *PÇTC'*lar klasik turbo kodlarla aynı performansı sergilemekte ve de turbo kodlara göre çok daha düşük çözümlenme sürelerine sahiptirler.

Önerilen kod yapılarında kullanılan mantığı, örgü kodlanmış modülasyona, turbo kanal denkleştirmeye ve uzay zaman örgü kodlama yapılarına uyarlayarak daha hızlı calışan iletişim sistemleri elde etmenin mümkün olduğunu da çalışmalarımızda göstermekteyiz. Paralel çözümle işleminin gerçeklenmesi esnasında bazı sorunlarla karşılaşılmaktadır. Bellek çarpışması bu sorunların en başta gelenlerinden birisidir. Bellek çarpışmasını önlemek amacıyla yeni bir serpiştirici yapısı öneriyoruz. Önerilen serpiştirici üniteleri, yaygın olarak kullanılan serpiştiricilere benzer performans göstermekte ve ek olarak bellek çarpışma sorununu engellemektedir.

Anahtar Kelimeler: yinelemeli çözüm, birleşik kodlar, yumuşak çözümleme algoritmaları, paralel işleme, çözüm gecikmesi, bellek çarpışması, turbo denkleştirme, örgü kodlanmış modülasyon, uzay zaman örgü kodlama

To Good People

ACKNOWLEDGMENTS

I would like to express my special thanks to Assist. Prof. Dr. A. Özgür Yılmaz for his guidance and support during my thesis work. I have benefited from his deep knowledge and his intellectual approach to research. I have gained too much from his discipline on research. I am grateful to Dr. Emre Aktaş for his invaluable suggestions for my research. I would like to thank Assoc. Prof. Dr. Melek Yücel for kindly being in my Ph.D. committee. I would like to thank to my family for all their support and love. My special thanks goes to my niece Yağmur and my nephew Gürkan whose love gave me energy during my Ph.D. work.

PREFACE

A three years study was spent for this doctoral thesis. The thesis was written according to the chronological order of progress. There are some future studies, yet the main ideas are clearly presented in the thesis. Some parts of this thesis have been published in journals, accepted for publication, or to be submitted for publication. Some were also presented in conferences or workshops.

Journal Papers

1. O. Gazi, A. O. Yılmaz, "Turbo Product Codes Based on Convolutional Codes," *ETRI Journal,* vol. 28, no. 4, Aug. 2006, pp. 453-460,
2. O. Gazi, A. O Yılmaz, "Zero State Doped Turbo Equalizer," *IEEE Communications Letters,* accepted for publication, Nov. 2006.
3. O. Gazi, A. O. Yılmaz, "Fast Decodable Turbo Codes," *IEEE Communications Letters,* accepted for publication, Nov. 2006.
4. O. Gazi, A. O. Yılmaz, "Analysis of Parallel Decodable Turbo Codes," *IEEE Transactions on Wireless Communications*, to be submitted.
5. O. Gazi, A. O. Yılmaz, "Bit vs Symbol Interleaved Joint Structures for *TCM*, *MIMO*, *PDSCCCs*," *IEEE Transactions on Wireless Communications*, in preparation

Conference Papers

1. O. Gazi, A. O. Yılmaz, "On Parallelized Concatenated Codes," *IEEE Wireless Communications and Networking Conference,* Hong Kong, 11-15 March 2007, Accepted.

2. O. Gazi, A. O. Yılmaz, "Konvolusyonel Kodların Performans Analizi," *İTÜSEM İletişim Teknolojileri Sempozyumu 2005,* pp. 171-176, Çukurova Üniversitesi, Adana-Türkiye.

3. O. Gazi, A. O. Yılmaz, "Serpiştiricinin Konvolusyonel Çarpım Kodlarının Performansı Üzerindeki Etkileri," *URSI Türkiye Ulusal Komitesi 3. Bilimsel Kongresi,* Hacettepe Üniversitesi, pp. 471-474, 6-8 Eylül 2006, Ankara-Türkiye.

4. O. Gazi, A. O. Yılmaz, "Parallel Decodable Concatenated Convolutional Codes," *ISEECE 2006,* pp. 1-7, Nov. 23-25, 2006, Nicosia, North Cyprus.

5. O. Gazi, A. O. Yılmaz, "Doğrusal Kodların Spektrum Ağırlık Fonksiyonlarının Hesaplanması," *ELECO 2006,* pp. 149-154, 6-10 Aralık 2006, Bursa-Türkiye.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Information theory was born within the field of communications with the publication of Shannon's paper in 1948 [1]. In his paper, Shannon stated that it was possible to transmit data reliably over a communication channel at a rate lower than the channel capacity if suitable error correction codes are used. Over fifty years researchers are developing error correcting codes to approach the limits determined by Shannon. Almost at the same time as Shannon's work, the first error correcting codes were introduced by Hamming [2] and Golay [3]. The general approach in both Hamming and Golay codes was the same, they divided information symbol blocks into sub-blocks of length k and they added $n-k$ parity symbols to the end of these sub-blocks. The resulting code is referred as a block code and referred to a (n, k) code.

Other block codes discovered are Reed-Muller, cyclic codes, BCH, and Reed Solomon (RS) codes [4, 5]. Reed Muller codes were more flexible for the number of codewords and the number of correctable errors per codeword when compared to Hamming and Golay codes. Reed Muller codes were popular during 1970's and widely used in space-craft communication. BCH constitutes an important class of cyclic codes. RS codes are the non-binary extension of BCH codes. Berlekamp introduced an efficient decoding algorithm for Reed Solomon codes in 1960 [6] which led to widespread application of RS codes in practical systems such as compact disc players (CD), digital versatile disc (DVD) players, and cellular digital packet data standard.

Having a frame oriented structure, block codes have some drawbacks. Some of these drawbacks are that the entire frame must be received before decoding starts, frame synchronization is needed, codeword size is not flexible etc. Convolutional codes are first introduced by Elias in 1955 [7]. Rather than grouping parity and data bits separately, convolutional codes mix data and parity bits uniformly. The convolutional encoding operation is performed continuously using shift registers. Decoding can also be done continuously which achieves a significant reduction in latency compared to block codes. Convolutional codes became popular and were applied in many communication systems after the introduction of the Viterbi algorithm in 1967 [8]. For instance, the GSM standard uses a convolutional code. It is also widely used in deep space communications [9]. Ungerboeck combined convolutional coding and modulation processes and introduced trellis coded modulation [10] in 1976. Trellis coded modulation was adopted in use for telephone modems and for many satellite communication applications [11].

In his paper Shannon stated that it is possible to achieve error free transmission using long channel codes. Some researchers attempted constructing long codes by concatenating codes. The earliest study in this field was done by Elias in [7] where product codes were introduced. Elias used two block codes sequentially to encode the information frame. In 1966 Forney [12] concatenated a block code and a convolutional code, where he used soft decision decoding for the inner code. The interest in soft decision decoding and interleaver use multiplied after Forney's work. For this reason, many researchers refer to Forney as the founder of the concatenated codes.

Turbo codes which are also called parallel concatenated convolutional codes were introduced in 1993 [13]. Turbo codes approached the Shannon limit the most when compared to all the available codes at that time. This flourished a large volume of research on concatenated codes and soft decision algorithms. Turbo codes were pursued by the development of the serially concatenated convolutional codes and block product codes. The success of turbo codes also led to the rediscovery of Galleger's low density parity check codes [14]. The good performance of turbo codes and their derivatives lies on the success of the well known soft decision algorithm BCJR. It was verified both empirically and analytically that it was possible to approach the Shannon limit using soft decision algorithms [13].

The idea of turbo decoding was further extended to channel equalization [15] where turbo equalization was introduced. An equalizer is a signal processing subsystem that mitigates the effects of frequency selective channels. A frequency selective channel can be considered as a rate-1 convolutional encoder that employs real or complex symbols. The combination of a convolutional code and an ISI channel (frequency selective channel) can be considered as a serially concatenated convolutional code. The combined system can be decoded using turbo decoding algorithms.

Although concatenated codes show very good performance at low *SNR* values, their large decoding latency due to complex decoding algorithms and long frame lengths is a major problem in communication systems. Two approaches are generally followed to reduce the decoding latency. One is the development of complexity reduction techniques for decoding algorithms, the other is the construction of more efficient hardware structures. We focus on the second method in this thesis where parallel processing will be utilized.

Parallel processing is a way to reduce the decoding latency. Some channel code families such as block product codes are more suitable for parallel processing operations. However, since these codes are constructed using block codes, a large number of states form for even small frame lengths which prevents the use of marginal a-posteriori (MAP) decoding for these codes and hence suboptimal decoding algorithms are used. Recently new classes of concatenated codes which enable parallel processing are proposed. These are woven convolutional codes [16], convolutional coupled codes [17], and woven turbo codes [18]. In these code structures, block codes or convolutional codes are concatenated in parallel and serial manner.

In this thesis we propose general parallelized structures for concatenated codes. We propose a general structure for parallel decodable serially concatenated codes and using the same idea we introduce parallel decodable turbo structures. The proposed structures are very suitable for parallel processing operations and show comparable performance to their counterparts. For parallel decoding of the concatenated codes, general trend is to devise systems for the receiver side. This type of systems suffer from extra memory use, complexity increase in decoding operations, and performance loss. The most widely known classical method for parallel decoding is the sliding window technique. It is based on dividing data block into small frames and processing each frame separately by different processors. Since this method divides the frames into sub-blocks, as the number of sub-blocks increase, undetermined boundary probabilities worsen code performance seriously. In our work, we follow a different approach. We parallelize encoder side and use it directly at the decoder side for parallel processing. This enables us to have more control on code parameters. Parallelized encoding operation can be illustrated using matrix notation and this matrix structure can be used for the determination of lower and upper bounds for the worst case minimum distance of the code. In addition, by the help of matrix notation the number of parallel decoders for a guaranteed worst case minimum distance can be determined. It is also shown that many proposed structures already available in literature are instances of our general structures. Many other different codes can be generated using the generalized structures proposed in this thesis.

Memory collision is one of the most important problems observed during parallel decoding operation. Memory collision occurs if two or more decoders try to access the same memory segment at the same clock instance. It is due to the permutation order of the interleaver. Using the matrix notation for parallelized encoding operation, specific collision interleavers can be constructed. We propose such a memory collision free novel interleaver which is called the row-column *S-random* interleaver. The proposed interleaver can be handled in a manner to guarantee a lower bound for worst case minimum distance. We also show that, using row-column *S-random* interleaver, it is possible to make a tradeoff between number of parallel decoders and code performance without memory collision. The proposed structures are highly suitable for integration with other communications units such as trellis coded modulation, space time trellis codes, multi-carrier communication. In our work, we study two such systems involving trellis codes modulation and space time trellis codes. The outline of the thesis is as follows.

In Chapter 2, block and convolutional codes are reviewed, decoding methods are discussed, and some fundamental concepts from information theory are given.

Chapter 3 focuses on concatenated codes. Turbo code *(TC)* and serially concatenated convolutional code *(SCCC)* structures are explained. The role of the interleavers on the performance of the *TCs* and *SCCCs* is inspected.

In Chapter 4 parallel decodable serially concatenated convolutional codes *(PDSCCCs)* are introduced. It is shown that the *SCCCs* and block product codes are instances of the proposed structure. Interleaving effects on the performance of *PDSCCCs* are studied. Analytical bounds using the uniform interleaver approach are drawn for *PDSCCC*. Simulation results which include the effects of minimum distance for different interleavers and trellis terminations are reported.

In Chapter 5 we introduce parallel decodable turbo codes *(PDTCs)*. In fact, this is a general structure for parallel concatenated codes. We analyze the effects of interleaving on the performance of *PDTCs*. Simulation results of the proposed system are depicted and latency gain over classical turbo codes are emphasized.

In Chapter 6, we study the extension of *PDSCCCs* and *PDTCs* to turbo equalization and *MIMO* communication systems. Zero state doped turbo equalizer is introduced. Joint structures for *PDSCCC*, trellis coded modulation, and *MIMO* trellis codes are studied. Finally, conclusions and some suggestions for future studies are given in Chapter 7.

CHAPTER 2

MODERN CODING THEORY

In this chapter some basic concepts of information theory will be reviewed. Linear block and convolutional codes will be explained shortly. Trellis representation for linear codes will be presented. Spectrum functions of the convolutional codes will be defined. Bound expressions for probability of bit error will be given. Some properties of the convolutional codes will be inspected.

2.1 Channel Coding and Information Theory

Channel coding has become a requisite apparatus for modern communications systems. Power and bandwidth are two important constraints during the design of channel codes. Coding gain, which is the difference in the signal energy between coded and uncoded communication systems to realize a given bit error probability, is an indicator for the code performance. Nearly 60 years ago it was stated by Shannon that large coding gains can be achieved using long enough coded sequences. Shannon proved the existence of some limits for reliable communication. For almost 60 years researchers are looking for codes approaching Shannon limits. The capacity formula for the additive white Gaussian noise channel is given as

$$C = Wlog(1 + \frac{P}{N_0 W})\ bits/sec, \tag{2.1}$$

where C is the capacity, W is the signal bandwidth, P is the signal power, and N_0 is the noise power spectral density.

It is obvious from eqn. (2.1) that as the signal power increases capacity increases as well. Hence, we can conclude that by increasing the signal power only we can set the channel capacity to any value. For reliable communication the transmission rate (bits per second) should be less than the channel capacity, i.e.,

$$R < Wlog(1 + \frac{P}{N_0 W}). \tag{2.2}$$

Defining $\rho = \frac{C}{W}$ which is called the spectral efficiency, eqn. (2.2) leads to

$$\begin{aligned} P &= R.E_b \\ \rho &= log(1 + \rho.\frac{E_b}{N_0}) \end{aligned} \tag{2.3}$$

where E_b is the energy consumed per data bit. It can be further simplified as:

$$\frac{E_b}{N_0} = \frac{2^\rho - 1}{\rho}. \tag{2.4}$$

This relation is plotted in Fig. 2.1. The reliable communication is possible in the region below the curve. As the R tends to zero, $E_b/N_0 = ln2 \sim -1.6dB$ is the minimum value of E_b/N_0 for reliable communication.

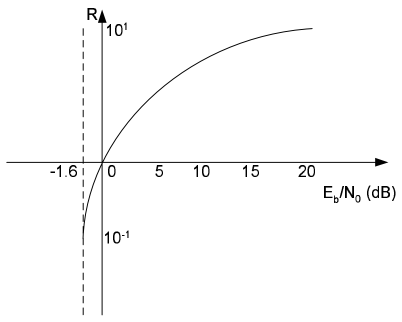


Figure 2.1: Communication Bound for AWGN Channel.

2.2 Error Correcting Codes

Channel coding is the addition of some redundant bits to information bits at the transmitter side. The added redundancy is used for the detection of the transmitted bits at the receiver side. Shannon stated the existence of good channel codes for error free communications. However, he didn't show how to find good channel codes. Linear codes have been dominantly studied due to their analytical tractability. Linear codes mainly consists of two classes of codes: block and convolutional codes. A linear code is a vector space [19]. Each element of the code is called a codeword. Codewords are symbol sequences. The symbols used in codewords are chosen from a finite field F, i.e., code $C = \{c_1, c_2, \ldots, c_{M-1}, c_M\}$, $c_i = [b_1, \ldots, b_n]$, where $b_j \in F$ for $\forall\ i,\ j$. The code rate R is defined as $R = log_q(M)/n$ where q is the number of elements in F. Systematic linear block codes are formed by appending parity check bits to the end of information bits. Convolutional codes do not usually operate in block fashion. However, they can be considered as block codes under some circumstances.

2.2.1 Block Code

A linear block code over a finite field F is a subspace of a vector space F^n. Linear block code elements are n-tuple vectors whose elements are chosen from F. The code elements (n-tuple vectors) are called codewords. An (n, k) block encoder takes a k-tuple data-word and matches this data-word to an n-tuple codeword from the code and uses this n-tuple codeword for transmission. For k-symbol words there are q^k possible number of data words. For n-symbol codewords, there are q^n vectors to be used as codewords, since $n > k$ we have more vectors than the data-words. Hence, sets of vectors can be carefully chosen for use in transmission considering distances among them. When the minimum distance between any codeword pairs is the criterion, the sets of vectors are chosen such that they have largest minimum distances.

At the receiver side, the received vectors can be separated from each other using the Euclidian distance measure. The distinguishability of these codewords at the receiver side gives an indication about the performance of the code. As the distances within the codebook becomes larger, better detection of the transmitted codeword is achieved at the receiver side. Every vector space has basis vectors that generate all the vectors. A generator matrix for a linear code is defined as the matrix whose rows are the basis vectors of the code. The dimension of the generator matrix of an (n, k) code is $k \times n$, i.e., there are k basis vectors of the code.

2.2.2 Trellis Diagrams for Block Codes

The generator matrix of a block code is used to generate the codewords. The parity check matrix of a code is used to generate the codewords of the dual code. The parity check matrix is also used to check the validity of the codewords. Let G denote the generator matrix of a block code and H denote the parity check matrix. For a linear (n, k) code C, size of the generator matrix G is $k \times n$ and the size of the the parity check matrix H is $(n - k) \times n$. Either the parity check matrix or the generator matrix of a block code can be used to generate the trellis diagram states. We explain the trellis diagram construction using the parity check matrix here. A more detailed discussion of the subject can be found in [19].

The parity check matrix H satisfies the equality $c.H^T = 0$ for a codeword c. The block code can also be described by a trellis diagram. The trellis diagram for a linear block code is constructed as follows. Consider the codeword $c = (c_0,\ c_1, \ldots, c_i, \ldots, c_{n-2}, c_{n-1})$ over the binary field $GF(2)$ $c_i \in 0, 1$. The trellis path consists of n stages. Each stage has some states. The number of states in a stage can be at most 2^{n-k}. At the beginning there is only one state which is the zero state and is denoted by $S_0(0)$. In general the states for stage l will be denoted by $S_i(l),\ 0 \leq i \leq 2^{n-k}$. The next state sequence is determined from the previous one according to the following equality

$$S_m(l+1) = S_i(l) \oplus \alpha_i^j h_{l+1}, \tag{2.5}$$

where

$$\alpha_i^j = \begin{cases} 1 & \text{if } j = 1 \\ 0 & \text{if } j = 0, \end{cases} \tag{2.6}$$

h_{l+1} is the $l+1$'th column vector of the parity check matrix, and $\oplus$ denotes the module-2 addition operation.
At trellis depth $l = n$

$$S_0(n) = S_0(0) \oplus \sum_{i=0}^{n-1} \alpha_i^j h_i = 0. \tag{2.7}$$

Since the first and last states are zero stated the above equality means that

$$\sum_{i=0}^{n-1} \alpha_i^j h_i = 0. \tag{2.8}$$

An example is given below to clarify the trellis structure of a block code.

Example: For the parity check matrix

$$H = [h_0\ h_1\ h_2\ h_3] = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \tag{2.9}$$

the trellis diagram is shown in Fig. 2.2. The state diagram is obtained as follows. We start with the zero state (00), then h_0 is multiplied with 0 and added to the zero state and h_0 is multiplied by 1 and added to the zero state, in this manner we find the two next states for zero state. In the second stage, we have two states. We consider those two states separately. For the first state, h_1 is multiplied with 0 and added to the first state, and h_1 is multiplied with 1 and added to the fist state. In this way we find two next state values for the first state of the second stage. These two next states correspond to input bits zero and one respectively. A similar procedure is performed for the second state of the second stage and two next states are obtained. This procedure goes on until the number of stages equals the number of bits in codewords. Not all the sequences produced by the trellis diagram are codewords. The codewords are those words that have path to the final zero state. We remove the nodes and branches that do not have zero state as the final state. This procedure is called the expurgation of the trellis diagram. The expurgated trellis diagram is shown in Fig. 2.3.

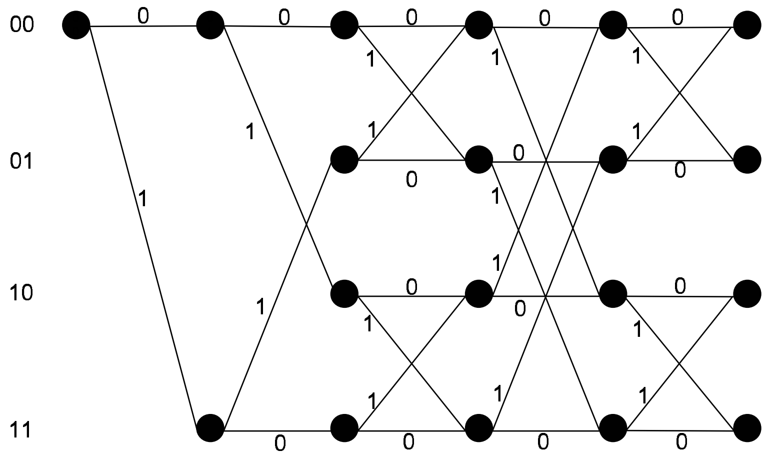


Figure 2.2: Block code trellis diagram.

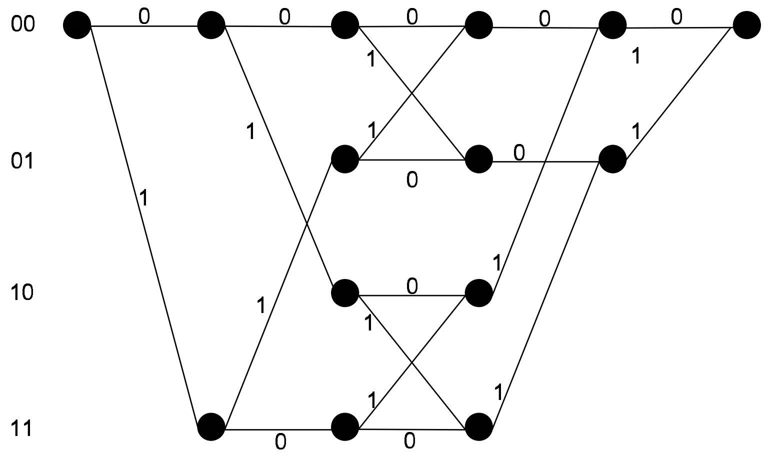


Figure 2.3: Expurgated trellis diagram.

2.3 Convolutional Codes

A binary convolutional encoder consists of binary shift registers and binary adders. Convolutional encoders are finite state machines. The encoding procedure can be illustrated using finite state machines which are represented by state diagrams, graphs, and trellises. The state of the convolutional encoder is determined by the contents of the shift registers. If the longest shift register of the convolutional encoder contains M flip-flops then the encoding procedure can be illustrated using a state machine of 2^{M} states. For each input information sequence the finite state machine produces a binary output sequence.

Convolutional encoders are divided into different categories with regard to encoders outputs. According to the encoding operation convolutional encoders are classified as recursive and non-recursive. They are classified as systematic, which is the case when data bits appear directly at the output, and non-systematic where data bits do not appear directly at the encoder output. A systematic recursive convolutional code has a recursive encoder structure and has data bits at the encoder output. The encoder structure of a non-recursive convolutional code is depicted in Fig. 2.4. If there is a feedback path in the encoder structure, the code is a recursive code. A recursive systematic convolutional encoder (RSC) is depicted in Fig. 2.5.

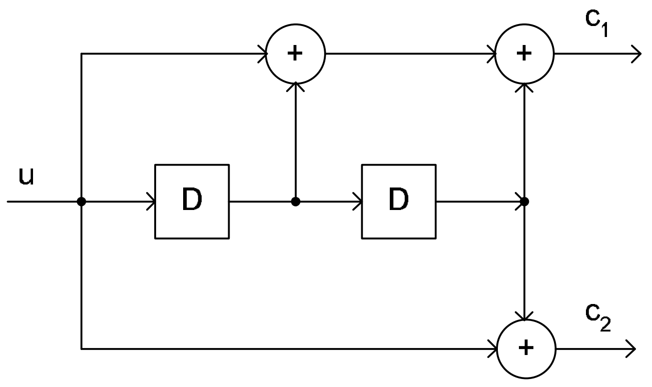


Figure 2.4: Non systematic convolutional encoder with feed forward.

The operation of the RSC in Fig. 2.5 is illustrated in Figs. 2.6, 2.7, 2.8 using trellis diagram, state machine, and state graph.

Graphical illustration of the state diagram is used to find the transfer function of a convolutional code [19]. The transfer function contains all the information about the convolutional code, i.e., the number of codewords with Hamming weight d. The transfer function of the RSC in Fig. 2.5 is found [11] using its state graph as in eqn. (2.10).

$$T(X) = X^5 + 2X^6 + 4X^7 + \dots \tag{2.10}$$

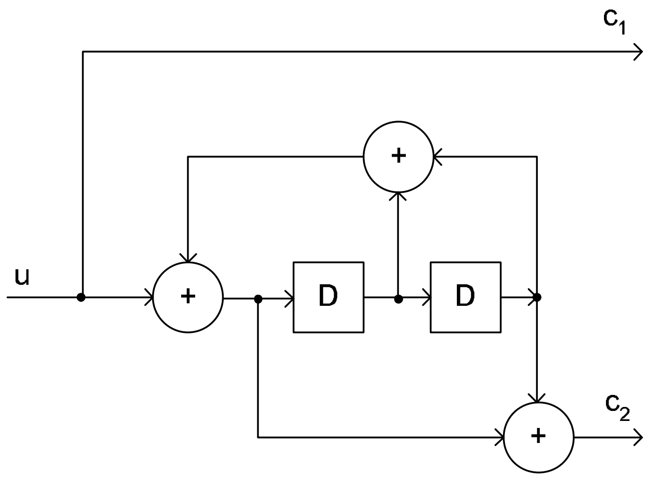


Figure 2.5: Systematic Convolutional encoder with feed back.

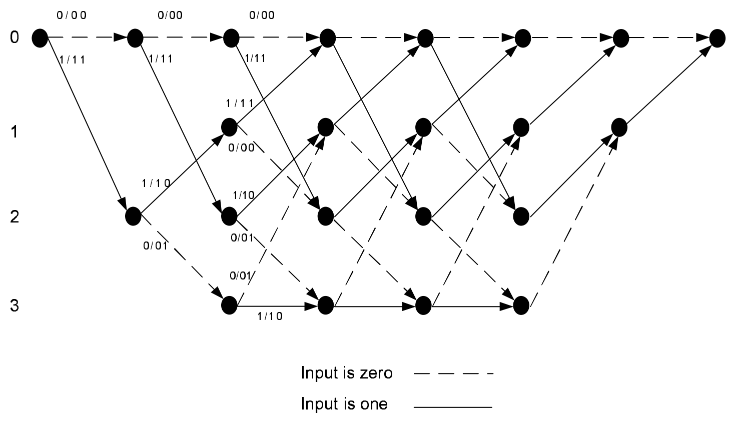


Figure 2.6: $(1, 5/7)_{octal}$ Convolutional encoder trellis diagram.

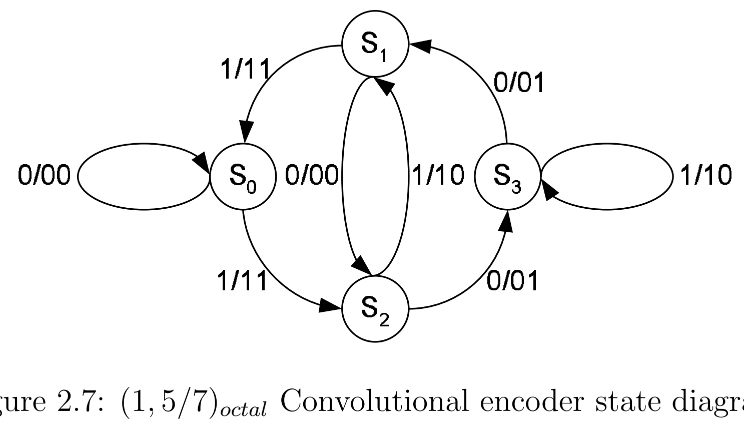


Figure 2.7: $(1, 5/7)_{octal}$ Convolutional encoder state diagram.

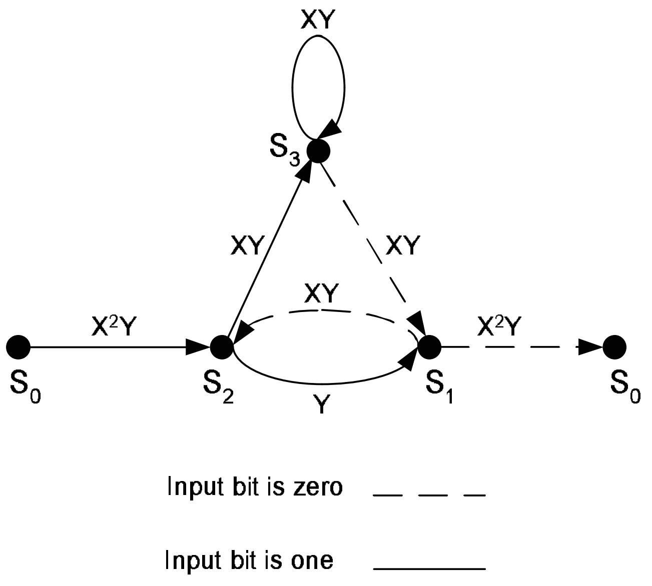


Figure 2.8: $(1, 5/7)_{octal}$ Convolutional encoder state graph.

The lowest power of X in transfer function $T(X)$ denotes the free distance of the convolutional code. If convolutional codes are trellis terminated, they can be considered as block codes. Hence, the studies made on block codes turn out to be valid for the convolutional codes; too.

2.4 Polynomial Representation of Convolutional Codes

For the single stage of a generic convolutional encoder in Fig. 2.9, let u^i be the information bit sequence at input port i, and c^j be the bit sequence at the convolutional encoder output port j. The sequences are given below

$$\begin{aligned} u^i &= [u_{0i}, u_{1i}, \ldots], \\ c^j &= [c_{0j}, c_{1j}, \ldots]. \end{aligned} \tag{2.11}$$

The information and code bit sequences (datawords and codewords) can be expressed in terms of the delay operator as

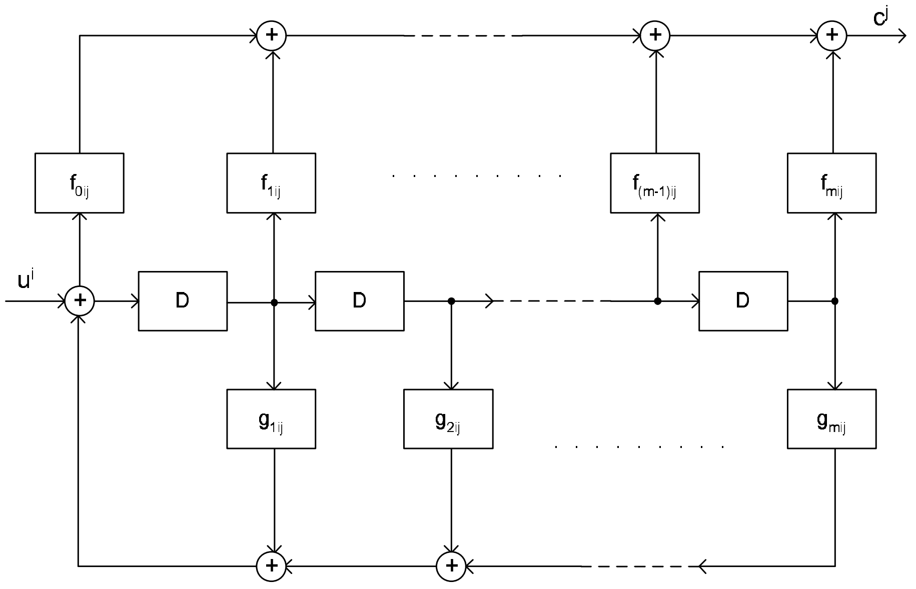


Figure 2.9: Controllable canonical form of a rational transfer function.

$$\begin{aligned} u^i(D) &= [u_{0i} + u_{1i}D^1 + \cdots], \\ c^j(D) &= [c_{0i} + c_{1i}D^1 + \cdots]. \end{aligned} \tag{2.12}$$

Let $f_{ij}(D)$ and $g_{ji}(D)$ be the forward and backward transfer functions of the convolutional encoder between input port i and output port j as depicted in Fig. 2.9 such that

$$\begin{aligned} f_{ij}(D) &= f_{0ij} + f_{1ij}D + \cdots + f_{mij}D^m, \\ g_{ji}(D) &= 1 + g_{1ji}D + \cdots + g_{mji}D^m. \end{aligned} \tag{2.13}$$

The codewords can be written in terms of the datawords using the transfer functions as [20];

$$c^j(D) = u^i(D)T_{ij}(D), \tag{2.14}$$

$$\begin{aligned} T_{ij}(D) &= f_{ij}(D)/g_{ji}(D) \\ &= \frac{f_{0ij} + f_{1ij}D + \cdots + f_{mij}D^m}{1 + g_{1ji}D + \cdots + g_{mji}D^m}. \end{aligned} \tag{2.15}$$

For non-systematic convolutional encoder with R input ports and S output ports, the generator matrix of the code is given as

$$\begin{aligned} [G]_{ij} &= \{T_{ij}(D)\}, \\ i &= 1 \ldots R, \\ j &= 1 \ldots S, \end{aligned} \tag{2.16}$$

$$G = \{T_{11}, T_{12}, \ldots, T_{1S}, T_{21}, \ldots, T_{RS}\}. \tag{2.17}$$

For systematic convolutional encoders the generator matrix using the delay operator is defined as

$$[G]_{ij} = \{1, T_{ij}(D)\}. \tag{2.18}$$

For example, the generator matrix of the convolutional encoder in Fig. 2.5 is $G(D) = (1, 1 + D^2/1 + D + D^2)$. The generator matrix is sometimes expressed in octal form. The convolutional encoder in Fig. 2.5 is also expressed by the generator matrix in octal form $(1, 5/7)_{octal}$.

2.5 Spectrum Functions of Convolutional Codes

Channel codes are described using spectrum functions. They provide information about the codewords, i.e., number of codewords of a specific Hamming weight, number of input sequences with a specific Hamming weight generating codewords of a certain Hamming weight, information about parity check sequences etc. We below summarize the well known spectrum functions. Knowing the spectrum functions of convolutional codes, analytical upper bound expressions for bit error probabilities can be determined. The computation of spectrum functions of convolutional codes is a difficult task. We introduce a method which makes computations of spectrum functions an easy task.

2.5.1 Input Redundancy Weight Enumerating Function

Input redundancy weight enumerating functions *(IRWEFs)* contain parity Hamming weight information. An *IRWEF* is defined as,

$$A(W,Z) = \sum_{w,z} A_{w,z} W^w Z^z, \tag{2.19}$$

where $A_{w,z}$ denotes the number of codewords generated with input sequences of Hamming weight w and having parity check weight of z. The coefficients $A_{w,z}$'s are named as *IRWEF* coefficients.

2.5.2 Weight Enumerating Function

Weight enumerating function gives information about the codewords, i.e., number of codewords with a specific Hamming weight. It is defined as,

$$A(X) = \sum_{i=d_{min}}^{n} A_i X^i, \tag{2.20}$$

where A_i is the number of codewords with Hamming weight i, X is a dummy variable, and d_{min} is the minimum distance of the code. *WEF* gives information only about the codewords. No information is available about the input information sequences or parity weights.

2.5.3 Input Output Weight Enumerating Function

Input Output Weight Enumerating Function *IOWEF* contains information about the Hamming weight of the input sequences along with the produced codewords as in,

$$A(W,X) = \sum_{w,i} A_{w,i} W^w X^i. \tag{2.21}$$

$A_{w,i}$ is the number of codewords of Hamming weight i generated by information sequences of Hamming weight w. W and X are dummy variables.

2.5.4 Conditional Weight Enumerating Function

Conditional weight enumerating function *(CWEF)* gives us information about the weight distribution of the codeword sequences that are generated from a Hamming weight w input sequences as defined in

$$A(w, X) = \sum_i A_{w,i} X^i, \tag{2.22}$$

where i is the Hamming weight of the codewords generated by input sequences of Hamming weight w. An alternative definition where parity check sequences are used is also available and is given below,

$$A(w, Z) = A_w(Z) = \sum_z A_{w,z} Z^z \tag{2.23}$$

where z is the Hamming weight of parity sequences produced by input words of Hamming weight w.

2.5.5 Computation of Spectrum Functions

Computation of spectrum functions is a difficult job especially for large input frame lengths. Spectrum functions of some low rate convolutional codes are computed in [21] where a forward-backward algorithm is employed in computation. In [22] also a Viterbi like method is proposed. To find the spectrum functions of the codes two approaches can be pursued. In the first approach, a computer program can be written for producing all input sequences and generating all the codewords, finding their Hamming weights and number of codewords for a specific Hamming weight of input sequences. This method is not efficient for codes with large sizes. The second approach is the polynomial approach we propose. Polynomial approach is introduced by an example in Appendix A.

2.6 Decoding

Channel codes are decoded using either algebraic or trellis based decoding methods. Algebraic techniques have some reduced complexity advantages, however their performances are worse than the trellis based decoding algorithms. Sequential decoding algorithms are the first practical algorithms to provide fast but sub-optimal decoding for convolutional codes. Trellis based algorithms show optimal performance. The most well known trellis based decoding algorithms are Viterbi, and marginal a posteriori (MAP). Hard and soft decision techniques can be applied to Viterbi algorithm. Soft output Viterbi algorithm (SOVA) is developed by integrating the soft decision approach to Viterbi algorithm. The classical Viterbi algorithm estimates data sequences, i.e., it minimizes sequence error rate. However, MAP algorithm minimizes bit error rate. Although the MAP algorithm shows better performance results compared to Viterbi algorithm, it is approximately twice more complex than the Viterbi decoding [19]. The MAP algorithm is recently modified and log-MAP and max-log-MAP algorithms were introduced [23]. The log-MAP algorithm is used for our simulations.

2.7 Performance Bounds

Trellis terminated convolutional codes can be considered as block codes. The analytical bound expressions developed for block codes are also valid for convolutional codes in this case. There are two criteria for code performance. These are bit error rate (BER) and frame error rate (FER) of a code. A frame is a sequence of a number of bits. The bit error probability of a (n, k) linear block code observed through an additive white Gaussian noise (AWGN) channel satisfies the bound

$$P_b \leq \sum_{d=d_{min}}^{n} \frac{\tilde{w}_d N_d}{k} Q(\sqrt{\frac{2dRE_b}{N_0}}), \tag{2.24}$$

where N_d is the number of code words of weight d, $\tilde{w}_d$ is the average weight of the N_d messages that produce weight d codewords, $N_0/2$ is the double sided noise power spectral density, E_b is the energy per bit, d_{min} is the minimum distance of the code, R is the code rate, n is the number of codewords and $Q(x)$ is an exponentially decreasing function defined by

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-y^2}{2}} dy. \tag{2.25}$$

At high *SNR* values, the bit error probability is approximated by the first term of summation in eqn. (2.24) and it is given by

$$P_b \approx \frac{\tilde{w}_{d_{min}} N_{dmin}}{k} Q(\sqrt{\frac{d_{min} 2rE_b}{N_0}}). \tag{2.26}$$

One of the most important criteria in code design is to maximize the minimum distance of the code and try to minimize the number of codes with the minimum Hamming weight.

2.8 Punctured Convolutional Codes

Puncturing which involves the elimination of parity bits is used to increase the rate of codes. However, heavily punctured codes suffer from performance loss. There is a tradeoff among the puncturing degree, E_b/N_0, and the performance of the code. Puncturing operation usually decreases the free distance of convolutional codes. Puncturing process is illustrated by the help of a matrix. For instance, for a rate 1/2 convolutional code, if

$$P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.27}$$

is employed as a puncturing matrix, we delete every other parity bit from the encoded bit stream. The meaning of the first column is that the first data and parity pair is kept, the second column indicates that for the next data and parity pair, data bit is kept however parity bit is eliminated. For instance, the code whose generator matrix $G = (1, 5/7)_{octal}$ has free distance $d_{free} = 5$, i.e., an input sequence '0111' produces minimum Hamming weight codeword

’00111011’. The puncturing matrix in eqn. (2.27) can also be expressed as a vector [1110] by concatenation of the column of P in eqn. (2.27).

