A New High Performance LDPC Code for DVB-S2

Massoud Khajeh

A Thesis

In

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science(Electrical Engineering) at
Concordia University
Montreal,Quebec, Canada

April 2006

Your file Votre référence
ISBN: 0-494-14264-2
Our file Notre référence
ISBN: 0-494-14264-2

ABSTRACT

A New High Performance LDPC Code for DVB-S2

Massoud Khajeh

In spite of their powerful error correcting capability, Low Density Parity Check codes (LDPC) have been ignored due to their high complexity. Around three decades after the invention of this code, researchers returned back their attention to it and tried to make some significant improvements in complexity. As the result, the LDPC codes were widely considered in the next generation error correcting codes in telecommunication systems. In 2005, the new standard for Digital Video Broadcasting (DVB-S2) used LDPC codes as its channel coding scheme. The features of the above mentioned code allow a transmission near the Shannon limit.

In this thesis, we first review the LDPC codes in general, and then we present the encoding and decoding scheme which is used in the DVB-S2 standard. We discuss regular and irregular LDPC codes and compare the advantages and disadvantages of these codes with each other.

In this thesis, we consider a higher block length for the LDPC code compared to DVB-S2 standard to improve the performance. We propose an efficient hybrid parity check matrix for this code. This parity check matrix has the same number of base addresses as the case for DVB-S2 which processes high block length with the same

complexity. At the end, simulation results are provided to show the improvement in the performance.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor Dr. Y. R. Shayan for his guidance, patience and insightful comments throughout this project. Without Dr Shayan help and advice, I would not have completed my thesis.

I would also like to thank my friends in VLSI and Wireless Laboratories. They helped me and encouraged me to finish this thesis. With their help, I overcome some technical and software problems. I wish that our friendship will continue beyond our professional lives.

I would like to give special thanks to my family. They support me throughout all the time that I've been studying and did not forget me even though they are living thousands of miles from me.

I am grateful to the department of Electrical and Computer Engineering in Concordia University for providing facilities for all students in their research program.

It is the time to appreciate all of the efforts, done by all of my teachers in my life, from kind elementary school teachers, to knowledgeable university professors. I respect their skills and sincerely thank all of them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

ACM	Adaptive Coded and Modulation
APP	A Posteriori Probability
APSK	Amplitude Phase Shift Keying
AWGN	Additive White Gaussian Noise
B	Number of Block
b	Generic Word
L	Block Length
BB	Base-Band
BCH	Bose,Chaudhuri, Hocquenghem Code
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
C	Channel Capacity
C	Codeword Matrix
CN	Check Node
D_b	Bit Node Degree in Block B
DVB-S	Digital Video Broadcasting Satellite
DVB-S2	Digital Video Broadcasting Satellite 2
d_c	Check Node Degree
d_v	Variable Node Degree
E_b/N_o	Bit Power over Noise Power

E_s	Average Symbol Energy
ECC	Error Correcting Codes
ETSI	European Telecommunication Standard Institute
f_N	Nyquist Frequency
f_o	Carrier Frequency
$f_{y\|x}$	Conditional Probability Density Function
FDM	Frequency Division Multiplexing
FEC	Forward Error Correcting
G	A Generator Matrix
H	Parity Check Matrix
$H^{(u)}$	Systematic Part of H
$H^{(p)}$	Parity Part of H
H^T	Transpose of Parity Check Matrix
I_k	A $k \times k$ Identity Matrix
IN	Information Node
IRA	Irregular Repeat Accumulate
k	Message Bits
L	Block Length
LDPC	Low Density Parity Check
LLR	Log Likelihood Ratio
m	Message Vector
MAP	Maximum A Posteriori

MLD	Maximum Likelihood Decoding
M_n	Set of Check Nodes for Bit Node b_n
MPEG	Moving Picture Expert Group
n	Coded Digit Length
$n-k$	Redundancy Bits (parity bits)
$N_o/2$	Two Sided Noise Power Spectral Density
N_m	Set of Bit Nodes Which Check at Check Node
n_n	Normal Random Variable of Noise
Pdf	Probability Density Function
PLframe	Pilot Frame
$P_{(b)}$	Probability of Bit Error
pmf	Probability Mass Function
PN	Parity Node
Q	Code Dependent Permanent
QPSK	Quadrature Phase Shift Keying
R	Channel Rate
R	Code Rate
r	Received Vector
RA	Repeat Accumulate
r_n	nth Received Vector
RF	Radio Frequency
SISO	Soft Input Soft Output

SNR	Signal to Noise Ratio
$t_{b,d}$	Base Addresses for Each Block
TDM	Time Division Multiplexing
VCM	Variable Coded Modulation
VN	Variable Node
W	Channel Bandwidth
z_n	nth Transmitted Symbol
8PSK	8 Phase Shift Keying
α	Roll of Factor
ϕ	Phase Vector
γ	Column Degree
8PSK	8 Phase Shift Keying
η	Parallelism Factor
λ_{b_n}	A Posteriori Log Likelihood of b_n
λ_{b_n,c_m}	Messages Leaving Bit Nodes to Check Nodes
λ_{c_m,b_n}	Messages Leaving Check Nodes to Bit Nodes
ρ	Row Degree
σ^2	Variance of the Channel Noise

Chapter 1

Introduction

1.1 Overview

In order to have a reliable communication with low power consumption over noisy channels, error correcting code may be used. Error correcting codes insert redundancy into the transmitted data stream so that the receiver can detect and possibly correct errors that occur during transmission. Several types of codes exist. Each of which are suitable for some special applications.

Researchers are searching for the best codes suitable for Communication systems. There exists a large design space with trade-offs between the area of the chip, speed of the decoding, complexity, Gap to Shannon limit, and power consumption. In this thesis we discuss a particular type of error correcting codes, namely, Low Density Parity Check (LDPC) codes and its application in DVB-S2 (Digital Video Broadcasting Satellite Version 2). These codes have very good performance over noisy channels.

In this chapter, we will start with an overview of digital communication systems and coding schemes. Then, a general description of error control codes and their applications will be given. Subsequently, LDPC codes, their properties, and applications will be discussed. Finally, some of the related works regarding the architecture design for LDPC codes will be reviewed.

1.2 Digital Communication System

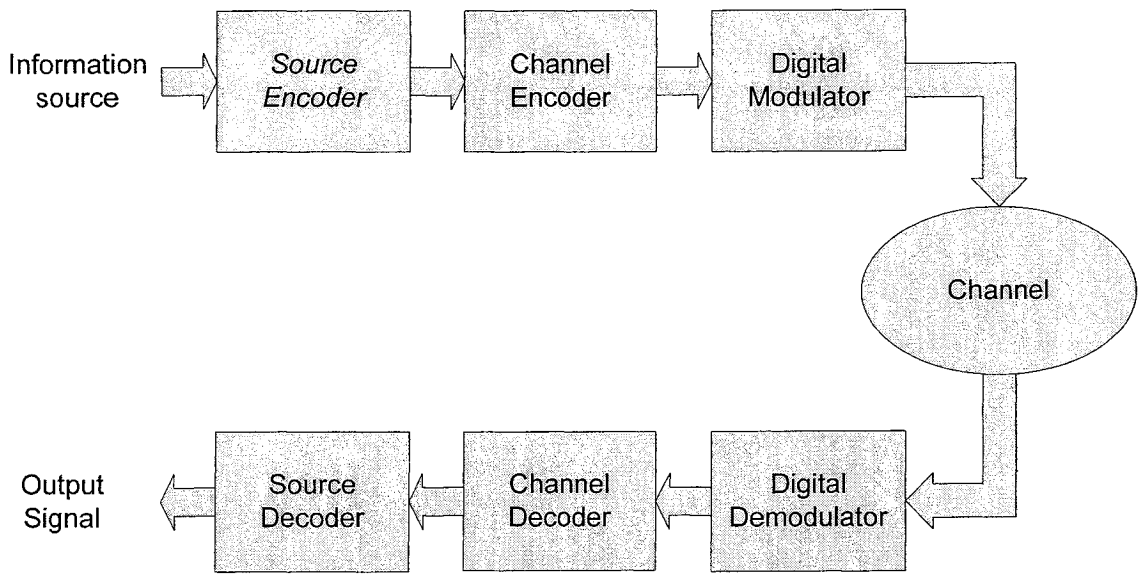


Figure 1.1: Basic elements of a digital communication system

Figure 1.1 shows a basic block diagram of a digital communication system [1]. The source input may be either an analog signal such as audio or video signal or a digital signal such as teletype machine. The messages produced by the source are converted into a sequence of binary digits. The purpose of the channel encoder is to introduce in a controlled manner, some redundancy in the binary information sequence that can be used to overcome the effects of noise and interference encountered in the transmission of the signal through the channel. Then, the binary sequence is passed to the digital modulator to map the information sequence into signal waveforms. The modulator acts as an interface between the digital signal and the channel.

The communication channel is the physical medium which is used to send the signal from the transmitter to the receiver which may be the atmosphere (for wireless

communications); wire line or optical fiber cable. The transmitted signal in all of these channels is corrupted in a random manner with additive thermal noise generated by electronic devices, manmade noise, atmosphere noise and etc.

At the receiving side the digital demodulator processes the channel-corrupted transmitted waveform and reduces the waveforms to a sequence of digital values that feed into the channel decoder. The decoder reconstructs the original information by the knowledge of the code used by the channel encoder and the redundancy contained in the received data. Then, the source decoder decompresses the data and retrieves the original information.

The basic decoding idea is based on the rules of channel encoding and the noise characteristics of the channel. The channel decoder makes an estimate of the transmitted message. There is a trade-off between the power of transmission, bandwidth, and the bit error rate. Researchers are trying to minimize the power consumption and bandwidth while maintaining a reliable communication. This increases a need for stronger codes with more error correction capabilities.

1.3 Error Correcting Codes

Error correcting coding (ECC) is the critical point of modern communication systems. In many applications a substantial portion of the baseband signal processing is dedicated to ECC. The wide range of ECC applications include, space and satellite communications, data transmission, data storage, and mobile communications.

In 1948, Claude Shannon founded the field of "Information Theory" which is the basis of modern ECC by discovering the noisy channel coding theorem [4]. In his work, he introduced a metric in which the information can be quantified. This theorem states that the capacity C (which is now known as the Shannon limit) of a band-limited Additive White Gaussian noise (AWGN) channel with bandwidth W, is given by:

$$C = W \log_2\left(1 + {}^{E_s}\!/_{N_0}\right) \text{ Bits per second (bps)} \tag{1.1}$$

where E_s is the average signal energy in each signaling interval of duration $T = 1/W$, and $N_0/2$ is the two-sided noise power spectral density. The proof of this theorem demonstrates that for any transmission rate R less than or equal to the channel capacity C, there exists a coding scheme that achieves error-free transmission. Conversely if R is greater than C, no coding scheme can achieve reliable performance. Since this theorem was published, an entire field of study has grown out to design coding schemes that approach the Shannon limit of various channels.

In table 1.1, we mention some of the applications and required coding schemes. Satellite downlinks are generally characterized as power-limited channels. On board batteries and solar cells are heavy and thus contribute significantly to launch costs. Therefore, strong error control codes need to operate efficiently at extremely low signal to noise ratio. LDPC and Turbo codes are choices for these applications. Similar principles apply to the wireless communications for cell phones and laptops. In order to increase battery life we need to use powerful codes like LDPC, or Turbo codes.

Table 1.1: Application of error correcting codes

Application	CODE	COMMENT
Wireless communications Satellite downlink	Convolution, Turbo, LDPC	Random Noise
CD player Tape storage	Reed – Solomon +cross- interleaving	Bursty channel
Computer memory	Hamming code	
Magnetic discs	Fire codes	
Computer networks	CRC	

Various applications exist for the error control codes in computer systems, such as memory (random access and read-only memory), disk storage, tape storage and inter-processor communication. Each of these has their unique characteristic that indicates the use of certain type of codes. Hamming codes are used for the computer memories, Fire codes for magnetic discs and Reed Solomon based system is used for the tape mass storage system. Computer networks and internet use Cyclic Redundancy Code (CRC) to detect packet errors [5].

1.4 LDPC Codes

In the past few years, LDPC codes have received much attention because of their excellent performance, and have been widely considered as the most promising ECC candidate for many applications in telecommunications and storage devices. LDPC codes were proposed by Gallager in 1962 [6]. He defined an (n, d_v, d_c) LDPC code as a code of block length n in which each column of the parity-check matrix contains d_v ones and

each row contains d_c ones. Due to the regular structure (uniform column and row weight) of Gallager's code, they are now called regular LDPC codes. Gallager provided simulation results for codes with blocks in the order of hundreds of bits. The results indicated that LDPC codes have very good potential for error correction. However, the high storage and computation requirements interrupted the research on LDPC codes. After discovering Turbo codes by Berrou [2], Mackay [7] [8] re-established the interest in LDPC codes during the mid to late 1990s.

LDPC codes can be used in any digital environment that high data rate and good error correction is important, such as optical fiber communications, satellite (digital Video and Audio broadcast), storage (magnetic, optical, holographic), wireless (mobile, fixed), and wired-line (cable modems, DSL). Table 1.2 shows a comparison between the best-known error correcting codes. Chung [9] showed that a rate $1/2$ LDPC code with block length of 10^7 Additive White Gaussian Noise channel can achieve a threshold of 0.0045 dB away from Shannon limit. This table shows that for very large block lengths, LDPC is the best-known code in terms of performance.

Table 1.2: Performance comparison between different types of the channel codes

Code Type	Shannon Limit	LDPC	Turbo	Convolutional /Viterbi
Performance(dB) $P_{error} = 10^{-6}$	0.18	0.185	0.6	4.5

While standards for Viterbi and turbo codes have been emerged in communication systems, the flexibility of designing LDPC codes allow for a larger family of codes and encoder /decoder structures. Some initial proposals for LDPC codes for DVB-S2 are emerging [10].

Another feature of LDPC codes is their simple graphical representation, which is based on Tanner's [11] representation of linear codes. This simple structure allows for accurate asymptotic analysis of LDPC codes as well as design of good irregular LDPC codes and optimization under specific constraints.

Irregular LDPC [12] codes can significantly outperform regular LDPC codes. These codes have approached Shannon limit for different channel type.

One issue with irregular LDPC codes is their encoding complexity. People have considered different ways to add structure to LDPC codes to make the encoding process less complex. One of the best solutions is Repeat Accumulate (RA) codes, which have a slightly different structure than LDPC codes and suffer minor performance loss compared to LDPC codes. The encoding complexity of RA codes grows linearly with the block length.

Irregular Repeat-Accumulate (IRA) codes have a linear-time encoding complexity with a straight forward hardware realization. It outperforms the currently deployed Turbo-codes. Like irregular LDPC codes, IRA [13] codes can be represented by a Tanner graph with arbitrary connections between nodes. An IRA code is defined by the degree distribution of the nodes. Their complexity strongly depends on the randomness of

the Tanner graph. Regularity in the Tanner graph simplifies the implementation but degrades the communication performance.

Table 1.3 shows a comparison between the complexity of the encoders and the decoders for three different types of coding. In this table n is the code length, d is the constraint length, J is the maximum number of the decoding iterations, d_c is the row degree, and d_v is the column degree of the nodes in the parity check matrix of the LDPC decoder. Comparison shows that LDPC decoding is linear with the block length, whereas in turbo codes, it has exponential relation with the constraint length.

Table 1.3: Complexity comparison

Code Type	Encoder	Decoder
Convolutional / Viterbi	$O(nd)$	$O(n2^d)$
Turbo	$O(n(d_1+1+d_2))$	$O(Jn(1+2^{d_1}+2^{d_2}))$
LDPC	$O(nd_c^{\ 2})$	$O(Jn(d_c+d_v))$

Complexity in iterative decoding has three parts. First, complexity of the computations at each node, second the complexity of the interconnection, and third the number of times that local computations need to be repeated, usually referred to as the number of iterations. All of these are manageable in practice. There is a trade-off between the performance of the decoder, complexity and speed of decoding.

1.5 DVB-S2

The new DVB-S2 [14] standard features a powerful forward error correction (FEC) system that enables transmission close to the theoretical limit (Shannon limit). This is enabled by using Low Density Parity Check (LDPC) codes one of the most powerful codes known today which can even outperform Turbo codes. To provide flexibility, 11 different code rates ranging from (R=1/4 up 9/10) are specified with a codeword length up to 64800 bits. This huge codeword length is the reason for the outstanding communications performance (~ 0.7 db to Shannon limit) of this DVB-S2 LDPC code proposal. The LDPC codes as defined in the DVB-S2 standard are IRA codes, thus the encoder realization is straightforward. Furthermore, the DVB-S2 code shows regularities that can be exploited for an efficient hardware realization.

The DVB-S2 standard uses a serial concatenation of two binary linear codes: an outer BCH code and inner low density parity check (LDPC) code. Note that the outer BCH code used by DVB-S2 helps to clean up additional errors at the output of the LDPC decoder and will improve the overall performance.

1.6 Thesis Contribution

In this thesis, we propose a hybrid parity check matrix with different block length which is used in DVB-S2 standard. LDPC codes that use this type of parity check matrix show performance improvements in communication systems. In this work, a parity check matrix is generated by using semi-random technique, consisting of two submatrices of $H^{(u)}$ and $H^{(p)}$. $H^{(u)}$ is defined as a randomly generated matrix, while $H^{(p)}$ is a

deterministic one. Moreover, $H^{(u)}$ is divided in small blocks, where each block has a special base address. Weight distribution of the random part has major effect in code performance. The method for replacing and expanding '1's in parity check matrix is done in both of regular and irregular techniques. In the irregular method, at first we find base addresses and then with the help of a simple rule, the other entries will be found. This hybrid parity check matrix can be considered as an efficient tool for practical hardware implementation of both decoder and encoder. As we will show in this thesis, the performance of the new code is better than the performance of the LDPC code which is used in DVB-S2. However, both codes have the same complexity.

1.7 Organization of the Thesis

In chapter 2 we provide the necessary background on DVB-S2 and LDPC codes. Various types of LDPC encoding and decoding algorithms are discussed.

In chapter 3 we explain new parity check matrix which can be used in DVB-S2 standard and the conventional design method for analysis, designing and checking performance of irregular LDPC codes.

In chapter 4 we describe the simulation procedure, the class of decoding algorithm and analysis of the decoder. In this chapter we show some simulation results.

Finally a summary of contributions and some suggestions for future work are provided in chapter 5.

Chapter 2

Background

2.1 DVB-S2

The European Telecommunications Standard Institute (ETSI) with the goal of standardizing digital television services founded the Digital video Broadcasting (DVB) project in 1993 [15]. Its initial standard for the satellite delivery of digital television, dubbed DVB-S, used a concatenation of an outer (204,188) shortened Reed Solomon code and an inner constraint length 7, variable rate (R ranging from 1/2 to 7/8) convolutional code.

