University of Alberta

DC-Free Error-Control Codes

by

Fengqin Zhai ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the

requirements for the degree of *Doctor of Philosophy*

Department of *Electrical and Computer Engineering*

Edmonton, Alberta
Fall 2005

Abstract

DC-free codes and error control (EC) codes are widely used in digital transmission and storage systems. To improve system performance in terms of code rate, bit error rate (BER), and low-frequency suppression, and to provide a flexible trade-off between these parameters, this thesis introduces a new class of codes with both dc-control and EC capability. The new codes integrate dc-free encoding and EC encoding, and are decoded by first applying standard EC decoding techniques prior to dc-free decoding thereby avoiding the drawbacks that arise when dc-free decoding precedes EC decoding. The dc-free code property is introduced into standard EC codes through guided scrambling (GS) multimode coding techniques, at the cost of a minor loss in BER performance on the additive white Gaussian noise channel, and some increase in implementation complexity, particularly at the encoder.

This thesis demonstrates that a wide variety of EC codes can be integrated into this dc-free GS-EC structure, including binary cyclic codes, binary primitive BCH codes, Reed-Solomon codes, Reed-Muller codes, convolutional codes, and some capacity-approaching EC codes such as turbo codes, low-density parity-check codes, and product codes with iterative decoding.

Performance of the new dc-free GS-EC codes is presented in terms of both spectral suppression and bit error rate (BER). It is shown that the new codes provide approximately the same suppression of low frequencies as the conventional concatenation of EC codes and dc-free codes, while offering superior BER performance. It is also shown that, on the noisy dc-constrained channel, the new codes result in superior BER performance when compared to EC coding techniques that do not ensure the coded sequence is dc-free, particularly when source data logic values are not equiprobable.

Acknowledgements

I would like to express my thanks to the following people and institutions for their contributions to this work:

- Dr. I. J. Fair, for supervising this project, providing financial support, offering an opportunity to work in the fields of error-control coding and constrained sequence coding, and encouraging me to develop to my full potential;
- the members of the examining committee, for taking the time to review this work;
- the University of Alberta, AUCC Canadian Wireless Telecommunication Association, and iCORE, for financial assistance;
- students in Communications Research Lab and in iCORE wireless Communications Lab at the University of Alberta, for sharing the enjoyable student life and for the technical support.
- my parents and parents-in-law, for their love, understanding, and support throughout my Ph.D. study;

and, especially my husband, Yan Xin, for his love, encouragement, support, and valuable technical discussion.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The requirements for efficient and reliable digital communication systems are greatly increasing in almost all fields and areas. Owing to its efficiency and reliability, coding has found widespread application in practical digital transmission and storage systems. In general, digital transmission systems and digital storage systems are analogous [1]. In transmission systems, the information data is sent to a destination through a transmission channel, while in storage systems the information data is transferred to a user through a recording medium, also called a storage channel. Figure 1.1 shows a block diagram of a basic digital transmission/storage system. The upper blocks indicate the signal transformations from the source to the modulator/writing unit. The lower blocks basically reverse the signal processing order executed by the upper blocks. The data source may represent any of a number of sources of information initiated by a person or a machine. The modulator/writing unit generates the appropriate waveforms that are suitable for transmitting/recording. The transmission/storage channels encountered in the real world distort the transmitted/recorded signal.

Between the data source and modulator/writing unit, three distinct types of encoding procedures can be implemented. A source encoder performs analog-to-digital (A/D) conversion (for an analog source) and removes redundant or unneeded information. Encryption generates secrecy codes that prevent unauthorized users from understanding messages. The channel encoder adds structured redundancy into the encrypted sequence for reliable transmission or storage of information data.

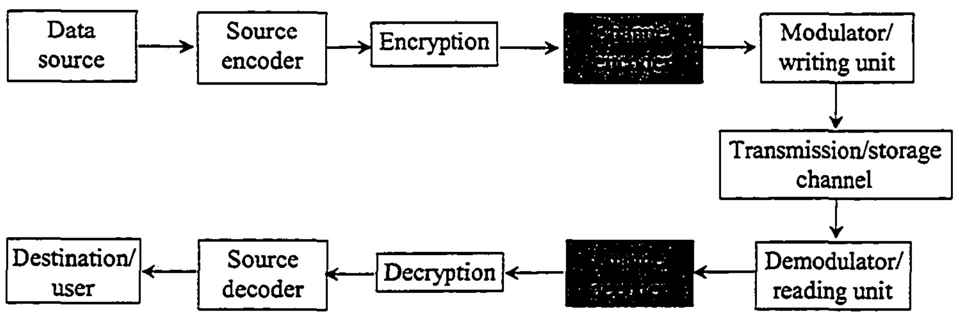


Figure 1.1 Block diagram of a basic digital transmission or storage system.

In most digital systems, channel codes generally include both error control (EC) codes and constrained sequence (CS) codes. EC codes allow the receiver to detect and/or correct errors when the coded sequence is corrupted due to the imperfect channel. CS codes make it possible for the channel-input sequences to satisfy the requirements of the constrained channel and to avoid some types of signal patterns that are known to result in degraded performance in practical systems.

In general, CS codes include dc-free codes and runlength limited (RLL) codes in which the coded sequences have constraints in the frequency domain and in the time domain, respectively. In a dc-free (dc-balanced) coded sequence, its power spectral density (PSD) function has value zero at zero frequency. In an RLL coded sequence, the lengths of consecutive like-valued symbols are limited to a prescribed range.

DC-free codes are commonly used in wired transmission systems to improve performance that is degraded due to the use of coupling components and/or isolating transformers [2], are widely employed in optical recording to allow low-frequency noise caused by dust and fingerprints to be filtered out with minimal loss of signal [3], and have been proposed in wireless systems to assist with the insertion of pilot tones [4].

The conventional method of incorporating both EC coding and CS coding into a digital communication or storage system is through concatenation of an EC code as an outer code and a CS code as an inner code [5], as shown in Figure 1.2 (note that for simplicity, some functional blocks shown in Figure 1.1 are omitted in this figure).

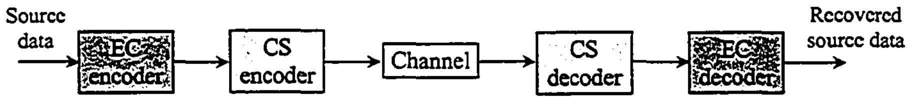


Figure 1.2 Block diagram of conventional concatenation of CS coding and EC coding.

In general, CS decoders exhibit error extension, expect binary input, and output hard decisions. To avoid the impact of error extension during CS decoding on the subsequent EC decoder, and to enable the use of soft-decision information during EC decoding, it is desired that EC decoding precede CS decoding [5]. Concatenation schemes with a partially reversed order of conventional CS coding and EC coding have been proposed in [6], [7], and the performance of these schemes has been evaluated in [5]. Constructions for some integrated binary dc-free EC codes have been proposed in [4], [8] –[12], however these codes have limited EC ability, and/or have limited flexibility regarding tradeoffs between BER and spectral performance. For example, in [8], one specific dc-free block code with minimum distance 4 was synthesized with a lookup table. In [9], dc-free block coset codes were proposed, where construction was limited to binary dc-free coset BCH codes and in general the number of augmenting bits for dc control is limited by the manner in which the codeword length can be factored. It remains to be determined how this coset coding technique can be applied to non-binary codes [1], product codes [13], and low-density parity-check (LDPC) codes [14] in order to generate dc-free EC block codes. DC-free *M*-ary error-correcting codes were introduced in [4], however the spectral performance of these codes diminishes when the number of parity-check bits in the EC code is relatively large. In [10]–[12], consideration focused on dc-free convolutional codes.

1.1 Objective of Thesis

This thesis introduces new general approaches for integrating dc-free codes and EC block/convolutional/turbo codes for highly-efficient and highly-reliable digital transmission and storage systems. The main idea is to map a source word to one or multiple complementary pairs of words through multimode encoding, and to encode these pairs with an EC encoder that preserves the complementary nature of these words. With convolutional and turbo codes, puncturing or flipping is also required to preserve the complementary nature of these words. Multimode selection techniques are then used to select one suitable codeword to generate a dc-free EC sequence. During decoding, the EC decoding is performed before dc-free decoding, therefore, the error performance of the new scheme is essentially determined by the EC codes used. This thesis will demonstrate that many well-established EC codes can be applied to the new coding schemes, including binary cyclic codes, binary primitive BCH codes, Reed-Solomon codes, Reed-Muller codes, convolutional codes, and some capacity-approaching EC codes such as LDPC codes, product codes, and turbo codes with iterative decoding.

1.2 Organization of Thesis

This thesis is arranged as follows. Chapter 2 presents the preliminaries that are necessary to understand later chapters. DC-free codes are introduced, including classification and spectral performance of these codes, and the guided scrambling (GS) technique [15] is discussed. EC codes are overviewed, including block codes, convolutional codes, and codes with iterative decoding. Several schemes for concatenating and integrating CS codes and EC codes are discussed.

Chapter 3 proposes a new dc-constrained noisy channel model through concatenation of a high-pass filter (HPF) and additive white Gaussin noise (AWGN).

Chapter 4 develops the new coding scheme for dc-free GS-EC block codes. EC block codes and some shortened EC block codes are discussed and constructed in order to include these codes in the new coding scheme. The performance of dc-free GS-EC block codes in terms of both PSD and error rate is also presented.

Chapter 5 extends the work in Chapter 4 to integrate GS dc-free codes with trellis-based EC codes such as convolutional codes and turbo codes. For convolutional codes and turbo codes, puncturing or flipping is used to ensure the generation of complementary word pairs for dc control after both GS encoding and EC encoding.

Chapter 6 concludes the thesis by summarizing the concepts and performance of the new schemes for dc-free EC codes, and presents some thoughts for future research.

Chapter 2

Preliminaries

This chapter reviews the basic principles of information, capacity, dc-free codes and EC codes, and discusses several schemes for concatenating and integrating CS codes and EC codes. Some definitions are also given. To assist with the following discussion, let n and n_b denote the lengths of a codeword and a binary codeword, respectively. When the code is binary, $n_b = n$, and when the code is nonbinary over the Galois field GF(2^m), $n_b = m \cdot n$.

2.1 Measure of Information and Information Rate [16], [17]

Uncertainty or entropy is a measure of the information content of an information source. Assume that the source is stationary. Consider a discrete memoryless source such that the independent source symbol X randomly takes on values from the symbol alphabet $S_X = \{x_i,\ i = 0,1,\ldots,n_X - 1\}$, where the probability of symbol x_i is $P(x_i)$, $i = 0,1,\ldots,n_X - 1$. The amount of information carried by the symbol x_i is defined as:

$$I(x_i) \triangleq \log_2\left(\frac{1}{P(x_i)}\right) = -\log_2 P(x_i), \tag{2.1}$$

where the units of information are bits, and if the base of the logarithm is e the units of information are nats.

The entropy of the source symbol X, denoted $H(X)$, is defined as the average amount of information:

$$H(X) = -\sum_{i=0}^{n_X-1} P(x_i)\log_2 P(x_i) \quad \text{[bits/symbol]}. \tag{2.2}$$

It can be shown that $H(X)$ satisfies the inequality:

$$H(X) \le \log_2 n_X, \tag{2.3}$$

where the equality holds when the n_X symbols of S_X are equiprobable.

Now consider the case when the discrete source contains dependence. Let X_j denote the source symbol at time j, and let X_j takes on the symbol x_{i_j} from S_X. Let $\{x_{i_j}, j = 0,1,\ldots,n\}$ denote a message sequence generated by the source. The entropy of the source, denoted $H_\infty(X)$, is defined as:

$$H_\infty(X) \triangleq \lim_{n\to\infty} H(X_n = x_{i_n} \mid X_{n-1} = x_{i_{n-1}},\ldots,X_0 = x_{i_0}) \quad \text{[bits/symbol]}, \tag{2.4}$$

where $H_\infty(X)$ indicates the average amount of information when considering the statistical dependence between symbols in a source message. Note that:

$$0 \le H_\infty(X) \le H(X), \tag{2.5}$$

where $H_\infty(X)$ will equal $H(X)$ when the symbols in the message sequence are independent.

Assume that the source generates the symbols at fixed points in time with interval T_s. The average information rate of the source, R_s, is defined as:

$$R_s \triangleq \frac{H_\infty(X)}{T_s} \text{ [bits/s]}. \tag{2.6}$$

2.2 DC-Free Codes

This section introduces the fundamentals of dc-free codes. The capacity, two performance metrics, the classification and performance of these codes, the GS technique, and PSD evaluation are discussed.

2.2.1 Capacity of DC-Free Codes

The running digital sum (RDS) of a sequence plays an important role in the analysis and design of dc-free codes. Let $\{x_j\}$ denote a coded sequence, where $x_j \in \{-1,1\}$. The RDS of $\{x_j\}$ at the lth time instant, S_l, is defined as:

$$S_l = \sum_{j=0}^{l} x_j. \tag{2.7}$$

It can be shown that if and only if S_l is bounded, the sequence $\{x_j\}$ is guaranteed to have a spectral null at dc [18], [19].

In order to transform an arbitrary sequence into a dc-free sequence, some redundancy needs to be added, and the code rate will be less than 1. The question regarding the maximum code rate, known as the capacity, arises when the RDS of the coded sequence is limited to within a given range.

A constrained channel does not allow input sequence containing prohibited sequences. Let $N(l)$ denote the number of acceptable sequences of length l. The capacity C_s for a constrained channel is defined as [16]:

$$C_s = \lim_{l\to\infty} \frac{\log_2 N(l)}{l} \text{ bits/symbol.} \tag{2.8}$$

Note that when the constrained channel is complicated, it is not easy to find an expression for $N(l)$ to allow direct calculation of the capacity. Since the channel constraints, in general, can be modeled as Markov information sources, the calculation of C_s can based on Markov information sources [16], [19].

Consider an *M*-state discrete random process $\{Z_l, l = 0,1,2,...\}$ where each dependent random variable will take a possible value from the state alphabet $\{s_i, i = 1,...,M\}$, and the dependence is determined by the following conditional probability:

$$P(Z_{l+1} = s_{i_{l+1}} \mid Z_l = s_{i_l}, Z_{l-1} = s_{i_{l-1}}, ..., Z_1 = s_{i_1}, Z_0 = s_{i_0}) = P(Z_{l+1} = s_{i_{l+1}} \mid Z_l = s_{i_l}), \tag{2.9}$$

where $Z_l = s_{i_l}$ indicates that the process is in state s_i at time *l*. Relation (2.9) implies that the conditional distribution of any future state Z_{l+1}, given the past states $Z_0, Z_1, ..., Z_{l-1}$ and the current state Z_l, is independent of the past states and depends only on the current state. Such a random process is known as a Markov chain. A Markov chain can be described by an $M \times M$ transition probability matrix, in which the entries are non-negative and the entries of each row sum to one. An entry in *i*th row and *j*th column of the matrix represents the probability that the process will, when in state s_i, transit into state s_j.

A Markov information source (Markov source in brief) outputs symbols to generate an output sequence (called an encoded sequence) corresponding to movement through the chain [19]. In the Mealy-type Markov source, the generated symbols are a function of the Markov chain, and are given by the labels on the edges between states that describe the Markov chain. The sequences generated by the Markov source are constrained by the characteristics of the Markov chain, enabling some types of constrained sequences to be modelled by this approach. A Mealy-type Markov source is unifilar if for each state the labels assigned to the outgoing edges are different. The entropy of a unifilar Markov source is defined as the weighted summation of the state entropies, which are weighted by the steady-state probabilities of the Markov chain being in each of the states [16], [19].

The connection matrix *F* of an *M*-state source is defined by matrix elements $f_{ij} = n_{ij}$ where n_{ij} is the number of edges exiting state s_i and entering state s_j. Given the connection matrix for a unifilar Markov source, it may be possible to choose transition probabilities to maximize the entropy of the Markov source. A Markov source with these transition probabilities is called maxentropic and the sequences generated by this type of source are called maxentropic sequences. Given the connection matrix of a unifilar Markov source, the maximum entropy of the source, i.e., the capacity C_s for a constrained channel, is [16]:

$$C_s = \log_2 \lambda_{\max} \quad \text{bits/symbol,} \tag{2.10}$$

where $\lambda_{\max}$ is the greatest eigenvalue of the connection matrix.

For a dc-free coded sequence, let $V_{\max}$, $V_{\min}$, and V denote the maximum RDS, the minimum RDS, and the digital sum variation, respectively. Note that $V = V_{\max} - V_{\min} + 1$. The generation of this sequence can be described by a Markov source with one state for each value of the RDS. The maximum entropy of the sequence given V is addressed in [20]. The connection matrix corresponding to the maxentropic Markov source has ones in the upper diagonal and lower diagonal, and zeros elsewhere. With the given parameter V, the capacity $C_s(V)$ of the dc-free coded sequence can be evaluated as [20]:

$$C_s(V) = 1 + \log_2 \cos\frac{\pi}{1+V}, \quad V \geq 3. \tag{2.11}$$

Note that since the variation of the value of V changes the number of allowable sequences, the larger the value of V, the higher capacity that is achievable; the smaller the value of V, the lower capacity that is achievable. Also note that it is not possible to design a code with code rate greater than the capacity.

2.2.2 Performance Metrics of DC-Free Codes

In practice, it is desired that the spectral components around zero frequency in the spectral-null sequence are largely suppressed. Let T, ω, and ω_T denote the symbol duration, radian frequency, and the normalized radian frequency ($\omega_T = \omega T$), respectively. A well-known approximation that provides an indication of the width of the spectral null at dc is given by [21]:

$$2\sigma_s^2 \omega_{T,0} \approx 1 \tag{2.12}$$

where σ_s^2 represents the variance of RDS (sum variance in brief), i.e., $E[S_l^2]$ when the mean value of S_l is zero, and $\omega_{T,0}$ denotes the normalized cutoff frequency at which the PSD of $\{x_j\}$, $H_x(\omega_T)$, equals 0.5 or –3 dB. The normalized cutoff frequency, which is indication of the width of the spectral null, is widely used as a performance metric for dc-free codes [19], [21]. Sum variance of maxentropic dc-free sequences has also been derived in [21].

Recently a performance metric called low-frequency spectrum-weight (LFSW) has been introduced for sequences with a spectral null at dc. It has been shown that at low frequencies the PSD of a dc-free sequence $\{x_j\}$ can be approximated as [22]:

$$\Phi_x(\omega_T) \approx \xi \omega_T^2 \tag{2.13}$$

where ξ denotes the LFSW which indicates the depth of the spectral null at low frequencies. It has been shown that the value of the LFSW [22] equals the zero-frequency content of the continuous PSD of the corresponding RDS sequence $\{S_i\}$.

2.2.3 Classification of DC-Free codes

DC-free codes can be classified as monomode, bimode and multimode codes. DC-free multimode codes have demonstrated advantages over other dc-free codes when both code rate and spectrum performance are considered [23].

A. Monomode Codes

The disparity of a binary codeword is defined as the difference in the number of ones and the number of zeros in the codeword. For example, the 10-bit codewords "1010110001", "1011110011", and "0101000010" have disparity 0, +4, and –4, respectively. In a monomode code, all codewords have zero disparity, and every source word is related to a codeword through a one-to-one mapping. Zero-disparity codewords are possible only if the codeword length is even. Monomode coding is the most straightforward method to generate sequences with a spectral null at zero frequency. The enumerative method [24] and Knuth's method [25] are among the techniques that have been proposed to implement monomode codes.

It has been shown that sum variance of the full-set monomode code can be expressed as [24]:

$$\sigma_s^2 = (1/6)(n+1), \tag{2.14}$$

and that the LFSW of the full-set monomode code equals [22]:

$$\xi = (1/12)n(n+1). \tag{2.15}$$

B. Bimode Codes

Given the same codeword length, a dc-free bimode code [1] can realize higher code rate than a dc-free monomode code. In bimode codes, each source word is, in general, mapped into two codeword candidates. Based upon a given selection criterion and the conditions of the coded sequence at the time of encoding, one of these two candidates is selected as the codeword to be sent to the channel.

An example of a bimode code is the polarity switch code [26] where the codeword is selected from the codeword pairs in order to minimize the absolute RDS value at the end of the transmitted word. In this thesis, this criterion is called the minimum word-end RDS (MRDS) selection criterion. By using this polarity switch code, the accumulated RDS values of the coded sequence can be bounded to be within a finite range, resulting in a dc-free sequence.

The sum variance of polarity switch codes has been derived in [19]:

$$\sigma_s^2 = (1/3)(2n-1)\,. \tag{2.16}$$

Based on a closed-form expression for the approximate PSD of the polarity switch code [26], it can be shown that the LFSW of the polarity switch code can be approximated as:

$$\xi \approx 1.03n^2 - 1.35n - 0.08\,. \tag{2.17}$$

Although bimode codes can result in rate-efficient dc-free codes, when both rate efficiency and spectrum performance based on cutoff frequency are considered, it has been shown that polarity switch codes do not provide performance as good as that of monomode codes at relatively low code rates [19].

C. Multimode Codes

In a multimode code, each source word has multiple representations. A selection criterion, which is critical to code performance, is employed to select the "best" representation from the selection set. In the literature, three main coding approaches have been developed to implement mappings between source words and the corresponding codeword sets in multimode codes: guided scrambling [15], dc-free coset coding [9], and the scrambling of a Reed-Solomon (RS) code [27]. Since guided scrambling is easy to implement and is well-developed [28]–[30], it has been employed as the primary coding technique in the development of dc-free multimode codes [23].

In order to ensure generation of a balanced sequence in dc-free multimode codes, it is required that there exist codewords with zero disparity, or with a different polarity of disparity in each selection set. Furthermore, for better spectrum performance it is desired that each selection set have as many such codewords as possible. Therefore, for a given codeword length, multimode codes offer more possibilities than bimode codes for selection of a "good" codeword from a selection set. The drawback is a loss of rate efficiency. With appropriate selection criteria, however, when both rate and spectrum performance are considered, dc-free multimode codes have been shown to have advantages over dc-free bimode codes [23]. The sum variance and the LFSW of GS dc-free codes will be discussed in Subsection 2.2.5.

2.2.4 Guided Scrambling

Guided scrambling [15] is one form of multimode coding. Its encoding involves augmenting the source words, scrambling the augmented words to form a set of quotients, and selecting an appropriate encoded word from the quotient selection set. Its decoding is very simple, involving unscrambling the received word and discarding the augmenting bits.

The scrambling (division) and unscrambling (multiplication) operations in GS are easily implemented. These arithmetic operations are from the ring of polynomials over the Galois field of order two, referred to as GF(2) [1]. The complexity of GS coding resides in quotient selection. Both the scrambling polynomial and selection criteria affect the power spectrum of the coded sequence.

It is convenient to use polynomial representations for the description of the encoding and decoding processes in GS coding. Let a binary word $\boldsymbol{u}=(u_{n_u-1},\ldots,u_j,\ldots,u_1,u_0)$ of length n_u $(n_u \geq 1)$ be represented by the polynomial:

$$u(x)=u_{n_u-1}x^{n_u-1}+\cdots+u_jx^j+\cdots+u_1x+u_0 \tag{2.18}$$

where $u_j \in \{0,1\}$ is the value of the bit in position j and n_u-1 denotes the most significant position of the word. The encoding and decoding processes of GS are presented in [15]. In each encoding interval, the source word $s(x)$ of length k_{GS} is preceded by all A-bit binary patterns $a_i(x)$, $i=0,1,\cdots,2^A-1$, to generate the augmented words $v_i(x)=a_i(x)x^{k_{GS}}+s(x)$ of length $n_{GS}=A+k_{GS}$. A quotient selection set Q is obtained by scrambling these augmented words. The quotient $q_i(x)$ of length n_{GS}, corresponding to the augmented word $v_i(x)$ is:

$$q_i(x)=Q_{d(x)}\left[x^Dv_i(x)\right] \tag{2.19}$$

where the operator $Q_{d(x)}[\cdot]$ denotes the formation of a quotient through modulo-2 division of its argument by the scrambling polynomial $d(x)$ of degree D $(D\geq 1)$:

$$d(x)=x^D+d_{D-1}x^{D-1}+\cdots+d_1x+1, \tag{2.20}$$

where $d_j=1$ or 0 for $j=1,2,\ldots,D-1$. Note that without loss of generality, the most and least significant coefficients of scrambling polynomials are defined to be 1 [28]. Based on a given selection criterion, a quotient $q(x)$ is selected as a codeword to be sent to the channel. Note that if the number of augmenting bits is one, a bimode code is generated; if the number of augmenting bits is larger than one, a multimode code is generated.

Let $v(x)$ denote the augmented word that generated the selected quotient $q(x)$. In GS decoding, the decoded augmented word $\hat{v}(x)$ is obtained by multiplying the received word $\hat{q}(x)$, which is the hard-decision output of the demodulator, by the scrambling polynomial $d(x)$ used in the encoder and discarding the D least significant bits of the product $\hat{q}(x)d(x)$:

$$\hat{v}(x) = Q_{x^D}\left[\hat{q}(x)d(x)\right]. \tag{2.21}$$

If there are no transmission errors at the output of the channel, $\hat{q}(x) = q(x)$, $\hat{v}(x) = v(x)$, and the GS decoder can recover the source word correctly by removing the augmenting bits from $\hat{v}(x)$. Otherwise, let $\hat{q}(x) = q(x) + e(x)$ where $e(x)$ is an error pattern at the output of the channel. In this error pattern, a coefficient of "1" for the term x^j means that there is an error at position j. From (2.21), it follows that:

$$\begin{aligned}\hat{v}(x) &= Q_{x^D}\left[(q(x) + e(x))d(x)\right] \\ &= v(x) + Q_{x^D}\left[e(x)d(x)\right].\end{aligned} \tag{2.22}$$

Removing the A augmenting bits from $\hat{v}(x)$ results in the decoded word $\hat{s}(x)$. Relation (2.22) demonstrates that at the output of the GS decoder, one error at the input of the decoder may be extended to multiple errors during GS decoding, and this extension is upper bounded by the weight of $d(x)$, denoted w_d. When GS is used in a noisy channel, to prevent large error extension, it is desirable to choose a scrambling polynomial with small weight [28]. The smallest possible weight of $d(x)$ is two.

Let $d_e(x)$ denote an even-weight scrambling polynomial with degree not greater than A, and let $d_{2,A}(x)$ denote the weight-two scrambling polynomial of degree A (i.e., $d_{2,A}(x) = x^A + 1$). It has been shown that use of $d_e(x)$ ensures the generation of complementary codeword pairs in the quotient selection set [31]. Since in general a scrambling polynomial with a higher degree (up to degree A) results in larger suppression of low frequencies, and scrambling polynomials of the same degree do not result in a significant difference in spectral performance [32], it is proposed that $d_{2,A}(x)$ be used in the new GS-EC coding schemes introduced in this thesis to yield good spectral performance and minimum error extension during GS decoding.

Due to the linearity of GS encoding, the same set of GS quotients can also be generated based on the addition of one GS quotient with predetermined additive patterns. For $i = 0,1,...,2^A - 1$, let $h_i(x)$ denote the predetermined additive patterns. In the original form of GS encoding, $q_i(x)$ is generated by:

$$q_i(x) = Q_{d(x)}[x^D(a_i(x)x^{k_{GS}} + s(x))]. \tag{2.23}$$

From (2.23), it follows that:

$$\begin{aligned} q_i(x) &= Q_{d(x)}[s(x)x^D] + Q_{d(x)}[a_i(x)x^{k_{GS}+D}] \\ &= q_0(x) + h_i(x), \end{aligned} \tag{2.24}$$

from which it is clear that $h_i(x)$ are dependent only on the augmenting bit patterns and the scrambling polynomial, and can therefore be predetermined. Let GS_0 denote an encoder that maps a source word $s(x)$ to a GS quotient $q_0(x)$ by scrambling $s(x)$ with $d(x)$. The remaining GS quotients can be constructed by adding $q_0(x)$ to $h_i(x)$, $i = 1,2,...,2^A - 1$. This equivalent form for GS encoding is called additive GS.

Shift registers can be used for the implementation of GS. GS encoding is performed through modulo-2 division of the augmented source word by the scrambling polynomial, and GS decoding is implemented through modulo-2 multiplication of the received word by the scrambling polynomial.

Figures 2.1 shows an example for the implementation of modulo-2 division. Let $V(x) = x^7 + x^4 + x^3$ ($\boldsymbol{V} = 10011000$, with length 8) be divided by $d(x) = x^3 + x + 1$ ($\boldsymbol{d} = 1011$). Note that the degree of $d(x)$ is $D = 3$. Assume that the initial register contents are set to be zero. For this implementation, the most significant bit of V is shifted into the register first, and starting from the $D+1 = 4$-th shift, the quotient appears serially at the output. The remainder resides in the register after the 8-th shift. For this example, the quotient is 10100 and the remainder is $\boldsymbol{r} = 100$.

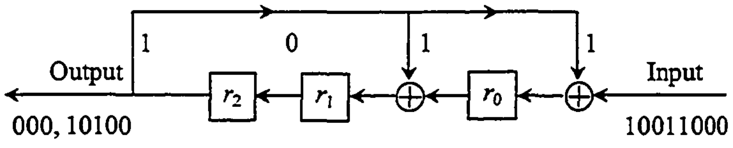


Figure 2.1 Shift register for modulo-2 division.

For this example, the operational steps of the shift register are shown in Table 2.1. Note that the quotient is read from top to bottom and the remainder is read from left to right.

Table 2.1 Operational steps of the shift register for the modulo-2 division

Input	Shift number	Register contents $r_2\, r_1\, r_0$	Output	Quotient
1	0	0 0 0	-	
0	1	0 0 1	0	
0	2	0 1 0	0	
1	3	1 0 0	0	
1	4	0 1 0	1	1
0	5	1 0 1	0	0
0	6	0 0 1	1	1
0	7	0 1 0	0	0
-	8	1 0 0 (remainder)	0	0

Figure 2.2 illustrates implementation of modulo-2 multiplication. Let $q(x)=10100$ ($\boldsymbol{q}=10100$) be multiplied by $d(x)=x^3+x+1$ ($\boldsymbol{d}=1011$, degree $D=3$). The product of $q(x)$ and $d(x)$ is generated by the operation of the shift register in the figure. Note that the shift register is initially set to zero, input $\boldsymbol{q}$ is followed by D zeros, and the most significant bit of $\boldsymbol{q}$ is shifted into the register first. For this example the product is 10011100. Note that the operational steps of the shift register are shown in Table 2.2, and the product is read from top to bottom.

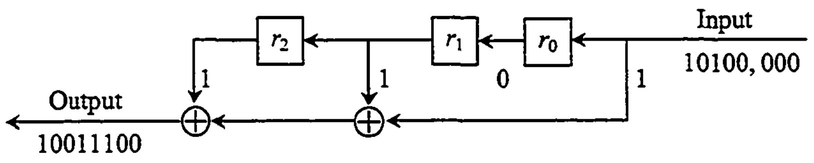


Figure 2.2 Shift register for modulo-2 multiplication.

Table 2.2 Operational steps of the shift register for the modulo-2 multiplication

Input	Shift number	Register contents $r_2\, r_1\, r_0$	Output	Product
1	0	0 0 0	-	
0	1	0 0 1	1	1
1	2	0 1 0	0	0
0	3	1 0 1	0	0
0	4	0 1 0	1	1
0	5	1 0 0	1	1
0	6	0 0 0	1	1
0	7	0 0 0	0	0
-	8	0 0 0	0	0

In the encoding and decoding procedures discussed above, the source words are considered independently, i.e., the encoder and decoder shift registers are cleared between each codeword. This form of coding is known as block GS coding. Continuous GS [28] is obtained by updating the encoder division registers with the remainder associated with the previously selected quotient resulting in a transmitted sequence that consists of a continuous quotient. In the decoding of this type of GS, the multiplication is continuous and the shift registers are not cleared before decoding each received codeword. Note that these two GS schemes are statistically equivalent if all source words have the same probability [28]. Only the block GS approach is considered in the proposed new method of GS-EC coding developed in this thesis.

2.2.5 Performance of GS DC-Free Codes [32]

There exist four coding parameters for GS coding, which include the GS quotient length n_{GS}, the number of augmenting bits A, the scrambling polynomial $d(x)$, and the selection criterion. These coding parameters jointly determine the dc-free performance of GS coding.

In [23], several selection criteria for dc-free multimode codes are developed and evaluated. In particular, the minimum squared weight (MSW) criterion is considered. The MSW criterion selects the word which results in minimum S_l^2 of the coded sequence where S_l is as defined in relation (2.7). The MSW criterion is shown to exhibit excellent dc-free performance [23].

With even-weight scrambling polynomials, a source word is mapped to multiple complementary GS quotients, which ensures that it is possible to generate good suppression of low frequency for the coded sequence. If the other three parameters are fixed, a scrambling polynomial with a higher degree ($D \leq A$) generally offers larger suppression of low frequencies, and scrambling polynomials of the same degree do not result in a significant difference in spectral performance [32]. With the consideration of both spectral performance and the error extension of GS decoding, the MSW selection criterion and weight-two scrambling polynomials $d_{2,A}(x)$ are considered in the following analysis of the spectral performance for GS dc-free codes.

Maxentropic dc-free sequences have information content that equals the channel capacity. Let $\tilde{V}$ denote a moderately large RDS variation on the order of at least ten or more. Accurate expressions for estimating the sum variance $\sigma_{s,m}^2$ and LFSW ξ_m of a maxentropic dc-free sequence with capacity $C(\tilde{V})$ are developed in [19] and [22], respectively:

$$\sigma_{s,m}^2 \approx 0.2326/(1 - C(\tilde{V})), \tag{2.25}$$

$$\xi_m \approx 0.2225/(1 - C(\tilde{V}))^2. \tag{2.26}$$

Expressions (2.25) and (2.26) demonstrate that the sum variance and the LFSW values of maxentropic dc-free sequences are mainly determined by the capacity, i.e., the maximum code rate. As with the maxentropic dc-free sequence, for GS dc-free codes with codeword length n_{GS} using $d_{2,A}(x)$ and the MSW selection criterion, the sum variance $\sigma_{s,GS}^2$ and the LFSW ξ_{GS} values of the coded sequences are related to the GS code rate $R_{GS} = 1 - (A/n_{GS})$ as shown by the following two expressions [32]:

$$\sigma_{s,GS}^2 \approx S_A \cdot [0.2326/(1 - R_{GS})], \tag{2.27}$$

$$\xi_{GS} \approx L_A \cdot [0.2225/(1 - R_{GS})^2], \tag{2.28}$$

where $S_A = \sigma_{s,GS}^2 / \sigma_{s,m}^2$ and $L_A = \xi_{GS} / \xi_m$, for $A = 1,2,...,8$, are factors that depend only on the number of augmenting bits A. Values for these factors are given in [32] are reproduced here in Table 2.3. Note that relations (2.27) and (2.28) are accurate only when R_{GS} is relatively large, i.e., for code rates above 0.9 which is often the case. From Table 2.3, it can be seen that when A is large, S_A and L_A approach 1, which indicates that the performance of GS dc-free codes approximates that of the maxentropic dc-free sequence.

Table 2.3 Factors S_A and L_A

A	1	2	3	4	5	6	7	8
S_A	2.9471	2.2023	1.7669	1.5003	1.3367	1.2289	1.1554	1.1033
L_A	7.6977	4.8579	3.1686	2.2402	1.7079	1.3818	1.1893	1.0591

2.2.6 Power Spectral Density Evaluation of Block Coded Signals

Signals can be classified as either deterministic or random. The signals discussed in this thesis are random signals. The output of a channel noise generator and speech signals are examples of random processes. The theory of probability forms the basis for describing random processes. A random process is said to be stationary in the strict sense if all its statistics are not affected by a shift of the time origin. A random process is said to be wide-sense stationary (WSS) if at least its mean and autocorrelation function are independent of a shift of the time origin.