When the puncturing matrix is applied to this code, its free distance decreases to $d'_{free} = 3$, i.e., deleting every second parity bit in a periodic manner the bit sequence ’001x101x’ is obtained from minimum Hamming weight codeword. The trellis diagram of punctured convolutional encoder $(1, 5/7)_{octal}$ happens to be as in Fig. 2.10.

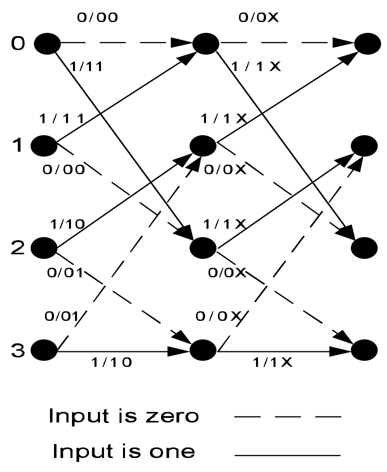


Figure 2.10: Punctured convolutional encoder trellis diagram.

If the puncturing matrix

$$P = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.28}$$

is employed for a rate 1/2 convolutional code, the code rate to increases to 4/5. An equivalent representation for the puncturing matrix in eqn. (2.28) is shown by the puncturing pattern [11101010].

2.9 Some Properties of Convolutional Codes

Convolutional codes can be linear or non-linear. Mostly, linear convolutional codes are used in practice. The number of states in a convolutional encoder state machine depends on the number of memory cells in shift registers. The trellis structure can be time varying or time invariant. In both cases, the trellis structure is a much more regular structure when compared to block code trellis structures. Convolutional codes are widely used in communication systems. Convolutional codes have found applications in voice band applications. Convolutional codes are also employed in deep space communication. An overview of the recent applications of convolutional codes in space communication is summarized in [9].

2.9.1 Hard Decision vs Soft Decision Decoding

Viterbi based algorithms can use both hard decision and soft decision methods. Hard decision is explained as follows. Assuming the use of $BPSK$ modulation during communication, the binary values 0 and 1 are represented by -1 and 1. When the modulated signals are passed through an AWGN channel, the received signals are never exactly -1 or 1 due to noise. The received signals can be quantized to the nearest constellation point and appropriate binary values can be assigned. This quantization converts the AWGN channel to the binary symmetric channel. Quantization causes some information loss. The binary sequence is lastly fed to the Viterbi decoder. The overall decoding operation is called hard decision Viterbi decoding, since before the Viterbi decoding operation the receiver makes a binary (Hard) decision about the received signals.

In soft decision Viterbi decoding operation, quantization unit is omitted. Actual received signal values are directly fed to the Viterbi decoder. Received signal values are called soft values, since no quantization and demodulation operations were performed, i.e., no hard decision is made. This is illustrated [24] in Fig. 2.11. Soft Viterbi decoding operations are the same as the hard Viterbi decoding procedure with the difference that the former one uses soft values for the branch metrics and path metrics use the squared Euclidian distance rather than the Hamming distance. Soft decision Viterbi decoding is illustrated in Fig. 2.11. Soft Viterbi decoding can be implemented using digital signal processors and *FPGAs*, and no noticeable performance degradation is observed compared to computer simulations when fixed point implementation is done even with a small number of bits.

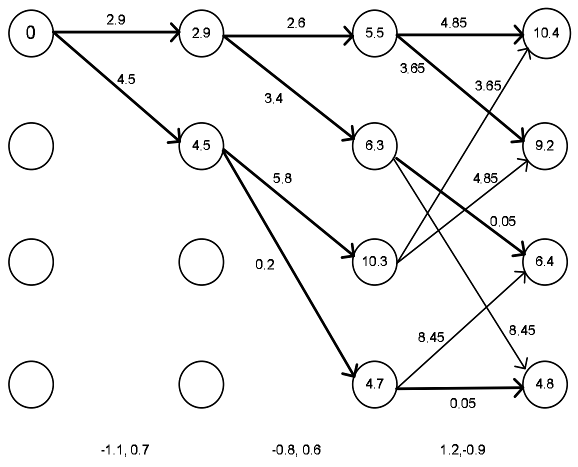


Figure 2.11: Soft decision Viterbi decoding illustration.

In order to obtain performance close to Shannon bounds, soft channel outputs are necessary. Hard Viterbi decoding apparently performs $2dB$ worse than soft Viterbi decoding.

2.9.2 Free Distance

Assuming all zero sequence is transmitted, an error event is defined as a departure from the all zero path followed by remerging to the all zero path. Error events start at state zero and return to state zero after some time, no return to the zero state in between occurs. A decoding error occurs if the received signal is closer to any other sequence other than the all zero sequence. Referring to Fig. 2.6, it is seen that there is a path whose received sequence is closer to the error sequence 111000 than the all zero sequence. Hamming distance between all zero path and the error sequence is 5. Since this is the smallest Hamming distance for error events, it is also called free Hamming distance and denoted by d_{free}. If the convolutional code trellis diagram is not zero state terminated then Hamming free distance has no meaning. The performance of a convolutional code is closely related to its free distance. Linear block codes are designed for specific distance properties. However, good convolutional codes are searched for a given number of memory elements. Free distance is the criteria for finding digital convolutional encoder circuits.

CHAPTER 3

CONCATENATED CODES

In this chapter we study well known parallel and serially concatenated convolutional codes. The reasons behind the good performance of concatenated convolutional codes are explained. Different type of interleavers are discussed. The design criteria behind the selection of constituent codes for good concatenated convolutional codes are clarified. We also explain soft-input-soft output iterative processing module, and show that using soft-in-soft-out modules any concatenated structure can be iteratively decoded.

3.1 Introduction

To achieve Shannon limits, construction of codes with large block lengths is essential. In fact, construction of large block length codes is not a difficult task. Randomly generated codes with large block sizes will achieve good performance with high probability. However, the decoding complexity for randomly generated codes increases exponentially with the codeword size. Hence, randomly generated codes with large block sizes quickly become unaffordable. To reduce the decoding complexity and to obtain codes with large block sizes code concatenation technique has for long been considered by researchers.

Elias introduced product codes [25] in which shorter block codes were combined to obtain a good larger block length code. Following the work of Elias concatenated codes were introduced in [12]. The main reason behind the code concatenation is to obtain high performance and low decoding complexity when compared to a single code which achieves the same performance. Low decoding complexity is achieved by decoding constituent codes separately. Constituent encoders can be concatenated in serial or parallel manner or a combination of both known as the hybrid concatenation. The constituent codes in a concatenated code structure can be chosen from block or convolutional codes.

3.2 Turbo Codes

Turbo codes which were introduced in 1993 by Berrou et. al. [13] represents a breakthrough in coding theory. A turbo code encoder is formed by two constituent systematic recursive convolutional encoders and an interleaver in between. The input information sequence is fed directly to the first encoder and it is fed to the second encoder after being interleaved. The exchange of extrinsic information between decoders occurs in a sequential manner. This extrinsic information exchange continues sufficiently many times. Usually an iteration number between 8 and 12 is sufficient. The performance of turbo codes is quite close to Shannon limits. This aroused a huge interest in coding society. Since the introduction of turbo codes, there has been an enormous amount of research in many aspects of encoding as well as decoding.

The astonishing performance of turbo codes were investigated in [26], and it was found that the role of the interleaver was one of the most critical factors for the performance of turbo codes. An analytical bound expression for the performance of turbo codes using uniform interleaver is given in [26]. Although the uniform interleaver approach determines bounds on the code performance, the bounds are not so tight.

Turbo codes were also selected for many telecommunication systems standards such as the UMTS standard proposed by the third generation partnership project (3GPP)[27]. Due to this popular interest, many iterative decoding algorithms to reduce decoding complexity have been proposed [28].

The encoder and decoder structure of a turbo code are depicted in Figs. 3.1 and 3.2. Recursive systematic convolutional codes are employed as constituent

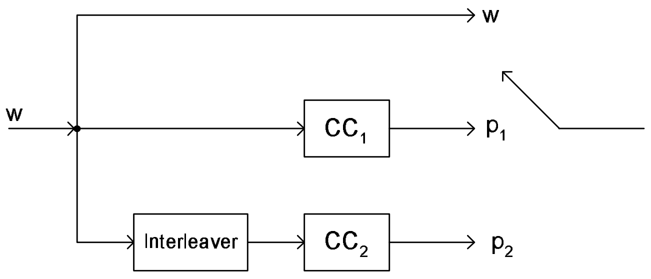


Figure 3.1: Parallel concatenated convolutional code. CC_1, CC_2 are constituent recursive systematic convolutional codes *(RSCs)* 1 and 2, respectively.

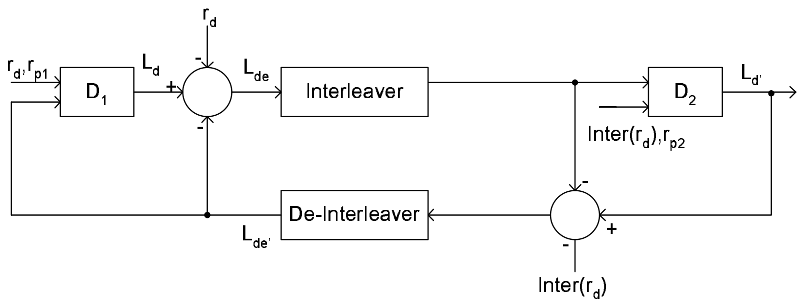


Figure 3.2: Turbo decoder structure. $D1$, $D2$ are constituent decoders 1 and 2 respectively. r_d, r_{p1}, r_{p2} are received signals. $L_d, L_{de}, L_{d'}, L_{de'}$ are log-likelihoods.

codes. The decoder modules use soft decision decoding algorithms.

3.3 Interleaving

Interleaving is the process of re-ordering a data sequence in a one-to-one correspondence. The inverse of this process is called de-interleaving. Interleaver is a device that changes the order of symbols in a sequence. Interleaving enhances the error correcting capability of the codes in certain situations. Interleaving operation is especially critical for channels with bursty error characteristics. Multipath communication channel is such a channel. Another such channel is the magnetic recoding channel where an interleaver can be placed between encoder and communication channel, at the receiver the received data is de-interleaved first and thus the errors are spread over time. This converts bursty errors to random errors. The errors within a codeword become independent to a decoder. Interleaving operation is also used in concatenated codes. The use of interleaver in turbo codes reduce number of coefficients of low weight codewords which results in a reduced bit error probability by a factor of 1/(interleaver length) and this reduction is called interleaving gain [23]. The basic roles of the interleaver in a turbo code can be outlined as below.

1. The basic duty of an interleaver is to combine two shorter codes of inferior performance into one longer code with superior performance.

2. Two decoders' input data are virtually de-correlated by interleaving such that, after corrections in the first decoder, some of the uncorrectable errors from the first decoder can be spread by the interleaver such that they can be corrected by the second decoder.

3. Another role of the interleaver is to break the low weight input sequences and increase the free distance of the codewords or reduce the number of low weight codewords.

Turbo like codes are potential candidates for implementation in communication systems. There are already some implementations of turbo-like codes [29]. During the implementation of concatenated codes, some problems arise. The commonly encountered problems are the inefficient use of memory, memory collision, and high hardware complexity. Concatenated codes have high decoding latency. To reduce the decoding latency parallelization at the decoder side is employed. If the interleaver is not designed in specific manner then it is highly probable that memory collision problem occurs during the parallel processing operation. Hence, many high performance interleavers can be of no use, if parallel decoding operation is performed at the receiver side. This issue led to the birth of an important research field which is the memory collision free interleaver design for parallel decoding operations [30]. Although parallelization reduces the decoding delay enormously it enhances hardware complexity. Hence, there is a tradeoff among code performance, reduced latency, and hardware complexity.

3.3.1 The effect of Interleaving on Code Performance

Interleaver design is usually not of analytical nature. Very often an interleaver is employed in an iterative decoding system, then the behavior of the interleaver is explained through simulation results. Each simulation result demonstrates the performance of the interleaver for a specific encoder and decoder pair, hence a general interleaver design method is not easily agreed upon. The role of the interleaver on the code performance will be discussed here considering the turbo code. A typical turbo code BER performance graph mainly consists of three regions, steady region, waterfall region, and error floor region. Turbo codes perform well at low SNR values close to Shannon limits. The performance of the turbo code at high SNR values is mainly determined by the structure of the interleaver. This region is called the error floor region. Good interleavers reduce the multiplicities of low weight code words and possibly increase minimum distance, and thus enchance code performance at high SNR values. The design of the good interleavers that result in a small number of low weight codewords has been an active research area over the past decade.

3.4 Interleaver Types

There are many interleavers proposed in the literature. An interleaver can be designed for a specific constituent code. For instance, for the constituent codes with the generator matrix $(1, 5/7)_{octal}$ the weight two input patterns 1001 and 100001 produce codewords with the smallest Hamming weights. Hence, a good interleaver designed for $(1, 5/7)_{octal}$ should break the low Hamming weight input patterns. However, some of the interleavers can be used for any constituent code and show good performance. We will here give a brief information about the most widely known interleavers. Interleavers can be divided into two main categories. These are block type interleavers, random interleavers, and code matched interleavers. Some of the block-type interleavers are block interleavers, multiplex interleavers, convolutional interleavers, shuffle interleavers, co-prime interleavers, Welch-Costas interleavers, Berrou-Glavieux interleavers, JPL interleavers, and Takeshito-Costello interleavers [31]. Interleavers may be given some attributes like causality, delay amount, memory, spreading factor, dispersion factor etc. [31]. Random interleavers are those ones which involve a pseudo-random operation during their construction. The use of random interleavers play a fundamental role in iterative decoding schemes. The uniform interleaver is actually not a real interleaver. It is used to estimate the performance of the concatenated codes considering all the available interleavers. Considering all the available interleavers there is at least one interleaver with the same performance as that of the uniform interleaver.

3.4.1 Block Interleaver

In a block interleaver, the incoming information sequence is written into a matrix row-wise and it is read out from the matrix in column-wise. Block interleavers are easy to implement. However, they may fail to break some low weight input sequence patterns. As a result of this, the performance of turbo codes with rectangular interleavers is not good enough. Block interleavers are efficient if error patters to be broken reside in a single row or column. For block interleavers the interleaving technique should be modified if the error patters are confined to several consecutive rows or columns. Block interleavers are also called rectangular interleavers. Odd-even block interleavers and helical interleavers are two different block interleavers. In the former one, even positions are mapped to odd positions and vice versa whereas data bits are read diagonally and written row-wise in helical interleavers. Helical interleavers prevent consecutive bits being output from the same column or row. These interleavers are a little bit better than the regular block interleaver.

3.4.2 Convolutional Interlavers, Cyclic Shift Interleavers

Convolutional interleavers were introduced by Forney [23]. Data is written into a matrix in a convolutional interleaver. Different row elements are read out with different time delays. The same row elements are read out with the same time delay. Convolutional interleavers have less end-to-end delays compared to block interleavers. In the cyclic shift interleaver, data is written into a matrix, each row is cyclicly shifted by different amount, and the cyclically row shifted matrix elements are read out in column-wise. Convolutional interleaving and de-interleaving operation is depicted in Fig. 3.3.

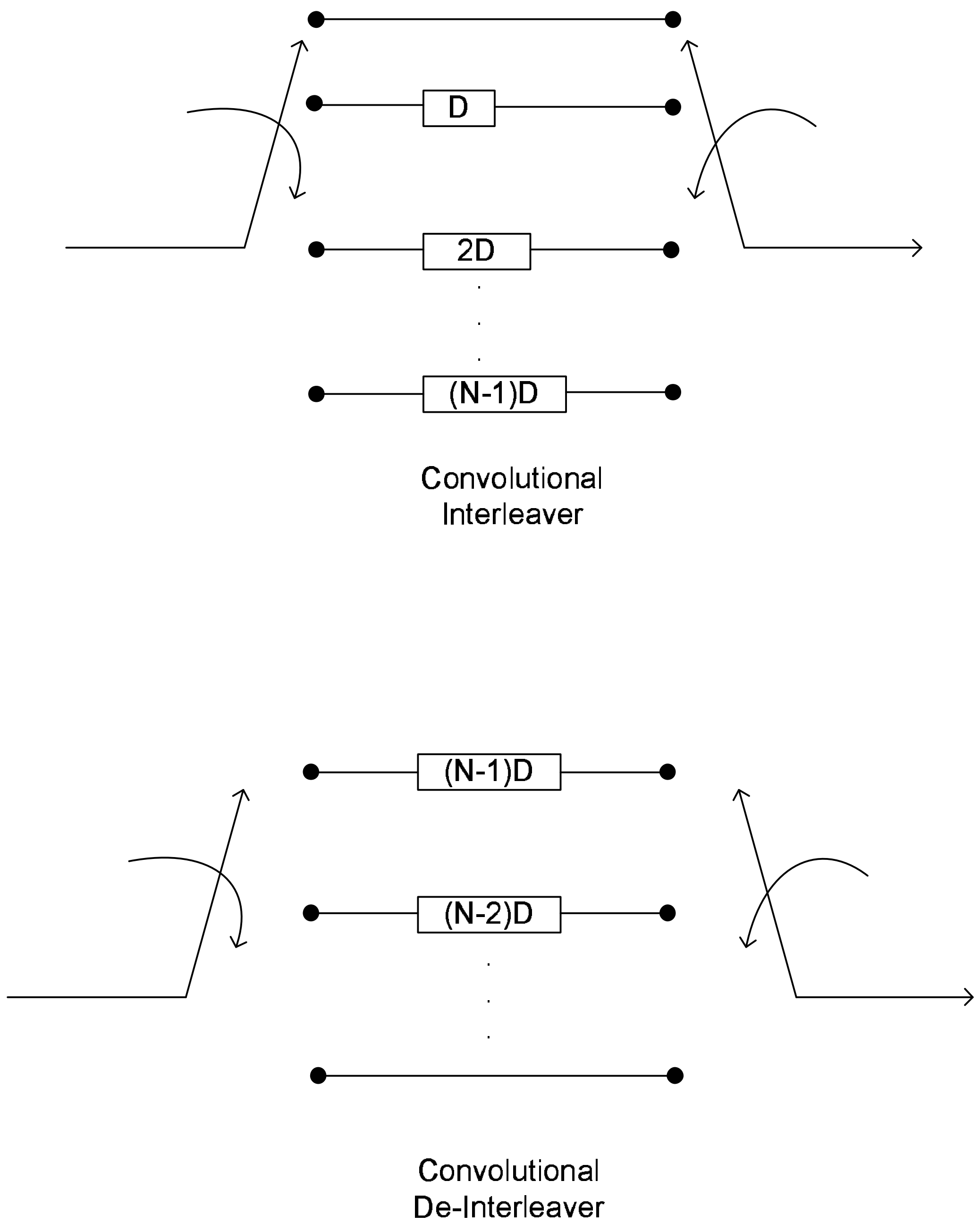


Figure 3.3: Convolutional interleaver and de-Interleaver.

3.4.3 Random Interleavers

3.4.3.1 Random Interleaver

A random interleaver is obtained by generating a pseudo-random sequence of length L. The sequence elements are selected from the set $S = \{1, 2, \cdots, L - 1, L\}$ and each element appears once in the random sequence. The size of the interleaver is L. The elements of the pseudo-random sequence are chosen with equal probability. A data sequence is interleaved by reading the input bits according to the generated pseudo-random sequence.

3.4.3.2 *S-random* Interleaver

S-random interleavers or spread interleavers are constructed by generating random numbers from 1 to L based on an S-constraint where S is the minimum interleaving distance.

The operation of the *S-random* interleaver is as follows. A randomly selected integer is compared to the previously S_1 selected integers. If the absolute differences between the selected integer and any of the S_1 previously selected integers are greater or equal to S_2 then the randomly selected integer accepted otherwise it is rejected. An *S-random* interleaver is determined based on

$$\mid I_i - I_j \mid \leq S_1 \quad j = i + 1, \cdots, S_1 + i \tag{3.1}$$

$$\mid P(I_i) - P(I_j) \mid \geq S_2 \quad j = i + 1, \cdots, S_1 + i \tag{3.2}$$

where I_k denotes the original index and $P(I_k)$ is the permuted index in the interleaved sequence. When identical constituent codes are used, it is appropriate to choose $S_1 = S_2 = S$, where $S \leq L/2$.

An *S-random* interleaver employed in a turbo code system can break input patterns with lengths up to $S + 1$ and produce high weight parity check sequences. This is achieved as explained in the following sentence. Assume that there exists an input pattern sequence of length up to $S + 1$. The upper constituent encoder produces a low weight parity sequence. The information sequence is broken by an *S-random* interleaver and the lower constituent encoder produces a high weight parity sequence in this case. *S-random* interleavers are widely employed in concatenated codes.

3.4.4 Code-Matched Interleavers

A code-matched interleaver is a special type of pseudo random interleaver designed for certain constituent codes. The major aim of the code-matched interleaver is to break the low weigth input sequences in such a manner that the first several spectral lines of the code are totaly eliminated. Due to the specific design concept, the performance of the code-matched interleaver is better than the *S-random* interleaver. However, its design is more cumbersome than the *S-random* interleaver and not general for all constituent codes.

3.5 Uniform Interleaver

The uniform interleaver is also called the average interleaver. The uniform interleaver is defined as a probabilistic device that maps a given input information sequence of length L and Hamming weight w into all distinct $\binom{L}{w}$ permutations of it with equal probability $1/\binom{L}{w}$. The uniform interleaver permits the estimation of the average interleaver gain, independent of the particular interleaver used in a turbo encoder. Overall performance of a turbo code is determined using the uniform interleaver. The performance estimation using the uniform interleaver concept is also applicable to serially concatenated convolutional codes and to many other concatenated code structures. Although the uniform interleaver approach provides an estimate about the system performance, it should be noted that it is usually a rough estimate.

3.6 Analytical Performance of Turbo Codes

An upper bound for the bit error probability P_b of turbo codes is computed by

$$P_b \leq \sum_{w=1}^{L} \sum_{d=d_{min}}^{\infty} wA(w,d)\frac{1}{L}Q(\sqrt{2dR\frac{E_b}{N_0}}), \tag{3.3}$$

where L is the size of the interleaver, w is the Hamming weight of the input information sequence which generates codewords of Hamming weight d. The code rate is denoted by R. $A(w,d)$ is the number of codewords of Hamming weight d generated by input sequences of Hamming weight w. The term $A(w,d)$ is often named as the distance spectrum of the code.

3.7 Serially Concatenated Convolutional Codes

The encoder and decoder structure of serial concatenated codes are depicted in Fig. 3.4. In encoding operation, datawords are encoded using an outer convolutional code after which the outer codewords are passed through an interleaver. The interleaved bit stream is fed to the inner convolutional encoder and outer codewords are generated and transmitted. The behavior of the decoder is a little bit different here than a turbo decoder. The outputs of the outer decoders are used as coded bit probabilities of the inner decoder and these coded bit probabilities are updated and given to the outer decoder for data bit probabilities. This iteration is repeated sufficiently many times.

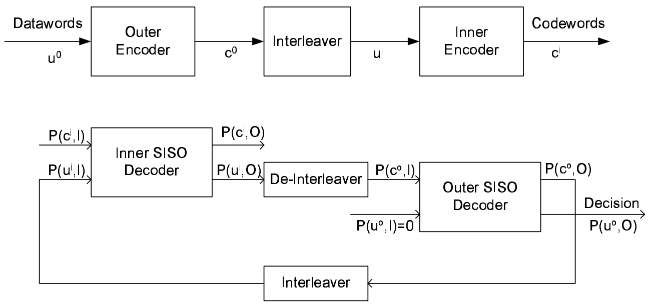


Figure 3.4: Serially concatenated encoder and decoder structure.

3.7.1 *SCCC* Design

The design of a *SCCC* involves the selection of inner and outer convolutional codes according to some criteria. The inner encoder should be a recursive convolutional encoder. The use of a recursive inner encoder provides an interleaver gain [32]. The improvement in bit error probability due to the use of interleaver is defined as the interleaver gain. The interleaver gain for *SCCC* equals $L^{-d^o_{free}/2}$ for even values of d^o_{free} and it is $L^{-(d^o_{free}+1)/2}$ for odd values of the free distance d^o_{free}. As a consequence of this statement, we should choose an outer code with a large value of odd free distance. Another design criteria is that the number of datawords generating codewords with Hamming weight d^o_{free} should be minimum and also Hamming weights of these datawords should be minimized. Since non-recursive codes have $w_{free} = 1$, they can be convenient to choose for outer convolutional codes. Inner code effective free distance ($d^i_{f,eff}$) which is the minimum weight codewords generated by weight 2 input data words should be maximized. The most critical difference between *PCCCs* and *SCCCs* is the great difference in the interleaver gain. For an interleaver of length L, *PCCCs* show an interleaver gain of L^{-1} whereas *SCCCs* provide an interleaver gain of $L^{-(d^o_{free}+1)/2}$ assuming that d^o_{free} is an odd number. PCCCs have better performance than *SCCCs* at low *SNR* values, *SCCCs* on the other hand have better performance at high *SNR* values. When error floor is taken into consideration, *PCCCs* reach their error floor region before the *SCCCs* where the intersection point is usually around error rates of practical interest.

3.8 Turbo Code Design

The first encoder of turbo code is usually trellis terminated. This eliminates the availability weight-2 codewords. The second encoder termination is not crucial and does not degrade the code performance too much [33]. The error floor of the turbo code is a consequence of the low minimum distance of the code. The error floor level can be lowered by increasing the size of the interleaver without changing the minimum distance of the code. Choosing constituent codes from *RSCs* in a turbo code system usually result in an increased minimum distance. The interleaver in a turbo code system reduces the number of low weight codewords. This is called spectral thinning [33].

The asymptotic performance of the turbo code is computed using (3.4) where L_{min}/L is the effective multiplicity. L_{min} is the number of minimum distance codewords,

$$P_b \approx \frac{L_{min}\tilde{w}_{min}}{L} Q\left(\sqrt{d_{min}\frac{2RE_b}{N_0}}\right). \tag{3.4}$$

The asymptotic performance of the turbo code is mainly determined by the minimum distance of the code. The error floor phenomena in turbo codes is due to the fact they have small minimum distance and as a result of this a relatively flat minimum distance asymptote appears. The role of the interleaver on the error floor region can be explained in two ways. Increasing the interleaver size and keeping minimum distance the same result in a lower effective multiplicity and this reduces the error floor level, without changing the slope of the asymptote. In this case the error floor is seen at higher *SNR* and lower BER values. Conversely, reducing interleaver size and keeping minimum distance and its multiplicity the same, causes error floor to rise and it is reached at lower *SNRs* and higher BERs. If the size of the interleaver is fixed, then the error floor slope can be changed by changing the minimum distance and keeping its multiplicity the same. The rectangular interleaver is one of those interleavers known for a long time. Rectangular interleaver results in large multiplicities of d_{min}. Increasing the rectangular interleaver size does not result in a considerable reduction in effective multiplicity of the minimum distance. Hence, the error floor of the turbo code is not lowered when rectangular interleaver is employed. Unlike the rectangular interleaver, the use of a random interleaver in a turbo code system affects the distance spectrum of the code. The use of random interleaver and *RSCs* cause a turbo code to have a sparse distance spectrum.

In [33] average turbo code is defined. The distance spectrum of an average turbo code is computed over all pseudo random interleavers of length L. Minimum distance codewords of an average turbo code are generated from weight-2 input words assuming that a large interleaver size is used. To maximize the minimum distance of the average turbo code, the constituent codes are chosen such that they give largest output weight codewords for weight-2 information words. The interleaving gain of a turbo code is found to be $L^{1-w_{min}}$ where L is the interlaver size and w_{min} is the minimum Hamming weight of the data-words which generates codewords with d_{min}. For recursive convolutional codes $w_{min} = 2$, hence when recursive constituent codes are used in a turbo code system, the interleaving gain becomes L^{-1}. For nonrecursive convolutional codes $w_{min} = 1$. Thus, turbo code system does not benefit from the interleaving gain when nonrecursive convolutional codes are employed as constituent codes.

3.9 Iterative Decoding and Soft-Input Soft Output (SISO) Module

MAP algorithm is used to decode trellis codes. As an abstraction in concatenated code architectures, the single input single output (SISO) modules are utilized. Concatenated codes are iteratively decoded by using the SISO modules for each of the constituent codes [34]. Component decoders are the SISO modules which exchange information and make use of the received information from other modules.

In Fig. 3.5 trellis encoder, a section of the trellis and SISO module is depicted. A single section of the trellis completely illustrates the behavior of the encoder. For the trellis section in Fig. 3.5, a transition occurs from state S_t to S_{t+1} along with the information bit u and code bit c. $P(u)$ and $P(c)$ are the probabilities for data and coded bits. The Soft-Input Soft-Output (SISO) module is a four-port device that takes input sequence, channel probabilities $P(u, I)$, $P(c, I)$, and produces the updated input sequence and channel probabilities $P(u, O)$, $P(c, O)$. The updated probabilities are used as input probabilities by other SISO modules. The SISO algorithm needs the entire sequence to be received before decoding starts. The use of SISO modules makes life easy for the decoding of any concatenated code. In Fig. 3.6 the encoder

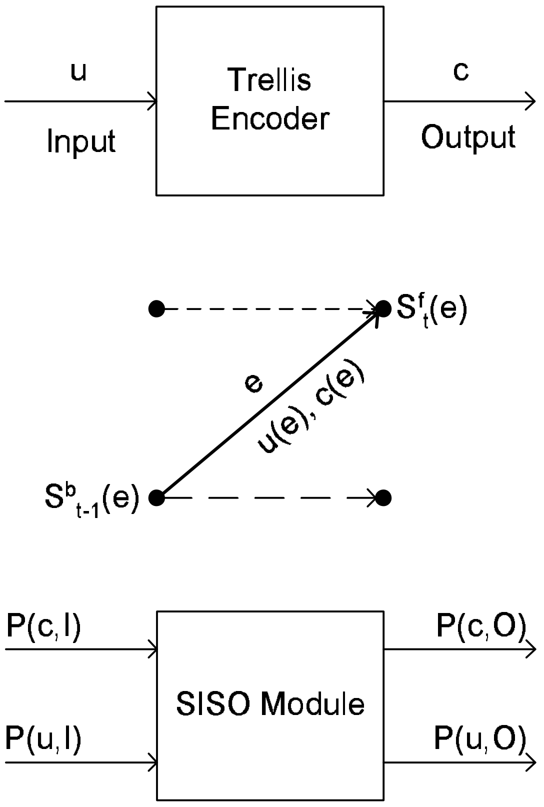


Figure 3.5: SISO module.

decoder structures for a hybrid code are depicted.

3.10 Major Drawbacks of Concatenated Codes

High speed turbo-like codes can offer great promises for reliable communication in real time applications such as wide band multimedia services. Although selected for many standards, turbo-like codes have a major drawback which is their significant amount of decoding latency due to iterative decoding. Especially real time applications such as voice and video communications require low latency. It is essential to find solutions for high latency since it is obvious that future communication systems will need higher throughput.

Latency reduction can be achieved via two approaches. One method is to decrease the complexity of the decoding algorithms i.e., to decrease the computation amount needed for decoding processes, the other is to use multiple processors for higher decoding speeds or use concurrent computation methods

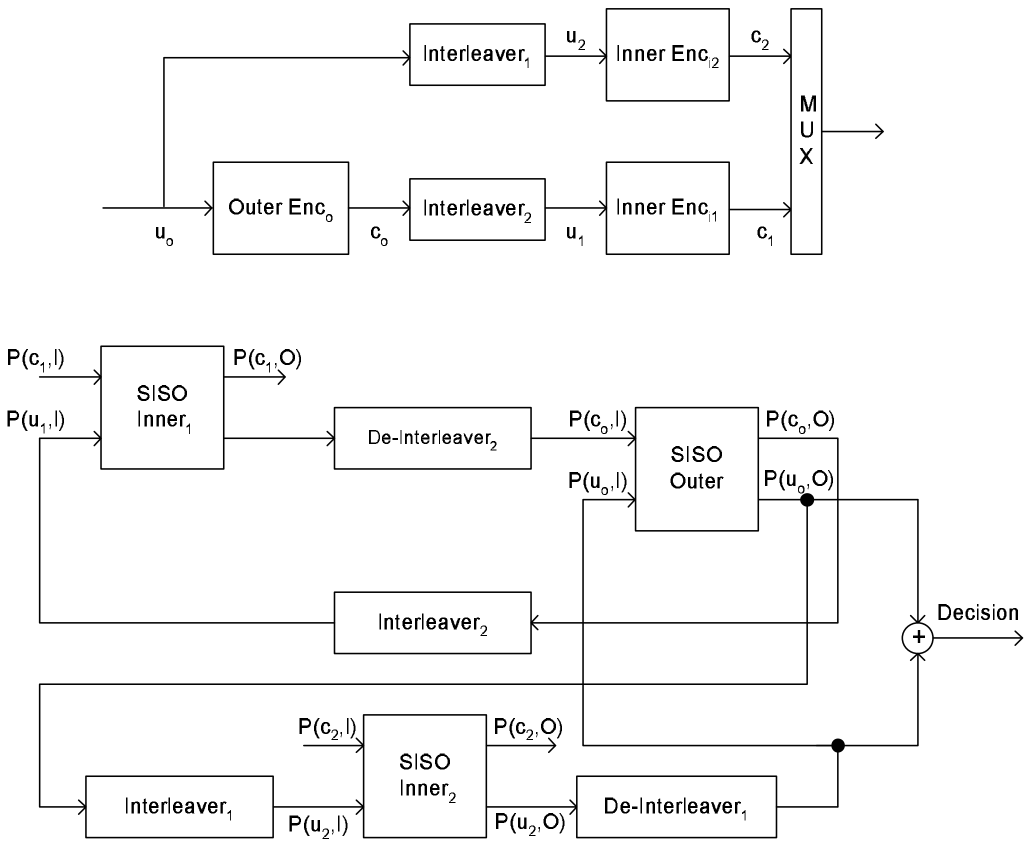


Figure 3.6: Hybrid encoder and decoder.

to calculate the parameters of the decoding algorithm. We first explain the studies made for the complexity reduction then clarify the method of using multiple processors.

There are two well known approaches to reduce the complexity of the MAP algorithm. These are *T-BCJR* and *M-BCJR* algorithms [35]. *M-BCJR* algorithm reduces the complexity of the algorithms using only M most likely states. *T-BCJR* algorithm uses a threshold to determine the states to be eliminated. When state probabilities fall below a threshold they are discarded. The *M-BCJR* uses a subset of M states while *T-BCJR* uses variable number of states at different trellis stages. Hence, *T-BCJR* shows a dynamic behavior whereas *M-BCJR* has a static behavior. The *T-BCJR* algorithm shows better performance when compared to the *M-BCJR* algorithm.

The number of iterations is also related to the computation amount hence to decoding latency. Low latency can be achieved by reducing the number of iterations which is achieved by defining good stopping criteria beyond which negligible performance improvement is observed. In [36, 37] a number of stopping criteria have been suggested to terminate the decoding procedure early when there is no more performance improvement observed.

Another work to reduce the decoding latency of turbo codes were done in [29] where radix-4, center-to-top algorithms are introduced. Center-to-top algorithms involve the concurrent computation of the forward and backward state probabilities. Early-stop algorithm for reduction of iteration numbers is proposed. An *FPGA* implementation of the proposed algorithms is also presented.

For the use of multiple processors in decoder systems, two kinds of structures have been reported in literature. These are the pipelining [38] and block partitioning methods [39]. For pipelining method, let W denote the number of processors and K be the iteration number, in this method each processor processes the entire block for K/W iterations and passes the extrinsic information to the next processors. Hardware complexity of pipelining method is greater than of block partitioning methods, i.e., need larger register space compared to block partitioning. In the block partitioning method, the received frame is divided into sub-frames, then sub-frames are processed by parallel processors. In [40] instead of overlapping frames the forward and backward variables computed in the previous stage are used as boundary distributions, hence eliminating the need for extra memory use for boundary probabilities and overlapping. In [41] two component decoders are run concurrently by which latency can be reduced approximately by half. This scheme corresponds to a parallelization in iterations.

CHAPTER 4

PARALLEL DECODABLE SERIALLY CONCATENATED CONVOLUTIONAL CODES

In this chapter we introduce novel code families called the convolutional product codes and, later generalize the concept and propose parallel decodable serially concatenated codes. The proposed structure here is a generalization of all the serially concatenated codes and many other variants can also be derived. An original interleaver structure called row-column *S-random* interleaver which prevents memory collision problem in parallel processing operations is also proposed.

4.1 Introduction

The aim of the concatenated codes is to built powerful error correcting codes with reasonable constituent decoder complexities. Concatenated codes are proposed by Forney [12] in 1966. Concatenated codes are similar to product codes. However, in the previous case an interleaver is usually employed between encoders. Forney showed that the use of a posteriori probabilities *(APPs)* as opposed to hard decision estimates was essential for enhancing decoder performance. Following Forney's work, new *APP* algorithms were developed. The *BCJR* [42] which is ignored for a long time is now one of the most widely known *APP* algorithms.

With the introduction of turbo codes (parallel concatenated convolutional codes, i.e., *PCCCs*) in 1993 a huge interest on iterative decoding aroused. Turbo codes can achieve bit error rate (BER) levels around 10^{-5} at code rates quite close to the corresponding capacity with reasonable decoding complexity. The use of soft-in soft-out decoding algorithms was a key in this success. This showed the way to the invention of serially concatenated convolutional codes *SCCCs* in [32], where the authors showed that they were better than *PCCCs* in some respects.

Block codes show good performance around rates 0.5. For lower rates or much higher rates their performance is not as good as rate 0.5 performance. High rate block product codes are studied in [43]. Although these codes also have large decoding complexity, they are very suitable for parallel decoding operations. Product codes studied so far have been constructed using linear block codes, such as Hamming, extended Hamming [44, 45], BCH [43, 46], and Reed Solomon [47] codes. Single parity check (SPC) product codes are studied in [48]. Three and more dimensional SPC product codes are studied in [49]. Product codes have also attracted practical attention lately. *DSP* and *FPGA* implementations are studied in [50].

Although block codes have time varying trellises, convolutional codes' trellis structures are usually time invariant. Moreover, the number of states in the trellis structure of an (n, k) block code is upper bounded by and usually on the order of $2^{(n-k)}$ where k is the information sequence length and n is the codeword length in bits [22]. However, the number of states in a convolutional code can be set as desired. The time-invariant trellis structure of convolutional codes makes them more convenient for implementation [32]. In addition, numerous practical techniques such as trellis coded modulation and puncturing can be simply utilized with convolutional codes as opposed to linear block codes. Convolutional codes are also integrated with space time trellis codes which is a variant of convolutional codes adapted for multi antenna transmission systems. Since convolutional codes are more suitable for practical implementations we will in this chapter propose convolutional product codes (*CPCs*) which are constructed using convolutional codes only. The use of convolutional codes in a product code setting lays the ground for utilizing all the flexibility and vast knowledge base for convolutional codes in parallel decoders. This type of product code has component codes with a time invariant trellis structure, as opposed to product codes constructed with linear block codes (Hamming, BCH, Reed Solomon etc.). Hence, *CPC* may be more favorable for implementation than linear block product codes. *CPC* has a matrix structure, and this makes it attractive for integration with multi-carrier communication. For example, the vertical dimension can be used for different sub-carriers.

We further extend the idea of *CPCs* and propose parallel decodable serially concatenated convolutional codes *(PDSCCCs)* which have reduced decoding delays compared to serially concatenated convolutional codes *(SCCCs)* and at the same time show comparable performance. The newly proposed codes have parallel processing property which results in significantly less decoding delays compared to *SCCCs*. *CPCs* constitute a subclass of *PDSCCCs*. The idea is expanded and parallel decodable concatenated codes *(PDSCCs)* are introduced. This is the most general structure for all the serially concatenated codes. It is easily observed that *PDSCCCs*, *SCCCs*, and block products codes are nothing but instances of *PDSCCs*. In the following sections, the encoding and decoding structures for the parallel structures will be explained and a novel interleaver design method to avoid some problems met in practical implementations will be presented.

4.2 *CPC* Encoder and Decoder (Code Structure)

4.2.1 *CPC* Encoder

A regular product code is constructed by placing the information bits/symbols into a matrix. The rows and columns are encoded separately using linear block codes [44, 46]. This type of a product encoder is shown in Fig. 4.1. It is seen from the figure that the data and parity bits are grouped separately.

