The Design of Binary Shaping Filter of Binary Code

by

Shao-Lun Huang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author ....
Department of Electrical Engineering and Computer Science
May 21, 2010

Certified by...
Lizhong Zheng
Associate Professor
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Theses

The Design of Binary Shaping Filter of Binary Code

by

Shao-Lun Huang

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

In information theory, in order to maximize the total throughput, it is required that the codebook has an empirical distribution that maximizes the mutual information over the channel. In coding theory, however, most codes we can generate have Bernoulli $(\frac{1}{2})$ distribution. In this thesis, we present a new coding scheme to efficiently generate binary codes with different distributions. Our main approach is to first encode the information bits by a linear code $\mathcal{C}$, and then quantized the codeword to the closest codeword in another linear code $\mathcal{C}_s$. The quantization error is then treated as the encoded codeword in our coding scheme. We discuss the practicality and advantage of such codes and give a design example.

Thesis Supervisor: Lizhong Zheng
Title: Associate Professor

Acknowledgments

I would first and foremost like to thank my family. My parents, Chih-Fang Huang and Shu-Mei Hsu, my sister, Shao-Yi Huang.

I would next like to thank my advisors, Professor Lizhong Zheng, for their invaluable guidance. I am very grateful for the opportunity to learn from him, in so many rich and diverse ways.

Contents

List of Figures

Chapter 1

Introduction

In information theory [2], it has been shown that for a given channel, the optimal channel codes should have the empirical distribution matching to the one that maximizes the mutual information. For example, the capacity achieving input distribution of BSC (binary symmetric channel) is Bernoulli($\frac{1}{2}$), and the practical codes used in such channels are linear binary codes, with roughly $\frac{1}{2}$ 0's and $\frac{1}{2}$ 1's in the codebook. We call such codes "balanced binary codes". However, for other channels, balanced binary code might not be optimal. In particular, in most network information theory results, it is often required that codes with specific distributions, which is unbalanced and far from Bernoulli($\frac{1}{2}$), to be used. However, practical codes including LDPC (low density parity check) codes, and PSK/QAM constellation for AWGN channel are based on Bernoulli $\frac{1}{2}$ distributed code. Therefore, we would like to develop an efficient scheme to generate channel codes with arbitrary distribution.

There are several ways to do this. First of all, the random coding in information theory can generate codes with arbitrary given distribution. However, random codes can not be used in practice since the encoding and decoding complexity grows exponentially with the codeword length, which is usually long in practice.

Another commonly used way to generate a codebook with a desired distribution is by employing a non-linear shaping filter. It is well known that if U $\sim$ Unif [0,1], and F is the cumulative distribution function (CDF) of the desired distribution, then $\mathrm{F}^{-1}(U)$ has this distribution. Using this fact, one can take a block of bits from a linear

binary code, and view that as a real value between $[0, 1]$, and apply the non-linear function F^{-1} to shape it into a desired distribution.

However, there are some practical issues of this approach. First of all, a block of n bits does not precisely correspond to a real valued random variable uniformly distributed in $[0, 1]$, but only take values on the grid of 2^{-n}. Thus if we apply the shaping to a short block, the output distribution is not shaped accurately. On the other hand, if we increase the block length n, the complexity increases too. Even more importantly, the capability of error correction of the resulting unbalanced code can sometimes degrade as n increases! To see that, one can think of the non-linear shaping map as an inner code of length n, and there is an outer code over multiple of such n-blocks. Thus the performance of such a code can be analyzed under the framework of concatenated code [1]. It turns out that the performance of such concatenated code can be significantly worse than the original code before shaping. A typical example of LDPC code with one thousand coded bits, broken into sub-blocks of length $n = 20$, and each individually shaped, can have significantly worse error performance. Intuitively, this happens since a small error in a particular block will make the detection of entire block of bits wrong, giving the outer code of the order $\frac{n}{2}$ incorrect bits. Now, in order to prevent from the performance degrading due to code concatenation, the block length n for the non-linear shaping should be increased to the same order of the block length of the outer code. Considering in our example above, the block length n should be in the order of one thousand bits. However, in this situation, the complexity of non-linear shaping map is extremely high. The goal of this thesis is to find an efficient way for high-dimensional shaping filter designs: one that can process an LDPC codeword in its entirety, and hence utilize the good distance property of the existing linear codes.

The key idea of the coding scheme that we would like to investigate is as follows. Firstly, the information bits are encoded by a linear encoder $\mathcal{C}_1$,such as LDPC encoder, and then use the decoder of another linear code as the non-linear shaping operator. This is illustrated in Figure 1-1. In the first stage, the linear code $\mathcal{C}_1$ is used to encode the information bits $\underline{d}$ into the codeword $\underline{u} \in \mathcal{C}_1$. In the second stage, the codeword $\underline{u}$

of $\mathcal{C}_1$ is quantized to the closest vector $\underline{\hat{u}}$ in another linear codebook $\mathcal{C}_2$. In particular, we have

$$\underline{\hat{u}} = \arg\min_{\underline{v}\in\mathcal{C}_2} |\underline{u} - \underline{v}|. \tag{1.1}$$

where $|\cdot|$ denotes the Hamming weight, and $-$ is the binary difference (XOR).

Finally, the quantization error $\underline{x} = \underline{\hat{u}} - \underline{u}$ is taken as the encoded codeword for the information bits $\underline{d}$. Equivalently, we can see from (1.1) that the quantized vector $\underline{\hat{u}}$ is just the decoded codeword, which passes $\underline{u}$ through the decoder of $\mathcal{C}_2$. Therefore, this quantization process can be implemented as decoding $\underline{u}$ by $\mathcal{C}_2$, and the quantization error $\underline{x}$ is the decoding error. Particularly, note that $\underline{x}$ can be viewed as shaping the codeword $\underline{u} \in \mathcal{C}_1$ non-linearly by the decoding process of $\mathcal{C}_2$.