A random process which is at least WSS can generally be classified as a power signal having a PSD [33]. For a WSS random process the PSD and autocorrelation form a Fourier transform pair. The PSD is particularly useful in communications system applications because it depicts how the power of a signal is distributed in the frequency domain.

Statistical averages of random digital signals cannot be time invariant, but often have an element of periodicity, and can be, at best, cyclostationary. For example, in a synchronous storage system [3], the signals that drive a write head during recording are pulse amplitude modulation (PAM) signals. The PAM signal $X(t)$ is of the form:

$$X(t)=\sum_{j=0}^{\infty}x_j g(t-jT) \tag{2.29}$$

where the channel input discrete sequence $\{x_j\}$ over bipolar alphabet $\{-1,+1\}$ can be random. The pulse shape $g(t)$ is deterministic and in the system considered in [3] is a rectangular pulse of duration T and unit amplitude.

Assume that the sequence $\{x_j\}$ is a WSS discrete random process. Then the channel-input PAM signal is wide-sense cyclostationary with period T [19], [34]. The mean $m_X(t)$ and autocorrelation $R_X(t,t+\Delta t)$ of $X(t)$ are:

$$m_X(t)=E[X(t)]=m_X(t+T) \tag{2.30}$$

$$R_X(t,t+\Delta t)=E[X(t)X(t+\Delta t)]=R_X(t+T,t+\Delta t+T) \tag{2.31}$$

where $E[\cdot]$ denotes expectation (statistical averaging). The periodic nature of the mean and autocorrelation functions are characteristics of a wide sense cyclostationary process. Although a cyclostationary process is non-stationary, it can be treated as though it were a stationary process by averaging the periodic statistical parameters of the cyclostationary process over one period, or, in effect, taking an average of averages over the interval of periodicity. Therefore, the average power spectral density of the cyclostationary stochastic process can be obtained [35]:

$$\Phi_X(\omega)=\frac{1}{T}\Phi_x(\omega)|G(\omega)|^2 \tag{2.32}$$

where $G(\omega)$ is the Fourier transform of the pulse $g(t)$, and $\Phi_x(\omega)$ is the power spectral density of the sequence $\{x_j\}$ which is the Fourier transform of the autocorrelation of the discrete random process $\{x_j\}$:

$$\Phi_x(\omega)=\sum_{k=-\infty}^{\infty}R_x(k)e^{-ik\omega T}=\sum_{k=-\infty}^{\infty}E[x_l x_{l+k}]e^{-ik\omega T} \tag{2.33}$$

where $i=\sqrt{-1}$. When the pulse shape of $g(t)$ is rectangular, the PSD of the channel-input signal is:

$$\Phi_X(\omega)=T\cdot\sum_{k=-\infty}^{\infty}E\left[x_l x_{l+k}\right]e^{-ik\omega T}\cdot\left(\frac{\sin(\omega T/2)}{\omega T/2}\right)^2. \tag{2.34}$$

Relation (2.32) indicates that power spectrum shaping of PAM signals $X(t)$ can be realized by designing the pulse shape $g(t)$ and by controlling the statistical correlation of symbols in the sequence $\{x_j\}$. However, for various communication channels, it is not possible to modify the standard pulse shape of channel input signals [19]. Instead, in order to shape the power spectrum of signals $X(t)$, the statistics of the sequence $\{x_j\}$ must be controlled.

The Fourier transform of discrete-time signals is discussed in [36]. Unlike the Fourier transform of continuous-time signals, the Fourier transform of discrete-time signals is periodic with period $2\pi/T$. The frequency components of a discrete-time signal are unique over the frequency interval $(-\pi/T,\pi/T)$, therefore, it is enough to evaluate the power spectrum of the symbol sequence for values of the normalized frequency ωT within the interval $(-\pi,\pi)$, or equivalently for values of the normalized frequency fT within $(-0.5,0.5)$. For real-valued signals, the power spectrum has even symmetry, and evaluation can be limited to values of ωT within $(0,\pi)$, or values of fT within $(0,0.5)$.

In this thesis, block-oriented codes are investigated. In the encoding of block-oriented codes, k source bits are encoded into n code bits per codeword. Given a stationary source, the coded symbol sequence is a wide-sense cyclostationary process with period nT [37], [38]. By considering a finite-state sequential machine (FSSM) model of the encoders, the method for calculating power spectra of block coded sequences reported in [37], [38] is widely used. However, when n is large, spectral analyses of block codes become impractical, and simulation of power spectra is considered.

Several methods to estimate the power spectrum of a random signal are discussed in [36]. Stationary random processes have finite average power and hence can be characterized by their power spectral density. In practice, only a finite-duration sequence $\{x_m\}$, $0\le m\le N-1$, is available for estimation of the spectrum of the signal, and a single realization of the random process is used. The corresponding estimate of the power spectral density is:

$$P_x(f) = \frac{1}{N}\left|\sum_{m=0}^{N-1} x_m e^{-j2\pi fm}\right|^2 = \frac{1}{N}|X(f)|^2, \tag{2.35}$$

where $X(f)$ is the Fourier transform of the sequence $\{x_m\}$. Relation (2.35) denotes a well-known PSD estimation method called the periodogram. By this method the estimated spectrum is asymptotically unbiased, but the variance of the estimated spectrum does not decay to zero as N goes infinity. Bartlett introduced an approach to reduce the variance in the periodogram [39]. In the Bartlett method, the N-point sequence is divided into some nonoverlapping segments with equal length, the periodogram for each segment is calculated, and the periodograms of the segments are averaged to obtain the power spectrum estimate. Welch improved the Bartlett method to allow overlapping of segments and to add windowing to the data segments [40].

The frequency resolution in the spectral estimate is determined by N, the length of the data record. Note that zero padding provides a method for interpolating the values of the measured spectrum at more frequencies, but it does not improve the frequency resolution. When a signal record over a finite time interval is statistically stationary, the longer the data record, the better the estimate that can be obtained. However, if the signal statistics are nonstationary, the data record cannot be arbitrarily long, but should be determined by the rapidity of the time variations in the signal statistics [36]. Note that in this thesis, in order to obtain correctly simulated spectra, the sequence length, the segment length, and the codeword length satisfy the constraints that the segment length is the multiple of the codeword length, and the sequence length is the multiple of the segment length.

2.3 Error-Control Codes

Error-control codes play an important role in transmission and storage systems. Using powerful error control coding is very helpful in obtaining sufficient quality for these systems. This section discusses the channel capacity, commonly used EC codes, and the error rate performance evaluation.

2.3.1 Channel Capacity

Section 2.1 introduced concepts related to how much information a source can generate. This subsection discusses the channel capacity of a noisy channel.

The communication channel offers a connection between the modulator (or writing unit) and the demodulator (or reading unit). Generally, physical channels include wireline, fiber optic, wireless, underwater acoustic, and storage channels [34]. In order to assist the design of the communication systems, mathematical models, which reflect the most important characteristics of the transmission medium, are constructed. In practice, the simple additive noise channel model is used frequently. Physically, the electronic components, amplifiers, and interference at the receiving side of the transmission and storage systems are sources of additive noise. If the noise has resulted from electronic components and amplifiers, it is usually described as thermal noise, which is characterized statistically as a Gaussian noise process. The resulting mathematical model for the channel is the well known additive Gaussian noise channel. This channel model applies to many transmission or storage channels and is mathematically tractable. It is also easy to include channel attenuation with this model. If the signal suffers attenuation in transmission or storage through the additive noise channel, the received signal is:

$$r(t) = a \cdot s(t) + n(t), \tag{2.36}$$

where $s(t)$ is the transmitted signal, $n(t)$ is the additive random noise process, and the a is an attenuation factor.

There are several channel models used in modeling and analysis of a coded system. A discrete memoryless channel (DMC) is characterized by n_X discrete input symbols, n_Y discrete output symbols, and a set of conditional probabilities (or transition probabilities). A binary symmetric channel (BSC) is a special case of a DMC ($n_X = n_Y = 2$) where the corresponding conditional probabilities are symmetric. If the output symbols of the DMC channel are not discrete, i.e., if they can be any real values, the discrete-input continuous-output channel is obtained. The most important channel of this type is the additive white Gaussian noise (AWGN) channel. In Chapter 3, the noisy dc-constrained channel will be modeled by use of a simple RC high-pass filter concatenated with an AWGN channel.

When the demodulator output consists of a continuous value or quantized approximations greater than the number of channel input values, the demodulator is said to make soft decisions and the corresponding decoding is called soft-decision decoding. Decoders which output symbol values from the same alphabet as the channel input symbols are called hard-decision decoders.

Consider the capacity of the DMC [17]. In order to assist the discussion, define the following variables:

- input alphabet of the channel $X = \{x_i, i = 0,1,\ldots,n_X - 1\}$,
- output alphabet of the channel $Y = \{y_j, j = 0,1,\ldots,n_Y - 1\}$,

- probability of input alphabet $P(x_i), i = 0,1,\ldots,n_X - 1$,
- probability of output alphabet $P(y_j), j = 0,1,\ldots,n_Y - 1$,
- transition probability $P(y_j \mid x_i), i = 0,1,\ldots,n_X - 1, j = 0,1,\ldots,n_Y - 1$,
- conditional probability $P(x_i \mid y_j), i = 0,1,\ldots,n_X - 1, j = 0,1,\ldots,n_Y - 1$,
- Input entropy $H(X) \triangleq -\sum_{i=0}^{n_X-1} P(x_i)\log_2 P(x_i)$ [bits/symbol],
- Output entropy $H(Y) \triangleq -\sum_{j=0}^{n_Y-1} P(y_j)\log_2 P(y_j)$ [bits/symbol],
- Joint entropy $H(X,Y) \triangleq -\sum_{i=0}^{n_X-1}\sum_{j=0}^{n_Y-1} P(x_i,y_j)\log_2 P(x_i,y_j)$ [bits/(symbol pair)],
- Conditional entropy $H(Y \mid X) \triangleq -\sum_{i=0}^{n_X-1}\sum_{j=0}^{n_Y-1} P(x_i,y_j)\log_2 P(y_j \mid x_i)$ [bits/symbol],
- Conditional entropy $H(X \mid Y) \triangleq -\sum_{i=0}^{n_X-1}\sum_{j=0}^{n_Y-1} P(x_i,y_j)\log_2 P(x_i \mid y_j)$ [bits/symbol].

Assume that $P(x_i)$ and $P(y_j \mid x_i)$ are known, and based on these two probabilities, the following three probabilities can be obtained:

- $P(x_i,y_j) = P(y_j \mid x_i)\cdot P(x_i)$,
- $P(y_j) = \sum_{i=0}^{n_X-1} P(x_i,y_j)$,
- $P(x_i \mid y_j) = P(x_i,y_j)/P(y_j)$.

Channel capacity is defined as the maximum information flow across a noisy channel. Define mutual information between X and Y through the channel as [16], [17]:

$$I(X;Y) \triangleq H(X) - H(X \mid Y) \text{ [bits/symbol]}, \tag{2.37}$$

where $I(X;Y)$ indicates the average information flow through the channel. The value of the capacity of a channel, C_n, is the maximum information flow through this channel [16], [17]:

$$C_n = \max I(X;Y). \tag{2.38}$$

Since, for a given channel, $I(X;Y)$ can be maximized through variation of the input symbol probabilities, the channel capacity is:

$$C_n = \max_{P(x_i)} I(X;Y). \tag{2.39}$$

Assume a BSC, and let $P(x_0) = q$. Therefore, $P(x_1) = 1-q$. Let $P(y_1 \mid x_0) = b$, and then $P(y_0 \mid x_0) = 1-b$, $P(y_0 \mid x_1) = b$, and $P(y_1 \mid x_1) = 1-b$, where b denotes the error probability. Let $\rho(b) = -b\log_2 b - (1-b)\log_2(1-b)$. It can be shown that:

$$I(X;Y) = \rho(q + b - 2qb) - \rho(b), \tag{2.40}$$

and that:

$$C_n = \max_{P(x_i)} I(X;Y)$$
$$= I(X;Y)|_{q=0.5}$$
$$= 1-\rho(b). \tag{2.41}$$

Therefore, for a BSC, the capacity depends on the error probability. For example, given $b = 0.05$, $C_n = 0.714$ bits/symbol.

The channel coding theorem [16], originally proved by Claude Shannon, states that by use of channel codes it is possible to achieve reliable transmission, with an arbitrary small probability of error, if the amount of information transmitted per symbol is less than the channel capacity.

Considering the BSC example above, although on average 5% of the transmitted binary digits will be demodulated in error, Shannon's channel coding theorem indicates that there exist channel codes that will encode the input symbols such that the input entropy being less than 0.714 bits/symbol, and allow this error rate to be reduced to as low as possible.

Consider the discrete-time continuous-amplitude AWGN channel. Every T_s seconds, the source emits a symbol selected from a possibly infinite alphabet, and Gaussian noise is introduced during the communication. Let X and Y be discrete-time continuous-amplitude input and output signals of the channel, respectively. Let P and σ^2 denote the average input power and average noise power, respectively. Define the entropy of X to be:

$$H(X) = \int_{-\infty}^{\infty} f_X(x)\log_2(1/f_X(x))dx, \tag{2.42}$$

where $f_X(x)$ is the probability density function of X. In a similar fashion, $H(Y)$, $H(X,Y)$, and $H(Y,X)$ can be defined.

Recall that for the discrete memoryless source, the maximum entropy of the source is obtained when the source symbols are equiprobable. It can be shown that for the continuous case, the maximum entropy of the source is obtained when the source is Gaussianly distributed with variance σ_X^2 [41], [17]:

$$H(X)|_{\max} = 0.5\log_2(2\pi e\sigma_X^2). \tag{2.43}$$

Note that $P = \sigma_X^2$. Since the source and the additive noise are both independently Gaussianly distributed, the channel output Y is also Gaussianly distributed with variance $\sigma_Y^2 = \sigma_X^2 + \sigma^2$.

Based on the maximization of the mutual information with respect to the input distribution, it can be shown that the capacity is [41], [17]:

$$\begin{aligned} C_n &= 0.5\log_2(1+\sigma_X^2/\sigma^2) \\ &= 0.5\log_2(1+P/\sigma^2) \ \text{[bits/channel use]}. \end{aligned} \tag{2.44}$$

If the channel is bandlimited with bandwidth B_l Hz, the Gaussian noise power is N_0B_l Watts, where N_0 denotes the single-sided power spectral density of the Gaussian noise. Therefore, the capacity of the bandlimited AWGN channel is:

$$C_n = 0.5\log_2(1+P/(N_0B_l)) \ \text{[bits/channel use]}. \tag{2.45}$$

Based on the Nyquist sampling theorem, in order to precisely recover a signal from its sampled version the sampling rate must be greater than or equal to $2B_l$. If the discrete-time signal X is viewed as a sampled version of a continuous-time and continuous-amplitude signal, transmitting one sample will form one "channel use". The capacity is [41]:

$$\begin{aligned} C_n &= 0.5\log_2(1+P/(N_0B_l)) \ \text{[bits/channel use]}\cdot 2B_l \ \text{[channel use/second]} \\ &= B_l\log_2(1+P/(N_0B_l)) \ \text{[bits/second]}. \end{aligned} \tag{2.46}$$

Note that in order to approach this capacity the transmitted signal must statistically approximate white noise.

Recall that R_s denotes the average information rate of the source. Therefore, the average amount of energy for transmitting a bit of information is $E_b = P/R_s$ [J/bit]. If error-control coding is added into the system, with code rate $R_{EC} < 1$, redundancy is introduced into the system. In order to keep the same rate for transmission of information, the rate of transmission must increase by a factor of $1/R_{EC}$. Therefore, after coding, the transmission rate is R_s/R_{EC}, and the bandwidth is at least $B_l = 0.5(R_s/R_{EC})$ according to the Nyquist sampling rate. Therefore, the capacity can be expressed as [17]:

$$\begin{aligned} C_n &= 0.5\log_2(1+P/(N_0B_l)) \\ &= 0.5\log_2(1+(E_bR_s)/(N_0B_l)) \\ &= 0.5\log_2(1+2R_{EC}E_b/N_0) \ \text{[bits/channel use]}. \end{aligned} \tag{2.47}$$

Since the capacity can be used to bound the probability of decoding error, the following inequality holds [17]:

$$R_{EC} \le C_n/(1-H_b(e)), \tag{2.48}$$

where $H_b(e) = \rho(P_b(e))$ is the error entropy corresponding to erroneous binary digits after decoding, and where $P_b(e)$ is the probability of decoded bit error. Introducing (2.47) into (2.48) yields [17]:

$$R_{EC} \le \frac{\log_2(1+2R_{EC}E_b/N_0)}{2(1-\rho(P_b(e)))}, \tag{2.49}$$

where with a given code rate R_{EC}, equality holds for the minimum possible error probability. Considering the equality of (2.49) yields:

$$R_{EC} = \frac{\log_2(1+2R_{EC}E_b/N_0)}{2(1-\rho(P_b(e)))},$$

$$E_b/N_0 = 10\log_{10}\frac{2^{2R_{EC}}((P_b(e))^{2R_{EC}P_b(e)}\cdot(1-P_b(e))^{2R_{EC}(1-P_b(e))}-1)}{2R_{EC}} \quad \text{[dB]}. \tag{2.50}$$

Based on (2.50), Figure 2.3 illustrates the relationship between the bit error probability $P_b(e)$ and E_b/N_0 for a given code rate. For a given code rate, only the area above the associated curve is achievable.

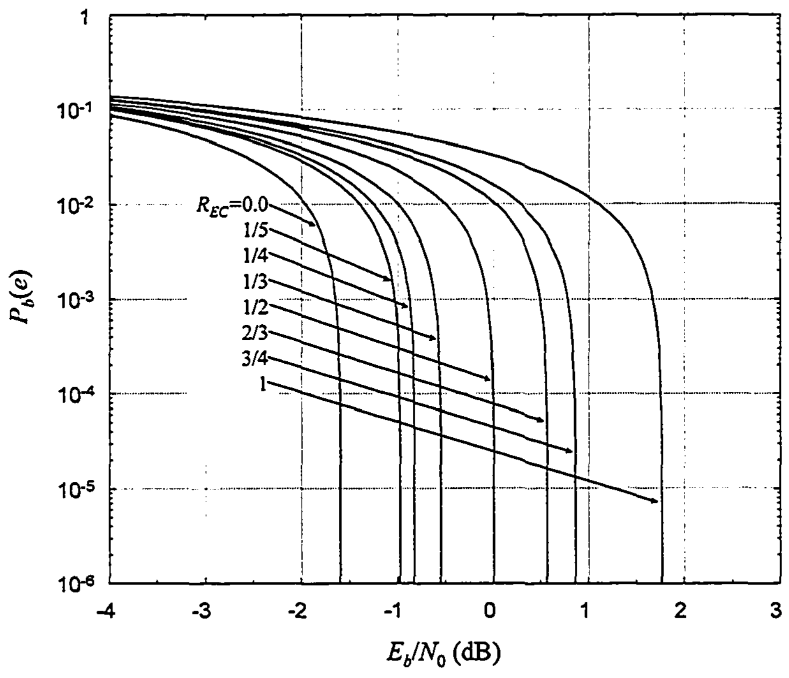


Figure 2.3 Relationship between the bit error probability $P_b(e)$ and E_b/N_0 with a given code rate for unconstrained Gaussian channel.

2.3.2 Classification of Error-Control Codes

Essentially, there are three classes of error control codes in use today including block codes, convolutional codes, and codes which use block or convolutional code structures with iterative decoding.

A. Block Codes

In block coding [1], the information is first grouped into blocks (or frames) of k symbols per block by the encoder. The message word is denoted by $\boldsymbol{u} = (u_{k-1}, \ldots, u_1, u_0)$. The encoder transforms each message word $\boldsymbol{u}$ to a codeword of length n denoted by $\boldsymbol{c} = (c_{n-1}, \ldots, c_1, c_0)$, where $\boldsymbol{u}$ and $\boldsymbol{c}$ are composed of symbols from the same alphabet, and $k < n$. In these codes, $n-k$ redundant symbols are added to the message words to obtain the codewords and to provide the error control capability of the (n,k) block codes. The code rate R_{EC} of these codes is defined as $R_{EC} = k/n$. If the code is binary, there are 2^k different possible message words and the same number of corresponding codewords. The encoding is memoryless in the sense that a codeword only depends on the input message word in that particular encoding interval.

Linear block codes are a subclass of block codes. In a linear block code, the vector sum of two codewords is still a valid codeword. The weight of a code vector is the number of nonzero coordinates in the code vector. The Hamming distance between two code vectors is the number of different coordinates between these two code vectors. The minimum distance, $d_{\min}$, of a block code *C*, is the minimum Hamming distance between all distinct pairs of codewords in *C*. If the code is linear, the all-zero codeword exists and $d_{\min}$ can be obtained by finding the minimum weight of the non-zero codewords. A block code with $d_{\min}$ is able to correct all $t = \lfloor (d_{\min} - 1)/2 \rfloor$ or fewer errors per word, where $\lfloor (d_{\min} - 1)/2 \rfloor$ denotes the largest integer not greater than $(d_{\min} - 1)/2$.

Cyclic codes, which were first proposed by Prange in 1957, form a subclass of linear codes [42], and have proven to be very important practical codes. Encoding and decoding of cyclic codes can be implemented by high speed shift registers. The famous Hamming codes (research work in 1947 and publication in 1950) [43], Golay codes (1949) [44], BCH codes (1959 and 1960) [45], [46], and Reed-Solomon codes (1960) [47] belong to the family of cyclic codes. Hamming codes are single error-correcting binary codes, and are perfect codes [1], [34] in that exactly t or fewer errors can be corrected with each word. The $(23,12)$ Golay code is the only multiple error-correcting binary perfect code, and can correct three or fewer random errors per word. BCH codes, which are a generalization of Hamming codes for multiple error correction, are powerful random error-correcting codes. The most important subset of nonbinary BCH codes is the set of Reed-Solomon codes.

Reed-Muller (RM) codes (1954), which are multiple error-correcting codes, are one of the best understood families of block codes [48], [49]. These codes were applied frequently during 1950's and 1960's [50]. They have good error correction properties if the codeword length is not too large. An advantage of the Reed-Muller codes is that they have a maximum likelihood decoding algorithm which can be implemented with very fast hardware.

Common descriptions for linear block codes and cyclic codes, and concepts and properties of fields are introduced below.

1) Matrix Description

An (n,k) linear block code can be described in terms of its generator matrix $\boldsymbol{G}$ and its parity-check matrix $\boldsymbol{H}$, where $\boldsymbol{G}$ is an $k \times n$ matrix and $\boldsymbol{H}$ is an $(n-k) \times n$ matrix. A codeword $\boldsymbol{c}$ can be easily generated by $\boldsymbol{c} = \boldsymbol{u} \cdot \boldsymbol{G}$. Also a codeword $\boldsymbol{c}$ satisfies $\boldsymbol{c} \cdot \boldsymbol{H}^T = 0$, where T denotes the transpose. Let $\boldsymbol{r}$ be the received word when the codeword $\boldsymbol{c}$ is transmitted over a noisy channel. Then the decoder can calculate the syndrome s by $s = \boldsymbol{r} \cdot \boldsymbol{H}^T$. If $s = 0$, it is an indication that $\boldsymbol{r}$ is a codeword, and if $s \neq 0$, it is an indication that $\boldsymbol{r}$ is not a codeword and that errors have been detected. The syndrome offers information regarding errors in the received word and therefore can be used for error correction.

2) Polynomial Description [1]

In a linear code *C*, if every cyclic shift of a code vector is also a code vector in *C*, this code is called a cyclic code. Cyclic codes can be conveniently described using polynomial notation similar to that introduced in the description of guided scrambling. The components of a code vector $\boldsymbol{v} = (v_{n-1}, v_{n-2}, \ldots, v_1, v_0)$ are treated as the coefficients of a code polynomial, $v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \cdots + v_1 x + v_0$. The terms "code vector" and "code polynomial" are interchangeable. In an (n,k) cyclic code, there exists a unique code polynomial of degree $n-k$, $g_C(x) = x^{n-k} + g_{n-k-1}x^{n-k-1} + \ldots + g_2 x^2 + g_1 x + 1$, which is called the generator polynomial of the code. In an (n,k) cyclic code, every code polynomial is a multiple of $g_C(x)$ and a binary polynomial of degree less than n is a code polynomial if and only if it is a multiple of $g_C(x)$.

3) Fields and Some Properties

In order to understand the definition of the generator polynomial for BCH codes, the concept of a field and some of its properties are described [1].

Generally speaking, a field is a set of elements in which elements can be added, subtracted, multiplied, and divided (by all elements except zero), and yield a unique result from that same set of elements. Addition and multiplication must obey the communicative, associative, and distributive laws. For example, the set of real numbers is a field with an infinite number of elements. Fields with a finite number of elements are used in coding theory. Finite fields are also called Galois fields. For any prime *p* there exists a finite field of *p* elements, denoted GF(*p*). In coding research the binary field GF(2) plays an important role. For any positive integer *m*, it is also possible to extend the prime field GF(*p*) to a field GF(p^m).

GF(2^m) can be constructed as follows. Let α be a primitive element from GF(2^m). Then the set $F^* = \{0, 1, \alpha, \alpha^2, \cdots, \alpha^{2^m-2}\}$ is a Galois field of 2^m elements, GF(2^m). Note that $\alpha^{2^m-1} = 1$. To understand multiplication of nonzero elements, let *i*, *j*, and *r* be integers such that $0 \le i, j, r \le 2^m - 2$. The multiplication of two nonzero elements has the following results:

$$\begin{cases} \alpha^i \cdot \alpha^j = \alpha^{i+j}, & \text{if } i+j < 2^m - 1 \\ \alpha^i \cdot \alpha^j = \alpha^{i+j} = \alpha^{(2^m-1)+r} = \alpha^r, & \text{if } i+j \ge 2^m - 1. \end{cases}$$

Note that $0 \cdot 0 = 0$ and $0 \cdot \alpha^i = 0$.

The nonzero elements of F^* can be represented by $2^m - 1$ distinct nonzero polynomials of α over GF(2) with degree *m*-1 or less. These polynomials are obtained by setting $p(\alpha) = 0$, where $p(x)$ is a primitive polynomial of degree *m* over GF(2) which generates GF(2^m). Addition of elements is carried out by using the polynomial representations of the elements. Since the coefficients of the polynomial representation are binary, there is a binary vector representation available for the corresponding element. Table 2.4 shows the three different representations for the elements of GF(2^4) generated by $p(x) = x^4 + x + 1$. By setting $p(\alpha) = \alpha^4 + \alpha + 1 = 0$, the identity $\alpha^4 = \alpha + 1$ is obtained for generating the polynomial representations and the corresponding binary vector representations of the field elements. For example,

$$\alpha^4 = \alpha + 1 \rightarrow 0011,$$
$$\alpha^5 = \alpha \cdot \alpha^4 = \alpha(\alpha + 1) = \alpha^2 + \alpha \rightarrow 0110,$$
$$\alpha^6 = \alpha \cdot \alpha^5 = \alpha(\alpha^2 + \alpha) = \alpha^3 + \alpha^2 \rightarrow 1100$$

Table 2.4 Three representations for the elements of GF(2^4) generated by $p(x) = x^4 + x + 1$

Power representation	Polynomial representation	Binary vector representation
0	0	0000
1	1	0001
α	α	0010
α^2	α^2	0100
α^3	α^3	1000
α^4	$\alpha + 1$	0011
α^5	$\alpha^2 + \alpha$	0110
α^6	$\alpha^3 + \alpha^2$	1100
α^7	$\alpha^3 + \alpha + 1$	1011
α^8	$\alpha^2 + 1$	0101
α^9	$\alpha^3 + \alpha$	1010
α^{10}	$\alpha^2 + \alpha + 1$	0111
α^{11}	$\alpha^3 + \alpha^2 + \alpha$	1110
α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$	1111
α^{13}	$\alpha^3 + \alpha^2 + 1$	1101
α^{14}	$\alpha^3 + 1$	1001

Let β be an element in GF(2^m). The following properties regarding the conjugate and minimal polynomial of an element in GF(2^m) exist [1]:

- Let $b(x)$ be a polynomial with coefficients from GF(2). If β is a root of $b(x)$, then for any $i > 0$, β^{2^i} is also a root of $b(x)$. The element β^{2^i} is called a conjugate of β.
- The elements of GF(2^m) form all the roots of $x^{2^m} + x$. Any element β in GF(2^m) is a root of the polynomial $x^{2^m} + x$ and β is a root of a polynomial over GF(2) with a degree less than 2^m. Let $\phi(x)$ be the polynomial of smallest degree over GF(2) such that $\phi(\beta) = 0$. This $\phi(x)$ is called the minimal polynomial of β, is irreducible, and divides $x^{2^m} + x$. The minimal polynomials of the zero element 0 and the unit element 1 of GF(2^m) are x and x+1, respectively.

Table 2.5 shows the conjugate roots and minimal polynomials in GF(2^4).

Table 2.5 Conjugate roots and minimal polynomials in GF(2^4) generated by $p(x) = x^4 + x + 1$.

Conjugate roots	Minimal polynomials
0	x
1	$x+1$
$\alpha,\alpha^2,\alpha^4,\alpha^8$	x^4+x+1
$\alpha^3,\alpha^6,\alpha^9,\alpha^{12}$	$x^4+x^3+x^2+x+1$
α^5,α^{10}	x^2+x+1
$\alpha^7,\alpha^{11},\alpha^{13},\alpha^{14}$	x^4+x^3+1

B. Convolutional Codes

Convolutional codes were introduced by Elias in 1955 as an alternative to block codes [51]. They differ from block codes in that their encoders involve memory from one encoding interval to the next. In any given time interval the n' encoder output digits depend not only on the k' input digits in that encoding interval but also on inputs during the previous ν intervals, where ν is called the memory of the convolutional code. Typically, n' and k' are much smaller integers than those in block codes. An (n',k',ν) convolutional code can be implemented through the use of a linear shift register.

The constraint length K of a convolutional code is defined as the maximum number of output bits affected by a single input bit. Note that $\nu = K - 1$. A convolutional encoder can be described by a state diagram, where the states of the encoder are defined as its shift register contents. The state diagram can be expanded in time into a trellis diagram. Among several decoding algorithms for convolutional codes, the Viterbi algorithm (VA) (1967) is the most famous [52]. The VA essentially performs maximum likelihood sequence decoding and limits decoding computational complexity by taking advantage of the structure of the code trellis diagram. However, the VA is not able to directly evaluate the *a posteriori* probability (APP) for each decoded bit. The APP is used to make optimum bit-by-bit decisions in a digital communication system. The BCJR algorithm (1974) can generate the desired APP information [53]. In most applications of interest, the performance of the BCJR and the VA for decoding of convolutional codes is effectively identical, and therefore as a result of its higher complexity, the BCJR algorithm did not attract much attention until the invention of turbo codes in 1993 [54].

The term minimum distance is from the world of block coding. In convolutional coding, because the codes are also linear, there is no loss in generality in defining $d_{\min}$ to be the minimum distance between each of the codeword sequences and the all-zero sequence; however, with convolutional codes, the term minimum free distance, denoted d_{free}, is usually used instead of minimum distance. In these codes, the Viterbi or BCJR decoders use the entire received codeword, or at least a substantial portion of the received sequence, to decode a single bit. Assuming that the all-zero input sequence is transmitted, paths of interest are paths of any length that diverge from and remerge onto the zero state and do not return to the zero state anywhere in between. Sequences corresponding to these paths, called self-terminating sequences, can be regarded as error events. The minimum free distance is the distance between all code vectors in the set of all arbitrarily long paths that diverge from and remerge onto the all-zero sequence.

Convolutional codes can be fashioned into a block code structure by forcing the encoder back to the zero state at prescribed intervals. When $k'=1$, forcing the encoder into the zero state requires the insertion of ν redundant bits into the soured data. The benefit of this structure is the ability to decode the received sequence on a block-by-block basis; the drawback is the additional redundancy required to terminate each block.

Convolutional codes have been used widely in telecommunications systems because of their capability to yield good coding gains on the additive white Gaussian channel (AWGN) for target bit error rates (BER) of around 10^{-5}.

C. Codes with Iterative Decoding

Three types of codes with iterative decoding are discussed below: turbo codes, LDPC codes, and product codes.

In 1993, Berrou *et al.* proposed turbo codes [54] whose performance on the AWGN channel has been shown to perform within a few tenths of a dB of the capacity limit at a BER of 10^{-5}. They are a new class of EC codes, originally based on the use of two recursive systematic convolutional (RSC) codes in parallel. The novelty of turbo codes was the use of RSC codes, a random interleaver, and iterative decoding using soft information. Because turbo codes are parallel concatenated convolutional codes, they are also called PCCCs. Using the same functional blocks, namely convolutional encoders and interleavers, serially concatenated convolutional codes (SCCCs) were proposed in 1996 [55]. Since the BCJR algorithm, used in the decoding of turbo codes and SCCCs, is an optimum symbol-by-symbol maximum *a posteriori* (MAP) decoding approach for linear block codes that minimizes symbol error probability, it is also called the MAP algorithm. The Log-MAP algorithm [56], which works in the logarithmic domain and has equivalent performance to the MAP algorithm, was developed to reduce the computational complexity of the MAP algorithm. Based on the Log-MAP algorithm, the additive soft-input soft-output (A-SISO) algorithm was proposed in [57]. The A-SISO algorithm is very general since it allows continuous decoding of the received sequence, applies to multilevel symbols, and permits parallel edges between each pair of states.

Turbo codes unarguably represent the most important development in coding theory in recent years, and their basic idea has been extended to other forms of code concatenations. Turbo codes have been accepted as a coding standard for third generation (3G) wireless communications systems like CDMA2000 [58] and WCDMA [59], and for satellite and deep space applications such as the new CCSDS (Consultative Committee for Space Data Systems) telemetry channel coding standard [60].

LDPC codes were originally proposed by Gallager in 1962 [61], however, there was little further work on LDPC codes for the next 30 years. In 1981, Tanner extended Gallager's decoding algorithm to the more general case and introduced a useful bipartite graph known as the Tanner graph [62]. Renewed interest in LDPC codes was triggered by the introduction of turbo codes and the rediscovery of LDPC codes by both MacKay and Neal [63] and Wiberg [64] independently in 1996.

LDPC codes are block codes whose parity-check matrix $\boldsymbol{H}$ is sparse, i.e., $\boldsymbol{H}$ has mostly zeros and only a small number of ones. The codes are usually constrained by the row weight and column weight of $\boldsymbol{H}$. In a regular LDPC code, each row of $\boldsymbol{H}$ is of fixed weight w_r and each column of the $\boldsymbol{H}$ is of fixed weight w_c. In an irregular LDPC code, the weights of the columns and rows are determined according to some distribution. It has been reported that irregular LDPC codes provide better performance than regular LDPC codes [65].