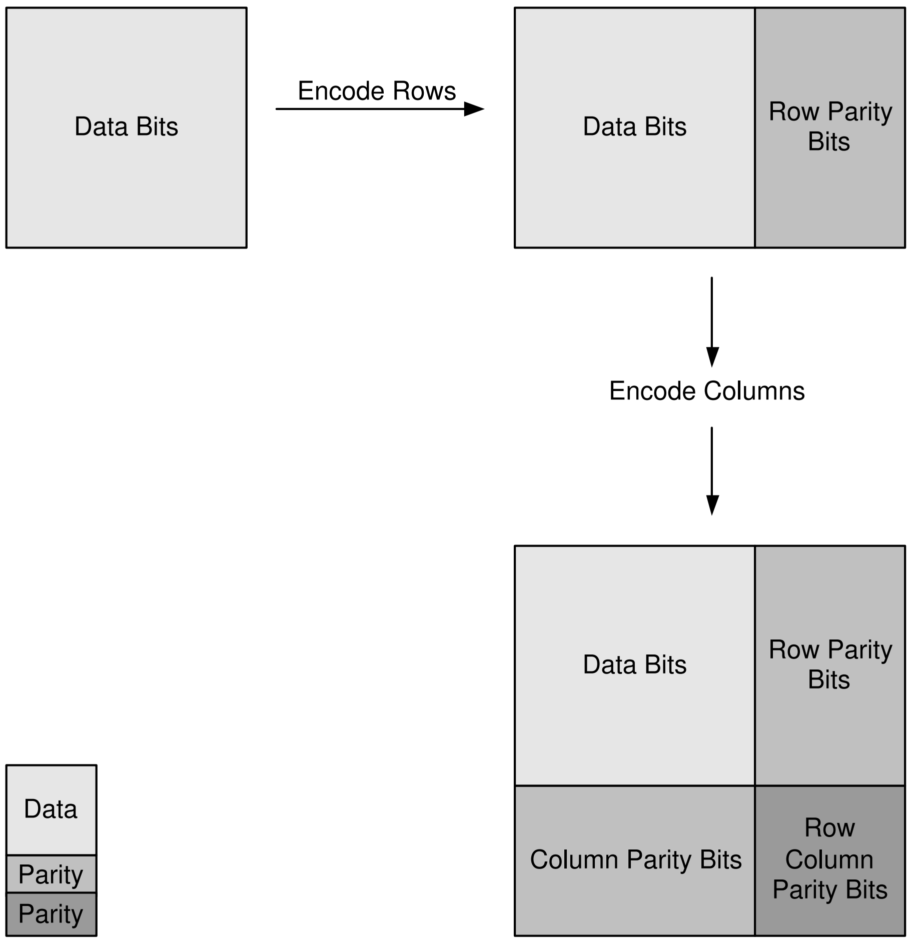


Figure 4.1: Regular product code encoding procedure, where a block code is used to encode rows and columns.

In our case we use convolutional codes instead of linear block codes to encode rows and columns. This is illustrated in Fig. 4.2. When compared to Fig. 4.1, it is obvious that data and parity bits are mixed uniformly.

Encoding is performed by using a matrix which determines how each encoder works. The data to be sent is put into a matrix. Each row of the matrix is encoded using a convolutional code. We use the same recursive systematic convolutional code (RSC) to encode each row, although different convolutional codes can be used for this purpose. Once each row is encoded, the matrix is sent, if desired, to an interleaver. Our data matrix dimension is $k \times k$ and the encoded data matrix dimension is $n \times n$, i.e., our code is an $(n^2,\ k^2)$ code. The interleaved matrix is coded column-wise. In our simulation we used a rate $1/2$

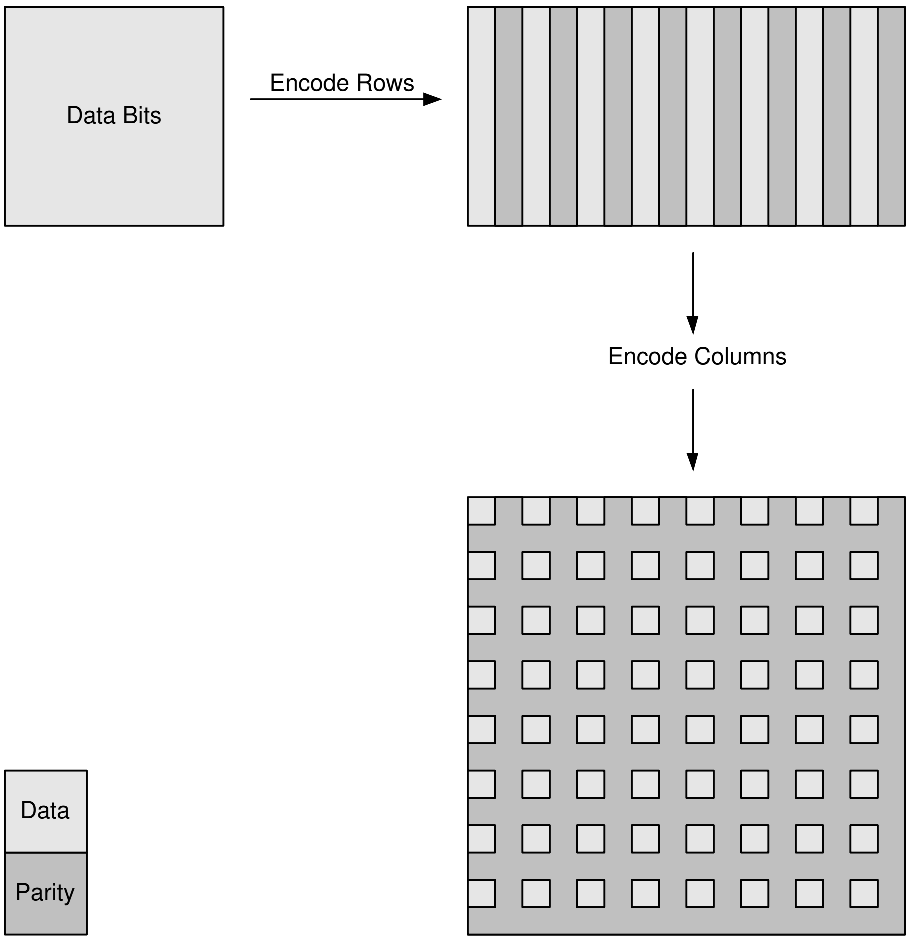


Figure 4.2: *CPC* encoding procedure without an interleaver.

recursive systematic convolutional code with the matrix generator $(1, 5/7)_{octal}$ to encode each row and column. Hence, the overall code rate is 1/4. The general encoding procedure, which includes any type of interleaver, is illustrated in Fig. 4.3.

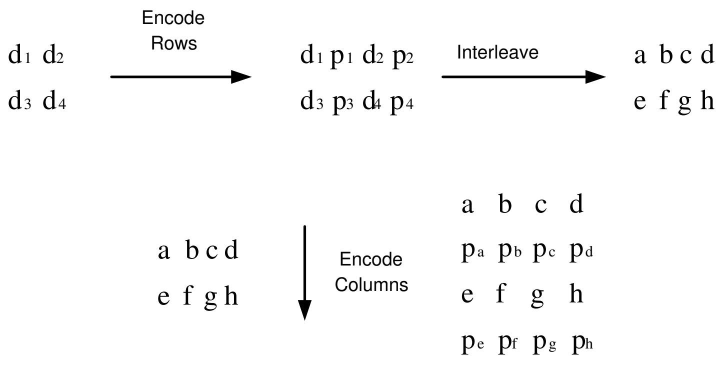


Figure 4.3: Convolutional product code encoder with any type of interleaver (d denotes data bits and p denotes parity bits).

4.2.2 *CPC* Decoder

Convolutional product coded data is multiplexed to a single stream and binary phase shift key (BPSK) modulated. The BPSK modulated signal is passed through an additive white Gaussian noise (AWGN) channel with double-sided noise power spectral density $\frac{N_0}{2}$, i.e., noise variance is $\sigma^2 = \frac{N_0}{2}$. We used the log-MAP soft decoding algorithm [32] to iteratively decode the convolutional product code. Each column is independently decoded one by one since columns were encoded last. The extrinsic information obtained from the columns is passed to the row decoder after being de-interleaved. Then row decoding proceeds; rows are decoded one by one and interleaved extrinsic information is passed to the column decoder. The *CPC* decoding procedure is depicted in Fig. 4.4. This procedure is repeated for a sufficient number of times. The decoding structure employed in this study is the same as that of serially concatenated codes in Fig. 4.5 [32]. For frames of equal length, an *SCCC* decoder uses two log-MAP decoders and performs quite well at low rates. *CPC* decoders can utilize many log-MAP decoders in parallel, thus showing smaller decoding delays. Therefore, we will compare the proposed *CPC* structure to that of *SCCC*.

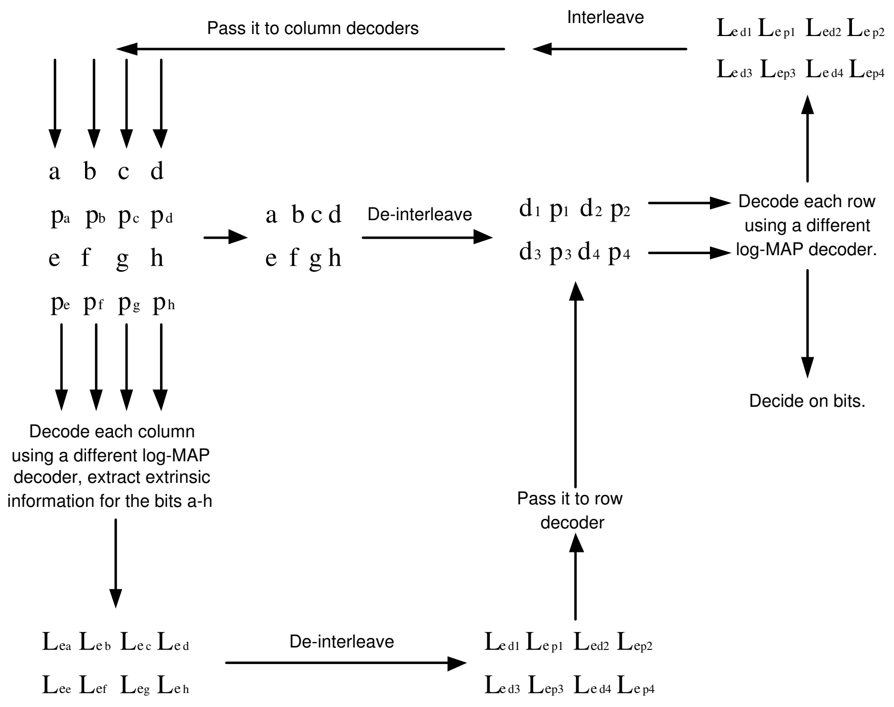

Pass it to column decoders
Interleave
L_{ed1} L_{ep1} L_{ed2} L_{ep2}
L_{ed3} L_{ep3} L_{ed4} L_{ep4}
a b c d
p_a p_b p_c p_d
e f g h
p_e p_f p_g p_h
a b c d
e f g h
De-interleave
d_1 p_1 d_2 p_2
d_3 p_3 d_4 p_4
Decode each row using a different log-MAP decoder.
Decide on bits.
Decode each column using a different log-MAP decoder, extract extrinsic information for the bits a-h
Pass it to row decoder
L_{ea} L_{eb} L_{ec} L_{ed}
L_{ee} L_{ef} L_{eg} L_{eh}
De-interleave
L_{ed1} L_{ep1} L_{ed2} L_{ep2}
L_{ed3} L_{ep3} L_{ed4} L_{ep4}

Figure 4.4: Decoding operation of the convolutional product code.

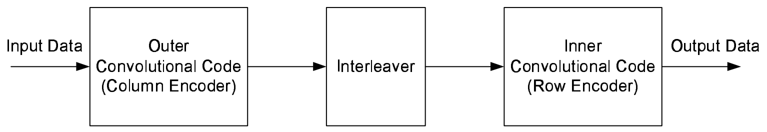

Input Data
Outer Convolutional Code (Column Encoder)
Interleaver
Inner Convolutional Code (Row Encoder)
Output Data

Figure 4.5: *SCCC* encoding operation.

4.3 Parallel Decodable Serially Concatenated Codes *(PDSCCs)*

CPC encoding operation involves the use of a matrix. Instead of using matrix notation for convolutional product encoding a serially concatenated system can be employed to realize the same code. To realize the equivalent structure two code clusters are concatenated in series and an interleaver is employed between code clusters. A code cluster is constructed by concatenating constituent codes in parallel. The number of constituent codes in outer and inner code clusters is variable. The number of constituent codes in outer cluster is denoted by N and the number of constituent codes in inner cluster is denoted by M. The constituent codes can be chosen from both block and convolutional codes. The general structure of the *PDSCCs* is depicted in Fig. 4.6 where $\mathrm{C}_{11}, \mathrm{C}_{12}, \ldots, \mathrm{C}_{1\mathrm{N}}$ and $\mathrm{C}_{21}, \mathrm{C}_{22}, \ldots, \mathrm{C}_{2\mathrm{M}}$ are the outer and inner constituent code (CC) encoders respectively. An interleaver is placed between the inner and outer encoder clusters. Serial and parallel converters are employed in *PDSCCs* structure. Serial to parallel conversion can be achieved by row-wise or column-wise writing of a matrix with data elements. Parallel to serial conversion corresponds to row-wise or column-wise reading of matrix elements. If only block codes are employed in Fig. 4.6, with determined N and M values according to the type of block codes employed, a regular product code as in Fig. 4.1 is obtained.

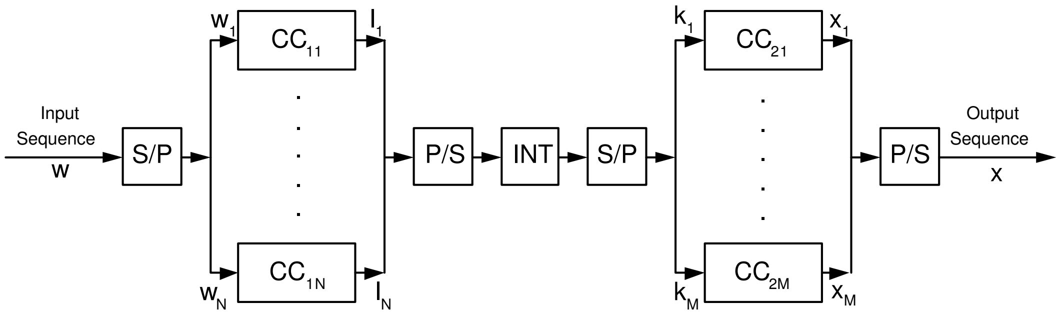


Figure 4.6: *PDSCC* structure.

If only convolutional codes are employed for constituent codes in *PDSCCs*, parallel decodable serially concatenated convolutional codes *(PDSCCCs)* are obtained. When M=N=1, *PDSCCCs* turn out to be *SCCCs*, i.e., *SCCCs* are a special case of *PDSCCCs.* Assuming that the rate of all the constituent encoders equals 1/2, for an input sequence of length (N.M)/2 where $\mathrm{N} \geq 2,\ \mathrm{M} = 2 \times \mathrm{k}$, k is a positive integer greater than 1, the overall encoding procedure can be demonstrated using a matrix notation. This special case corresponds to previously mentioned *CPCs.* In *CPCs* M value is chosen as 2N so that input matrix becomes a square matrix. Hence it can be said that *CPC* is an instance of general *PDSCCCs* structure. The length of the information frames processed by constituent codes are shorter than the original frame length. Due to this reason trellis termination criteria becomes more important for *PDSCCCs* especially for large number of parallel branches.

Addition of trellis termination bits decreases code rate. When all the constituent codes are trellis terminated the code rate for an information frame of length L can be computed as

$$R = \frac{L}{\frac{L}{R_1 R_2} + \frac{T_1 N}{R_1 R_2} + \frac{T_2 M}{R_2}}, \tag{4.1}$$

where R_i and T_i are the rates and number of memory elements respectively for constituent codes in clusters. The effect of trellis termination bits on code rate for different number of rate 1/2 and memory-2 outer and inner constituent codes for information frame length of 1024 are tabulated in table 4.1 where it is assumed that all the constituent encoders add trellis termination bits. As it is clear from table 4.1 that code rate decreases slightly even for large number of parallel branches.

Table 4.1: *PDSCCC* rates for different number of parallel branches in outer and inner clusters.

N \ M	1	4	8	16	32	64
1	0.249	0.248	0.247	0.245	0.242	0.234
4	0.247	0.247	0.246	0.244	0.240	0.233
8	0.245	0.245	0.244	0.242	0.238	0.231
16	0.242	0.241	0.240	0.238	0.235	0.228
32	0.235	0.234	0.233	0.231	0.228	0.222
64	0.222	0.221	0.220	0.219	0.216	0.210

It may be noted that the structure in Fig. 4.6 is very general. One may use different codes of even different rates and number of memory elements as constituent codes. Although this is the case we will consider the rate 1/2 memory-2 *RSC* with $(1, 5/7)_{octal}$ without loss of generality.

Interleaver use can be varied in *PDSCCS*. Many different use of the interleaver are possible, a single interleaver can be used between clusters, separate interleavers can be employed in front of the inner constituent codes, different interleavers can be used before the outer constituent codes, or we can use a different interleaver for each of the constituent codes in outer and inner clusters. The use of interleavers for inner constituent codes corresponds to the use of interleavers for columns of the row encoded matrix in *CPC* case. We will talk about the column *S-random* interleaved *CPC* in the following sections. The use of separate interleavers for all the constituent codes leads to the discovery of memory collision free row-column *S-random* interleaver. The memory collision problem and row-column *S-random* interleaver which prevents the memory collision problem will be introduced in the following sections.

4.3.1 *PDSCC* Decoder

The decoder of a *PDSCC* is seen in Fig. 4.7. It is clear from the decoder structure that decoders $D_{11} \ldots D_{1N}$ and decoders $D_{21} \ldots D_{2M}$ can be run in parallel. The decoders may employ any soft decision algorithms. The decoders working in parallel process shorter frame lengths and they need less clock cycles to process the entire frame.

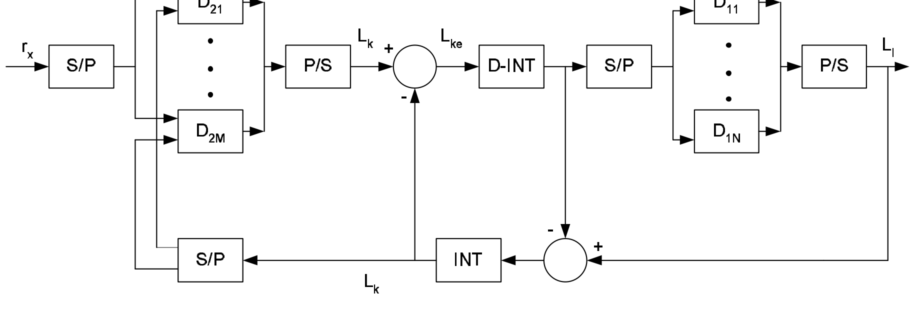


Figure 4.7: *PDSCC* decoder. r_x is the received signal value, S/P and P/S are the serial to parallel and parallel to serial converters respectively. *INT* is the interleaver. L_k and L_l are the bit probabilities.

4.4 Practical Implementation Issues

4.4.1 Memory Collision Problem

Although parallelization of the concatenated codes reduces the latency, it creates new problems. Memory collision and extra memory need for the storage of the boundary distributions are such two problems. The memory collision is explained as follows. The output of the decoders and the received data should be stored according to the permutation order of the interleaver. Classical serial concatenated convolutional has two constituent decoders. However, parallel decodable serially concatenated convolutional code has two constituent decoder clusters each of which contains many decoders. All the decoders in one of the clusters run in parallel and try to access the memory locations where the extrinsic information generated by the other cluster's decoders is stored. During this access two or more decoders in the same cluster may try to use the same memory segment at the same clock instant, and this is an impossible event. Hence, memory collision occurs which is due to the permutation order of the interleaver. An interleaver without memory collision is shown in Fig. 4.8. An interleaver which causes memory collision is depicted in Fig. 4.9.

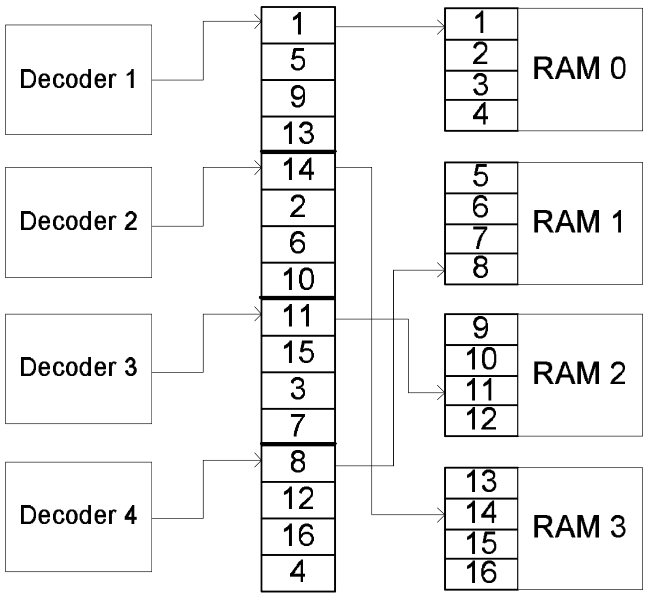


Figure 4.8: An Interleaver without memory collision.

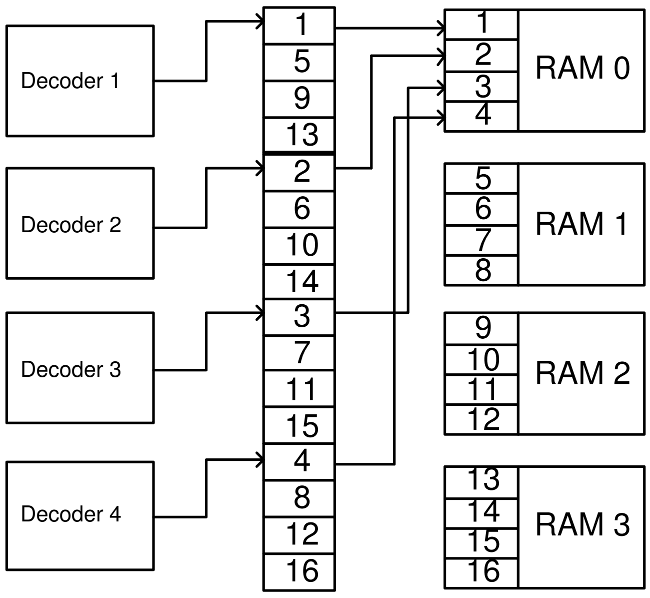


Figure 4.9: An Interleaver causing memory collision.

To avoid the memory collision problem in parallel decoding operation, the number of memory blocks where interleaved data is stored should be greater than or equal to the number of decoders in constituent decoder clusters and furthermore each decoder in a cluster should try to access different memory segments at each clock. This can be achieved using specifically designed interleavers. We propose novel row-column *S-random* interleaver in the following section and illustrate the avoidance of memory collision with row-column *S-random* interleaver.

4.4.2 Row-Column *S-random* (*RCS-random*) Interleavers

In this section we propose row-column *S-random* interleaver to prevent the memory collision problem and to achieve similar performance to that of the classical *S-random* interleaver. This type of interleaver can be interpreted as a joint structure of the rectangular and random interleaver. This structure both benefits from the randomness and the advantages of the matrix structure. *RCS-random* interleaving is applicable to any type of code, i.e., it can be used for serially concatenated convolutional codes or parallel concatenated convolutional codes. The operation of the *RCS-random* interleaver is as fol-

lows. The data sequence is put into a matrix. First each row of the matrix is interleaved by distinct interleavers then each column of the matrix is interleaved by different interleavers. Finally the interleaved matrix elements are encoded row-wise. Hence, an equal number of constituent codes are utilized, i.e., N=M. This sequence of operations guarantees the prevention of memory collision if the variables for decoding of each row is kept in a separate memory block. There would be a collision if two rows have bits at the same column whose variables are stored in the same memory block. But this is impossible since they could only go to different column by the first row interleaving operation. This type of encoding is illustrated in Fig. 4.10. The number of memory blocks equal the number of rows of the data matrix. If the number of memory blocks is less than the number of rows of the data matrix memory collision cannot be avoided. Since in this case, two decoder varbles shares the same memory block which causes memory collision.
To clarify the concept more, we give an example illustrating the use of row-column *S-random* interleaver for collision free parallel decoding operations. The overall procedure is illustrated in Fig. 4.11. For simplicity a data matrix of size 2×2 is considered. The number of parallel encoders in outer and inner clusters were chosen as 2. First rows are encoded by *RSCs*, each row of the matrix is separately interleaved, then each column of the matrix is separately interleaved, finally rows are encoded again by *RSCs*. Different column elements of the interleaved matrix resides in different memory locations and this prevents memory collision issue. This is clearly seen in Fig. 4.11.

In *CPC* encoding operation, the encoding is performed column-wise after *RCS-random* interleaving. This does not prevent memory collision as can be observed from the fact that no interleaving is a special case of *RCS-random* interleaving and it will cause memory collision. Minimum distance of *PDSCCCs* and *CPCs* with *RCS-random* interleavers will be investigated in the following sections. It should be noted that the *RCS-random* interleaver introduce in this section is its plainest form with equal number of encoders in each cluster. Generalizations for this scheme are possible as well.

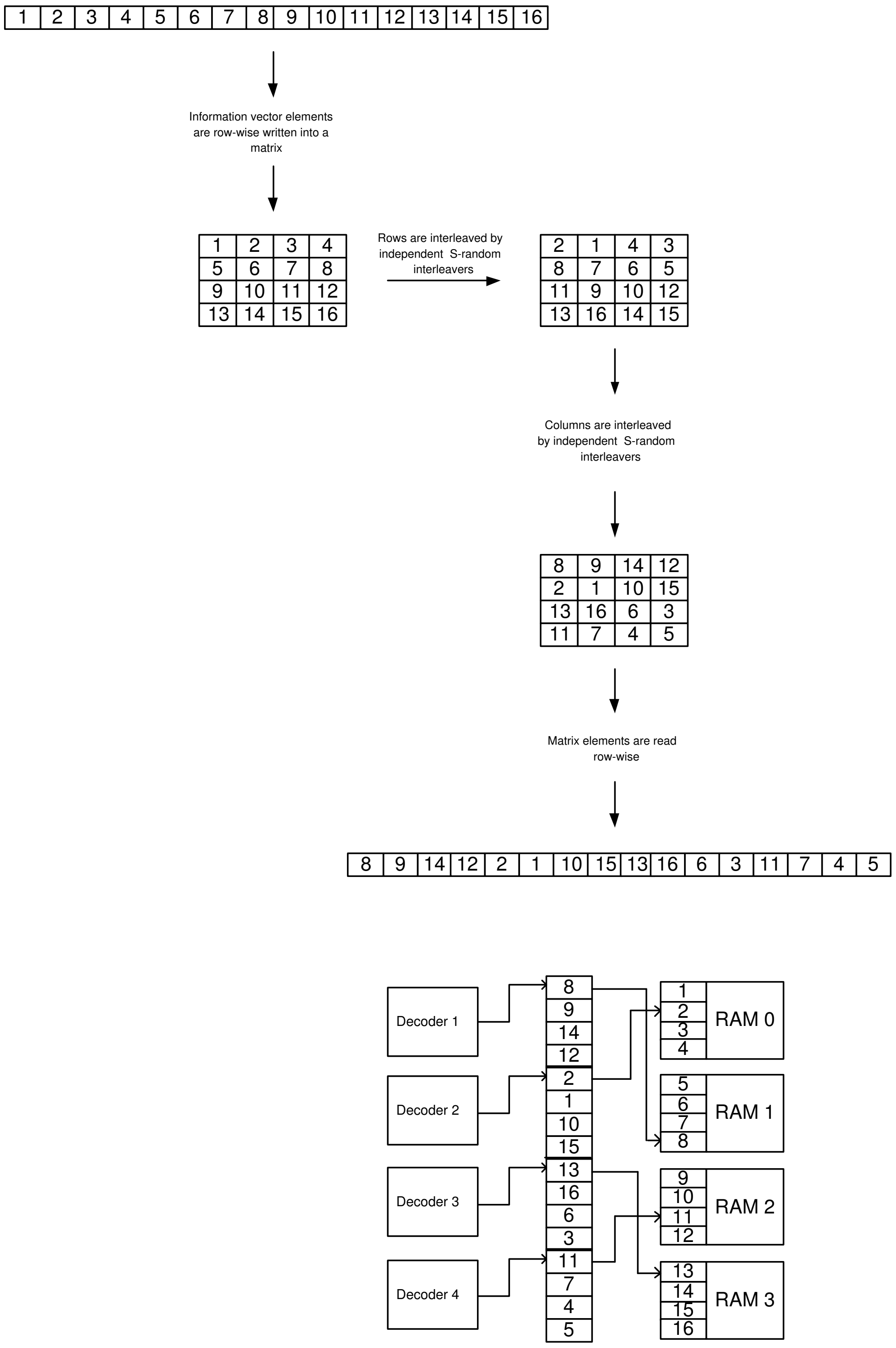


Figure 4.10: Row column *S-random* interleaver without memory collision.

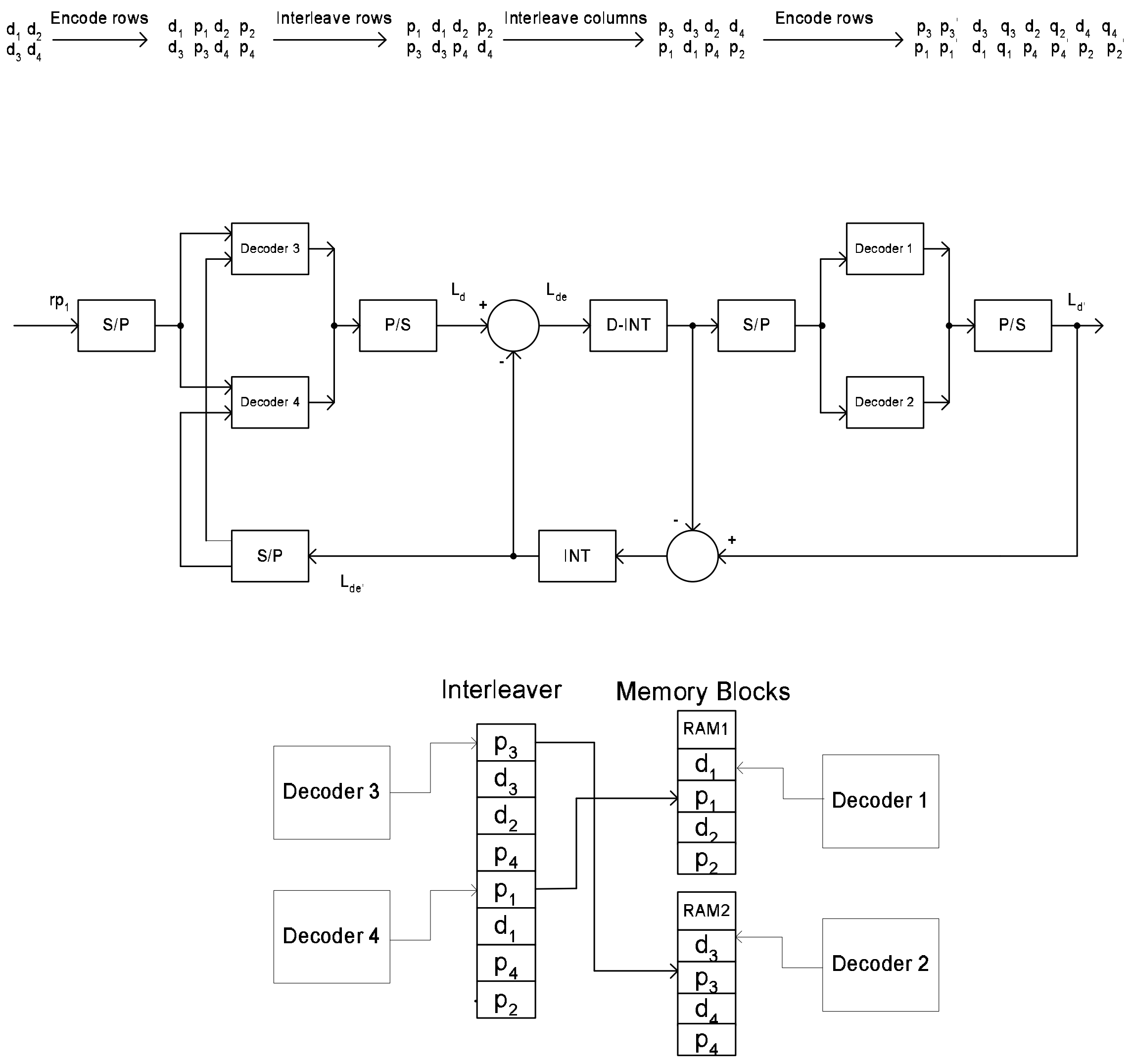


Figure 4.11: *PDSCCC* with collision free row-column *S-random* interleaver.

4.5 *CPC* Minimum Distance and Its Asymptotic Performance

The Hamming weight of a binary codeword is defined as the number of '1's available in the codeword [51]. The minimum distance of a linear code is the minimum Hamming weight of all the codewords. The minimum distance plays an important role in code performance. As it gets larger, code performance becomes better, especially at high *SNR* values [51]. We assume that d_{free} is the free distance of the component convolutional codes used in *CPCs* with trellis termination. We will investigate the minimum distance of *CPCs* according to the type of the interleavers.

4.5.1 *CPC* with Rectangular Interleaver

After the first stage of the *CPC* encoding operation (row encoding), it is obvious that one of the rows of the row-encoded matrix should contain at least d_{free} number of '1's. This means that there are d_{free} columns containing at least a single '1' in row-encoded matrix. When columns are encoded, there exist at least d_{free} number of columns each containing at least d_{free} '1's. Hence, in total there are at least d_{free}^2 '1's in the coded matrix. This is the minimum distance of the *CPC* whose component convolutional codes have trellis termination constraint. In Fig. 4.12 this concept is explained for $(1, 5/7)_{octal}$ component convolutional codes whose free distance is 5. In summary, if rectangular interleaver is used the *CPC* minimum distance is d_{free}^2.

4.5.2 *CPC* with Column *S-random* Interleaver

Both to preserve the d_{free}^2 minimum distance of the *CPC*, and to get benefit from the interleaving gain, after row encoding one can use *S-random* interleavers for each column, i.e., each column is interleaved but different column elements are not mixed. In this way it is guaranteed that d_{free} number of columns contain a single '1' before column encoding operation. We call this

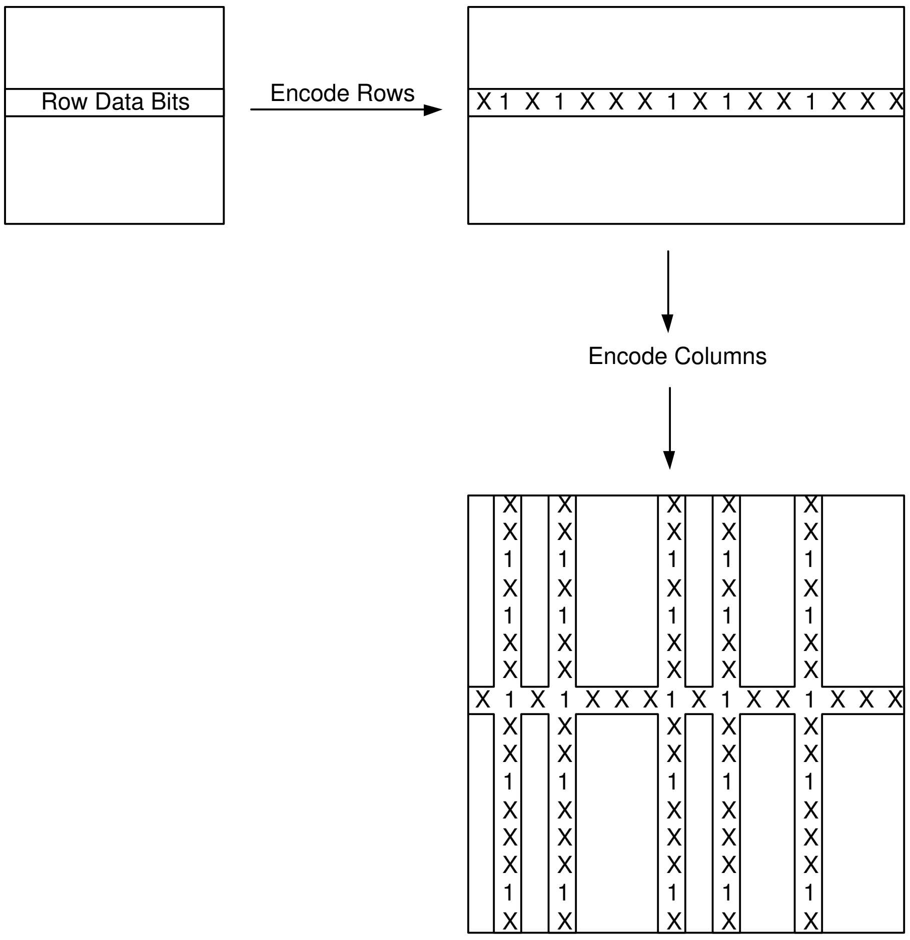


Figure 4.12: If the columns elements are not mixed d^2_{free} is preserved.

type of interleaving column *S-random* interleaving to distinguish it from regular *S-random* interleaving. A helical interleaver [52] also does not mix the different column elements. A helical interleaver and a combination of helical and column *S-random* interleavers will also be considered.

4.5.3 *CPC* with Full *S-random* Interleaver

If a single *S-random* interleaver is used for all the elements of the matrix after row encoding, the number of columns that contain a single '1' is not necessarily equal to d_{free}. This leads to the fact that *CPC* minimum distance is not necessarily equal to d_{free}^2 any more. In fact, after interleaving operation all the '1's may appear in a single column. This means that *CPC* minimum distance is lower bounded by d_{free}. We call this type of interleaving full *S-random* interleaving. In Fig. 4.13 the effect of the full *S-random* interleaver is illustrated. It is seen from the Fig. 4.13 that when the row encoded matrix is *S-random* interleaved all the '1's appearing in a row may go to a single column. This verifies that *CPC* minimum distance is lower bounded by d_{free}. Although the use of *S-random* interleaver guarantees a smaller minimum distance compared to rectangular interleaver, its performance is much better than rectangular interleaver.