There is a geometric way to interpret this encoding scheme. In Figure 1-2, suppose that the large outer ball is $\mathbb{F}_2^n$, the vector space of binary codewords with length n. Consider every codeword $\underline{\hat{u}} \in \mathcal{C}_2$ and the corresponding maximal likelihood decoding region $\mathcal{D}(\underline{\hat{u}})$,

$$\mathcal{D}(\underline{\hat{u}}) \triangleq \{\underline{v} \in \mathbb{F}_2^n : \forall \underline{u}' \in \mathcal{C}_2, |\underline{v} - \underline{\hat{u}}| \leq |\underline{v} - \underline{u}'|\}. \tag{1.2}$$

Then the decoding regions $\mathcal{D}(\underline{\hat{u}})$ are balls packed into $\mathbb{F}_2^n$. Let us define the translating map $\mathcal{T}$ as that for all $\underline{\hat{u}} \in \mathcal{C}_2$,

$$\mathcal{T} : \underline{v} \in \mathcal{D}(\underline{\hat{u}}) \longrightarrow \underline{v} - \underline{\hat{u}} \in \mathcal{D}(\underline{0}), \tag{1.3}$$

Now one can see that by controlling the rate of $\mathcal{C}_2$, we can control the distribution of the output of this process. The main point here is that when processing very high dimensional binary strings, the above non-linear processing can be efficiently implemented by using the decoder of $\mathcal{C}_2$, which is assumed to be readily available.

We argue that under certain conditions, the resulting code map $\mathcal{T} \cdot \mathcal{C}_1 : \underline{d} \to \underline{x}$ is a good code, in the sense that the image, i.e., the collection of codewords are evenly and sparsely distributed in $\mathcal{D}(0)$, and thus have the desired distribution as well as the capability of error correction. Intuitively, this nice property is ensured

since the codewords of $\mathcal{C}_1$ are themselves evenly and sparsely distributed, thus after some shifting, the desired "random-like" behavior remains.

In remaining of this thesis, we will explain in details how to implement the above intuition in practice. We will give an example and demonstrate its performance with numerical simulations.

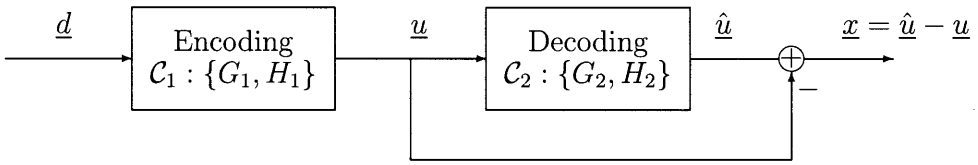

Figure 1-1: The encoding scheme. The information bits $\underline{\mathrm{d}}$ is first encoded to $\underline{\mathrm{u}}$ by a linear code $\mathcal{C}_1$, and then quantized to $\underline{\hat{u}}$ by the decoder of another linear code $\mathcal{C}_2$. Then the quantization error $\underline{x}$ is our codeword.

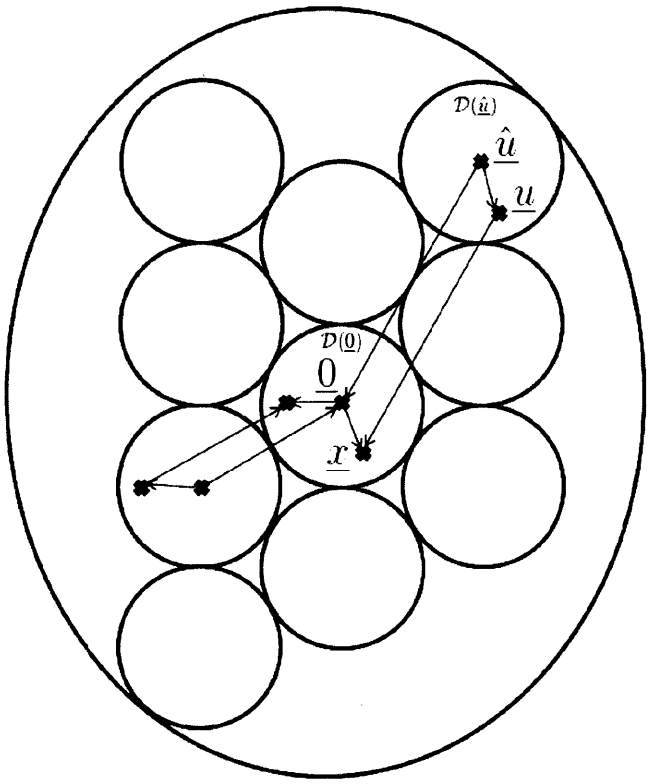

Figure 1-2: The large outer ball is the n dimensional binary vector space $\mathbb{F}_2^n$, the small balls are the decoding regions with respect to the corresponding codewords in $\mathcal{C}_2$. Suppose that the codeword $\underline{u}$ in $\mathcal{C}_1$ is in the decoding region $\mathcal{D}(\underline{\hat{u}})$ of $\underline{\hat{u}} \in \mathcal{C}_2$. After the parallel translating map $\mathcal{T}$, $\underline{\hat{u}}$ and $\underline{u}$ map to $\underline{0}$ and $\underline{x}$ respectively. Then the quantization error corresponding to $\underline{\hat{u}}$ and $\underline{u}$ is $\underline{x}$.