The latest standard, called DVB-S2, replaces the concatenated Reed-Solomon /convolutional coding approach of DVB-S with a concatenation of outer BCH code and inner low-density parity check (LDPC) code. The result is a 30% increase in capacity over DVB-S.

The new DVB-S2 standard presents a powerful forward error correction (FEC) system, which enables transmission close to the theoretical limit (Shannon limit). This is possible using low density parity check (LDPC) codes which are one of the most powerful kinds, even outperforming Turbo codes. To provide flexibility, 11 different code rates ranging from (R=1/4 up to 9/10) are specified with a codeword length up to 64800 bits. This huge maximum codeword length is the reason for the outstanding communications performance (~0.7 dB to Shannon limit).

DVB-S2 is a single and flexible standard that covers a variety of applications by satellite, as described below. It is characterized by the following features:

- A flexible input stream adapter, suitable for operation with single and multiple input streams of various formats (packetized or continuous);
- A powerful FEC system based on LDPC (low density parity check) codes concatenated with BCH codes, allowing Quasi-Error-Free operation at about 0.7 dB to 1 dB from the Shannon limit, depending on the transmission mode;
- A wide range of code rates (from 1/4 to 9/10); 4 constellations, ranging in spectrum efficiency from 2 bits/s/HZ to 5 bits/s/HZ, optimized for operation over non-linear transponders;
- A set of three spectrum shapes with roll-off factors 0.35, o.25and 0.20;
- Adaptive coding and modulation (ACM) functionality, optimizing channel coding and modulation on a frame-by-frame basis.

DVB-S2 is suitable for use on different satellite transponder bandwidths and frequency bands. The symbol rate is matched to give transponder characteristics, and in the case of multiple carriers per transponder (FDM), it is possible to adopt bandwidth to the frequency plan. Digital transmissions via satellite are affected by power and bandwidth limitations. Therefore, DVB-S2 provides for many transmission modes (FEC coding and modulations) giving different trade-offs between power and spectrum efficiency.

DVB-S2 is compatible with Moving Picture Experts Group (MPEG-2 and MPEG-4) coded TV services with a transport stream packet multiplex. Multiplex flexibility allows the use of the transmission capacity for a variety of TV service configurations,

including sound and data services. All service components are Time Division Multiplexed (TDM) on a single digital carrier.

2.1.1 Transmission System Description

Figure 2.1 shows the functional block diagram of DVB-S2 [14] system.

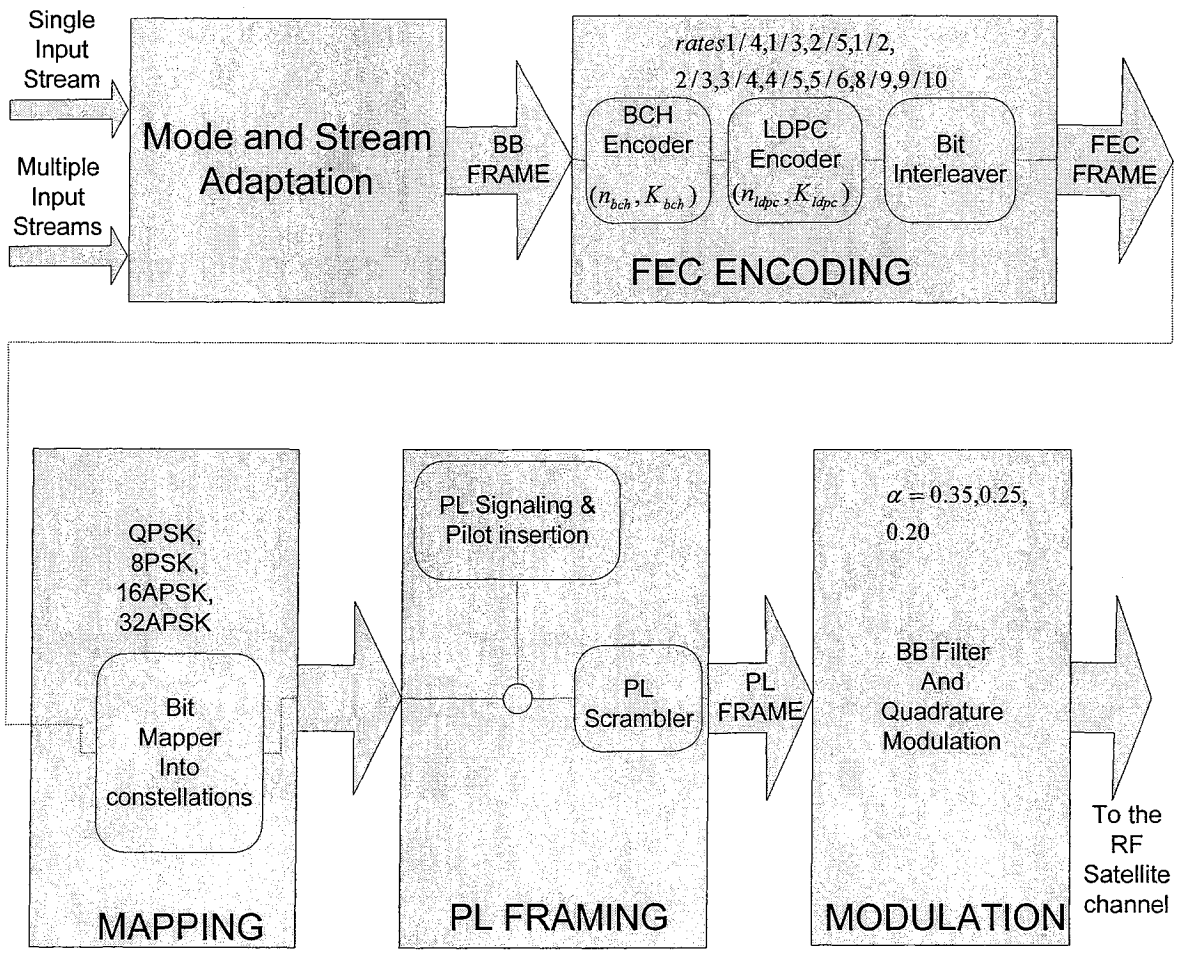


Figure 2.1: Functional block diagram of the DVB-S2 transmission system

2.1.2 Mode and Stream Adaptation

The mode and stream adaptation block provides input stream interfacing, input stream synchronization (optional), null packet deletion (for the ACM and Transport Stream input format only), CRC-8 coding for error detection at packet level in the receiver (for packetized input stream only), and slicing into Data Fields. This block also provides padding to complete the base-band frame and base-band scrambling.

2.1.3 Forward Error Correction (FEC) Encoding

It will be carried out by the concatenation of BCH outer codes and LDPC inner codes (rates 1/4,1/3,2/5,1/2,3/5,2/3,3/4,4/5,5/6,8/9,9/10). Depending on the application area, the FEC coded block will have the length of 64800 or 16200 bits. When VCM and ACM are used, FEC and modulation mode may be changed in different frames but they remain constant within a frame. Bit interleaving will be applied to FEC coded bits for 8PSK, 16APSK, and 32APSK.

Each BBFRAME (K_{bch} bits) will be processed by the FEC coding subsystem to generate a FECFRAME (n_{ldpc} bits). The parity check bits (BCHFEC) of the systematic BCH outer code append after the BBFRAME and the parity check bits (LDPCFEC) of the inner LDPC encoder append after the BCHFEC field as shown in Figure 2.2.

For 8PSK, 16APSK, and 32APSK modulation formats, the output of the LDPC encoder will be interleaved using a block interleaver. Data are serially written into the interleaver column-wise and serially read out row-wise.

Unlike turbo codes, LDPC codes have an easily parallelizable decoding algorithm which consists of simple operations such as addition, comparison, and look-up table. Moreover, the degree of parallelism is adjustable, which makes it easy to trade-off throughput and complexity.

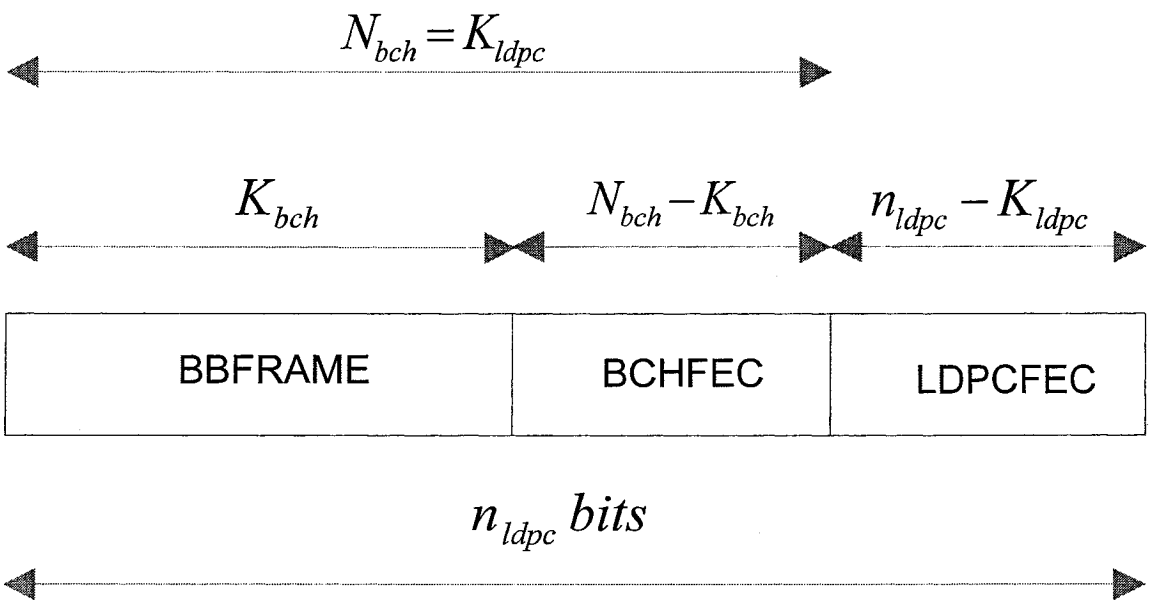


Figure 2.2: Format of data before bit interleaving

(n_{ldpc}=64800 bits for normal FECFRAME , n_{ldpc}=16200 bits for short FECFRAME)

2.1.4 Mapping

After bit interleaving, mapping QPSK, 8PSK, 16APSK, and 32APSK constellations will be applied depending on the application area. Gray mapping of constellations will be used for QPSK and 8PSK.

Each FECFRAME (which is a sequence of 64800 bits for normal FECFRAME, or 16200 bits for short FECFRAME) must be converted to parallel sequence (parallelism factor η_{mod}=2 for QPSK, 3 for 8PSK, 4 for 16APSK, and 5 for 32 APSK). Each parallel sequence will be mapped into a constellation. This mapping generates a sequence of variable length (I, Q) depending on the selected modulation efficiency η_{mod}.

The input sequence is FECFRAME and the output sequence is XFECFRAME. Complex XFECFRAME are composed of $64800/\eta_{\text{mod}}$ or $16200/\eta_{\text{mod}}$ modulation symbols. Each modulation symbol will be a complex vector in the format (I, Q) (I being the in-phase component and Q the quadrature component) or in the equivalent format $\rho\exp(j\varphi)$ (ρ being the modulus of the vector and φ being its phase).

For QPSK and 8PSK, the system employs conventional Gray-coded modulation with absolute mapping (no differential coding). Bit mapping for QPSK and 8PSK are shown in Figure 2.3 and Figure 2.4 respectively. The normalized average energy per symbol is equal to $\rho^2 = 1$.

The 16APSK modulation constellation is composed of two centric rings of uniformly spaced 4 and 12 PSK points; the radius of inner ring is R1 and the outer ring is R2. If $4[R1]^2 + 12[R2]^2 = 16$, the average signal energy becomes 1.

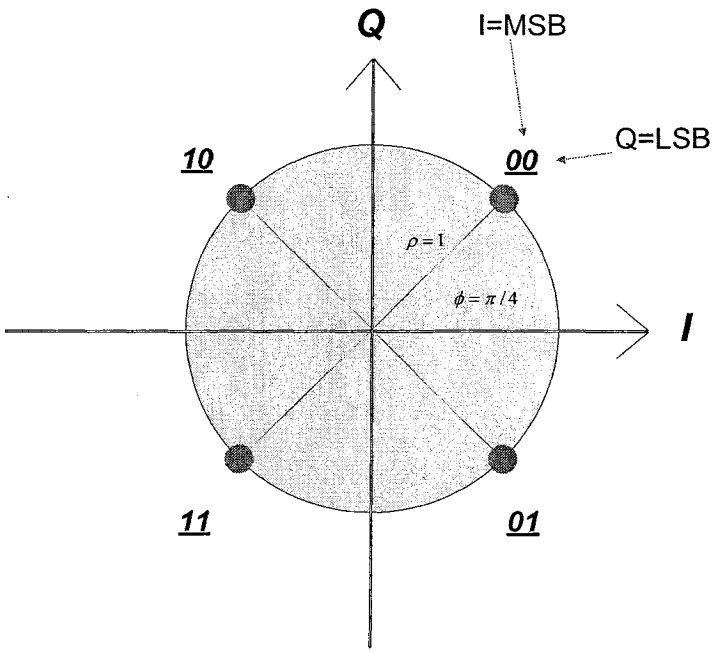


Figure 2.3: Bit mapping into QPSK constellation

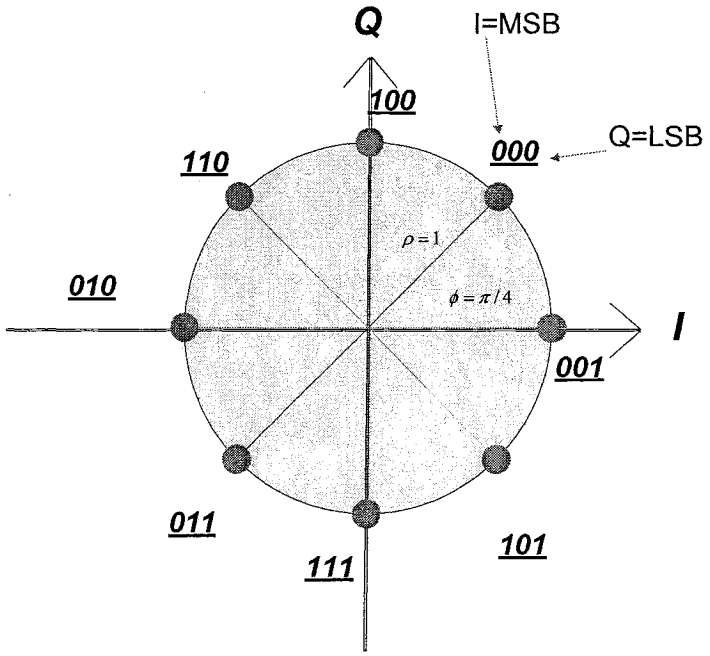


Figure 2.4: Bit mapping into 8PSK constellation

The 32APSK modulation is composed of three concentric rings of uniformly spaced 4, 12 and 16PSK points in the inner ring of radius R1, the intermediate ring of radius R2 and the outer ring of radius R3, respectively. If $4[R1]^2 + 12[R2]^2 + 16[R3]^2 = 32$, the average signal energy becomes equal to 1. Constellation labeling is shown in Figures 2.5 and 2.6. 16APSK and 32APSK (as opposed to 16QAM and 32QAM) are chosen due to their "non-linearity friendly" characteristics. Moreover, their performances on linear channels are almost as good as their QAM competitors.

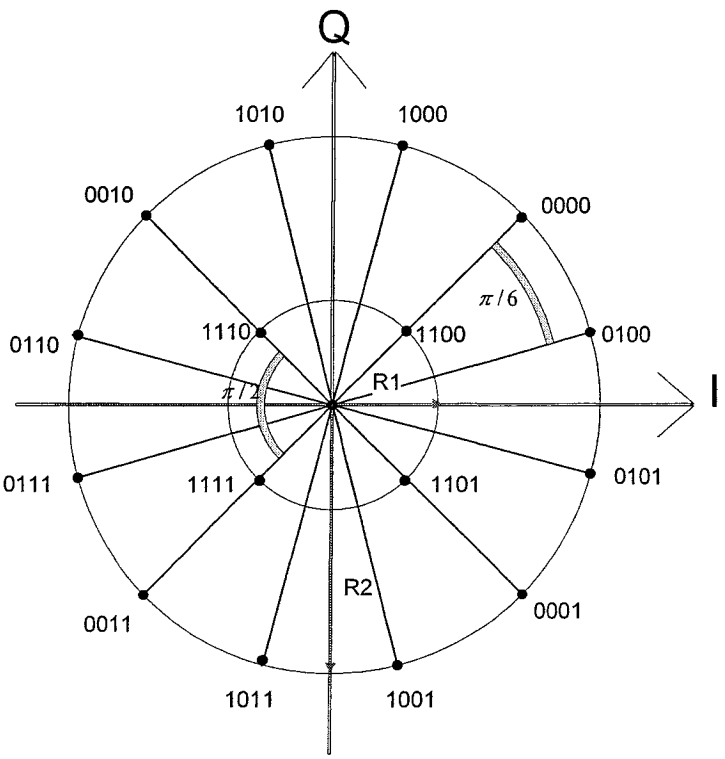


Figure 2.5: 16 APSK signal constellation

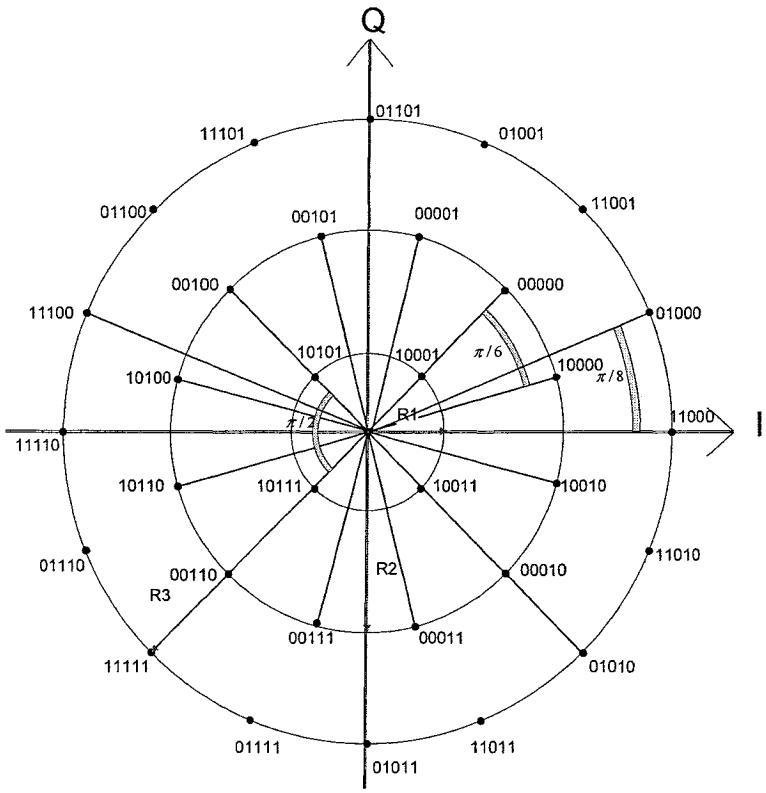


Figure 2.6: 32 APSK signal constellation

2.1.5 Physical Layer Framing

This block is employed for synchronization with the FEC frames and provides dummy PLFRAME insertion, physical layer signaling, pilot symbols insertion (optional), and physical layer scrambling for energy dispersal. Dummy PLFRAMEs are transmitted when no useful data are ready to be sent on the channel. Carrier recovery in the receivers may be facilitated by the introduction of a regular raster of pilot symbols, while a pilot-less transmission mode is also available, offering an additional 2.4% useful capacity.