4.5.4 *CPC* with *RCS-random* Interleaver

In this interleaving method, minimum distance d_{free}^2 is preserved for *CPC* and maximum interleaving gain is achieved. After row encoding operation, each row of the row-encoded matrix is *S-random* interleaved with a different *S-random* interleaver. In this way, the number of '1's in each row stays the same, i.e., one of the rows contains at least d_{free} number of '1's. Then each column of the row-encoded matrix is *S-random* interleaved with a different *S-random* interleaver. Different column elements are not mixed. Hence, there are at least d_{free} number of columns each containing at least a single '1'. Then second stage of the encoding operation proceeds. Columns are encoded

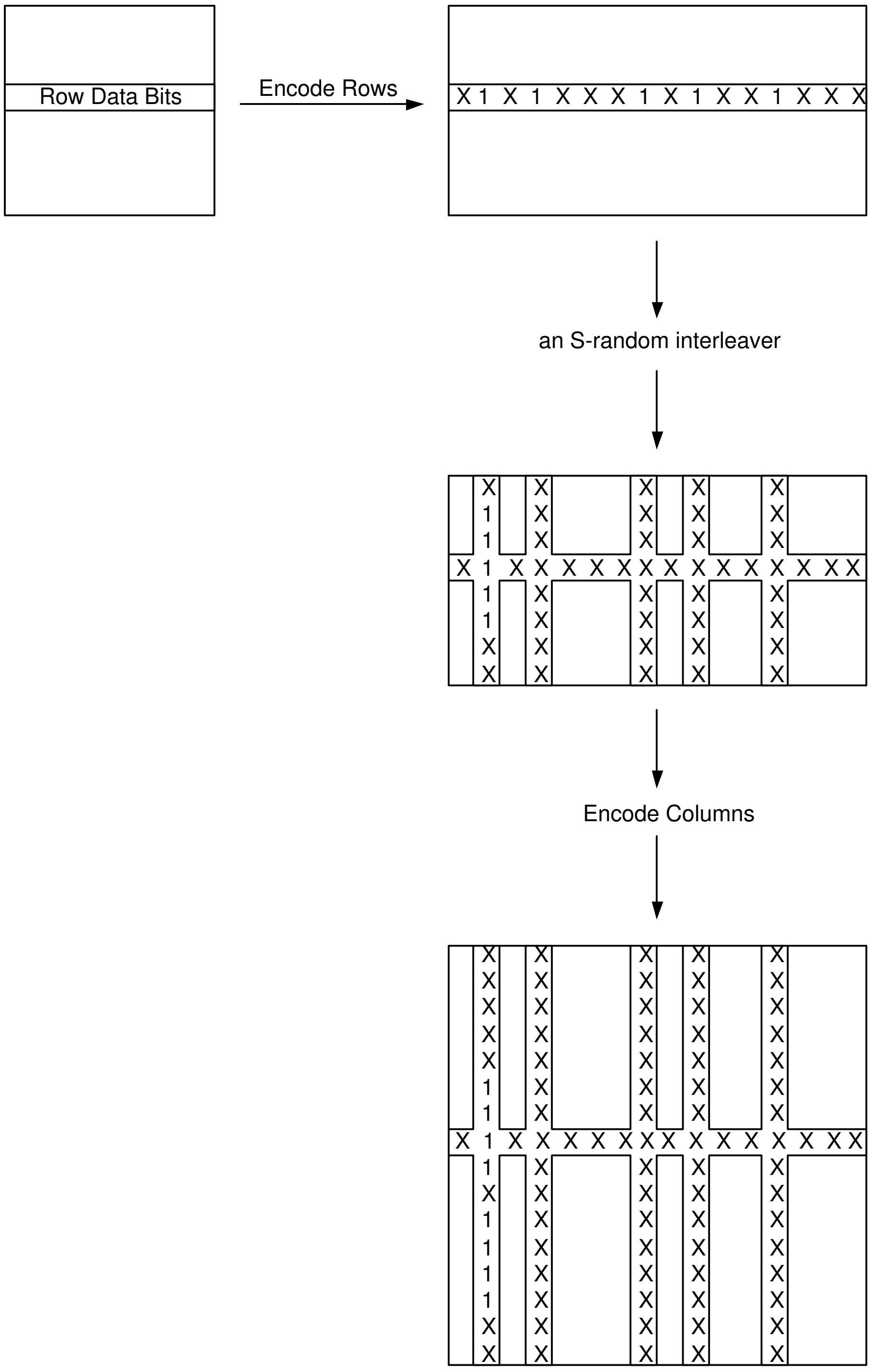


Figure 4.13: d^2_{free} is not preserved if *S-random* interleaver is used ('x' stands for a single or a group of 0's).

separately using a *RSC*. After column encoding operation, there exists at least d_{free} number of columns each containing at least d_{free} number of '1's. Hence, the minimum distance of the code becomes d_{free}^2. The encoding procedure using row-column *S-random* interleaver is depicted in Fig. 4.14.

4.5.5 *PDSCCC* with *RCS-random* Interleaver

The use of row-column *S-random* interleaver in *CPC* encoding operation guarantees $d_{min} = d_{free}^2$ equality. However, for *PDSCCCs* with *RCS-random* interleavers $d_{min} = d_{free}^2$ is not guaranteed unless completely different random column interleavers are used during the row-column interleaving operation. This is due to a similar discussion in the previous sub-section. For a frame length of 1024, the data matrix size is 32×32, row encoded matrix size is 32×68, it is possible to find 68 completely different column random interleavers of length 32.

4.5.6 Asymptotic Performance

If row and column convolutional codes are trellis terminated, the row and column convolutional codes can be considered as block codes. Asymptotic performance studies made for block codes are here valid for convolutional product codes. BER probability of the *CPCs* can be approximated using the formula,

$$P_b \simeq \frac{N_{c,d_{free}}^2 w_{c,d_{free}}^2}{k^2} Q\left(\sqrt{d_{free}^2 \frac{2E_s}{N_0}}\right), E_s = RE_b, \tag{4.2}$$

where d_{free} is the free distance of the convolutional code used to construct the convolutional product code. $N_{c,d_{free}}$ is the number of convolutional codewords with free distance d_{free}, k^2 is the length of the data information frame, and $w_{c,d_{free}}$ is the average Hamming weight of the information words that produces convolutional codewords with Hamming weight d_{free}, R is the overall rate of the *CPC*. The BER approximation in (4.2) is valid if any one of the rectangular interleaver, column *S-random* interleaver, or row-column *S-random* interleaver is used during the *CPC* encoding operation.

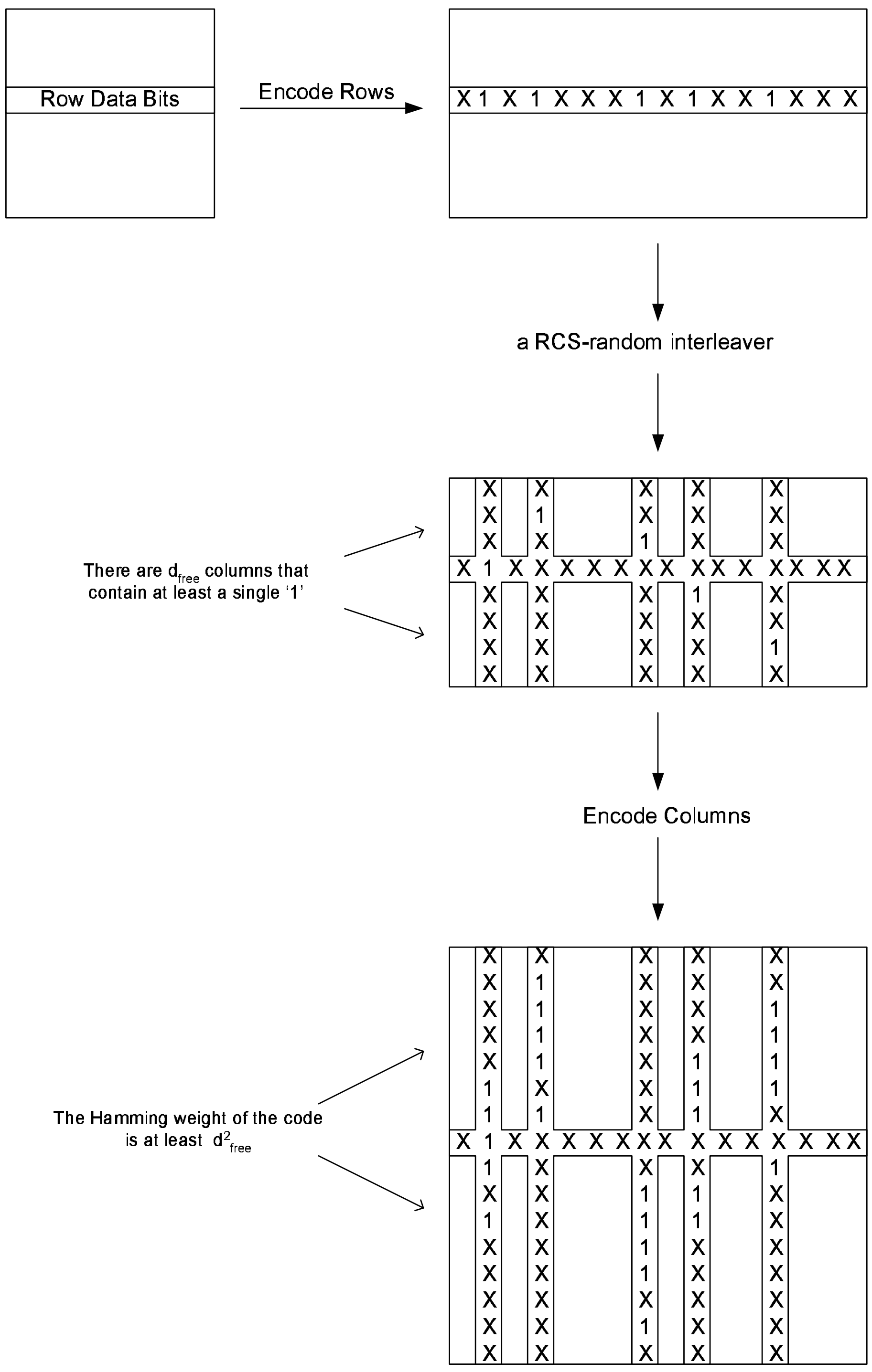

Row Data Bits
Encode Rows
X 1 X 1 X X X 1 X 1 X X 1 X X X
a RCS-random interleaver
There are d_{free} columns that contain at least a single '1'
Encode Columns
The Hamming weight of the code is at least d^2_{free}

Figure 4.14: *CPC* Encoding operation using *RCS-random* interleaver. ('x' stands for a single or a group of 0's).

4.6 Practical Implementation Advantages

The implementation advantage of *CPC* will be discussed herein with the parameters used in this study. Trellis termination will be neglected in calculation and will not alter the results significantly with regard to complexity. Let t and c be latency and computational complexity of a single stage of *RSCs'* trellis where identical codes are used as constituent codes. In *SCCC*, for a given transmit data vector of length L, two log-MAP decoders are needed. The first decoder has a complexity $c.(2L)$ and a time delay $t.(2L)$. The second decoder has a shorter input, thus it has a complexity $c.(L)$ and a time delay $t.(L)$. In total, the complexity is $c.(3L)$ and the time delay is $t.(3L)$. In *CPC* columns are decoded first. The use of separate log-MAP decoders for each row and column makes parallel processing operations possible. Each column decoder has complexity $c.(\sqrt{L})$ and time delay $t.(\sqrt{L})$. Since these decoders are run in parallel the total column decoding complexity is $c.(2L)$ but the time delay is $t.(\sqrt{L})$. Similarly, row decoding has a total complexity $c.(L)$ and time delay $t.(\sqrt{L})$. Hence, although both complexities are the same, time delays differ very much and brings about a $\sqrt{L}$ times increase in decoding rate. Hence, the main advantage of *CPCs* lies on its suitability for parallel decoding procedure. Although there are some proposed methods for the parallel decoding of *SCCCs* and *PCCCs*, these methods usually propose extra algorithms to solve problems met in parallel processing. (Such algorithms not only bring extra complexity to the decoding operation [30, 40], but also may suffer from performance loss). This situation is totally remedied with the proposed *CPCs*.

4.7 Performance of *CPCs*

We used recursive systematic convolutional code $(1, 5/7)_{octal}$ for all the constituent encoders. An *S-random* interleaver ($S = 18$) is employed between clusters. Input information sequence frame length is chosen as 1024 bits. We formed a size 32×32 information matrix. Thus, N (number of row decoders) equals 32 and M (number of column decoders) equals 68 (i.e., trellis termination bits are added after row encoding). The encoded data in matrix form is multiplexed to a single stream and binary phase shift key *(BPSK)* modulated. The *BPSK* modulated signal is passed through an additive white Gaussian noise *(AWGN)* channel with double-sided power spectral density $\frac{N_0}{2}$. We used the *log-MAP* soft decoding algorithm [13] to iteratively decode the *CPCs*. We used 12 iterations for decoding. We also simulated the serially concatenated convolutional code which is a special case of *CPCs* with $M = N = 1$. The signal-to-noise ratio values given in the figures are normalized with the proper code rates for all scenarios with trellis termination taken into account. Non-recursive systematic convolutional codes were also tried and it was seen that their performance is not as good as the *CPC* system where *RSCs* are employed. The type of the interleaver is very critical on the performance of the *CPCs*. We will separately investigate the effects of each case. Trellis termination and puncturing effects will be investigated separately.

4.7.1 Trellis Termination Effects

We investigated the effects of trellis termination for three different scenarios. Trellis termination bits are added by the outer *RSC* encoders (row encoders) *(R-T)* with rate 0.235, both outer and inner constituent *RSC* encoders (row and column encoders) added trellis termination bits to the codeword *(TT)* with rate 0.221, neither outer and inner constituent *RSC* encoders added trellis termination bits to the codewords *(No-TT)* with rate 0.25. Although addition of trellis termination bits decreases the code rate, they are so critical for good performance of the *CPCs* as seen in Fig. 4.15. The addition of trellis termination bits in turbo or serially concatenated code brings a negligible improvement on the code performance [53]. However, without trellis termination the performance of the *CPCs* degrades drastically.

The performance graphs are seen in Fig. 4.15. *CPCs* have better performance at very low E_b/N_0 levels when only inner codes are trellis terminated.

They perform worse at higher E_b/N_0 levels when compared to the inner and outer codes trellis terminated case. Though quite close up to *BER* 10^{-7}, *SCCC* seems to have an error curve of higher slope compared to *TT* at higher E_b/N_0 values. The analytical bound is evaluated for *TT* case using the uniform interleaver approach. However as it is seen from the Fig. 4.15 the bound obtained by uniform interleaver concept gives a rough idea about the code performance.

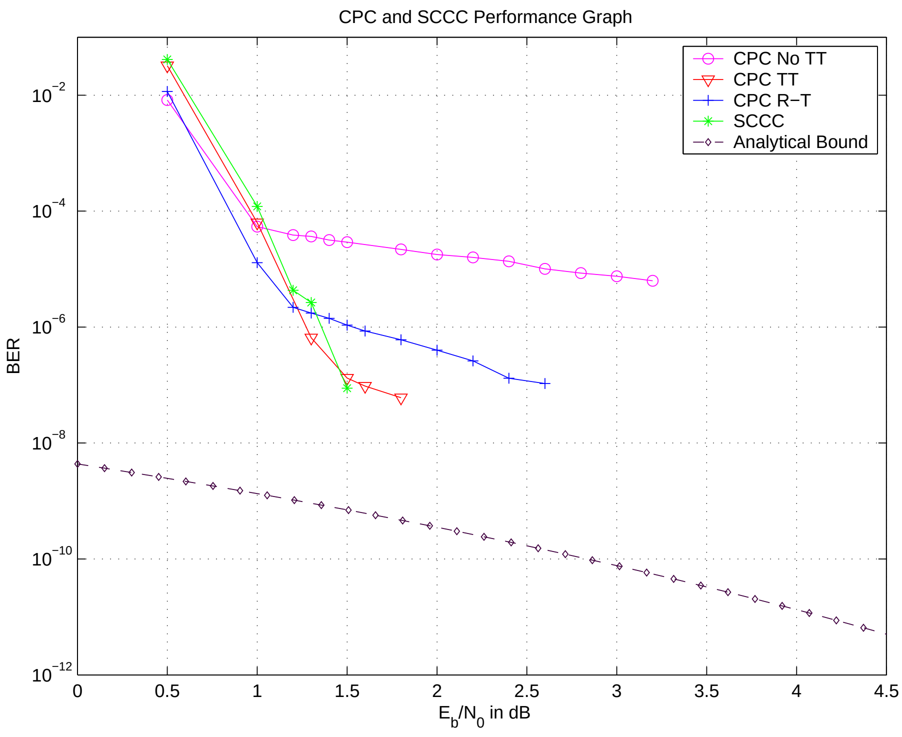


Figure 4.15: *CPCs* and *SCCC* performance graph. *CPCs* with no trellis termination *(CPC No TT)*. *CPCs* when rows (outer codes) are trellis terminated *(CPC R-T)*. *CPCs* when rows and columns (outer and inner codes) are trellis terminated *(CPC TT)*. Frame length=1024, Iteration number=12.

4.7.2 Interleaving Effects

4.7.2.1 Rectangular Interleaver

In this case, rectangular interleaving operation is performed after row encoding. Trellis termination bits are added both to rows and columns. The addition of trellis termination bits are necessary to guarantee that each row and column has at least d_{free} number of '1's. The total number of '1's which is the minimum distance of the *CPC* is $d_{min} = d_{free}^2 = 25$. With trellis termination bits worst case $d_{min} = d_{free}^2$, otherwise d_{min} is not equal to d_{free}^2 anymore. The performance graph of this code is shown in Fig. 4.16. It is seen from the graph that the performance of this *CPC* is not good for low *SNR* values, although its minimum distance is large. This is due to large multiplicities of low weight codewords. As well known, minimum distance dominates the performance of the code at high *SNR* values.

4.7.2.2 Full *S-random* Interleaver

After the row encoding operation, an *S-random* interleaver ($S = 18$) is used. We also simulated a serially concatenated convolutional code to compare against *CPC*. The performance graph is seen in Fig. 4.16. As seen from the performance curve, the performance is very good compared to the cases where interleavers different than *S-random* are used. Due to the *S-random* interleaver use after row encoding, the minimum distance of the *CPC* is not necessarily equal to d_{free}^2. In fact, the worst case d_{min} of *CPC* with a full *S-random* interleaver is lower bounded by d_{free}. Although *CPC* with a full *S-random* interleaver has a smaller worst case d_min, it shows the best performance at low rates due to the large interleaver gain and low multiplicities of the minimum distance codewords.

4.7.2.3 Column *S-random* Interleaver

To obtain both better performance than the no interleaver case and to preserve the $d_{min} = d_{free}^2$ of *CPC*, we applied an *S-random* interleaver ($S = 3$) to each column separately. We called such interleaving as column *S-random* interleaving. Different column elements are not mixed. From Fig. 4.16 it is seen that the performance is better compared to the *CPC* in which rectangular interleaver is used. Its performance is worse than *CPC* where a full *S-random* interleaver is used after row encoding. The use of the helical interleaver also guarantees that minimum distance of *CPC* equals d_{free}^2. We also investigated the case that an helical interleaver is followed by a column *S-random* interleaver. It is seen that such an interleaver results in slightly better performance than the one where only column *S-random* interleaver is used during the encoding procedure.

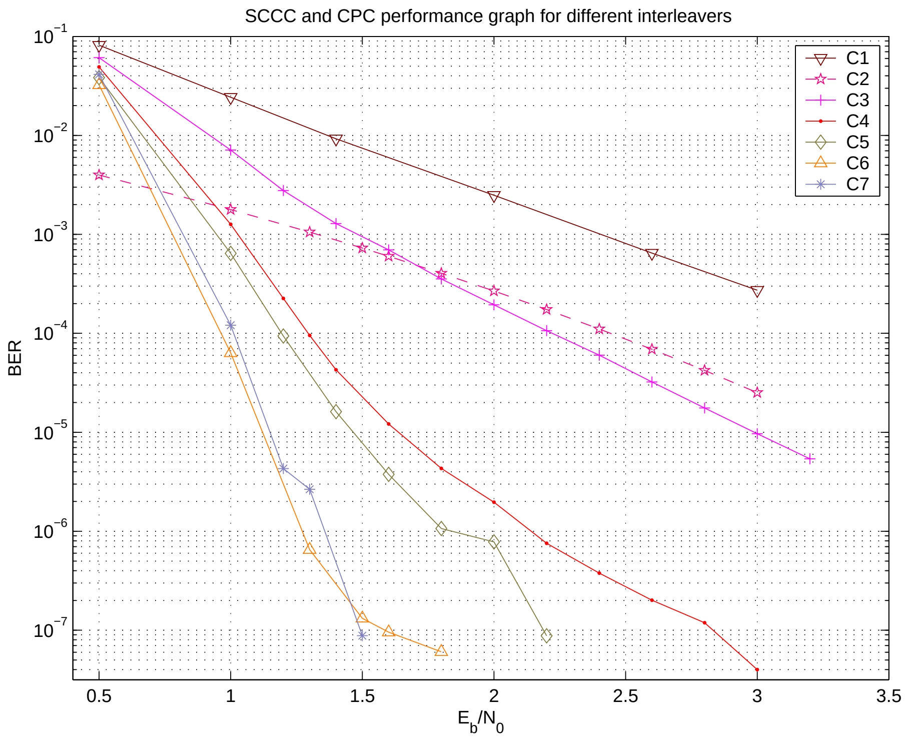


Figure 4.16: *SCC* and *CPC* performance graph for different interleavers. Iteration number=12. Frame length 1024. The graph is explained below.
C1: Rectangular interleaver is used (Rate $\approx 1/4$)
C2: Theoretical bound for *CPC* with rectangular interleaver using eqn. 4.2 (Rate $\approx 1/4$)
C3: Helical interleaver is used (Rate $\approx 1/4$)
C4: Each column is *S-random* interleaved (Column *S-random*) (Rate $\approx 1/4$)
C5: Helical + column *S-random* interleaver is used (Rate $\approx 1/4$)
C6: Full *S-random* interleaver is used (Rate $\approx 1/4$)
C7: *SCCC* with *S-random* interleaver (Rate $\approx 1/4$)

4.7.3 Puncturing Effects

Puncturing is a widely used tool to increase the code rate of convolutional codes [54]. The puncturing operation increases the rate of a code, but decreases the free distance. This results in a worse error rate performance compared to the non-punctured case. We used the puncturing vectors [1110] and [11101010] to puncture the convolutional component codes. We applied the puncturing vector to columns only and both to rows and columns. For the puncturing vector [1110] , when puncturing is applied only to the column encoders, it results in a code rate of 2/3 each. The overall code rate becomes $(1/2)\times(2/3) = 1/3$. When trellis termination is used for rows and columns, a convolutional code with a slightly smaller overall code rate ($\leq 1/3$) is produced. In the other case, when puncturing is applied to each row and column encoder, it results in an increased code rate of approximately $(2/3) \times (2/3) = 4/9$. When the puncturing vector [11101010] is employed, the code rates are 2/5 for row punctured *CPCs* and 16/25 for row-column punctured *CPCs*.

The puncturing operation decreases the free distance of convolutional codes. We puncture the $(1, 5/7)_{octal}$ component convolutional code which has $d_{free} = 5$, i.e., an input sequence '0111' produces minimum Hamming weight codeword '00111011'. When every other parity bit punctured its free distance decreases to $d'_{free} = 3$, i.e., '001x101x' is obtained from minimum Hamming weight codeword. Hence, the *CPCs* constructed using punctured component convolutional codes have smaller minimum distance. In fact, the minimum distance equals $d^2_{free} = 9$, if no interleaving operation is performed or column *S-random* interleaver is used. When puncturing vector $v = [11101010]$ is employed, the free distance decreases to $d'_{free} = 3$.

When puncturing vector $P = [1110]$ is applied only to rows, it results in a rate 2/3 *CPC*. From Fig. 4.17, it is seen that the performance of the *CPC* with a full *S-random* interleaver is good after being punctured. Its performance is comparable to that of the *SCCC*. When the puncturing process is applied to both rows and columns, it results in a *CPC* of rate approximately 4/9. If puncturing vector $P = [11101010]$ is employed code rate increases up to 16/25. The simulation results of rate 4/9 *PDSCCCs* and *CPCs* using different interleavers are depicted in Fig. 4.18. It is clear from Fig. 4.18 that rate 4/9 *CPC* with row-column *S-random* interleaver shows the best performance, this can be attributed to the guaranteed largest minimum distance d^2_{free}. For code rate 16/25, simulation results are illustrated in Fig. 4.19. In Fig. 4.19 two *CPCs* utilizing full *S-random* and row-column *S-random* interleavers are compared. Recall that d_{min} is not necessarily lower bounded by d^2_{free} when an *S-random* interleaver is used. However, when row-column *S-random* interleaver is used $d_{min} \geq d^2_{free}$. This results in a better performance and it is clearly seen in Fig. 4.19. When both punctured code simulation graphs are compared, it is seen that as the puncturing amount increases, the minimum distance of the code becomes a critical factor for the code performance.

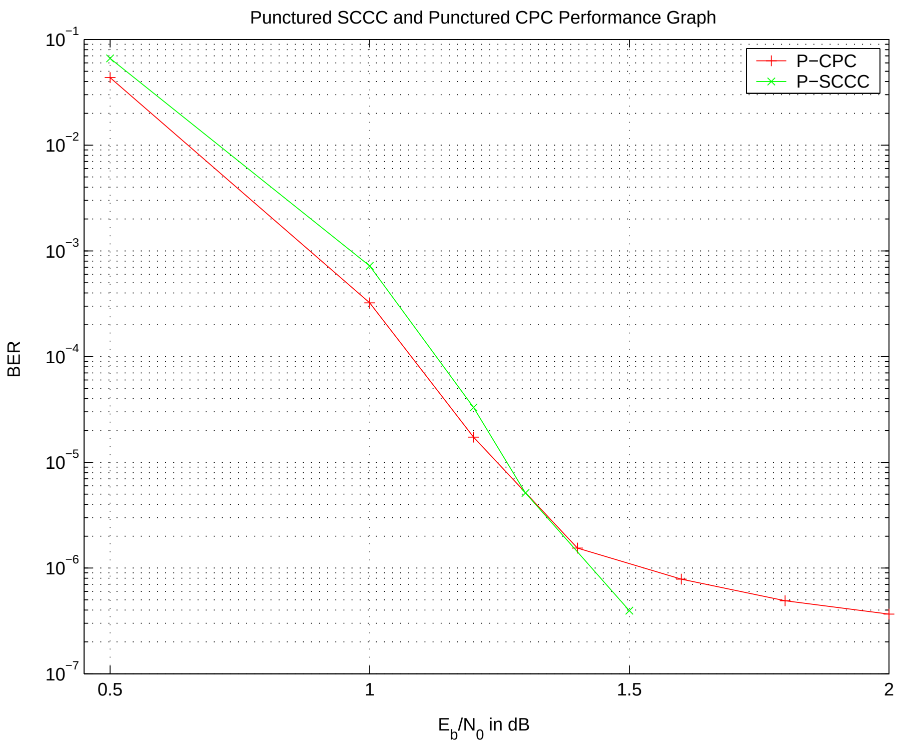


Figure 4.17: Punctured *CPC* (Rate:2/3) and punctured *SCCC* (Rate:2/3) simulation graph.

4.8 Peformance of *PDSCCCs*

We simulated *PDSCCCs* using row-column *S-random* interleaver and full *S-random* interleavers. To prevent memory collision issue *RCS-random* interleaver is used in *PDSCCC*. The performance results for *PDSCCC* with *RCS-*

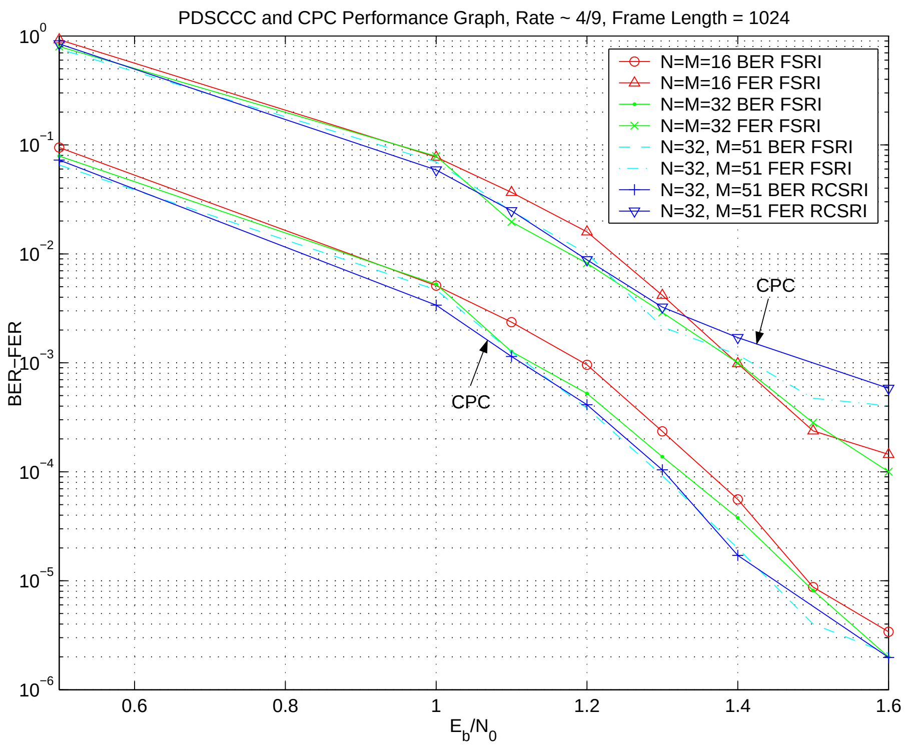


Figure 4.18: *PDSCCC* and *CPC* Performance graph. RCSRI is the row-column *S-random* interleaver. FSRI is the full *S-random* interleaver. Frame length = 1024. Iteration number = 12

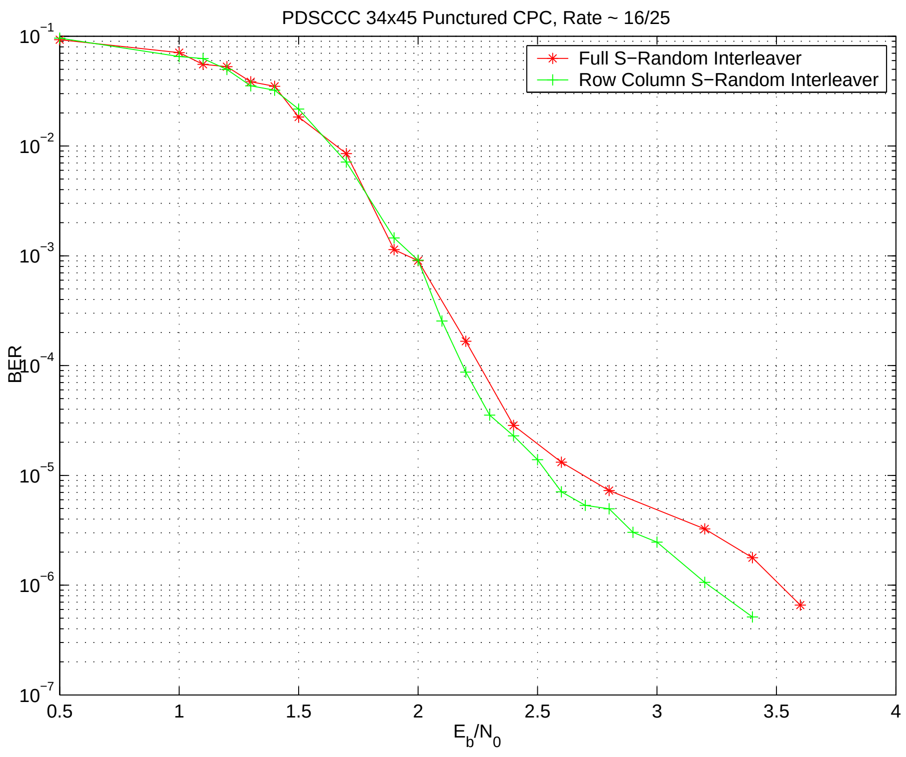


Figure 4.19: Punctured *CPC* Performance graph when full *S-random* and row-column *S-random* interleaver is employed. Rate$\simeq$ 16/25. Frame length = 1024. Iteration number = 12

random interleaver and full *S-random* interleaver is depicted in Fig. 4.20. It is clear from this figure that, *RCS-random* interleaver shows almost the same performance when compared to *S-random* interleaver. However, for moderate E_b/N_0 values they both achieve almost the same performance. In Fig. 4.21 *SCCC* is compared to that of the *PDSCCC* whose outer and inner cluster numbers are 16. As it is obvious from Fig. 4.21 *PDSCCC* shows better performance at high E_b/N_0 values. For low E_b/N_0 regions *SCCC* shows slightly better performance.

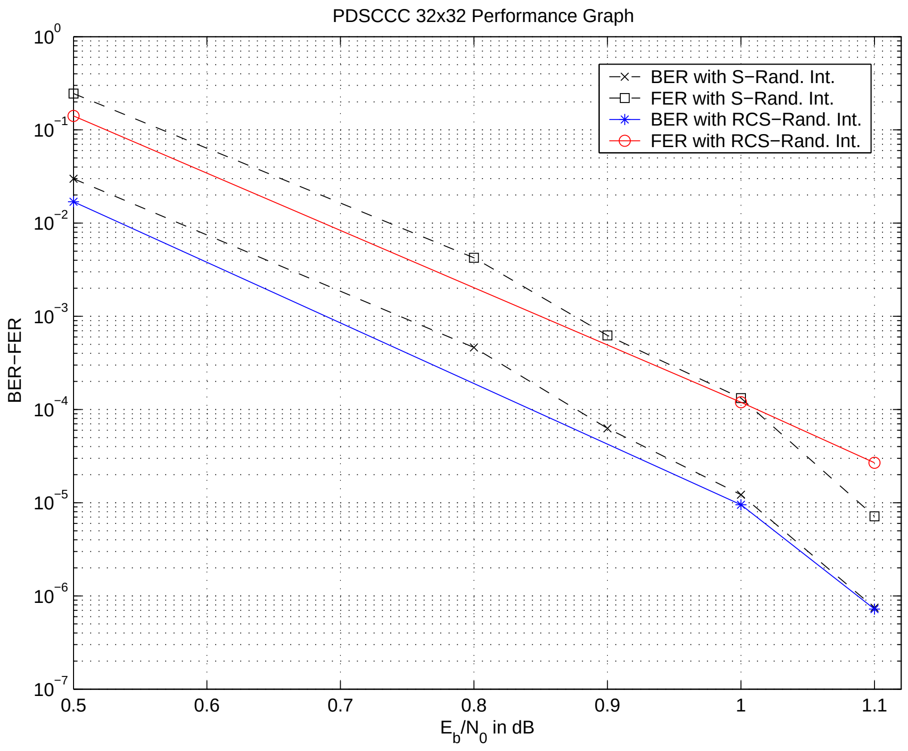


Figure 4.20: PDSCCC Performance graph for full *S-random* interleaver and collision free *RCS-random* interleaver. N=M=32. Frame length=1024.

4.9 Analytical Analysis of *PDSCCCs*

An analytical bound expression for the performance of serially concatenated codes is given in [32] where the authors employ uniform interleaving. The uniform interleaver permits the estimation of the average interleaver gain, independent of the particular interleaver used in concatenated codes. Thus, the

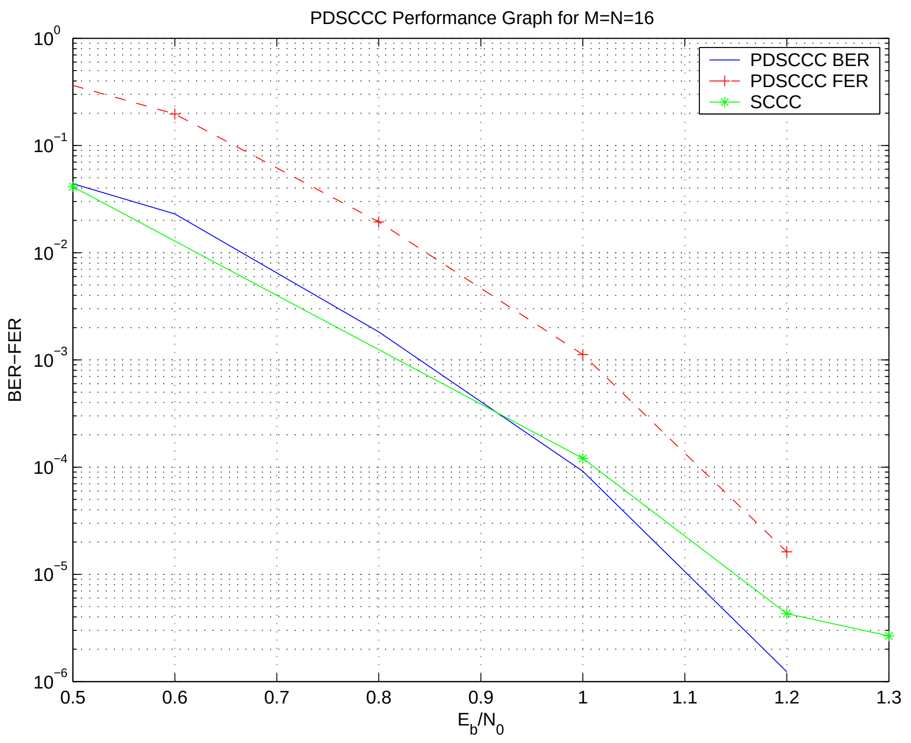


Figure 4.21: *PDSCCC* and *SCCC* performance graph. N=16, M=16. Frame length = 1024. Iteration number = 12

performance of a *SCCC* over all possible interleavers can be determined. We define the functions necessary to evaluate the performance in the remainder of this section.

Using spectrum functions of the *SCCCs*, performance upper bound to the bit error probability for maximum likelihood soft decoding of the code is found in the form [26],

$$P_b(e) \leq \sum_{w,i} \frac{w}{k} A_{w,i} erfc(\sqrt{\frac{iRE_b}{N_0}}). \tag{4.3}$$

This is also written using two separate terms,

$$P_b \cong \frac{1}{2} \sum_{m} D_i erfc(\sqrt{\frac{iRE_b}{N_o}}) \tag{4.4}$$

where R is the code rate, $\frac{E_b}{N_0}$ is the bit energy to noise ratio of the *AWGN* channel, D_i is obtained from the *IOWEF* coefficients according to:

$$D_i = \sum_w \frac{w}{k} A_{w,i}. \tag{4.5}$$

4.9.0.1 IOWEF of a Serial Concatenated Code

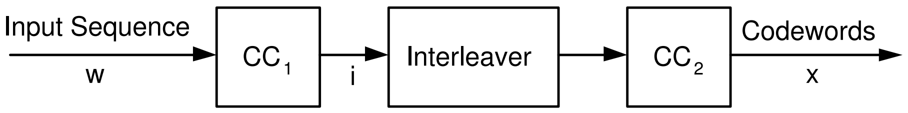


Figure 4.22: Serial concatenated code. $CC1$ and $CC2$ are constituent codes *CCs* 1 and 2, respectively. These can be block codes, convolutional codes or a mix of both. The interleaver size is L.

IOWEF $A^{C_s}(W,X)$ of serial concatenated code is expressed as a product of the two *CWEFs* of the constituent codes, which is normalized by the number of the possible permutations (i.e., uniform interleaver)

$$A^{C_s}(W,X) = \sum_{i=0}^{L} \frac{A^{CC_1}(W,i) \times A^{CC_2}(i,X)}{\binom{L}{i}}. \tag{4.6}$$

Once *IOWEF* of a concatenated code is available, upper bound to the bit error probability can be calculated using (4.4) and (4.5).

4.10 BER Bounds for *PDSCCCs*

The uniform interleaver approach is used to evaluate the average performance of the *PDSCCCs*. Since the information bits of parallel subsequences of inner and outer clusters are independent, *IOWEFs* of the inner and outer clusters can be evaluated as

$$A^{C_1}(W,X) = \prod_{i=1}^{N} A_{1i}(W,X) \tag{4.7}$$

$$A^{C_2}(W,X) = \prod_{i=1}^{M} A_{2i}(W,X), \tag{4.8}$$

where $A_{1i}(W,X)$ and $A_{2i}(W,X)$ are the *IOWEFs* of the CCs in inner and outer clusters. We considered different scenarios and evaluated analytical bounds for the performance of the CPCs. The analytical bounds for square input matrices are shown in Fig. 4.23 for different interleaver lenghts. Trellis termination bits are used by both inner and outer clusters. It is clear from the Fig. 4.23 that as the interleaver size increases better performance is obtained.