For decoding LDPC codes, either hard or soft decision decoding can be employed. Gallager presented two iterative probabilistic decoding algorithms, the bit-flip algorithm for iterative hard decision decoding and a probabilistic decoding algorithm known as the sum-product algorithm for iterative soft decision decoding. The probabilistic decoding algorithm is essentially the same as the belief propagation algorithm [66] and is more computationally intensive than iterative hard decision decoding which has low complexity. However, soft decision decoding provides significantly better performance than hard decision decoding. For binary codes, the sum-product algorithm [63], [14] can be implemented more efficiently in the logarithmic domain, in which case it is called the log sum-product algorithm (Log-SPA) [67].

In many communication and storage channels, mixed random and burst errors occur. Interleaving is an effective way to combat these mixed errors [1]. Generally, product codes [68] and concatenated codes [69] are capable of correcting random errors and burst errors. An interleaver is explicitly included in the concatenated codes, while it is implicit in product codes. Therefore product codes can be thought of as a special case of concatenated codes with a block interleaver.

Elias introduced product codes in 1954. Let C_1 and C_2 be (n_1,k_1) and (n_2,k_2) linear codes, respectively. Then an (n_1n_2,k_1k_2) linear product code can be obtained as depicted in Figure 2.4. If the codes C_1 and C_2 have minimum distance $d_{\min 1}$ and $d_{\min 2}$, respectively, the resulting product code has minimum distance $d_{\min 1}d_{\min 2}$ and is capable of correcting $\lfloor (d_{\min 1}d_{\min 2}-1)/2 \rfloor$ errors.

A linear code that is capable of correcting all error bursts up to length l_b is said to have burst error correcting capability l_b. Let l_{b1} and l_{b2} be the burst error correcting capabilities of code C_1 and C_2, respectively. The burst error correcting capability of the corresponding product code is at least the maximum of n_1l_{b2} and n_2l_{b1} [1]. At the same conference where Berrou *et al.* first introduced turbo codes, Lodge *et al.* introduced the iterative MAP decoding of product and concatenated codes [70], demonstrating very good decoding performance. The solution is based on the trellis of the block codes and is limited to short block codes. In 1994, Pyndiah *et al.* introduced the near optimum decoding of product codes [71] by proposing an iterative decoding scheme where the constituent decoders use the Chase type-II algorithm [72]. Product codes with iterative decoding are also called block turbo codes or turbo product codes. Note that product codes with iterative decoding provide one of the best solutions in terms of BER performance and complexity at high code rate [71].

Conventional product codes can be viewed as a concatenated codes with a block interleaver. Note that product codes can also be constructed with other interleavers such as a random interleaver.

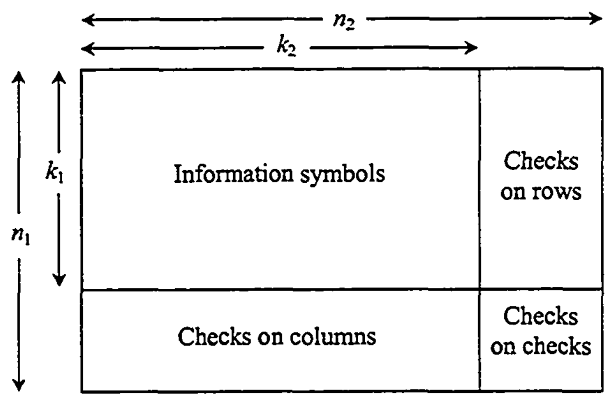


Figure 2.4 Construction of product codes.

2.3.3 Error Rate Performance Evaluation

As illustrated in Figure 1.1, demodulator follows the transmission channel. When the demodulator output consists of a continuous value or quantized approximations greater than the number of channel input values, the demodulator is said to make soft decisions and the corresponding decoding is called soft-decision decoding. Decoders which output symbol values from the same alphabet as the channel input symbols are called hard-decision decoders.

In a coded system, the error correcting performance is measured by the bit error rate (also called the probability of bit error), symbol error rate (where multiple bits are represented by each coded symbol), and frame error rate (also called block error rate or word error rate) versus E_b / N_0 in decibels (dB).

Positive coding gain is defined as the reduction in the required E_b / N_0 (dB) to achieve the desired error performance of the coded system over an uncoded system with identical modulation. Generally, coding gain is obtained only after an E_b / N_0 threshold has been exceeded. This threshold is code dependent.

2.4 Prior Art of DC-Free EC Codes

As was shown in Figure 1.2, in conventional concatenation of CS coding and EC coding, CS decoding proceeds EC decoding. The disadvantages of this conventional approach include that:

- since a CS code, in general, has weak EC capability, or no EC capability at all, the CS decoder may result in large error propagation before EC decoding.

- since most CS codes employ hard-decision decoding, this limits the use of soft decision decoding in the subsequent EC decoder.

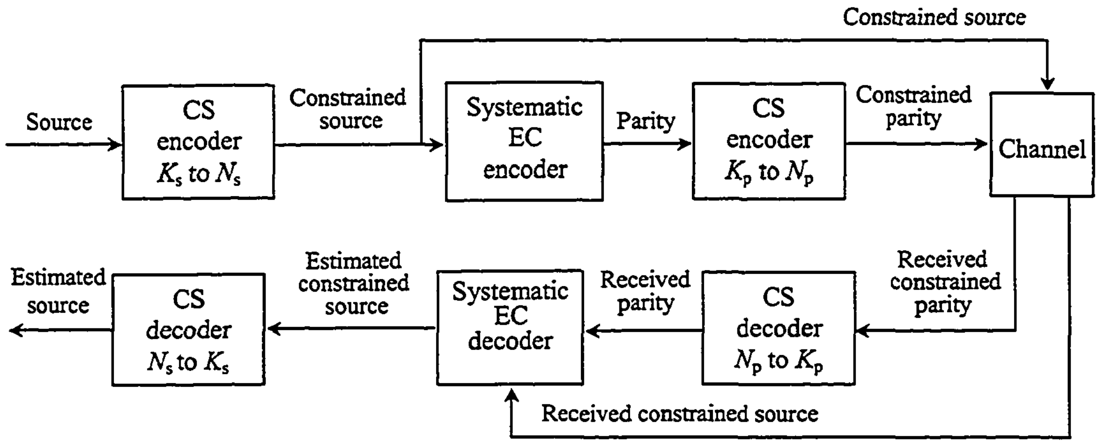
In order to avoid the problem of large error propagation caused by CS decoding, to increase the code efficiency by allowing the use of long codewords (a greater code rate can be obtained by using a longer codeword [7]), and to allow use of a soft decoding algorithm in EC decoding, it is desirable that at the receiver EC decoding precedes CS decoding.

In the following, several schemes for concatenating and integrating constrained codes and error control codes are discussed.

2.4.1 Concatenation Schemes with a Partially Reversed Order

A. Bliss' Scheme

Figure 2.5 illustrates Bliss' scheme [6], which partially reverses the order of CS coding and EC coding. Source words of length K_s are forwarded to the CS encoder to generate constrained source words of length N_s. These constrained source words are passed to the systematic EC encoder as well as directly to the channel since they comprise the systematic portion of the EC codewords and they satisfy the channel constraints. The EC encoder generates a parity sequence of length K_p which is then encoded by the CS encoder to obtain the constrained parity of length N_p that adheres to the prescribed constraints. The constrained parity is then forwarded to the channel.



Constrained source
Source
CS encoder K_s to N_s
Constrained source
Systematic EC encoder
Parity
CS encoder K_p to N_p
Constrained parity
Channel
Estimated source
CS decoder N_s to K_s
Estimated constrained source
Systematic EC decoder
Received parity
CS decoder N_p to K_p
Received constrained parity
Received constrained source

Figure 2.5 Bliss' concatenated scheme.

At the receiver site, the received constrained parity is decoded first, and as a result, is susceptible to error extension that depends on the type of constrained code and the value of K_p [5], [6]. Note that as the code rate of the EC codes increases, it is expected that the impact of the error propagation encountered during constrained parity decoding will decrease.

B. Immink's Scheme

Immink noticed a weakness that exists in Bliss' approach: there is an expansion in the length of the input words to the EC encoder in Bliss' method compared to the conventional method [7]. The larger codewords are less efficient in their use of parity to protect the information transmitted over the channel, and may require more decoding time. Also if the error control codes are nonbinary codes, such as Reed-Solomon codes[1], some design freedom for the codes, such as block lengths, may be lost [7]. Immink introduced lossless compression into this approach to overcome the problem mentioned above. Figure 2.6 shows the scheme proposed by Immink.

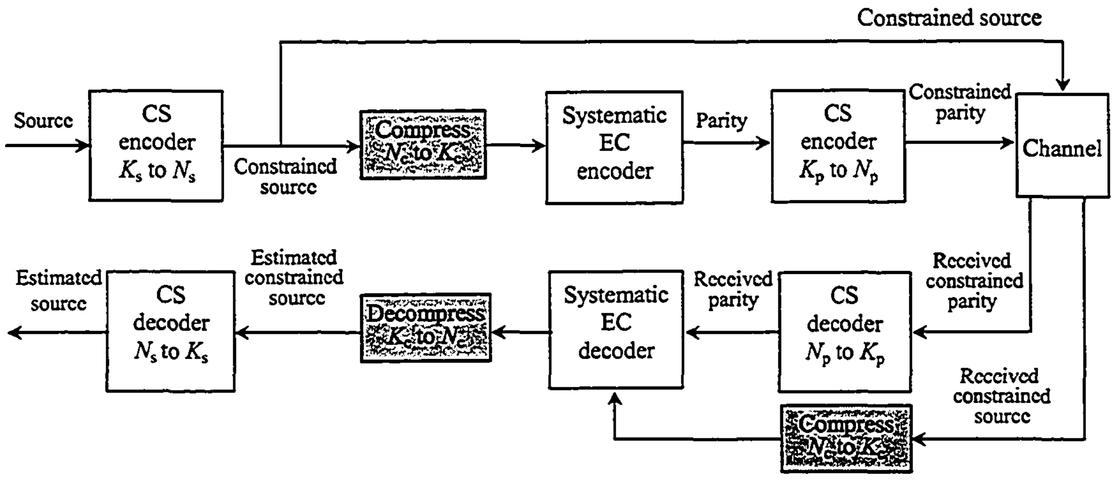

Constrained source
Source
CS encoder K_s to N_s
Constrained source
Compress N_c to K_c
Systematic EC encoder
Parity
CS encoder K_p to N_p
Constrained parity
Channel
Estimated source
CS decoder N_s to K_s
Estimated constrained source
Decompress K_c to N_c
Systematic EC decoder
Received parity
CS decoder N_p to K_p
Received constrained parity
Received constrained source
Compress N_s to K_s

Figure 2.6 Immink's lossless compression scheme.

As shown in Figure 2.6, a lossless compression block (compressing N_c bits to K_c bits) is introduced before the EC encoder. Before the EC decoding, this same compression step needs to take place in order to generate the appropriate sequence for the EC decoder. In this compression step, special measures can be taken to avoid error propagation if nonbinary EC codes are used. After EC decoding, decompression is required.

2.4.2 Integrated Scheme with DC-Free Block Coset Codes

Deng and Herro investigated a class of block coset codes with disparity and run-length constraints [9]. The codes they constructed simultaneously satisfy the dc constraints and the error-correcting requirement. In developing their approach they considered and analyzed only binary BCH codes. In this thesis, the codes they constructed are called dc-free coset BCH codes.

In order to construct a dc-free coset BCH code from a binary BCH code with codewords of length $n = 2^p - 1$ and error-correcting capability t, two odd integers A_c and n_p are selected which satisfy $A_c \cdot n_p = n$ and $n_p \geq 2t + 1$, where A_c denotes the number of augmenting bits for dc control, and n_p denotes the sub codeword length for flipping a section of the codeword. The following steps are then required to obtain a dc-free coset code generator matrix ***G***:

- Start from a common binary nonsystematic BCH code generator matrix $\boldsymbol{G}_{comm}$ with generator polynomial $g_{BCH}(x)$. Each row of $\boldsymbol{G}_{comm}$ consists of cyclically shifted coefficients of $g_{BCH}(x)$.
- Obtain a modified generator matrix $\boldsymbol{G}_{modify}$ from $\boldsymbol{G}_{comm}$ by replacing the first A_c rows of $\boldsymbol{G}_{comm}$ with the coefficients of $z_1(x) = x^{(n_p - 1)A_c} + \cdots + x^{2A_c} + x^{A_c} + 1$, $z_2(x) = x z_1(x)$, $\cdots$, $z_{A_c}(x) = x^{A_c - 1} z_1(x)$.
- Construct the desired ***G*** through permutation of the columns of $\boldsymbol{G}_{modify}$ such that the first A_c rows of ***G*** are vector representations of the coefficients of $c_1(x) = x^{n_p - 1} + \cdots + x^2 + x + 1$, $c_2(x) = x^{n_p} c_1(x)$, $\cdots$, $c_{A_c}(x) = x^{(A_c - 1)n_p} c_1(x)$. Note that ***G*** is non-systematic.

During encoding, appropriate augmenting bits are introduced into the first A_c positions of the source word in order to "flip" portions of the codeword and achieve dc balance. Based on this encoding method for the dc-free coset BCH code, it can be shown that this code is similar to a polarity switch code of length n_p. Note that if the Berlekamp-Massey algorithm is used to decode the dc-free coset BCH code, error extension cannot be avoided since the source word is recovered from the decoded codeword by solving equations based on the non-systematic generator matrix. This scheme is limited by the use of binary codes, by limitations on the block length *n*, and by encoding through a matrix calculation.

2.4.3 Integrated Schemes Based on Convolutional Codes

A. Scheme of Wadayama and Vinck

The scheme proposed by Wadayama and Vinck based on convolutional codes [11] is related to the work of [9] based on binary block codes. Wadayama and Vinck use the semi-infinite generator matrix of convolutional codes, in which a convolutional code is treated as a linear block code [1], for the construction of the new codes. The scheme is mainly based on: (*i*) additive encoding using a binary linear code, (*ii*) upper and lower bounds on the RDS for an additive encoder, and (*iii*) splitting a convolutional code into infinite sequences of a linear block code called a window code. The class of additive encoders can be regarded as a subclass of guided scrambling. Redundancy is required for the dc-free constraint.

The direct sum decomposition of a binary linear code, in which a generator matrix is split into two sub-matrices, is important in this scheme. Soft decision decoding is possible with this coding technique. Although a cascaded structure with the order of the RDS control encoder, the convolutional encoder, the channel, the Viterbi decoder, and the RDS control decoder was given, the two encoding procedures can be integrated because the RDS control encoder can generate an RDS control vector as well as a convolutional codeword. A decomposition matrix is defined for the RDS control. Because this decomposition matrix is also directly related to the RDS control decoder, it requires special design to avoid error extension. When this matrix is generated from a row permutation of an identity matrix, no error extension occurs for this scheme. The overall code rate is small for this scheme.

B. Chiu's Scheme

The coding scheme proposed by Chiu [10] is obtained by the direct sum of a convolutional code C_1 and a dynamically searched code sequence from another convolutional code C_2. The information sequence is first encoded by the code C_1 and then the Viterbi algorithm is used to search over the code C_2 for a code sequence so that the resulting direct sum of both code sequences satisfies the dc constraint. The branch metrics used by the VA are dependent on the selection criteria. The constructed code C_b is a nonlinear subcode of a convolutional code $C_{12} = C_1 \oplus C_2$. The time-invariant trellis of C_b in general requires more states than those for the minimum trellis [73] of C_{12}, therefore, a Viterbi decoder that operates on the minimal trellis of C_{12} is recommended on the receiving side. To recover the information sequence, matrix calculations are required and limited error propagation results after EC decoding.

One advantage of this scheme is that no additional redundancy is required for introducing the dc constraint. However, the selection of convolutional codes is limited and the distance of the constructed codes is generally smaller than that of the optimum convolutional codes [1].

Chapter 3

HP-AWGN Channel

Although dc-free coded sequences are required in transmission and storage systems, a simple channel model for noisy dc-free channels has not been reported in the literature. Therefore, in this chapter, a simple dc-constrained channel model is developed by use of a first-order RC high-pass filter (HPF), and a noisy dc-constrained channel model is obtained by concatenating a HPF with AWGN.

3.1 RC High-Pass Model

Consider the use of a rectangular waveform with channel symbol duration T and amplitude +1 to represent logic ones and amplitude −1 to represent logic zeros. In order to simplify notation, let time and frequency be normalized by the channel symbol duration T and the channel symbol rate $1/T$, respectively. Accordingly, let $t_T = t/T$, $f_T = fT$, and $\omega_T = \omega T$ denote the normalized time and normalized frequencies.

A simple high pass filter is proposed to model systems that do not pass low frequencies. A first-order RC high-pass filter and its effect of on a binary signal are depicted in Figures 3.1 and 3.2, respectively.

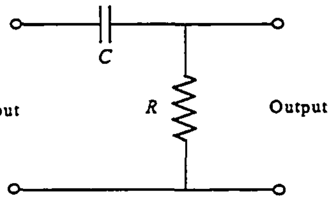


Figure 3.1 First-order RC high-pass filter.

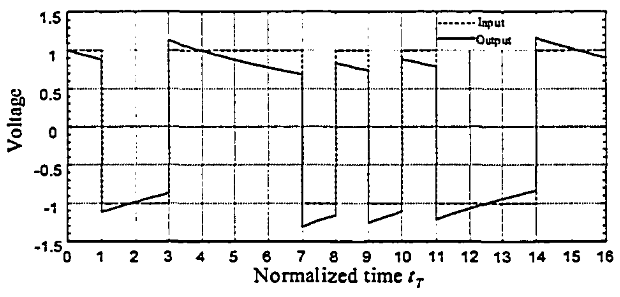


Figure 3.2 Input and output of the high-pass filter.

A version of this chapter has been published in part in *Proceedings of Ninth International Conference on Communication Systems*, 2004. pp. 451–455. F. Zhai, Y. Xin, and I. J. Fair.

From Figure 3.2, it can be seen that there is a change in amplitude of the output signal at the start of each symbol interval when the symbol in that interval is different from the previous one, and that the magnitude of the change is 2. It is also clear that the output signal does not change in value at the start of a symbol interval if the symbol in that interval is identical to the previous one. During each symbol interval, the output decays exponentially towards zero; the rate of decay is dependent on the time constant of the RC circuit (the normalized time constant $\tau_T = RC/T$), the symbol duration, and the value of the signal at the start of the interval. Figure 3.3 shows the output within one symbol interval when the start value is 1, for normalized time constants between 1 and 64.

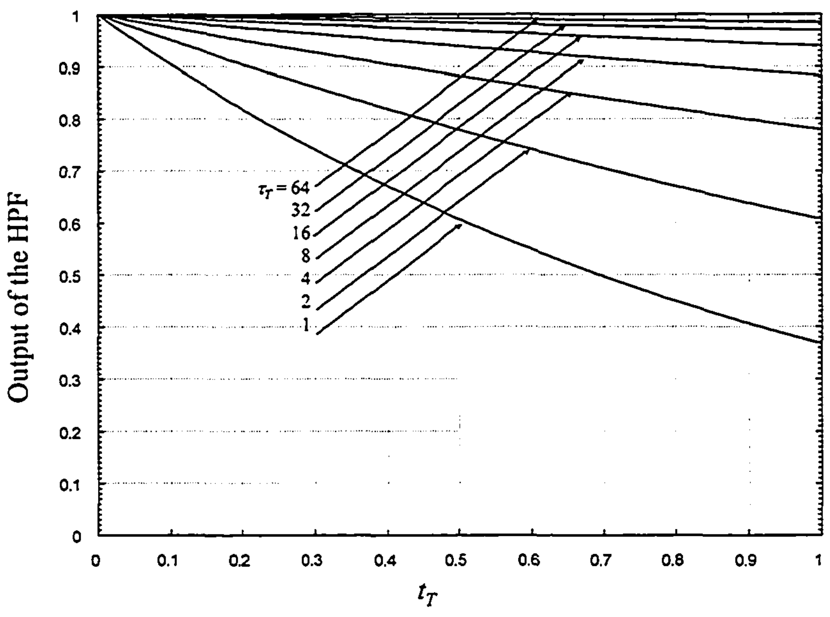


Figure 3.3 Output of the HPF within one symbol interval for various values of τ_T.

Let $V_{s,l}$ and $V_{e,l}$ be the voltages at the start and the end of the lth interval ($l \geq 0$), respectively. During this interval, the output of the filter is $V(t) = V_{s,l}e^{-(t_T - l)/\tau_T}$ where $t_T \in [l, (l+1)]$. At the end of the interval, $V_{e,l} = V_{s,l}e^{-1/\tau_T}$. At the start of the next interval, $V_{s,l+1}$ takes on one of the values:

$$V_{s,l+1} = \begin{cases} V_{e,l} & \text{, if the logic values are the same,} \\ V_{e,l} + 2 & \text{, if the logic values change from -1 to 1,} \\ V_{e,l} - 2 & \text{, if the logic values change from 1 to -1.} \end{cases} \tag{3.1}$$

Let $H_{HP}(f_T)$ denote the frequency response of the HPF. It is straightforward to show that the magnitude squared of the frequency response of the HPF is:

$$|H_{HP}(f_T)|^2 = \frac{(2\pi f_T \tau_T)^2}{1+(2\pi f_T \tau_T)^2}. \tag{3.2}$$

Let $x(t)$ and $y(t)$ be the random input signal and the output signal of the HPF, respectively. The PSD of the output process $S_{yy}(f_T)$ is related to the PSD of the input $S_{xx}(f_T)$ by:

$$S_{yy}(f_T) = |H_{HP}(f_T)|^2 \, S_{xx}(f_T). \tag{3.3}$$

It has been shown [22] that at low frequencies the PSD of a dc-free sequence $\{x_j\}$ can be approximated as $\Phi_x(\omega_T) \approx \xi\omega_T^2$, where $\omega_T = 2\pi f_T$ and $\xi = 0.5\Phi_x^{(2)}(0)$ is determined by the second derivative of $\Phi_x(\omega_T)$ at zero frequency. The parameter LFSW ξ indicates the depth of the spectral null at low frequencies. Since there exists a spectral null at dc in the transfer function of the HPF, the concept of LFSW can be extended to the HPF. Let $\Psi(\omega_T) = |H_{HP}(\omega_T)|^2$ and define the LFSW of the HPF to be $\xi_{HP} = 0.5\Psi^{(2)}(0)$. It is straightforward to show that:

$$\begin{aligned} \xi_{HP} &= 0.5\Psi^{(2)}(\omega_T)\Big|_{\omega_T=0} \\ &= \frac{\tau_T^2 - 3\tau_T^4\omega_T^2}{(1+\tau_T^2\omega_T^2)^3}\Bigg|_{\omega_T=0} \\ &= \tau_T^2. \end{aligned} \tag{3.4}$$

Therefore at low frequencies:

$$|H_{HP}(f_T)|^2 \approx \tau_T^2 4\pi^2 f_T^2. \tag{3.5}$$

Figure 3.4 depicts $|H_{HP}(f_T)|^2$ and its low frequency approximation for various values of f_T and τ_T. From this figure, it is clear that the HPF can be used to model the restrictions imposed by a dc constrained channel where flexibility is available through selection of the value of τ_T.

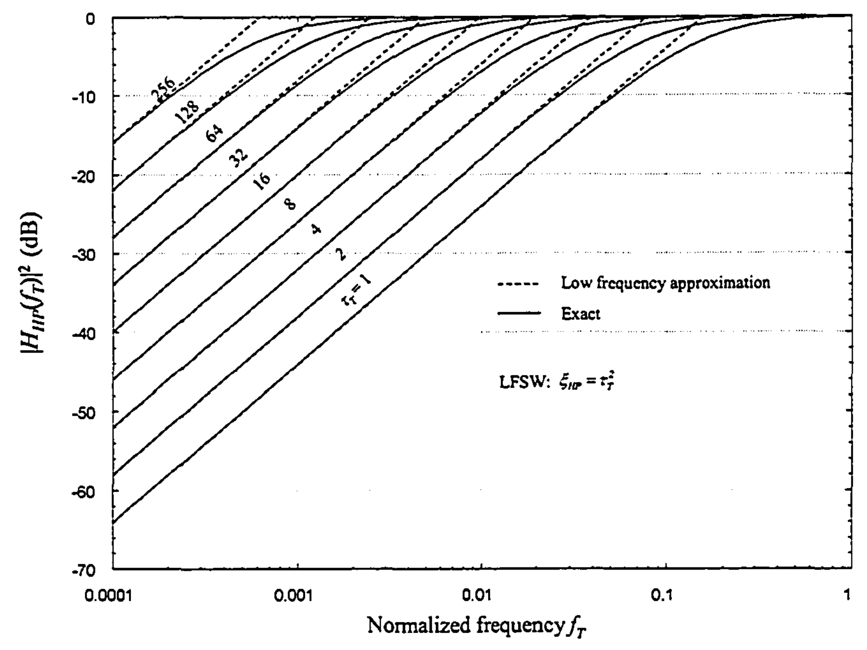


Figure 3.4 $|H_{HP}(f_T)|^2$ and its low frequency approximation versus normalized frequency.

3.2 HP-AWGN channel and Receiver for Corrupted Signals

Assume that the channel corrupts the transmitted signal $x(t_T)$ by high-pass filtering and the addition of white gaussian noise $n(t_T)$ with single-sided power spectral density N_0. In this thesis, this will be denoted as the HP-AWGN channel. Let $y(t_T)$ be the output of the HPF. Thus, the received signal is:

$$r(t_T) = y(t_T) + n(t_T), \tag{3.6}$$

where $y(t_T) = V_{s,l}e^{-(t_T-l)/\tau_T}$ during the lth interval, assuming the use of rectangular waveforms.

Suppose that the received signal $r(t_T)$ is passed through a receiving filter with impulse response $h_r(t_T)$, $0 \le t_T \le 1$. During the lth interval, the output of this filter is sampled at time $t_T = l+1$. The output signal-to-noise ratio (SNR) will be maximized when the receiving filter is a matched filter, matched to the time reverse of the pulse shape which also includes the expected exponential decay [36]. Assume that rectangular waveforms are used to represent the logic values, as outlined in the previous section. Then, for matched filtering when $0 \le t_T \le 1$:

$$\begin{aligned} h_{rM}(t_T) &= B_1 y(1-t_T)\big|_{l=0} \\ &= B_1 V_{s,0} e^{-(1-t_T)/\tau_T} \\ &= Be^{(t_T-1)/\tau_T} \end{aligned} \tag{3.7}$$

where B_1 is an arbitrary constant and $B = B_1 V_{s,0}$. It can be shown that the output SNR for the matched-filter receiver is [36]:

$$\begin{aligned}\text{SNR}_{oM} &= \frac{\int_l^{l+1} (V_{s,l} e^{-(t_T - l)/\tau_T})^2 \, dt_T}{\frac{1}{2} N_0} \\ &= V_{s,l}^2 \tau_T (1 - e^{-2/\tau_T}) / N_0 .\end{aligned} \tag{3.8}$$

If instead the receiving filter ignores the exponential decay introduced by the channel and has a rectangular shape impulse response within a symbol interval, then:

$$h_{rR}(t_T) = B , \tag{3.9}$$

which is equivalent to an integrate-and-dump receiver, and it can be show that the output SNR will be [36]:

$$\begin{aligned}\text{SNR}_{oR} &= \frac{[\int_l^{l+1} B \cdot V_{s,l} e^{-(t_T - l)/\tau_T} \, dt_T]^2}{\frac{1}{2} N_0 \int_l^{l+1} B^2 dt_T} \\ &= 2[V_{s,l} \tau_T (1 - e^{-1/\tau_T})]^2 / N_0 .\end{aligned} \tag{3.10}$$

Comparing the output SNR for these two different receiving filters yields:

$$\text{SNR}_{oR} / \text{SNR}_{oM} = 2\tau_T (1 - e^{-1/\tau_T})^2 / (1 - e^{-2/\tau_T}) , \tag{3.11}$$

which as shown in Figure 3.5, quickly tends to one as τ_T increases. For example, this ratio is 0.99 when $\tau_T = 3$. Therefore, the performance with these two receiving filters is very similar for reasonable values of τ_T. For this reason, and to limit the complexity of simulations, in this thesis a rectangular receiving filter will be used when simulating code performance over the HP-AWGN channel.

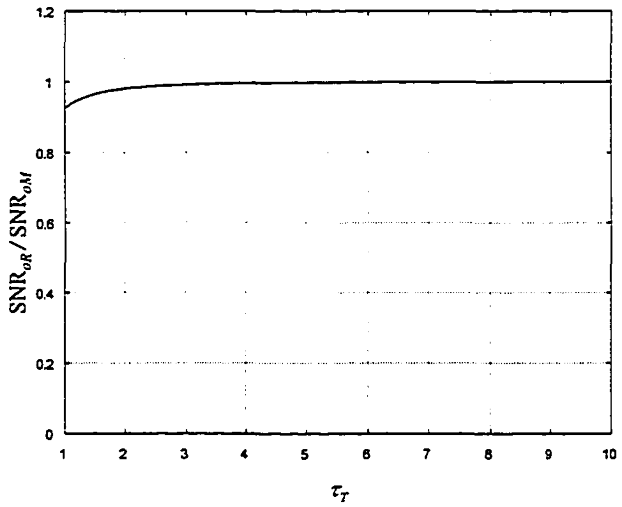


Figure 3.5 Output SNR for two different receiving filters.

Introduction of the HPF to the channel will result in received symbol energy which varies from one symbol interval to the next. Let E_S denote the received normalized symbol energy ($T = 1$) for the AWGN only channel with matched filtering. Let E_{SHR} and E_{SHM} denote the received symbol energy for the HPF-AWGN channel, using a rectangular receiving filter and matched receiving filter, respectively. It is straightforward to show that:

$$\sqrt{E_S} = 1, \tag{3.12}$$

$$\sqrt{E_{SHR}} = |V_{s,l}|\,\tau_T(1 - e^{-1/\tau_T}), \tag{3.13}$$

$$\sqrt{E_{SHM}} = |V_{s,l}|\sqrt{0.5\tau_T(1 - e^{-2/\tau_T})}\,. \tag{3.14}$$

Given the value of N_0, which is the single-sided power spectral density of the AWGN, the characteristics of the HP-AWGN channel are dependent on the statistics of the input sequence and τ_T. For simulation of the error performance of the system, the symbol sequence at the input to the HP-AWGN must be monitored in order to obtain $V_{s,l}$ for calculation of the received symbol energy.

Chapter 4

DC-Free EC Block Codes

Integration of GS codes with well established EC block codes is considered in this chapter. A number of EC block codes will be considered and it will be demonstrated that they can be combined with GS codes in order to offer dc-free and EC performance.

4.1 New Coding Scheme of DC-Free GS-EC Block Codes

As discussed in Subsection 2.2.1, the power spectrum density (PSD) of a sequence equals zero at dc if and only if the running digital sum (RDS) values of the sequence are finite [18]. It has been shown that in binary multimode encoding, existence of at least one complementary codeword pair in each codeword selection set guarantees the ability to generate a dc-free coded sequence [15], and that use of even-weight scrambling polynomials $d_e(x)$ in GS ensures the generation of complementary GS codeword pair(s) [31].

It is also straightforward to show that it is possible to use linear EC block codes to generate complementary EC codewords when input words to the EC encoder are complementary. Note that the sum of complementary words is the all-one word, and that a linear systematic EC code which contains the all-one codeword will map the all-one input word to the all-one codeword (note that this is also possible with some nonsystematic codes). Owing to the linear nature of this EC code, it will also map complementary input words to complementary codewords. To demonstrate this, let two k-bit input words be q_i and q_j, respectively, and let the n-bit codewords be $c_i = F[q_i]$ and $c_j = F\left[q_j\right]$, respectively. Let $w_{m-1}(x)$ denote the polynomial with degree m-1 and all coefficients equal to 1, and let w_{m-1} be the all-one binary word of length m. In a linear systematic EC code with the binary all-one codeword, $w_{n-1} = F[w_{k-1}]$. Let q_i and q_j be complementary input words, i.e., $q_i = \overline{q}_j$. Then:

$$c_i = F[q_i]$$

A version of this chapter has been published in part in *Proceedings of the 2001 IEEE International Symposium on Information Theory*, p. 76, and has been accepted for publication in part in *IEEE Transactions on Communications*, 2005. F. Zhai, Y. Xin, and I. J. Fair.

$$
\begin{aligned}
c_j &= F[q_j] \\
&= F[q_i + w_{k-1}] \\
&= F[q_i] + F[w_{k-1}] \\
&= c_i + w_{n-1} \\
&= \overline{c}_i ,
\end{aligned}
$$

where + denotes elementwise addition. Note that for binary codes it denotes elementwise modulo-2 addition. Therefore, when the inputs to an EC encoder are complementary words, it is sufficient for the all-one word to be an EC codeword in order to ensure the generation of complementary EC codewords. For example, a systematic (7, 4) Hamming code has the all-one codeword. This code has eight pairs of complementary codewords, and the encoder will map a pair of complementary input words to a pair of complementary codewords as illustrated in Table 4.1.

Table 4.1 Input words and codewords of (7, 4) Hamming code listed in the usual order and with paired words

Organized in usual order		Organized with paired words	
Input words	Codewords	Input words	Codewords
0000	0000000	0000	0000000
0001	0001011	1111	1111111
0010	0010110	0001	0001011
0011	0011101	1110	1110100
0100	0100111	0010	0010110
0101	0101100	1101	1101001
0110	0110001	0011	0011101
0111	0111010	1100	1100010
1000	1000101	0100	0100111
1001	1001110	1011	1011000
1010	1010011	0101	0101100
1011	1011000	1010	1010011
1100	1100010	0110	0110001
1101	1101001	1001	1001110
1110	1110100	0111	0111010
1111	1111111	1000	1000101

Consider integration of a GS encoder, which uses A $(A \geq 1)$ augmenting bits and the scrambling polynomial $d_{2,A}(x) = x^A + 1$, with an EC block code which contains the all-one codeword. Figure 4.1 shows the new coding structure for dc-free EC block codes.

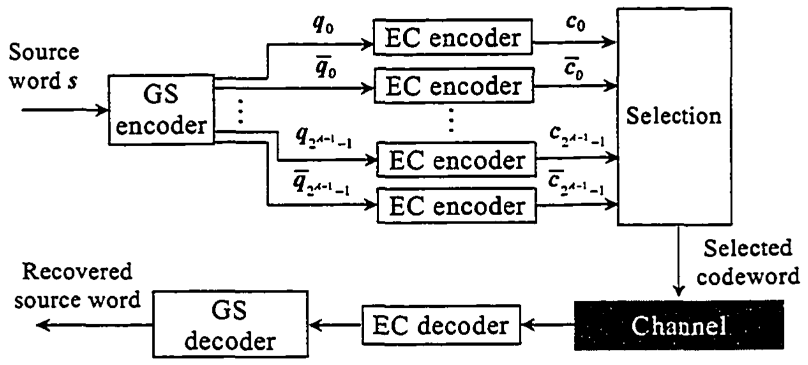


Figure 4.1 Block diagram of the new dc-free error-control block code.

In Figure 4.1, to encode, a source word s is mapped to a set of 2^{A-1} pairs of complementary EC codewords through concatenation of a GS encoder with 2^A identical EC encoders. The GS encoder generates 2^{A-1} pairs of complementary words q_j and $\bar{q}_j$, and the EC encoders encode these complementary words to form the complementary EC words c_j and $\bar{c}_j$, $j=0,1,...,2^{A-1}-1$. As discussed in Subsection 2.2.3, formation of complementary words before selection ensures the ability to generate a dc-free sequence. In this figure, based on a predetermined selection criterion, an EC codeword from the selection set is chosen to ensure that the RDS of the coded sequence is bounded. Therefore, the input to the channel is a fully EC-protected dc-free sequence.