2.1.6 Base-band Filtering and Quadrature Modulation

This block is employed for shaping the signal spectrum and for generating the RF signal. Its frequency characteristic is the square-root raised cosine with the roll-off factor of 0.35 or 0.25 or 0.20. The baseband square root raised cosine has an analytical function defined by the following expression:

$$H(f) = 1 \qquad for \quad |f| < f_N(1-\alpha) \tag{2.1}$$

$$H(f) = \left\{\frac{1}{2} + \frac{1}{2}\sin\frac{\pi}{2f_N}\left[\frac{f_N - |f|}{\alpha}\right]\right\}^{1/2} \quad for \quad f_N(1-\alpha) < |f| < f_N(1+\alpha) \tag{2.2}$$

$$H(f) = 0 \qquad for \quad |f| > f_N(1+\alpha) \tag{2.3}$$

where $f_N = \frac{1}{2T_s} = \frac{R_s}{2}$ is the Nyquist frequency and α is the roll-off factor.

Quadrature modulation will be performed by multiplying the in-phase and quadrature samples (after baseband filtering) by $\sin(2\pi f_0 t)$ and $\cos(2\pi f_0 t)$ respectively, where f_0 is the carrier frequency. The two resulting signals will be added to obtain the modulator output signal.

2.2 Linear Block Codes

The low-density parity check coding scheme is one of the hottest topics in coding theory. It was introduced by Gallager in his Ph.D thesis in early 1960. It is a special case of linear block codes and for this reason, an introduction to these classes of codes is necessary for a better understanding of LDPC codes.

Linear block codes are a class of parity check codes that can be characterized by (n,k) notation [16]. The encoder transforms a block of k message digits into a longer block of n codeword digits constructed from a given alphabet of elements. For linear codes, the mapping transformation is linear. Linear codes have the following interesting properties:

1. The linear combination of any set of codeword is also a codeword. Therefore, linear codes always contain the all zero codeword.
2. The minimum distance of a linear code is equal to the weight of the lowest weight nonzero codeword.
3. The undetectable error patterns for a linear code are independent of the codeword transmitted and always consist of a set of all nonzero codewords.

A desirable property that a linear block code may posses is the systematic structure of the codeword. A systematic codeword is divided into two parts: the message part and the redundant checking part

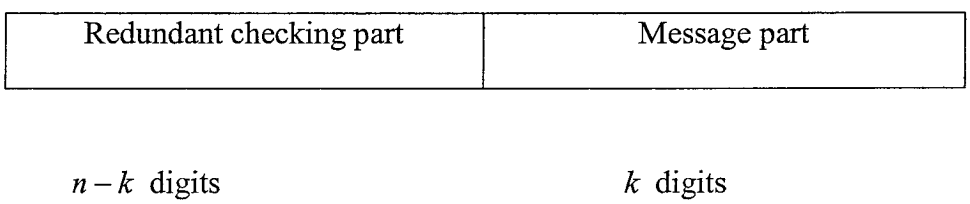


Figure 2.7: Systematic format of a codeword

the message part consists of k unaltered information digits, and the redundant checking part consists of $n-k$ parity check digits, which are linear sums of the information digits.

A linear systematic (n,k) code is completely specified by $k \times n$ matrix G of the following form:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ . \\ . \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} \cdots & p_{0,n-k-1} & \vdots & 1 & 0 & 0 & . & . & . & . & 0 \\ p_{10} & p_{11} \cdots & p_{1,n-k-1} & \vdots & 0 & 1 & 0 & . & . & . & . & 0 \\ p_{20} & p_{21} \cdots & p_{2,n-k-1} & \vdots & 0 & 0 & 1 & . & . & . & . & 0 \\ . & . & . & \vdots & & & & & & & & . \\ . & . & . & \vdots & & & & & & & & \\ p_{k-1,0} & p_{k-1,1} \cdots & p_{k-1,n-k-1} & \vdots & 0 & 0 & 0 & . & . & . & . & 1 \end{bmatrix} \quad (2.4)$$

where p_{ij} =0 or 1 and I_k is $k \times k$ identity matrix. Then, $G = [P | I_k]$.

The generation of the codeword U is written in matrix notation as the product of m (message vector) and G, and we write

$$U = mG \quad (2.5)$$

Thus, the code vector corresponding to a message vector is a linear combination of the rows of G. Since the code is totally defined by G, the encoder needs only to store the k rows of G instead of the total vectors of the code.

There is another useful matrix associated with all linear block codes, called the parity check matrix that will enable us to decode the received vectors. For any $k \times n$ matrix G with k linearly independent rows, there exists an $(n-k) \times n$ matrix H with $n-k$ linearly independent rows, such that any vector in the row space of G is orthogonal to the rows of H, and any vector that is orthogonal to the rows of H is in the row space of G. Hence, $GH^T = 0$, where H^T is the transpose of H and 0 is a $k \times (n-k)$ all zeros

matrix. To fulfill the orthogonality requirements for a systematic code, the components of the H matrix are written as:

$$H = \left[I_{n-k} \middle| P^T \right] \tag{2.6}$$

We can use the parity check matrix to test whether or not a received vector is a valid member of the codeword set. U is a codeword generated by matrix G if and only if :

$$U.H^T = 0. \tag{2.7}$$

When a data block is encoded using a systematic generator matrix, the data block is embedded without modification in the last k coordinates of the resulting codeword.

$$\begin{aligned} c &= mG \\ &= [m_0 \quad m_1 \cdots\cdots m_{k-1}] \quad [P|I_k] \\ &= [c_0 \quad c_1 \cdots\cdots c_{n-k-1} \vdots \quad m_0 \quad m_1 \cdots\cdots m_{k-1}] \end{aligned}$$

After decoding, the last k symbols are removed from the selected codeword and passed along to the data sink. Given a systematic generator matrix, the corresponding parity check matrix can be obtained as shown below:

$$H = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & \vdots & p_{00} & p_{10} & \cdots & p_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & \vdots & p_{01} & p_{11} & \cdots & p_{k-1,1} \\ 0 & 0 & 1 & \cdots & 0 & \vdots & p_{02} & p_{12} & \cdots & p_{k-1,2} \\ & & & & & & & & & \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \vdots & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \tag{2.8}$$

and we can obtain the generator matrix from the parity check matrix in systematic form by using Gaussian elimination. By these definitions and properties, we are ready to discuss LDPC code properties.

2.3 Low Density Parity Check Codes

LDPC codes are systematic, block codes that can be represented in either of the two basic ways: analytical or graphical. The analytical way is a matrix representation, whereas the graphical approach is described with Tanner graphs (or bi-partite graphs). In matrix representation, LDPC codes can be described with a check constraints matrix H which is sparse. This means that the density of '1's in this matrix is very low.

The matrix H has n columns and $n-k$ rows, n being the size of the codeword and k the size of the information block. $n-k$ are also referred to as the number of constraints of the LDPC code.

$$H = \{h_{ij}\}_{n-k,n} \qquad i = 0,\cdots,n-k-1 \qquad ,j = 0,\cdots,n-1$$

Elements h_{ij} are binary symbols drawn from the set $\{0,1\}$ and the jth coded bit is checked at the ith constraint node if and only if h_{ij}=1.

An LDPC code is defined as the null space of a parity check matrix H that has the following structural properties:

- each row consists of ρ 1's;
- each column consists of γ 1's;

- the number of '1's common between any two columns, denoted by z, is no greater than 1 ; that is, z=0 or 1;
- both ρ and γ are small compared to the length of the code and the number of rows in H.

This is the definition of regular LDPC codes but if degrees per row or column are not constant, then the code is irregular. Some of the irregular codes have shown better performance than regular ones. Code rate R is equal to k/n (k be information bit and n is code length) which means that $(n-k)$ redundant bits have been added to the message so as to correct the errors.

2.3.1 Tanner Graph

LDPC codes can be represented effectively by a bi-partite graph called a "Tanner" graph [11]. A graph is most commonly represented by a diagram in which the vertices are represented by points and each edge by a line joining its end vertices. Figure 2.8 shows a graph consisting of six vertices and 10 edges. The number of edges that are incident on a vertex v_i is called the degree, denoted by $d(v_i)$ of vertex v_i as shown in Fig 2.8.

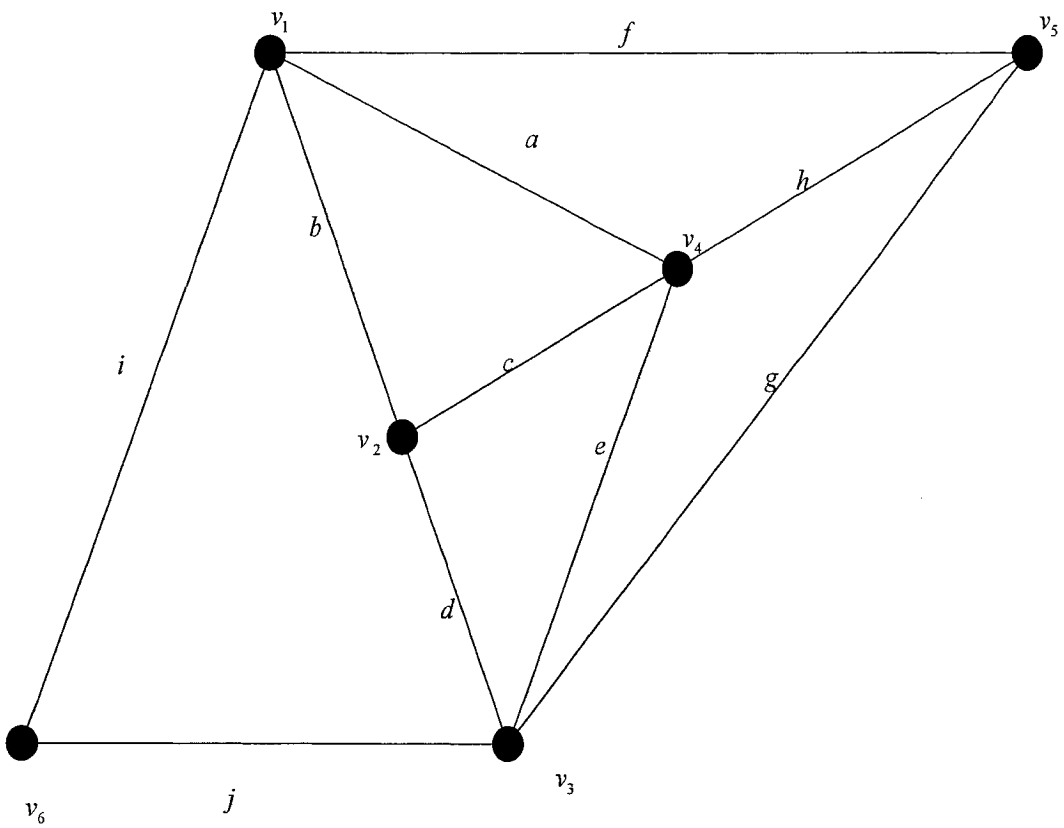


Figure 2.8: A graph with six vertices and 10 edges

In the Tanner graph, following characteristics hold:

- Two edges that are incident on common vertex are said to be adjacent or connected.
- Two vertices are said to be adjacent if they are connected by an edge.
- A graph with a finite number of vertices as well as a finite number of edges is called a finite graph.
- The number of edges on a path is called the length of the path.
- A cycle of length l in a graph is a path comprised of l edges which closes back on itself.
- The length of the shortest cycle in a graph is called the girth of the graph.

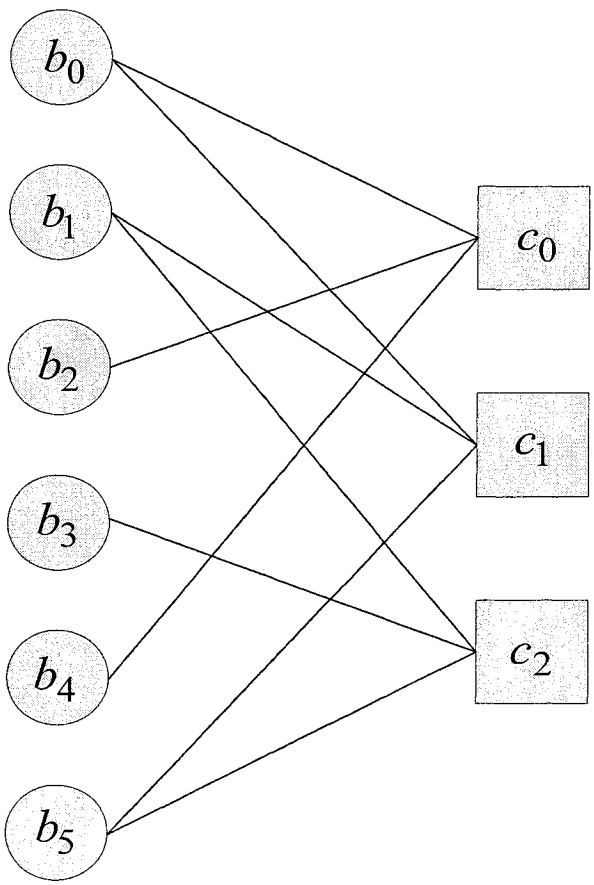
- A graph without cycles is said to be acyclic and is called a tree.
- An edge for which the initial and terminal vertices are the same forms a cycle of length 1 called a self loop.

A graph $g = (v, \varepsilon)$ is called a bipartite graph if its vertex set v can be partitioned into two disjoint subsets v_1 and v_2 such that every edge in ε joins the vertex in v_1 with a vertex in v_2 and no two vertices in either v_1 or v_2 are connected. If a bipartite graph $g = (v, \varepsilon)$ contains cycles, then all the cycles have even lengths and it is obvious that a bipartite graph has no self loop.

For LDPC codes with the length n, and the parity-check matrix H, a Tanner graph consists of two sets of vertices. The first set consists of n vertices that represent the n code bits of the code and is called the code-bit vertices (or variable nodes). The second set consists of $(n-k)$ vertices that represent the $(n-k)$ parity –check sums (or equations) that the code bit must satisfy. These vertices are called the check-sum vertices (or check nodes). No two code-bit vertices are connected, and no two check-sum vertices are connected.

The degree of a code-bit vertex is simply equal to the number of the parity sums that contain this code-bit, and the degree of the check-sum vertex is simply equal to the number of code bits that are checked by this check-sum vertex. For a regular LDPC code, the degrees of all the code-bit vertices in the tanner graph of the code are the same and equal to γ (the column weight of the parity-check matrix); and the degrees of all the check-sum vertices are the same and equal to ρ (the row weight of the parity-check matrix). For an irregular LDPC code, the degrees of code-bit vertices and check-sum vertices are different and depend on codes that have been used. From the definition of

LDPC codes and for better performance in decoding, no two code bits of an LDPC code are checked by two different parity-check sums. Now let us introduce a simple example of an LDPC code. Consider parity check matrix H which describes an irregular LDPC code with rate $R = 1/2$, information block size k=3, and code length n=6. Figure 2.9 shows the matrix and the related bipartite graph.



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

Figure 2.9: Tanner graph of the irregular LDPC code of matrix H

The shortest possible cycle in a bipartite graph is clearly length 4 which means four '1's that lie in the corners of a sub-matrix of H. We are interested in cycles, especially in short cycles, because they have a negative effect on the decoding algorithm for the LDPC codes.

2.4 Designing LDPC Code

The first step in designing an LDPC code is answering the following questions:

- What is the desired block length of the code? It is obvious that codes with large block lengths have very good performance, but the problem is that large block lengths have a high complexity and are unfeasible in practice.
- What is the type of code? Regular or irregular?
- What is the degree of each bit node or check node? In other words, how many ones are allowed in each row or column of the parity check matrix? For the regular codes, degrees of all bit nodes will be the same. For irregular codes, one should decide how many different degrees are allowed for the bit nodes and check nodes. High degrees mean that more computations should be done in each node to generate the outgoing message. Also, nodes with higher degrees have faster convergence to their correct value, and in decoding with a high degree, complexity will be increased.
- What is the rate of the code? The rate of the code show us how much redundancy we have in the code and how much bandwidth is allowed to increase, because if for better performance we increase the rate it means that we must increase our bandwidth at the end of communication systems.

- What is the maximum number of decoding iterations? It is possible that by increasing the maximum iteration number, code performance can improve but it increases decoding time which is important in some applications. After deciding the above parameters, we can design the parity check matrix.

2.4.1 Designing the Parity Check Matrix

The parity check matrix plays a major role in the performance of the LDPC encoding/decoding. As mentioned by Gallager, this matrix should be very sparse. It also determines the complexity of the encoder/ decoder. Depending on the application, this matrix can be random or structured. Random matrices are suitable for decoders running on general purpose processors, but for special hardware like FPGAs or ASIC, it is better to have a structured matrix. The structure in the parity check matrix leads to a more efficient hardware representation. It also requires less memory to keep the matrix. Here, we will list ways to generate a sparse matrix H. Some of these ways are more complex than others, but they don't necessarily lead to a better code.

- Start from an all zero matrix of the size $(n-k)\times n$ and randomly invert some elements in the matrix to reach the resulting degrees for different nodes.
- Generate H by randomly creating weight γ columns.
- Generate H with weight γ columns and uniform row weight ρ.
- Generate H with columns of weight γ and uniform rows of weight ρ; with no two columns having an overlap of more than one. This condition removes all the length four cycles which results in better performance.

- Generating H and avoiding all short cycles (4, 6…).
- Generating the parity check matrix in a structured manner. For example, a structure that is used in hardware design is to generate this matrix using a combination of the shifted blocks of identity matrices.
- Generate the parity check matrix using a polynomial.

Each of the above ways has their advantages and disadvantages, and we can choose them based on the application. Now we are ready to discuss LDPC encoding.

2.5 LDPC Encoding

The encoding problem consists of solving a set of linear equations or constraints. A generic word U with size n is a valid codeword if and only if the following condition is met:

$$H.U^T = 0 \tag{2.9}$$

The dot product in (2.9) provides the parity of the codeword. In particular, even parity is adopted, and equation (2.9) states that the number of '1's in the codeword b must be even at any check constraint.