Chapter 2

The Encoding and Decoding Structure

In the following of this paper, we assume that all the channel models are binary and the information bits are independent and identity distributed (i.i.d.) Bernoulli($\frac{1}{2}$). Our goal is to generate codewords with probability distribution of Bernoulli(p) for some $p < \frac{1}{2}$.

2.1 The Encoder and Decoder

The information bits $\underline{d}$ is assumed to be a k vector with i.i.d. Bernoulli($\frac{1}{2}$) elements. Suppose that we pick two linear codes $\mathcal{C}_1, \mathcal{C}_2$ with generator matrices G_1, G_2 and parity matrices H_1, H_2 respectively. Assume that G_1 and G_2 are $n \times k$ and $n \times (n - n_2)$ matrices respectively, then the corresponding parity check matrices H_1 and H_2 are $(n - k) \times n$ and $n_2 \times n$ matrices. In order to encode $\underline{d}$, we first encode $\underline{d}$ by linear code $\mathcal{C}_1$. Let the n vector $\underline{u}$ be the encoded codeword of $\underline{d}$ by codebook $\mathcal{C}_1$, that is,

$$\underline{u} = G_1 \underline{d}. \tag{2.1}$$

Note that in many cases, we can obtain $G_1\underline{d}$ in an efficient way rather than multiplying a matrix to a vector. The second step is to quantize $\underline{u}$ to the closest vector

$\hat{\underline{u}}$ to $\underline{u}$, where $\hat{\underline{u}}$ is a codeword of another linear code $\mathcal{C}_2$. In particular, $\hat{\underline{u}}$ can be equivalently defined as (1.1). The quantization error

$$\underline{x} = \hat{\underline{u}} - \underline{u}, \tag{2.2}$$

is output codeword for $\underline{d}$ through our encoding scheme. Note that quantizing $\underline{u}$ to $\hat{\underline{u}}$ is equivalent to pass $\underline{u}$ through the decoder of $\mathcal{C}_2$. Therefore, this quantization procedure can be implemented as decoding $\underline{u}$ by $\mathcal{C}_2$, and the quantization error $\underline{x}$ is just what the decoder thinks the noise is. The block diagram of the encoder is shown in Figure 1-1, and note that since $\underline{x}$ is an n vector, the rate of this channel code is $r = \frac{k}{n}$.

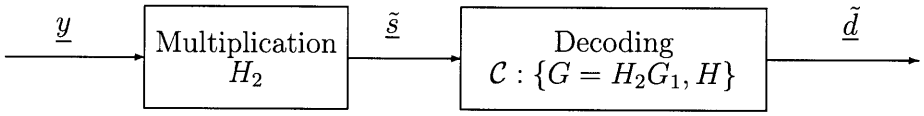

Figure 2-1: The decoder is first multiplying the received codeword $\underline{y}$ by the parity check matrix H_2 of $\mathcal{C}_2$, and then decoding by the linear code $\mathcal{C}$ with generator matrix $G = H_2G_1$.

On the decoder side, we receive $\underline{y}$, which is the noisy version of $\underline{x}$. To decode $\underline{d}$ from $\underline{y}$, a common way is to reverse the procedure in the encoder, namely, first map $\underline{y}$ back to an estimate $\tilde{\underline{u}}$ of $\underline{u}$, and then decode $\tilde{\underline{u}}$ to $\tilde{\underline{d}}$ by linear code $\mathcal{C}_1$. However, since the mapping from $\underline{u}$ to $\underline{x}$ is not bijective, see (1.3).

To solve this problem, we first observe that

$$H_2\underline{x} = H_2\underline{u} = \underline{s}, \tag{2.3}$$

is the syndrome of $\underline{u}$ to the linear code $\mathcal{C}_2$. Now we try to recover $\underline{s}$, instead of $\underline{u}$, from the noisy received signal $\underline{y}$.

Combining (2.1) and (2.3), we have the following,

$$\underline{s} = H_2\underline{x} = H_2G_1\underline{d}. \tag{2.4}$$

We define a new linear code $\mathcal{C}$ such that its generator matrix is

$$G \triangleq H_2G_1, \tag{2.5}$$

then the decoder can be described as follows. First we multiply $\underline{y}$ by H_2, which gives an estimate $\tilde{\underline{s}}$ of the syndrome $\underline{s}$, and then decode $\tilde{\underline{s}}$ by the codebook $\mathcal{C}$ to recover the data. Intuitively, this approach can correct the channel noise since the noise only causes a low weight difference between $\tilde{\underline{s}}$ and $\underline{s}$, which is in turn corrected by the linear code $\mathcal{C}$. In the following, we will refer this new code $\mathcal{C}$ as the *base code*. The decoding structure is shown in Figure 2-1.

Note that in order to guarantee that the linear code $\mathcal{C}$ is a well defined channel code, since its generating matrix G is an $n_2 \times k$ matrix, we have the constraint

$$n_2 > k. \tag{2.6}$$

Assuming that G_1 and H_2 are full rank, it is easy to check that the encoder in Figure 1-1 is a well defined channel encoder.

2.2 Sphere Packing

Figure 1-2 gives an intuitive interpretation of the proposed coding scheme. Here, one can easily visualize the codewords of $\mathcal{C}_1$ as points evenly spread over the space $\mathbb{F}_2^n$. These points are then shifted to $\mathcal{D}(0)$ by the map $\mathcal{T} : \underline{u} \to \underline{x} = \underline{u} - \hat{\underline{u}}$. The result should be points evenly spread over $\mathcal{D}(0)$. Mathematically, if the codewords in $\mathcal{C}_1$ are independently and uniformly chosen from $\mathbb{F}_2^n$, then after the shifting, their images are also uniformly distributed in $\mathcal{D}(0)$.