Several selection criteria have been developed for multimode codes. They include minimum word-end running digital sum (MRDS), which selects the word from the selection set which results in the minimum absolute word-end RDS value [15], and minimum squared weight (MSW), which selects the word with minimum sum of the squared RDS values at each bit position within the word [19]. It has been shown that the MSW criterion yields excellent spectral performance for dc-free multimode codes when $A>1$ and provides approximately the same performance as that of MRDS when $A=1$ [23]. Since MRDS is simpler to implement than MSW, in this thesis MRDS will be used when $A=1$ and MSW will be used when $A>1$.

Consider an example which integrates GS codes with EC (7, 4) Hamming codes when $A=1$. Since $A=1$, the MRDS selection criterion and scrambling polynomial $d_{2,1}(x)=x+1$ are used. Assume the RDS prior to encoding the following sequence is 2. Given the 3-bit source words 101 010 011, Table 4.2 depicts the encoding process. The corresponding coded sequence is 0110001 1100010 0010110. Figure 4.2 shows the change of the RDS corresponding to the coded sequence.

Table 4.2 Encoding process for the integration of GS (A=1) and (7, 4) Hamming codes

Source words	Augmented words	GS words	Hamming words	Word disparity	Selected codeword	Word-end RDS
101	0101	0110	0110001	-1	0110001	1
	1101	1001	1001110	1		
010	0010	0011	0011101	1	1100010	0
	1010	1100	1100010	-1		
011	0011	0010	0010110	-1	0010110	-1
	1011	1101	1101001	1		

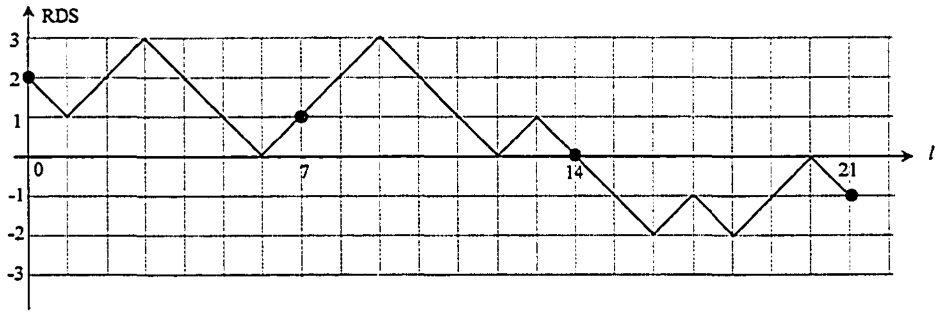


Figure 4.2 RDS of GS-Hamming coded sequence.

Note that after the selection of the second codeword, the word-end RDS is 0. For the selection of the third codeword, the two candidates, 0010110 and 1101001, are "equally good" based on the MRDS criterion. This is the situation called "tie". Many methods can be developed to break the tie. For example, the tie can be broken according to random generation of 0 and 1 with equal probabilities or according to the last bit of the previously selected word. In Table 4.2 and Figure 4.2, the tie is broken during the selection of the third codeword according to the running average of the word-end RDS. When a tie occurs, if the running average of the word-end RDS is positive, the candidate with negative word disparity will be selected, otherwise the candidate with positive word disparity will be selected. Note that with even word length, multiple candidates may have zero word disparity, which is another type of tie and methods similar to the above can be used to break the tie.

It can be shown that the all-one binary codeword exists in most linear EC block codes including binary cyclic codes, binary primitive BCH codes, the Golay code, Reed-Solomon codes, and Reed-Muller codes. For LDPC codes [14], if each row of the parity-check matrix has even weight, the all-one word is a codeword. Furthermore, product codes [13] with the above codes as component codes also have the all-one codeword. Appropriate EC codes will be considered in more detail in the remaining sections of this chapter.

Note that the use of EC codes that map the all-one input word to the all-one codeword is not strictly necessary in order to ensure valid operation of this new encoding approach. However, existence of the all-one codeword is sufficient to ensure that the coded sequence can be dc balanced. In general, in order to ensure that the RDS of the coded sequence can be bounded, it must be ensured that in each selection set there exist words of opposing disparity, but not necessarily equal magnitude of disparity, or at least one word with zero disparity. As shown in [15], if $A=1$, the above condition reduces to (*i*) with odd-length words, the all-one word is required, and (*ii*) with even-length words, a word with at most one zero is required. With $A>1$, a more concrete criterion than the general one expressed above has yet to be determined. Certainly, there is not the requirement that words be related by the all-one word, however, it is convenient for the construction of dc-free codes if there are complementary pairs in the selection set. Note that the all-one codeword is also required in the dc-free coset codes [9] and *M*-ary error-correcting dc-free codes [4].

Since it is not clear how to construct a dc-free EC code with this general requirement, instead of considering this general construction of dc-free EC codes, in this thesis it is proposed that GS and EC code parameters be selected as outlined above to ensure the presence of complementary words in each selection set. It is well known how to generate complementary words at the output of the GS encoder through use of $d_e(x)$, and as shown above, an EC encoder with an all-one codeword will preserve this complementary nature. Alternatively, if the EC encoder maps some input word $\boldsymbol{h}$ rather than the all-one word to the all-one codeword, the GS encoder could be designed to guarantee that words in the GS selection set are related through elementwise modulo-2 addition with $\boldsymbol{h}$. This would involve construction of the appropriate GS scrambling polynomial whose weight would be greater than two [28]. Further details regarding this construction are given in Section 4.2.

An equivalent form of the new coding scheme is shown in Figure 4.3. The difference between these two structures is the manner in which encoding is implemented. Since both the GS encoder and the EC encoders in Figure 4.1 are linear, there are predetermined additive patterns between the codewords $\{c_0,\bar{c}_0,\ldots,c_{2^{A-1}-1},\bar{c}_{2^{A-1}-1}\}$ which depend only on the GS scrambling polynomial and the EC encoder. As discussed in Subsection 2.2.4, in this figure GS$_0$ denotes an encoder that maps a source word $s(x)$ to a GS word $q_0(x)$ by scrambling $s(x)$ with $d(x)$. In order to determine these additive patterns, one can augment the all-zero source word with all patterns of the augmenting bits a_i, $i=0,1,\ldots,2^A-1$, scramble these augmented words with the scrambling polynomial, and encode these scrambled words to EC codewords $\{p_0,\bar{p}_0,\ldots,p_{2^{A-1}-1},\bar{p}_{2^{A-1}-1}\}$. Then the codeword candidates $\{c_0,\bar{c}_0,\ldots,c_{2^{A-1}-1},\bar{c}_{2^{A-1}-1}\}$ in each encoding interval are related by $c_j=c_0+p_j$ and $\bar{c}_j=c_0+\bar{p}_j$, $j=0,1,\ldots,2^{A-1}-1$, where addition is elementwise in GF(2). Figure 4.3 demonstrates that the same codeword set as that generated in Figure 4.1 can be constructed by generating c_0 and then forming the rest of the codeword alternatives through addition with the predetermined additive patterns stored within the encoder. Note that since p_0 is the all-zero sequence, it is not indicated in Figure 4.3, and that since the predetermined additive patterns are complementary, only half of them need to be stored.

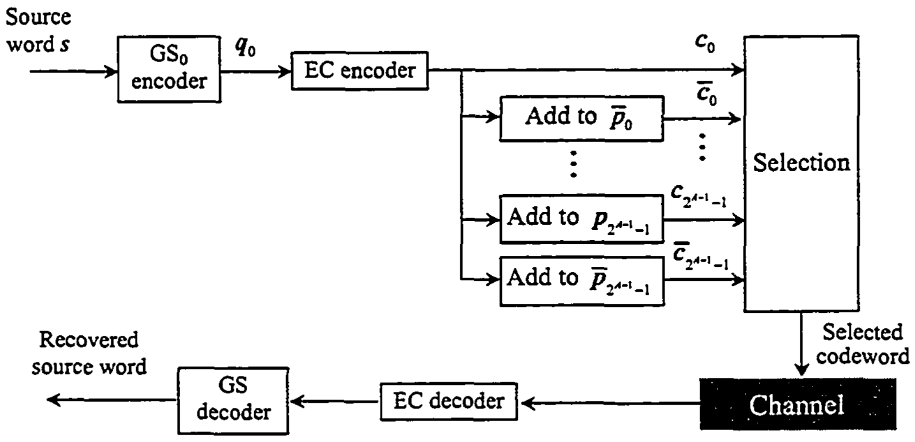


Figure 4.3 Block diagram of the equivalent new dc-free EC block code.

Note that in the proposed scheme, GS and EC coding are not connected in an arbitrary manner. It is required that GS encoders generate complementary pairs, that the GS scrambling polynomial has low weight, that different selection criteria be chosen according to the number of augmenting bits, and that the EC code has the all-one codeword. Therefore, a number of issues are considered in order to ensure that the proposed coding technique will result in good dc-free and EC performance.

As a result, in the proposed technique, the additive patterns $\{p_0, \bar{p}_0, ..., p_{2^{A-1}-1}, \bar{p}_{2^{A-1}-1}\}$ are not an arbitrary subcode of the EC code containing the all-one codeword. They are generated through the careful concatenation of the GS and EC encoders. Given the EC code, the additive patterns are determined by the scrambling polynomial in GS coding, that is, the additive patterns and the scrambling polynomial have a one-to-one mapping. The spectral performance and, to a lesser extent, the BER performance of the proposed codes are affected by the scrambling polynomial employed. In Subsection 2.2.4, the use of good scrambling polynomials was discussed, which implicitly addresses the construction of good additive patterns.

Experiments demonstrate [32] that scrambling polynomials of even weight with a relatively high degree result in good spectral performance in GS coding. Different scrambling polynomials may result in significantly different spectral performance, which indicates that randomly picking additive patterns can result in a significant difference in terms of the spectral performance. In addition, if a subcode of an EC code containing the all-one codeword is randomly picked as the additive patterns, explicit side information would be required to indicate which codeword in the subcode is selected. However, at the receiver, if the side information is erroneous, it will cause error extension through the whole codeword. Additional error protection of side information would cause rate loss which, in turn, would increase the complexity of decoding.

Figures 4.1 and 4.3 show that decoding of the new coding scheme is completed first through EC decoding and then through GS decoding, and that the EC decoding is independent of the GS decoding. Therefore:

- soft-decision information available at the output of the demodulator can be used by the EC decoder, resulting in improved BER performance;
- error extension that occurs during GS decoding follows the EC decoder and therefore has no detrimental impact on the performance of the EC decoder.

The BER performance of this new technique is governed largely by the characteristics of the EC code. Since $d_{2,A}(x) = x^A + 1$ is used for GS, subsequent error extension in GS decoding is upper bounded by two.

As defined in Section 4.1, $w_{n-1}(x)$ denotes the binary all-one word of length n. Let $w'_{n-1}(x)$ denote the non-binary all-one word of length n over GF(2^m). In the following sections the existence of the all-one codeword in a number of different EC block codes will be investigated in order to demonstrate that these EC codes can be integrated into the proposed new dc-free EC block coding scheme.

4.2 DC-Free GS-Cyclic Codes

Binary cyclic codes are well-developed and form a subclass of linear cyclic codes [1]. Note that only binary cyclic codes are considered in this section. Let $q(x)$ and $c(x)$ be the input polynomial and code polynomial related to an input word q and a codeword c of the (n, k) encoder respectively, and let $g_C(x)$ be the generator polynomial of the binary cyclic code. As noted in Subsection 2.3.2, a word c is a codeword if only if $c(x)$ is divisible by $g_C(x)$. Also, the generator polynomial $g_C(x)$ of degree $(n - k)$ of all cyclic (n, k) codes divides x^n+1. Since x^n+1 has even weight, it is divisible by $x+1$. It is straightforward to verify that x^n+1 equals the product of $x+1$ and $w_{n-1}(x)$. If $g_C(x)$ divides $w_{n-1}(x)$, the all-one word of length n is a codeword. When a systematic encoder is used with such a $g_C(x)$, the all-one codeword of length n is uniquely generated by the all-one source word of length k. Therefore, if $g_C(x)$ is selected from the factors of x^n+1 except for $x+1$, the all-one codeword is available in the cyclic code and it can be used for the dc constraint. Therefore, any cyclic code in which the generator polynomial divides $w_{n-1}(x)$ can be included in the new dc-free error control scheme shown in Figure 4.1.

For a nonsystematic code, in general, the all-one codeword of length n is not generated by the all-one input word of length k. For example, the nonsystematic generator matrix for a (7, 4) cyclic code with $g_C(x)=x^3+x+1$ ($g_C=1011$) is:

$$G_n=\begin{bmatrix}1011000\\0101100\\0010110\\0001011\end{bmatrix}.$$

Based on this generator matrix, the input word 1101 (not 1111) is encoded to 1111111. This input word is the GS quotient relationship pattern required to ensure that complementary words are present in the final codeword selection set. Based on this relationship pattern, an appropriate scrambling polynomial can be constructed [28], as shown in the following.

For GS encoding, recall that a k_{GS}-bit source word $s(x)$ generates ($A+k_{GS}$)-bit quotients $q_i(x)$, $i=0,1,\ldots,2^A-1$. Let the scrambling polynomial be $d(x)=d_Dx^D+d_{D-1}x^{D-1}+\ldots+d_1x+d_0$, where $D = A + k_{GS} - 1$. Let $q_i(x)$ and $q_j(x)$ be two quotients and let $q_{ij}(x) = q_i(x) + q_j(x)$ be their quotient relationship pattern, and let $a_i(x)$ and $a_j(x)$ be the two corresponding augmenting words and let $a_{ij}(x) = a_i(x) + a_j(x)$. Relation (2.19) yields:

$$q_i(x) = Q_{d(x)}\left[x^D(a_i(x)x^{k_{GS}} + s(x))\right],$$

$$q_j(x) = Q_{d(x)}\left[x^D(a_j(x)x^{k_{GS}} + s(x))\right],$$

$$q_i(x) + q_j(x) = Q_{d(x)}\left[x^D x^{k_{GS}}(a_i(x) + a_j(x))\right],$$

$$q_{ij}(x) = Q_{d(x)}\left[x^D x^{k_{GS}} a_{ij}(x)\right],$$

$$Q_{x^D}\left[d(x)q_{ij}(x)\right] = x^{k_{GS}} a_{ij}(x). \tag{4.1}$$

The possible values for $\boldsymbol{a}_{ij}$ are taken from the A-bit binary representations of $1, 2, \ldots, 2^A - 1$. For example, when $A = 1$, $\boldsymbol{a}_{ij}$ takes on only the single value 1, and when $A = 2$, $\boldsymbol{a}_{ij}$ takes on the three values 01, 10, and 11. Therefore, there are multiple values of $\boldsymbol{a}_{ij}$ when $A > 1$. For convenience, the value of $\boldsymbol{a}_{ij}$ in relation (4.1) can be chosen to be the binary representation of 2^{A-1}, i.e., $a_{ij}(x) = x^{A-1}$. Therefore,

$$Q_{x^D}\left[d(x)q_{ij}(x)\right] = x^{A+k_{GS}-1}. \tag{4.2}$$

Consider again the nonsystematic cyclic code introduced above. Relation (4.2) can be used to determine the scrambling polynomial that will result in the GS quotient relationship pattern 1101. Consider the case when $A = 1$. Then, $A + k_{GS} = 4$, and the scrambling polynomial will be of the form $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$ with $D = 3$. Values of the coefficients $d_i, i = 0, 1, 2, 3$, must be evaluated such that (4.2) is satisfied. Figure 4.4 illustrates the construction for $d(x)$ and Figure 4.5 shows a simplified structure for the construction.

$d(x)q_{ij}(x)$:

				d_3	d_2	d_1	d_0
			$\times$	1	1	0	1
				d_3	d_2	d_1	d_0
		d_3	d_2	d_1	d_0		
$+$	d_3	d_2	d_1	d_0			
	d_3	d_3+d_2	d_2+d_1	$d_3+d_1+d_0$	d_2+d_0	d_1	d_0

$Q_{x^D}[d(x)q_{ij}(x)]$:	d_3	d_3+d_2	d_2+d_1	$d_3+d_1+d_0$
$x^{A+k_{GS}-1} = x^3$:	1	0	0	0

$d_3 = 1, d_2 = 1, d_1 = 1, d_0 = 0.$

Figure 4.4 Construction of the scrambling polynomial.

$q_{ij}(x)$:	1	1	0	1
$Q_{x^p}[d(x)q_{ij}(x)]$:				d_3
		d_3	d_2	d_1
+	d_3	d_2	d_1	d_0
$x^{A+k_{cs}-1} = x^3$:	1	0	0	0

$$d_3 = 1, d_2 = 1, d_1 = 1,\ d_0 = 0.$$

Figure 4.5 Simplified construction of the scrambling polynomial.

For a given source word it can be verified that use of the scrambling polynomials $d(x) = d_g(x)$ or $d(x) = x^r d_g(x)$, where r is an arbitrary positive integer, will generate the same quotients [28]. Therefore, if one or more least significant bit(s) of the constructed scrambling polynomial are zero, the degree of the scrambling polynomial can be reduced to make the least significant bit be one through right-shifts of the scrambling polynomial. Therefore, the corresponding scrambling polynomial constructed by Figures 4.4 and 4.5 is $d(x) = x^2 + x + 1$. Note that the weight of this $d(x)$ is 3, which is larger than the weight of $d_{2,A}(x)$ which is 2. For cyclic codes, the systematic form of the generator matrix can be obtained from row operations on the non-systematic generator matrix. In this example, the corresponding systematic form of the generator matrix is:

$$\boldsymbol{G}_s = \begin{bmatrix} 1000101 \\ 0100111 \\ 0010110 \\ 0001011 \end{bmatrix}.$$

From this generator matrix, it is clear that the all-one source word 1111 is mapped to the all-one codeword 1111111. With $A = 1$, the scrambling polynomial that generates the required all-one quotient relationship pattern is $d(x) = d_{2,1}(x) = x + 1$ with weight 2. Since a low weight GS scrambling polynomial is preferred to limit the error extension during GS decoding, systematic EC codes are considered in this thesis for construction of the new codes.

Note that some nonsystematic codes can map the all-one source word to the all-one codeword. For example, if two or more columns of the above $\boldsymbol{G}_s$ are permutated where at least one of the columns is from the first four columns, the resulting generator matrix is in non-systematic form and the all-one source word is still mapped into the all-one codeword. In practical systems, systematic codes are preferred in order to easily recover the source data after decoding by simply discarding the parity-check bits.

4.3 DC-Free GS-BCH Codes

Binary BCH codes are a large class of random error-correcting cyclic codes. An (n,k) binary BCH code exists for positive integers m $(m \geq 3)$ and t $(t < 2^{m-1})$ with the following parameters:

Block length:	$n = 2^m - 1$
Number of parity-check digits:	$n - k \leq mt$
Minimum distance:	$d_{\min} \geq 2t + 1$.

Since this code is capable of correcting t or fewer random errors in a block, it is called a t error-correcting BCH code. Note that Hamming codes of length $2^m - 1$ are single error-correcting BCH codes. The generator polynomial $g_{BCH}(x)$ of a binary BCH code is the lowest-degree polynomial over GF(2) that has $\alpha, \alpha^2, \ldots, \alpha^{2t}$ as its roots. It follows from the property of GF(2^m) described in Subsection 2.3.2 that $\alpha, \alpha^2, \ldots, \alpha^{2t}$ and their conjugates are all the roots of $g_{BCH}(x)$. Let $\phi_i(x)$ denote the minimal polynomial of α^i. Then $g_{BCH}(x)$ is the least common multiple (LCM) of the minimal polynomial of α^i $(1 \leq i \leq 2t)$:

$$g_{BCH}(x) = \text{LCM}\{\phi_1(x), \phi_2(x), \ldots, \phi_{2t}(x)\}.$$

This expression can be further simplified by considering that if i is even, α^i must have a conjugate α^j where j is odd and $1 \leq j < 2t$, therefore:

$$g_{BCH}(x) = \text{LCM}\{\phi_1(x), \phi_3(x), \ldots, \phi_{2t-1}(x)\}.$$

The BCH codes as defined above are typically called binary primitive BCH codes.

Since $\phi_0(x) = x + 1$ is not included in $g_{BCH}(x)$, and since $x^n + 1 = (x+1)w_{n-1}(x)$ and $g_{BCH}(x)$ must divide $x^n + 1$, then $w_{n-1}(x)$ is a multiple of $g_{BCH}(x)$. Therefore, all binary systematic primitive BCH codes exhibit the characteristic of all-one input/all-one output encoding, and inherently satisfy the requirement to be used in dc-free GS-EC codes. Thus binary systematic primitive BCH codes can act as the EC code in the proposed new scheme illustrated in Figure 4.1.

4.3.1 Shortened GS-BCH Codes

To meet system design requirements such as codeword length or number of information digits, it may be desirable to shorten a code. Note that BCH codes are cyclic codes. The conventional approach to shortening a cyclic code is as follows. Consider the set of codewords of an (n,k) cyclic code C and select the codewords in which the z leading high-order information digits are zeros to form a subcode of C. Delete the z zero information digits from each of these codewords to obtain a set of 2^{k-z} vectors of length $n-z$, which form an $(n-z,k-z)$ linear code C_{sh}. This code is called a shortened cyclic code, even though it is not cyclic. The same circuits as those used for the original cyclic codes can be employed for the encoding and decoding of the shortened cyclic codes [1]. Table 4.3 gives a (7, 4) systematic BCH code with $g_{BCH}(x) = x^3 + x + 1$ and the shortened (6, 3), (5, 2), and (4, 1) codes based on the (7, 4) code. Note that the (7, 4) systematic BCH code is also a (7, 4) Hamming code.

Table 4.3 (7, 4) systematic BCH code and its shortened codes

Full length BCH code		Shortened BCH codes					
(7, 4)		(6, 3)		(5, 2)		(4, 1)	
Input words	Codewords	Input words	Codewords	Input words	Codewords	Input words	Codewords
0000	0000 000	000	000 000	00	00 000	0	0 000
0001	0001 011	001	001 011	01	01 011	1	1 011
0010	0010 110	010	010 110	10	10 110		
0011	0011 101	011	011 101	11	11 101		
0100	**0100 111**	100	100 111				
0101	0101 100	101	101 100				
0110	0110 001	110	110 001				
0111	0111 010	111	111 010				
1000	1000 101						
1001	1001 110						
1010	1010 011						
1011	1011 000						
1100	1100 010						
1101	1101 001						
1110	1110 100						
1111	**1111 111**						

Unfortunately, the all-one codeword is not present in BCH codes shortened in this manner, which can be observed in Table 4.3. This precludes their straightforward use in the new proposed scheme. Instead, consider the original codewords that have the length $n-k$ all-one parity-check. Note that if $2k > n$, there are $2^k / 2^{n-k} = 2^{2k-n}$ of these words. For example, there are 2 and 128 codewords which have all-one parity-check in the (7, 4) and (15, 11) BCH codes, respectively. From Table 4.3, it is evident that 1111,111 and 0100,111 are the two codewords with all-one parity-check in that (7, 4) code.

It is also straightforward to verify that 00110111111,1111 is one codeword with the all-one parity-check in the binary (15, 11) BCH code generated by $g_{BCH}(x) = x^4 + x^3 + 1$. This codeword indicates how the (15, 11) code can be shortened to a (12, 8) code in a manner which permits inclusion in the GS-EC code structure. Let the zeros in the codeword 00110111111,1111 indicate the positions in which data bits in all words will be ensured to be zero. This codeword is called the shortening pattern in this thesis. Since these bits are known to be zero, they do not need to be communicated to the receiver. To construct the codewords, insert three zeros in the 8-bit message word $\boldsymbol{u} = u_7u_6u_5u_4u_3u_2u_1u_0$ to obtain the 11-bit word $\tilde{\boldsymbol{u}} = 00u_7u_6 0u_5u_4u_3u_2u_1u_0$. This word $\tilde{\boldsymbol{u}}$ is processed as the input word of the (15, 11) encoder, and the three zeros are removed after encoding to create a (12, 8) shortened code. At the receiver, these three zeros are inserted into the demodulated word prior to BCH decoding.

Based on the shortening pattern, it is straightforward to construct the shortened GS-BCH scheme by modifying Figures 4.1 and 4.3 to include appropriate insertion of zeros before the BCH encoder and removal of these zeros after the BCH encoder, and insertion of zeros (or appropriate soft-decision values) before the BCH decoder and removal of these zeros after the BCH decoder. Figure 4.6 shows a block diagram for the dc-free shortened GS-BCH scheme based on the modification of Figure 4.1.

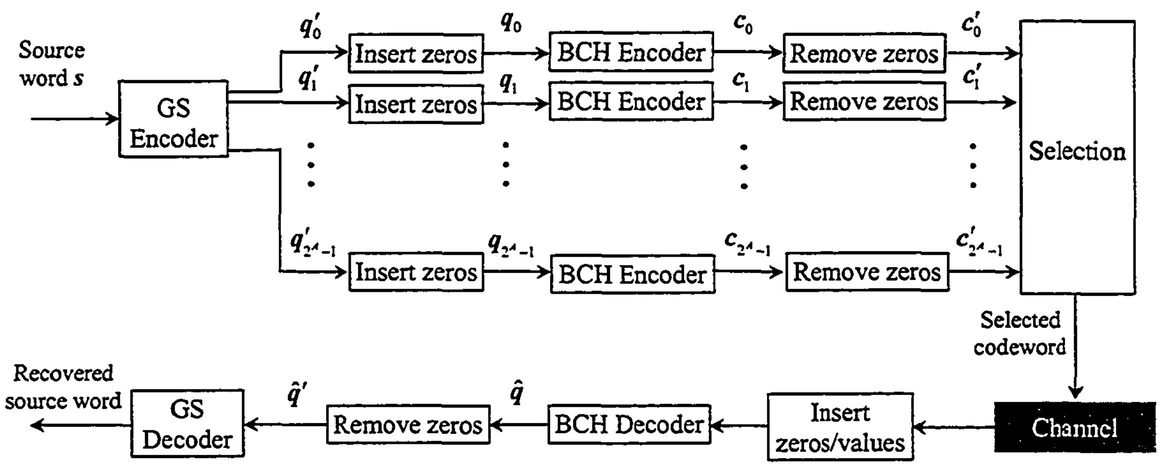


Figure 4.6 Block diagram of the shortened GS-BCH scheme.

Note that if the modification is based on Figure 4.3, the additive patterns need to be modified accordingly.

A number of shortening patterns can be constructed for any given BCH code. For example, for a (15, 11) BCH code with generator polynomial $g_{BCH}(x) = x^4 + x^3 + 1$ ($g_{BCH} = 11001$), Figure 4.7 demonstrates the construction of a shortening pattern for a shortened (12, 8) code by adding shifted versions of g_{BCH}, which is equivalent to evaluating a multiple of g_{BCH} and therefore constructing a valid codeword in the code.

$$\begin{array}{ccccccccccccccc}
 & & & & & & & & & & 1 & 1 & 0 & 0 & 1 \\
 & & & & & & & & & 1 & 1 & 0 & 0 & 1 & \\
 & & & & & & & & 1 & 1 & 0 & 0 & 1 & & \\
 & & & & & & 1 & 1 & 0 & 0 & 1 & & & & \\
 & & & & & 1 & 1 & 0 & 0 & 1 & & & & & \\
 & & 1 & 1 & 0 & 0 & 1 & & & & & & & & \\
\hline
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}$$

Figure 4.7 Construction of a shortening pattern for a (15, 11) BCH code shortened to (12, 8) BCH code.

Ensuring that the first, second, and fifth bits in each length-15 codeword are zero, and not transmitting these three bits shortens the code from a (15, 11) code to a (12, 8) code.

If the original code is to be only slightly shortened, there is a simpler way of constructing the shortening pattern. For example, in the above given code, g_{BCH} can be concatenated with 0_{10}, the all-zero vector of length 10, to yield the word 110010000000000. Since this word is divisible by g_{BCH}, it is a codeword. The complement of this codeword, 001101111111111, is also a codeword because the all-one word of length 15 is also a codeword. In this way the identical shortening pattern 001101111111111 is obtained.

Based on this (15, 11) code, the following shortened codes can be obtained: (12, 8), (11, 7), (10, 6), (9, 5), (8, 4), (7, 3), (6, 2), and (5, 1). Note that for the (15, 11) code there are $2^{2k-n} = 128$ codewords that have the length $n-k=4$ all-one parity-check including the all-one codeword. Therefore, a total of 127 codewords can be used to be shortening patterns. In these shortening patterns, there are 64 codewords whose most significant bit is zero. Table 4.4 lists these 64 shortening patterns grouped for different shortened codes. Note that in this code the minimum number of bits by which the code can be shortened, z, is equal to the weight of $g_{BCH}(x)$.

Table 4.4 Shortening patterns for construction of shortened BCH codes based on the (15, 11) BCH code

(12, 8)	(11, 7)	(10, 6)	(9, 5)	(8, 4)	(7, 3)	(6,2)	(5,1)
001101111111111	*000111111101111*	*000111001111111*	000001011111111	000000111011111	000010100101111	000011000001111	000000001001111
010111110111111	001011101111111	001011011101111	000011110011111	000001101101111	000100000111111	010000000011111	
011110111101111	001110110111111	001111010011111	000110101011111	000010010111111	000101010001111		
011111101011111	010100111111111	010010101111111	001010111001111	000100110101111	001000010101111		
	011001111011111	010011111001111	001100011011111	000101100011111	001001000011111		
	011011010111111	010101011011111	001101001101111	000110011001111	010000110001111		
	011101110101111	011000011111111	001111100001111	001000100111111	011010000001111		
	011110001111111	011010110011111	010001100111111	001001110001111			
	011111011001111	011011100101111	010010011101111	001010001011111			
		011101000111111	010011001011111	001100101001111			
			010101101001111	001110000101111			
			010110010011111	010001010101111			
			010111000101111	010100001101111			
			011000101101111	010110100001111			
			011100100011111	011001001001111			
				011100010001111			

4.3.2 Extended GS-BCH Codes

The most typical extension of a binary code is the inclusion of an overall parity-check which is defined as the modulo-2 sum of all other bits in the word [1]. An (n, k) code C thus becomes an (n+1, k) code C_{ex}, which is called an extension of C. Clearly, the codewords of C_{ex} have even weight. When $d_{\min}$ of the original code is odd, the minimum distance of the extended code is increased by one.

Since the codeword length of the binary BCH code is $n = 2^m - 1\,(m \geq 3)$, which is odd, it is straightforward to verify that the all-one codeword of length n will be extended to the all-one codeword of length $n+1$ in extended BCH codes. Therefore, the extended GS-BCH scheme can be implemented as outlined in Figure 4.1.

4.4 DC-Free GS-RS Codes

It is straightforward to generalize binary codes to nonbinary codes. If p is a prime number and q is power of p, there exist codes, called q-ary codes, with code symbols from the Galois field GF(q). For any positive integers s and t, there exists a q-ary BCH code of length $n = q^s - 1$, which needs no more than 2st parity-check digits and is able to correct t or fewer errors. When $s = 1$, there is the most important subclass of q-ary BCH codes, Reed-Solomon codes [47]. The (n, k) t-error-correcting RS codes with code symbols from GF(q) are defined with the following parameters:

Block length:	$n = q - 1$
Number of parity-check digits:	$n - k = 2t$

Minimum distance: $d_{\min} = 2t + 1$.

RS codes defined over GF(2^m) (i.e., $q = 2^m$) are the most common RS codes. Let α be a primitive element in GF(2^m). The generator polynomial of an $(n, n-2t)$ primitive t-error correcting RS code of length $n = 2^m - 1$ is:

$$\begin{aligned} g_{RS}(x) &= (x+\alpha)(x+\alpha^2)(x+\alpha^3)\cdots(x+\alpha^{2t}) \\ &= x^{2t} + g_{2t-1}x^{2t-1} + \ldots + g_2x^2 + g_1x + g_0. \end{aligned} \tag{4.3}$$

It is clear that $g_{RS}(x)$ has $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ as all its roots and has coefficients $g_{2t-1}, \ldots, g_2, g_1, g_0$ from GF(2^m). RS codes are cyclic codes. Encoding of these codes is similar to the binary case [1], where arithmetic operations embedded in the shift registers are defined over GF(q) rather than GF(2).

In order to implement the dc-free GS-RS scheme, the all-one binary codeword should be present in the RS code. The binary codewords of RS codes are obtained by replacing the code symbols of the RS codewords in GF(2^m) with the corresponding binary vectors. For example, given the $(7,5)$ RS code defined over GF(2^3) shown in Table 4.5, with generator $g_{RS}(x) = (x+\alpha)(x+\alpha^2) = x^2 + (\alpha+\alpha^2)x + \alpha^3 = x^2 + \alpha^4 x + \alpha^3$, one code word is:

$$\begin{aligned} c(x) &= g_{RS}(x)\cdot(\alpha x^4 + \alpha^3x^3 + \alpha^2x^2 + \alpha^4x + \alpha) \\ &= \alpha x^6 + \alpha^2x^5 + \alpha^3x^4 + \alpha^4x^3 + \alpha^5x^2 + \alpha^4x + \alpha^4. \end{aligned}$$

In vector form:

$$c = \alpha\alpha^2\alpha^3\alpha^4\alpha^5\alpha^4\alpha^4.$$

The binary representation of this codeword is 010 100 011 110 111 110 110 (for clarity, a space has been inserted between the binary vectors).

Table 4.5 Three representations for the elements of GF(2^3) generated by $p(x) = x^3 + x + 1$

Power representation	Polynomial representation	Binary vector representation
0	0	000
1	1	001
α	α	010
α^2	α^2	100
α^3	$\alpha + 1$	011
α^4	$\alpha^2 + \alpha$	110
α^5	$\alpha^2 + \alpha + 1$	111
α^6	$\alpha^2 \quad + 1$	101

The 2^m-1 nonzero elements of GF(2^m) form all the roots of $x^n+1=x^{2^m-1}+1$ [1]. It is straightforward to show that $x^{2^m-1}+1$ is equal to the product of $x+1$ and $w'_{n-1}(x)$, where $w'_{n-1}(x)$ denotes the all-one vector of degree n-1 in GF(2^m).

$$\begin{aligned} x^{2^m-1}+1 &= (x+1)\{(x+\alpha)(x+\alpha^2)\cdots(x+\alpha^{2t})\}(x+\alpha^{2t+1})\cdots(x+\alpha^{2^m-2}) \\ &= (x+1)g_{RS}(x)(x+\alpha^{2t+1})\cdots(x+\alpha^{2^m-2}) \\ &= (x+1)w'_{n-1}(x)\,. \end{aligned}$$

Therefore, $w'_{n-1}(x)$ is a multiple of $g_{RS}(x)$, and it is a codeword. However, the binary representation of $w'_{n-1}(x)$ is not all-one when $m>1$. For example, for the above (7, 5) RS code in GF(2^3), $w'_6(x)=x^6+x^5+x^4+x^3+x^2+x+1$. The binary representation of $w'_6(x)$ is 001 001 001 001 001 001 001. But in a linear code over GF(q), the product of any codeword by a field element is a codeword [74]. Let α^γ ($m<\gamma\le 2^m-2$) from GF(2^m) be the element whose binary vector representation is all-one vector of length m. Then, $\alpha^\gamma w'_{n-1}(x)$, whose binary representation is an all-one word, exists in RS codes over GF(2^m). Therefore RS codes can be included in the proposed scheme outlined in Figure 4.1.