LDPC codes are systematic codes, thus the codeword b can be thought as:

$$U = (b, P)$$

where b is the systematic part (the information data) of b with size k , and P is the redundant part with size $n-k$:

$$\begin{cases} b = (b_0 \quad b_1 \cdots \quad b_{k-1}) \\ P = (p_0 \quad p_1 \cdots \quad p_{n-k-1}) \end{cases} \tag{2.10}$$

The redundant part P represents the added value of the encoding process and allows for forward error correction at the receiver.

Having the parity check matrix of a set of LDPC codes, we can draw the corresponding Tanner graph. To give a general perspective about encoding of the LDPC codes first, we assign each of the information bits to bit nodes in the graph, and then the values of the remaining bit nodes are determined so that all the parity check constraints satisfy. In this way, encoding LDPC codes is selecting the nodes for assigning information bits and a strategy for calculating the values of the other bit nodes. This is the process of encoding in the matrix notation:

$$H = [P^T \vdots I] \Rightarrow G = [I \vdots P] \Rightarrow C = mG$$

in which C is the n bit codeword and $G_{k \times n}$ is the generator matrix of the code and m is the k - bit message.

At first glance, encoding might seem to be a computationally extensive task, since all the parity check equations should satisfy, which can be in quadratic relation with the code length. But in reality, encoding can be done very efficiently and the encoding complexity can be a fraction of the decoding complexity. Several low complexity algorithms exist for the encoding of the LDPC codes. Some techniques exploit the sparseness of the parity check matrix for efficient encoding. Another approach is to impose some structure to the Tanner graph so that the encoding is transparent and simple. Repeat –Accumulate codes are examples of structured graphs.

2.5.1 Gaussian Elimination

Gaussian elimination represents the most straightforward way to encode a generic LDPC code [17]. With Gaussian elimination the check constraint matrix H can be put in the following form by permutation of its rows and columns:

$$H = \left(H^{(u)}, H^{(p)} \right) \tag{2.11}$$

$H^{(u)}$ is the $(n-k)\times k$ systematic part and $H^{(P)}$ the parity part of H. Matrix $H^{(u)}$ is sparse, while $H^{(P)}$ is a lower triangular matrix with size $n-k$:

$$H^{(P)} = \begin{pmatrix} 1 & 0 & & 0 \\ x & 1 & 0 & 0 \\ x & x & 1\cdots & 0 \\ \vdots & & & \\ x & x\cdots & & 1 \end{pmatrix} \tag{2.12}$$

where $x = \{1,0\}$. This approach allows the encoder to determine the parity bits of the codeword by means of so called back substitution. With the same notation, it is easy to show that the encoding problem ends up in solving the following set of equations:

$$p_l = \sum_{i=0}^{k-1} H_{l,i}^{(u)} u_i + \sum_{i=0}^{l-1} H_{l,i}^{(p)} p_i \tag{2.13}$$

which are successively computed for $l = 0,1,\cdots\cdots,n-k-1$. It can be proven that the complexity of the encoding process is proportional to the square of information bit size, $O(n^2)$.

2.5.2 Efficient Encoding Techniques

Several techniques are currently used to reduce the encoding complexity such as the one proposed by Urbanke and Richardson [18]. They show that the encoding is

accomplished in two steps, a preprocessing step, which is an offline calculation performed once only for the given code, and the actual encoding step which is the only data-dependent part.

The efficiency of the encoder is proportional to the sparseness of the parity-check matrix H and the algorithm can be applied to any sparse H. If by performing row and column permutations we convert the parity-check matrix into the form indicated in Figure 2.10, we say that H is in the approximate lower triangular form.

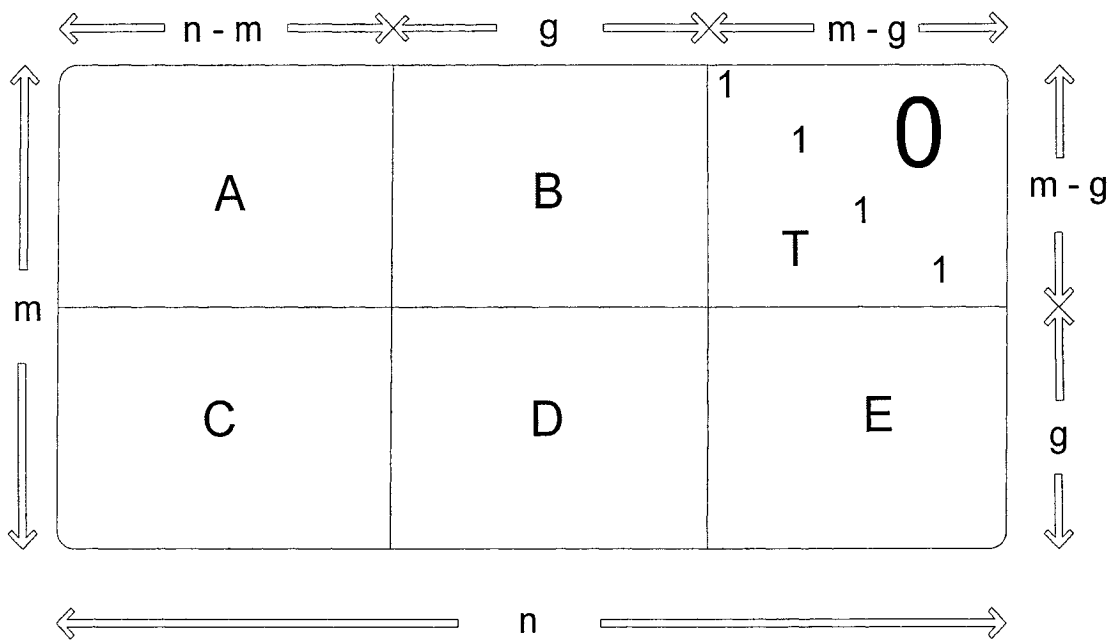


Figure 2.10: The parity check matrix in approximate lower triangular form

The matrix in Figure 2.10 can be shown as,

$$H=\begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix} \tag{2.14}$$

where A is $(m-g)\times(n-m)$, B is $(m-g)\times g$, T is $(m-g)\times(m-g)$, C is $g\times(n-m)$, D is $g\times g$ and finally E is $g\times(m-g)$. Further, all of these matrices are sparse and T is lower triangular with ones along the diagonal. Multiplying this matrix from the left by

$$\begin{pmatrix} I & 0 \\ -ET^{-1} & I \end{pmatrix} \tag{2.15}$$

we obtain

$$\begin{pmatrix} A & B & T \\ -ET^{-1}A+C & -ET^{-1}B+D & 0 \end{pmatrix} \tag{2.16}$$

If $x=(s,p_1,p_2)$ where s denotes the systematic part, p_1 and p_2 combined denote the parity part, p_1 has length g and p_2 has length $(m-g)$, then we can define the equation $Hx^T=0^T$ naturally into the two following equations:

$$As^T+Bp_1^T+Tp_2^T=0 \tag{2.17}$$

$$(-ET^{-1}A+C)S^T+(-ET^{-1}B+D)p_1^T=0 \tag{2.18}$$

By defining as $\phi=-ET^{-1}B+D$ and assuming that ϕ is non singular, then we conclude from (2.18) that

$$p_1^T=-\phi^{-1}(-ET^{-1}A+C)s^T \tag{2.19}$$

$$p_2^T=-T^{-1}(As^T+Bp_1^T) \tag{2.20}$$

The proposed encoding procedure can be summarized in two steps: a preprocessing step and actual encoding step. In the preprocessing step, we first perform row and column permutations to bring the parity check matrix into approximate lower triangular form with small gap.

2.5.3 Semi-random LDPC

In semi-random LDPC the parity check matrix features some regularity which is used to reduce the encoding complexity. There are two main examples of semi-random LDPC:

- π - rotation LDPC codes [19].
- Codes proposed by Hughes Network Systems [20] for the DVB-S2 standard [14].

As we show, matrix H splits into two parts which corresponds to the systematic and parity bits of the codeword. Matrix $H^{(u)}$ is a random matrix, whilst $H^{(p)}$ is bi-diagonal.

$$H^{(p)} = \begin{pmatrix} 1 & 0 & & & \cdots & 0 \\ 1 & 1 & 0 & & \cdots & 0 \\ 0 & 1 & 1 & 0\cdots & & 0 \\ 0 & 0 & 1 & 1 & 0\cdots & 0 \\ \vdots & & & & & \\ 0\cdots & & & & 1 & 1 \end{pmatrix} \tag{2.21}$$

From equation (2.9) we can write:

$$H^{(u)}.u^T + H^{(p)}.p^T = 0 \tag{2.22}$$

or equivalently:

$$H^{(u)}.u^T = H^{(p)}.p^T = z. \tag{2.23}$$

To encode one frame at first, we solve $H^{(u)}.u^T = z$, which in overall calls for $k.(d_c - 1)$ XOR's (assume regular LDPC code). Then, the redundant bit should be computed through back substitution:

$$\begin{cases} p_0 = z_0 \\ p_l = z_l + p_{l-1} \qquad l = 1,2,\ldots\ldots,n-k-1 \end{cases} \tag{2.24}$$

It should be noted that all the computations for the encoding are on binary values. So, instead of adders and multipliers, XORs and AND gates can be used which are cheaper. As back substitution only requires $n-k-1$ XOR's, the overall computational complexity amounts to $n(1-R)d_c - 1 \approx O(n)$.

Matrix H must also be stored in a memory and from sparseness of this matrix it is efficient to store only the address of its non-zero elements, so the resulting memory requirement is:

$$(n - k).(d_c - 2).q \tag{2.25}$$

where $q = \log_2 n$ is the number of bits used to represent the addresses of the '1's within a row of H.

2.6 LDPC Decoding

Decoding of LDPC codes is an iterative message passing algorithm. The reason for their name is that at each round of the algorithm, messages are passed from message (bit) nodes to check nodes, and from check nodes back to message nodes [21]. The messages from message nodes to check nodes are computed based on the observed value of the message node and some of the messages passed from the neighboring check nodes to that message node. One important subclass of message passing algorithms is the belief propagation algorithm. The messages passed along the edges in this algorithm are probabilities or beliefs. More precisely, the message passed from a message node v to a check node c is the probability that v has a certain value given the observed value of that message node, and all the values communicated to v in the prior round from check

nodes incident to v other than c. On the other hand, the message passed from c to v is the probability that v has a certain value given all the messages passed to c in the previous round from message nodes other than v.

In order to describe the iterative decoding, we need to use a Tanner graph for LDPC coding. We have to, because the messages are passing along the edges of the Tanner graph. Local computations are done in each node of the graph. To facilitate the subsequent iterative processing, the graph must be designed as sparse as possible. Although that approach can be suboptimal, it is usually quite close to optimal and has an excellent complexity versus performance trade off.

2.6.1 Bit Flipping Algorithm

In this section we explain a simple hard decision decoding algorithm which is known as "bit flipping algorithm" [6]. It is interesting to use this algorithm in very high speed applications, such as optical networks. Despite its lower performance, bit flipping algorithm has lower complexity advantage over message passing. This algorithm works based on the hard decision of the received signal and the messages are considered to be single bits.

The idea behind this algorithm is to flip the least number of bits until all the check constraints are satisfied. Assume that each bit node starts with a value of either zero or one. In each iteration, the bit node decides to flip its value or to keep it unchanged. When large numbers of neighboring check equations are unsatisfied, the bit node decides to flip its value. The algorithm has been considered from the assumption that the bit node value in error has the most number of unsatisfied check equations. This process is repeated

until all the parity check constraints are satisfied, at this point the modified received sequence is a codeword. This decoding is an iterative decoding algorithm. For error patterns whose number of errors is less than or equal to the error-correcting capability of the code, the decoding will be completed in one or few iterations. Otherwise, more decoding iterations are needed. Therefore, the number of decoding iterations is a random variable and is a function of the channel SNR. A limit may be set on the number of iterations. When this limit is reached, the decoding process is terminated to avoid excessive computations. Owing to the nature of low-density parity checks, bit flipping decoding algorithm corrects many error patterns whose number of errors exceeds the error-correcting capability of the code. This process is easier when H is low density, when only a few bits are involved in each check equation and each bit is involved in only a few equations. A very simple BF decoding algorithm is given here:

- Compute the parity check sums (syndrome bits). If all the parity check sums are zero, stop the decoding.
- Find the number of failed parity check equations for each bit, denoted by f_i, $i = 0,1,\cdots,n-1$.
- Identify the set s of bits for which f_i is the largest.
- Flip the bits in set s.
- Repeat steps 1 to 4 until all the parity check sums are zero or a preset maximum number of iterations is reached.

2.6.2 The Sum-Product Algorithm

The sum-product algorithm is an iterative decoding algorithm based on belief propagation that is extremely efficient for decoding LDPC codes. This algorithm is a

symbol-by-symbol soft-in, soft-out decoding algorithm. It processes the received symbols iteratively to improve the reliability of each decoded code symbol. With sparse parity-check matrix H of an LDPC code, parity-check sums are computed from the hard decisions of the received symbols in an iterative manner. The computed reliability measures of the code symbols at the end of the each decoding iteration are used as input for the next iteration. The decoding iteration process continues until a certain stopping condition (or criteria) is satisfied. Then, based on the computed reliability measures of code symbols, hard decisions are made.

To perform this decoding type, real number addition, subtraction, multiplication, division, exponential and logarithm operations are needed. The computational complexity and decoding delay (or decoding time) of the sum product algorithm increases when the number of iterations increases. Long decoding delays are not desirable in high speed communication and data storage systems.

With sum product algorithm, the error performance of an LDPC code depends on several important parameters:

- the girth of its Tanner graph;
- The minimum distance;
- the column and row weights of the parity-check matrix;
- the number of minimum weight codeword.

2.7 Code Performance

The performance of a coded communication system is generally measured by probability of bit error (BER) rate. A coded communication system should be designed to

keep the bit error rate as low as possible under certain system, such as power, bandwidth and decoding complexity.

The other performance measure of a coded communication system is the coding gain. Coding gain is defined as the reduction in the E_b / N_o required to achieve a specific error probability (BER) for a coded communication system compared to an uncoded system. In low values of E_b / N_o, the coding gain becomes negative. There exists a specific value for SNR which is called coding threshold. Any SNR below this threshold degrades the performance of a coded system compared to uncoded one. It is important to keep this threshold low and to be sure that the code system operates at higher than this threshold.

Another method that measures the code power is gap to Shannon limit. Shannon limit is based on Shannon noisy coding theorem [4]. This theoretical limit says that for coded system with code rate R, error free communication is achievable only if the SNR exceeds this limit. As long as the SNR exceeds this limit, Shannon's coding theorem guarantees the existence of a coded system capable of achieving error free communication. For transmission over a binary input, continuous output AWGN channel with BPSK modulation, the Shannon limit (in terms of E_b / N_o) as a function of code rate R does not have a closed form, however, it can be evaluated numerically. Table 2.1 shows the Shannon limit as a function of the code rate R for BPSK modulation on a continuous output AWGN channel [16]. Gap from Shannon limit can be reduced by using a longer and more powerful code, decoded with an efficient soft decision MLD or near MLD algorithm.

Table 2.1: Shannon limit of a continuous output AWGN channel with BPSK signaling for various code rates R

R	$E_b / N_o (dB)$	R	$E_b / N_o (dB)$	R	$E_b / N_o (dB)$	R	$E_b / N_o (dB)$
0.01	-1.548	0.35	-0.432	0.69	1.208	0.954	4.304
0.02	-1.531	0.36	-0.394	0.70	1.275	0.958	4.425
0.03	-1.500	0.37	-0.355	0.71	1.343	0.961	4.521
0.04	-1.470	0.38	-0.314	0.72	1.412	0.964	4.618
0.05	-1.440	0.39	-0.276	0.73	1.483	0.967	4.725
0.06	-1.409	0.40	-0.236	0.74	1.554	0.970	4.841
0.07	-1.378	0.41	-0.198	0.75	1.628	0.972	4.922
0.08	-1.347	0.42	-0.156	0.76	1.708	0.974	5.004
0.09	-1.316	0.43	-0.118	0.77	1.784	0.976	5.104
0.10	-1.285	0.44	-0.074	0.78	1.867	0.978	5.196
0.11	-1.254	0.45	-0.032	0.79	1.952	0.980	5.307
0.12	-1.222	0.46	0.010	0.80	2.045	0.982	5.418
0.13	-1.190	0.47	0.055	0.807	2.108	0.983	5.484
0.14	-1.158	0.48	0.097	0.817	2.204	0.984	5.549
0.15	-1.126	0.49	0.144	0.827	2.302	0.985	5.615
0.16	-1.094	0.50	0.188	0.837	2.402	0.986	5.681
0.17	-1.061	0.51	0.233	0.846	2.503	0.987	5.756
0.18	-1.028	0.52	0.279	0.855	2.600	0.988	5.842
0.19	-0.995	0.53	0.326	0.864	2.712	0.989	5.927
0.20	-0.963	0.54	0.374	0.872	2.812	0.990	6.023
0.21	-0.928	0.55	0.424	0.880	2.913	0.991	6.119
0.22	-0.896	0.56	0.474	0.887	3.009	0.992	6.234
0.23	-0.861	0.57	0.526	0.894	3.114	0.993	6.360
0.24	-0.827	0.58	0.574	0.900	3.205	0.994	6.495
0.25	-0.793	0.59	0.628	0.907	3.312	0.995	6.651
0.26	-0.757	0.60	0.682	0.913	3.414	0.996	6.837
0.27	-0.724	0.61	0.734	0.918	3.500	0.997	7.072
0.28	-0.687	0.62	0.791	0.924	3.612	0.998	7.378
0.29	-0.651	0.63	0.844	0.929	3.709	0.999	7.864
0.30	-0.616	0.64	0.904	0.934	3.815		
0.31	-0.579	0.65	0.960	0.938	3.906		
0.32	-0.544	0.66	1.021	0.943	4.014		
0.33	-0.507	0.67	1.084	0.947	4.115		
0.34	-0.469	0.68	1.143	0.951	4.218		

Chapter 3

A New Parity Check Matrix for LDPC Code Suitable for DVB-S2

3.1 Organization of the Parity Check Matrix

As mentioned before in chapter 2 LDPC codes have recently been selected as the forward error correcting scheme for the new generation satellite Digital Video Broadcasting (DVB-S2) [14].

LDPC codes for DVB-S2 are semi-random and has parity check matrix as in 2.11. A high degree of regularity in random part of the matrix $H^{(u)}$ helps to implement highly parallelized decoder architectures.

The LDPC codes as defined in the DVB-S2 standard are IRA codes, thus the encoder realization is straightforward. Furthermore, the DVB-S2 code shows regularities which can be exploited for an efficient hardware realization [22]. This type of LDPC codes can be represented by a bipartite graph. The Tanner graph for DVB-S2 is shown in Figure 3.1. For the DVB-S2 code, 64800 variable nodes (VN) and $64800 \times (1-R)$ check nodes exist. LDPC codes can be specified through their parity check matrices. For LDPC code in DVB-S2 certain structure is imposed on parity check matrix H. The main criteria for parity check matrix structure is having low complex encoding and decoding algorithm.