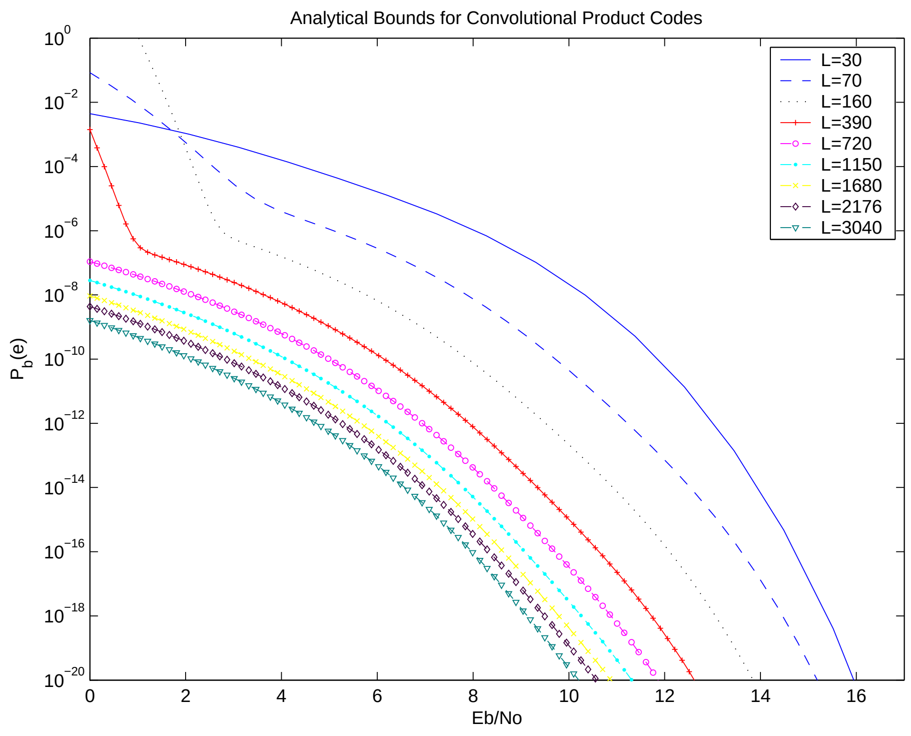


Figure 4.23: *CPCs*' analytical bounds for different interleaver sizes. Square input matrices are used. Trellis termination bits are added to codewords by both outer and inner encoders. Constituent Encoders are *RSC* $(1, 5/7)_{octal}$. L is the interleaver size.

We considered shorter interleaver lenghts and analyzed asymmetric cases in Figs. 4.24 and 4.25. It is obvious from the Figs. 4.24 and 4.25 that *CPCs*' performance is comparable to that of *SCCCs* (i.e., case M=N=1) for smaller M and N values (e.g., M=N=8 or N=1, M=8).

For larger interleaver sizes the bounds are shown in Figs. 4.26 and 4.27. With a larger interleaver length of 2176 one can choose greater M and N values such that almost the same performance is achieved as that of *SCCCs*, e.g. the bounds for N=M=32 and N=M=1 (i.e., *SCCC*) cases are almost the same. When greater M and N values are used, the degradation in performance is much less as compared to the case where shorter interleaver lengths (256) are used.

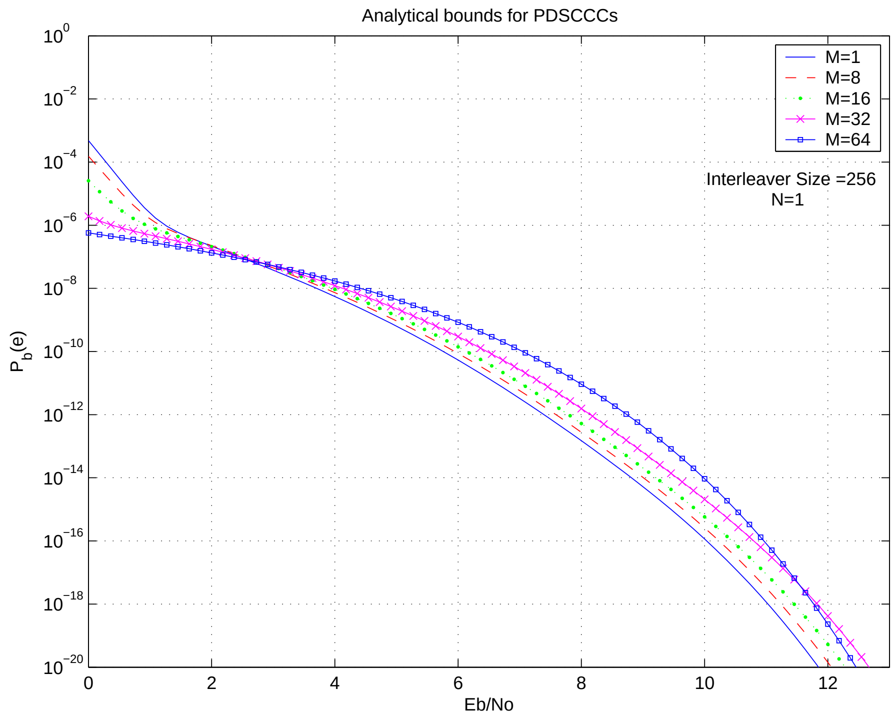


Figure 4.24: *PDSCCC* analytical bounds for small interleaver size.

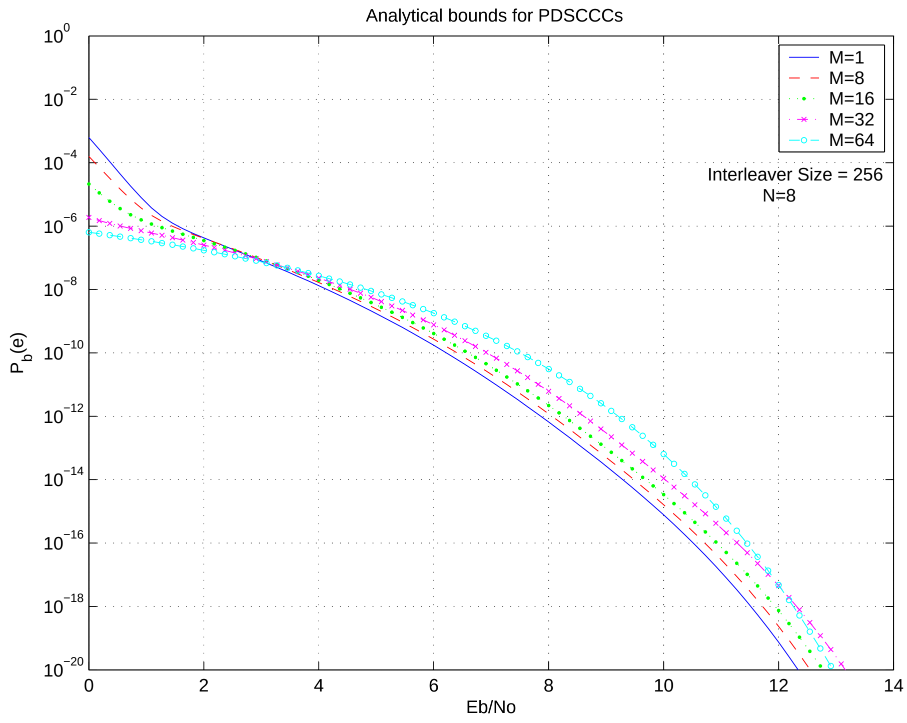


Figure 4.25: *PDSCCC* analytical bounds for a small interleaver size.

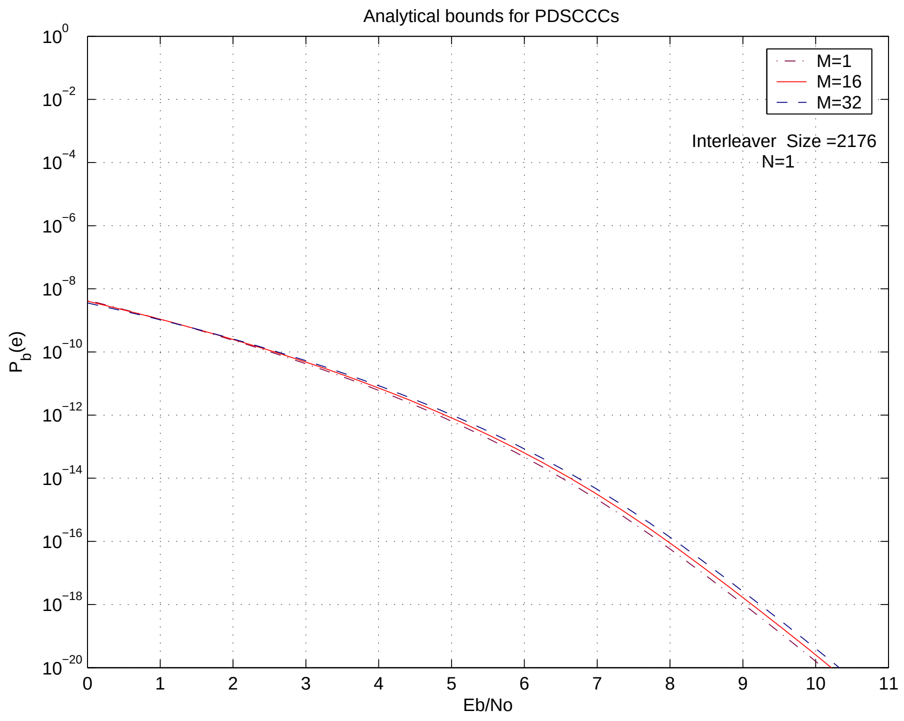


Figure 4.26: *PDSCCC* analytical bounds for large interleaver size.

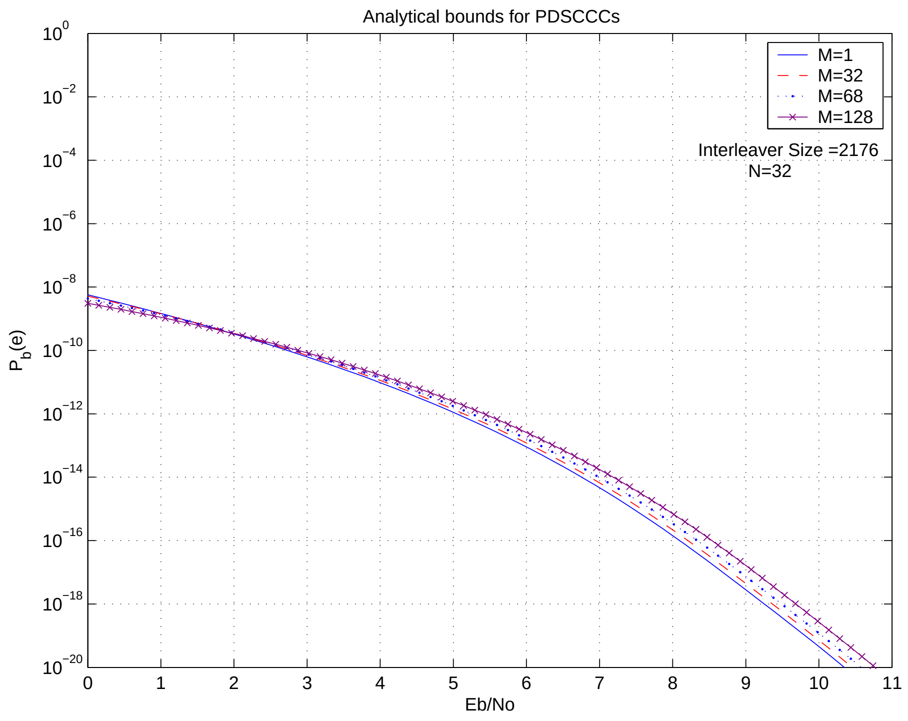


Figure 4.27: *PDSCCCs*' analytical bounds for large interleaver size.

Analytical bounds are also evaluated for the case when the trellis termination is not applied to the constituent codes as shown in Fig. 4.28. When compared to Fig. 4.23 the degradation in performance is obvious.

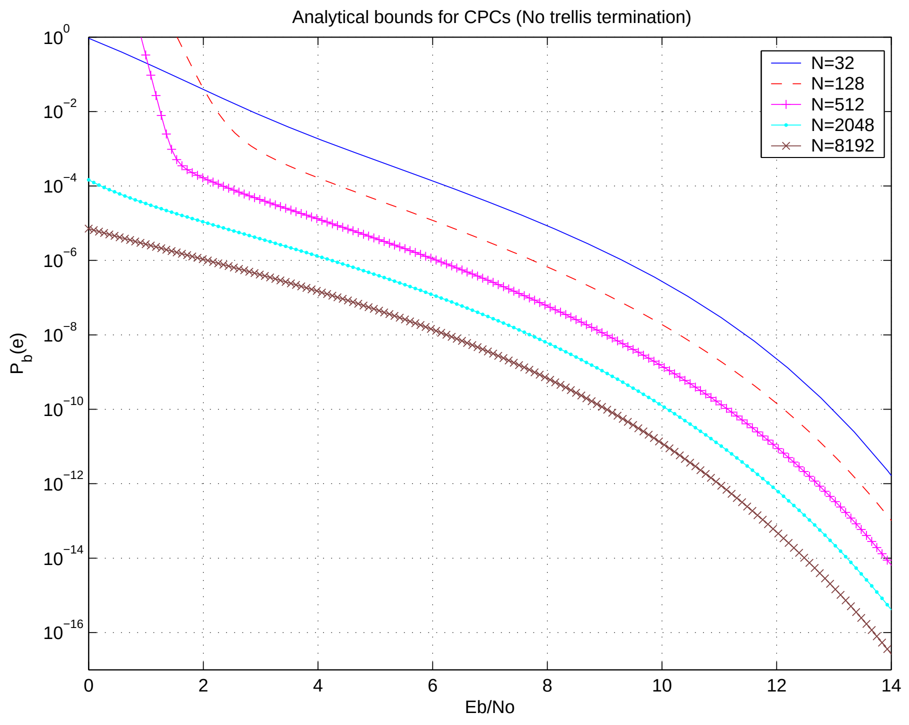


Figure 4.28: *CPCs*' analytical bounds when trellis termination bits are not used.

CHAPTER 5

PARALLEL DECODABLE TURBO CODES

In this chapter we propose a new structure for turbo codes which is suitable for parallel decoding operations. The proposed scheme here utilizes parallelization at the encoder side and uses it for parallel decoding at the receiver side. Minimum distance of the proposed code is computed. The effects of different interleavers on code performance is investigated. Parallel decodable turbo code systems with and without memory collision interleavers are simulated.

5.1 Introduction

Turbo codes show good performance at low *SNR* regions. However their high decoding latency lower their attractiveness for practical implementations. The use of multiple processors for decoding is a way to reduce the decoding latency. In this chapter we introduce a low latency architecture for turbo codes, and the proposed architecture can easily be decoded using multiple processors. Main benefit of this structure lies on reduced latency by a factor of number of parallel processors employed. On the other hand, the proposed structure hardware complexity is increased by the same factor.

Although the use of multiple processors decreases decoding latency significantly, some difficulties and drawbacks in practical implementations arises. One difficulty is the memory collision problem which occurs mainly due to the permutation order of the interleaver. This problem can be avoided by designing collision free interleavers. There are a number of collision free interleavers suggested. In [55] an algebraic collision free interleaver design method is proposed. This method can be applied to block or multi-state interleavers. Dividable interleaving method to prevent collision is introduced in [56]. Parallelization within iteration is achieved in [57] by the use of a collision free rectangular type interleaver designed with an empirically found set of parameters. In [30, 58, 59] a number of algebraic techniques for parallel turbo code interleaver design are introduced. In this chapter, we also employ our collision free row-column *S-random* interleaver introduced in previous chapter for the parallel decodable turbo codes. Analytical analysis of the proposed structure is performed using the uniform interleaver approach.

5.2 *IRWEF* of Parallel Concatenated Code

The classical encoder and decoder of a typical parallel concatenated code (C_p) is shown in Fig. 5.1. Constituent codes can be chosen either from block codes or convolutional codes only. Block and convolutional codes can also be concatenated in parallel. If constituent codes are selected from recursive systematic convolutional codes *(RSCs)* the well known Turbo codes are obtained.

Using the uniform interleaver approach, *CWEF* of the parallel concatenated code is found by [26],

$$A_w^{C_p}(Z) = \frac{A_w^{CC_1}(Z) \times A_w^{CC_2}(Z)}{\binom{L}{w}}, \tag{5.1}$$

where $A_w^{CC_1}(Z)$ and $A_w^{CC_2}(Z)$ are the *CWEFs* of the constituent codes CC_1 and CC_2, $A_w^{C_p}(Z)$ is the *CWEF* of the parallel concatenated code. L is the interleaver size, w and z are the Hamming weights of the information and parity sequences, respectively. We obtain the *IRWEF* of the code C_p as in

$$A^{C_p}(W, Z) = \sum_{w=1}^{k} W^w A_w^{C_p}(Z), \tag{5.2}$$

where k is the length of the input information sequences. A performance upper bound to the bit error probability for the maximum likelihood soft decoding

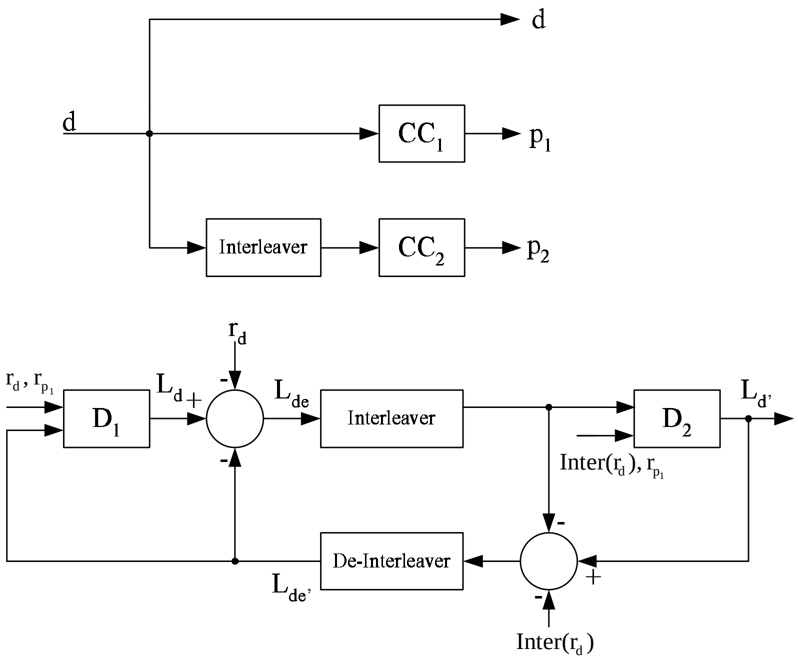


Figure 5.1: Parallel concatenated code. CC_1, CC_2 are constituent recursive systematic convolutional codes 1 and 2 respectively, D_1 and D_2 are soft-in soft-output decoders. r_d, r_{p_1} and r_{p_2} are the received signal values for data and parity bits. L_d, L_{de}, L'_{de} are log-likelihood values.

of the code in an *AWGN* channel of double sided noise power spectral density $N_0/2$ is found as below [26]

$$P_b \cong \frac{1}{2}\sum_m D_m erfc(\sqrt{m\frac{R_c E_b}{N_o}}), \tag{5.3}$$

where R_c is the code rate, $\frac{E_b}{N_o}$ is the bit energy to noise ratio of the *AWGN* channel, D_m is obtained from the *IRWEF* coefficients according to

$$D_m = \sum_{z+w=m} \frac{w}{k} A_{w,z}. \tag{5.4}$$

We give an example in Appendix B for the computation of eqn. (5.3) for a block code.

5.3 Parallel Decodable Turbo Codes *(PDTCs)*

Our proposed structure is seen in Fig. 5.2. As it is seen from Fig. 5.2 that we employ parallelization at the encoder side and use this parallel structure at the decoder side. The input information sequence is sent to a serial to parallel converter to form subsequences, and these subsequences are encoded using *(RSCs)*. Although we employ only *RSCs* for constituent codes any combination of block and convolutional codes can be used for constituent codes. The number of *RSCs* in the upper cluster is denoted by N, and M is the number of *RSCs* in the lower cluster. The decoder is depicted in Fig. 5.3. Serial-to-parallel and parallel-to -serial converters are employed in this structure. Serial-to-parallel converter gets an input sequence and forms a matrix, the matrix elements can be filled row-wise or column-wise. We prefer to fill the matrix using row-wise. Parallel to serial converter does the opposite job, i.e., it forms a sequence of bits by either multiplexing matrix columns or multiplexing matrix rows. Although the encoding operation can be performed using multiple processor, it is not a necessity. A single processor can also be used at the encoder side. As it is clear from Fig. 5.3 that decoders $D_{11}, \ldots, D_{1N}$ and $D_{21}, \ldots, D_{2M}$ can be run in parallel. This reduces decoding delay considerably. For turbo codes upper encoder should be trellis terminated, however this is not a necessity for the lower encoder. Constituent codes in upper and lower clusters may use trellis termination bits. This decreases the code rate. If all the constituent codes are trellis terminated than the code rate for *PDTC* is computed using

$$R = \frac{L}{(\frac{1}{R_1} + \frac{1}{R_2} - 1)L + \frac{T_1 N}{R_1} + \frac{T_2 M}{R_2}}, \tag{5.5}$$

where R_i and T_i denote rates and number of memory elements for the constituent codes in clusters. The effect of the trellis termination bits on the code rate for different number of parallel branches are shown in table 5.1. Trellis termination bits decrease the code rate more when compared to rate table for *PDSCCCs*.

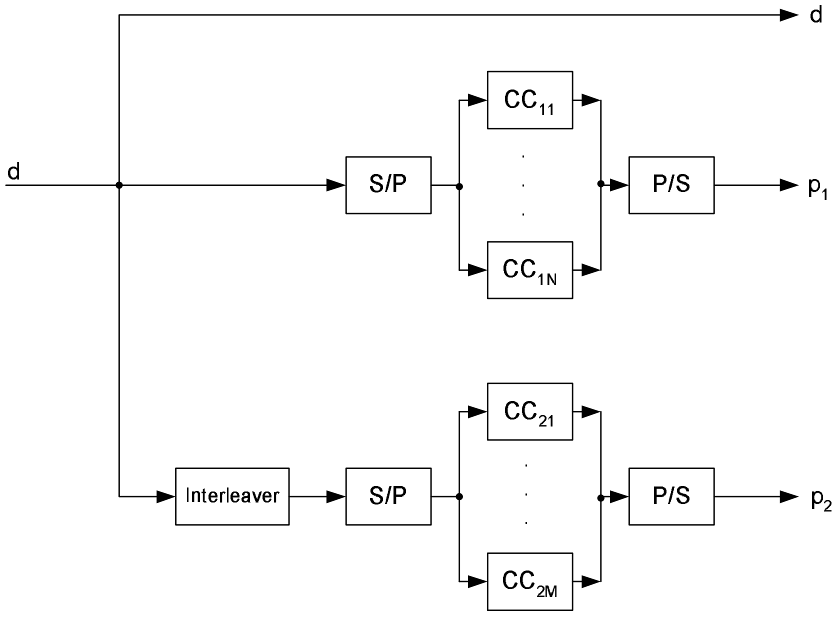


Figure 5.2: *PDTC* encoder. CC_{1i}, CC_{2i} are constituent *RSCs*. S/P and P/S are the serial to parallel and parallel to serial converters.

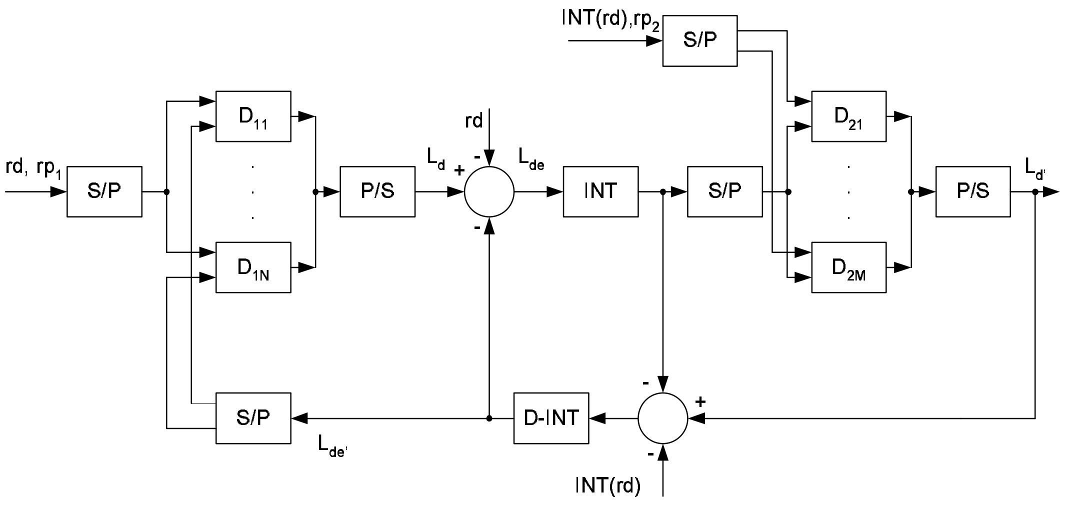


Figure 5.3: *PDTC* decoder. $D_{1i}, D_{2j} (i = 1 \ldots N, j = 1 \ldots M)$ are the decoders for *RSCs* CC_{1i} and CC_{2i} respectively. r_d, r_{p_1} and r_{p_2} are the received signals for data and parity bits. L_d, L_{de}, L'_d and L'_{de} are log-likelihoods.

Table 5.1: *PDTC* rates for different number of parallel branches in outer and inner clusters. Interleaver size is 1024. $R_1 = R_2 = 1/2$, $T_1 = T_2 = 2$.

N \ M	1	4	8	16	32	64
1	0.332	0.331	0.329	0.326	0.319	0.307
4	0.331	0.329	0.328	0.324	0.318	0.306
8	0.329	0.328	0.326	0.323	0.316	0.304
16	0.326	0.324	0.323	0.320	0.313	0.301
32	0.319	0.318	0.316	0.313	0.307	0.296
64	0.307	0.306	0.304	0.301	0.296	0.285

5.3.1 Decoding Delay

Let c and t be the computational complexity and latency of a single stage of the code trellis. In a classical turbo code, for a given transmit data vector of length L, two log-MAP decoders are needed. Both decoders have a computational complexity of $(c.L)$ and a time delay of $(t.L)$. In total, the complexity of each iteration is $(c.2L)$ and the time delay is $(t.2L)$. In *PDTC* the use of separate log-MAP decoders for each constituent code makes parallel processing operation possible. Assuming that M = N, all decoders in the first and second decoder clusters have a complexity of $(c.L/\text{N})$ and a time delay of $(t.L/N)$ ignoring the use of trellis termination bits. The total computational complexity is $(c.2L)$ and the decoding delay is $(t.2L/\text{N})$. As seen from this discussion, the decoding delay is reduced by a factor of N while keeping the total computational complexity almost the same as before (excluding a small overhead for the serial to parallel and the opposite operations). Naturally, the decoding latency decreases in this case at the expense of increased hardware complexity as in all parallel processing operations. In fact, hardware complexity is increased by the same factor of number of parallel processors employed, assuming that there are equal number of decoders in clusters. Hence, a tradeoff between latency and hardware complexity is observed.

5.4 Performance Analysis of Parallel Decodable Turbo Codes

We used the $(1, 5/7)_{octal}$ *RSC* for all the constituent codes. Trellis termination is performed in each encoder where both the terminating data and parity bits are appended to the encoded sequence. This slightly decreases the overall code rate and is taken into account while calculating the $\frac{E_b}{N_0}$ values both in computation of the analytical bounds and in simulations. $A_{1i}(W, Z)$'s are the *IRWEF*'s of the constituent codes $CC_{1i}(i = 1, \ldots, N)$, similarly $A_{2i}(W, Z)$'s are the *IRWEF*'s of the constituent codes $CC_{2i}(i = 1, \ldots, M)$. $A_1(W, Z)$, $A_2(W, Z)$ are the *IRWEF* of the equivalent upper and lower codes respectively. Since the information bits are independent, these *IRWEF*'s can be evaluated as

$$A^1(W, Z) = \prod_{i=1}^{N} A_{1i}(W, Z) \tag{5.6}$$

$$A^2(W, Z) = \prod_{i=1}^{M} A_{2i}(W, Z). \tag{5.7}$$

We evaluated *IRWEF*'s for different lengths of input information sequences. Once *IRWEF*'s of the *PDTC*'s are available, *the uniform interleaver* analytical bound approach can be applied and equations (5.3), (5.4) can be used to determine upper bounds. Analytical bounds for different interleaver lengths and N, M values are shown in Figs. 5.4 and 5.5.

In Fig. 5.4 the analytical bounds for a number of M = N values are depicted. The code rates are readily calculated with L/(3L + 4(M + N)) by taking trellis termination into account. The classical turbo code corresponds to M = N = 1. It is seen that there is a rise in the error floor with increasing M and N. However, when a larger interleaver size is used as in Fig. 5.5, it is seen that *PDTC* error floor is almost the same as that of the classical turbo code for moderate M and N values. When asymptotes of the bounds are inspected, i.e., at very high SNR, it is observed that the bounds are off from each other by

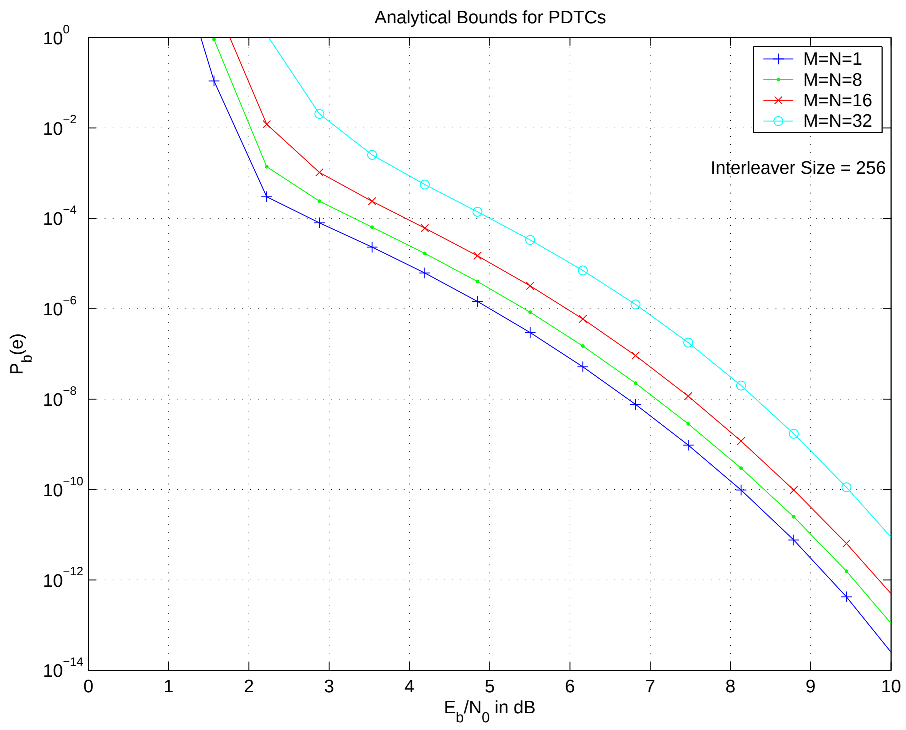


Figure 5.4: Analytical bounds for *PDTCs* using uniform interleaving. Constituent codes are trellis terminated.

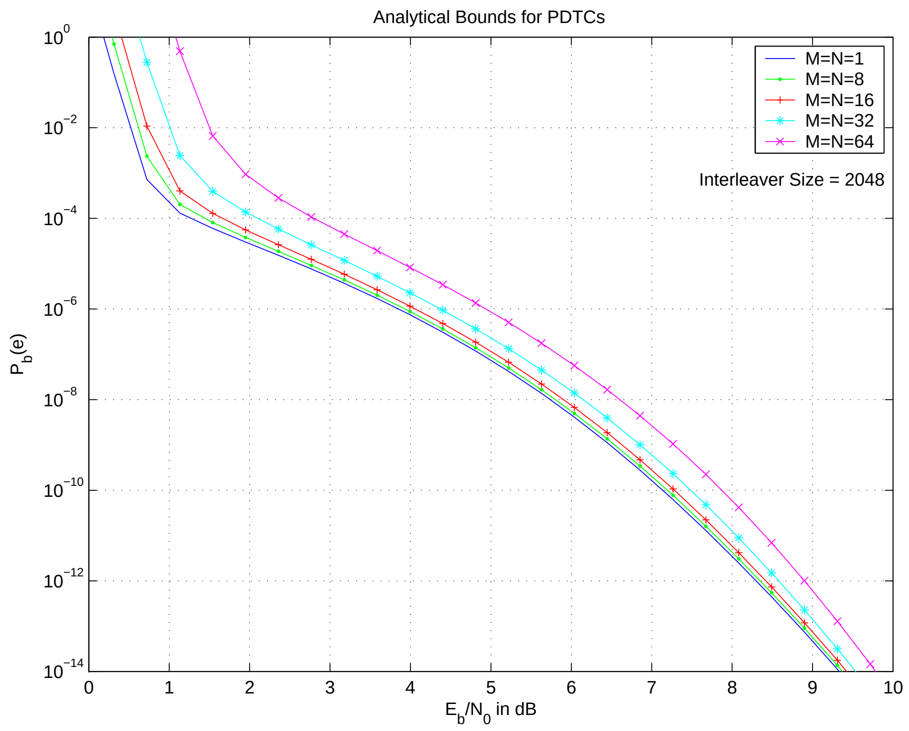


Figure 5.5: Analytical bounds for *PDTCs* using uniform interleaving. Constituent codes are trellis terminated.

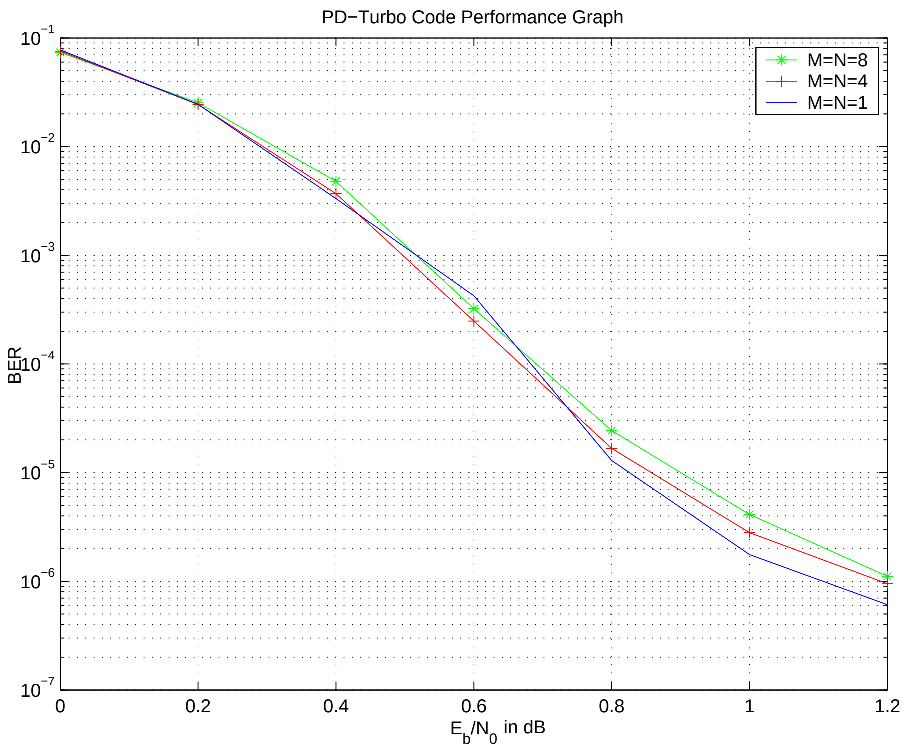


Figure 5.6: *PDTC* performance graph. Interleaver size = 2048. N and M are the number of constituent encoders in upper and lower clusters. *S-random* ($S = 20$) interleaver is used. The curves refer to 12 iterations.

an amount directly related to code rates. This suggests that the parallelized codes have similar *WEFs* and the rise in error floors is mainly due to rate loss stemming from trellis termination. The rise in error floor is more visible for large values of M and N but diminishes as interleaver size increases. Hence, we can choose larger M and N values with larger interleaver sizes and gain more in decoding delay. Simulation results of the proposed system is depicted in Fig. 5.6. The log-MAP algorithm is used for simulations. It is seen from Fig. 5.6 that the *PDTC* performance is almost the same as that of the classical turbo code along with a slight rise in error floor predicted analytically.

5.5 Some Instances of *PDTCs*

There are some instances of *PDTCs* in literature. We present below two instances of *PDTCs* which are woven turbo codes and convolutional coupled codes.

5.5.1 Woven turbo codes

Woven turbo codes [18] are an instance of the proposed general structure where convolutional codes are employed for the upper and lower cluster codes and a rectangular interleaver is used. A single encoder in outer cluster is used. The woven turbo encoder is illustrated in Fig. 5.7.

5.5.2 Convolutional coupled codes

The encoding operation of the convolutional coupled codes [17] is explained as follows. Data is put into a matrix. Rows are encoded using a *RSCs* and convolutional parity matrix is obtained. Next, rows are interleaved separately, and columns are encoded using a block encoder and block parity matrix is formed. The parity bits in convolutional and block parity matrices are multiplexed and transmitted. The convolutional coupled encoding operation is depicted in Fig. 5.8. It is clear form Fig. 5.8 that the convolutional coupled

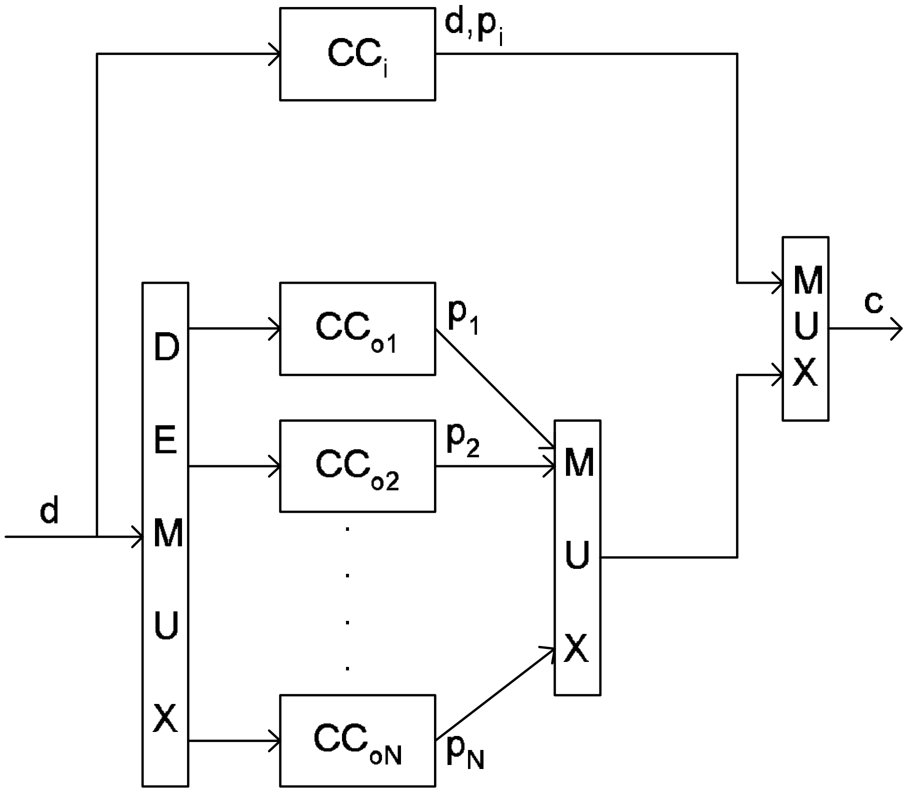


Figure 5.7: Woven Turbo Encoding Operation.

codes are an instance of the proposed general system.

5.6 Further Analysis of *PDTCs*

In this section we inspect the minimum distance (d_{min}) of the parallel decodable turbo codes (*PDTC*). We give bound expressions for d_{min} of *PDTC*. The effect of the puncturing on d_{min} is also investigated. Different interleavers will be tried, and the best one will be emphasized.