4.4.1 Shortened GS-RS Codes

As with the shortened BCH codes, the all-one codeword is not present in RS codes shortened in the usual manner. A method similar to that discussed in the case of BCH codes can be used to find an appropriate shortening pattern for RS codes. However, that method, which is based on the summation of shifted versions of the generator polynomial, is not as straightforward in the RS case because of the nonbinary elements in GF(2^m). For example, the (7, 5) RS code discussed above has $g_{RS}(x)=(x+\alpha)(x+\alpha^2)=x^2+\alpha^4x+\alpha^3$; refer to Table 4.5 for the different representation of the elements, and note that $\alpha^7=1$. Two shortening patterns $000\alpha^5 0\alpha^5\alpha^5$ for a (3, 1) shortened code and $0\alpha^5 00\alpha^5\alpha^5\alpha^5$ for a (4, 2) shortened code can be found with the construction shown in Figure 4.8.

Note that the shortening pattern for the (3, 1) code, when written with coefficient 1 instead of α^5 and omitting the first three zeros, is x^3+x+1, which is the generator polynomial of a binary primitive (7, 4) BCH code. This observation provides an easier approach to construct the shortening patterns for the GS-RS scheme based on the generator polynomials of binary primitive BCH codes which are given in many EC text books.

		x^6	x^5	x^4	x^3	x^2	x	1	
$g_{RS}(x)$						1	α^4	α^3	
$\alpha^3 x g_{RS}(x)$	+				α^3	α^7	α^6		
$(1+\alpha^3 x)g_{RS}(x) = c_0(x)$					α^3	0	α^3	α^3	
$\alpha^2 c_0(x) = c_{s1}(x)$		0	0	0	α^5	0	α^5	α^5	✓
$\alpha^2 c_{s1}(x) = c_2(x)$					1	0	1	1	
$x^2 c_2(x)$	+		1	0	1	1			
$(1+x^2)c_2(x) = c_3(x)$			1	0	0	1	1	1	
$\alpha^5 c_3(x) = c_{s2}(x)$		0	α^5	0	0	α^5	α^5	α^5	✓

Figure 4.8 Construction of the shortening patterns $c_{s1}(x)$ and $c_{s2}(x)$ respectively, for a (3, 1) shortened RS code and a (4, 2) shortened RS code, shortened from the (7, 5) RS code.

Consider a primitive binary (n, k_B) BCH code and an (n, k_R) RS code, such that both codes are constructed over GF(2^m) and have error-correcting capability t. Note that although these two codes have the same t, the binary BCH code can correct t binary errors and the RS code can correct t code symbol errors, where a code symbol consists of m bits. Since $g_{BCH}(x)$ has $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ and their conjugates as all its roots and $g_{RS}(x)$ has $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ as all its roots [1], $g_{BCH}(x)$ is multiple of $g_{RS}(x)$ and is a codeword of the (n, k_R) RS code. Therefore the shortening patterns for the GS-RS scheme can be obtained through summation of shifted versions of the corresponding $g_{BCH}(x)$, and subsequent replacement of the coefficients of value 1 by α^γ.

For example, consider the (15, 11) RS code defined over GF(2^4) with t = 2; refer to Table 2.4 for the different representations of the elements. In order to construct a shortening pattern, first locate the corresponding (15, 7) BCH code with t = 2 and $g_{BCH}(x) = x^8 + x^7 + x^6 + x^4 + 1$ ($g_{BCH} = 111010001$). In this example $g_{RS}(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4) = x^4 + \alpha^{13}x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10}$, and it is straightforward to verify that $g_{RS}(x)$ divides $g_{BCH}(x)$. Based on $g_{BCH}(x)$, the shortening patterns for (10, 6), (9, 5), and (8, 4) shortened RS codes are as follows:

$000\alpha^{12}0\alpha^{12}\alpha^{12}\alpha^{12}0\alpha^{12}\alpha^{12}\alpha^{12}\alpha^{12}\alpha^{12}\alpha^{12}$ for (10, 6),

$00\alpha^{12}0\alpha^{12}\alpha^{12}0\alpha^{12}0\alpha^{12}0\alpha^{12}\alpha^{12}\alpha^{12}\alpha^{12}$ for (9, 5),

$0\alpha^{12}0\alpha^{12}\alpha^{12}00\alpha^{12}000\alpha^{12}\alpha^{12}\alpha^{12}\alpha^{12}$ for (8, 4).

Therefore, these shortened RS codes can be used in the new shortened GS-EC scheme depicted in Figure 4.6, with the BCH encoders and decoder replaced by RS coding procedures.

4.5 DC-Free GS-Reed-Muller Codes

Reed-Muller codes constitute a class of linear codes over GF(2). They are easy to describe and can be decoded using majority-logic circuits. For any integers m and r, $r = 0,1,\ldots,m$, there is a binary r-th order RM code $R(r,m)$ with the following parameters [74]:

Block length:	$n = 2^m$
Number of source digits:	$k = \sum_{i=0}^{r} \binom{m}{i}$
Minimum distance:	$d_{\min} = 2^{m-r}$

The generator matrix of the r^{th} order RM code of block length of 2^m is defined as the array:

$$G = \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ \vdots \\ G_{r\text{-}1} \\ G_r \end{bmatrix}$$

where G_0 is the all-one vector of length n; G_1 is an $m \times n$ matrix in which each binary m-bit vector appears once as a column; G_2 is an $\binom{m}{2} \times n$ matrix constructed by taking all possible products of two rows of G_1 to be rows of G_2; and in general G_r is an $\binom{m}{r} \times n$ matrix constructed by taking all possible products of r rows of G_{r-1} to be rows of G_r when $r \geq 2$. For example, the (2, 4) RM code (with $r = 2$ and $m = 4$) is a (16, 11) block code with $d_{\min} = 4$. Its generator matrix G as described above is nonsystematic, but it can be converted to a systematic form G_s as shown below:

$$G = \begin{bmatrix} G_0 \\ \hline G_1 \\ \hline G_2 \end{bmatrix} = \begin{bmatrix} 1111111111111111 \\ \hline 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \\ \hline 0000000000001111 \\ 0000000000110011 \\ 0000000001010101 \\ 0000001100000011 \\ 0000010100000101 \\ 0001000100010001 \end{bmatrix}, \qquad G_s = \begin{bmatrix} 1000000000011111 \\ 0100000000011100 \\ 0010000000011010 \\ 0001000000011001 \\ 0000100000001110 \\ 0000010000001101 \\ 0000001000001011 \\ 0000000100010110 \\ 0000000010010101 \\ 0000000001010011 \\ 0000000000100111 \end{bmatrix}.$$

In order to include RM codes in the new dc-free EC codes, an all-one source word needs to be mapped to an all-one codeword. Based on the nature of the RM generator matrix constructed above, the codeword set contains the all-one codeword. However, in a binary linear block code the all-one source word cannot be mapped to the all-one codeword if even one column weight of the generator matrix is even; only if all column weights of the generator matrix are odd is the all-one source word mapped to the all-one codeword. It will now be shown that this property holds for systematic RM generator matrices.

For an (n, k) binary linear systematic block code, a codeword can be separated into the source-bit part of length k and the parity-check-bit part of length n-k. Each of the 2^k codewords uniquely uses one of the 2^k binary patterns as its source-bit part. If the all-one word of length n is a codeword, this codeword must be mapped to the all-one source word.

Equivalent linear block codes have generator matrices that are related by elementary row operations and/or column permutations. Therefore, equivalent codes have the same distance properties. Any linear block code can be transformed into an equivalent systematic code. If the generator matrix is transformed through elementary row operations, the codeword set remains the same. If this transformation is through column permutation, the codeword set is changed. However, if there exists an all-one codeword, this codeword remains in the codeword set after any transformation of the generator matrix. Thus for systematic RM codes there exists a one-to-one mapping between the all-one source word and the all-one codeword. This can be verified by the $\boldsymbol{G}_S$ of above example since the column weights of this generator matrix are odd. Therefore, systematic RM codes can be included in the proposed scheme shown in Figure 4.1.

4.6 DC-Free GS-LDPC Codes

LDPC codes are linear capacity-approaching block codes. In an LDPC code, the parity-check matrix of $\boldsymbol{H}$ is sparse, i.e., it consists of a number of low weight columns or rows. The codes are usually subject to some regularity constraints, like fixed row weight w_r and fixed column weight w_c in $\boldsymbol{H}$. The codes are usually denoted as (n, w_c, w_r) [75], where n denotes block length. If the proportion of bits with value one is ρ for the rows and columns, since $\boldsymbol{H}$ is an $(n-k)$ by n matrix, then $w_c = \rho(n-k)$ and $w_r = \rho n$, and the code rate is $R_{EC} = 1-(w_c / w_r)$ when $w_c < w_r$. It can be seen that the number of parity-check bits can be derived given the code parameters (n, w_c, w_r). Conditioned on these constraints, $\boldsymbol{H}$ is randomly chosen. The randomness ensures a good code, while the sparseness enables efficient decoding.

For implementation of the new dc-free LDPC codes, it is required that the all-one word be a codeword, and it is convenient if the all-one source word generate the all-one codeword. This in turn implies that each row of the parity-check matrix $\boldsymbol{H}$ must have even weight, and that all columns of the generator matrix in systematic form must have odd weight. For example, a (12,3,6) regular LDPC code constructed in [75] satisfies the above requirement as shown below:

$$\boldsymbol{H} = \begin{bmatrix} 110100001101 \\ 111110010000 \\ 101011000110 \\ 000111101010 \\ 011000110011 \\ 000001111101 \end{bmatrix}.$$

In order to obtain the systematic form of $\boldsymbol{H}$, elementary row operations and permutations of the columns are needed. If all rows of $\boldsymbol{H}$ have even weight, then after an elementary row operation, the resulting row still has even weight because the modulo-two sum of two even weight vectors yields an even weight vector. Column permutations do not affect the weight of rows, therefore the row weight of a systematic parity-check matrix $\boldsymbol{H}_s = \left[\boldsymbol{P}^T \mid \boldsymbol{I}_{n-k}\right]$, where $\boldsymbol{P}$ is an $k \times (n-k)$ matrix and $\boldsymbol{I}_{n-k}$ is an $(n-k) \times (n-k)$ identity matrix, is still even. It is clear that the row weight of $\boldsymbol{P}^T$ is odd, or equivalently, that the column weight of $\boldsymbol{P}$ is odd. Thus the column weight of the systematic generator matrix $\boldsymbol{G}_s = [\boldsymbol{I}_k \mid \boldsymbol{P}]$ is odd. The $\boldsymbol{H}_s$ and $\boldsymbol{G}_s$ of the above example are given below:

$$\boldsymbol{H}_s = \begin{bmatrix} 001011100000 \\ 111110010000 \\ 110010001000 \\ 010101000100 \\ 111110000010 \\ 010011000001 \end{bmatrix}, \qquad \boldsymbol{G}_s = \begin{bmatrix} 100000001011 \\ 010000111110 \\ 001000110010 \\ 000100010101 \\ 000010111110 \\ 000001010011 \end{bmatrix}.$$

Note that for the above example, when $\boldsymbol{H}_s$ is constructed from $\boldsymbol{H}$, column permutation is not required. Therefore, $\boldsymbol{G}_s$ can be used to generate a systematic code and the sparse matrix $\boldsymbol{H}$ can be used for the decoding of the codes. If column permutation is required when $\boldsymbol{H}_s$ is constructed from ***H***, ***H*** should be first modified to form H' by the column permutation to avoid column permutation once H_s is generated. Note that H' and ***H*** are equivalent in the terms of row weight and column weight.

Therefore, the all-one word is a codeword in an LDPC code when the parity-matrix has even row weight, and dc-free GS-LDPC codes are straightforward to implement according to the scheme outlined in Figure 4.1.

4.7 DC-Free GS-Product Codes

As described in Subsection 2.3.2 and shown in Figure 2.4, an (n_1n_2, k_1k_2) product code is constructed through use of an (n_1, k_1) inner block code and an (n_2, k_2) outer block code. If linear codes are used as component codes, the "checks on checks" part will be the same regardless whether the row code or column code is encoded first [1]. Product codes provide an easy way to generate complicated codes from simple codes. For example, a simple parity-check code has minimum distance 2. This code can detect a single error but it cannot correct errors. If both the row and column codes are parity-check codes, the resulting product code has minimum distance 4, and the code can correct all single errors and can also detect all double errors.

It will now be shown that new dc-free GS-product codes, which use product coding as the EC coding, can be constructed according to Figure 4.1. Iterative decoding algorithms introduced in [13] can be fully employed in the product decoding of this new coding scheme.

In the new codes, two short systematic linear block codes, denoted as (n_1, k_1) and (n_2, k_2) codes (note that the two block codes could be the same) are used as component codes to construct the product code. If all-one binary codewords exist for both component codes, it is straightforward to show that through such a product encoder, for column encoding, k_1 ones are encoded to a length-n_1 all-one column word, and for row encoding, k_2 ones are encoded to a length-n_2 all-one row word. Therefore, an all-one input word of length k_1k_2 results in the all-one codeword of length n_1n_2. This code, combined with appropriate GS encoding, will guarantee the existence of product codeword pair(s) in each selection set. Appropriate selection of the codewords will guarantee generation of dc-free sequence. Note that all codes discussed above can be used as component codes in these product codes.

4.8 Performance of DC-Free GS-EC Block Codes

In this section, the performance of the proposed dc-free GS-EC block codes will be evaluated and will be compared with dc-free coset coding and the conventional approach of concatenating EC and dc-free codes. Note that for the simulation results reported in this section, independent and equiprobable information bits are assumed unless otherwise specified.

In the following, advantages of the new coding scheme are presented. Performance results are reported when an RS code is used as the component EC code to show that the new coding scheme yields coding gain when compared to the conventional concatenation of RS and GS codes. Results are also presented when an LDPC code is used as the component EC code to illustrate integration of soft-decision EC decoding, which is not possible, in general, in the conventional scheme. Note that in the following results $d_{2,A}(x)$ and the MSW selection criterion are used in the proposed codes.

The performance of the proposed new GS and RS integrated coding scheme (GS-RS scheme) is shown in Figures 4.9 through 4.11. The simulation parameters include: (1912, 1904) GS code with $A=8$; (255, 239) RS code with $p=435$ (in octal) for generating GF(2^8). For comparison, the BER performance for other systems is also given, including uncoded signalling, (255, 239) RS coding, and conventional concatenation of RS and GS codes. In this conventional code, the (255, 239) RS code is used as the outer code and a (2048, 2040) GS code is used as the inner code.

Figure 4.9 depicts the PSD performance of the GS-RS scheme. The corresponding conventional scheme provides almost the same low frequency suppression as that of the new scheme; RS coding alone does not result in suppression of low frequencies.

Figure 4.10 presents the BER performance of this new GS-RS scheme over the additive white Gaussian noise (AWGN) channel, and compares this to the performance of uncoded signalling, the conventional scheme, and RS coding without spectrum control. Eb denotes the average energy per information bit, N0 is the single-sided power spectral density of the white Gaussian noise, and matched filtering is used. RS decoding is implemented using the hard-decision Berlekamp-Massey algorithm [1]. As indicated in this figure, under these conditions, the new GS-RS scheme offers 1 dB gain at BER=10-7 when compared to conventional concatenation of RS and GS codes.

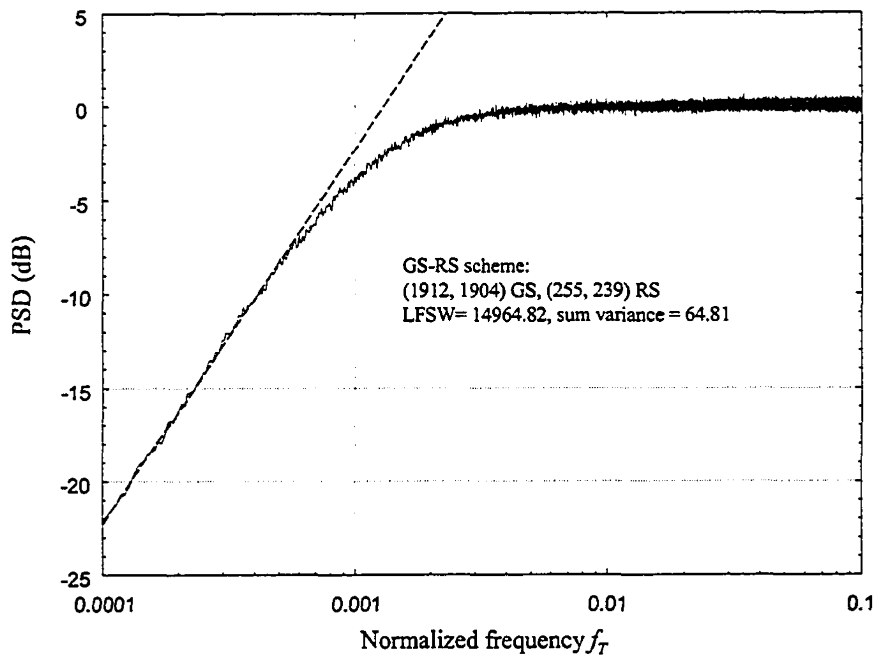


Figure 4.9 Simulated PSD (solid curve) and approximate PSD (dashed line, based on the simulated LFSW) for a dc-free GS-RS coding scheme.

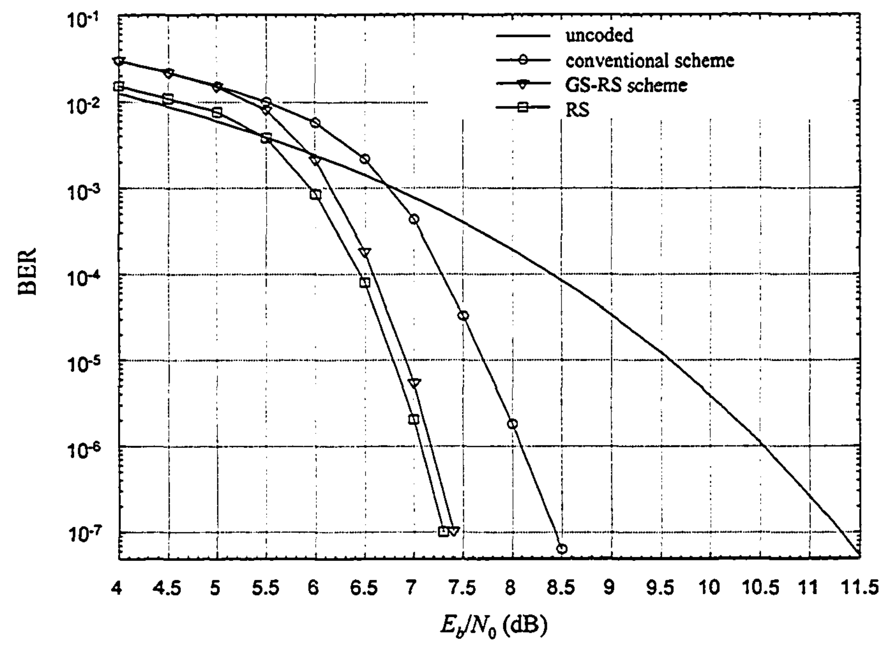


Figure 4.10 BER performance of several schemes over the AWGN channel.

In Chapter 3, the noisy dc-constrained channel is modeled using a simple first order HPF and AWGN (HP-AWGN channel). The HPF can be used to model a specific dc constraint through selection of the value of the normalized time constant τ_T as illustrated in Figure 3.4.

Figures 4.11 (a) and (b) illustrate BER performance over the HP-AWGN channel of the new GS-RS coding scheme and the other three schemes for different source logic probabilities when $\tau_T = 120$ and a rectangular pulse shape and a rectangular receiving filter are used. The results are presented when the probability of a logic 0 in the source sequence, p_0, is 0.5, 0.45, and 0.4. It can be seen that the change of this probability does not affect the performance of the GS-RS scheme and the conventional scheme significantly, but has a large, negative effect on the performance of the uncoded and RS codes, which justifies the use of dc-constrained codes on this channel. While these results are for the simple first-order HP-AWGN channel model, it is expected that these trends would also be apparent in more realistic noisy dc-free constrained systems.

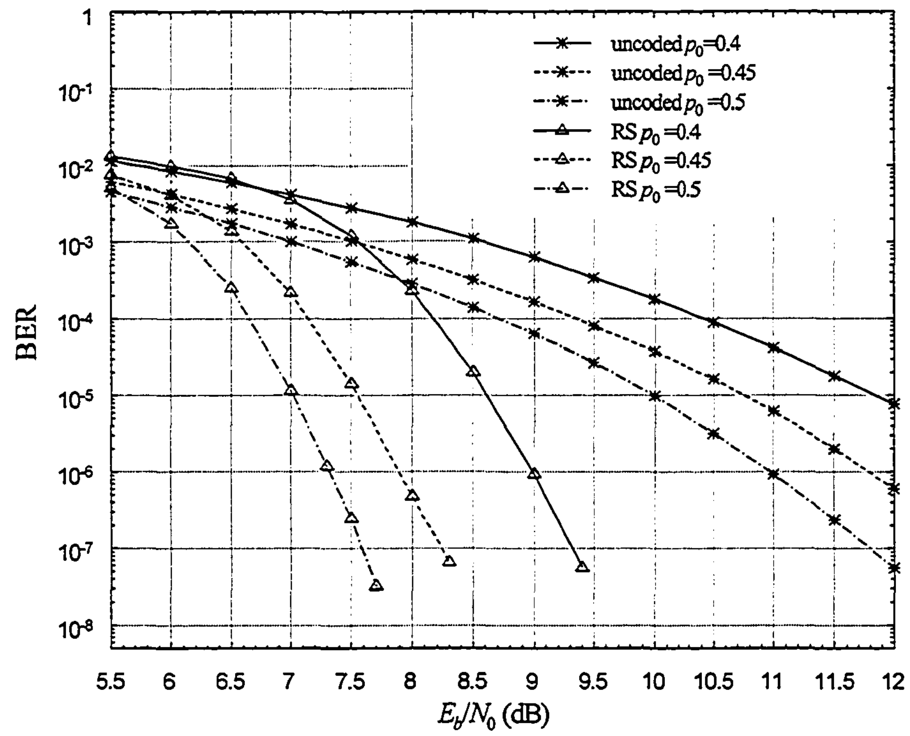


Figure 4.11 (a) BER performance of uncoded signalling and RS coding schemes with different source symbol probabilities over the HP-AWGN channel ($\tau_T = 120$).

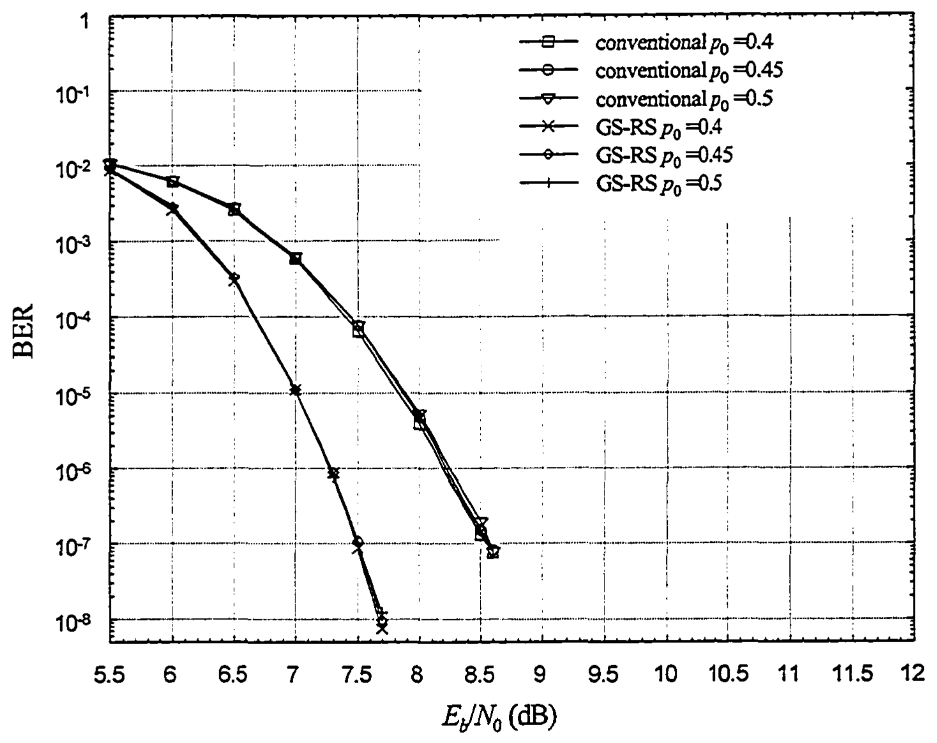


Figure 4.11 (b) BER performance of the conventional scheme and the GS-RS coding scheme with different source symbol probabilities over the HP-AWGN channel ($\tau_T = 120$).

The performance in terms of PSD and BER of the proposed new GS and LDPC integrated coding scheme (GS-LDPC) is shown in Figures 4.12 and 4.13. The rate 1/2 (504, 252) LDPC code uses a parity-check matrix $\boldsymbol{H}$ from [76] with column weight 3 and row weight 6, and LDPC decoding is performed using the iterative sum-product algorithm [14], [63], [67] with a maximum of 1000 iterations.

Figure 4.12 illustrates the PSDs for this GS-LDPC scheme with $A=2$ and $A=4$. Clearly, a 10 dB improvement in performance is obtained at $f_T = 10^{-4}$ when $A=4$ compared to the case with $A=2$. This is in agreement with the results obtained with increasing A in GS dc-free multimode codes without provision for error control [23]. Figure 4.13 shows the BER performance in the AWGN channel for the GS-LDPC scheme with $A=2$ and $A=4$, and the BER performance for LDPC codes without the dc constraint. It is evident that when the BER is less than 10^{-5} the BER performance of the new GS-LDPC scheme is within 0.15 to 0.2 dB of that of the LDPC code without the dc constraint. This loss in performance in the AWGN channel is due to the error extension in GS decoding and the rate penalty associated with including GS augmenting bits. As with the GS-RS scheme, however, the performance of the GS-LDPC scheme will be superior on channels with a dc constraint.

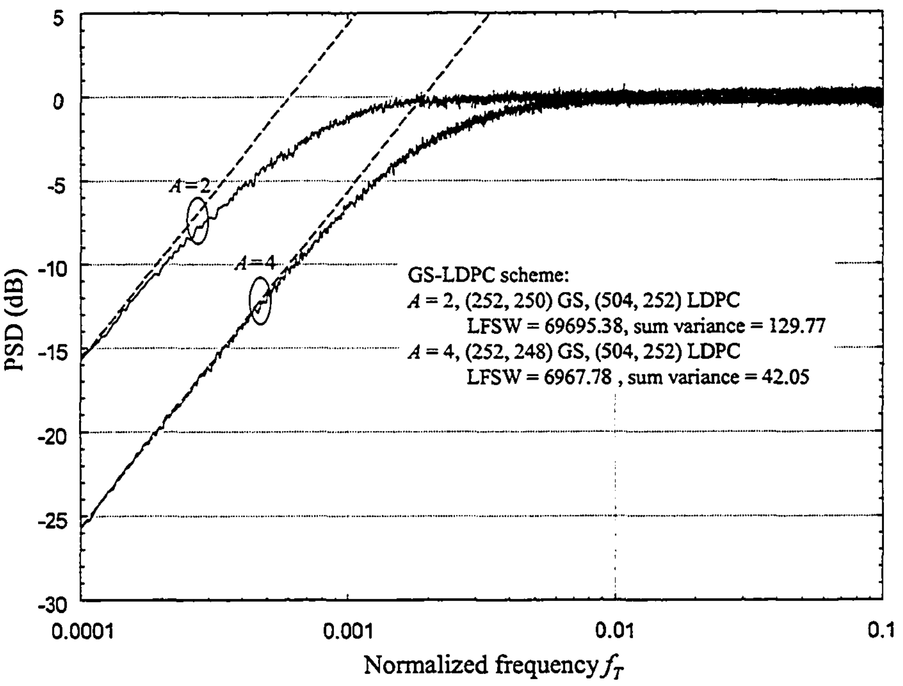


Figure 4.12 Simulated PSDs (solid curves) and approximate PSDs (dashed lines, based on the simulated LFSW) of two GS-LDPC codes.

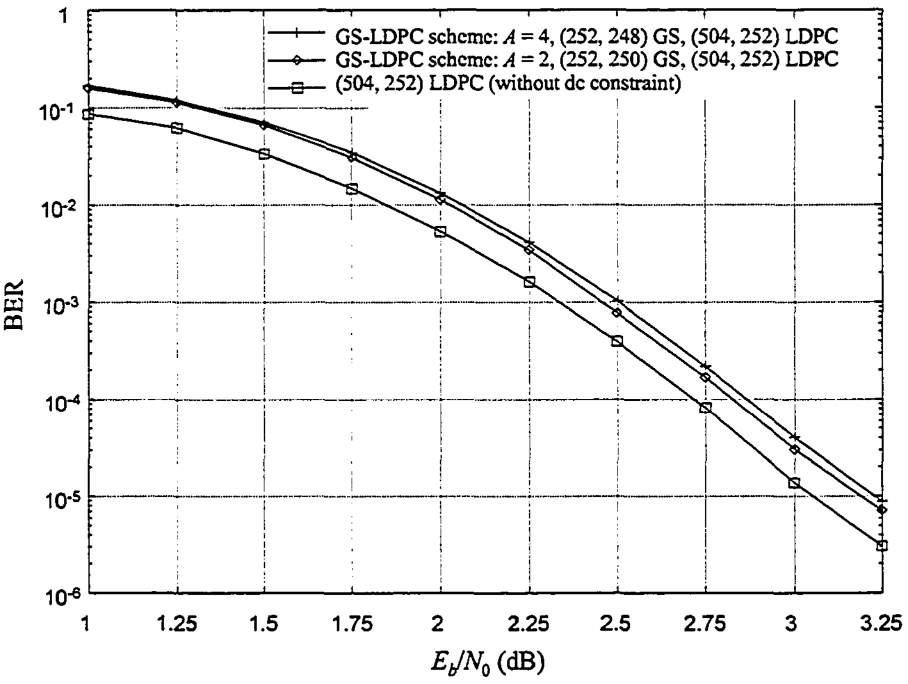


Figure 4.13 BER performance of the GS-LDPC coding scheme and LDPC coding over the AWGN channel.

The remainder of this section analyzes the performance of the new coding scheme and provides performance comparison between the new coding scheme and the coset coding scheme.

Due to the complexity of GS dc-free multimode codes, it is not yet known how to fully analyze their dc-free performance in a practical manner [23]. Instead, the spectral performance of these codes is obtained through simulation or approximations. In both cases, metrics such as sum variance and LFSW are used to assess the performance of these codes. A random drawing model was proposed to estimate the sum variance of multimode codes which used the MRDS selection criterion [23], and a method has been developed to estimate the sum variance and the LFSW of GS dc-free multimode codes which use $d_{2,A}(x)$ and the MSW criterion [32].

As discussed in Subsection 2.2.5, the sum variance and LFSW of GS dc-free multimode codes which use the MSW criterion and $d_{2,A}(x)$ can be approximated by relations (2.27) and (2.28), respectively. Since the spectral performance of the proposed scheme is almost independent of the EC code used, and is dominated by the parameters of the GS code, relations (2.27) and (2.28) can also be applied to the proposed scheme to accurately estimate sum variance and LFSW values. Therefore, based on relations (2.27) and (2.28), the sum variance σ_s^2 and the LFSW ξ of the dc-free GS-EC codes can be approximated as:

$$\sigma_s^2 \approx 0.2326 \cdot S_A \cdot n_b / A, \tag{4.4}$$

$$\xi \approx 0.2225 \cdot L_A \cdot (n_b / A)^2, \tag{4.5}$$

where n_b is the binary codeword length of the EC code, and S_A and L_A are factors which are dependent only on the number of augmenting bits A. Values of S_A and L_A are listed in Table 2.3.

Figures 4.14 and 4.15 compare the sum variance and LFSW calculated from (2.27) and (2.28) with simulated sum variance and LFSW for GS dc-free multimode codes. These figures also show simulated sum variance and LFSW for the new dc-free GS-EC block codes where the (255, 239) BCH code, (504, 252) LDPC code, and (255, 239) RS codes are used as EC codes.

These figures show that relations (4.4) and (4.5) can also be used to accurately access sum variance and LFSW values of the proposed dc-free GS-EC block codes.

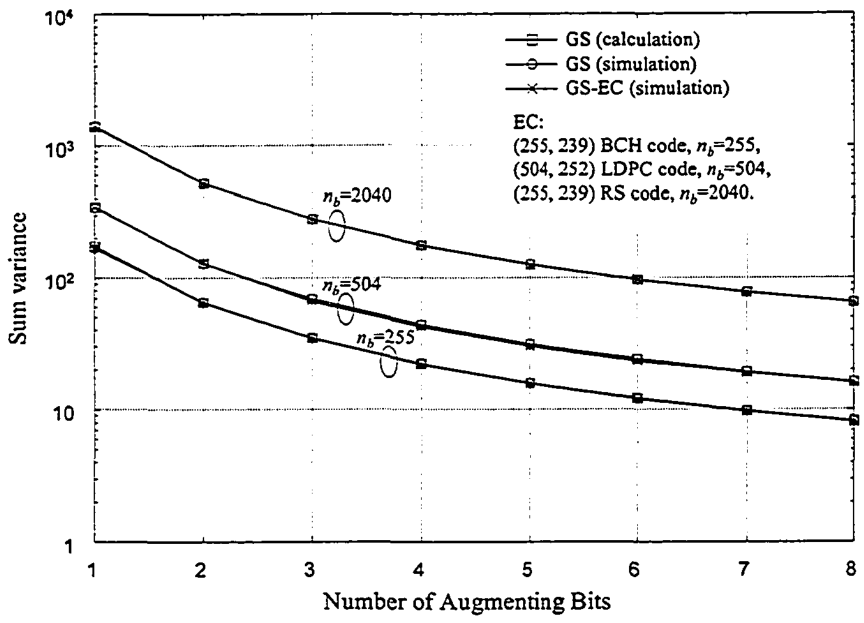


Figure 4.14 Sum variance of GS dc-free multimode codes (calculation and simulation), and simulated sum variance of three different dc-free GS-EC codes.

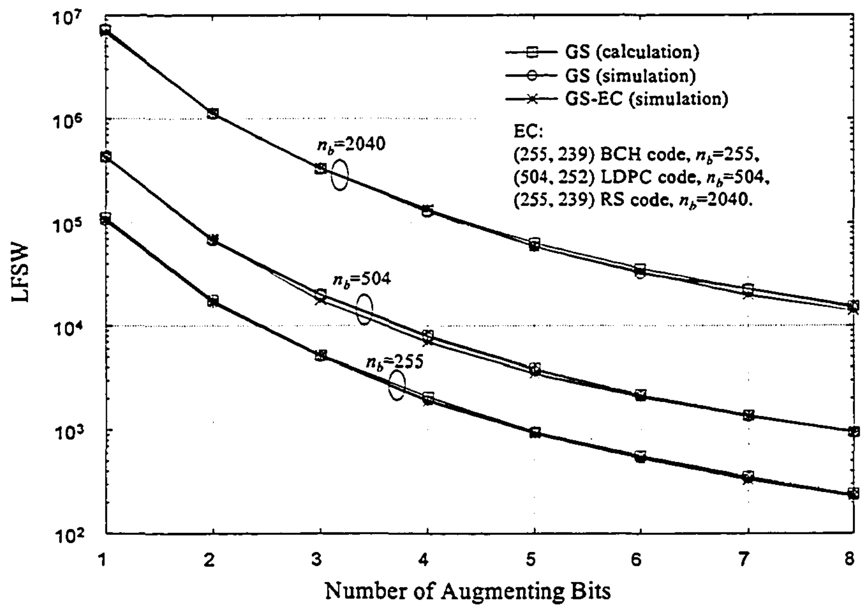


Figure 4.15 LFSW of GS dc-free multimode codes (calculation and simulation), and simulated LFSW of three different dc-free GS-EC codes.