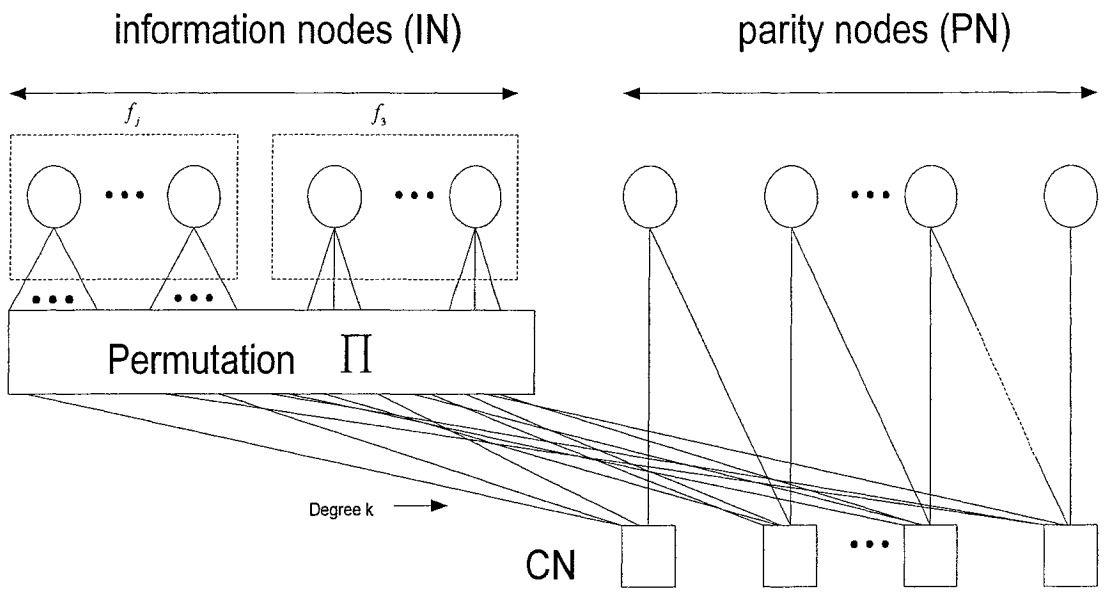


Figure 3.1: Tanner graph for the DVB-S2 LDPC code

To understand the organization of the DVB-S2 parity check matrix; let us see Fig 3.2 that focuses on the code rate $R = k_{LDPC} / n_{LDPC}$. If we assume block length of $n_{LDPC} = 64800$, therefore the information word is $k_{LDPC} = n_{LDPC} \times R$. Placement of the '1's into the parity check matrix is performed by resorting to code specific table that assembles the base address of the parity check matrix. More precisely we just assign the placement of the '1's into a set of $L = 360$ consecutive columns of the parity check matrix $H^{(u)}$. As a consequence, the parity check matrix is split overall into $B = k_{LDPC} / L$ blocks. This is graphically depicted in Figure 3.2.

Within a block, the '1's are placed according to the base addresses specified in the table. There is an address in the table for each sub-block of $H^{(u)}$. In particular, if we show the sub-blocks of $H^{(u)}$ with $b = 0,1,.....,B-1$, the '1's of the b th block are placed at the locations (i, j) where:

- $i \in [0, n_{LDPC} - k_{LDPC} - 1]$ is the row address determined from the table.
- $j = 0, \ldots\ldots, L-1$ is the column address related to the current block.

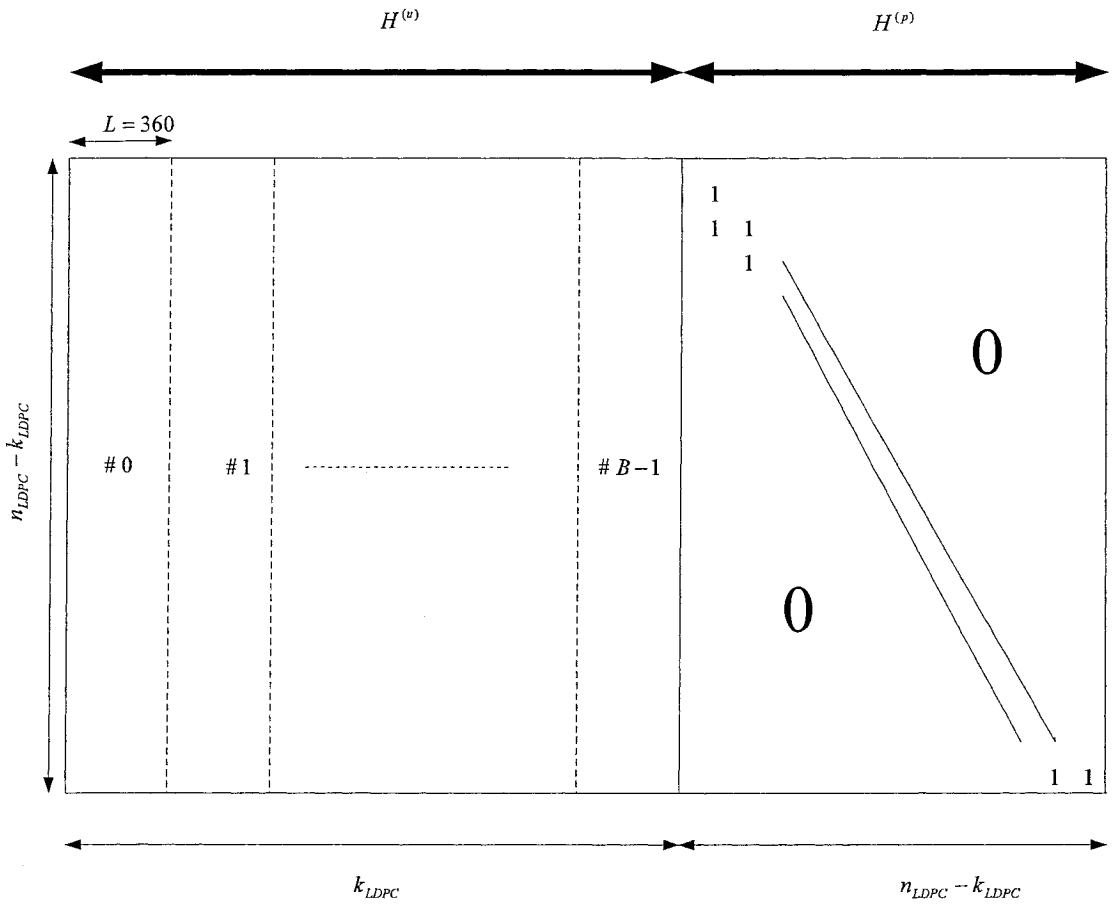


Figure 3.2: Organization of the parity check matrix of the DVB-S2 with code rate R

The DVB-S2 codes exhibit certain regularity that row addresses of the '1's are computed in a deterministic way from the table. If D_b is the bit node degree of the bit nodes in the b th block (degree D_b is common to any bit node within block b) and $t_{b,d}$ is the address of '1's in the first column that exist in table and $d = 0,1,\ldots\ldots, D_b - 1$. In

each column of the block, the location of the non-zero elements is formed by the following algorithm:

$$i = \left| t_{b,d} + j.Q \right|_{(n_{LDPC} - k_{LDPC})} \qquad (3.1)$$

In the above equation, i is in the range $[0, n_{LDPC} - k_{LDPC} - 1]$, j is the column index (relative to block b) and $Q = \frac{n_{LDPC} - k_{LDPC}}{L}$ is a code-dependent parameter (for example for $R = 2/3$ $Q = \frac{64800 - 43200}{360} = 60$). Address table gives the address of the '1's of the first column of the current block then the remaining '1's are derived from these base addresses and the above algorithm. For better illustration, the generic structure of a sub-block of H is plotted in Fig 3.3. As we know, address table does not specify the same bit node degree for any sub-block of the parity check matrix because of the DVB-S2 codes are irregular. For example for code rate $R = 2/3$ [14], the first twelfth sub-blocks has weight 13 ($D_b = 13, b = 0, 1, ..., 11$) and the following $B - 12 = 102$ blocks has weight 3 ($D_b = 3, b = 12, 13, ..., B - 1$). This means that $12 \times L = 4320$ bit nodes out of 64800 have degrees 13 and $102 \times L = 36720$ have degree 3. The remaining bit nodes are only involved in the parity part $H^{(p)}$ of H, therefore their degree is simply 2 (except the last bit node, whose degree is 1). The list of bit node degrees and the total number of nodes with those degrees are shown in table 3.1 for all code rates. For each code rate, a parity check matrix is specified by listing adjacent check nodes for the first bit node in a group of $L = 360$. For a group of L bit nodes, if the check nodes connected to the ith bit node

of degree, say D_b are numbered as $c_1, c_2, c_3, \ldots\ldots\ldots\ldots, c_{D_b}$ then the check nodes connected to the j th bit node $(j \leq L)$ are numbered as:

$$\{c_1 + jQ\} \bmod (n_{LDPC} - k_{LDPC}), \ldots\ldots\ldots\ldots\ldots\ldots, \{c_{D_b} + jQ\} \bmod (n_{LDPC} - k_{LDPC}) \qquad (3.2)$$

where $n_{LDPC} - k_{LDPC}$ = total number of check nodes [23].

For the following groups of L bit nodes, the check nodes connected to the first bit node of the group are in general randomly chosen so that the resulting LDPC code is cycle-4 free and occurrence of cycle-6 is minimized. Too many short cycles (such as cycle-4 and cycle-6) in bipartite graph decreases the code performance since they lead to non-extrinsic information being fed back after a small number of iterations.

From the above description, it is clear that the adjacent check nodes of only one bit node need to be specified in a group of L, simplifying the code description by a factor of L. In DVB-S2 standard, L is equal to 360.

As we see in the first group we select bit nodes with high degree. The reason for this is due to the fact that bit nodes with high degrees collect more information from their adjacent check nodes. At first they get corrected after a small number of iterations. Then they help other bit nodes get corrected through iterative decoding, similar to 'wave effect'. When all bit nodes have the same degrees, as in regular codes, this wave effect is not present and all the bit nodes can simultaneously correct during the decoding process. The list of bit node degrees and the total number of nodes with those degrees are shown in Table 3.1 for all codes.

Even though the above design restricts the parity check matrix to be structured, the performance is still very good due to the careful choice of check node/bit node

connections. The maximum number of decoder iterations is 50. If a valid codeword is not found at the end of iteration, the decoder estimates its outputs.

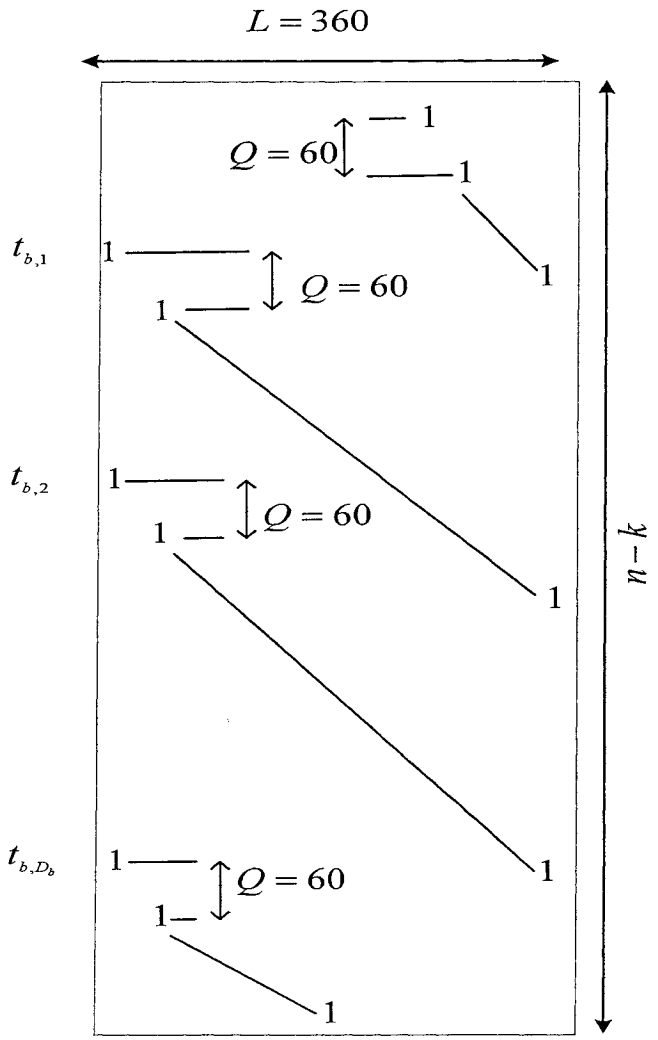


Figure 3.3: Internal organization of the bth sub-block matrix $H^{(u)}$

Table 3.1: Number of bit nodes of various degrees in DVB-S2

Code rate	13	12	11	8	4	3	2	1
1/4		5400				10800	48599	1
1/3		7200				14400	43199	1
1/2				12960		19440	32399	1
3/5		12960				25920	25919	1
2/3	4320					38880	21599	1
3/4		5400				43200	16199	1
4/5			6480			45360	12959	1
5/6	5400					48600	10799	1
8/9					7200	50400	7199	1
9/10					6480	51840	6479	1

3.2 LDPC Codes: Analysis

Knowledge about iterative decoding is required to analyze different types of LDPC codes. An iterative decoder in each iteration uses two sources of knowledge about the transmitted codeword:

- Intrinsic information: information from the channel
- Extrinsic information: information from the previous iteration

From these two sources of information, the decoding algorithm attempts to obtain a better knowledge about the transmitted codeword and uses this knowledge as the extrinsic information for the next iteration (Fig 3.4).

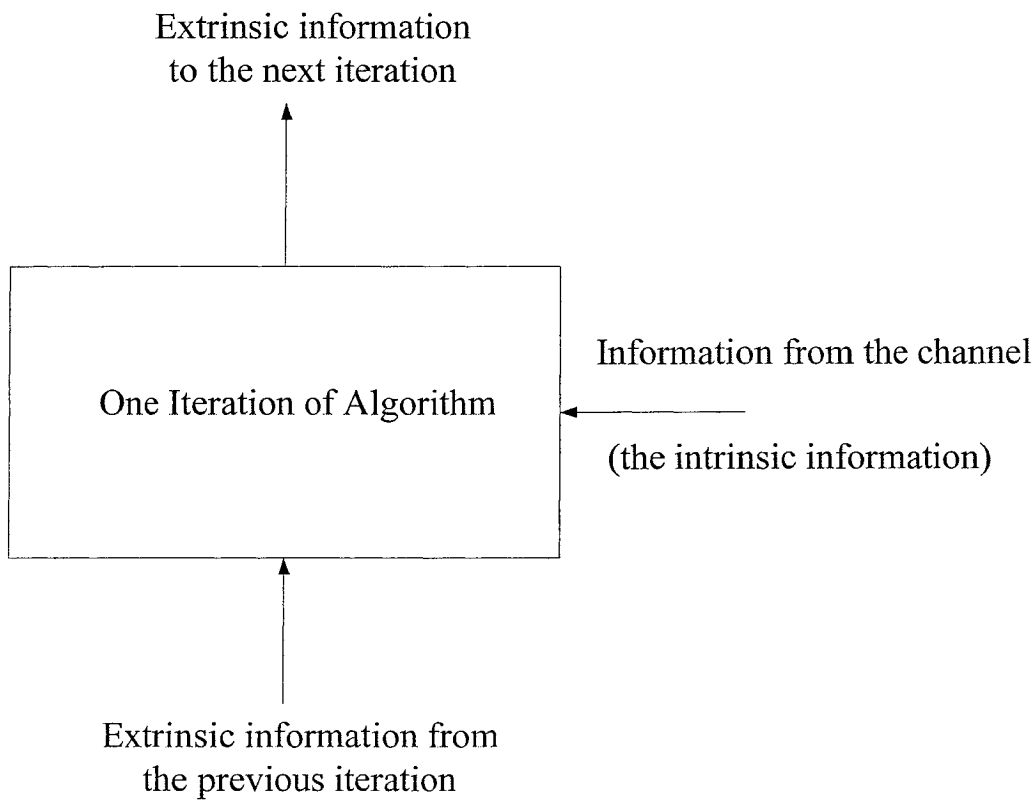


Figure 3.4: The principle of iterative decoding

In a successful decoding, the extrinsic information gets better and better as the decoder iterates. Therefore, in all methods of analysis of iterative decoders, the statistics of the extrinsic messages in each iteration are studied.

Density evolution is the most complete analysis method for studying the evolution of the pdf of extrinsic messages iteration by iteration [21].

Consider an updated message from a variable node to a check node in the decoder. This message is computed from incoming messages and the channel message. Those incoming messages are in fact the outgoing messages of some check nodes, which were updated previously. It forms a decoding tree of depth one as shown in Figure 3.5.

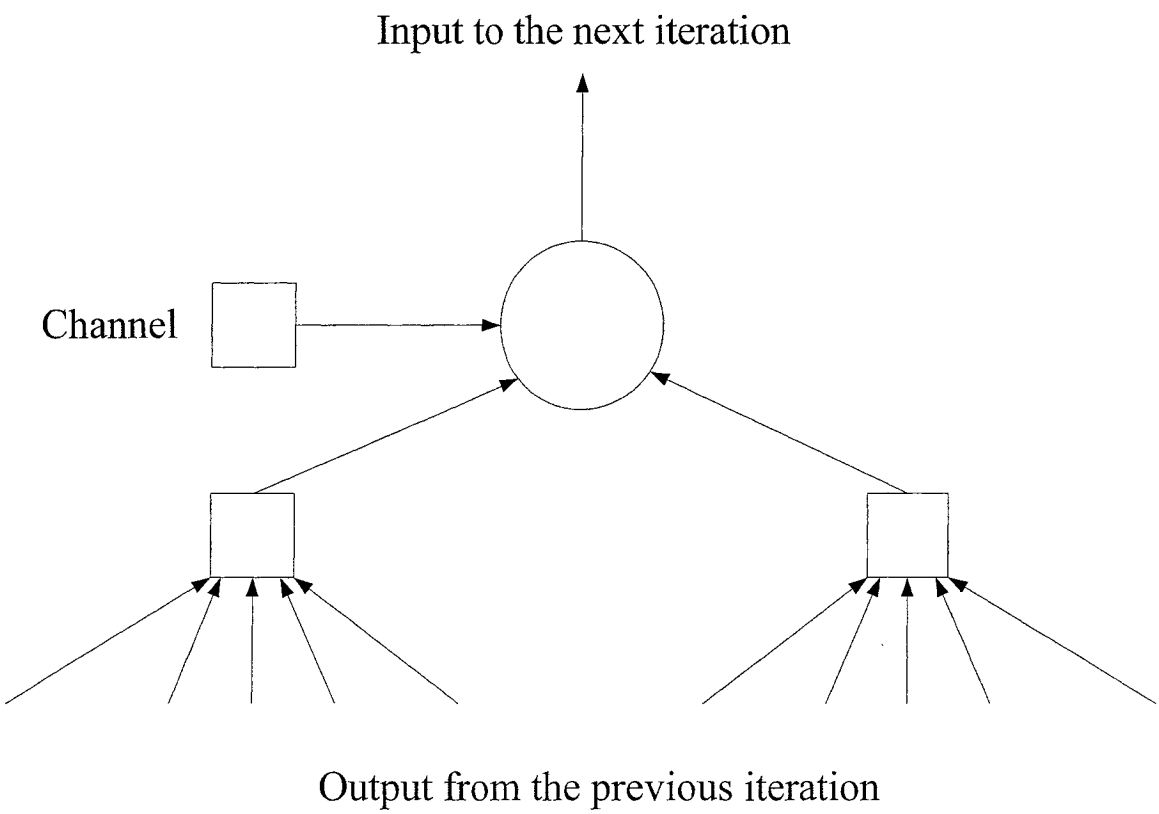


Figure 3.5: The depth-one decoding tree

Continuing in the same fashion, we can get the decoding tree for any depth. Figure 3.6 shows an example of a depth-two decoding tree for an irregular LDPC code. As we see for an irregular code decoding tree for different variable nodes are different.