5.6.1 Minimum Distance of *PDTC* with Rectangular Interleavers

Minimum distance of the *PDTCs* with rectangular interleaver is found as follows. The encoding operation with rectangular interleaver is given in Fig. 5.9 using matrix notation and it is graphically illustrated in 5.10. Rows are encoded first which corresponds to encoding in the upper cluster. Parity bits

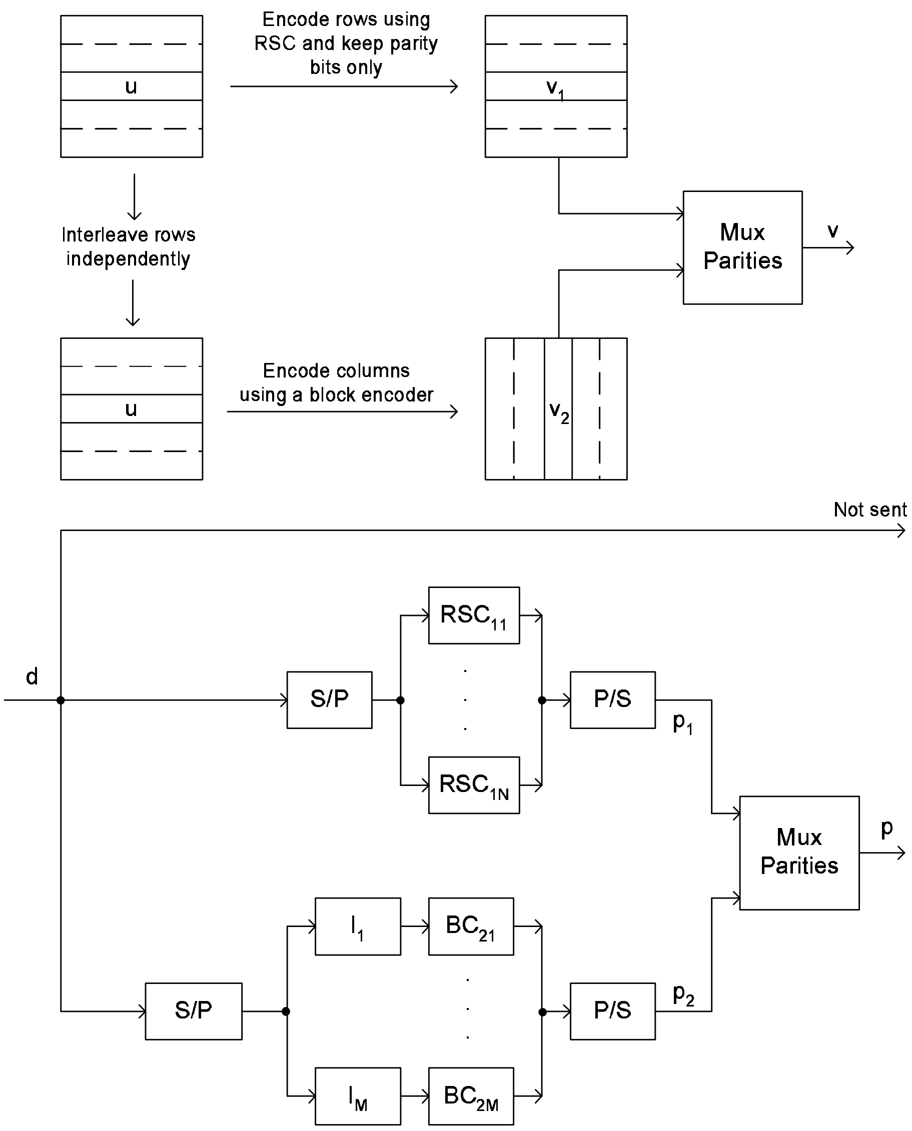

Encode rows using RSC and keep parity bits only
u
v1
Interleave rows independently
Mux Parities
v
Encode columns using a block encoder
u
v2
Not sent
d
S/P
RSC11
RSC1N
P/S
p1
Mux Parities
p
I1
BC21
S/P
P/S
p2
IM
BC2M

Figure 5.8: Convolutional Coupled Codes Encoding Operation. $I_1 \cdots I_M$ are interleavers. $BC_{21} \cdots BC_{2M}$ are the constituent block codes.

are left intact and only data bits are encoded in the vertical direction which corresponds to the coding at the lower cluster of a *PDTC* encoder.

Assume that codewords of Hamming weight d_{free} are generated from input sequences of Hamming weights w_{free} in the constituent convolutional code. Consider a single row of the data matrix whose elements are non-zero and its Hamming weight equals w_{free}. After the first encoding operation we have a row vector of Hamming weight d_{free}. The data bits have a Hamming weight w_{free}, hence the parity bits have a Hamming weight of $d_{free} - w_{free}$. When the second stage of encoding operation is performed (column encoding), we have w_{free} number of columns each of whose Hamming weight is at least d_{free}. Hence the total Hamming weight of the encoded matrix is (minimum distance):

$$d_{min} = w_{free} \times d_{free} + d_{free} - w_{free}. \tag{5.8}$$

For systematic codes $w_{free} \leq d_{free}$. When used in (5.8), the following upper bound is obtained for d_{min} of the *PDTCs*

$$d_{min} \leq d_{free}^2. \tag{5.9}$$

For *RSCs* a nonzero input sequence contains at least two '1's. Hence, for the case $w_{free} = 2$ which corresponds to the lowest possible Hamming weight of the input sequence we can determine the following bound for d_{min}

$$3 \times d_{free} - 2 \leq d_{min} \leq d_{free}^2. \tag{5.10}$$

5.6.2 Minimum Distance of *PDTC* with *S-random* Interleavers

The encoding operation when an *S-random* interleaver is used is depicted in Fig. 5.11 for the case that there exists at least one row with Hamming weight w_{free}. After the first encoding operation (upper cluster encoding), we have at least one row whose Hamming distance is d_{free}. The parity bits have a

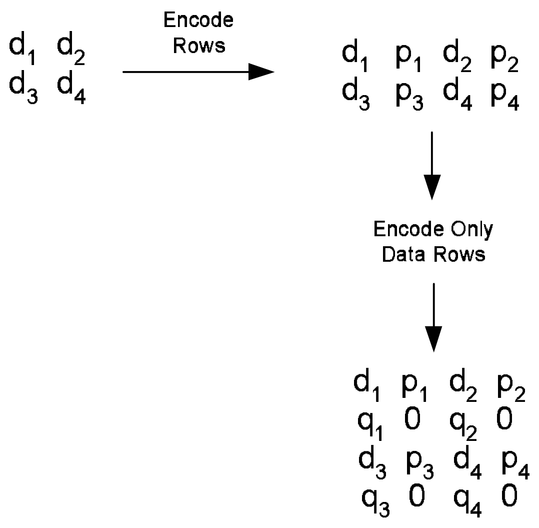


Figure 5.9: *PDTC* encoding operation matrix illustration. Rectangular interleaver is employed.

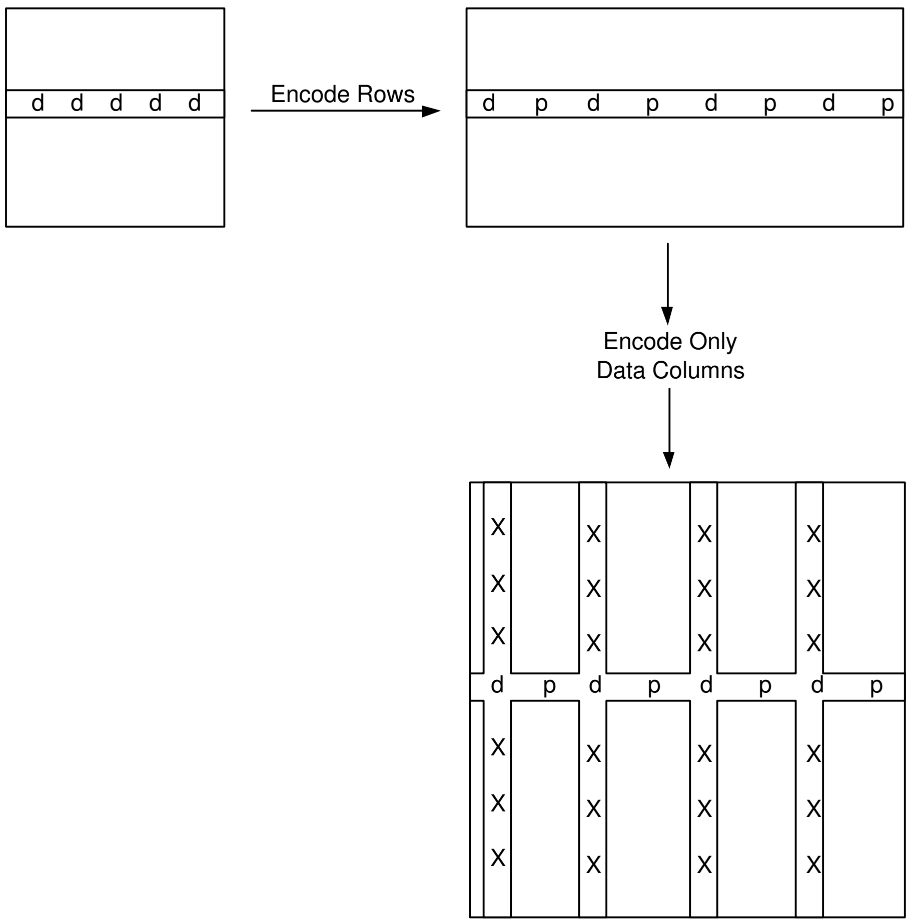


Figure 5.10: Free distance computation of *PDTCs* when rectangular interleaver is employed.

Hamming weight of $d_{free} - w_{free}$. When the data bits are *S-random* interleaved, it is possible that all the nonzero data bits goes to the first column. This corresponds to the worst case in terms of minimum distance. The data bits at the columns are encoded next, this corresponds to the encoding procedure at the lower side of the *PDTC* system. The first column at least has a Hamming weight of d_{free}. The total Hamming weight of the matrix is the sum of the Hamming weight of the first column and Hamming weight of the parity bits which was $d_{free} - w_{free}$. Hence, the worst case d_{min} of the *PDTC* equals $d_{free} - w_{free} + d_{free}$, i.e., $d_{min} = 2 \times d_{free} - w_{free}$. For systematic codes $w_{free} \leq d_{free}$. When it is used in $d_{min} = 2 \times d_{free} - w_{free}$ we obtain lower bound $d_{free} \leq d_{min}$. Considering the case $w_{free} = 2$, we get the bound $d_{free} \leq d_{min} \leq 2d_{free} - 2$. To obtain higher minimum distances for the *PDTC* row-column *S-random* interleaver can be used.

5.6.3 Minimum Distance of *PDTC* with Row-Column *S-random* Interleaver

To maximize the minimum distance of the *PDTC* and to get the maximum benefit from the interleaving gain, we will make use of row-column *S-random* interleavers. The encoding operation of *PDTCs* using *RCS-random* interleaver is as follows. Data is put into a matrix, each row of the matrix is encoded by *RSCs.* This corresponds to encoding operation in the upper cluster. The encoded data is sent to an *RCS-random* interleaver. However, during the interleaving operation only the data bits are involved, i.e., the places of parity bits are fixed and they are left intact. The use of *RCS-random* interleaver guarantees that there are at least w_{free} number of columns containing at least a single '1' in data matrix. When the encoding is performed along the vertical dimension, it is clear that the number of '1's in the matrix structure is at least

$$d_{min} = w_{free} \times d_{free} + d_{free} - w_{free}. \tag{5.11}$$

which is the worst case d_{min} of the *PDTC.* In our study we used constituent

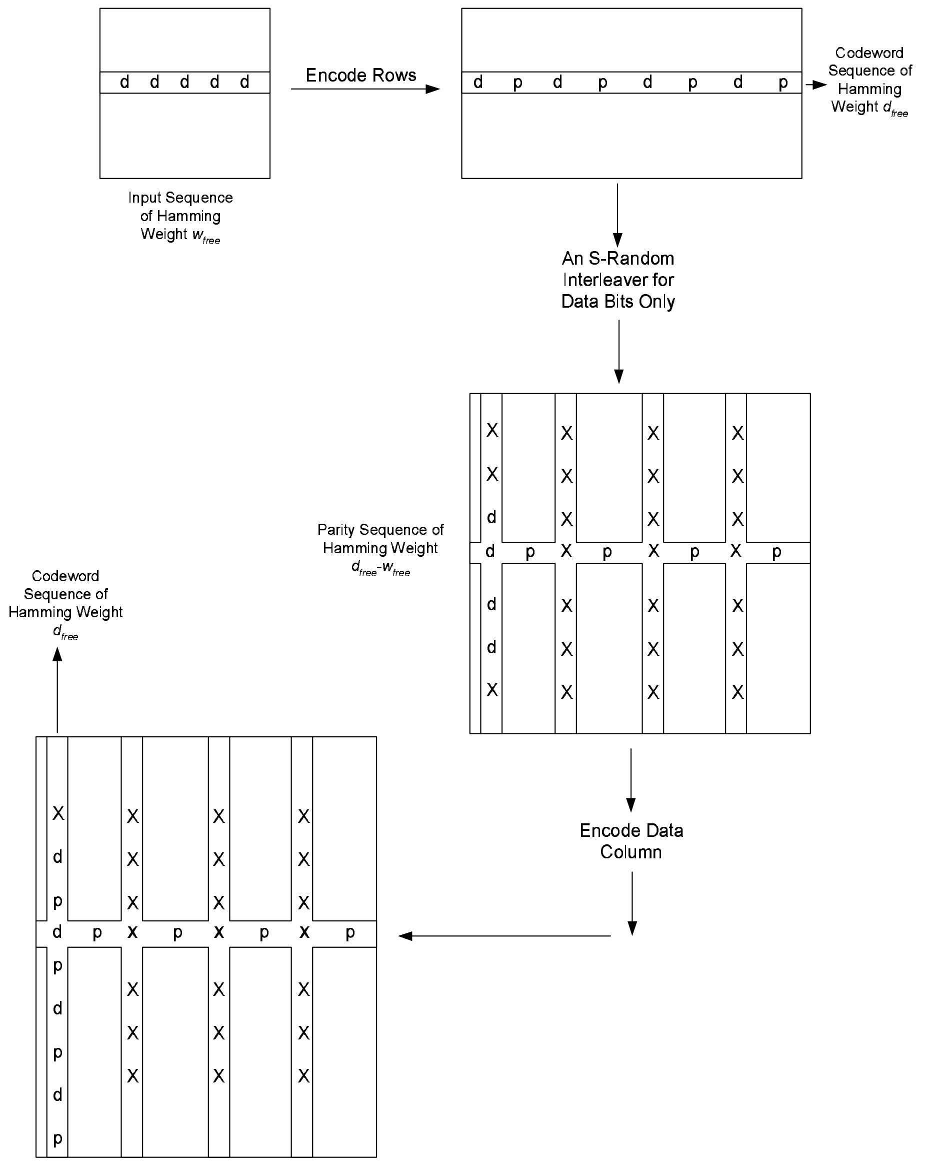

d d d d d
Input Sequence of Hamming Weight w_{free}
Encode Rows
d p d p d p d p
Codeword Sequence of Hamming Weight d_{free}
An S-Random Interleaver for Data Bits Only
Parity Sequence of Hamming Weight d_{free}-w_{free}
Encode Data Column
Codeword Sequence of Hamming Weight d_{free}

Figure 5.11: *PDTC* Encoding operation matrix illustration. An *S-random* interleaver is used.

convolutional codes with the generator matrix $(1, 5/7)_{octal}$. The free distance of this convolutional code is $d_{free} = 5$. The Hamming weight of the input sequences that produce d_{free} Hamming weight codewords is $w_{free} = 3$. Putting these values into the eqn. (5.11), we get $d_{min} = 17$, which is the worst case minimum distance of the *PDTC* when *RCS-random* interleaver is employed. Parallel decodable serially concatenated codes are studied in Chapter 5. Considering the rate 1/3 parallel decodable serially concatenated convolutional systems whose d_{min} equals 15 when row-column *S-random* interleaver is used, *PDTC* system has a higher minimum distance. Hence, it is expected that *PDTC* system has an error floor at higher *SNR* values compared to the punctured *PDSCCCs*. *PDTC* encoding operation using the *RCS-random* interleaver is depicted in Fig. 5.12.

5.6.4 Memory Collision Problem

When parallel decodable turbo codes (*PDTCs*) are concerned, memory collision does not occur if after row-column *S-random* interleaving operation the encoding is performed row-wise by the lower cluster encoders. However, in this case worst case minimum distance of the *PDTC* is not guaranteed to be equal to $d_{min} = w_{free} \times d_{free} + d_{free} - w_{free}$. If columns of the interleaved data matrix are encoded by lower cluster encoders, memory collision occurs but minimum distance of *PDTC* has its greatest value. This is clearly seen from simulation results in Fig.5.13 where *PDTC* with MC shows better performance due to guaranteed higher minimum distance. However, during the *RCS-random* interleaving operation, if completely distinct random interleavers are used for each column, minimum distance of the *PDTC* again gets is maximum value $w_{free} \times d_{free} + d_{free} - w_{free}$. However, for short frame lengths it may not be possible to find required number of completely different interleavers. The performance of the *PDTC* with a collision free interleaver is depicted in Fig. 5.13.

Although, *PDTC* with MC shows better performance for M=32 and N=32,

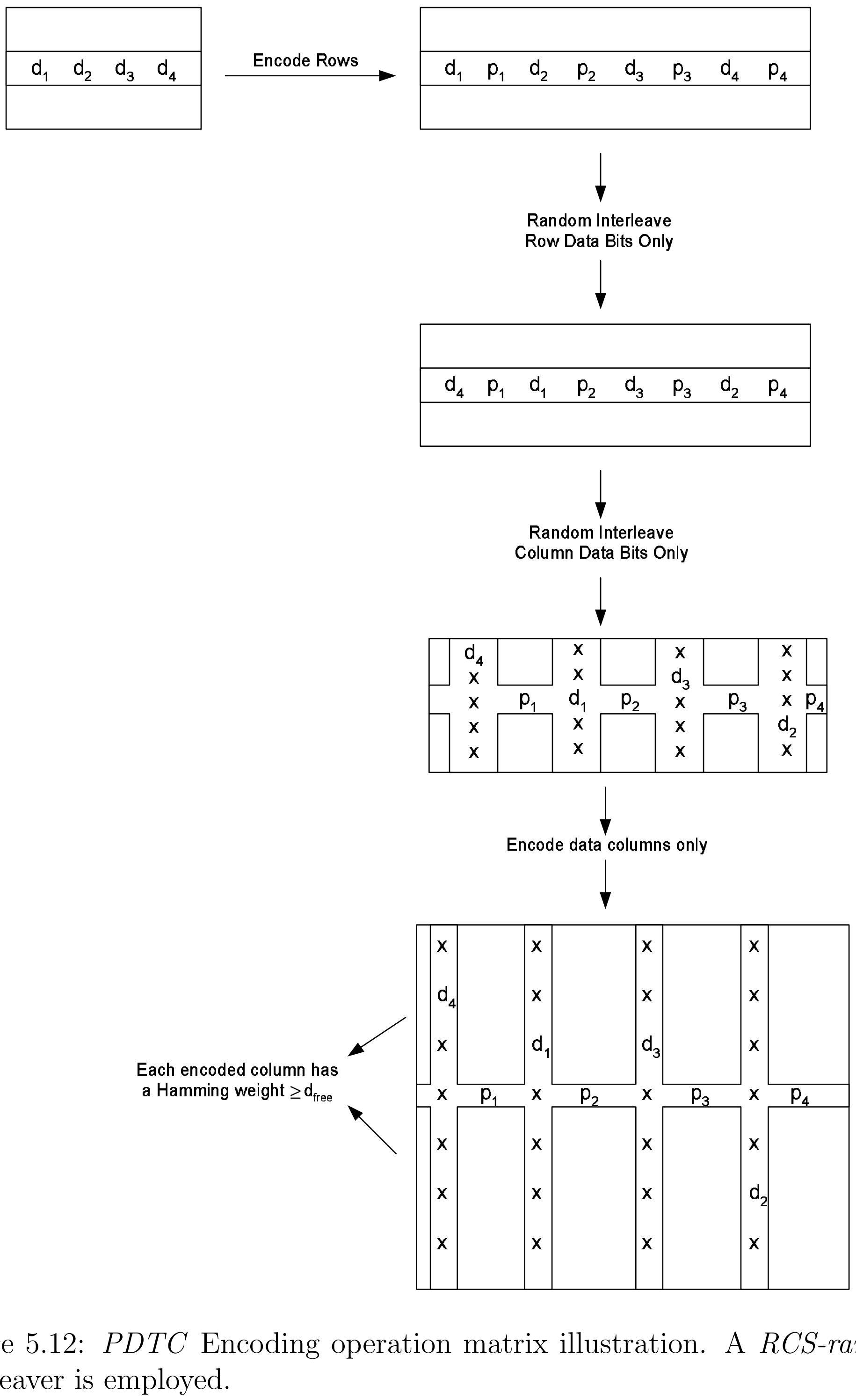

Encode Rows
Random Interleave Row Data Bits Only
Random Interleave Column Data Bits Only
Encode data columns only
Each encoded column has a Hamming weight ≥ d_free
d1 d2 d3 d4
d1 p1 d2 p2 d3 p3 d4 p4
d4 p1 d1 p2 d3 p3 d2 p4

Figure 5.12: *PDTC* Encoding operation matrix illustration. A *RCS-random* interleaver is employed.

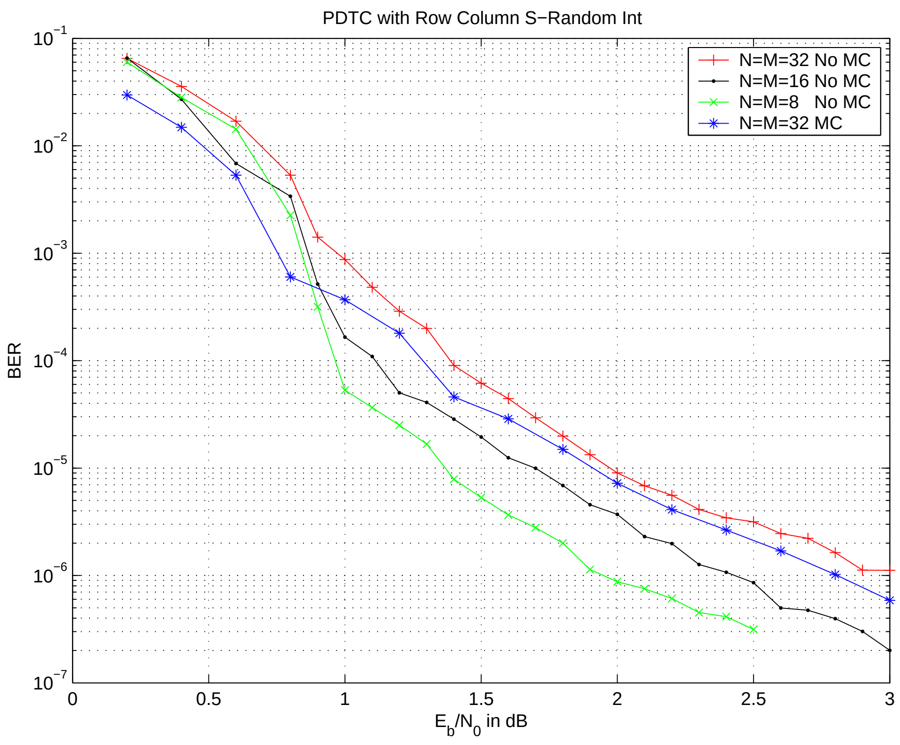


Figure 5.13: *PDTC* performance graph. *RCS-random* interleaver is employed. MC refers to an interleaver with memory collision

as the number of lower cluster encoder increases (number of columns increases), the performance of the *PDTC* degrades in spite of large free distance. This is due to the shorter frame lengths of the lower cluster encoders and large rate loss due to trellis termination bits for a great number of encoders. Performance of *PDTC* for a large number of column encoders is illustrated in Fig. 5.14. Changing the size of the matrix does not affect worst case minimum distance of the code, this is clear from Fig. 5.14, since all the lines almost have the same slope. The slope of the performance lines depends on the minimum distance of the code.

The effect of the minimum distance on the performance of *PDTC* is better seen when N=32, M=32, i.e., frame length equals 1024. The performance graph is depicted in Fig. 5.15. It is clear that *PDTCs* with *RCS-random* interleaver with memory collision show better performance.

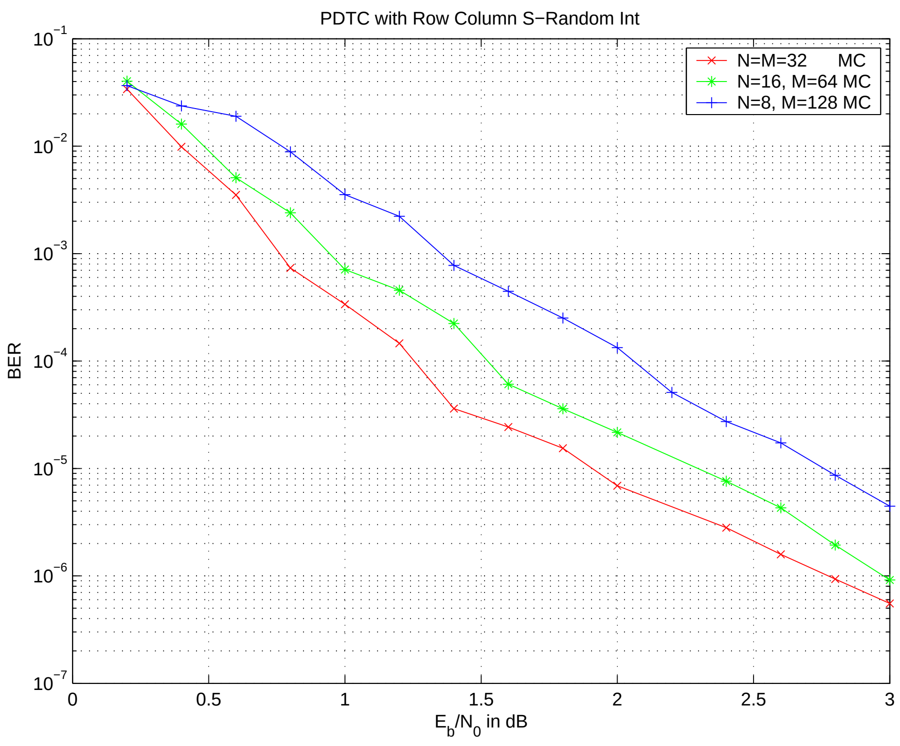


Figure 5.14: *PDTC* performance graph. *RCS-random* interleaver is employed. MC means memory collision

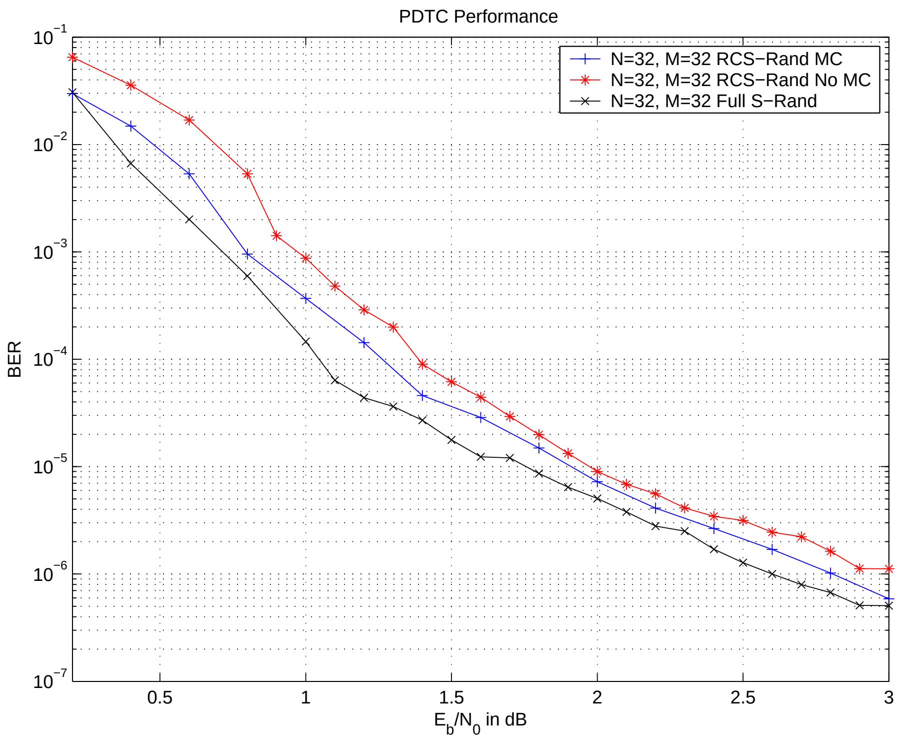


Figure 5.15: *PDTC* performance graph for *RCS-random* and *S-random* interleavers. MC means memory collision

5.7 Increasing Code Rate by Puncturing

The d_{min} of *PDTC* will be evaluated here when puncturing is applied. Assume the use of the puncturing pattern $v = [1110]$, i.e., every second parity bit is eliminated. Assuming the use of rectangular or *RCS-random* interleaver, the minimum distance of the *PDTC* is computed as

$$d_{min} = w_{free} \cdot d^p_{free} + d^p_{free} - w_{free} \tag{5.12}$$

where d^p_{free} is the free distance of the convolutional code after puncturing operation. For the convolutional code with generator polynomial $(1, 5/7)_{octal}$, the use of the puncturing pattern $v = [1110]$ results in $d^p_{free} = 3$ and $w_{free} = 3$. Using eq. (5.12) the minimum distance of the punctured *PDTC* is found as 9. The simulation results for punctured *PDTC* is depicted in Fig. 5.16. *RCS-random* interleaver is employed for the simulation. As it is clear from the Fig. 5.16 that *PDTC* with memory collision has a slightly better performance than *PDTC* without memory collision. This is due to the guaranteed largest minimum distance of *PDTC* given in eqn. 5.11.

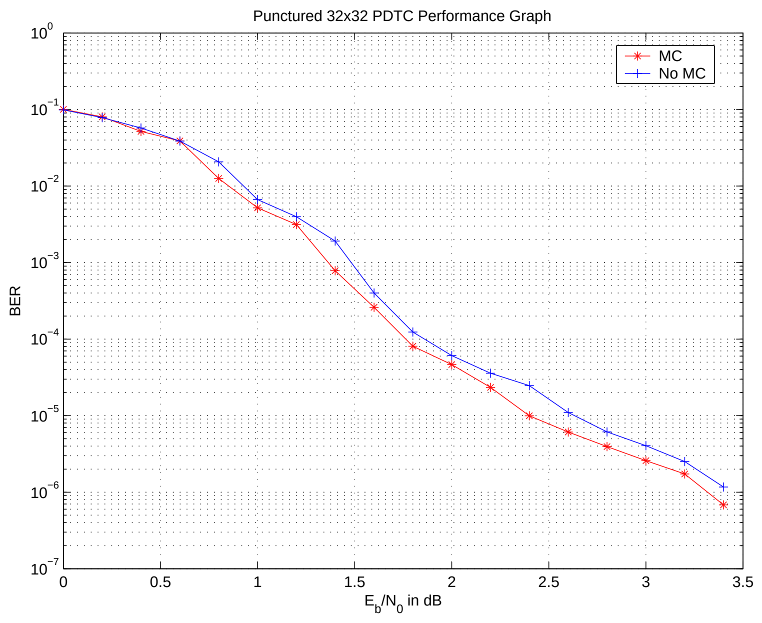


Figure 5.16: Punctured PDTC performance graph with *RCS-random* interleaver. Frame length =1024. Rate is 0.47. Iteration number is 12.

CHAPTER 6

SOME APPLICATIONS OF THE PROPOSED PARALLELIZATION TECHNIQUES

The proposed parallelized structures can be applied in many different communication scenarios. We will present a few exemplary systems to show the generality of the proposed methodology.

6.1 Turbo Equalization

Communication systems suffer from inter-symbol interference (ISI) in frequency selective channels. A channel equalizer is usually employed at the receiver side to estimate the data and alleviate the effect of ISI. Traditional equalizers are usually linear filters for which various criteria are used for their performance, such as minimum mean square error (MMSE). Zero forcing equalizer (ZFE)[60] is the simplest one which is achieved by just inverting the channel.

With the invention of the turbo codes in 1993 [13] iterative algorithms gained much popularity among communication society. The iterative approach was extended to channel equalization in [15] where turbo equalizer was introduced.

Turbo equalization is the process of combining channel equalization and decoding operations in an iterative manner. The information produced in the decoders and equalizers are traded back and forth to enhance the overall performance. A turbo equalizer (TE) system usually employs soft-input, soft output (SISO) algorithms at the decoder and equalizer block. The popular SISO algorithms employed are the marginal a posteriori (MAP), and Viterbi based algorithms which are sequential in nature. However, sequential solutions such as MAP and soft output Viterbi suffer from high complexity and high latency which is a limiting factor in practical applications in spite of their good performances.

To reduce complexity and the large latency, a number of algorithms to use at the equalizer block have been suggested in [61] where minimum mean square error (MMSE) equalizer is used at the equalizer block. However, its performance degrades seriously in severe ISI cases [62]. In [63], a least mean squares based update algorithm is used for the determination of the linear filter coefficients. A joint structure that combines coding and equalization different than turbo equalization is suggested in [64] where decision feedback equalizer soft information is exchanged with the decoder hard decisions in an iterative manner. Although the use of sub-optimum algorithms result in reduced complexity, the performance loss cannot be tolerable for some cases [62].

In our study, we introduce zero state doped turbo equalizer *(ZSDTE)* structure which uses trellis based optimal decoding algorithms and is very suitable for parallel processing at the receiver side. We use zero state trellis termination method which permits parallelization of the turbo equalizer. ZSDTE has significantly low latency compared to the classical turbo equalizer structure and at the same time exhibits the same performance.

6.1.1 System Model

The transmitter and receiver sides of a communication system that sends data through a frequency selective channel is depicted in Figs. 6.1 and 6.2. The channel is also included into the transmitter side for ease of representation.

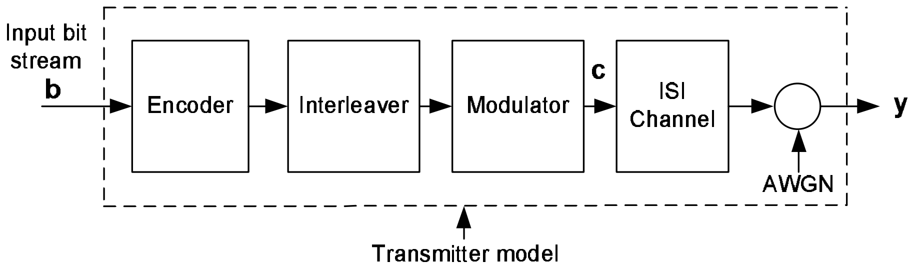


Figure 6.1: Transmitter side of a communication system.

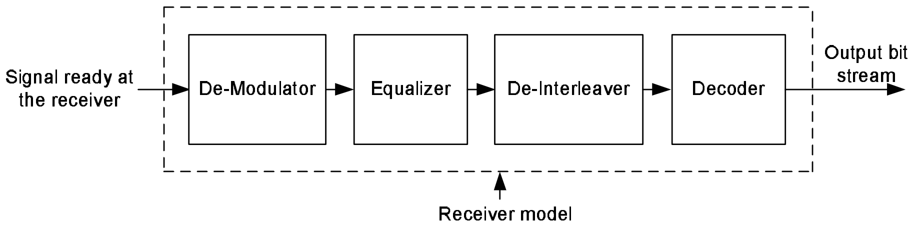


Figure 6.2: Receiver side of a communication system.

In the transmitter model, the transmit filter, the channel, and the receive filter are incorporated into the ISI channel model which is represented by a discrete time linear transversal filter with the finite length impulse response given by [60]

$$h[n] = \sum_{k=0}^{L_f-1} h_k \delta[n-k], \tag{6.1}$$

where h_k are the filter coefficients and L_f is the filter length. The channel coefficients h_k are time-invariant and available at the receiver. A tapped delay line model for the frequency selective channel with three channel taps is depicted in Fig. 6.3. The received signal samples y_k can be expressed in terms of channel coefficients and input symbols by

$$y[k] = n[k] + h[k] * c[k] \tag{6.2}$$

where $'*'$ denotes the convolution operation, $n[k]$ are the independent and identically distributed Gaussian noise samples with probability density function

$$f(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-n^2/2\sigma^2}. \tag{6.3}$$

We assume that all the symbols in the model are real without loss of generality. The frequency selective channel can be represented using a trellis diagram. For the three tapped ISI channel the trellis representation is depicted in Fig. 6.4 for the channel taps $h_0 = 0.407$, $h_1 = 0.815$ and $h_2 = 0.407$.

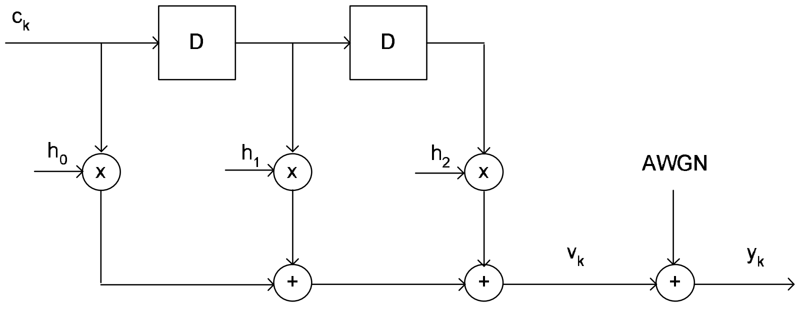


Figure 6.3: Tapped delay line model for three tap frequency selective channel.

To mitigate the effects of the ISI at the receiver, decoding and equalization operations are combined such that soft information exchange among blocks occurs. This iterative structure is called the turbo equalizer which is depicted in Fig. 6.5. Although we have used MAP algorithm for both the equalizer and decoder blocks in Fig. 6.5 any SISO algorithm can be adopted for both blocks.

6.1.2 Zero State Doped Turbo Equalizer *(ZSDTE)*

By parallel processing high latency caused by decoder and equalizer modules can be reduced. To enable the parallel processing operation, we will partition trellis structure of the encoder and equalizer modules, i.e., we will obtain parallel trellis structures. This is achieved by the addition of trellis termination bits to the binary information bits in a periodic manner so that the trellis diagram of the convolutional encoder visits the zero state periodically. The same procedure is repeated for the ISI channel input symbol sequence, i.e., trellis termination symbols are inserted in a periodic manner. Hence, the ISI channel trellis is also periodically terminated. This procedure is illustrated in Fig. 6.6. Instead of the trellis termination bits, tail biting can be used in the convolutional encoder. Using tail biting it is guaranteed that the starting and ending states are the same [65]. However, the insertion of the trellis

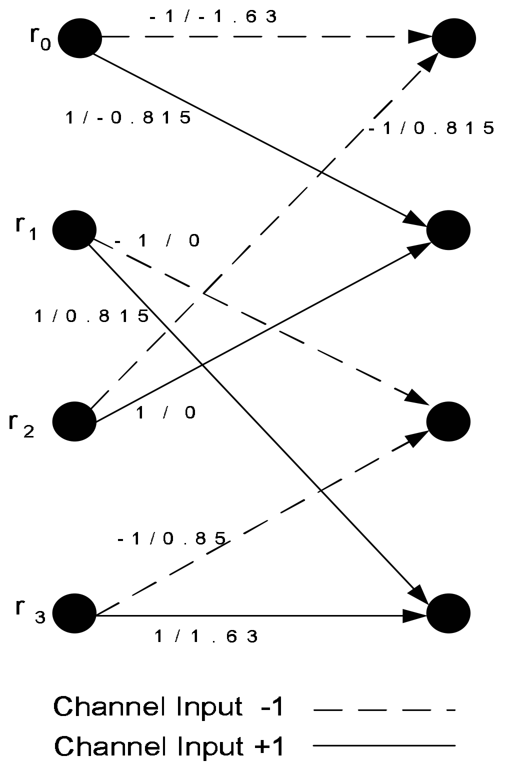


Figure 6.4: Trellis representation for a three tap frequency selective channel. The possible content of the delay elements $r_0 = (1, 1)$, $r_1 = (-1, 1)$, $r_2 = (1, -1)$, $r_3 = (-1, -1)$ are the states. The transitions from a state S_t at time t to another state S_{t+1} at time $t + 1$ occurs according to the input and output pair v_t/y_t where v_t is the channel input and y_t is the channel output.

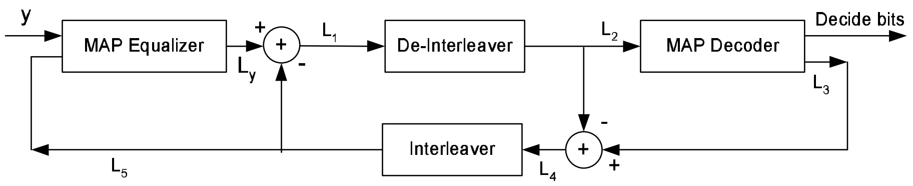


Figure 6.5: Turbo equalizer. y is the received signal. $\mathrm{L_y}$, $\mathrm{L_1}$, $\mathrm{L_2}$, $\mathrm{L_3}$, $\mathrm{L_4}$ and $\mathrm{L_5}$ denote log-likelihoods.

termination symbols for the channel part is a must, since by employing tail biting symbols for the ISI channel trellis it is impossible to obtain parallelized trellis structure.