In our designs, we use the above insights drawn from random coding as a bench-

mark of the performance. In particular, we can derive the distances between codewords from a sphere packing argument. In a regular balanced binary code, with rate $r = k/n$, the decoding region of the 2^{nr} codewords are packed in $\mathbb{F}_2^n$, with volume 2^n. Hence, each decoding region has size of the order $2^{n-k} = 2^{n(1-r)}$, corresponding to a sphere with radius

$$2^{nH_b(d/2)} \approx 2^{n(1-r)} \Leftrightarrow d = 2 \cdot H_b^{-1}(1-r)$$

where $H_b(\cdot)$ is the binary entropy function in bits, and d is the typical minimum distance between codewords.

For unbalanced code with the same length n and rate $r = k/n$, however, we should not expect the distance between codewords to be as large as the corresponding balanced code, since the space where the codewords are packed is reduced from $\mathbb{F}_2^n$ to $\mathcal{D}(0)$. With a similar calculation, we should expect that the typical minimum distance between codewords to be

$$d = 2 \cdot H_b^{-1}(H_b(p) - r)$$

where $p < 1/2$ is the distribution of the codebook. Clearly, it is necessary that we choose a code rate $r < H_b(p)$. Moreover, the design goal is to make sure that all the codewords are evenly spread out. Mathematically, this means that the minimum distance between any pair of codewords is close to the typical minimum distance computed above from the sphere packing argument.

It is worth noticing that since the code we generate here is obviously not a linear code, we no longer have the nice property of linear codes that the nearest neighbor from any codeword always have the same distance. Thus, in general, we need to check the nearest neighbors from each codeword to verify that the above design goal is met. Moreover, to assure the error performance of our design at any noise level, one need to check the the distance spectrum, i.e., the number of neighbors at any given distance, from each codeword.

2.3 The Modification of The Encoder

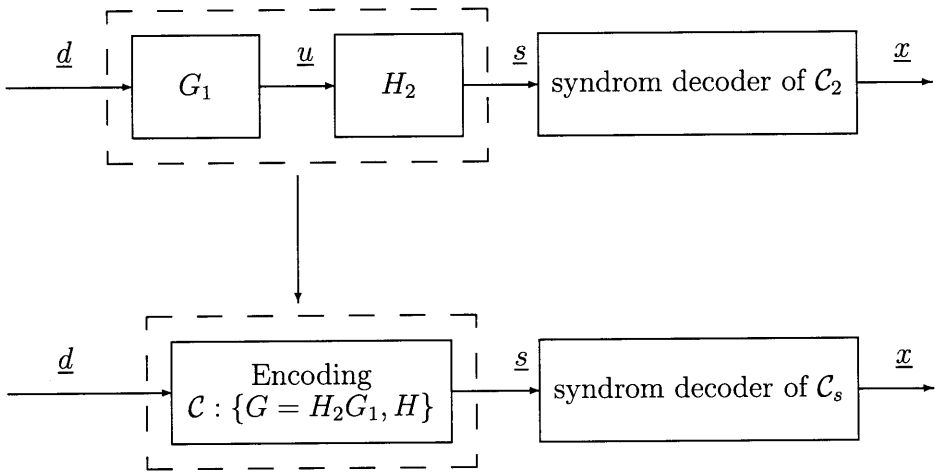

Figure 2-2: The decoder of $\mathcal{C}_s$ can be split out to the multiplication of the parity check matrix H_s and the syndrome decoder of $\mathcal{C}_s$. Thus the multiplication of G_1 and H_s can be combined to be the encoding process of the linear code $\mathcal{C}$ to generate the modified encoder.

Figure 2-2 shows a slight variation of the coding scheme. In the original structure in Figure 1-1, the output of $\mathcal{C}_1$, $\underline{u}$ is fed to the decoder of $\mathcal{C}_2$. Here, we make the assumption that this decoder first compute the syndrome $\underline{s} = H_2\underline{u}$, and then apply syndrome decoding to find the syndrome leader:

$$\arg \min_{\underline{x}: H_2\underline{x}=\underline{s}} |\underline{x}|$$

As discussed in the previous section, the performance of the overall unbalanced code relies on the sparseness of the $\underline{s}$ strings, instead of that of the $\underline{u}$ strings. Thus, it makes sense to directly chose the base code $\mathcal{C}$, as a good linear code with good distance properties, without explicitly worrying about whether the component codes $\mathcal{C}_1, \mathcal{C}_2$ are good error correction codes themselves. Furthermore, we choose $\mathcal{C}_2$ for the

syndrome decoder separately from $\mathcal{C}$, without requiring any relation between $\mathcal{C}_2$ and $\mathcal{C}$. The advantage of this separation is clear: we can focus on finding the base code $\mathcal{C}$ with good distance properties, so that the overall unbalanced code has a good error correction capability; and separately, we will focus on finding $\mathcal{C}_2$, which we refer to as the *shaping code* $\mathcal{C}_s$ from now on, to control the codeword composition with reduced complexity. In following section, we will discuss in details the design criteria of these two codes and give specific examples of these codes.

Chapter 3

The Designs of Shaping Codes

Our goal in this section is to discuss the design criteria of the base codes $\mathcal{C}$ and shaping code $\mathcal{C}_s$, such that for the entire coding scheme, the encoding and decoding complexity are practically implementable, and the error correcting capability of the code is good. We will also give an example and show its performance with numerical simulations.

3.1 The Design in Achieving Low Encoding and Decoding Complexity and Good Error Correcting Capability

Figure 2-2 shows that the encoder is composed by the encoder of the base code $\mathcal{C}$ and the syndrome decoder of the shaping code $\mathcal{C}_s$, and Figure 2-1 shows that the decoder is composed by the multiplication of the parity check matrix of the shaping code to a vector and the decoder of the base code $\mathcal{C}$. Thus in order to achieve low encoding and decoding complexity, we have to design the base code $\mathcal{C}$ with efficient encoding and decoding algorithms, and the shaping $\mathcal{C}_s$ with efficient syndrome decoding algorithm.