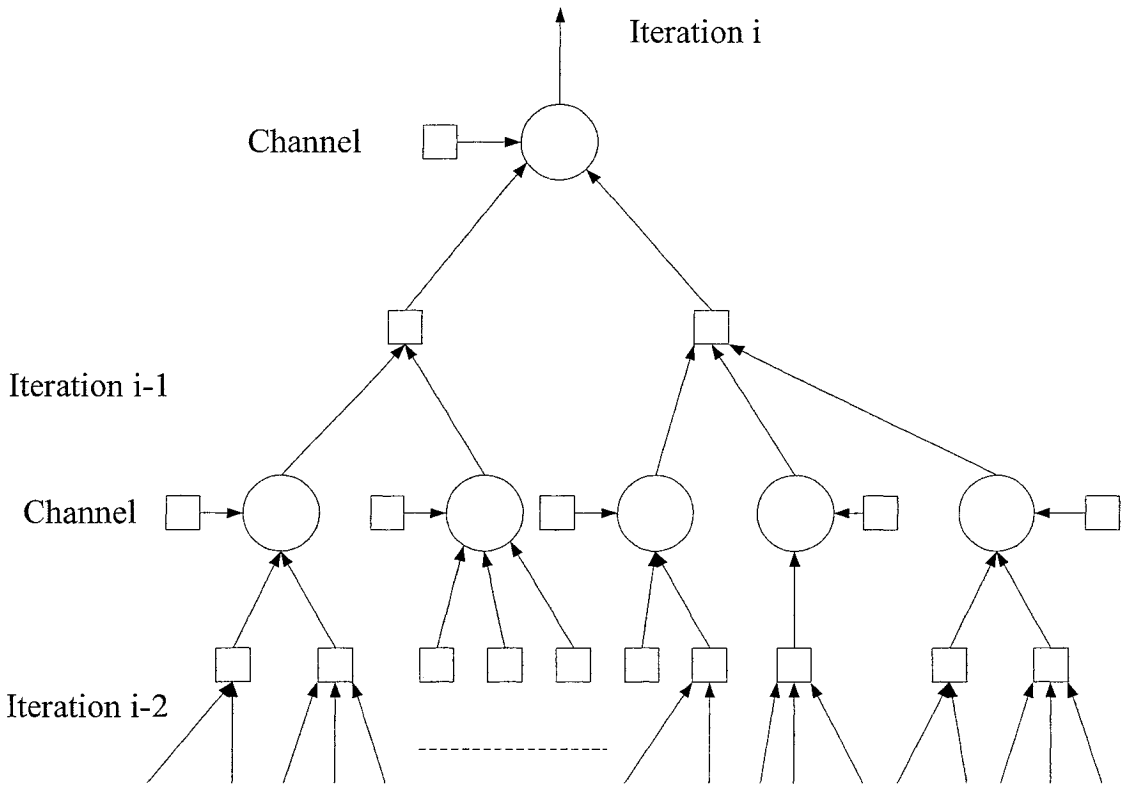


Figure 3.6: A depth-two decoding tree for an irregular LDPC code

Notice when the factor graph is tree, the messages in the decoding of any depth is independent. If the factor graph has cycles and its girth is l, then up to depth $\frac{l}{2}$ the messages in the decoding tree are independent. Therefore, the independence assumption is correct up to $\frac{l}{2}$ iterations and it is an approximation for further iterations.

3.3 Density Evolution for LDPC Codes

In 2001, Richardson and Urbanke extended the main idea of LDPC code analysis. They proposed a technique called density evolution, which tracks the evolution of the pdf of the messages, iteration by iteration. They also showed that for many interesting channels, including additive white Gaussian noise (AWGN) channels, one can calculate a threshold value for the ensemble of randomly constructed LDPC codes which determines the boundary of the error-free region asymptotically, as the block length tends to infinity.

To be able to define a density for the messages, they needed a property for channel and decoding, called symmetry conditions. The symmetry conditions require the channel and the decoding update rules to satisfy some symmetry properties as follows.

The channel is said to be output-symmetric if

$$f_{Y|X}(y|x=1) = f_{Y|X}(-y|x=-1) \tag{3.3}$$

where $f_{Y|X}$ is the conditional pdf of Y given X. The check node update rule is symmetric if

$$\mathrm{CHK}(b_1 m_1, b_2 m_2, \ldots\ldots\ldots, b_{d_c-1} m_{d_c-1}) = \mathrm{CHK}(m_1, m_2, \ldots\ldots\ldots, m_{d_c-1}) \prod_{i=0}^{d_c-1} b_i \tag{3.4}$$

for any $^{+}_{-}1$ sequence $(b_1, b_2, \ldots\ldots\ldots\ldots, b_{d_c-1})$. Here, CHK ( ) is the check update rule, which takes $d_c - 1$ messages to generate one output message. The variable node update rule is symmetric if

$$\mathrm{VAR}(-m_0, -m_1, -m_2, \ldots\ldots\ldots\ldots, -m_{d_v-1}) = \text{-}\mathrm{VAR}(m_0, m_1, m_2, \ldots\ldots\ldots\ldots, m_{d_v-1}). \tag{3.5}$$

Here, VAR ( ) is the variable node update rule, which takes $d_v - 1$ messages together with the channel message m_0 to generate one output message.

The symmetry conditions arise because, under symmetry conditions, the convergence behavior of the decoder is independent of the transmitted codeword.

Assume that we send all zero codeword, under this assumption a message carrying a belief for '0' is a correct message and a message carrying a belief for '1' is an error message and error rate can be defined for the messages.

Density evolution is not specific to LDPC codes. It is a technique which can be adopted for other codes defined on graphs associated with iterative decoding. However, it becomes intractable when the constituent codes are complex (for example turbo codes). Even in the case of LDPC codes with the simplest constituent code (simple parity checks), this algorithm is quite computationally intense.

For possible computer implementation, if the message alphabet is quantized and we use pmfs (probability mass function) instead of pdfs (power spectral density), we have discrete density evolution. In [21] Richardson and Urbank proved that there is a worst case channel condition for which the message error rate approaches zero as the number iterations approaches infinity. This is the threshold of the code for this channel. For example the threshold of the regular (3, 6) code on the AWGN channel under sum-product decoding is 1.1015 dB, which means that if infinity (3, 6) code were used on an AWGN channel, convergence to the zero error rate is guaranteed when E_b/N_o is more than 1.1015. If the channel condition is worse than the threshold, a non-zero error rate is guaranteed. In practice, when finite-length codes are used there is a gap from performance to the threshold which grows as the code length decreases.

If the threshold of a code is equal to the Shannon limit, then the code is said to be capacity achieving.

Density evolution provides exact analysis for infinitely long codes, and is an approximate analysis for finite codes.

3.4 Analysis and Design of Irregular LDPC Codes

Analysis of LDPC codes is possible through density evolution. In using density evolution, we assume a code with a very long block length. Given the initial probability density function (pdf) of the log likelihood ratio (LLR) messages, one can compute the pdf of LLR messages at any iteration. With this method, we can check whether or not the decoder converges to zero error probability for a special channel. This allows the design of irregular LDPC codes to perform very close to the Shannon limit, and to find the convergence threshold of different irregular codes and to choose the best one.

Irregular LDPC codes are most commonly designed and constructed based on their Tanner graphs. One such approach is to design these codes in terms of degree distributions of the variable and check nodes of their Tanner graphs. In Tanner graph the variable nodes correspond to the columns of H and the check nodes correspond to the rows of H. The degree of a node is defined as the number of edges connected to it. The degree of a variable node is equal to the weight of its corresponding column in H, and the degree of a check node is equal to the weight of its corresponding row in H. Let

$$\gamma(X) = \sum_{i=1}^{d_v} \gamma_i X^{i-1} \tag{3.6}$$

be the degree distribution of the variable nodes, where γ_i denotes the fraction of variable nodes in Tanner graph with degree i, and d_v denotes the maximum variable-node degree. Let

$$\rho(X) = \sum_{i=1}^{d_c} \rho_i X^{i-1} \tag{3.7}$$

be the degree distribution of the check nodes that denotes the fraction of the check nodes in Tanner graph with degree i; and d_c denotes the maximum check node degree. Urbank and Richardson [12], [21] have shown that the error performance of an irregular LDPC code depends on the variable and check node degree distributions. As we described before, we can design LDPC codes with good performance by optimizing the two degree distributions via density evolution. We have shown; for straightforward encoding, we need to have about $(n_{LDPC} - k_{LDPC})$ degree 2 bit nodes; and with this large number of degree 2 variable nodes, a high error floor may result; and the code may perform poorly in the low bit error rates. Note that a large number of degree 2 variable nodes, in general, results in poor minimum distance. To overcome these problems and increase code performance the following additional design rules have been proposed:

1) the degree 2 variable nodes are made cycle free;
2) degree 2 variable nodes must correspond to the parity check bits $(n_{LDPC} - k_{LDPC})$ of a codeword;
3) the code graph is free of length 4 cycles;
4) length 6 cycles must be minimized;

5) the number of degree 2 variable nodes must be smaller than the number of parity check bits of the code; in other words the number of columns of the parity check matrix H with weight 2 must be smaller than the number of rows of H [24], [25];

6) Low degree variable nodes are made to correspond to non systematic bits.

As we showed in our work, some of these rules are satisfied. This type of parity check matrix have $n_{LDPC} - k_{LDPC} - 1$ weight 2 columns plus one weight 1 with a bi-diagonal structure. This structure uses straightforward encoding and satisfies rules 1, 2 and 5 above. In constructing parity check matrix we put columns with weight 2 in the parity check position and make them cycle free.

$$H^{(p)} = \begin{bmatrix} 1 & 0 & 0 & 0 \cdots\cdots\cdots\cdots 0 \\ 1 & 1 & 0 & 0 \cdots\cdots\cdots\cdots 0 \\ 0 & 1 & 1 & 0 \cdots\cdots\cdots\cdots 1 \\ 0 & 0 & 1 & 0 \cdots\cdots\cdots\cdots 0 \\ \vdots & \vdots & \vdots & \vdots \qquad\qquad \vdots \\ 0 & \cdots & & \qquad\qquad 1 \end{bmatrix}$$

To satisfy these rules, $H^{(u)}$, as shown in 3.2 should have variable nodes greater than weight 2. In DVB-S2 standard, for example, for code rate $2/3$ as shown we have:

$$i = 1, \gamma_1 = 0.0000154$$
$$i = 2, \gamma_2 = 0.33331$$
$$i = 3, \gamma_3 = 0.60000$$
$$i = 13, \gamma_{13} = 0.06666$$

and

$$\gamma(X) = 0.0000154 + 0.33331X + 0.60000X^2 + 0.06666X^{12}.$$

In our design to achieve low complexity we choose $L = 720$ and for good performance we increase the block length to two times of DVB-S2 case. Therefore, we simplify the code by a factor 720 compared to random case. In this design, with same memory space for base addresses we encode and decode a higher block length. In this condition we have:

$$i = 1, \gamma_1 = 0.0000077$$
$$i = 2, \gamma_2 = 0.33332$$
$$i = 3, \gamma_3 = 0.53333$$
$$i = 13, \gamma_{13} = 0.13333$$

and

$$\gamma(X) = 0.0000077 + 0.33332X + 0.53333X^2 + 0.13333X^{12}$$

Simulation results show that we have better performance in a comparison with DVB-S2. It is obvious because we consider high degree weight for first selected variable nodes, and we increase the number of high degree variable nodes. If there are errors in these variable nodes, they are corrected quickly and then they help the other variable nodes to be corrected as well.

Assigning (1)'s for base addresses is the most important thing in parity check matrix designing. With the allocation and the rules explained before, parity check matrix will be distinguished. Base addresses are selected randomly to satisfy all of the above rules for better performance and low complexity. The degree distribution of the new parity check matrix is given in table 3. 3.

Table 3.2: Degree distribution in the LDPC code of DVB-S2

Degree / Rate	1	2	3	4	8	11	12	13
1/4	0.0000154	0.749	0.166666				0.083333	
1/3	0.0000154	0.666651	0.222222				0.111111	
1/2	0.0000154	0.499998	0.3		0.2			
3/5	0.0000154	0.399998	0.4				0.2	
2/3	0.0000154	0.333317	0.6					0.066666
3/4	0.0000154	0.249984	0.666666				0.083333	
4/5	0.0000154	0.199984	0.7			0.1		
5/6	0.0000154	0.166651	0.75					0.083333
8/9	0.0000154	0.111095	0.777777	0.111111				
9/10	0.0000154	0.099984	0.8	0.1				

Table 3.3: Degree distribution in the new LDPC code

Degree / Rate	1	2	3	4	8	11	12	13
1/4	0.0000077	0.7490154	0.166666				0.083333	
1/3	0.0000077	0.6666587	0.222222				0.111111	
1/2	0.0000077	0.49999877	0.3		0.2			
3/5	0.0000077	0.4000057	0.4				0.2	
2/3	0.0000077	0.3333247	0.6					0.066666
3/4	0.0000077	0.2499917	0.666666				0.083333	
4/5	0.0000077	0.1999917	0.7			0.1		
5/6	0.0000077	0.1666587	0.75					0.083333
8/9	0.0000077	0.1111027	0.777777	0.111111				
9/10	0.0000077	0.0999917	0.8	0.1				

Chapter 4

Simulation Results

4.1 Simulation Procedure

This chapter presents simulation results of the irregular random LDPC code with coded block size equal to 64800 bits used in DVB-S2 system and the new LDPC codes that have been developed in this thesis.

The block diagrams in Fig 4.1 shows the procedure for simulation of LDPC code. As far as the transmission is concerned, a BPSK modulation has been assumed and signal propagates over an AWGN channel. Codes with different coding rates are present in this chapter and their performances are reported for comparison. In the simulation, message bits are randomly produced depending on the code rate and are encoded with the LDPC encoder. Then, the coded bits are modulated by a BPSK modulator. The AWGN function generates Additive White Gaussian Noise which is added to the modulated signal. At the receiver side after BPSK demodulation, the received symbols are decoded by the LDPC decoder and then the decoded bits are compared with the message bits in order to find the number of errors. To have a good result, the simulation has to be repeated for many numbers of different message blocks. However, this results in a higher processing time.

The LDPC coding scheme in study is associated with the specific parity check matrix previously discussed. For LDPC decoding, the message passing technique is used which is known as "belief propagation". The decoding technique has a major role in the LDPC codes performance. Therefore, the LDPC decoding technique which is used in simulation should be described.

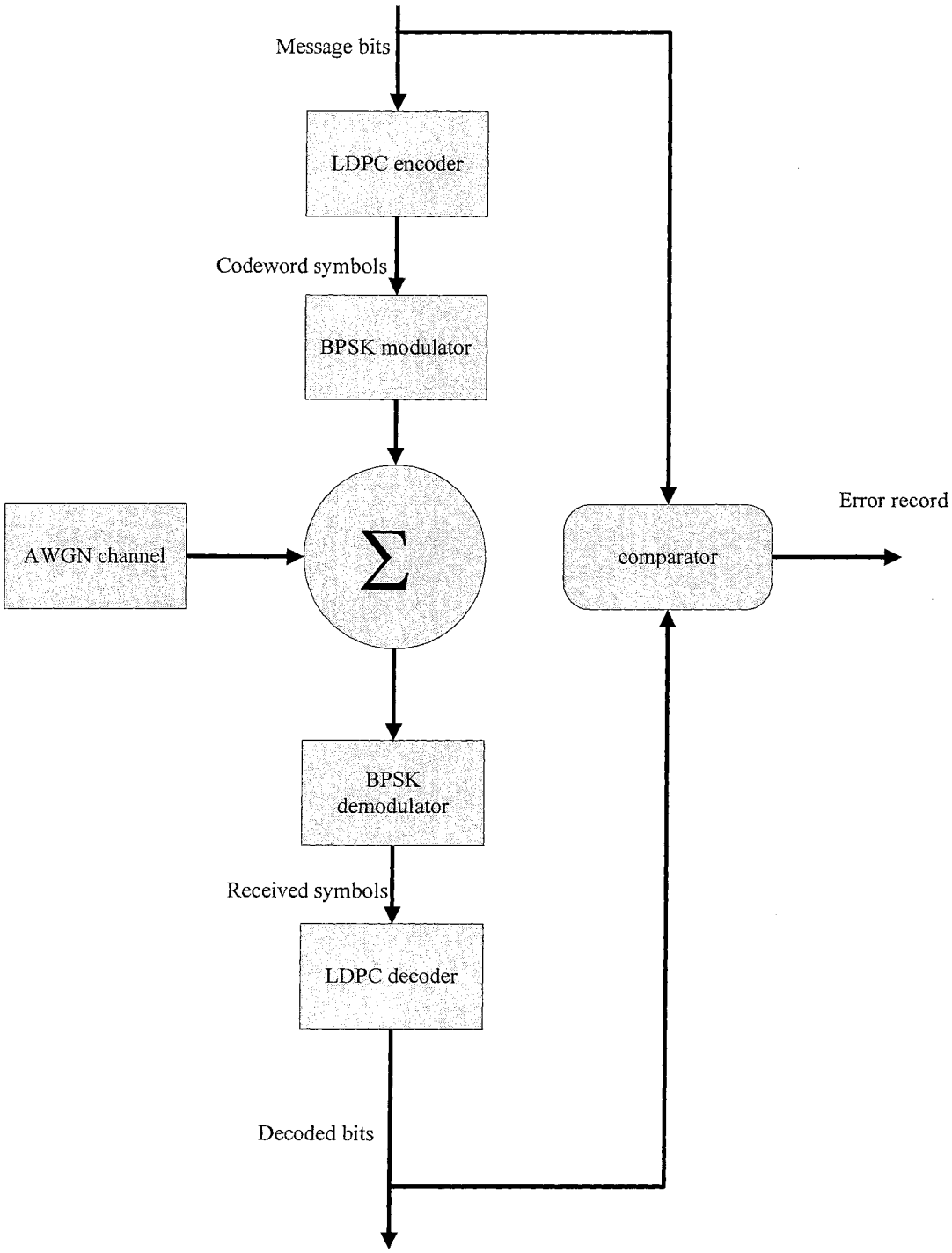


Figure 4.1: Block diagram for simulating LDPC code

4.2 Maximum a posteriori Decoding

For minimizing bit error rate of the decoded sequence, detection of LDPC is performed by applying the maximum a posteriori (MAP) algorithm [1]. Usually, in order to reduce complexity the log version of the MAP algorithm (log-MAP) is adopted. The MAP algorithm iteratively estimates the a posteriori probabilities (APP) $p_r\{b_n \mid r\}$, where $b_n (n = 0,1,\ldots\ldots\ldots\ldots, n_{LDPC} - 1)$ is the n th data bit and r is the received vector from the channel. For any bit in b, the following ratio is evaluated:

$$\lambda_{b_n} \doteq \log \frac{P_r\{b_n = 1 \mid r\}}{P_r\{b_n = 0 \mid r\}} \tag{4.1}$$