In [9], a dc-free coset code is constructed from a binary BCH code through modification of the non-systematic generator matrix of the BCH code. The constructed code is called a dc-free coset BCH code. For a binary t-error-correcting BCH code of length $n = 2^m - 1$, let A_c and n_p be two odd integers such that $A_c \cdot n_p = n$ and $n_p \geq 2t+1$. As described in [9], the dc-free coset BCH code of length n can be generated with A_c augmenting bits. Due to the similarity of the approach to dc control in the dc-free coset BCH code [9] of length n and a polarity switch code of length n_p, it is proposed to estimate the sum variance and LFSW of the dc-free coset BCH code by use of relations (2.16) and (2.17). Therefore, the sum variance and LFSW of the dc-free coset BCH code can be approximated as :

$$\sigma_s^2 \approx (2 \cdot n_p - 1)/3, \tag{4.6}$$

$$\xi \approx 1.03 n_p^2 - 1.35 n_p. \tag{4.7}$$

Note that since n_p is relatively large in this dc-free coset BCH code, the coefficient 0.08 in (2.17) is omitted in (4.7).

In order to compare the performance of the proposed GS-BCH code with the dc-free coset BCH code, consider the (255, 239) BCH code with $g_{BCH} = 267543$ (in octal) [1] and $A = A_c = 5$ augmenting bits in both the GS-BCH code and the dc-free coset BCH code; in this coset BCH code, $n_p = 51$. Note that the BCH code used in the proposed GS-BCH code is a standard systematic BCH code and the code used in the dc-free coset BCH code is a nonsystematic modified BCH code [9]. From (4.4) through (4.7), noting that $S_A = 1.3367$ and $L_A = 1.7079$ when $A = 5$ as listed in Table 2.3, it follows that $\sigma_s^2 \approx 15.86$ and $\xi \approx 988.40$ for the GS-BCH code, and $\sigma_s^2 \approx 33.67$ and $\xi \approx 2610.10$ for the dc-free coset BCH code. The simulated σ_s^2 and ξ values equal 15.58 and 912.15 for the GS-BCH code, and 33.67 and 2905.22 for dc-free coset BCH code, respectively.

Figure 4.16 shows simulated PSDs for both these codes. These results demonstrate that with the same number of augmenting bits, the GS-BCH code results in significant improvement in spectral performance compared to the dc-free coset BCH code. Figure 4.17 illustrates the simulated distribution of RDS values for these two codes. From [9], the RDS of this dc-free coset BCH code can be upper-bounded by $|RDS| \leq 51 + \lfloor 51/2 \rfloor = 76$. Figure 4.17 shows that for this code RDS values with probability 10^{-6} are still far from this bound. Note that the dc-free coset BCH code is nonsystematic and the generated matrix is column-permuted from a non-systematic BCH generator matrix. The code can be decoded by the following steps: (*i*) unpermuting the received code bits, which will make the code a nonsystematic BCH code; (*ii*) performing Berlekamp-Massey decoding [1], which will output a estimated BCH codeword ; (*iii*) acquiring the augmented source word by solving the equations obtained by the estimated codeword and the non-systematic BCH generator matrix, and (*iv*) recovering the source word by discarding the augmented bits. It is clear that step three may cause error extension.

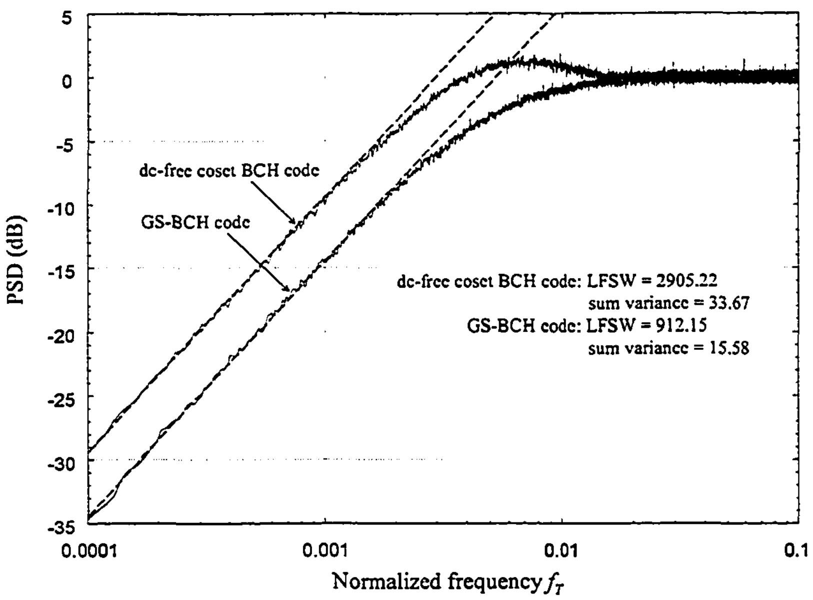


Figure 4.16 Simulated PSDs (solid lines) and approximate PSDs (dashed lines, based on simulated LFSW) of the dc-free coset BCH code and the GS-BCH code employing the (255, 239) BCH code with A=A_c=5.

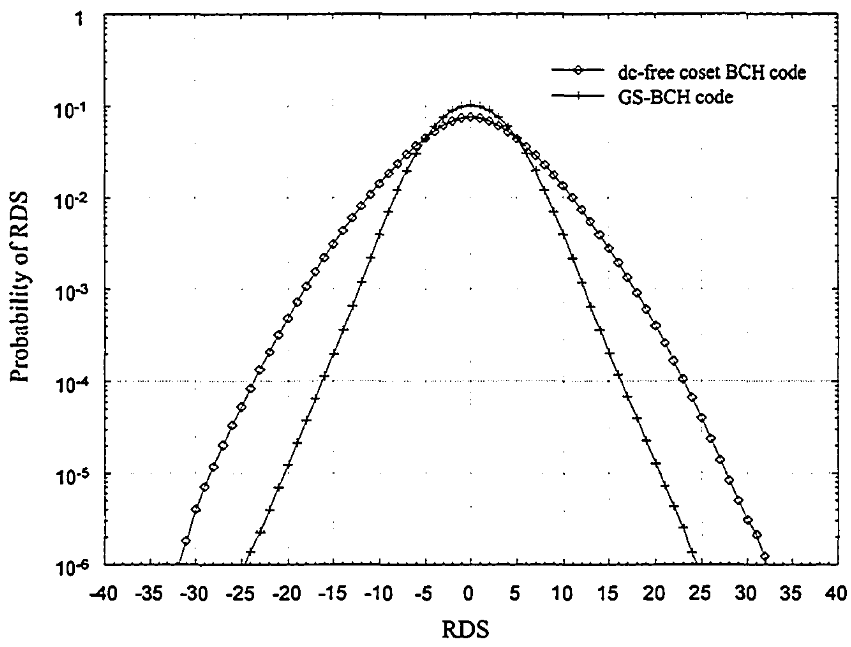


Figure 4.17 Simulated distribution of RDS values for the dc-free coset BCH code and the GS-BCH code employing the (255, 239) BCH code with A=A_c=5.

When compared to the dc-free coset code [9], the proposed new scheme is much more general, can be applied to a much wider range of codes, is significantly easier to implement, and can result in significant improvement in performance. Some of the differences between the proposed new codes and the dc-free coset codes are highlighted below.

- The proposed new technique is much more general. As described in [9], the dc-free coset code is limited to nonsystematic binary block codes and only construction of the binary dc-free BCH codes is given in the paper. As shown above, the new technique can be applied to non-binary codes as well as binary codes, as well as LDPC codes and product codes with iterative decoding. It is not yet clear how the coset coding technique could be extended to:
    - non-binary codes such as Reed-Solomon codes, because that method relies on the modification of a binary generator matrix;
    - LDPC codes, since LDPC code design is based on the construction of an appropriate parity check matrix, not the generator matrix. Modifications to the generator matrix required for dc-free coset coding will require alteration of the parity check matrix, which will reduce the performance of the LDPC code;
    - product codes, because it is not known how to add the redundant bits necessary for dc control in the dc-free coset coding scheme into product codes.
- The proposed new technique is significantly easier to implement. As discussed in the previous several sections of this chapter, standard EC block encoders and decoders can be used. This is in contrast to the technique described in [9], which as a result of column permutations of the generator matrix, eliminates the cyclic structure of BCH codes and therefore precludes the use of an encoding structure that has been designed for cyclic codes. Also, [9] does not allow for a polynomial interpretation of the BCH code, and therefore requires the use of matrix multiplications during encoding instead of the use of a shift register encoder. If the popular Berlekamp-Massey algorithm is used for the decoding of the dc-free coset BCH code, the received word needs to be permutated before decoding and the decoded codeword needs to be mapped to the source word through solving equations determined by the non-systematic generator matrix and the decoded codeword.
- The proposed new technique allows for a wide range of code lengths through efficient techniques for shortening the embedded EC code. As discussed earlier in this chapter, these shortening techniques require only minor modification of insertion and deletion of

zeros, and therefore still allow for use of standard and simple encoding and decoding techniques.

- The proposed new technique is more flexible than that of [9] in terms of code design and performance tradeoffs. For a given selection criterion, the spectral performance of dc-free EC block codes is largely determined by the number of redundant bits introduced for dc control. In dc-free coset codes, due to the manner in which the generator matrix is constructed, selection of the number of redundant bits is limited and code lengths are restricted. In the proposed scheme, the number of redundant bits can be selected to be any value as long as the value is less than the number of input bits to the EC block encoder.

In summary, the GS-EC block coding structure introduced in this chapter is a flexible coding technique that is capable of integrating dc-free constrained coding with a large number of different EC codes. The spectral properties of the encoded sequence are largely governed the properties of the GS code, while the BER performance is largely dependent on the error control performance of the EC code. In addition to its flexibility, it has been shown that this new coding structure outperforms the conventional approach to concatenating dc-free and EC codes and other previously developed integrated dc-free EC techniques.

Chapter 5

DC-Free Convolutional Codes and DC-Free Turbo Codes

In Chapter 4, a method was introduced for integrating GS dc-free codes and EC block codes that fully reverses the conventional order of dc-free decoding and EC decoding. This chapter extends the work in Chapter 4 and proposes methods of integrating dc-free codes and trellis-based EC codes such as convolutional codes [50] and turbo codes [54].

5.1 Encoders of Convolutional Codes and Turbo Codes, and the All-One Word

As discussed in the previous chapter, in order to generate dc-free GS-EC codes with the proposed block coding structure, it is required that GS encoders generate complementary pairs, that the GS scrambling polynomial has low weight, that appropriate selection criteria be chosen according to the number of augmenting bits, and that the EC block code has the all-one codeword. However, if convolutional/turbo codes are considered as EC codes in the dc-free GS-EC scheme shown in Figure 4.1, a problem arises since the all-one word is not, in general, an EC codeword in these EC codes. Therefore, further consideration is required in order to generate selection sets containing complementary codeword pairs. In this section, the encoders of convolutional codes and turbo codes and their trellis diagram are introduced, and the codeword corresponding to the all-one input word is analyzed.

Consider block-oriented terminated convolutional codes whose encoder is ensured to start from the zero state and is terminated in the zero state at the end of a block of input data. The encoders for these (n',k',ν) convolutional codes can be implemented with k'-input, n'-output linear sequential circuits with input memory ν. Let the number of interval input words (of length k') and the number of interval output words (of length n') be N_1 and N, respectively, where $N = N_1 + \nu$ to take into account termination. Therefore a block-oriented convolutional code is equivalent to an $(n'N, k'N_1)$ block code. Note that, omitting the termination bits, the code rate of a convolutional code is $R_{EC} = k'/n'$. This is the code rate that is commonly stated even when the convolutional code is terminated with additional termination bits.

A version of this chapter has been published in part in *Proceedings of the 2005 IEEE International Conference on Communication.* F. Zhai, Y. Xin, and I. J. Fair.

Conventional convolutional codes are non-recursive (NR) codes. It is well known that the BER performance of a non-recursive systematic convolutional (NRSC) code is worse than that of a non-recursive nonsystematic convolutional (NRNSC) code with the same constraint length at large SNR's [54]. But at low SNR's, the performance of NRSC codes is slightly better. NRNSC codes are the most commonly used convolutional codes. As discussed in Subsection 2.3.2, RSC codes are component codes of turbo codes. An RSC code that integrates the properties of NSC and SC codes can result in better BER performance than the corresponding NRNSC code for code rates greater than or equal to 2/3 [77]. Let $G(g_1, g_2)$ be the generator for a rate 1/2 convolutional encoder, where g_1 and g_2 denote two connection patterns that are commonly expressed in octal. Note that when considering the integration of convolutional codes into the new GS-EC code structure, NRNSC codes and RSC codes with $k' = 1$ will be considered in this chapter.

Figure 5.1 shows a shift-register based encoder structure for a rate 1/2 NRNSC code with ν= 2, and generator $G(7,5)$ in octal. Its corresponding trellis diagram is shown in Figure 5.2 with $N_1 = 6$ and $N = 8$. In this figure, the solid lines correspond to paths taken when input bit is a 0, and the dashed lines correspond to paths taken when the input bit is a 1. Labels on each line indicate the encoded output values. State values are determined by the contents of the encoding shift register. Note that with an NRNSC code and $k' = 1$, the code is terminated in the zero state at time N by padding ν zeros to the end of the length N_1 input data sequence. From Figure 5.2, it can be seen that when the input word is all-one, 111111, the corresponding output word is 11 01 10 10 10 10 01 11, which is not an all-one word. Further examination of the trellis indicates that the all-one output word cannot be generated, regardless of the input data sequence.

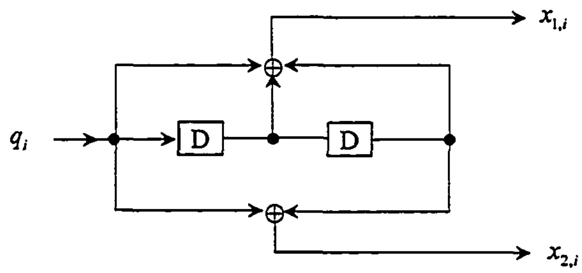


Figure 5.1 Rate 1/2 NRNSC code encoder with generator $G(7,5)$.

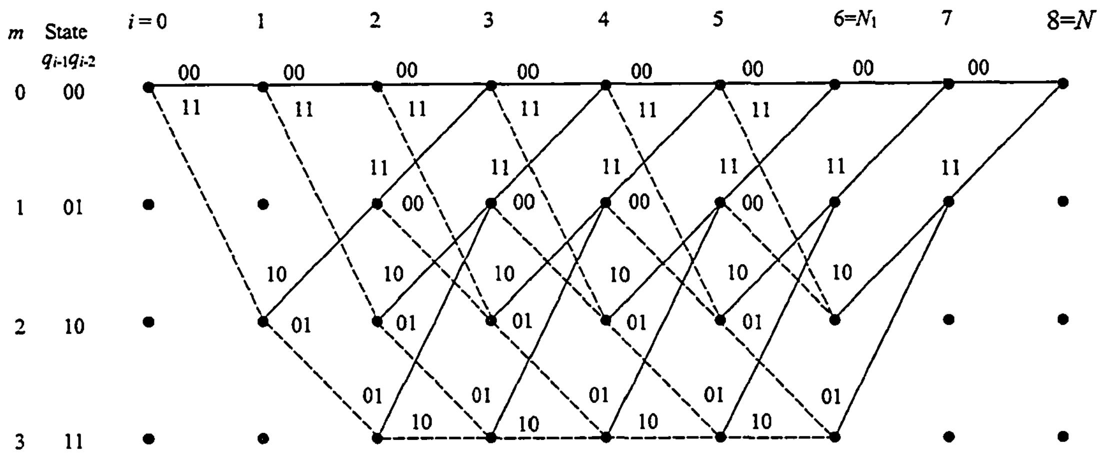


Figure 5.2 Trellis diagram for rate 1/2 NRNSC code with generator $G(7,5)$.

A binary RSC code can be obtained from an NRNSC code by using a feedback loop and setting one of the two outputs equal to the input bit. Figure 5.3 illustrates the RSC encoder corresponding to Figure 5.1. With this RSC code, there exist two output bits $x_{s,i}$ and $x_{p,i}$ at time i, where $x_{s,i}$ is the systematic bit and $x_{p,i}$ is the parity bit. The states of the RSC encoder are dependent on b_i, and not directly on q_i. Since the encoder is recursive, it is not sufficient to set the last v input bits to zeros in order to drive the encoder to the zero state; setting the last v bits of b_i to zeros will drive the encoder to the zero state. The last v input bits q_i required to ensure that the bits b_i are zeros can be calculated from the contents of the registers as follows. For the first N_1 bits, set the switch to position 1, and $x_{s,i} = q_i$, $i = 1,2,\ldots,N_1$. Then move the switch to position 2, such that $x_{s,i}$, $i = N_1 + 1,\ldots,N$, are obtained from the feedback path of the registers. During this period, b_i is always zero and this drives the state of the encoder to zero at time N.

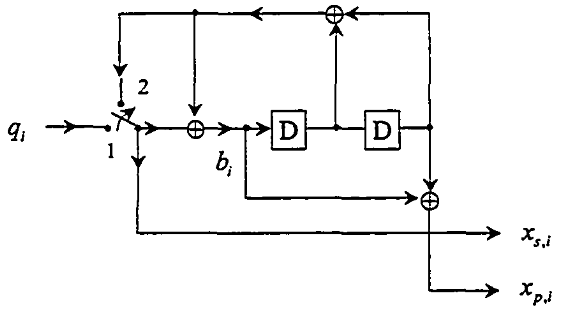


Figure 5.3 Terminated rate 1/2 RSC encoder with generator $G(7,5)$.

Figure 5.4 shows the corresponding trellis diagram. Note that the state connections and encoded sequence labels are identical to those in Figure 5.2; what differs is the input values that result in the various state transitions. Based on this trellis diagram, the output word is 11 10 11 11 10 11 00 00 when the input word is all-one. Note that since this code is systematic, the only possibility for generation of the all-one output word is when the input word is all-one. Therefore, it can be concluded that all-one words are generally not valid coded sequences for RSC codes.

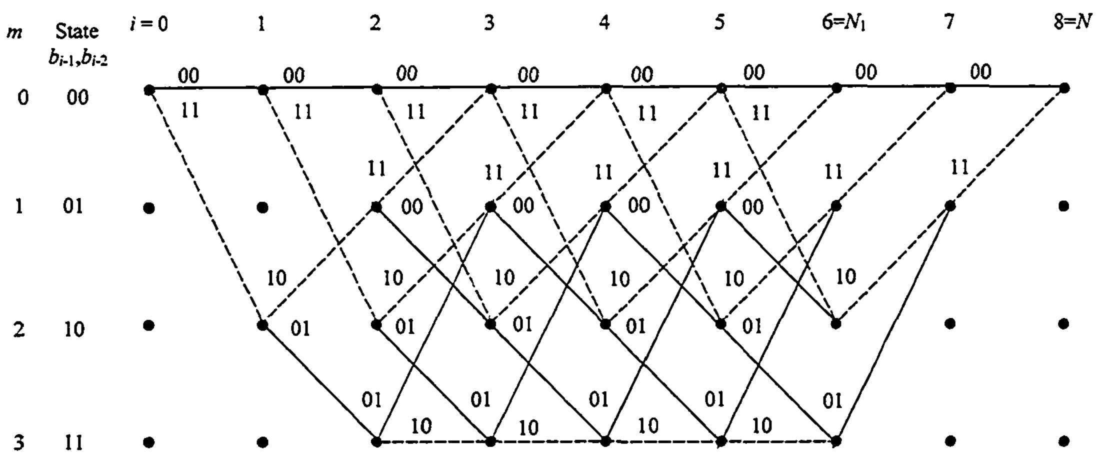


Figure 5.4 Trellis diagram for rate 1/2 RSC code with generator $G(7,5)$.

Figure 5.5 shows a rate 1/3 turbo encoder with generator $G(7,5)$, which consists of a random interleaver denoted by Π and two identical RSC encoders shown in Figure 5.3. The input of the first RSC encoder is the bit sequence x_s. The interleaver permutes this sequence to form the sequence x_s^I, which is the input sequence of the second RSC encoder. At time i, $x_{s,i}$ is the systematic bit of the encoded output, $x_{1p,i}$ is the parity bit generated by the first RSC encoder, and $x_{2p,i}$ is the parity bit generated by the second RSC encoder. Note that only the first RSC constituent encoder is terminated and the second one is left unterminated. The parity bits can be punctured according to a desired puncturing matrix to obtain a higher code rate. When puncturing is not used, the resulting code is a rate 1/3 (omitting the terminating bits) turbo code with codeword $c_t = \{x_{s,i}, x_{1p,i}, x_{2p,i}\} = (x_{s,1}, x_{1p,1}, x_{2p,1}, \ldots, x_{s,N}, x_{1p,N}, x_{2p,N})$.

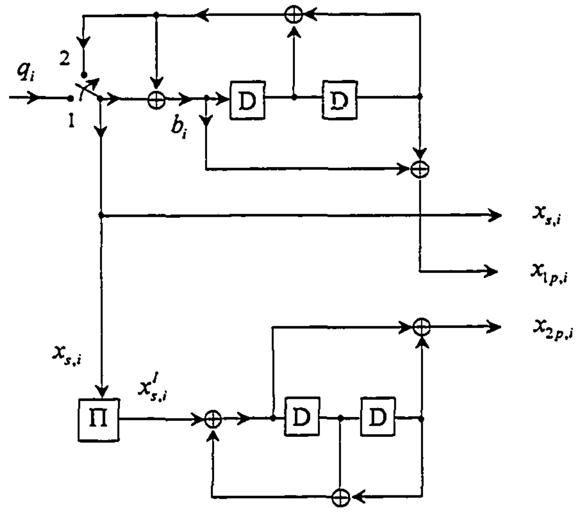


Figure 5.5 Turbo encoder with generator $G(7,5)$.

As an example consider this turbo encoder with $N = 8$ and $\Pi = \begin{bmatrix} 12345678 \\ 86254371 \end{bmatrix}$, where the first row of Π denotes the positions of $x_{s,i}$, and the second row of Π denotes the positions of $x_{s,i}$ after interleaving. When the input word is all-one of length $N_1 = 6$, Table 5.1 shows the generation of the turbo codeword. It is clear that the codeword is not all-one when the input word is all-one. Since this code is systematic, the only possibility for generation of the all-one output sequence occurs when the input sequence is all-one. Therefore, it can be concluded that all-one sequences are, in general, not valid encoded sequences of turbo codes.

Table 5.1 Generation of the turbo codeword when the input word is all-one

q	1 1 1 1 1 1
x_s	1 1 1 1 1 1 0 0
x_{1p}	1 0 1 1 0 1 0 0
x^I_s	0 1 1 1 1 1 0 1
x_{2p}	0 1 0 1 1 0 0 0
c_t	110 101 110 111 101 110 000 000

Based on above analysis, it is clear that the all-one codeword does not in general exist in NRNSC codes, in RSC codes, and in turbo codes. In the following two sections, methods will be proposed to generate the all-one word in order to permit the integration of these codes into the GS-EC structure outlined in Chapter 4.

5.2 DC-Free GS-Convolutional Codes

Based on the number of zeros in an NRNSC codeword which is related to the all-one input word, and based on the number of zeros in an RSC codeword which is closest in terms of minimum Hamming distance to the all-one word, methods to generate all-one words after EC encoding are proposed below.

5.2.1 DC-Free GS-NRNSC Codes

Let c_{I1} be the NRNSC codeword generated by the all-one input word of length N_1 and v terminating bits (note that in an NRNSC code, the v terminating bits are all-zero). As discussed in the previous section, c_{I1}, in general, is not the all-one word. By investigating the number of zeros in c_{I1}, the following two methods are proposed to construct the all-one output word.

(1) If the number of zeros in c_{I1} is relatively small, the zero bit positions can be punctured to generate the all-one output word. This puncturing method will result in a higher code rate at the cost of some decrease in EC performance.

(2) If the number of zeros in c_{I1} is relatively large, the zero bit positions can be flipped in order to generate the all-one output word. At the receiving site, detection based on the Viterbi algorithm can be used to determine if the transmitted word contains flipped bits. This flipping operation intentionally introduces significant additional noise, and therefore, it is not difficult for the Viterbi algorithm to detect if the bits are flipped when E_b / N_0 is not too low. With NRNSC codes, flipping detection and VA decoding can be combined. Note that this flipping method is used in [78] to construct a run-length constrained LDPC code without flipping detection at the receiver.

Based on generators of the code, a trellis diagram can be established for the analysis of the code. A trellis diagram which depicts N time intervals can be divided into three stages: the initial stage (the first v time intervals), the intermediate stage ($N_1 - v$ time intervals), and the terminating stage (v time intervals).

Table 5.2 lists the codewords generated by the all-one input word when $N_1 \geq 11$ for the rate 1/2 optimum NRNSC codes with memory from 2 to 8 [79]. In these codewords, the zeros indicate the positions for puncturing or flipping that will result in the all-one word. This table shows that in the intermediate stage, the output sequence is either the all-one sequence or the alternating sequence. In practice, ν is much smaller than N_1 and the sequence pattern corresponding to the intermediate stage will determine whether the codes should be punctured or flipped. Clearly, in order to obtain the all-one codeword, the codes in the table with memory 2, 3, 4, 5, and 7 should undergo flipping, and the codes with memory 6 and 8 should be punctured. Note that in all cases, additional puncturing can be applied to obtain higher rate codes.

Table 5.2 c_{I1} of NRNSC Codes

ν	Generator	c_{I1}		
		Initial stage	Intermediate stage	Terminating stage
2	G(7,5)	1101	101010...10	0111
3	G(17,13)	110110	010101...01	100011
4	G(27,31)	11100010	010101...01	10110111
5	G(57,65)	1110001101	101010...10	0100100111
6	G(117,155)	111010011000	111111...11	000101100111
7	G(237,345)	11101101110010	010101...01	10111000100111
8	G(657,435)	1101011110011000	111111...11	0010100001100111

Figures 5.6 and 5.7 present the new dc-free GS-NRNSC coding schemes with puncturing or flipping. As an extension of the discussion in Section 4.1, since both the GS encoder and the NRNSC encoder are linear, there exist predetermined additive patterns, which depend only on the GS scrambling polynomial, the NRNSC encoder, and the puncturing or flipping pattern, to implement additive GS-NRNSC encoding. A word c from $\{c_0, \bar{c}_0, c_1, \bar{c}_1, \ldots, c_{2^{A-1}-1}, \bar{c}_{2^{A-1}-1}\}$ is selected with a selection criterion to generate an EC protected dc-balanced sequence. Since decoding of the new coding scheme is completed first through Viterbi decoding and then through GS decoding, the Viterbi decoding is independent of the GS decoding, therefore the BER performance of the new techniques is governed largely by distance properties of the NRNSC code.

Now consider the additive patterns which generate the dc-free GS-NRNSC sequence in the new schemes. Note that the first additive pattern is zero. Let $\{g_0, \bar{g}_0, g_1, \bar{g}_1, \ldots, g_{2^{A-1}-1}, \bar{g}_{2^{A-1}-1}\}$ be the 2^A GS quotients when the source word is set to be all-zero; note that they are complementary pairs since $d(x) = x^A + 1$ is assumed. Let $\{z_0, \bar{z}_0, z_1, \bar{z}_1, \ldots, z_{2^{A-1}-1}, \bar{z}_{2^{A-1}-1}\}$ denote the 2^A NRNSC codewords generated when the inputs to the NRNSC encoder are $\{g_0, \bar{g}_0, g_1, \bar{g}_1, \ldots, g_{2^{A-1}-1}, \bar{g}_{2^{A-1}-1}\}$. For the puncturing approach as shown in Figure 5.6, the additive patterns $\{p_0, \bar{p}_0, p_1, \bar{p}_1, \ldots, p_{2^{A-1}-1}, \bar{p}_{2^{A-1}-1}\}$ are constructed by puncturing the words $\{z_0, \bar{z}_0, z_1, \bar{z}_1, \ldots, z_{2^{A-1}-1}, \bar{z}_{2^{A-1}-1}\}$ according to the puncturing pattern determined as outlined above. At the receiver, erasures are inserted into the positions where the bits have been punctured before Viterbi decoding [50]. After Viterbi decoding, GS decoding is performed to obtain the estimate of the source word.

For the flipping approach shown in Figure 5.7, the additive patterns $\{f_0, \bar{f}_0, f_1, \bar{f}_1, \ldots, f_{2^{A-1}-1}, \bar{f}_{2^{A-1}-1}\}$ are obtained by setting $f_i = z_i$ and letting $\bar{f}_i$ be equal to the flipped form of $\bar{z}_i$ according to the flipping pattern for the code, as described above. Since only some of the bits in the 2^{A-1} words are flipped, this approach is called partial flipping. Since no information is transmitted to indicate whether or not the transmitted codeword contains flipped bits, both the received word and the received word with inverted values in the flipped positions are decoded with parallel Viterbi decoders, and the word corresponding to the most likely encoded codeword (the one with the better path metric) is forwarded to the GS decoder.

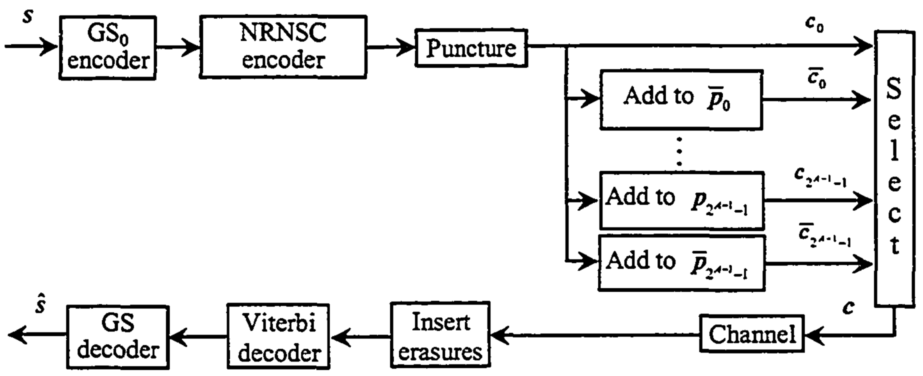


Figure 5.6 Block diagram of the new GS-convolutional coding scheme with puncturing.

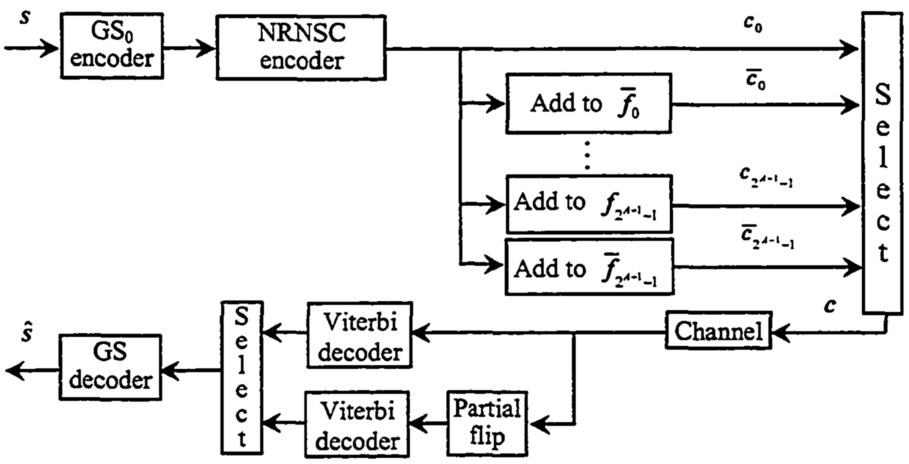


Figure 5.7 Block diagram of the new dc-free GS-NRNSC coding scheme with partial flipping.

Figure 5.8 shows a simplified structure corresponding to Figure 5.7 which reduces the number of the stored additive patterns by half. Once c is obtained, it is complemented in its entirety to generate $\bar{c}$. At the receiving site, both noisy versions of c and $\bar{c}$ are decoded, and the most probable outcome is forwarded to the GS decoder. Therefore, compared to the partial flipping approach, the total flipping approach does not require knowledge of the flipping pattern at the receiver.

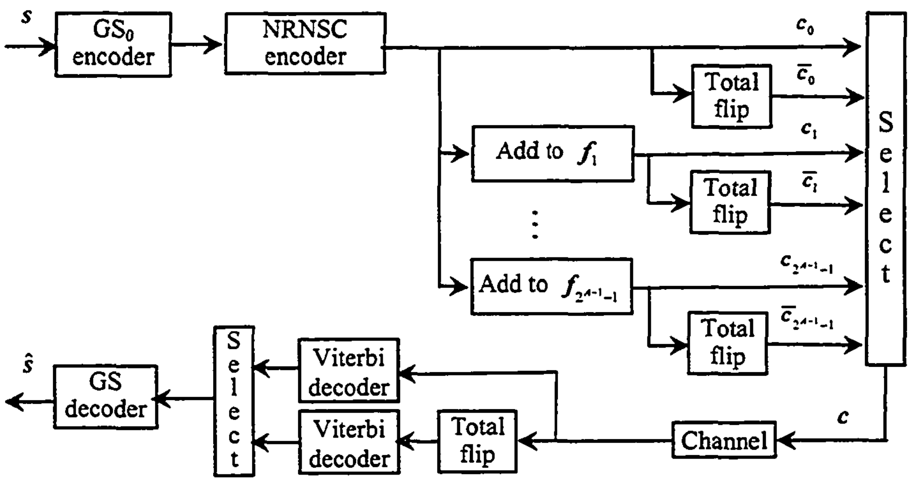


Figure 5.8 Block diagram of the new dc-free GS-NRNSC coding scheme with total flipping.

It is straightforward to show that the encoding and decoding procedures outlined in Figures 5.7 and 5.8 are equivalent in terms of sequence generation and error control, other than the detection of the appropriate sequence for decoding. Note that when the E_b/N_0 is not very low and when the correct detection is made, Figures 5.7 and 5.8 are fully equivalent. To further demonstrate this equivalence, Figure 5.9 illustrates specific encoding and decoding procedures when the number of augmenting bits is $A=1$. Note that the received word is $\boldsymbol{y}=\boldsymbol{c}+\boldsymbol{n}$ where $\boldsymbol{c}$ denotes the selected codeword and $\boldsymbol{n}$ denotes the AWGN noise. In this figure, $\boldsymbol{n}'$ denotes the partially flipped version of $\boldsymbol{n}$, and $\bar{\boldsymbol{n}}$ denotes the totally flipped version of $\boldsymbol{n}$. Note that these flipping operations on $\boldsymbol{n}$ do not change the mean and variance of the noise, therefore these three noise processes are statistically equivalent in terms of their impact on system performance. Also in this figure, c_0 and $\tilde{c}_0$ are two NRNSC codewords corresponding to the GS complementary quotients q_0 and $\bar{q}_0$ respectively, $\bar{c}_0$ is the complement of c_0 and they are related through total flipping, $\bar{c}_0$ and $\tilde{c}_0$ are related with partial flipping, c_0' is the partially flipped version of c_0, and $\hat{s}$ is the estimate of s.