In appendix A, we show that LDPC codes can be efficiently encoded and decoded, and have the syndrome decoding algorithm by belief propagation [3]. Hence LDPC codes are good choices in designing base code $\mathcal{C}$ and shaping code $\mathcal{C}_s$ with efficient

encoding and decoding algorithms in the entire coding scheme.

The error correcting capability of the composition of the base code and the shaping code can be intuitively shown in Figure 1-2. As discussed in the previous section, the base code $\mathcal{C}$ can be directly chosen as a good linear code with good distance property, which is usually achieved by well designed LDPC codes. With this design, the codewords of $\mathcal{C}$ are evenly spread over $\mathbb{F}_2^{n_2}$, thus the codewords of $\mathcal{C}_1$ will be evenly spread over $\mathbb{F}_2^n$ in Figure 1-2. As mentioned previously, if the codewords in $\mathcal{C}_1$ are independently and uniformly chosen from $\mathbb{F}_2^n$, then after the map $\mathcal{T}$, their images are also uniformly distributed in $\mathcal{D}(0)$. Thus if the decoding region $\mathcal{D}(0)$ of the shaping code is evenly spread in every dimension in $\mathbb{F}_2^n$, the resulted codeword of the entire coding scheme will have good distance property.

This intuition can be further supported by the following argument. Suppose that the block length of the codeword $\underline{s}$ is n, and $p_i = \mathsf{Pr}(\text{Among all possible } \underline{s}\text{'s}, s(i) = 1)$, then the mean distance between all possible $\underline{s}$'s is

$$\mu_s = 2\sum_{i=1}^{n} p_i(1-p_i). \tag{3.1}$$

Assume that $\mathcal{C}_r$ is the random code with the corresponding distribution Bernoulli$\left(\frac{\sum_{i=1}^n p_i}{n}\right)$, then the mean distance between codewords in $\mathcal{C}_r$ is

$$\mu_{rand} = 2n\left(\frac{\sum_{i=1}^n p_i}{n}\right)\left(1-\frac{\sum_{i=1}^n p_i}{n}\right). \tag{3.2}$$

Note that $f(x) = 2x(1-x)$ is a concave function in $[0,1]$, by Jensen's inequality, we have

$$\mu_{rand} \geq \mu_s, \tag{3.3}$$

and the equality holds if and only if all p_i's are equal. Observe that for large n, if we randomly choose two different $\underline{s}$'s, then with high probability, the distance between these two codewords will be close to μ_s. Therefore, in order to come up with good distance property, compared to the corresponding random code, for codewords $\underline{s}$'s, every bit of $\underline{s}$ has to have the same probability to be zero or one. Geometrically, this

implies that the codewords $\underline{s}$ in $\mathcal{D}(0)$ is evenly spread in every dimension in $\mathbb{F}_2^n$. This provides the proof of our geometric intuition.

It is worth noting that the shaping code $\mathcal{C}_s$ does not need to be a good linear code, in the sense that the minimum distance between codewords is irrelevant in our problem. Here, since the syndromes are chosen from the base code $\mathcal{C}$, we do not have to consider the weights of the syndrome leaders corresponding to all possible syndromes. Instead, only a small fraction of them, namely the codewords of $\mathcal{C}$ can be passed to the syndrome decoder. Thus, the only design requirement for $\mathcal{C}_s$ is the simplicity of implementation, which broadens the choices.

It is known that although random codes achieve the capacity, the encoding and decoding complexity are exponentially high. Therefore we use the LDPC code to achieve low encoding and decoding complexity, and we will use random codes as our benchmark in characterizing the error correcting capability (the distance distribution and bit error rate) of our designed code. Some tradeoffs in obtaining implementable complexity and the error correcting capability will be shown in the next section.

3.2 The Design of The Shaping Code and Simulation

Convolutional codes often have very efficient encoding and decoding algorithm. They are particularly good choices of shaping codes as they have flexible lengths. In this section, we will design the shaping code $\mathcal{C}_s$ as the rate $\frac{1}{2}$ (1,3) systematic convolution code as an example, and give the numerical simulation. The block diagram and the Tanner graph of the (1,3) systematic convolution code are shown in Fig. 3-1.

As mentioned earlier, we will use the random code with the same distribution and rate as a benchmark of the performance, and compare the distance spectrum. To compare these two choices of shaping code fairly, we choose the base code here as a random code with block length N and distribution Bernoulli($\frac{1}{2}$).