Expression (4.1) is the log of a posteriori probabilities ratio (log-APP) of data bit b_n and it is often known as the a posteriori Log Likelihood Ratio (LLR). At the outputs of the decoder the a posteriori LLR of information bit helps to estimate the information bits U. Information bits are u_k $(k = 0,1,\ldots\ldots k_{LDPC} - 1)$ and we have λ_{u_k} as a posteriori LLR of an information bit via maximum a posteriori rule:

$$\begin{cases} \lambda_{u_k} & \rangle & 0 & \Rightarrow & \hat{u}_k = 1 \\ \lambda_{u_k} & \langle & 0 & \Rightarrow & \hat{u}_k = 0 \end{cases} \tag{4.2}$$

($\hat{u}_k$ is the estimation of the information bit u_k) or in other words:

$$\hat{u} = sign\{\lambda_{u_k}\}. \tag{4.3}$$

For information bit probability consider that:

$$P_r\{u_k = 1\} = P_r\{u_k = 0\} = 1/2 \quad .$$

In the decoder r_n is the observation of the n th coded bit b_n in the received vector, and log- MAP of this bit node is:

$$\lambda_{b_n} = \log \frac{P_r\{b_n = 1 | r_n, (r_{i\neq n})\}}{P_r\{b_n = 0 | r_n, (r_{i\neq n})\}} . \tag{4.4}$$

Applying the Bays' theorem to (4.4), we have

$$\begin{aligned} \lambda_{b_n} &= \log \frac{P_r\{r_n, b_n = 1, (r_{i\neq n})\}}{f\{r_n, (r_{i\neq n})\}} \cdot \log \frac{f\{r_n, (r_{i\neq n})\}}{P_r\{r_n, b_n = 0, (r_{i\neq n})\}} \\ &= \log\left(\frac{P_r\{r_n | b_n = 1, (r_{i\neq n})\} \cdot f\{b_n = 1, (r_{i\neq n})\}}{f\{r_n | (r_{i\neq n})\} \cdot f\{(r_{i\neq n})\}} \cdot \frac{f\{r_n | (r_{i\neq n})\} \cdot f\{(r_{i\neq n})\}}{P_r\{r_n | b_n = 0, (r_{i\neq n})\} \cdot f\{b_n = 0, (r_{i\neq n})\}} \right) \end{aligned} \tag{4.5}$$

If observation of the received sample r_n is independent of the set of samples $(r_{i\neq n})$ we can simplify the last expression as

$$\lambda_{b_n} = \log\left(\frac{f\{r_n | b_n = 1\} \cdot P_r\{b_n = 1 | (r_{i\neq n})\}}{f\{r_n | b_n = 0\} \cdot P_r\{b_n = 0 | (r_{i\neq n})\}} \right) \tag{4.6}$$

or

$$\lambda_{b_n} = \log\left(\frac{f\{r_n | b_n = 1\}}{f\{r_n | b_n = 0\}} \right) + \log\left(\frac{P_r\{b_n = 1 | (r_{i\neq n})\}}{P_r\{b_n = 0 | (r_{i\neq n})\}} \right) . \tag{4.7}$$

The first term relates to the information based on channel information r_n known as intrinsic information. The second term is the part of overall log likelihood ratio λ_{b_n} which is based on the observation of the received samples except r_n and is called extrinsic information. The extrinsic information is produced by the decoder and depends on all the samples received from the channel excluding the sample entering the current variable (bit) node.

In our simulation we use Additive White Gaussian Noise (AWGN) channel with Binary Phase Shift Keying (BPSK) modulation. Note that intrinsic information relates to the modulation type and the channel used for this signal as shown in (4.11). For this type of systems assume that for 0 we send -1 and for 1 we send +1 and intrinsic part can be written as:

$$z_n = (2b_n - 1) \tag{4.8}$$

z_n is the nth transmitted symbol for nth coded bit b_n. At the receiver the received symbol is:

$$r_n = (2b_n - 1) + n_n \tag{4.9}$$

This is a normal random variable with zero mean and variance σ^2 and by definition conditional Probability Density Function (PDF) is expressed as

$$f(r_n|b_n) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(r_n - b_n)^2}{2\sigma^2}} \tag{4.10}$$

and intrinsic information for bit node b_n can be computed:

$$\log \frac{f\{r_n|b_n = 1\}}{f\{r_n|b_n = 0\}} = \log \frac{e^{-\frac{(r_n-1)^2}{2\sigma^2}}}{e^{-\frac{(r_n+1)^2}{2\sigma^2}}} = \frac{(r_n+1)^2}{2\sigma^2} - \frac{(r_n-1)^2}{2\sigma^2} = \frac{2r_n}{\sigma^2} \tag{4.11}$$

and consequently for bit node b_n equation (4.7) becomes:

$$\lambda_{b_n} = \frac{2r_n}{\sigma^2} + \log \frac{P_r\{b_n = 1|(r_{i\neq n})\}}{P_r\{b_n = 0|(r_{i\neq n})\}} \quad . \tag{4.12}$$

For BPSK modulation with signal to noise ratio E_b/N_o and AWGN channel noise the power σ^2 is characterized as:

$$\sigma^2 = \frac{1}{2\eta_M R \frac{E_b}{N_o}} \tag{4.13}$$

where R is the code rate, η_M is the modulation efficiency (for BPSK modulation $\eta_M = 1\ bit/symbol$). Therefore, Equation (4.12) becomes:

$$\lambda_{b_n} = 4r_n R \frac{E_b}{N_o} + \log \frac{P_r\{b_n = 1 | (r_{i \neq n})\}}{P_r\{b_n = 0 | (r_{i \neq n})\}} \tag{4.14}$$

To find variable and check node update rules and for better description assume that LDPC code is a regular code. Tanner graph of a regular LDPC code for n th bit node is shown in Fig 4.2. As we see, M and L are bit and check node degrees respectively. s_m is the set of bit nodes connected to the check node c_m and equals to:

$$s_m = d_m + b_n \tag{4.15}$$

from the even parity check rule for every parity check nodes:

$$\Phi(s_m) = 0 \tag{4.16}$$

with substitution (4.15):

$$\Phi(d_m) + b_n = 0 \tag{4.17}$$

and consequently:

$$b_n = \Phi(d_m), \qquad m = 0,1,2,\ldots\ldots\ldots, M-1 \tag{4.18}$$

From the last expression, it is obvious that for bit node b_n, M constraints must be satisfied.

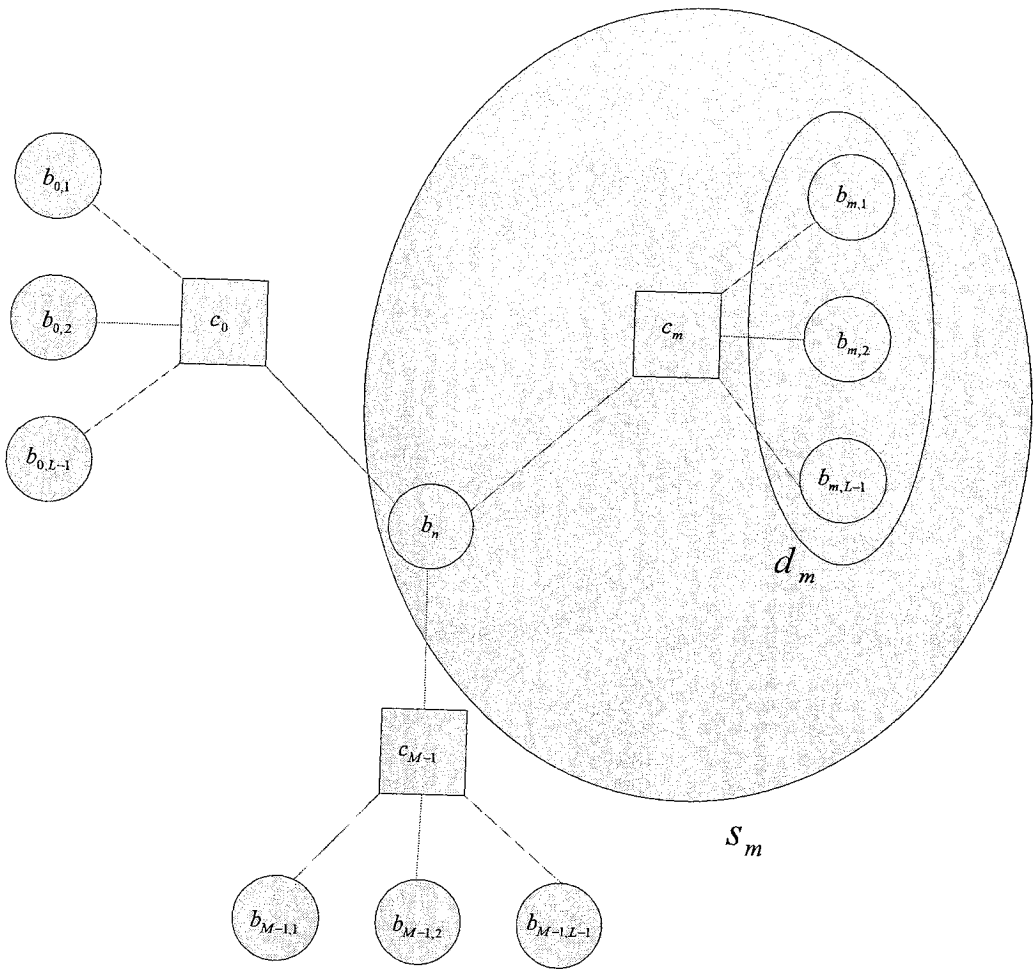


Figure 4.2: Tanner graph of regular LDPC code for nth bit node

4.3 Variable Node Processing

By using even parity check function and equation (4.12), extrinsic LLR becomes:

$$\lambda_{b_n,extr} \doteq \log \frac{P_r\{b_n = 1 | (r_{i \neq n})\}}{P_r\{b_n = 0 | (r_{i \neq n})\}} \tag{4.19}$$

using (4.18) we can write,

$$\lambda_{b_n,extr} = \log \frac{P_r\{\Phi(d_m)=1, m=0,1,.....,M-1|(r_{i\neq n})\}}{P_r\{\Phi(d_m)=0, m=0,1,.....,M-1|(r_{i\neq n})\}} \tag{4.20}$$

Assume that the Tanner graph has no cycle and therefore the joint probabilities can split to:

$$\lambda_{b_n,extr} = \log \frac{\prod_{m=0}^{M-1} P_r\{\Phi(d_m)=1|(r_{i\neq n})\}}{\prod_{m=0}^{M-1} P_r\{\Phi(d_m)=0|(r_{i\neq n})\}} \tag{4.21}$$

or

$$\lambda_{b_n,extr} = \sum_{m=0}^{M-1} \log \frac{P_r\{\Phi(d_m)=1|(r_{i\neq n})\}}{P_r\{\Phi(d_m)=0|(r_{i\neq n})\}} . \tag{4.22}$$

Equation (4.22) shows that extrinsic information at bit node b_n is computed as the sum over the constraint nodes linked to the same bit node. For bit node b_n, the extrinsic pieces of information computed at check node c_m can be expressed as

$$\lambda_{c_m,b_n} \doteq \log \frac{P_r\{\Phi(d_m)=1|(r_{i\neq n})\}}{P_r\{\Phi(d_m)=0|(r_{i\neq n})\}} \tag{4.23}$$

Equation (4.23) shows that the check node processor must compute the even parity of the set of incoming messages excluding the one coming from the bit node b_n (set d_m in Figure 4.2). Equation (4.22) can be shown as

$$\lambda_{b_n,extr} = \sum_{m=0}^{M-1} \lambda_{c_m,b_n} \tag{4.24}$$

and finally

$$\lambda_{b_n} = \frac{2r_n}{\sigma^2} + \sum_{m=0}^{M-1} \lambda_{c_m,b_n} \tag{4.25}$$

4.4 Check Node Processing

To find the check node update rule, first Tanh Rule is defined as follows. For a set of independent, non-equiprobable bits $\chi = x_0, x_1, ..., x_{N-1}$ the following relation holds:

$$\tanh\left(-\frac{\lambda_y}{2}\right) = \prod_{i=0}^{N-1} \tanh\left(-\frac{\lambda_i}{2}\right) \tag{4.26}$$

where λ_i is the a priori LLR of x_i ,

$$\lambda_i \doteq \log \frac{P_r\{x_i = 1\}}{P_r\{x_i = 0\}} \tag{4.27}$$

and $y = \phi(\chi)$ is the even parity of vector χ , and similarly express as,

$$\lambda_y \doteq \log \frac{P_r\{y = 1\}}{P_r\{y = 0\}} \tag{4.28}$$

By applying Tanh Rule to $\lambda_{c_m,\, b_n}$:

$$\tanh\left(-\frac{\lambda_{c_m,b_n}}{2}\right) = \prod_{l=1}^{L-1} \tanh\left(-\frac{\lambda_{b_m,l,c_m}}{2}\right) \tag{4.29}$$

where index l in the set d_m shows bit nodes connected to check node c_m excluding $b_{m,0} = b_n (l \neq 0)$. By solving equation (4.29), the check node update rule will be:

$$\lambda_{c_m,b_n} = -2\tanh^{-1}\left\{\prod_{l=1}^{L-1} \tanh\left(-\frac{\lambda_{b_m,l,c_m}}{2}\right)\right\} \tag{4.30}$$

In Fig (4.3), all of these procedures are represented. This procedure is simply computed by summing the channel observation $\frac{2r_n}{\sigma^2}$ with the extrinsic information $\lambda_{b_n,extr}$ based on

the observation of every received sample except r_n. The decoding of LDPC is simply regarded as the exchange of messages along the edges of the graph, which explains the name 'message passing'. Check node c_m collects $L-1$ messages from all the connected bits, computes the a posteriori LLR $\lambda_{b_n,extr}$ of the parity and sends updated message to bit node b_n. Finally incoming messages summed together according to (4.25) and hard detector receives the outputs of LLR and gives the final estimates of b_n according to (4.2).

4.5 Iterative Decoding

As expressed before, reliability of the messages exchanged along the edges of the graph increases iteration by iteration. After a certain number of messages pass, the iterative message passing algorithm converges on the true a posteriori Log Likelihood ratios defined in (4.7). For this reason, the message passing algorithm is not exact and approximately represents the maximum a posteriori decoding.

Message leaving the bit node toward the check node (λ_{b_n,c_m}) is the sum of all messages entering the same bit node b_n including the channel observation, except the message from the check node c_m. From (4.25):

$$\lambda_{b_n} = \frac{2r_n}{\sigma^2} + \sum_{i=0,i\neq m}^{M-1} \lambda_{c_i,b_n} + \lambda_{c_m,b_n} = \lambda_{b_n,c_m} + \lambda_{c_m,b_n} \tag{4.31}$$

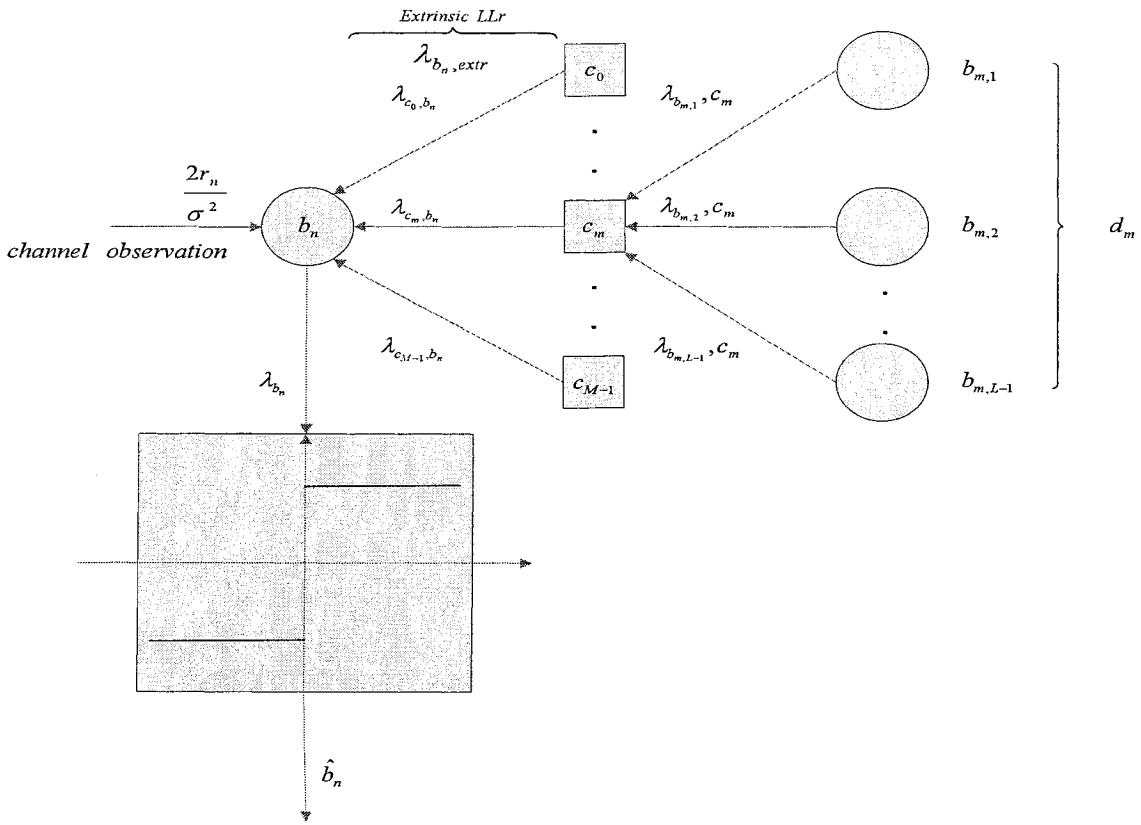


Figure 4.3: Message passing on the Tanner graph

At first decoding iteration, all messages from check nodes are zero and there is no information in check nodes. For this reason, to start iterative decoding, the decoder needs to be initialized.