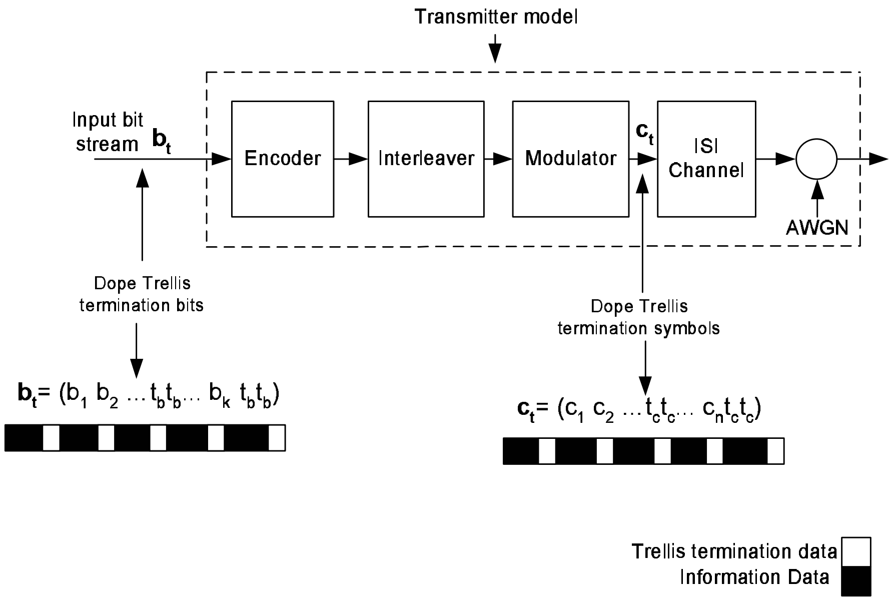

Transmitter model
Input bit stream $\mathbf{b}_t$
Encoder
Interleaver
Modulator
$\mathbf{c}_t$
ISI Channel
AWGN
y
Dope Trellis termination bits
Dope Trellis termination symbols
$\mathbf{b}_t = (b_1\ b_2 \ldots t_b t_b \ldots b_k\ t_b t_b)$
$\mathbf{c}_t = (c_1\ c_2 \ldots t_c t_c \ldots c_n t_c t_c)$
Trellis termination data
Information Data

Figure 6.6: Zero state doping process at the transmitter side. $\mathrm{t_b}$, $\mathrm{t_c}$ are the trellis termination bits and symbols respectively.

This approach can also be interpreted as pre-coding. The binary information sequence is pre-coded and then passed through a convolutional encoder. The encoded binary data is binary phase shift keying (BPSK) modulated. The BPSK symbol sequences are pre-coded and passed through an ISI channel. White Gaussian noise with double sideband power spectral density $N_0/2$ is added to the channel output. The pre-coding effectively divides the incoming sequence of blocks into smaller blocks which can be processed separately by multiple equalizers and decoders in a parallel manner at the receiver side. To clarify the pre-coding operation further, an equivalent representation that better illustrates the overall operation in Fig. 6.6 is explained in the following section.

6.1.3 Transmitter Side Equivalent Model

The zero ZSDTE model in Fig. 6.6 may seem complex in the first look. However, the same system can be expressed with a simplified model. This model is depicted in Fig. 6.7. As it is seen from the Fig. 6.7, serial to parallel (S/P) and parallel to serial (P/S) converters are employed. In fact, if we employ multiplexers instead of S/P converters and de-multiplexers for the place of P/S converters both structures become exactly the same. S/P and P/S converters are nothing but multiplexers and de-multiplexers preceded by rectangular interleavers. All of the outer constituent encoders add trellis termination bits to the coded data sequences. Inner ISI channel trellises are also terminated. This transmitter side equivalent model enables us for parallel decoding and equalization operations at the receiver side. The structure in Fig. 6.7 has the same form as the parallelized codes in chapters 5 and 6.

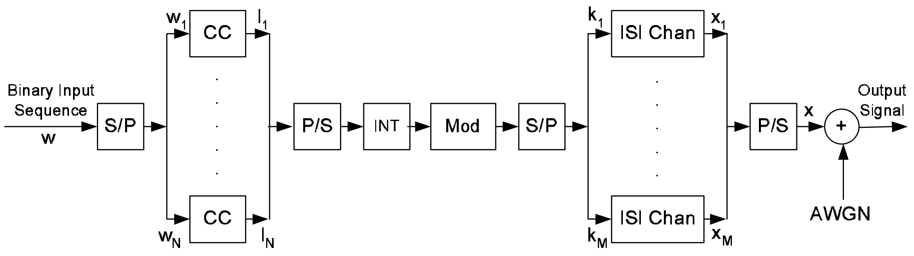


Figure 6.7: Parallel equivalent model.

Parallel decoding and equalization can be performed similar to *PDSCCCs* as detailed in chapter 4. The decoding latency is decreased by a factor of number of constituent elements in clusters assuming that equal number of branches are used in clusters. The memory collision problem can be avoided with row-column *S-random* interleavers.

6.1.4 Simulation Results

We use *RSC* $(1, 5/7)_{\text{octal}}$ for the outer code. An *S-random* ($S = 18$) interleaver is employed between convolutional encoder and the BPSK modulator. The three tap ISI channel model in [60] is adopted. The channel coefficients are $h_0 = 0.407$, $h_1 = 0.815$ and $h_2 = 0.407$. Both ISI channel and outer convolutional code has memory 2, i.e., the number of states in trellis structures are the same. The log-MAP algorithm is used for both the decoder and equalizer blocks. Trellis termination bits were taken into account while computing the overall rate and thus the energy per bit. Input information frame length is chosen as 1024 bits. Twelve iterations are performed for each frame. For statistical significance, the simulations were run until at least 60 erroneously encoded frames have been received. The number of outer and inner branches were chosen equal, i.e., N=M=1, N=M=4, N=M=16. The asymmetric case was also considered, i.e., N value changes and M=1 fixed. We also simulated turbo equalizer using the sliding window method in [40]. The performance graphs are shown in Figs. 6.8, 6.9, and 6.10.

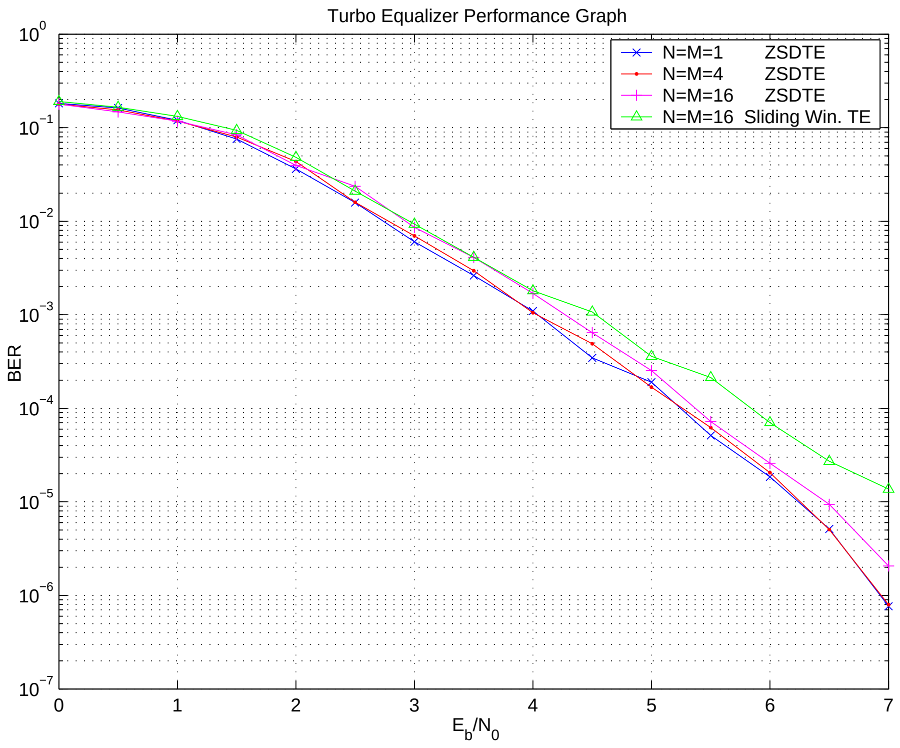


Figure 6.8: Bit error rate (BER) performance of ZSDTE and sliding window turbo equalizer (SWTE). N=M=1 corresponds to the classical turbo equalizer.

As seen from Figs. 6.8 and 6.9, ZSDTE shows very close performance to that of the classical turbo equalizer for even large N and M values, i.e., N=M=16. As the number of branches increases, BER and FER performance

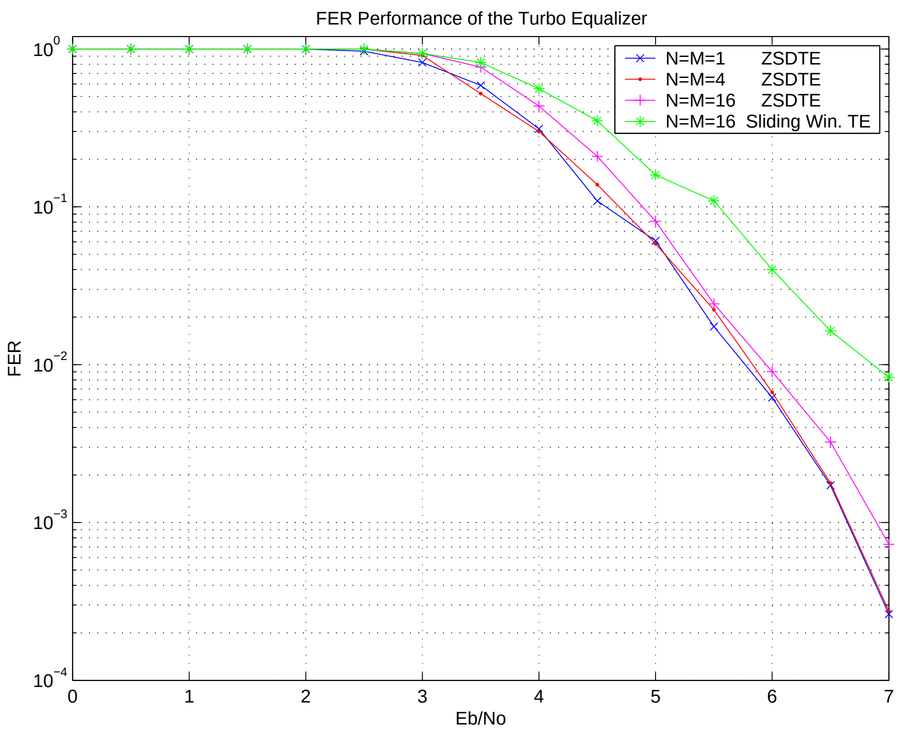


Figure 6.9: Frame error rate performance of the ZSDTE and SWTE. N=M=1 corresponds to the classical turbo equalizer.

degrades slightly. As it is obvious from the Figs. ZS doping method shows better performance than the sliding window technique. The simulation results for the asymmetric situation is depicted in Fig. 6.10. It is seen that the performance with ZSDTE is quite comparable to that of classical turbo equalizer.

6.2 Joint Structures for Trellis Coded Modulation, Convolutional Product Codes and Space Time Trellis Codes

By employing joint structures it is possible to obtain better communications systems. Those communication units employing convolutional codes can be jointly designed. Some of those communication schemes employing trellis codes are trellis coded modulation and space time trellis coding. *CPC* which is a

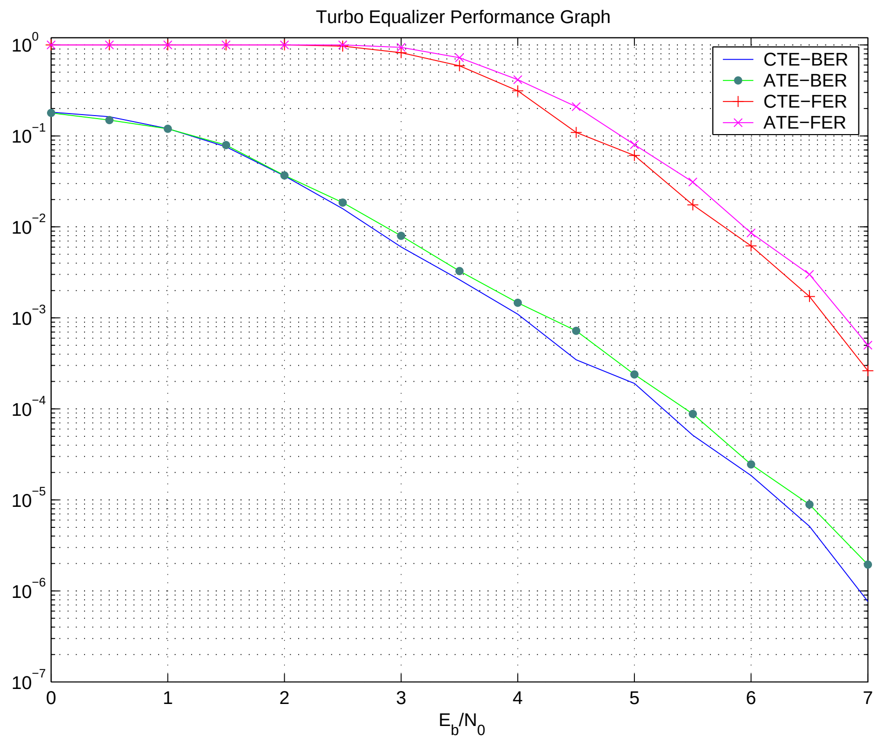


Figure 6.10: Error rates for the ZSDTE for the asymmetric case, i.e., N=16, M=1. ATE and CTE mean asymmetric and classical turbo equalizer respectively.

special case of *PDSCCC* can be incorporated with trellis coded modulation and with space time trellis coding. Since *CPC* has a matrix structure and enables parallel processing operations, better jointly designed communication systems with low latency can be obtained. In this section joint structure for *CPC* involving both space time trellis codes and convolutional product codes will be introduced. We first provide some background information and then introduce our joint structures.

6.2.1 Trellis Coded Modulation

For error free communication the use of channel codes before transmission is a must. Block and convolutional code provide coding gain through the insertion of redundant bits. Redundancy is measured in terms of the code rate R, which is the ratio of the information bits to the total number of transmitted bits. The addition of the redundancy bits causes code rate to decrease. If the information transmission rate is to be the same, then symbol transmission rate for the coded system should be increased by R^{-1}. This means an increased bandwidth. In a bandwidth-limited system this is a severe problem.

Modulation and coding were usually regarded as two separate processes until the discovery of trellis coded modulation in 1976 by Ungerböeck [10]. Ungerböeck used convolutional coding and modulation jointly and showed that it was possible to decrease the probability of error in bandlimited channels by introducing redundancy with the three step method in [10] without increasing the transmission bandwidth. In *TCM* the coding operation is performed in the modulation signal space. The performance of an uncoded modulation scheme depends on the Euclidian distance between signals in the modulation signal space. The purpose of *TCM* is to maximize the free Euclidian distance of the code by properly matching the codewords to signal points in the constellation diagram. Trellis coded modulation was adopted in use for telephone modems and for many satellite communication applications [11]. Trellis coded modulation can be described as below.

- Add p bits of redundancy to every set of m source bits.
- Expand signal constellation from 2^m to 2^{m+p}.
- Use the $m+p$ bits encoded blocks to select from the expanded signal constellation.

Symbol transmission rate is the same. Hence no additional bandwidth is needed. The main idea of Ungerböeck's system lies on the manner by which the m information bits are mapped onto the 2^{m+p} signals in the expanded constellation. This procedure is achieved through set partitioning. The goal is to maximize the Euclidian distance between symbols sequences. In general there are three main stages of trellis coded modulation. Ungerböeck proposed that choosing $p = 1$ is sufficient to achieve the additional performance.

- Signal constellation expansion.
- Signal constellation partitioning.
- Selection of the the partitions by convolutional encoders.

The general procedure for Ungerböeck encoder is shown in Fig. 6.11.The number of information bits in one data symbol is denoted by m where k of the m information bits are encoded using a rate $\frac{k}{k+1}$ convolutional encoder. The resulting $k+1$ bits are used to select one of the 2^{k+1} partitions of the 2^{m+1} signal constellations at the $(k+1)$st level of the constellation's partition tree. The remaining $m-k$ bits are used to select a signal from the chosen partition three. The selection of the signal is determined by the convolutional encoder structure, hence the complete system is referred as trellis coded modulation.

In addition, the spectral efficiency of a system is defined as the number of bits per second transmitted per 1 Hz of bandwidth. A brief explanation is given for each of the three main stages of *TCM* in the following sections.

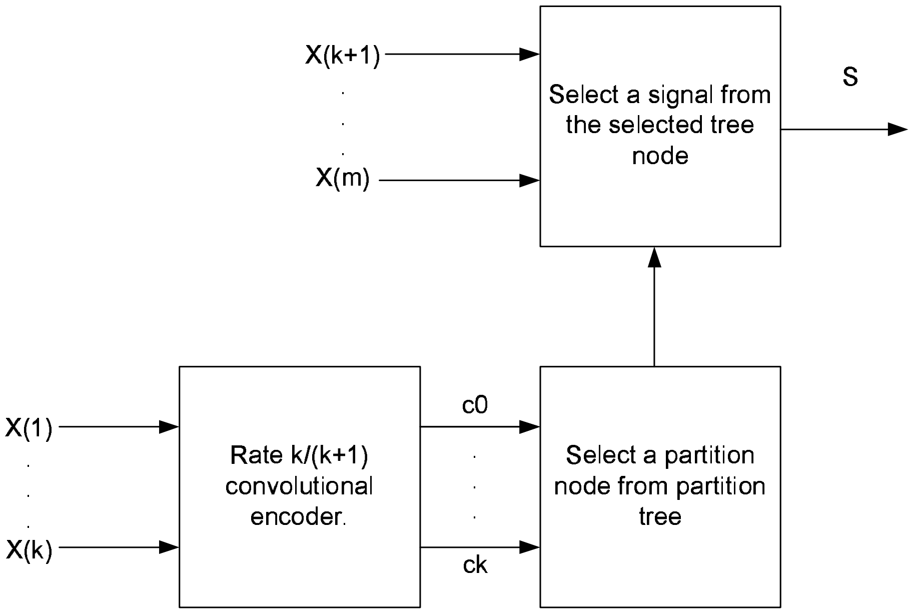

X(k+1)
X(m)
Select a signal from the selected tree node
S
X(1)
X(k)
Rate k/(k+1) convolutional encoder.
c0
ck
Select a partition node from partition tree

Figure 6.11: Trellis coded modulation procedure.

6.2.1.1 Constellations

There are three main constellations used in communication systems.

- One dimensional constellations.
- Rectangular constellations.
- MPSK (M-ary phase shift keying) constellations.

Rectangular constellations provide better minimum distance versus average energy performance, however they are distorted when they pass through non-linear devices. In MPSK although the minimum distance between constellation points is relatively small, the modulated signal has a constant envelope and is not distorted by non-linearities. In Fig. 6.12 4-PSK and 8-PSK constellations are depicted as examples.

6.2.1.2 Set Partitioning

Set partitioning involves dividing the signal constellation into subsets with larger minimum distance between points in the subset than the minimum dis-

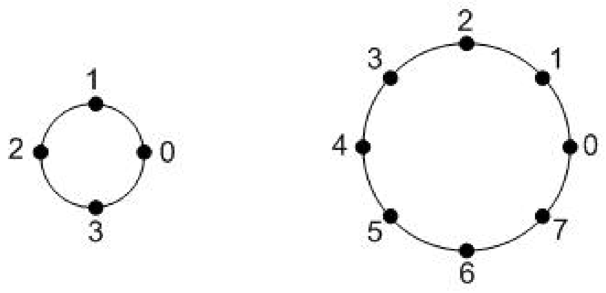


Figure 6.12: 4 PSK and 8 PSK Constellations.

tance between points in the original constellation.

An 8-PSK signal constellation partitioning is illustrated in Fig. 6.13. The partitioning is explained as follows. First constellation A is divided into two subsets B_0 and B_1, then subsets B_0 and B_1 are divided into two subsets C_0, C_1 and C_2, C_3 respectively. At each level of subdivision, the minimum distance between symbols increases compared to the previous level.

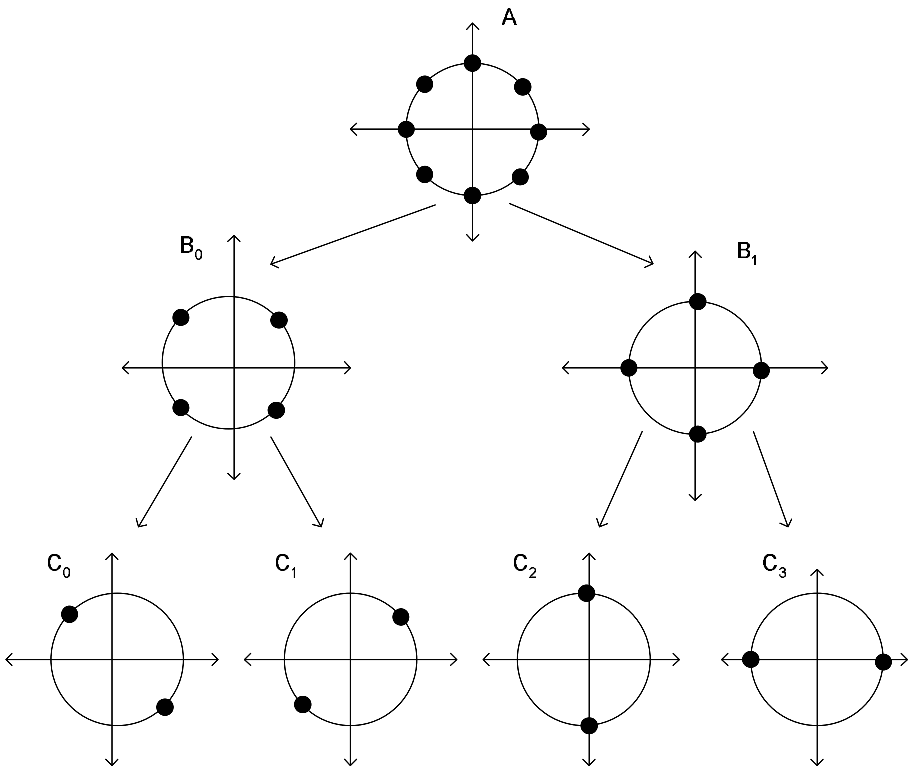


Figure 6.13: Constellation expansion and partitioning of the constellation. Codewords are carefully assigned to the constellation points.

6.2.1.3 Selection of the the partitions by convolutional encoders

A convolutional encoder is used in *TCM*. Convolutional encoder can be systematic or non-systematic. If the convolutional encoder is a systematic encoder, there exists parallel transitions between states in the trellis diagram of the convolutional encoder. If non-systematic code is used then there exists no parallel transitions between states in trellis diagram. While assigning code words to the signal partitions care should be given to maximize the Euclidian distance between parallel transitions. This situation is considered in Ungerböeck's work. Now we state the Ungerböeck design rules,

1. Signals in the same lowest partition region are assigned to the parallel transitions, if parallel transitions exists, this rule maximizes distance between parallel paths.

2. Signals in the preceding partition are assigned to transitions that start or stop in the same state. This specification maximizes distance between non-parallel paths.

3. All signals are used equally often.

For QPSK to 8-PSK expansion we will use the convolutional encoder in Fig. 6.14. Codewords will be mapped to the constellation points considering Ungerböeck's design principles. The assignment of codewords to constellation points is illustrated in Fig. 6.15.

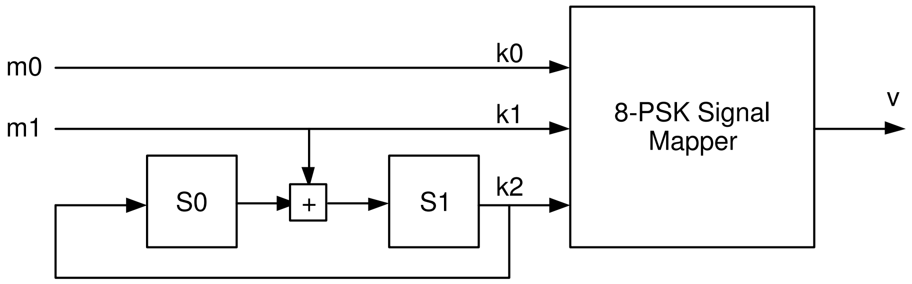


Figure 6.14: Systematic Convolutional Encoder.

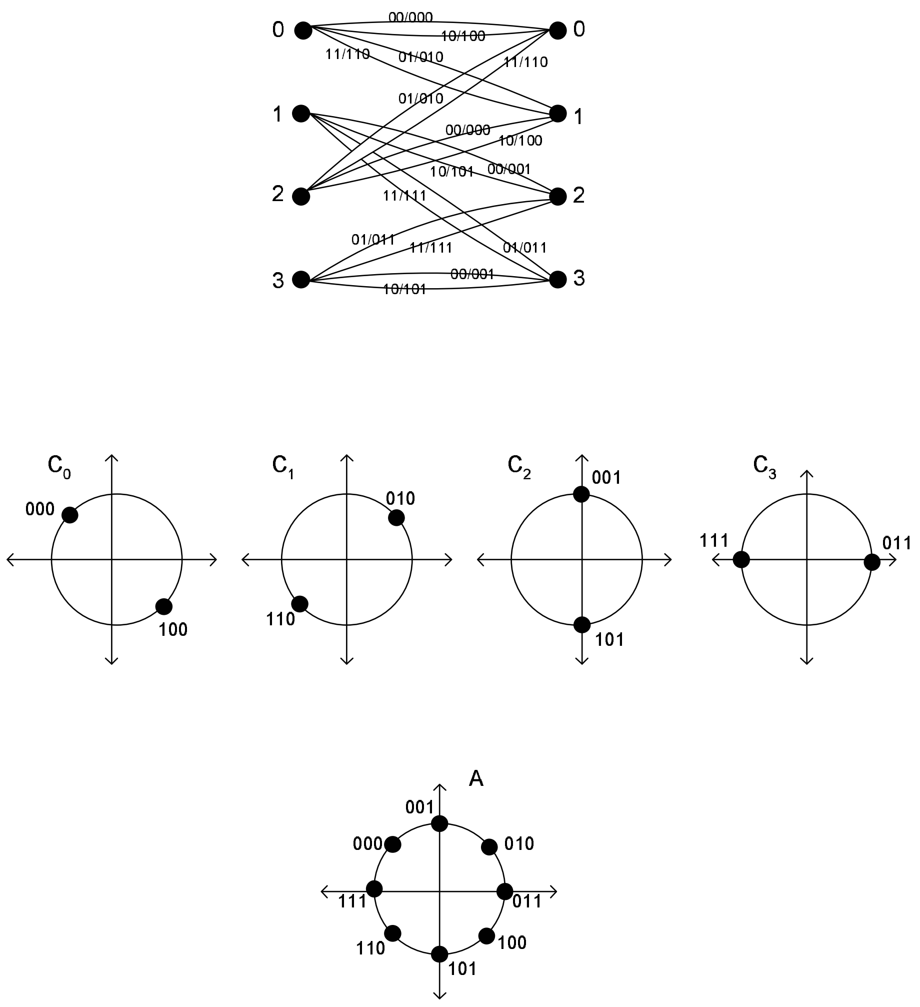


Figure 6.15: State transition diagram of the convolutional encoder in Figure 6.14. Codewords are carefully assigned to the constellation according to Ungerböeck design rules.

Ungerböeck's design principles maximize the minimum free distance of the encoder. The minimum free distance of the encoder is the minimum of free distance between parallel and non-parallel transitions. In other words,

$$d_{free} = min\{d_{free_{parallel}}, d_{free_{non-parallel}}\}.$$

6.2.2 Trellis Coded Modulated *CPC*

In this section we explain the joint structure involving *TCM* and *CPC*. The process at the transmitter side for the joint system is described as follows. Data is put into a matrix. Each row of the data matrix is first encoded using a rate 1/2 recursive systematic convolutional code whose generator polynomial is $(1, 5/7)_{octal}$. Next, column encoding operation proceeds. Each column of the data matrix is encoded using Ungerböck's 8-State *RSC* [66, 67]. The encoding operation is illustrated in Fig. 6.16. Column encoded data is 8-PSK modulated and passed through an AWGN channel. An interleaver can be employed after row decoding if desired. For simplicity of the illustration we have omitted the interleaver in our Figs. The role of the interleaver is so critical for the good performance of the *TMCPC*, hence in our simulations we used an *S-random* interleaver after row encoding operation.

6.2.2.1 *TMCPC* Decoding

TMCPC decoding operation is more complex than classical *CPC* decoding operation. Symbol-wise log-MAP decoders are used for the *TMCPC* decoding. In addition, symbol probability partition (bit probabilities are formed using symbol probabilities) and symbol probability formation (symbol probabilities are computed using bit probabilities) operations are performed during the decoding of *TMCPC*. Columns are decoded first and symbol probabilities are obtained. Data and parity bit probabilities are extracted from column symbol probabilities.

A similar procedure can be considered for obtaining the symbol probabilities from the bit probabilities (i.e., obtain symbol probabilities from the

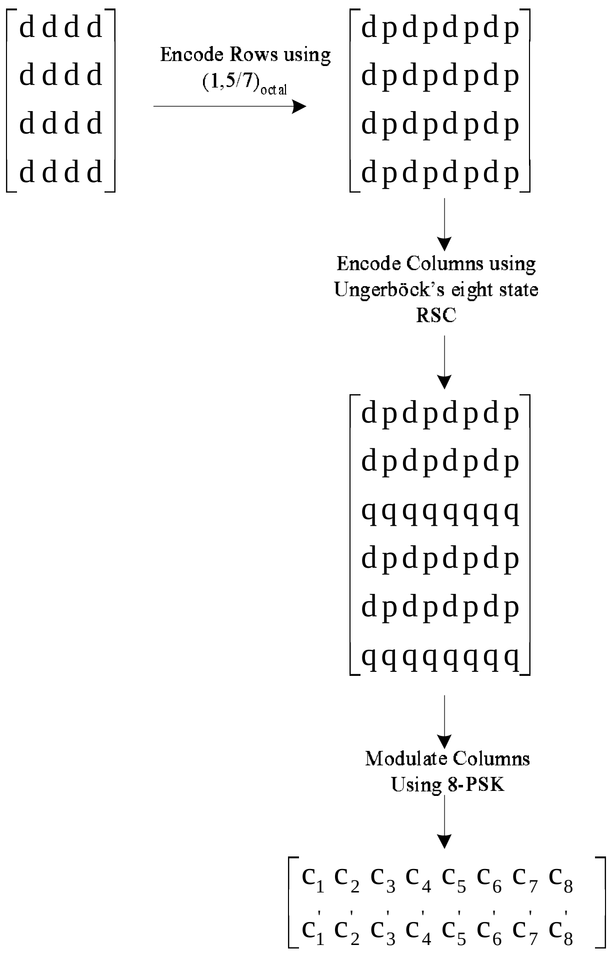

Encode Rows using (1,5/7)octal
Encode Columns using Ungerböck's eight state RSC
Modulate Columns Using 8-PSK

Figure 6.16: *TMCPC* encoding operation.

marginal bit probabilities). Using the bit probabilities, symbol probabilities for the row symbols are found assuming that the bits forming the symbols are independent. These are given to the symbol-wise row decoders. Row symbol-wise log-MAP decoders produce new likelihood symbol probabilities. Subtracting old symbol probabilities from new symbol probabilities, extrinsic symbol probabilities are obtained and given to the column decoders. This procedure is repeated sufficiently many times. Overall decoding procedure is illustrated in Fig. 6.17

The system performance is depicted in Fig. 6.18. Shannon limit for 8-PSK [66] modulation for the given code rate is approximately $\frac{Eb}{N_0} = 0$ dB. From performance graph it is seen that *TMCPC* is 1.75 dB away from the Shannon limit. In [68] a detailed study of serially concatenated trellis coded modulation has been made. From the results in [68], classical serially concatenated *TCM* has almost the same performance as *TMCPC*. They both reach to 10^{-5} at $E_b/N_0 = 1.6dB$.

6.2.3 Joint *CPC* and *STTC*

In this section joint structure involving space time trellis codes and convolutional product codes will be given. We first give a brief information about space time codes then explain our joint structure.

6.2.3.1 Space Time Codes

Multipath fading could severely degrade the performance of wireless systems. Traditional block codes and convolutional codes are generally used in single antenna transmission systems. In multi antenna transmission systems space time block codes and space time trellis codes are used to enable diversity gain and forward error correction. These codes benefit from diversity and enables better communication. Space time block codes provide diversity gain with a simple decoding algorithm. However, they do not provide coding gain. Therefore they may not be considered as a block code. Space time trellis

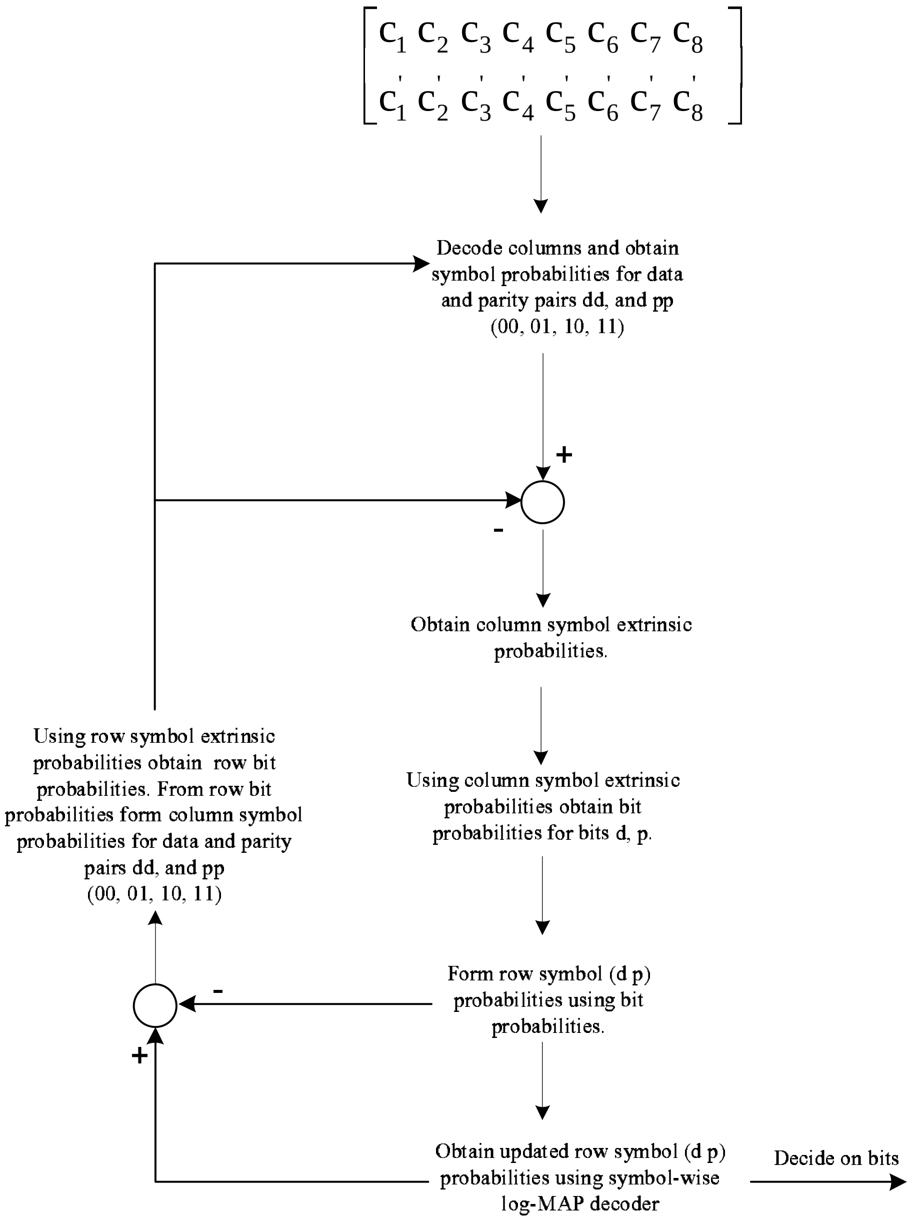

$c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8$
$c'_1\ c'_2\ c'_3\ c'_4\ c'_5\ c'_6\ c'_7\ c'_8$
Decode columns and obtain symbol probabilities for data and parity pairs dd, and pp (00, 01, 10, 11)
+
-
Obtain column symbol extrinsic probabilities.
Using column symbol extrinsic probabilities obtain bit probabilities for bits d, p.
Form row symbol (d p) probabilities using bit probabilities.
Obtain updated row symbol (d p) probabilities using symbol-wise log-MAP decoder
Decide on bits
-
+
Using row symbol extrinsic probabilities obtain row bit probabilities. From row bit probabilities form column symbol probabilities for data and parity pairs dd, and pp (00, 01, 10, 11)

Figure 6.17: *TMCPC* decoding operation.

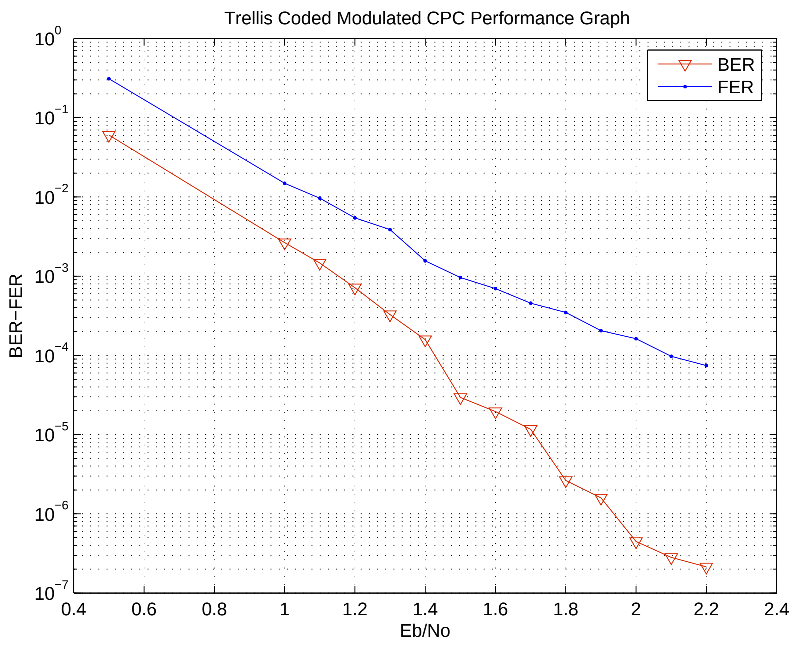
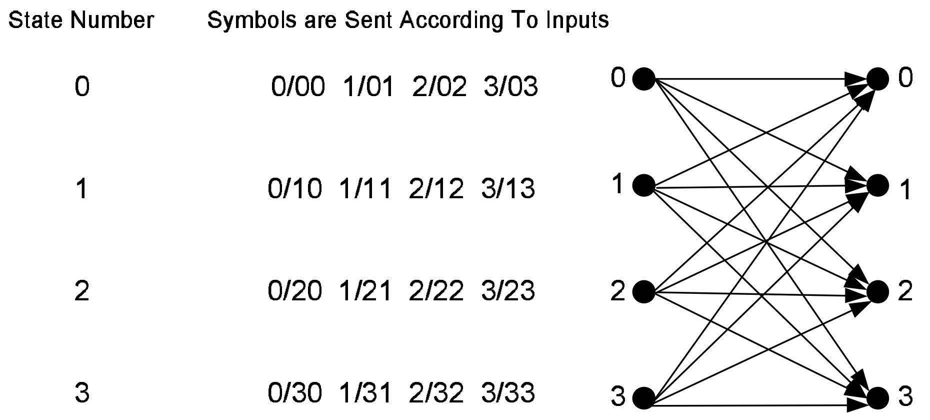


Figure 6.18: *TMCPC* performance graph. Frame length=1024. Iteration number=12. *S-random* interleaver is used ($S = 20$). N=32, M=34

codes *(STTCs)* were first introduced by Tarokh, Seshadri and Calderbank [69]. *STTCs* achieve coding gain in addition to diversity gain. This makes them more powerful when compared to space time block codes. However, their decoding is more complex and this makes them less attractive due to the existence of limited computation facilities. The trellis diagram of Tarokh's 4 state *STTC* is depicted in Fig. 6.19 where input symbols are *QPSK* modulated and represented by 0, 1, 2, 3. The modulated signals are sent to the state machine. Parity symbols are added to the input symbols and each symbol is sent using a transmitter antenna.