Encoding:

When $c = c_0$, $y = c_0 + n$, and when $c = \bar{c}_0$, $y = \bar{c}_0 + n$.

Decoding with partial flipping:

Decoding with total flipping:

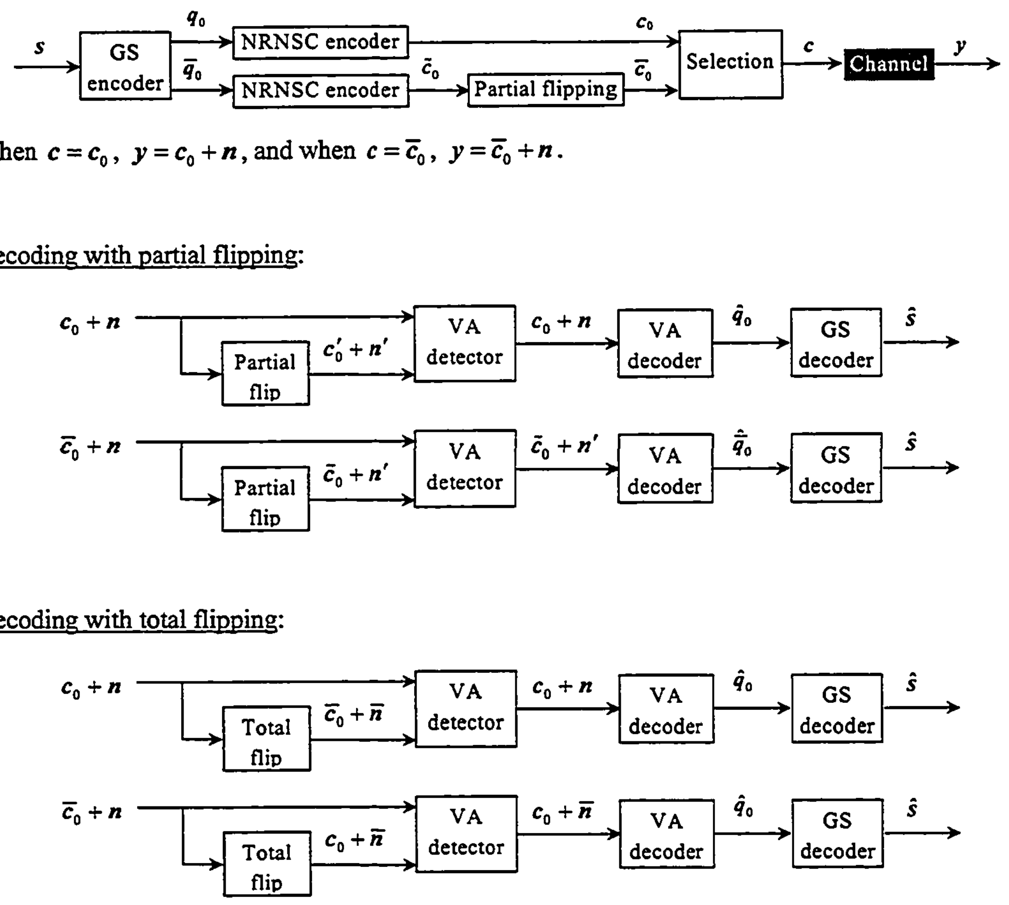


Figure 5.9 Illustration of the dc-free GS-NRNSC coding scheme with partial/total flipping.

Note that in Figure 5.9, the VA detector and VA decoder are separated in order to illustrate the processing of the data clearly, but these two processing units can be integrated into one when the EC codes are convolutional codes. However, it should be pointed out that since the VA detector with partial flipping and total flipping are making decisions between different pairs of sequence, their performance could vary, which could impact the overall system BER. Simulations reported in Section 5.4 indicate that this difference in performance is negligible, however, and therefore the partial flipping and total flipping approaches can be considered, for practical purpose, to be equivalent.

5.2.2 DC-Free GS-RSC Codes

As noted previously, with RSC codes it is not sufficient to set the last v systematic bits to zero in order to drive the encoder to the zero state; the values of the terminating bits are determined by the state at time N_1. For example, for RSC G(7, 5) codes, there are four patterns of terminating bits: 00, 01, 10, and 11, where the values of the required terminating bits of the RSC encoder depend on the value of N_1. Therefore, with partial flipping, unlike the NRNSC codes, the flipping pattern corresponding to the terminating bits is dependent on the value of N_1.

It can be shown that when the input word is the all-one word, the output of the RSC encoder is a periodic sequence or a truncated periodic sequence since the feedback in the RSC encoder constitutes a division process. Let $G_f(x)$ of degree ν be the irreducible feedback polynomial used by the RSC encoder. It can be shown that any irreducible polynomial of degree ν divides $x^{2^\nu-1}+1$ in GF(2) [50]. As discussed in Section 4.2, x^n+1 equals the product of $x+1$ and $w_{n-1}(x)$. Since in general ν is greater than one, $w_{n-1}(x)$ is a multiple of $G_f(x)$ when n is a multiple of $2^\nu-1$. From the trellis structure of the RSC codes, it follows that the all-one input sequence causes the pattern in the change of states to be periodic or truncated periodic. Since the initial state is all-zero, a periodic sequence will return to all-zero state at the end of each period. If $G_f(x)$ is primitive, it can be shown that the period of the output sequence is $2^\nu-1$; if $G_f(x)$ is irreducible but not primitive, the smallest period is less than $2^\nu-1$ and divides $2^\nu-1$. If $N=B\cdot(2^\nu-1)$ where B is an arbitrary positive integer, the terminating bits of the RSC encoder will be all-one for the all-one input word since the all-one input drives the encoder into the all-zero state at these values of N.

Recall that with an NRNSC code the decision whether the codes should be punctured or flipped to generate a dc-free GS-NRNSC code is determined by the codeword c_{I1}. In the case of RSC codes, c_{I1} is a periodic or truncated periodic sequence due to the recursive nature of the encoder, and is not always helpful in determining which approach should be used since when flipping is involved the accuracy of detection should also be considered. Consider the RSC codeword, c_{C1}, which is an RSC codeword and is closest in terms of minimum Hamming distance to the all-one word. c_{C1} can be obtained using a Viterbi decoder by evaluating the most likely encoded path when the input to the decoder is the all-one sequence. If c_{C1} is a periodic or truncated periodic sequence containing zeros, the number of zeros in c_{C1} will increase as the word length increases and flipping becomes the preferred approach to generate dc-free GS-RSC codes. If the intermediate stage of c_{C1} is the all-one sequence, then it is appropriate to use puncturing to generate the dc-free GS-RSC codes.

An equivalent method to determine whether flipping or puncturing is appropriate is based on the output of the all-one state of the RSC encoder. The total number of states in each RSC code is 2^{ν} and these states can be denoted by binary digits of length ν from all-zero to all-one. Let $k' = 1$. The all-one state will transit to the all-one state while generating an interval output word of length n'. If this interval output word, which includes the systematic bit, is all-one, there exists a codeword whose intermediate stage is all-one and puncturing is an appropriate method; if this interval output word is not all-one, flipping is more suitable. For example, RSC codes with generator G(13,15) are well suited to puncturing and RSC codes with generator G(13,17) are well suited to flipping.

Table 5.3 lists c_{C1} when $N = 240$ for the most commonly used rate 1/2 RSC codes with memory $\nu = 2, 3, 4$. The value of c_{C1} may be divided into three segments: the segment with the repetitive pattern, the segment before the repetitive pattern, and the segment after the repetitive pattern. Note that the values of c_{C1} with G(7,5) and G(13,17) consists of only the repetitive patterns when $N = 240$.

Table 5.3 c_{C1} of RSC Codes

ν	Generator	c_{C1}		
		Segment before the repetitive pattern	Segment with the repetitive pattern	Segment after the repetitive pattern
2	G(7,5)		111011...111011	
3	G(13,15)	111000	1111...1111	000111
	G(13,17)		11011111...11011111	
4	G(37,21)	111001	1111111110...1111111110	11111111011011
	G(31,33)	11111011110101011	11111110110111...11111110110111	11111110110111

Therefore, with RSC codes the decision whether the codes should be punctured or flipped to generate a dc-free GS-RSC code is determined by the codeword c_{C1}. When the flipping scheme is selected, c_{I1} is still used to determine the flipping positions when partial flipping is performed. In this manner, RSC codes can be used in the new dc-free GS-EC scheme depicted in Figures 5.6 through 5.8 with the NRNSC encoders replaced by RSC encoders.

5.3 DC-Free GS-Turbo Codes

Since the component codes of turbo codes are convolutional codes, the all-one input word generally does not result in the all-one output word. Similar operations as described above to generate dc-free convolutional codes, including puncturing or flipping, can be used to generate dc-free turbo codes.

Two important issues which affect the design of dc-free turbo codes include termination and interleaving. For an interleaver, if the input is all-one, the output is also all-one, indicating that a pair of complementary words at the input of an interleaver results in a pair of complementary words at its output. In turbo codes the first RSC code is usually terminated and the second can be left unterminated. Since the component encoders in turbo codes are recursive, the values of the terminating bits are determined by the state at time N_1. Since the terminating bits of the first RSC constituent encoder are possibly not all-one when the input word to the first RSC constituent encoder is the all-one word, additional flipping in the partial flipping scheme may be required to ensure the input word to the interleaver is all-one.

As discussed in the previous section, when the input word is the all-one word, the output of the RSC encoder is a (possibly truncated) periodic sequence. If the degree of the irreducible feedback polynomial is ν, the period of the output sequence is $2^{\nu}-1$. If $N=B\cdot(2^{\nu}-1)$ where B is an arbitrary positive integer, the terminating bits of the RSC encoder will be all-one for the all-one input word. Therefore, additional flipping in the partial flipping scheme is not required when N is a multiple of $2^{\nu}-1$. For example, based on the trellis diagram shown in Figure 5.4, the output sequence of the rate 1/2 RSC code with generator G(7,5) and $\nu=2$ has a period of 3 and when $N_1=3B-2$, the two terminating bits are all-one.

Since termination and interleaving do not involve the second constituent RSC encoder of the turbo encoder, determination of whether puncturing or flipping should be used for the entire turbo code can be determined by considering only the characteristics of the RSC constituent codes.

Note that the effect of the interleaver must be taken into consideration when the puncturing method is used to generate dc-free GS-turbo codes. For example, consider turbo codes with G(13,15) and $\nu=3$. The data sequence of length $N=B\cdot(2^{\nu}-1)$ associated with c_{C1} is x_s=110 1111...1111 001. The three zeros in this sequence need to be considered due to the existence of the interleaver. The turbo encoder used in this puncturing scheme can be modified as follows:

- Choose the size of the interleaver and the length of input word to be $N-\nu-1$.
- Pass the input word to the interleaver.
- Insert a zero at position 3 into the input words to both RSC encoders.
- Terminate both RSC encoders.

- Puncture the positions of the codeword according to the puncturing pattern 111100000 1111...1111 000011111. Note that zeros in this puncturing pattern correspond to the bits which should be punctured at the output of the turbo encoder.

This is sufficient to allow generation of dc-free GS-turbo codes with puncturing when a GS encoder precedes this modified turbo encoder.

The block diagram for GS turbo coding with puncturing is very similar to Figure 5.6 with the NRNSC encoder replaced by the modified turbo encoder described above, and with the Viterbi decoder replaced with a modified turbo decoder that, if required, will remove bit(s) from the turbo decoded word (a bit at position 3 associated with step 3 of the encoding process in the above example).

For the flipping approach, either partial flipping or total flipping can be used. For partial flip encoding, if $N = B\cdot(2^{v}-1)$, only one flipping pattern for the codeword is required. Otherwise, two flipping patterns are required: one for flipping terminating bits to allow the terminated systematic input word to pass through interleaver, and the other for flipping codewords. These two flipping patterns cannot be combined due to the interleaver. The flipping patterns are determined by c_{l1}. In contrast, total flipping is easy to implement and the flipping patterns are not required during encoding and sequence recovery. Figure 5.10 shows the block diagram for encoding and decoding of dc-free GS-turbo codes with total flipping. The additive patterns $\{f_{t,0}, f_{t,1}, \ldots, f_{t,2^{A-1}-1}\}$ are the 2^{A-1} turbo codewords generated when the input words of the turbo encoder are $\{g_0, g_1, \ldots, g_{2^{A-1}-1}\}$ as defined in Section 5.2.

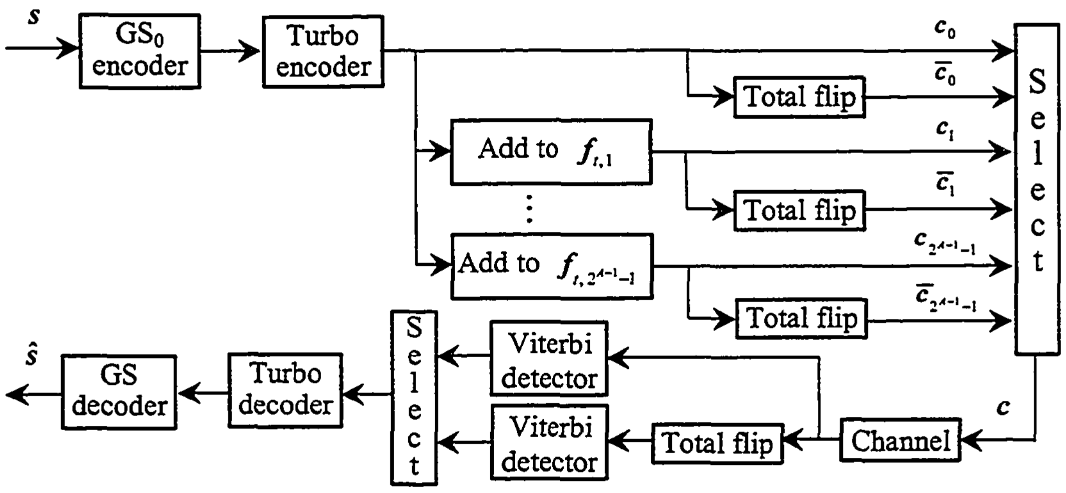


Figure 5.10 Block diagram of the new dc-free GS-turbo coding scheme with total flipping.

Note that, by using methods similar to those described above for generation of dc-free GS-turbo codes, SCCCs can also be integrated with GS codes to form dc-free GS-SCCC with puncturing or flipping.

Let the detection error rate (DER) be defined as the number of detection word errors that occur when the receiver mis-detects the flipping status of the transmitted words, divided by the total number of received words. DER is critical in the flipping schemes: decoding performance will be satisfactory only when the DER of flipping detection is much smaller than the BER of the decoder. The larger the number of zeros in c_{C1}, which indicates the distance to the all-one word, the better the DER performance is expected to be. In the following investigations, it is found that a Viterbi detector is sufficient to provide a good DER in the dc-free GS-convolutional/turbo coding schemes proposed above.

5.4 Performance of DC-Free GS-Convolutional/Turbo Codes

This section presents the spectral and error performance of GS-convolutional codes and GS-turbo codes. In these results, equiprobable source data logic values are assumed unless p_0, the probability of a logic 0 in the source sequence, is specifically indicated. Since implementation of the schemes with puncturing is straightforward, only the performance of the schemes with flipping is considered here.

Figure 5.11 shows the PSD performance of two GS-convolutional codes with partial flipping; the same spectra would result with total flipping. Both codes use the same NRNSC code with $G(7,5)$ and N_1 = 32, however, the GS parameters differ. In one code, the number of augmenting bits is $A=1$, the scrambling polynomial is $d_{2,1}(x)=x+1$, and the selection criterion is MRDS; in the other code, A = 2, the scrambling polynomial is $d_{2,2}(x)=x^2+1$, and the selection criterion is MSW. This figure demonstrates that GS-convolutional codes have a spectral null at dc, and that with very little redundancy, low frequency components can be effectively suppressed. It also indicates that spectral performance is highly dependent on the parameters of the GS code.

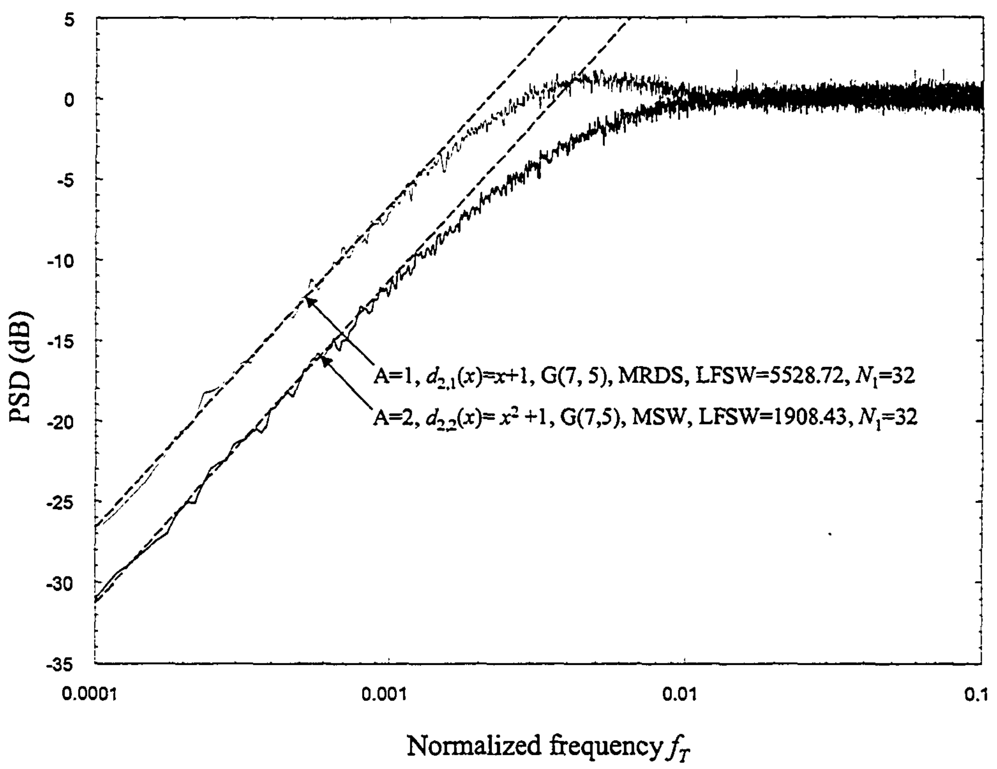


Figure 5.11 Simulated PSDs (solid curves) and approximate PSDs (dashed lines, based on the simulated LFSWs) of the dc-free GS-convolutional codes using partial flipping.

Figure 5.12 illustrates the simulated BER and DER performance of the GS-convolutional (NRNSC) code over the AWGN channel with coding parameters $A = 1$, $d_{2,1}(x) = x + 1$, G(7,5), $N_1 = 32$, partial flipping, and Viterbi decoding with eight-quantization-level soft decision. This figure demonstrates that the DER decreases faster than the BER as E_b / N_0 increases, and that when E_b / N_0 equals 5 dB, the DER is three orders of magnitude lower than the BER indicating that the DER has only a minor impact on BER performance. Figure 5.12 also compares the BER performance of this GS-convolutional code with the same convolutional code which is not dc-balanced. This figures shows that when the BER equals 10^{-5}, the dc-free GS-convolutional code has less than 0.4 dB loss in E_b / N_0 on the AWGN channel compared to the corresponding convolutional code without any spectral null constraints. This degradation in performance on the AWGN channel is the result of the additional redundancy in the GS-convolutional code, and the error extension encountered with GS decoding. As with the GS-EC block codes discussed in

Chapter 4, and the GS-turbo codes discussed below, the benefit of GS-convolutional coding becomes evident on noisy dc-constrained channels.

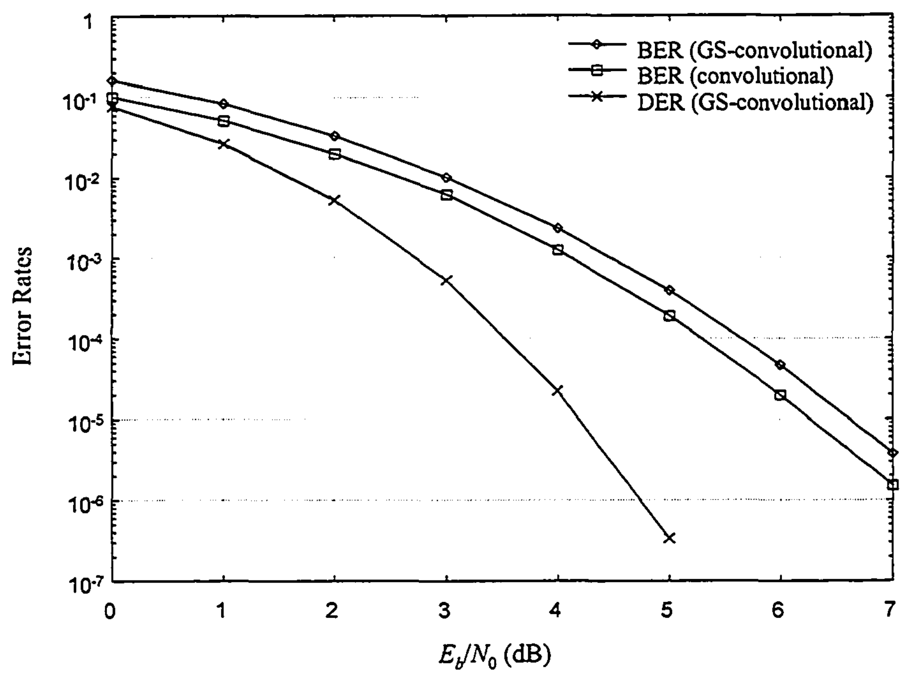


Figure 5.12 BER and DER performance of a dc-free GS-convolutional code and BER of the convolutional code on the AWGN channel.

The performance of the new dc-free GS-turbo coding scheme is shown in Figures 5.13 through 5.17. The simulation parameters include: (1021, 1013) GS code with $A = 8$ and $d_{2,8}(x) = x^8 + 1$; rate 1/3 turbo codes with G(13,17) and $N = 1024$, spread random (SR) interleaver [80], termination of only the first RSC encoder; MSW selection criterion; Viterbi detection with 8 quantization levels; Log-MAP turbo decoding with 10 iterations.

Figure 5.13 depicts the PSD performance of the dc-free GS-turbo scheme with total flipping. Note that the schemes with partial flipping and total flipping will yield identical spectral performance. Figure 5.14 shows that a dc-free GS-SCCC scheme with similar parameters provides almost the same spectrum performance as the GS-turbo scheme.

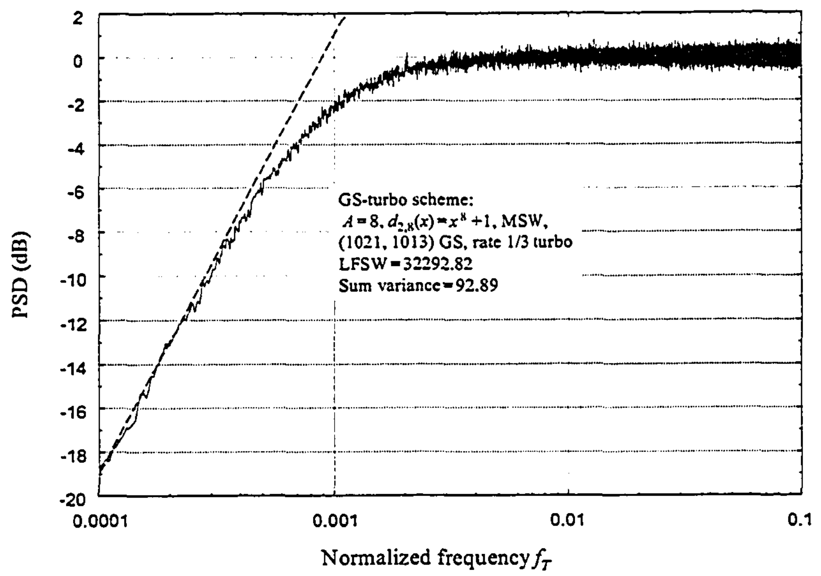


Figure 5.13 Simulated PSD of the dc-free GS-turbo scheme.

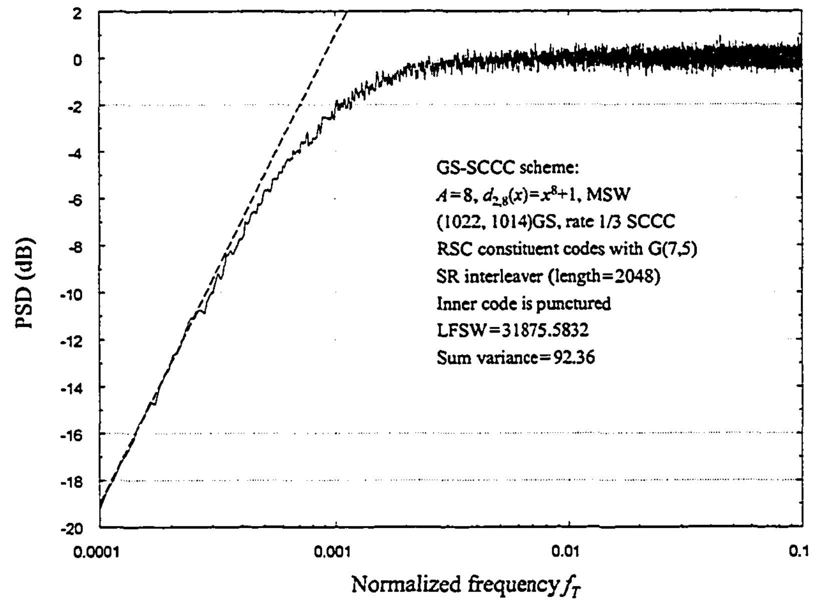


Figure 5.14 Simulated PSD of the dc-free GS-SCCC scheme.

Figure 5.15 illustrates the BER performance of the turbo code without dc-balancing over the AWGN channel and the HP-AWGN channel with $\tau_T = 120$, with different source probabilities. It can be seen that the BER performance of the turbo code is degraded over the HP-AWGN channel, especially when the source probabilities are not equal.

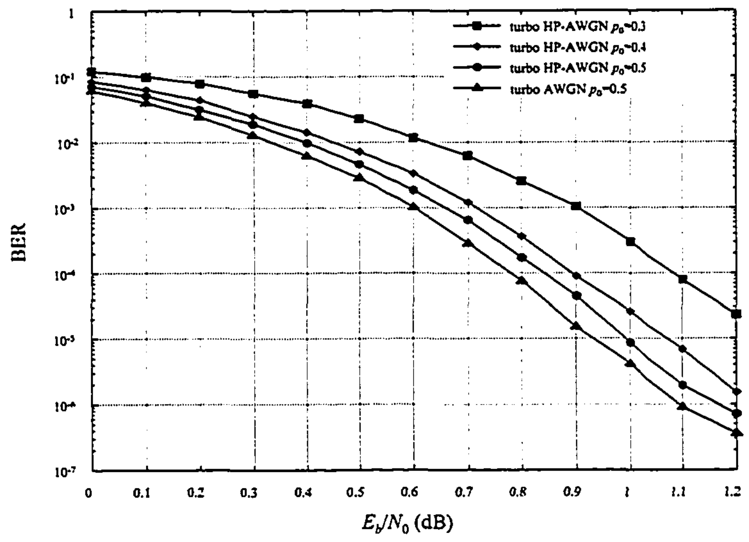


Figure 5.15 BER of turbo codes over the AWGN and the HP-AWGN channels with different source probabilities.

Figure 5.16 shows the BER and DER performance for the dc-free GS-turbo code. Code performance changes only slightly when the channel is changed from AWGN to HP-AWGN with $\tau_T = 120$, and is independent of source probabilities. As shown, this coding scheme with partial flipping (pF) or total flipping (tF) provides almost identical DER performance, therefore BER performance is reported only for one of these strategies, the code implemented with total flipping.

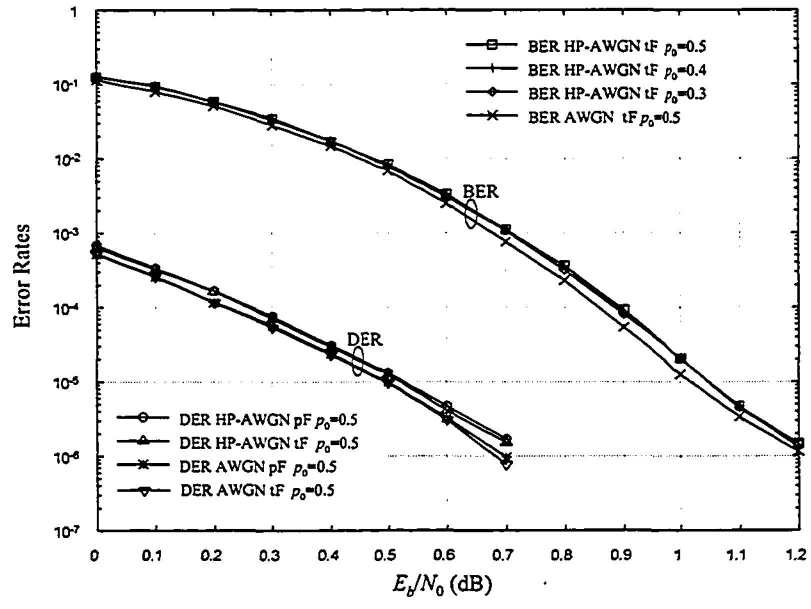


Figure 5.16 BER of the dc-free GS-turbo code with total flip over the AWGN and the HP-AWGN channels with different source probabilities, and DER of this code over these two channels with different flipping methods.

Figure 5.17 compares the BER performance of turbo codes and dc-free GS-turbo codes over the HP-AWGN channel. It can be seen that the change of source probability affects the performance of turbo codes significantly. A performance loss of 0.25 dB in turbo codes is observed at a BER of 10^{-4} when p_0 is changed from 0.5 to 0.3. However, the performance of GS-turbo codes does not vary with a change of p_0. This indicates the usefulness of dc-free EC codes over the dc-constrained noisy channel. The reason that the BER of dc-free GS-turbo codes is worse than that of turbo codes alone at $p_0 = 0.5$ is because of the error extension of GS decoding and the rate loss due to the introduction of the augmenting bits. However, note that turbo codes alone cannot provide a spectral null at dc.

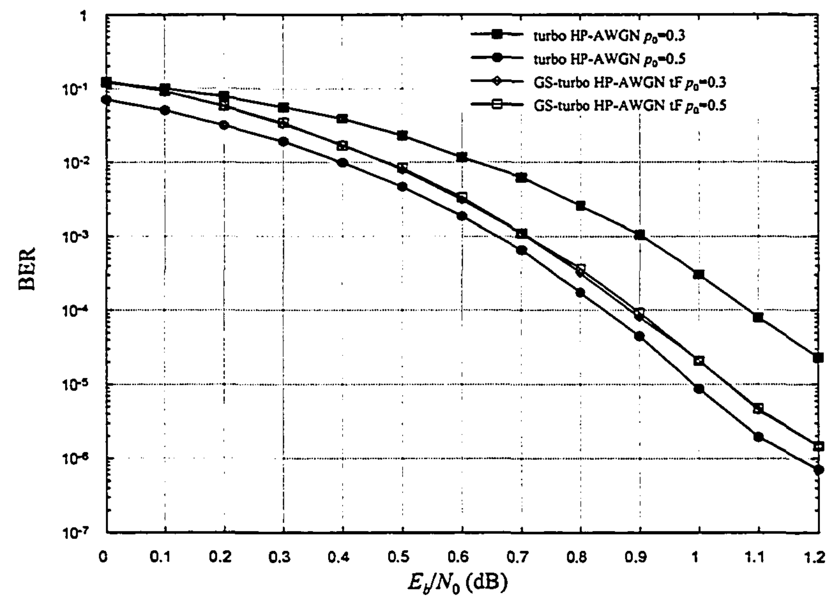


Figure 5.17 BER comparison of turbo codes and GS-turbo codes over the HP-AWGN channel with different source probabilities.

Figures 5.18 and 5.19 compare the SV and LFSW calculated from relations (4.4) and (4.5) with simulated sum variance and LFSW for GS dc-free codes. These figures also show simulated sum variance and LFSW for dc-free GS-block/convolutional/turbo codes where the block codes used as EC codes are (255, 239) RS over GF(2^8) (binary block length $n_b = 255 \cdot 8 = 2040$), (504, 252) binary LDPC, and (255, 239) binary BCH, and the trellis-based codes used as EC codes are rate 1/2 convolutional ($n_b = 2N = 68, N = 34$), and rate 1/3 turbo codes ($n_b = 3N = 3072$, $N = 1024$). These figures show that the simulation results of the GS codes and the GS-EC codes are very close to that of calculation, demonstrating that relations (4.4) and (4.5) can also be applied to dc-free GS-convolutional/turbo codes to accurately estimate sum variance and LFSW values.

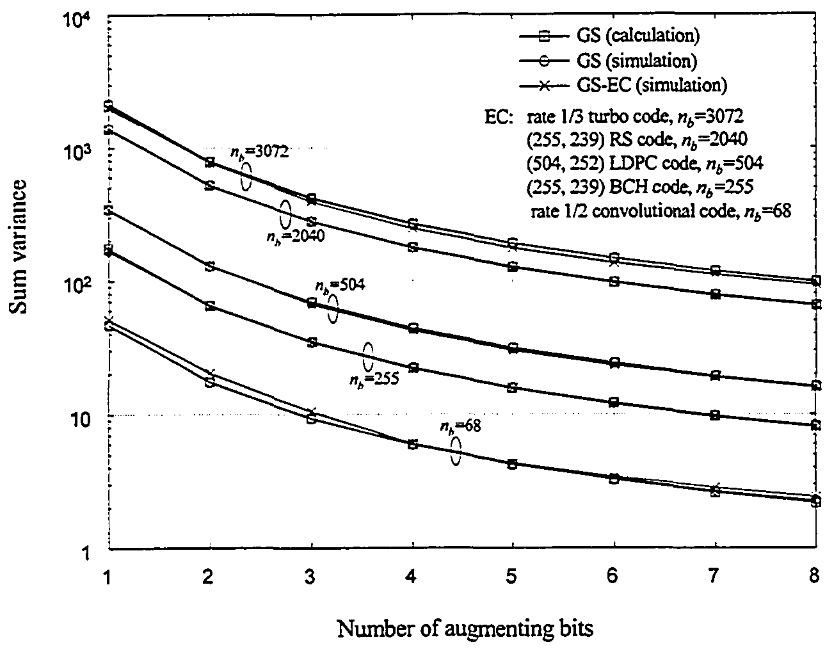


Figure 5.18 Sum variance of GS dc-free GS codes (calculation and simulation), and simulated sum variance of five different dc-free GS-EC codes.