The syndrome decoder of the (1,3) systematic convolution code can be imple-

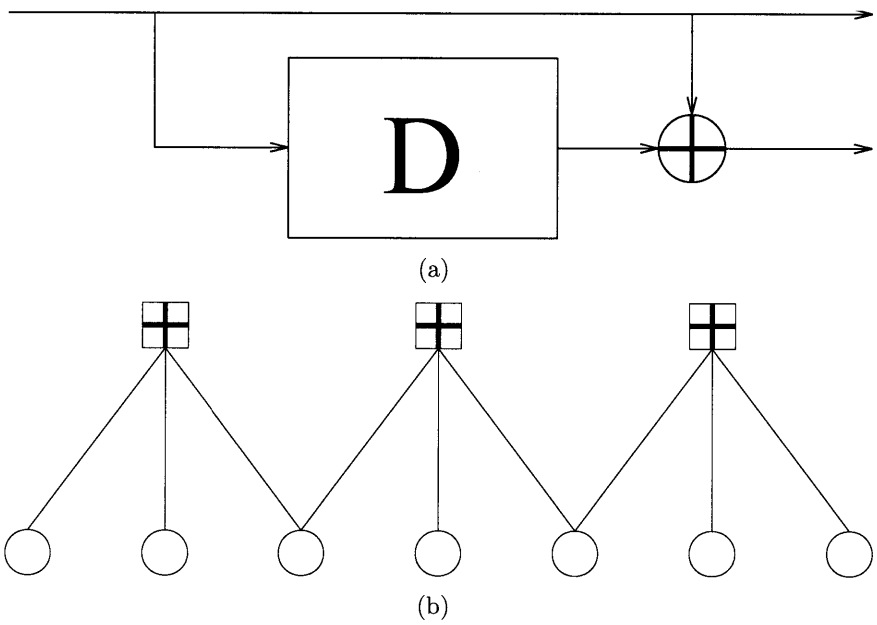

Figure 3-1: (a) The block diagram of the (1,3) systematic convolution code. (b) The corresponding Tanner graph of the convolutional code.

mented by the Viterbi Algorithm, however, there is an even simpler way to achieve this.

To see this, consider the Markov chain in Figure 3-2. Initially, at time 0, the first bit $s(0)$ of the syndrome is the initial state of the Markov chain. Suppose that after processing the $n-1$-th bit $s(n-1)$ in the syndrome at time $n-1$, the Markov chain is in some specific state, and the next bit $s(n)$ is received at time n, then $x(2n-1)$ and $x(2n)$ can be generated by following the Markov chain in Figure 3-2. For example, if at time n the Markov chain is in state 1, and $s(n)=0$, then we have $x(2n-1)=1$ and $x(2n)=0$. Finally, after N steps, we will obtain the codeword $\underline{x}$. It can be proven that the codeword $\underline{x}$ generated by this algorithm is the vector with the minimal Hamming weight in the same syndrome class as $\underline{s}$ of the (1,3) systematic convolution code, and we omit the prove here.

From the Markov chain in Figure 3-2, it is easy to show that the distribution of $\underline{x}$ is Bernoulli($\frac{1}{6}$), hence we can compare the distance spectrum of designing the shaping code as (1,3) systematic convolution code $\mathcal{C}_s$ and the random code with

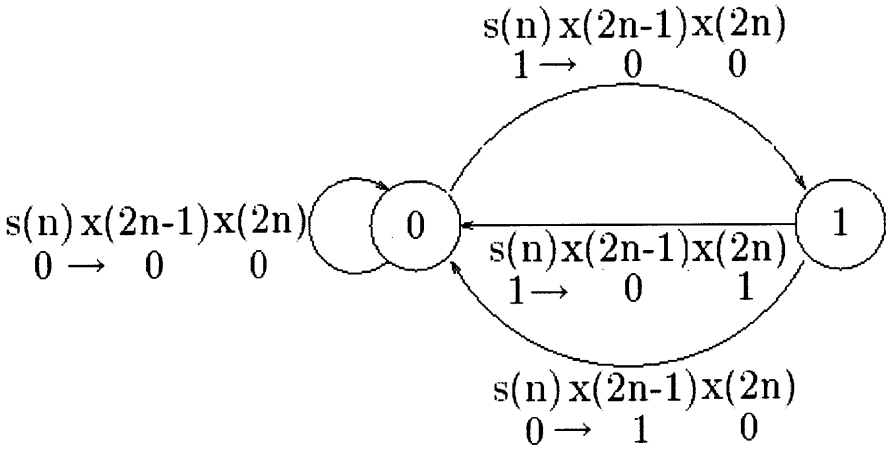

Figure 3-2: The Markov chain for encoding. Assume that at time n, the Markov chain is in some specific state, and receive s(n), then x(2n-1) and x(n) are obtained corresponding to this Markov chain.

rate $\frac{1}{2}$ and distribution Bernoulli($\frac{1}{6}$). Note that $\mathsf{Pr}(x(i) = 1) = \frac{1}{6}$ for all i, which satisfies the equality of (3.3), and hence satisfies the design criterion given in the previous subsection. The comparison is shown in Fig. 3-3, and the (1,3) systematic convolution code has better distance spectrum than the random code.

Practically, we can not design the base code as the random code, and should design it as a good LDPC code. For example, $\mathcal{C}$ can be designed as the widely used DVB-S.2 standard LDPC code [4] [5]. Suppose that the channel is BSC with crossover probability p. The numerical simulation of designing the base code as the DVB-S.2 standard LDPC code and the shaping code as (1,3) systematic convolution code with different crossover probabilities is shown in Figure 3-5.

Rather than implementing the multiplication of the parity check matrix of the shaping code to the received vector in the decoder in Figure 2-1 by directly multiplying them, we run the Viterbi algorithm on the Trellis graph in Figure 3-4. It is easy to observe that Figure 3-4 is the corresponding trellis graph to Figure 3-2, and hence running the Viterbi algorithm on the trellis graph is the maximal likelihood decoding. Figure 3-6 shows the bit error rate of these two decoding strategies with different crossover probabilities, where the bit error rate is measured before entering