The *MIMO* transmission system for any number of transmitter and receiver antennas is illustrated in Fig. 6.20. The channel between N_t transmitter and M_r receiver antennas is represented with a matrix H of size $M_r \times N_t$. The elements of the channel matrix h_{ij} are channel coefficients, and they represent the channel gain between transmitter antenna i and receiver antenna j for a channel without intersymbol interference.



State Number	Symbols are Sent According To Inputs
0	0/00 1/01 2/02 3/03
1	0/10 1/11 2/12 3/13
2	0/20 1/21 2/22 3/23
3	0/30 1/31 2/32 3/33

Figure 6.19: *STTC* 4-state encoder diagram.

$$H = \begin{pmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,N_t} \\ h_{2,1} & h_{2,2} & \dots & h_{2,N_t} \\ \vdots & \vdots & \dots & \vdots \\ h_{M_r,1} & h_{M_r,2} & \dots & h_{M_r,N_t} \end{pmatrix} \tag{6.4}$$

If the encoded frame length equals L, then we can form a matrix X of size $N_t \times T$, where T equals L/N_t, N_t is the number of transmitter antennas, M_r is the number of receiver antennas. The received signal can be expressed by a matrix Y of size $M_r \times T$ as in

$$Y = \sqrt{\frac{\rho}{M}} HX + W, \tag{6.5}$$

where W is the $M_r \times T$ additive noise matrix whose elements are samples of independent zero-mean complex Gaussian random variables with variance $N_0/2$, and ρ is the signal-to-noise ratio (*SNR*) per receiver antenna.

6.2.4 Joint *CPC* and *STTCs*

CPC is a special case of *PDSCCCs* and it can be integrated with *STTCs*. In Figs. 6.21, 6.22, and 6.23 joint encoding and decoding operations are illustrated. Interleaver is omitted from Figs. 6.22 and 6.23 for simplicity after row encoding operation. In fact the use of interleaver is critical for the good operation of the joint coded system. The state diagram given in Fig. 6.19 uses

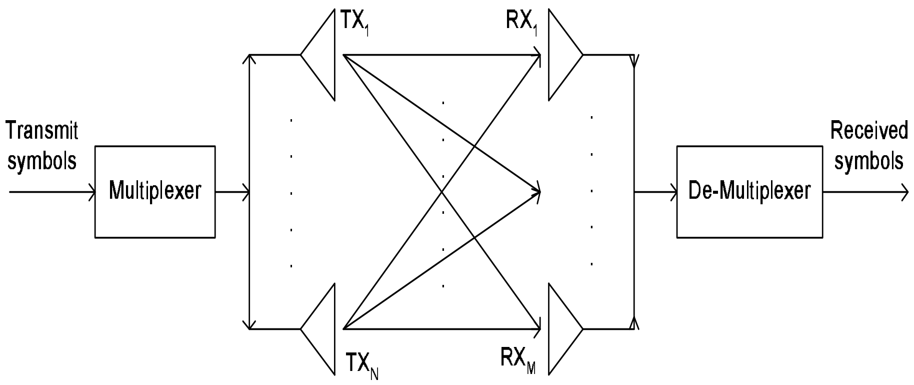


Figure 6.20: *MIMO* communication system.

two transmitter antennas.

Decoding operation of the *CPC-MIMO* systems are similar to that of the *TCM-CPC* systems. *STTCs* can be decoded using the Viterbi [51] algorithm. In addition, the MAP algorithm [51] can also be employed for the decoding operation of *STTCs*. For our simulations we used MAP. The overall decoding operation is given in Fig. 6.23.

In MAP and Viterbi algorithms, branch metrics (BM) are calculated according to [22]

$$BM = \sum_{i=1}^{M_r} \mid y_i - \sum_{j=1}^{N} {}_t h_{ij} \tilde{x}_j \mid^2, \tag{6.6}$$

where N_t is the number of transmitter antennas, M_r is the number of receiver antennas, h_{ij} is the channel coefficient between transmitter i and receiver j, $\tilde{x}_j$ is the transmitted symbol from transmitter antenna j and is determined from the state transition diagram of the *STTC*. It is assumed that perfect channel information is available at the receiver. *CPC-STTC* encoding operation is similar to that of the *CPC-TCM* structure. After row encoding operation, mapping is performed for column bits, and space time trellis coding is performed along columns. The encoded symbols are multiplexed and transmitted using multi antenna systems. For the simulation of this joint structure we have used 2 transmitter and 2 receiver antennas. Joint structure involves *CPC* with constituent codes $(1, 5/7)_{octal}$ and Tarok's 4-state *STTCs*. The simula-

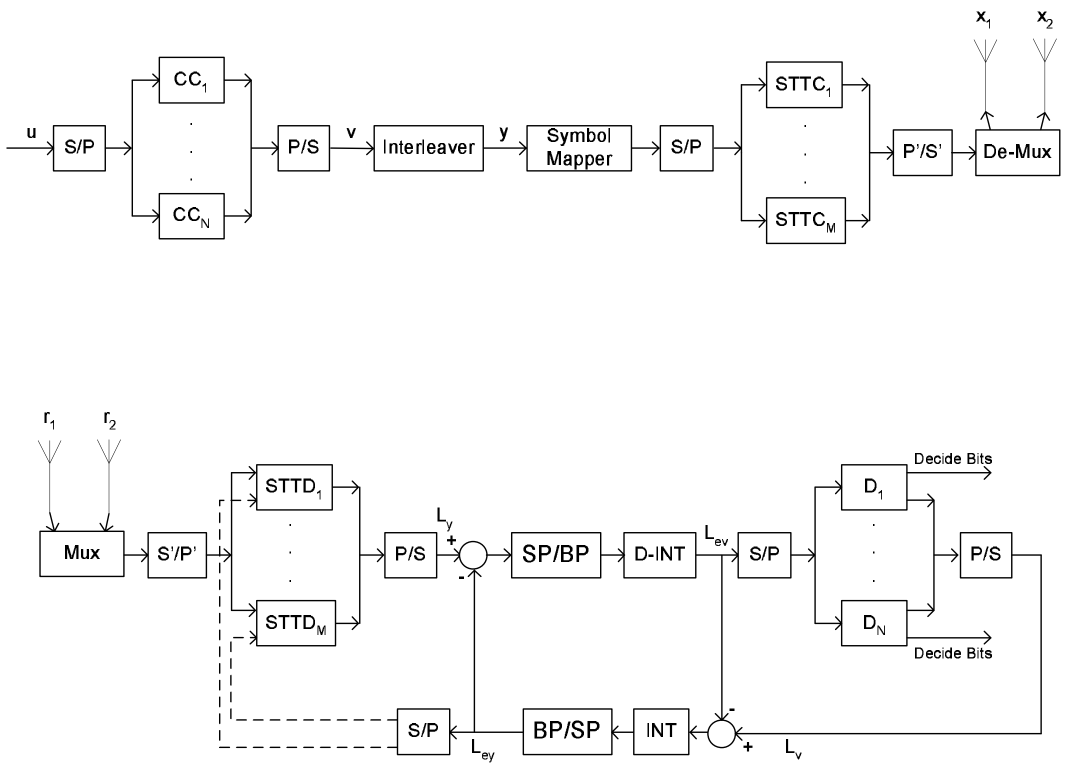
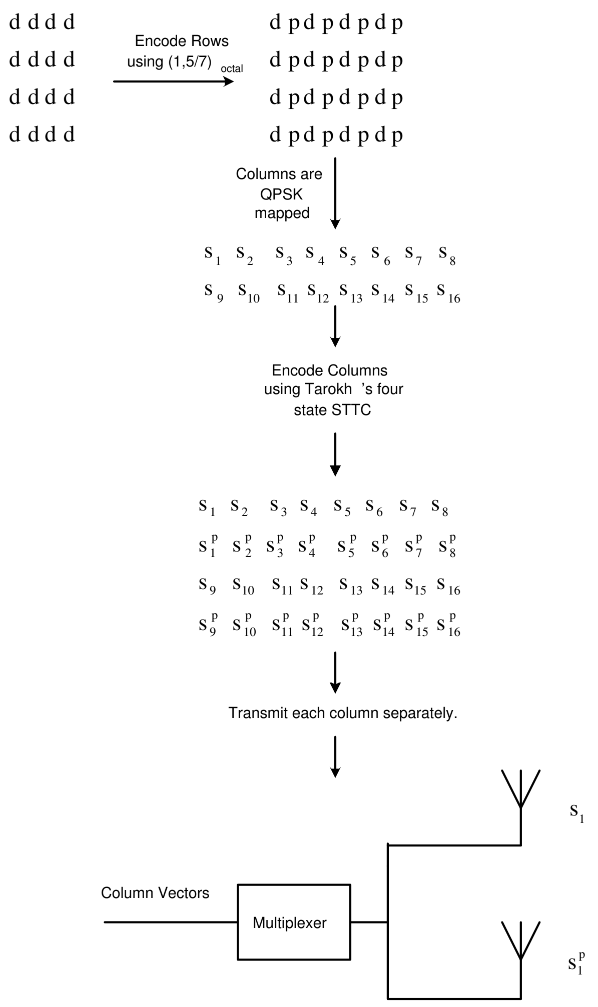
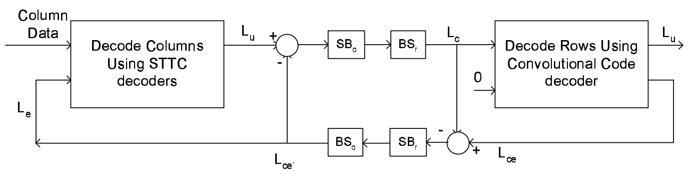


Figure 6.21: Parallel decodable concatenated space time trellis code encoder and decoder blocks. S'/P' is the pairwise serial to parallel converter, i.e., cells holding data and parity symbol pairs from *STTCs* are multiplexed and transmitted.



Encode Rows using (1,5/7) octal
Columns are QPSK mapped
Encode Columns using Tarokh 's four state STTC
Transmit each column separately.
Column Vectors
Multiplexer

Figure 6.22: *MIMO-CPC* encoding operation.



Column Data
Decode Columns Using STTC decoders
L_u
SB_c
BS_r
L_c
Decode Rows Using Convolutional Code decoder
L_u
0
L_e
BS_c
SB_r
$L_{ce'}$
L_{ce}

Figure 6.23: *MIMO* decoding operation. SB_c denotes symbol to bit probability conversion for column vectors. BS_r denotes bit to symbol probability conversion for rows.

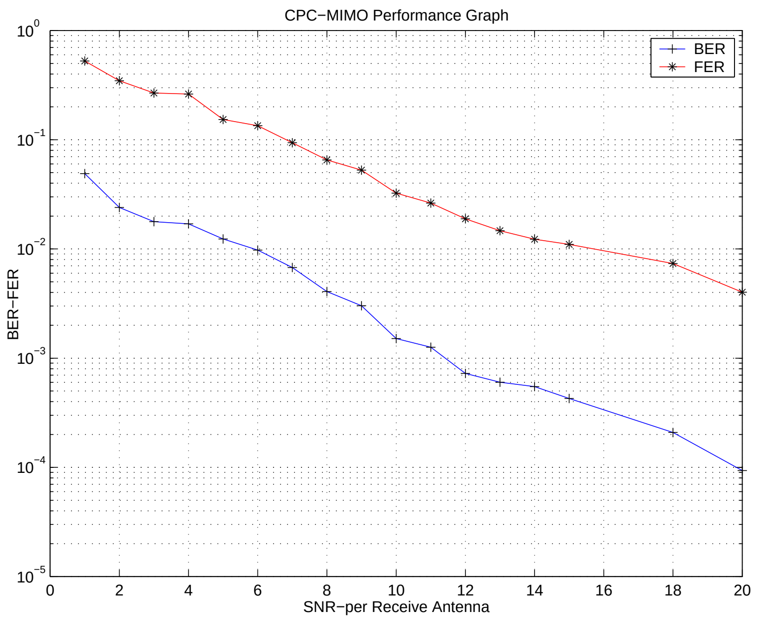


Figure 6.24: *CPC-MIMO* performance graph.

tion result is depicted in Fig. 6.24. Serial concatenation of space time trellis codes and convolutional codes are studied in [70] From the results of [70] it is clear that the proposed system achieves similar performance, and has much less decoding latency.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Iteratively decoded codes perform very close to the Shannon limits. However, iterative decoding operation limits their use in practice due to large decoding latency. The work on this thesis concentrates on enhancing the latency of iterative decoding. In this thesis work, we proposed parallel decodable concatenated codes which reduces the latency by parallel processing. The latency is decreased by a factor of number of parallel branches employed during the decoding operation. The main idea for the construction of these systems is to employ parallel encoders at the transmitter side and make use of it at the decoder side. This provides us with more flexibility on code design. Using this idea, convolutional product codes and a more general structure called parallel decodable serially concatenated codes *(PDSCCs)* are suggested. We proposed parallel decodable serially concatenated convolutional codes *(PDSCCCs)* using the *PDSCC* structure. We showed that using *PDSCC* structure regular product codes, serially concatenated convolutional codes, and convolutional product codes can be generated. The effect of different interleavers on the performance of *CPCs* are investigated. Minimum distance of the *CPCs* are inspected with regard to different interleavers. The effect of trellis termination bits on *CPC* performance is studied. The proposed structures are analytically investigated using the uniform interleaver approach.

Parallel processing operation introduces some new problems. One of the most critical problems is memory collision. Specific interlavers are designed in literature to prevent the memory collision problem and these interleavers are usually constructed using algebraic techniques. We propose row-column *S-random* interleavers which has random behavior like *S-random* interleavers and show comparable performance to that of *S-random* interleaver. This type of interleavers also prevents the memory collision problem. Row-column *S-random* interleaver can be considered as a joint structure of helical and *S-random* interleavers. The performance of the *PDSCCCs* with memory collision free row-column *S-random* interleavers are demonstrated.

Pursuing the same idea, parallel decodable turbo codes (*PDTCs*) are proposed. It is seen that *PDTCs* show comparable performance to that of the classical turbo codes with a significant reduced latency. Minimum distance of the parallel decodable turbo codes are investigated and we provided bounds for the minimum distance considering the use of different interleavers. It is seen that, to get maximum benefit from the interleaving gain and maximize the d_{min} of the *PDTC*, row-column *S-random* interleaver is the best choice. Simulation results for different interleavers are presented. It is also seen that some instances of our general structure *(PDTC)* are already available in literature.

It is soon realized that the same methodology can be extended to any communication unit which does have a regular trellis structure. Turbo equalizer and space time trellis codes are two such structures. We proposed joint structures involving *CPCs*, trellis coded modulation, and space time trellis codes. The joint structures benefit from reduced latency and have similar performance compared to their counterparts.

Although parallel processing reduces the decoding latency enormously, hardware complexity is increased almost by the same factor. Hence, a tradeoff exists between decoding latency and hardware complexity for the proposed structures. Using different settings in the proposed structures, various systems can be obtained and the best one can be decided considering complexity, latency, and hardware complexity.

Erasure decoding is an important concept for bursty error channels. Since our proposed units have a matrix structure, they may be useful for erasure decoding and this is a subject we currently investigate on. Another future study is the joint structure involving *CPC* and *OFDM*. Two dimensional structure of *CPC* enables it to be easily integrated with *OFDM*. We look into integrating the *CPC* frames into a time selective *OFDM* system so that joint channel estimation and decoding is easily performed. The implementation of a *CPC* decoder on a *FPGA* platform is also under way. Bit and symbol interleaved joint *PDSCCC*, *TCM*, *PDTC*, and *MIMO* systems is another future topic under investigation.

APPENDIX A

Computation of Spectrum Function of Convolutional Codes

We below give an example to illustrate the spectrum function computations of convolutional codes using the polynomial approach.

Example:

The task is to find *IRWEF* of the convolutional code with the generator polynomial $(1, 5/7)_{octal}$. State and block diagrams for this encoder are shown in Fig. A.1. The state diagram is transformed into a state transition matrix $T(W, Z)$ as shown below. Each branch label of the state diagram is replaced with the polynomial $W^w Z^z$, where W and Z are dummy variables which facilitate the enumeration of the input weight(w) and parity weight(z), respectively. The element of the $T(W, Z)$ at index location (i, j) resembles the branch label from state transition i to j.

$$T(W,Z) = \begin{pmatrix} 1 & 0 & WZ & 0 \\ WZ & 0 & 1 & 0 \\ 0 & W & 0 & Z \\ 0 & Z & 0 & W \end{pmatrix} \tag{A.1}$$

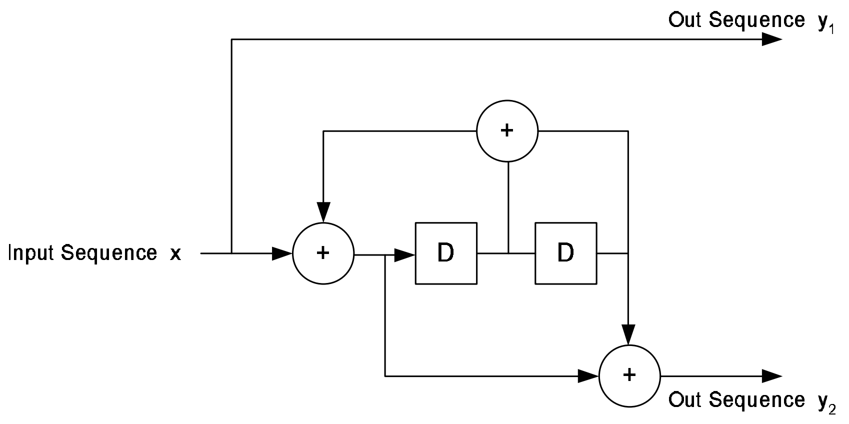
For an input sequence of length k, frame transition matrix $F(W, Z)$ is defined by,

$$F(W,Z) = T(W,Z)^k. \tag{A.2}$$

Each element of the $F(W,Z)$ matrix is a polynomial. The sum of all the polynomials (i.e., the sum of elements of $F(W,Z)$) equals the *IRWEF* ($A(W,Z)$) of the convolutional code).

$$A(W,Z) = \sum_{i,j} F_{i,j}(W,Z) \tag{A.3}$$

The element of $F(W,Z)$ at index position $(2,3)$ is the *IRWEF* for the codewords whose trellis diagrams start at state 2 and ends at state 3. Since convolutional codes that we use are always trellis terminated (i.e., trellis diagrams end at state 0), the $(0,0)$ element of $F(W,Z)$ gives complete information about the convolutional code, i.e., *IRWEF* is $[F(W,Z)]_{1,1}$



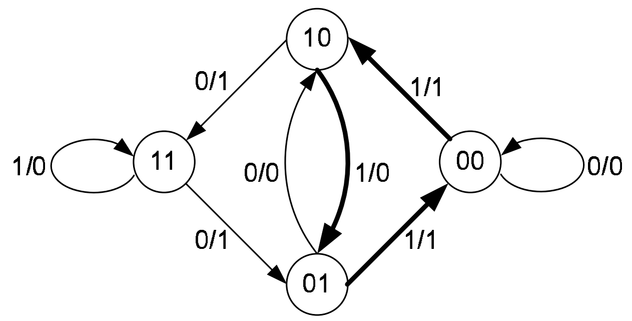


Figure A.1: State machine for convolutional code generator $(1, 5/7)_{octal}$.

For moderate even small input sequence lengths it is difficult to find the *IRWEFs* of the linear codes. Some mathematics software is usually used to compute *IRWEFs*. Polynomial multiplication equals convolution of the coefficients of the variables which are written in increasing order. If your polynomial uses two variables then two dimensional convolution should be used. For n variable polynomials n dimensional convolution is used. To find the frame transition matrix for an input sequence of length k (i.e., $T(W,Z)^k$) each element of the transition matrix $T(W,Z)$ is expressed as a matrix of size $p \times p$, here p is arbitrary and depends on the user's will. A larger p gives more accurate results, however an intermediate p is sufficient due to the fast decay rate of the error function. Once each element of the transfer function $T(W,Z)$ is expressed as a matrix, then any power of the $T(W,Z)$ can be taken. $T(W,Z)$ elements are polynomials, and element multiplication corresponds to $2D$ convolution of their matrix representation. After each $2D$ convolutional operation resultant matrix dimension doubles, hence we truncate the resultant matrix to the original size by taking the first p rows and columns and forming a new matrix. When all the powers are taken, the $[T(W,Z)]_{1,1}$ element is our transfer function for the trellis terminated convolutional code.

Example: $[T(W,Z)]_{2,1} = WZ$ This element is expressed in matrix form as:

$$T_{(2,1)}(W,Z) = \begin{array}{cccccc|c} W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & \\ 0 & 0 & 0 & 0 & 0 & 0 & Z^0 \\ 0 & 1 & 0 & 0 & 0 & 0 & Z^1 \\ 0 & 0 & 0 & 0 & 0 & 0 & Z^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & Z^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & Z^4 \\ 0 & 0 & 0 & 0 & 0 & 0 & Z^5 \end{array}$$

Figure A.2: Matrix representation of polynomial WZ.

APPENDIX B

Analytical Bound Calculation

The bit error probability computation of a block code is explained below by an example where we assume that *IRWEF* of the block code is already available.

Example:

For BCH(7,4) code there are only 2^4 codes, and *IRWEF* for the BCH code is given below:

$$\begin{aligned} A^C(W,Z) &= 1W^0Z^0 + 3W^1Z^2 + W^1Z^3 + 3W^2Z^1 + 3W^2Z^2 + 1W^3Z^0 \\ &\quad + 3W^3Z^1 + 1W^4Z^3 \qquad \text{(B.1)} \\ &= 1 + W(3Z^2 + Z^3) + W^2(3Z + 3Z^2) + W^3(1 + 3Z) + W^4Z^3 \qquad \text{(B.2)} \end{aligned}$$

The *CWEFs* are obtained from *IRWEFs* as follows.

$$A_0^C(Z) = 1 \qquad w = 0 \qquad \text{(B.3)}$$

$$A_1^C(Z) = 3Z^2 + Z^3 \qquad w = 1 \qquad \text{(B.4)}$$

$$A_2^C(Z) = 3Z + 3Z^2 \qquad w = 2 \qquad \text{(B.5)}$$

$$A_3^C(Z) = 1 + 3Z \qquad w = 3 \qquad \text{(B.6)}$$

$$A_4^C(Z) = Z^3 \qquad w = 4 \qquad \text{(B.7)}$$

The relation between *IRWEF* coefficients and *CWEF* is given in eqn. (B.8).

$$A_w^C(Z) = \sum_j A_{w,j} Z^j \qquad \text{(B.8)}$$

CWEF of the concatenated code is obtained by

$$A_w^{C_p} = \frac{A_w^{C_1} \times A_w^{C_2}}{\binom{N}{w}}, \tag{B.9}$$

where the coefficients are readily evaluated for our example as follows.

$$A_0^{C_p}(Z) = \frac{1 \times 1}{\binom{4}{0}} = 1 \tag{B.10}$$

$$A_1^{C_p}(Z) = \frac{(3Z^2 + Z^3) \times (3Z^2 + Z^3)}{\binom{4}{1}} = \frac{9}{4}Z^4 + \frac{3}{2}Z^5 + \frac{1}{4}Z^6 \tag{B.11}$$

$$A_2^{C_p}(Z) = \frac{(3Z + Z^2) \times (3Z + Z^2)}{\binom{4}{2}} = \frac{3}{2}Z^2 + 3Z^3 + \frac{3}{2}Z^4 \tag{B.12}$$

$$A_3^{C_p}(Z) = \frac{(1 + 3Z) \times (1 + 3Z)}{\binom{4}{3}} = \frac{1}{4} + \frac{3}{2}Z + \frac{9}{4}Z^2 \tag{B.13}$$

$$A_4^{C_p}(Z) = \frac{Z^3 \times Z^3}{\binom{4}{4}} = Z^6 \tag{B.14}$$

The *IRWEF* for parallel concatenated code is computed using (B.15) and is given in (B.16).

$$A^{C_p}(W, Z) = \sum_w A_w^{C_p}(Z) W^w \tag{B.15}$$

$$\begin{aligned} A^{C_p}(W, Z) = 1 + W(\frac{9}{4}Z^4 + \frac{3}{2}Z^5 + \frac{1}{4}Z^6) + W^2(\frac{3}{2}Z^2 + 3Z^3 + \frac{3}{2}Z^4) \\ + W^3(\frac{1}{4} + \frac{3}{2}Z + \frac{9}{4}Z^2) + W^4(Z^6) \end{aligned} \tag{B.16}$$

D_m's are computed using the *IRWEF*. Using D_m's bit error probability $P_b(e)$ is computed as:

$$P_b(e) \approx \frac{1}{2} \sum_{j+w=m} D_m erfc\left(\sqrt{\frac{mRE_b}{N_0}}\right) \tag{B.17}$$

$$\begin{aligned}P_b(e) = \frac{1}{2}\Bigg[&0.1875erfc\left(\sqrt{\frac{3RE_b}{N_0}}\right)\\&+ 1.875erfc\left(\sqrt{\frac{4RE_b}{N_0}}\right)\\&+ 4.3125erfc\left(\sqrt{\frac{5RE_b}{N_0}}\right)\\&+ 1.125erfc\left(\sqrt{\frac{6RE_b}{N_0}}\right)\\&+ 0.0625erfc\left(\sqrt{\frac{7RE_b}{N_0}}\right)\\&+ erfc\left(\sqrt{\frac{10RE_b}{N_0}}\right)\end{aligned}$$

APPENDIX C

The MAP Algorithm

The MAP algorithm as implemented through the BCJR algorithm is summarized here. Assume that there are P complex numbers used in the constellation scheme. The constellation alphabet consists of the symbols $C = \{c_1, c_2, \ldots, c_{P-1}, c_P\}$.

The received signal vector is denoted by y and has length L, i.e., L symbols are transmitted. The task is to determine the transmitted symbols. Given the received signal vector, the probability that a specific symbol value is sent has to be determined. Given the received vector, the probability of the k^{th} data symbol being equal to each of the symbols should be evaluated as

$$p(u_k = c_i|y) = \sum_{s,s':u_k=c_i} p(s_{k-1} = s',\ s_k = s, |\ y). \tag{C.1}$$

Since $p(y)$ does not depend on c_i, we will actually evaluate (C.1) using the states at time epochs $k-1$ and k

$$p(s',\ s,\ y) = p(s_{k-1} = s',\ s_k = s,\ y), \tag{C.2}$$

where s_n is the state of the finite state machine at time epoch n.

For $y = [y_1,\ y_2,\ \ldots,\ y_L]$, the vector $y_1^k = [y_1, \ldots, y_k]$ is defined. Eqn. (C.2) can now be stated as

$$\begin{aligned}
p(s', s'y) &= p(s', s, y_1^{k-1}, y_k, y_{k+1}^K) \\
&= p(y_{k+1}^L | s', s, y_1^{k-1}, y_k) p(s', s, y_1^{k-1}, y_k) \\
&= p(y_{k+1}^L | s', s, y_1^{k-1}, y_k) p(s, y_k | s', y_1^{k-1}) p(s', y_1^{k-1}) \\
&= p(y_{k+1}^L | s) p(s, y_k | s') p(s', y_1^{k-1}) \\
&= \beta_k(s) \gamma_k(s', s) \alpha_{k-1}(s').
\end{aligned} \tag{C.3}$$

Based on (C.3),

$$\begin{aligned}
\alpha_k(s) &= p(s, y_1^k) \\
&= \sum_{s'} p(s', s, y_1^k) \\
&= \sum_{s'} p(s, y_k | s', y_1^{k-1}) p(s', y_1^{k-1}) \\
&= \sum_{s'} p(s, y_k | s') p(s', y_1^{k-1}) \\
&= \sum_{s'} \gamma_k(s', s) \alpha(s').
\end{aligned} \tag{C.4}$$

Hence, the probability $\alpha_k(s)$ can be computed recursively. Similarly the probability $\beta_k(s)$ can be computed in a recursive way as in

$$\beta_{k-1}(s') = \sum_{s} \beta_k(s) \gamma_k(s', s). \tag{C.5}$$

For encoders starting from the zero state, α_0 is initialized as

$$\alpha_0(s) = \begin{cases} 1 & \text{if } s = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{C.6}$$

Similarly $\beta_L(s)$ s are initialized with,

$$\beta_L(s) = \begin{cases} 1 & \text{if } s = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{C.7}$$

for the encoders with trellis termination to the zero state. The main variable used in recursion operations is evaluated by

$$\begin{aligned}
\gamma_k(s', s) &= p(s, y_k|s') \\
&= \frac{p(s', s, y_k)}{p(s')} \\
&= \frac{p(s', s)}{p(s')}\frac{p(s', s, y_k)}{p(s', s)} \\
&= p(s|s')p(y_k|s', s) \\
&= p(u_k)p(y_k|u_k),
\end{aligned} \tag{C.8}$$

where the event u_k corresponds to the transition from s' to s. Its values is nonzero if there is a valid transition otherwise it is zero.

REFERENCES

[1] C. Shannon, "A mathematical theory of communication," *Tech. Rep., Bell Systems*, vol. 27, pp. 379–423, 1948.

[2] R. W. Hamming, "Error detecting and correcting codes," *Tech. Rep., Bell Systems*, vol. 26, no. 2, pp. 147–160, 1950.

[3] M. J. E. Golay, "Notes on digital coding," *Proc. IEEE*, vol. 37, p. 657, 1949.

[4] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, pp. 69–79, 1960.

[5] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM Journal on Applied Mathematics*, vol. 8, pp. 598–606, 1960.

[6] E. R. Berlekamp, "Nonbinary bch decoding," *IEEE Trans. Inform. Theory*, vol. 14, no. 2, pp. 300–304, 1968.

[7] P. Elias, "Coding for noisy channels," *IRE. Conv. Record*, vol. 4, pp. 37–47, 1955.

[8] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. 13, pp. 260–269, 1967.

[9] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of error-control coding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2531–2560, 1998.

[10] G. Ungerboeck and I. Csajka, "On improving data-link performance by increasing the channel alphabet and introducing sequence coding," in *Int. Symp. Inform. Theory*, Ronneby, Sweden, June 1976.

[11] S. Wicker, *Error Control Systems for Digital Communications and Storage.* Prentice Hall, Inc.: Englewood Cliffs, 1995.

[12] J. G. D. Forney, *Concatenated Codes.* MA, USA: M.I.T. Press, 1966.

[13] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. ICC'93*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[14] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, 1962.

[15] C. Douillard *et al.*, "Iterative correction of intersymbol interference: Turbo equalization," *Eur. Trans. on Telecommun.*, vol. 6, pp. 507–511, 1995.

[16] S. Host, "On woven convolutional codes," Ph.D. dissertation, Lund University, Sweden, Sept. 1999.

[17] S. Chaoui, "Convolutional coupled codes," Ph.D. dissertation, University of Darmstadt, Germany, Feb. 2003.

[18] J. Feudenberger, M. Bossert, V. Zyablov, and S. Shavgulidze, "Woven turbo codes," in *Seventh International Workshop on Algebraic and Combinatorial Coding Theory*, Bansko, Bulgaria, June 2000, pp. 145–150.

[19] S. Lin and D. J. Costello, *Error Control Coding.* Prentice Hall, 2004.

[20] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding.* IEEE Press, 1999.

[21] J. Conan, "The weight spectra of some short low-rate convolutional codes," *IEEE Trans. Commun.*, vol. 32, no. 9, pp. 1050–1053, 1984.

[22] L. Hanzo, T. H. Liew, and B. L. Yeap, *Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels.* John Wiley and Sons, 2002.

[23] B. Vucetic and J. Yuan, *Turbo Codes Principles and Applications.* Kluwer Academic Publishers, 2004.

[24] R. D. Wesel, "Convolutional codes," *Encyclopedia of Telecommunications*, vol. 1, pp. 598–606, 2003.

[25] P. Elias, "Error free decoding," *IRE Trans. Inform. Theory*, vol. IT-4, pp. 29–37, 1954.

[26] S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 409–428, 1996.

[27] F. Marx and J. Farah, "Improved turbo-coded umts systems with unequal error protection of compressed video sequences transmitted over frequency-selective channels communications," in *IEEE Int. Conf.*, vol. 5, 2004, pp. 3091–3095.

[28] A. Shibutani, H. Suda, and F. Adachi, "Complexity reduction of turbo decoding," in *IEEE VTS 50th*, Ansterdam, Netherlands, Sept. 1999, pp. 1570–1574.

[29] J. W. Jung *et al.*, "Design and architecture of low-latency high-speed turbo decoders," *ETRI Journal*, vol. 27, no. 5, pp. 525–532, 2005.

[30] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel turbo and ldpc decoder architectures," *IEEE Trans. Inform. Theory*, vol. 50, no. 9, pp. 2002–2009, 2004.

[31] C. Heegard and S. B. Wicker, *Turbo Coding.* Kluwer Academic Press, 1999.

[32] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serially concatenation of interleaved codes: Design and performance analysis," *IEEE Trans. Inform. Theory*, vol. 44, pp. 909–926, 1998.

[33] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 591–600, 1996.

[34] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-input, soft output modules for the construction and distributed iterativ decoding of code networks," vol. 9, no. 5, pp. 155–172, 1998.

[35] V. Franz and J. B. Anderson, "Concatenated decoding with a reduced search based bcjr algorithm," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 186–195, 1998.

[36] D. Lee and I. Park, "A low complexity stopping criterion for iterative turbo decoding," *IEICE Trans. Commun.*, vol. 88, no. 1, pp. 399–401, 2005.

[37] F. Zhai and I. Fair, "Techniques for early stopping and error detection in turbo decoding," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1617–1623, 2003.

[38] C. Schurgers, E. Catthoor, and M. Engels, "Optimized map decoder," in *IEEE Workshop on Signal Processing Systems*, Lafayette, LA, USA, Jan. 2000, pp. 245–254.

[39] P. Robertson, E. Villebrum, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," in *IEEE Int. Conf. on Comm.*, vol. 2, Seattle, USA, 1995.

[40] S. Yoon and Y. Bar-Ness, "A parallel map algorithm for low latency turbo decoding," *IEEE Commun. Lett.*, vol. 6, no. 7, pp. 288–290, 2002.

[41] Y. Wang, J. Zhang, M. Fossorier, and J. S. Yedidia, "Reduced latency turbo decoding," in *SPAWC, IEEE 6th Workshop*, Honolulu, USA.

[42] 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. IT-20, pp. 284–287, 1974.

[43] R. M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Commun. Lett.*, vol. 6, no. 8, pp. 1003–1010, 1998.

[44] E. Hewitt, "Turbo product codes for lmds," in *IEEE Radio and Wireless Conference*, 1998, pp. 107–111.

[45] N. Y. Yu, Y. Kim, and P. J. Lee, "Iterative decoding of product codes composed of extended hamming codes," in *Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, Antibes, France, July 2000, pp. 732–737.

[46] T. Shohon, Y. Soutome, and H. Ogiwara, "Simple computation method of soft value for iterative decoding of product code composed of linear block code," *IEIC Trans. Fundamentals*, vol. 82, no. 10, pp. 2199–2203, 1999.

[47] O. Aitsab and R. Pyndiah, "Performance of reed solomon block turbo codes," in *Proc. IEEE GLOBECOM'96 Conf.*, vol. 1/3, London, U.K., Jan. 1996, pp. 121–125.

[48] D. Rankin and T. A. Gulliver, "Single parity check product codes," *IEEE Trans. Commun.*, vol. 49, no. 8, pp. 1354–1362, 2001.

[49] D. Rankin and T. Gulliver, "Randomly interleaved single parity check product codes," in *Proc. IEEE Int. Symp. on Inform. Theory*, June 2000, p. 88.

[50] A. Goalic and R. Pyndiah, "Real time turbo decoding of product codes on a digital signal processor," in *Int. Symposium on turbo codes and related topics*, Brest, Sept. 1997, pp. 624–628.

[51] S. Lin and D. J. Costello, *Error Control Coding.* Prentice Hall, 2004.

[52] B. Vucetic and J. Yuan, *Turbo Codes Principles and Applications.* Kluwer Academic Publishers, 2000.

[53] P. Robertson, "Illuminating the structure of parallel concatenated recursive (turbo) codes," in *Proc. GLOBECOM'94*, no. 15, San Francisco, CA, Nov. 1994, pp. 1298–1303.

[54] J. Hagenauer, "Rate-compatible punctured convolutional codes (rcpc codes) and their applications," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 909–926, 1998.

[55] A. Giulietti, L. van der Perre, and M. Strum, "Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements," *Electron. Lett.*, vol. 38, no. 5, pp. 232–233, 2002.

[56] J. Kwak and K. Lee, "Design of dividable interleaver for parallel decoding in turbo codes," *Electron. Lett.*, vol. 38, no. 22, pp. 1362–1364, 2002.

[57] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jezequel, "Designing good permutations for turbo codes: towards a single model," in *IEEE Int. Conf. on Comm.*, vol. 1, June 2004, pp. 341–345.

[58] R. Dobkin, M. Peleg, and R. Ginosar, “Parallel interleaver design and vlsi architecture for low-latency map turbo decoders,” *IEEE Trans. VLSI Syst.*, vol. 13, no. 4, pp. 427–438, 2005.

[59] L. Dinoi and S. Benedetto, “Variable-size interleaver design for parallel turbo decoder architectures,” in *IEEE Commun. Society Globecom*, Dallas, USA, Sept. 2004, pp. 1833–1840.

[60] J. Proakis, *Introduction to Magnetic Materials.* McGraw-Hill: Digital Communications, 4th ed., 2001.

[61] M. Tuchler, R. Koetter, and A. Singer, “Turbo equalization: Principles and new results,,” *IEEE Trans. Commun.*, vol. 50, no. 5, pp. 754–767, 2003.

[62] F. R. Rad and J. Moon, “Low complexity turbo equalization for high density magnetic recording,” in *Proc. ICC 2005*, vol. 1, May 2005, pp. 688–692.

[63] A. Glavieux, C. Laot, and J. Labat, “Turbo equalization over a frequency selective channel,” in *Proc. Int. Symp. Turbo Codes*, Brest, France, Sept. 1997, pp. 96–102.

[64] S. Ariyavisitakul and Y. Li, “Joint coding and decision feedback equalization for broadband wireless channels,” *IEEE J. Select. Areas Commun.*, vol. 16, pp. 1670–1678, 1998.

[65] J. B. Anderson and S. M. Hladik, “Tail biting map decoders,” *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 297–302, 1998.

[66] G. Ungerboeck, “Trellis coded modulation with redundant signal sets, part i: Introduction,” *IEEE Commun. Mag.*, vol. 25, no. 2, pp. 5–11, 1987.

[67] ——, “Trellis coded modulation with redundant signal sets, part ii: State of the art,” *IEEE Commun. Mag.*, vol. 25, no. 2, pp. 12–21, 1987.

[68] P. K. Gray, “Serially concatenated trellis coded modulation,” Ph.D. dissertation, University of South Australia, Mar. 1999.

[69] N. S. V. Tarokh and A. R. Calderbank, “Space-time codes for high data rate wireless communication: performance criterion and code construction,” *IEEE Trans. Inform. Theory*, vol. 44, no. 2, pp. 744–765, 1998.

[70] Z. W. Z. Chi and K. K. Parhi, “Iterative decoding of space-time codes and related implementation issued,” in *Proc. of 2000 IEEE Asimolar Conference*, vol. 1, 2000, pp. 562–566.

VITA

Orhan Gazi received the BS, MS, and Ph.D. degrees in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey in 1996, 2001, and 2007 respectively. His research includes error control coding and signal processing. He is currently employed as an Instructor in Cankaya University, where he delivers lectures in electrical engineering.