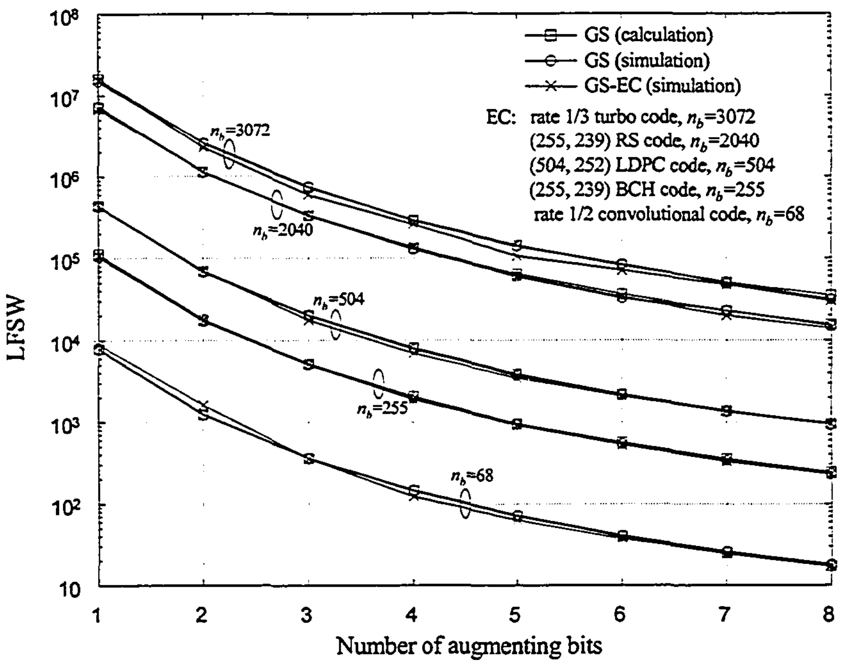


Figure 5.19 LFSW of GS dc-free multimode codes (calculation and simulation), and simulated LFSW of five different dc-free GS-EC codes.

Collectively, these performance results confirm that the new integrated coding technique developed in this thesis efficiently and effectively combines dc-free GS coding with convolutional or turbo EC coding, in order to guarantee that the encoded sequence exhibits a spectral null at dc, as well as contains the characteristics required for error correction. Since convolutional and turbo codes are widely used and dc-free data sequences are required in a variety of digital communication systems, these EC codes with their integrated dc-free property will be very useful for digital communication systems.

Chapter 6

Conclusion

This thesis has introduced new dc-free GS-block/convolutional/turbo coding techniques for digital transmission and storage systems. This chapter summarizes the development of these techniques and provides suggestions for further work.

6.1 Thesis Summary

Chapter 1 introduced the requirements for dc-free EC coding techniques for digital transmission and storage systems, the recent research activities related to these techniques, the basic ideas for the new general approaches for integrating dc-free GS codes and EC codes, and the organization of this thesis.

Chapter 2 summarized background information that was required in subsequent chapters. The capacity of dc-free coding, its performance metrics, and the classification of dc-free coding techniques was introduced. Since GS is used to provide dc control in the proposed new technique, it was discussed in detail, including its encoding and decoding procedures, the impact of its scrambling polynomial, the error extension during GS decoding, the selection criteria for the GS quotients, and its performance. In general, the performance of dc-free coding is evaluated by the PSD of the coded sequence. Methods to estimate the power spectrum of a random sequence were also introduced in this chapter.

EC codes also play an important role in the proposed new technique. In Chapter 2, the channel capacity, and the classification, description, fields, properties, and the error rate performance evaluation of EC codes were described. Prior art of dc-free EC codes was also discussed. It was shown that the dc-free EC techniques developed to date have limited EC ability and/or have limited flexibility regarding tradeoffs between spectral performance and BER performance.

Chapter 3 developed a simple noisy dc-constrained channel model by use of a first-order RC HPF concatenated with AWGN. It was shown that the HPF can be used to model the restrictions imposed by the dc constrained channel with flexibility through selection of the normalized time constant of the RC circuit. Equations were given for calculation of the received symbol energy with the HP-AWGN channel using two different receiving filters .

Chapter 4 proposed the dc-free GS-EC block coding technique. This technique is based on the integration of GS encoder with an EC block code which contains the all-one codeword. Through the proposed integrated encoding scheme, a source word is encoded to 2^{A-1} pairs of GS complementary quotients, these quotients are encoded to 2^{A-1} pairs of complementary EC codewords, and an EC codeword is selected based on a selection criterion to ensure that the RDS of the coded sequence is bounded. Therefore, the input to the noisy constrained channel is a fully EC-protected dc-free sequence. In order to reduce encoding complexity, an equivalent form of the proposed scheme is possible based on the addition of predefined relationship sequences since both the GS encoder and the EC encoder are linear. It was shown that decoding of the received sequences is completed first through EC decoding and then through GS decoding. Therefore, soft EC decoding can be used to result in improved BER performance, the effect of error extension during GS decoding is limited, and the BER performance is mainly governed by the characteristics of the EC codes.

Since it is well known how to generate complementary quotients with GS coding, it was shown that this property, along with the existence of the all-one EC codeword, is sufficient to ensure that the coded sequence can be dc balanced. Therefore, in order to ensure that the new scheme will result in a spectral null at dc, it is critical that the all-one EC block codeword exist. It was demonstrated in this chapter that the all-one codeword exists in many well-established EC codes such as binary cyclic codes, binary primitive BCH codes, extended binary primitive BCH codes, RS codes, and Reed-Muller codes. Although the all-one codeword is not present in BCH/RS codes that are shortened in the usual manner, the all-one codeword exists with a different shortening technique developed in this chapter. It was also shown that with LDPC codes, when the parity-check matrix has even row weight, which is often the case, the all-one codeword exists. If each of the two component codes of a product code contains the all-one word, the all-one codeword exists for the overall product code. Note that all the EC codes discussed above can be used as component codes in a product code. Therefore, each of these EC block codes can be integrated with a GS code to generate a dc-free GS-EC block code according to the new structure developed in this chapter.

The performance of dc-free GS-EC block codes was presented at the end of Chapter 4. The PSD performance of the proposed coding scheme was compared to that of other schemes, and it was shown that the proposed coding scheme yields better PSD performance than other integrated dc-free EC coding schemes, and approximately the same PSD performance as EC codes and dc-free codes concatenated in the conventional manner. The BER performance was reported when an RS code was used as the component EC code to show that the proposed coding scheme yields coding gain when compared to the conventional concatenation of RS and GS codes. Results were also presented when an LDPC code is used in the proposed scheme to illustrate integration of soft-decision EC decoding, which is not possible, in general, in the conventional scheme.

Chapter 5 extended the work in Chapter 4 and presented methods of integrating dc-free GS codes with trellis-based EC codes such as convolutional codes and turbo codes. When convolutional codes or turbo codes are considered as EC codes in the dc-free GS-EC coding scheme proposed in Chapter 4, a problem arises since the all-one word is not, in general, an EC codeword in these EC codes. Commonly used convolutional codes include NRNSC codes and RSC codes. Based on an NRNSC codeword corresponding to the all-one input word, and an RSC codeword which is closest in terms of Hamming distance to the all-one word, methods including puncturing or flipping were proposed to assist in the generation of the all-one word for these two types of convolutional codes. There are two approaches that can be to be used for flipping: partial flipping and total flipping. Two important issues which affect the design of dc-free GS-turbo codes include termination and interleaving. These two issues need to be considered when the partial flipping method is used, but do not need to be considered when the total flipping method is used.

The spectral and error performance of the GS-convolutional/GS-turbo codes was given. In addition to the PSD performance and BER performance, the detection error rate was also given for schemes that involved flipping. It was shown that changes of the source probability affect the performance of turbo codes significantly, however, the performance of the dc-free GS-turbo codes does not vary with changes of source probability. This indicates the usefulness of dc-free EC codes over the noisy dc-constrained channel.

Regarding the design of good dc-free EC codes, it is helpful to distinguish between good performance in terms of EC, and good performance in terms of low frequency suppression. The proposed scheme uses standard EC codes (or slightly modified codes in the case of shortened codes) which dominate the EC performance of the code. Good EC performance is therefore obtained by the choice of a powerful EC code, which is independent of the parameters of the GS code. Good spectral performance is achieved with appropriate selection of the GS code. As in guided scrambling, the value of A, along with the scrambling polynomial and type of selection criterion used, dominates the spectral performance of these dc-free EC codes. It was found that the dependence on A for the new codes is very similar to the dependence on A for GS: the larger the value of A, the larger the number of alternatives in the selection set, therefore there is a greater opportunity to select a word with good characteristics, and the spectral performance of the code improves. This relationship, however, relies on the use of an appropriate scrambling polynomial. In this thesis, the use of the polynomials $d_{2,A}(x) = x^A + 1$ is recommended to ensure the presence of 2^{A-1} complementary pairs of words at the output of the GS encoder. Use of a linear EC block code which contains the all-one codeword, or a convolutional/turbo code with the assistance of puncturing/flipping, ensures the presence of 2^{A-1} complementary pairs of words in the selection set, which allows for selection from 2^A different words while ensuring that the sequence RDS is bounded.

6.2 Further Work

This thesis presents general techniques for the integration of dc-free GS codes and EC codes. Numerous well-established EC codes were investigated, and it was found that they can be used in the proposed techniques. Work remains regarding reducing the complexity of codeword selection, comparison of the proposed scheme with other schemes, integration of GS codes with other EC codes, and consideration of a dc-free RLL GS-EC scheme.

The following provides suggestions for further research work.

(1) In Chapters 4 and 5, the dc-free code property is introduced into standard EC codes through GS multimode coding techniques, at the cost of some increase in implementation complexity at the encoder. Throughout the simulations of the new schemes, the encoding implementation is based on the additive structures such as those shown in Figures 4.3, 5.6 through 5.8, and 5.10. With this encoding approach, it can be seen that the selection process dominates the encoding complexity. Methods should be considered to reduce the complexity of codeword selection. Since complementary words have opposing sign and equal magnitude of disparities, as shown in Table 4.2, and calculation of word disparity is essential for the WRDS and the MSW selection criteria, this property can be used to reduce the complexity. Also, instead of exhaustive evaluation of all candidate words in order to select a best one, an early stopping technique can be used based on a threshold determined from the simulation. However, further simulation is required to determine the appropriate

thresholds for different selection criteria and to find the tradeoffs between the spectral performance and this reduction in complexity.

(2) In this thesis, the proposed schemes were compared with dc-free coset coding scheme [9] and the conventional scheme [5]. The newly proposed schemes could also be compared with other dc-free EC coding schemes developed in the literature [4], [6], [7], [10], [11] for a more comprehensive comparison of the spectral performance, EC performance, and the complexity of encoding and decoding procedures.

(3) In this thesis, many EC codes were investigated to determine their suitability for inclusion in the proposed schemes, including binary cyclic codes, primitive binary BCH codes, RS codes, RM codes, LDPC codes, product codes, convolutional codes, turbo codes (PCCCs), and SCCCs. Other EC codes should be evaluated to determine if they can be included in the proposed schemes, such as Euclidean geometry (EG) codes and projective geometry (PG) codes, EG-LDPC codes, and PG-LDPC codes [1], etc.

(4) It has been shown that it is possible to design selection mechanisms for multimode codes that guarantee a primary criterion, such as generation of a dc-free sequence, and simultaneously ensure, with high probability, a second constraint, such as meeting runlength limitations [19]. Attempting to meet this second constraint usually results in some loss of performance in terms of the primary characteristic. Satisfying the dc-free constraint while attempting to satisfy a runlength constraint can be implemented in the new dc-free GS-block/convolutional/turbo codes as follows. Instead of choosing the best codeword in terms of dc-free characteristics, a subset of S codewords with the S best dc-free characteristics is formed. From this subset, the word that best satisfies the runlength constraint is selected. This approach does not guarantee that the desired runlength constraint will be met, but if S is sufficiently large, reasonable runlength constraints will be met with high probability. Note that this method of selecting a codeword will not result in the codeword with the best dc-free characteristics being selected in every encoding interval, therefore there will be some loss of performance in terms of spectrum compression at low frequencies. This selection technique can be further modified to guarantee that reasonable runlength constraints are met.

As indicated above, a subset of S codewords with the S best dc-free characteristics can be formed, and from this subset, the word that best satisfies the runlength constraint can be selected. If this word contains one or more sequences that violate the desired runlength constraint, a bit in the offending sequence(s) can be complemented to ensure that the runlength constraint is met. No attempt need be made to indicate to the decoder that this bit was complemented. Instead, it can be left to the EC decoder to correct this bit prior to GS decoding. This modified selection mechanism will result in some degradation in terms of BER performance, however, this degradation should be small when the size of the subset is chosen appropriately, and the runlength constraints enforced are not excessively tight. Further development, analysis, and simulation of this approach is necessary to clarify the observations and to determine performance results and tradeoffs.

Bibliography

[1] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, 2nd edition, Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[2] K. W. Cattermole, "Principles of digital line coding," *Int. J. Electron.*, vol. 55, pp. 3–33, July 1983.

[3] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2260–2299, Oct. 1998.

[4] A. Kokkos, A. Popplewell, and J. J. O'Reilly, "A power efficient coding scheme for low-frequency spectral suppression," *IEEE Trans. Commun.*, vol. 41, pp. 1598–1601, Nov. 1993.

[5] J. L. Fan and A. R. Calderbank, "A modified concatenated coding scheme with applications to magnetic data storage," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1565-1574, July 1998.

[6] W. G. Bliss, "Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation," *IBM Tech. Discl. Bull.*, vol. 23, pp. 4633-4634, 1981.

[7] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1389-1399, Sept. 1997.

[8] H. C. Ferreira, "Lower bounds on the minimum Hamming distance achievable with runlength constrained or dc free block codes and the synthesis of a (16, 8) d = 4 dc free block code," *IEEE Trans. Magnetics*, vol. MAG-20, pp. 881-883, Sept. 1984.

[9] R. H. Deng and M. A. Herro, "DC-free coset codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 786-792, July 1988.

[10] M. C. Chiu, "DC-free error-correcting codes based on convolutional codes," *IEEE Trans. Commun.*, vol. 49, pp. 609–619, Apr. 2001.

[11] T. Wadayama and A. J. H. Vinck, "DC-free binary convolutional coding," *IEEE Trans. Inform. Theory*, vol. 48, pp. 162-173, Jan. 2002.

[12] I. J. Fair and D. R. Bull, "DC-free error control coding through guided convolutional coding," in *Proc. 2002 IEEE Int. Symp. Information Theory*, 2002, p. 297.

[13] R. M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 1003–1010, Aug. 1998.

[14] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[15] I. J. Fair, W. D. Grover, W. A. Krzymien, and R. I. MacDonald, "Guided scrambling: a new line coding technique for high bit rate fiber optic transmission systems," *IEEE Trans. Commun.*, vol. 39, pp. 289–297, Feb. 1991.

[16] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, July, 1948.

[17] S. Benedetto and E. Biglieri, *Principles of Digital Transmission With Wireless Applications*, New York, NY: Kluwer Academic / Plenum Publishers, 1999.

[18] G. L. Pierobon, "Codes for zero spectral density at zero frequency," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 435-439, Mar. 1984.

[19] K. A. S. Immink, *Codes for Mass Data Storage Systems*, 2nd Edition, Shannon Foundation Publishers, The Netherlands, 2004.

[20] T. M. Chien. "Upper bound on the efficiency of dc-constrained codes," *Bell Syst. Tech. J.*, vol. 49, pp. 2267–2287, Nov. 1970.

[21] J. Justesen, "Information rates and power spectra of digital codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 457-472, May 1982.

[22] Y. Xin and I. J. Fair, "A performance metric for codes with a high-order spectral null at zero frequency," *IEEE Trans. Inform. Theory*, vol. 50, pp. 385-394, Feb. 2004.

[23] K. A. S. Immink and L. Patrovics, "Performance assessment of dc-free multimode codes," *IEEE Trans. Commun.*, vol. 45, pp. 293–299, Mar. 1997.

[24] K. A. S. Immink, *Coding Techniques for Digital Recorders*. Englewood Cliffs, NJ: Prentice Hall International, 1991.

[25] D. E. Knuth, "Efficient balanced codes," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 51–53, Jan. 1986.

[26] L. J. Greenstein, "Spectrum of a binary signal block coded for DC suppression," *Bell Syst. Tech. J.*, vol. 53, pp. 1103–1126, July-Aug. 1974.

[27] A. Kunisa, S. Takahasi, and N. Itoh, "Digital modulation method for recordable digital video disc," *IEEE Trans. on Consumer Electron.*, vol. 42, no. 3, pp. 820–825, Aug. 1996.

[28] I. J. Fair, Q. Wang, and V. K. Bhargava, "Polynomials for guided scrambling line codes," *IEEE J. Select. Areas Commun.*, vol. 13, no. 3, pp. 499–509, Apr. 1995.

[29] I. J. Fair, Q. Wang, and V. K. Bhargava, "Characteristics of guided scrambling encoders and their coded sequences," *IEEE Trans. Inform. Theory*, vol. 43, no. 1, pp. 342–347, Jan. 1997.

[30] I. J. Fair, V. K. Bhargava, and Q. Wang, "Evaluation of the power spectral density of guided scrambling coded sequences," *IEE Proc.-Commun.*, vol. 144, 70–78, Apr. 1997.

[31] Y. Xin and I. J. Fair, "Polynomials for generating selection sets with complementary quotients in guided scrambling line coding," *Electron. Lett.*, vol. 37, pp. 365-366, Mar. 2001.

[32] Y. Xin and I. J. Fair, "Low-frequency performance of guided scrambling dc-free codes," to appear in *IEEE Commun. Letters.*

[33] B. P. Lathi, *Modern Digital and Analog Communication Systems*, 3rd ed., New York, NY: Oxford University Press, 1998, pp. 487–525.

[34] J. G. Proakis, *Digital Communications.* 3rd ed., Boston, MA: Mc-Graw-Hill, 1995.

[35] W. R. Bennett, "Statistics of regenerative digital transmission," *Bell Syst. Tech. J.*, vol. 37, pp. 1501–1521, Nov. 1958.

[36] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications.* 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1996.

[37] G. L. Cariolaro and G. P. Tronca, "Spectra of block coded digital signals," *IEEE Trans. Commun.*, vol. 22, pp. 1555–1564, Oct. 1974.

[38] G. L. Cariolaro, G. L. Pierobon, and G. P. Tronca, "Analysis of codes and spectra calculations," *Int. J. Electron*, vol. 55, pp. 35–79, July 1983.

[39] M. S. Bartlett, "Smoothing periodograms from time series with continuous spectra," *Nature* (London), vol. 161, pp. 686–687, May 1948.

[40] P. D. Welch, "The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short modified periodograms," *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 70–73, June 1967.

[41] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 623–656, Oct., 1948.

[42] E. Prange, "Cyclic error-correcting codes in two symbols," Air Force Cambridge Research Center, Cambridge, MA, Tech. Rep. AFCRC-TN-57-103, Sept. 1957.

[43] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, pp. 147–160, Apr. 1950.

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

[45] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, pp. 68–79, Mar. 1960.

[46] A. Hocquenghem, "Codes corecteurs d'erreurs," *Chiffres* (Paris), vol. 2, pp. 147–156, Sept. 1959.

[47] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM J. Appl. Math.*, vol. 8, pp.300–304, June 1960.

[48] D. E. Muller, "Application of Boolean algebra to switching circuit design," *IEEE Trans Comput.*, vol. 3, pp. 6–12, Sept. 1954.

[49] I. S. Reed, "A class of multiple-error-correcting codes and a decoding scheme," *IEEE Trans Inform. Theory*, vol. 4, pp. 38–49, Sept. 1954.

[50] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice Hall, 1995.

[51] P. Elias, "Coding for noise channels," *IRE Conv. Record*, Part 4, pp. 37–47, 1955.

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

[53] 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. 248–287, Mar. 1974.

[54] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo-codes (1)," in *Proc. 1993 IEEE Int. Conf. Communications*, 1993, pp. 1064–1070.

[55] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," Jet Propulsion Laboratory, Pasadena, CA, JPL TDA Progress Rep. 42-126, pp. 1–26, Aug. 1996.

[56] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. 1995 IEEE Int. Conf. Communications*, 1995, pp. 1009–1013.

[57] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European Trans. Telecommunications*, vol. 9, pp.155–172, Mar.–Apr., 1998.

[58] "Physical layer standard for cdma2000 spread spectrum systems," 3rd Generation Partnership Project 2, 3GPP2 C.S.0002-C, v1.0, May 2002.

[59] "Multiplexing and channel coding (FDD)," 3rd Generation Partnership Project; Technical Specification Group Radio Access Network, 3GPP TS 25.212. v5.4.0, Mar. 2003.

[60] F. Chiaraluce, E. Gambi, R. Garello, P. Pierleoni, G. P. Calzolari, and E. Vassallo, "On the new CCSDS standard for space telemetry: turbo codes and symbol synchronization," in *Proc. IEEE Int. Conf. Communications*, 2000, pp. 451–454.

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

[62] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.

[63] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 33, pp. 457–458, Mar. 1997.

[64] N. Wiberg, "Codes and decoding on general graph," Ph. D. dissertation, no. 440, Dept. Elect. Eng., Linköping Univ., Linköping, Sweden, 1996.

[65] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 427, pp. 619–637, Feb. 2001.

[66] J. Pearl, *Probability Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[67] H. Futaki and T. Ohtsuki, "Low-density parity-check (LDPC) coded OFDM systems," in *Proc. IEEE Vehicular Tech. Conf.*, Fall, 2001, pp. 82–86.

[68] P. Elias, "Error-free coding," *IRE Trans. Inform. Theory*, vol. PGIT-4, pp. 29–37, Sept. 1954.

[69] G. D. Forney, Jr., *Concatenated Codes*. Cambridge, MA: MIT Press, 1966.

[70] J. Lodge, R. Young, P. Hoeher and J. Hagenauer, "Separable MAP "filters" for the decoding of product and concatenated codes," in *Proc. 1993 IEEE Int. Conf. Communications*, 1993, pp. 1740–1745.

[71] R. M. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes," in *Proc. 1994 IEEE GLOBECOM Conf.*, 1994, pp. 339–343.

[72] D. Chase, "A class of algorithm for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 170–182, Jan. 1972.

[73] R. J. McEliece and W. Lin, "The trellis complexity of convolutional codes," Jet Propulsion Laboratory, Pasadena, CA, JPL TDA Progress Rep. 42-123, pp. 122–138, Nov. 1995.

[74] R. E. Blahut, *Theory and Practice of Error Control Codes.* Reading, MA: Addison-Wesley Publishing Company, 1983.

[75] P. Sweeney, *Error Control Coding: From Theory to Practice.* Baffins Lane, Chichester, England: John Wiley & Sons, Ltd., 2002.

[76] D. J. C. MacKay, "Online code resources of low-density parity-check codes," available at http://www.inference.phy.cam.ac.uk/mackay/codes/data.html.

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

[78] B. Vasic and K. Pedagani, "Run-length-limited low-density parity check codes based on deliberate error insertion," *IEEE Trans. Magnetics*, vol. 40, pp. 1738–1742, May 2004.

[79] B. Sklar, *Digital Communications: Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice Hall, 1988.

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

List of Symbols and Abbreviations

The following symbols and abbreviations are used throughout this thesis. Other notation is introduced where required.

a	Attenuation factor.
A	Number of augmenting bits in GS dc-free coding.
A_c	Number of augmenting bits in dc-free block coset coding.
b_i	A variable in RSC encoder. It determines the states.
B	An arbitrary constant.
c	A codeword.
$\overline{c}_j$	Complement of c_j.
C	Capacitance.
C_n	Channel capacity for a noisy channel.
C_s	Capacity for a constrained channel.
$C_s(V)$	Capacity of a dc-free code given the digital sum variation V.
$C_s(\tilde{V})$	Capacity of a dc-free code given a moderately large digital sum variation $\tilde{V}$.
$d(x)$	Scrambling polynomial of GS coding.
$d_e(x)$	Even-weight scrambling polynomial of GS with degree not great than A.
$d_{2,A}(x)$	Weight-two scrambling polynomial of degree A for GS ($d_{2,A}(x)=x^A+1$).
$d_{\min}$	Minimum distance of a block code.
d_{free}	Minimum free distance of a convolutional code.
D	Degree of the scrambling polynomial $d(x)$.
$E[X(t)]$	Expectation of $X(t)$.
E_b / N_0	Ratio of bit energy to noise PSD of the white Gaussian noise.
E_s	Channel symbol energy.
E_S	Received normalized symbol energy ($T=1$) for the AWGN channel.
E_{SHM}	Received symbol energy for the HP-AWGN channel using a matched receiving filter.

E_{SHR}	Received symbol energy for the HP-AWGN channel using a rectangular receiving filter.
E_s / N_0	Ratio of channel symbol energy to noise power spectral density of the white Gaussian noise, $E_s / N_0 = R_C E_b / N_0$, where R_C is code rate.
F	Connection matrix between states.
f	Frequency (hertz).
f_T	Normalized frequency with symbol rate $1/T$, $f_T = fT$.
$f_X(x)$	Probability density function of X.
F^*	Set of 2^m elements over GF(2^m).
$\boldsymbol{G}$	Generator matrix of a linear block code.
$\boldsymbol{G}_n$	Nonsystematic generator matrix of a linear block code.
$\boldsymbol{G}_s$	Systematic generator matrix of a linear block code.
$g_{BCH}(x)$	Generator polynomial of a binary BCH code.
$g_C(x)$	Generator polynomial of a binary cyclic code.
$g_{RS}(x)$	Generator polynomial of an RS code .
$g(t)$	A pulse shape.
$G(\omega)$	Fourier transform of $g(t)$.
$G(g_1, g_2)$	Generators for a rate 1/2 convolutional encoder, where g_1 and g_2 denotes two feedforward connection patterns associated with the generation of parity bits. Generally g_1 and g_2 are octal numbers (e.g. $G(7,5)$). For an RSC encoder, g_1 denotes the feedback generator and g_2 denotes the feed-forward generator.
GF(2)	Galois field (finite field) with 2 elements.
GF(p)	Galois field with p elements where p is a prime.
GF(2^m)	Galois filed with 2^m elements. This field is generated by a primitive polynomial $p(x)$ of degree m over GF(2).
GF(p^m)	Galois field with p^m elements. This field is extended from GF(p).
GS_0	An encoder that maps a source word $s(x)$ to a GS word $q_0(x)$ by scrambling $s(x)$ with $d(x)$.
$\boldsymbol{H}$	Parity-check matrix of a linear block code.
$\boldsymbol{H}_s$	Systematic parity-check matrix of a linear block code.

$h_i(x)$	Predetermined additive patterns for GS encoding with A-bit augmenting, $i = 0,1,\ldots,2^A - 1$.
$H_{HP}(f_T)$	Frequency response of the HPF.
$h_r(t_T)$	Impulse response of a receiving filter.
$h_{rM}(t_T)$	Impulse response of a matched receiving filter.
$h_{rR}(t_T)$	Impulse response of a rectangular receiving filter.
$H(X)$	Entropy of the source symbol X.
$H_\infty(X)$	Entropy of the source when considering the statistical dependence between symbols in a source message.
$H(X,Y)$	Joint entropy of input X and output Y.
$H(Y \mid X)$	Conditional entropy.
$I(x_i)$	Amount of information carried by the symbol x_i.
$I(X;Y)$	Mutual information between X and Y through the channel.
K	Constraint length of convolutional codes.
k	Length of an input word in a linear block code.
k_b	Length of a binary input word.
k_{GS}	Input word length of GS encoding.
k'	Number of input bits of convolutional codes in a time interval.
l	Discrete time.
L_A	A factor for the LFSW estimation of GS dc-free codes.
M	Number of States.
$m_X(t)$	Mean of $X(t)$.
n_b	Length of a binary codeword.
n	Length of a codeword. When the code is binary, $n_b = n$, and when the code is nonbinary over a Galois field GF(2^m), $n_b = m \cdot n$.
n_{GS}	Codeword length of GS encoding.
n_{ij}	Number of edges exiting state s_i and entering state s_j.
n_X	Number of discrete input symbols of a DMC channel.
n_Y	Number of discrete output symbols of a DMC channel.
n'	Number of encoder output digits of convolutional codes in a time interval.
$n(t)$	Additive random noise process.

$N(l)$	The number of acceptable sequences of length l for a constrained channel.
p	A prime.
$p(x)$	A primitive polynomial of degree m.
P	Average power of input signals of the channel.
$P(x_i)$	Probability of symbol x_i.
$P(y_j \mid x_i)$	Transition probability of a DMC.
$P(x_i, y_j)$	Joint probability.
$P_b(e)$	Probability of decoded bit error.
Q	A quotient selection set.
$q_0(x)$	A GS codeword obtained by scrambling a source word with the GS scrambling polynomial.
$\boldsymbol{q}_i$	k-bit input word of EC codes.
$\boldsymbol{r}$	A received word.
$r(t)$	Received signal.
R	Resistance.
R_C	Code rate.
R_{GS}	Code rate of GS codes.
R_{EC}	Code rate of EC codes.
R_s	Average information rate of the source.
$R_X(t, t+\Delta t)$	Autocorrelation of $X(t)$.
$\boldsymbol{s}$	Syndrome which indicates whether the received word is a codeword.
s_i	ith state.
S_A	A factor for the sum variance estimation of dc-free GS codes.
S_l	Running digital sum of a coded sequence at the lth time instant.
$s(t)$	Transmitted signal.
$S_{xx}(f_T)$	PSD of the random signal $x(t)$.
SNR_{oM}	Output SNR for the matched-filter receiver.
SNR_{oR}	Output SNR for the rectangular filter receiver.
t	Time or error-correction capability.
T	Symbol/pulse duration or transpose of a matrix.
t_T	Normalized time with symbol duration T, $t_T = t/T$.

T_s	Time duration of a source symbol.
$\boldsymbol{u}$	A input word.
V	Digital sum variation, which indicates the excursion of RDS values in the coded sequence.
$V_{\max}$	Maximum RDS.
$V_{\min}$	Minimum RDS.
$V_{e,l}$	Output voltage of the HPF at the end of the lth interval.
$V_{s,l}$	Output voltage of the HPF at the start of the lth interval.
$V(t)$	Output voltage of the HPF.
w_c	Column weight of the parity check matrix.
w_r	Row weight of the parity check matrix.
w_d	Weight of GS polynomial.
$\{x_j\}$	A coded sequence or a WSS discrete random process, where $x_j \in \{-1,1\}$.
$x(t)$	Random input signal of the HPF.
$X(t)$	PAM signal.
$y(t)$	Random output signal of the HPF.
z	Number of shortened bits.
α	A nonzero element of a finite field GF(2^m).
β	An element of a finite field GF(2^m).
β^{2^i}	A conjugate of β over GF(2^m).
$\lambda_{\max}$	The greatest eigenvalue of the connection matrix.
v	Memory of convolutional codes.
ξ	Low-frequency spectrum-weight.
ξ_m	LFSW of a maxentropic dc-free sequence.
ξ_{GS}	LFSW of a GS dc-free sequence.
ξ_{HP}	LFSW of the HPF.
σ^2	Average noise power of AWGN.
σ_s^2	Variance of running digital sum (sum variance in brief) of a coded sequence.
$\sigma_{s,GS}^2$	Sum variance of a dc-free GS coded sequence.
$\sigma_{s,m}^2$	Sum variance of a maxentropic dc-free coded sequence.

τ_T	Normalized time constant by the symbol duration T, $\tau_T = RC/T$.
$\phi(x)$	Minimal polynomial of β over GF(2^m).
$\Phi_x(\omega_T)$	PSD of the sequence $\{x_j\}$ (note that the frequency is normalized).
$\Psi(\omega_T)$	Magnitude squared of the frequency response of the HPF, $\Psi(\omega_T) = \lvert H_{HP}(\omega_T)\rvert^2$.
Π	Random interleaver.
ω	Radian frequency ($\omega = 2\pi f$).
ω_T	Normalized radian frequency with symbol rate $1/T$, $\omega_T = \omega T$.
$\omega_{T,0}$	Normalized cutoff frequency at which the power spectral density of a coded sequence equals 0.5 or –3 dB.

APP	*A Posteriori* Probability The probability $\Pr(u_k = i \mid Y_1^N)$ is an APP for u_k having the value of element i after Y_1^N is received, where Y_1^N is a demodulated received word with word length of N, and u_k is the k-th symbol in the input word.
A-SISO	Additive soft-input soft-output (A-SISO) algorithm, which works in the logarithmic domain and has equivalent performance to the SISO algorithm.
AWGN	Additive White Gaussian Noise An uncorrelated Gaussian (normally distributed) noise process that is independent of the transmitted signals, corrupts the signal through addition, and has flat PSD for all frequencies. The adjective "white" is used in the sense that white light contains equal amounts of all frequencies within the visible band of electromagnetic radiation.
BCH	Bose, Chaudhuri, and Hocquenghem (three researchers who discovered BCH codes)
BER	Bit Error Rate BER is the probability that a message bit is incorrect. The number of erroneous bits is averaged over the entire number of bits transmitted.
BPSK	Binary Phase Shift Keying BPSK is a binary modulation format in which the data is modulated onto the carrier by varying the phase of the carrier by $\pm\pi$ radians.
BSC	Binary symmetric channel.

CDMA	Code division multiple access. CDMA is a spread spectrum technique. Through this technique, multiple users occupies the same band simultaneously by having different codes. This leads to universal frequency reuse.
CDMA2000	CDMA2000 is one of the standards for third generation (3G) wireless communications systems. It was developed by the Third Generation Partnership Project 2 (3GPP2), a partnership consisting of five telecommunications standards bodies: ARIB and TTC in Japan, CWTS in China, TTA in Korea, and TIA in North America.
CCSDS	Consultative Committee for Space Data Systems.
CS	Constrained sequence.
DMC	Discrete memoryless channel.
E_b / N_0	The average energy per information bit over the single-sided power spectral density of the white Gaussian noise.
EC	Error control.
GS	Guided scrambling.
HPF	High-pass filter.
LDPC	Low-density parity-check.
LFSW	Low-frequency spectrum-weight.
Log-MAP	The Log-MAP algorithm, which works in the logarithmic domain and has equivalent performance to the MAP algorithm, is developed to reduce the computational complexity of the MAP algorithm.
Log-SPA	Sum product algorithm in the logarithmic domain.
MAP	Maximum *a posteriori* The MAP algorithm leads to the selection of i that maximizes the probability $\Pr(i \mid Y)$ for some received Y. It is also called minimum error algorithm, since on average this algorithm yields the minimum number of incorrect decisions. Note that this algorithm is optimum only when all types of errors are equally harmful or costly. When some of the error types are more costly than others, a algorithm that incorporates relative cost of errors should be employed.
MRDS	Minimum word-end running digital sum selection criterion, which selects the word from the selection set and results in the minimum absolute word-end RDS value.
MSW	Minimum squared weight selection criterion, which selects the word with minimum sum of the squared RDS values at each bit position within the word.
PAM	Pulse amplitude modulation.

PCCCs	Parallel concatenated convolutional codes PCCCs are obtained through parallel concatenation of convolutional encoders with interleavers. In this thesis, PCCCs are referred to as turbo codes.
PSD	Power spectral density.
RDS	Running digital sum of a coded sequence.
RS	Reed and Solomon (two discoverers of RS codes).
SCCCs	Serially concatenated convolutional codes The SCCCs are generated through serial concatenation of convolutional encoders with interleavers.
SISO	Soft-input soft-output algorithm which performs an update of the *a posteriori* probabilities of both information and coded symbols based on the code constraint.
SNR	Signal-to-Noise Ratio SNR is a ratio of average signal power to average noise power.
SPA	Sum product algorithm.
SR	Spread Random The SR interleaver is based on the random generation of N integers from 1 to N with a minimum constraint on the amount of spreading between adjacent positions.
VA	Viterbi algorithm.
WCDMA	Wideband code division multiple access, which is one of the standards for third generation (3G) wireless communications systems. It is based on radio access technique proposed by ETSI (European Telecommunications Standards Institute) Alpha group and the specifications was finalized in 1999.
WSS	Wide-sense stationary.