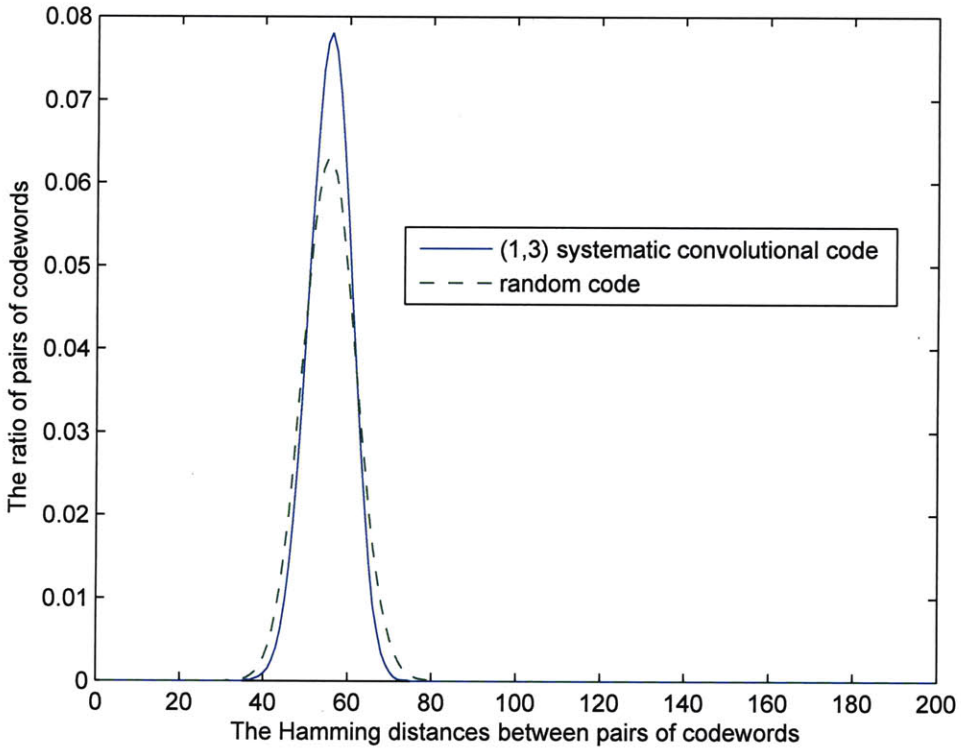

Figure 3-3: The distance spectrum of designing the shaping code $\mathcal{C}_s$ as the (1,3) systematic convolution code and the random code.

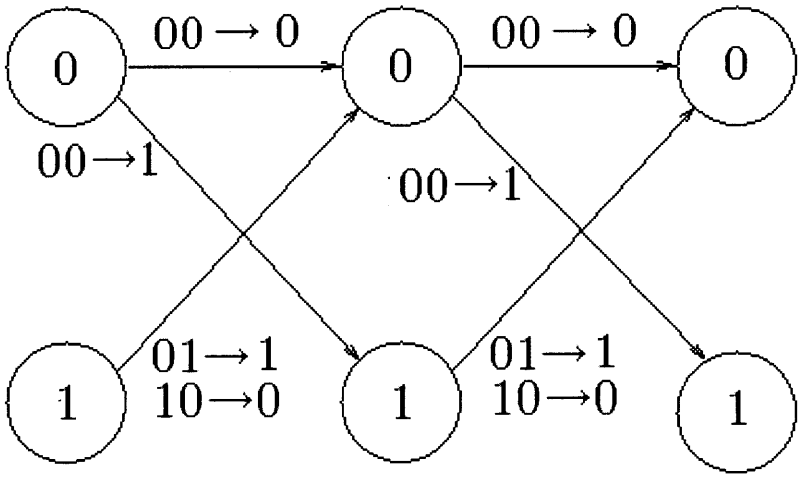

Figure 3-4: The trellis graph corresponding to the Markov chain in Fig. 3-2. Running Viterbi algorithm on this trellis graph gives the maximal likelihood decoding.

the decoder of the base code $\mathcal{C}$. We can see a significantly performance gain in using the Viterbi algorithm.

Since the rate of the DVB-S.2 standard LDPC code is $\frac{1}{2}$, we can compare our simulation in Figure 3-5 with a random code with rate $\frac{1}{4}$ and distribution Bernoulli($\frac{1}{6}$). It can be shown that for such a random code, if the crossover probability p is less than 0.135, the information can be communicated reliably. In Figure 3-5, our code can communicate information reliably at the crossover probability around 0.042. If we use the BPSK constellation and transfer the crossover probability to SNR by using the Gaussian Q-function, then the performance gap becomes to 1.9dB. This performance gap is mainly because that the base code (DVB-S.2 standard) is not capacity achieving, and we decode the shaping code and the base code separately.

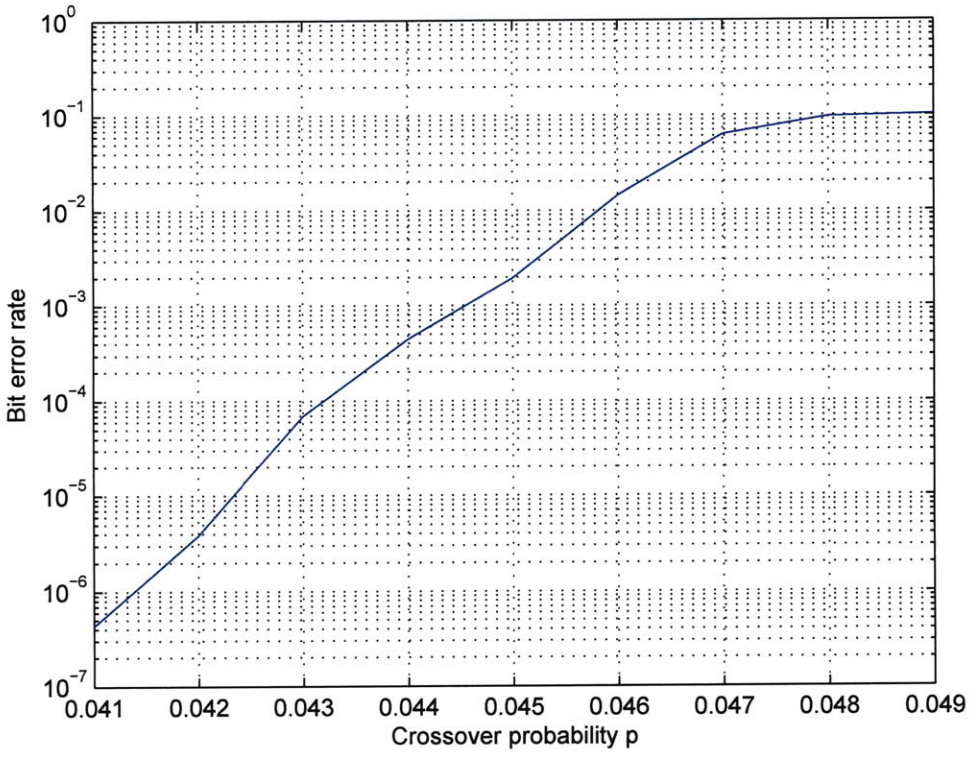

Figure 3-5: The bit error rate of cascading the DVB-S.2 standard LDPC code and the shaping (1,3) systematic convolution code for different crossover probabilities.

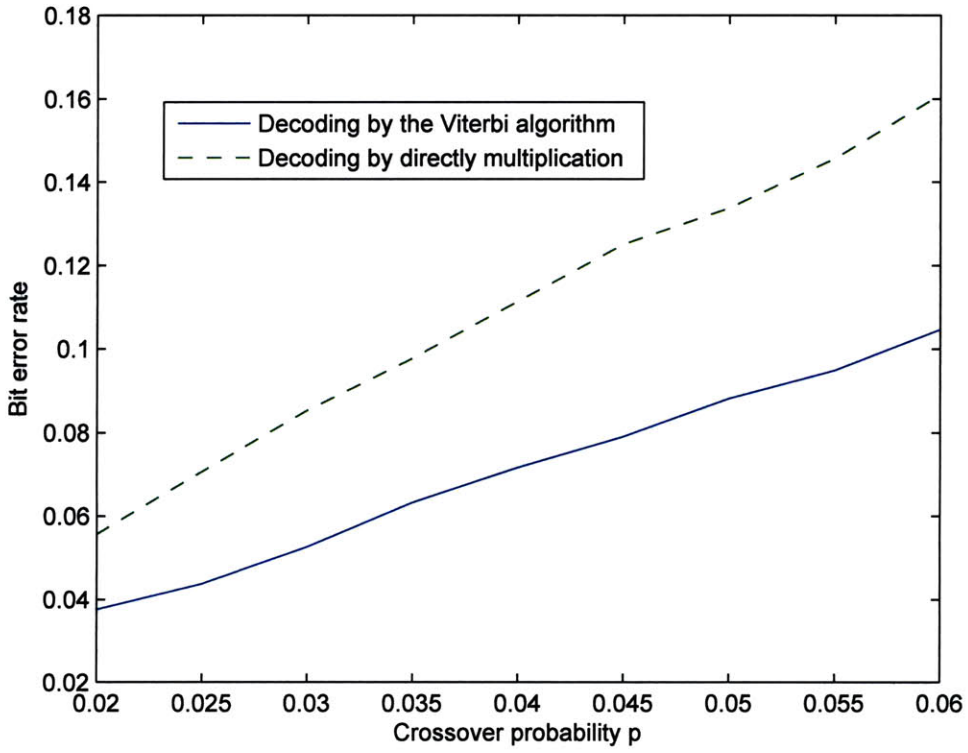

Figure 3-6: The bit error rate of using two different strategies : the Viterbi algorithm and directly multiplication. Note that the bit error rate here is measured before entering the decoder of the outer code $\mathcal{C}$.

Appendix A

Syndrome Decoding of LDPC Codes

We will present a syndrome decoding algorithm for LDPC code that is similar to belief propagation in this appendix. Suppose that we receive the syndrome $\underline{s} = H\underline{d}$ and want to find the syndrome decoding error $\underline{x}$. Firstly, recall that for the belief propagation algorithm on the LDPC code with parity check matrix H, after receiving the noisy codeword $\underline{y}$, the algorithm finds the vector $\underline{x}$, which is the closest vector to $\underline{y}$ that satisfies

$$H\underline{x} = \underline{0}. \tag{A.1}$$

Note that (A.1) is obtained by restricting the sum of each parity check node to be zero in every step in the algorithm. Now, if we substitute the restriction of the sum of the n-th parity check node to be $s(n)$, and remain everything else in the algorithm unchanged, then after finishing the belief propagation algorithm, we will obtain a vector $\underline{x}$, which is the closest vector to $\underline{y}$ that satisfies

$$H\underline{x} = \underline{s}. \tag{A.2}$$

Furthermore, if we choose $\underline{y} = \underline{0}$, then this revised belief propagation algorithm will produce a vector $\underline{x}$, which has smallest Hamming weight that satisfies (A.2).

Therefore, by using this revised belief propagation algorithm, we can obtain the syndrome decoding error $\underline{x}$ with the same complexity as the original belief propagation algorithm.

Bibliography

[1] G. D. Forney Jr., *Concatenated codes*, Massachusetts Inst. Technol., Cambridge, MA, 1966.

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

[3] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, Vol. 47, No. 2, pp. 498–519, Feb. 2001.

[4] European Telecommunications Standards Institude (ETSI), "Digital Video Broadcasting (DVB) Second generation framing structure for broadband satellite applications; EN 302 307 V1.1.1," www.dvb.org.

[5] A. Morello and V. Mignone, "DVB-S2: The Second Generation Standard for Satellite Broad-Band Services," *Proceedings of the IEEE*, vol. 94, pp. 210.227, 2006.