Let M_n be the set of check nodes to be presented to bit node $b_n, n = 0,1,2,.....,n_{LDPC}-1$ and N_m be the set of bit nodes which checked at $c_m, m = 0,1,.....,n_{LDPC}-k_{LDPC}-1$. Referring to matrix H, M_n is the set of '1's in column n and N_m is the set of '1's in row m which specify the parity check matrix. The initial condition for the decoding is

$$\lambda^{(0)}{}_{c_m,b_n} = 0, \qquad m = 0,1,.....,n_{LDPC}-k_{LDPC}-1,\ n \in N_m \tag{4.32}$$

At the first iteration the a posteriori LLRs of bit nodes are observed from channel as

$$\lambda^{(0)}{}_{b_n} = \frac{2r_n}{\sigma^2}, \qquad n = 0,1,\ldots\ldots,n_{LDPC}-1 \tag{4.32}$$

Then, the messages are sent from bit nodes to check nodes as

$$\lambda^{(0)}{}_{b_n,c_m} = \frac{2r_n}{\sigma^2}, \qquad n = 0,1,\ldots\ldots,n_{LDPC}-1, m \in M_n \tag{4.34}$$

At each iteration check node processors update outgoing messages according to (4.30). Check node processors collect messages from the bit nodes which produced at the previous iteration. Finally for *lth* iteration, check nodes update rule becomes:

$$\lambda^{(l)}{}_{c_m,b_n} = -2\tanh^{-1}\left[\prod_{i\in N_m, i\neq n} \tanh\left(-\frac{\lambda^{(l-1)}{}_{b_i,c_m}}{2}\right)\right] \tag{4.35}$$
$$n = 0,1,2,\ldots\ldots\ldots,n_{LDPC}-k_{LDPC}-1, n \in N_m$$

Bit node processors update their outgoing messages according to (4.25). The messages passed by check nodes at the same iteration and the message generated by channel are used. Finally for iteration *lth* , bit nodes update rule becomes:

$$\lambda^{(l)}{}_{b_n,c_m} = \frac{2r_n}{\sigma^2} + \sum_{i\in M_n, i\neq m} \lambda^{(l)}{}_{c_m,b_n} \tag{4.36}$$
$$n = 0,1,\ldots\ldots,n_{LDPC}-1, \quad m \in M$$

Output LLRS are updated in a similar way and output update rule becomes:

$$\lambda^{(l)}{}_{b_n} = \frac{2r_n}{\sigma^2} + \sum_{i \in M_n} \lambda^{(l)}{}_{c_m,b_n} = \lambda^{(l)}{}_{c_m,b_n} + \lambda^{(l)}{}_{b_n,c_m}$$
$$n = 0,1,\ldots\ldots, n_{LDPC} - 1, \quad m \in M_n \tag{4.37}$$

Equation (4.37) is the last expression for bit node b_n evaluation and after this step we decide about bit node b_n in this iteration.

Decoding normally runs for fix a number of iterations (50 for DVB-S2) to reach a good performance. The correct codeword is normally detected before the end of iterations. The best way to check the correctness of the codeword is to re-encode it. Another way to determine if the codeword is right or not, is to check the set of equations for each check node and make sure they are satisfied. If these constraints are satisfied at each check node, the valid codeword is detected and the decoder stops iteration, otherwise the decoder continues to reach the valid codeword. However, sometimes after maximum iteration the received word is not corrected. In this situation in order to avoid infinite loop the decoding iteration must stop and the decoder fails to correct the received word.

As an illustrative example, assume the LDPC code with parity check matrix and Tanner graph shown in Fig 4.4. From parity check matrix, the codeword block is 6 and information bits are 3 and consequently $R = 1/2$. Let codeword (0 1 1 0 1 1) which satisfies equation (2.9) be sent and signal to noise ratio in the channel be $E_b/N_o = 6.8$ dB. With these parameters and under AWGN channel condition:

$$\sigma^2 = \frac{1}{2\eta_M R \frac{E_b}{N_o}} = 0.2089.$$

Now consider that the received vector r is:

$$r = (\,0.0555\,,1.2398\,,\,0.9946\,,-0.5826\,,1.0256\,,\,0.4940\,).$$

The decoding process starts by (4.11) as

$$\lambda_{b_n,\mathrm{int}\,r} = \frac{2r}{\sigma^2} = (\,0.5317\,,11.8680\,,9.5210\,,-5.5771\,,9.8174\,,4.7286\,)$$

If hard decision detector is used and the decoded block is estimated, there is one error in bit number 0. For managing the decoding algorithm described before, we number the edges in Fig 4.4. As depicted in the figure, numbering from bit nodes view is (1,2,3,4,5,6,7,8,9) and from check nodes view is (1,5,7,2,3,8,4,,6,9). With this type of numbering in each bit node processing, edges are read and written in bit node view and on the contrary in each check node processing edges are considered from check nodes view. Edge numbering is reported in Table 4.1.

For initialization, all messages from check nodes are zero:

$$\lambda^{(0)}{}_{c_m,b_n} = 0$$

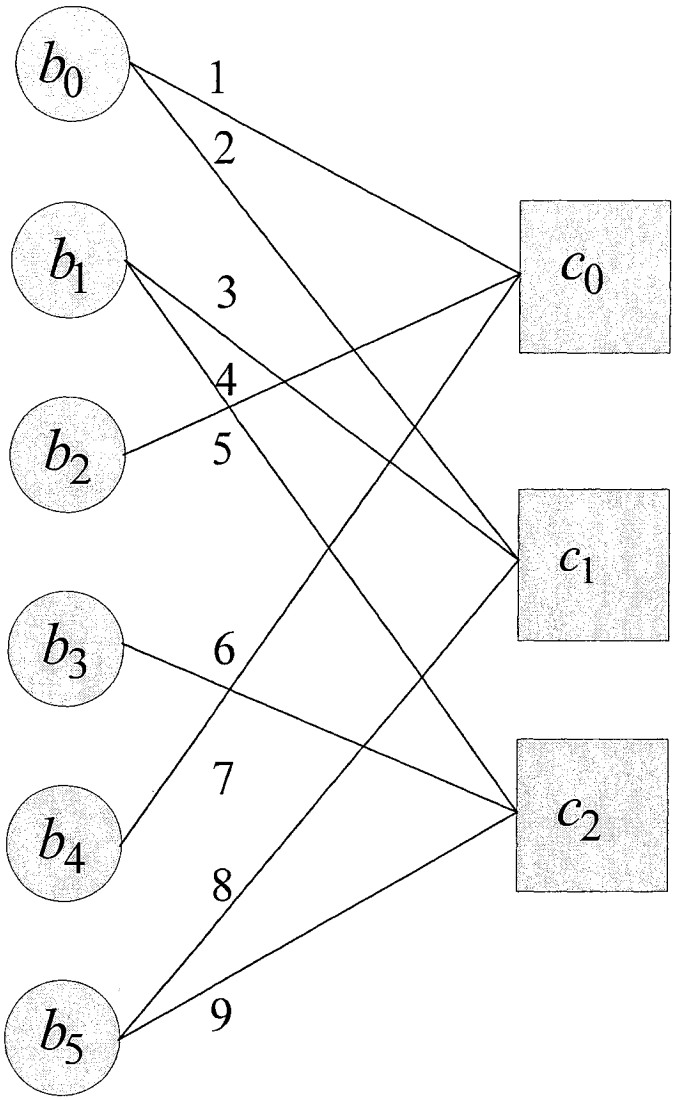


Figure 4.4: Edge numbering on the Tanner graph

Table 4.1: Edge numbering from the bit node view

Edge No	*Bit Node*	*Check Node*
1	b_0	c_0
2	b_0	c_1
3	b_1	c_1
4	b_1	c_2
5	b_2	c_0
6	b_3	c_2
7	b_4	c_0
8	b_5	c_1
9	b_5	c_2

and

$$\lambda^{(0)}{}_{b_n,c_m} = \lambda_{b_n,\text{int}\,rinsic} = (0.5317, 11.8680, 9.5210, -5.5771, 9.8174, 4.7286).$$

Check node processes the received messages and sends back updated messages towards bit node processor according to (4.35). for check node zero

$$\lambda^{(1)}{}_{c_0,b_0} = -2\tanh^{-1}\left[\tanh\left(-\frac{\lambda^{(0)}{}_{b_2,c_0}}{2}\right)\tanh\left(-\frac{\lambda^{(0)}{}_{b_4,c_0}}{2}\right)\right]$$

$$= -2\tanh^{-1}\left[\tanh\left(-\frac{9.5210}{2}\right)\tanh\left(-\frac{9.8174}{2}\right)\right] = -8.96511118$$

or for check node 2

$$\lambda^{(1)}{}_{c_2,b_5} = -2\tanh^{-1}\left[\tanh\left(-\frac{\lambda^{(0)}{}_{b_1,c_5}}{2}\right)\tanh\left(-\frac{\lambda^{(0)}{}_{b_3,c_5}}{2}\right)\right]$$

$$= -2\tanh^{-1}\left[\tanh\left(-\frac{11.8680}{2}\right)\tanh\left(-\frac{-5.5771}{2}\right)\right] = 5.57524865.$$

For the other messages from check nodes to bit nodes, computing are done in the same manner which is described and are reported in Fig 4.5 and Table 4.2.

Bit node processor processes information received from check nodes and sums them with channel observation according to (4.36) and updates messages toward check nodes. For example for bit node zero,

$$\lambda^{(1)}{}_{b_o,c_0} = \frac{2r_0}{\sigma^2} + \sum_{i\in M_n, i\neq m} \lambda^{(1)}{}_{c_m,b_n}$$

$$= 0.5317 + \lambda^{(1)}{}_{c_1,b_0} = 0.5317 + (-4.7278) = -4.1961,$$

or for bit node 5

$$\lambda^{(1)}{}_{b_5,c_2} = \frac{2r_5}{\sigma^2} + \sum_{i\in M_n, i\neq m} \lambda^{(1)}{}_{c_m,b_n}$$
$$= 4.7286 + \lambda^{(1)}{}_{c_1,b_5} = 4.7286 + (-0.5317) = 4.1969.$$

The complete list of messages is reported in Table 4.2.

For all iterations when messages are changed, the outputs must be updated according to (4.37). For bit number zero in second iteration:

$$\lambda^{(1)}{}_{b_0} = \frac{2r_0}{\sigma^2} + \sum_{i\in M_n} \lambda^{(1)}{}_{c_m,b_n} = 0.5317 + \lambda^{(1)}{}_{c_0,b_0} + \lambda^{(1)}{}_{c_1,b_0}$$
$$= 0.5317 + (-8.9651) + (-4.7278) = -13.1612$$

after hard detector

$$\hat{b}_n{}^{(1)} = 0$$

in the second iteration this algorithm corrects the failure bit.

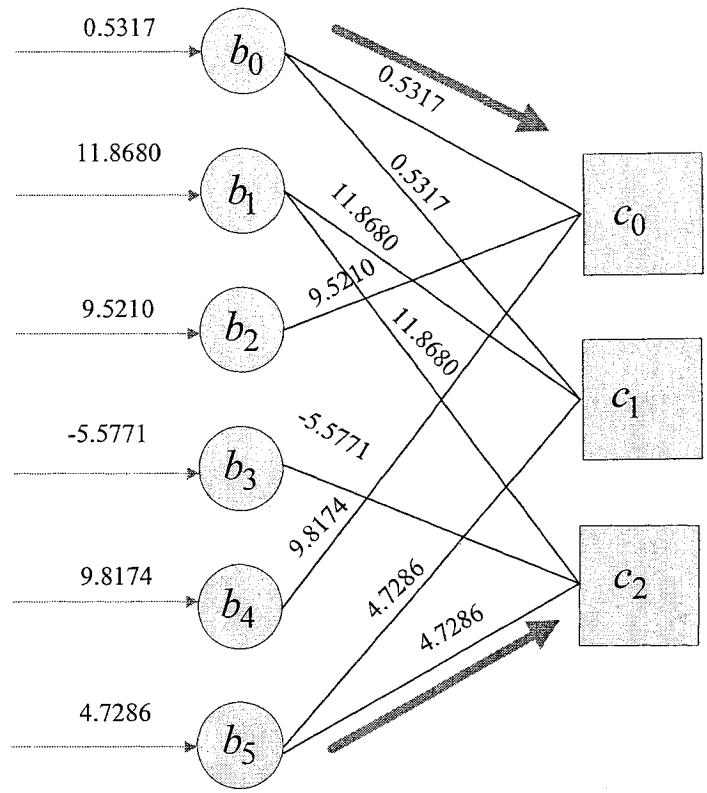
or for bit number 5

$$\lambda^{(1)}{}_{b_5} = \frac{2r_5}{\sigma^2} + \sum_{i\in M_n} \lambda^{(1)}{}_{c_m,b_5} = 4.7286 + \lambda^{(1)}{}_{c_1,b_5} + \lambda^{(1)}{}_{c_2,b_5}$$
$$= 4.7286 + (-0.5317) + (5.5752) = 9.7721$$

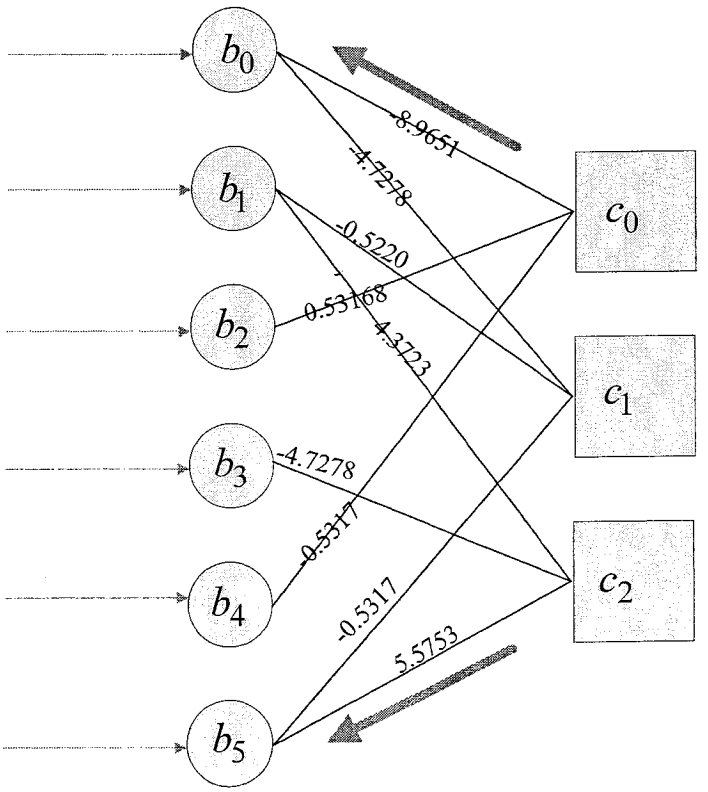
after hard detector:

$$\hat{b}_n{}^{(1)} = 1$$

b_5 is correct same as the last iteration. Table 4.2 shows the messages exchanged along Tanner graph during the first three iterations and Table 4.3 shows soft and hard outputs of the decoder.



(a) Bit node update



(b) Check node update

Figure 4.5: Messages passing at first iteration

Table 4.2: Exchange messages between bit and check nodes during first three iterations

Edge No	$\lambda^{(0)}{}_{c_m,b_n}$	$\lambda^{(0)}{}_{b_n,c_m}$	$\lambda^{(1)}{}_{c_m,b_n}$	$\lambda^{(1)}{}_{b_n,c_m}$	$\lambda^{(2)}{}_{c_m,b_n}$	$\lambda^{(2)}{}_{b_n,c_m}$
1	0.0	0.53174	-8.9651	-4.1961	-8.9651	-9.7695
2	0.0	0.53174	-4.7278	-8.4334	-10.3012	-8.4334
3	0.0	11.8680	-0.5220	16.2403	8.2901	15.8406
4	0.0	11.8680	4.3723	11.3460	3.9726	20.1581
5	0.0	9.5210	-0.5317	9.5210	4.1925	9.5210
6	0.0	-5.5771	-4.7278	-5.5771	-4.1961	-5.5771
7	0.0	9.8174	-0.5317	9.8174	4.1912	9.8174
8	0.0	4.7286	-0.5317	10.3038	8.4330	10.3026
9	0.0	4.7286	5.5752	4.1969	5.5740	13.1616

Table 4.3: Soft and hard decision outputs

Bit No	$\lambda^{(0)}{}_{b_n}$	$\hat{b}_n{}^{(0)}$	$\lambda^{(1)}{}_{b_n}$	$\hat{b}_n{}^{(1)}$	$\lambda^{(2)}{}_{b_n}$	$\hat{b}_n{}^{(2)}$
0	0.5317	1	-13.1612	0	-18.7346	0
1	11.8680	1	15.7183	1	24.1`307	1
2	9.5210	1	8.9893	1	13.7135	1
3	-5.5771	0	-10.3049	0	-9.7732	0
4	9.8174	1	9.2875	1	14.0086	1
5	4.7286	1	9.7721	1	18.7335	1

Tables 4.2-4.3 show that the message passing algorithm can correctly recover the erroneously received codeword after the second iteration. The next iterations will improve the reliability by increasing the magnitudes of soft decisions. Given that the set of equations are satisfied in iteration, the decoding process will be stopped.

In this chapter, after discussing about LDPC code simulation, some simulation results will be presented which are related to irregular LDPC codes with new parity check matrix. Irregular LDPC codes applied in DVB-S2 are used as a benchmark to compare new types of LDPC codes.

Figure 4.6 compares the performance of two types of LDPC code with $R = 1/3$. Shannon limit for this code rate is -0.507 dB, while the new LDPC code is about 0.8 dB away from Shannon limit, as shown in Figure 4.6. With these results in Figure 6, it is possible to compare new LDPC code with the one used in DVB-S2. As shown, error performance has been improved about 0.1 dB, which can be considered as a noticeable improvement in the system performance. Moreover, the results in Figure 4.6 show that the error performance curves are sharp enough to highly decrease the BER, by increasing 0.2 dB to the value of E_b / N_o.

Figure 4.7 shows the performance of other LDPC codes with the code rate of R=4/5. For this code rate and for BPSK modulation, the Shannon limit in AWGN channel is 2.045 dB. The results in Figure 4.7 show that the considered code is just 0.2 dB away from Shannon limit. Moreover, it is shown that in this case we have 0.2 dB performance improvements over DVB-S2 LDPC code. Other Figures show the simulation results for two types LDPC codes and compare the performances.

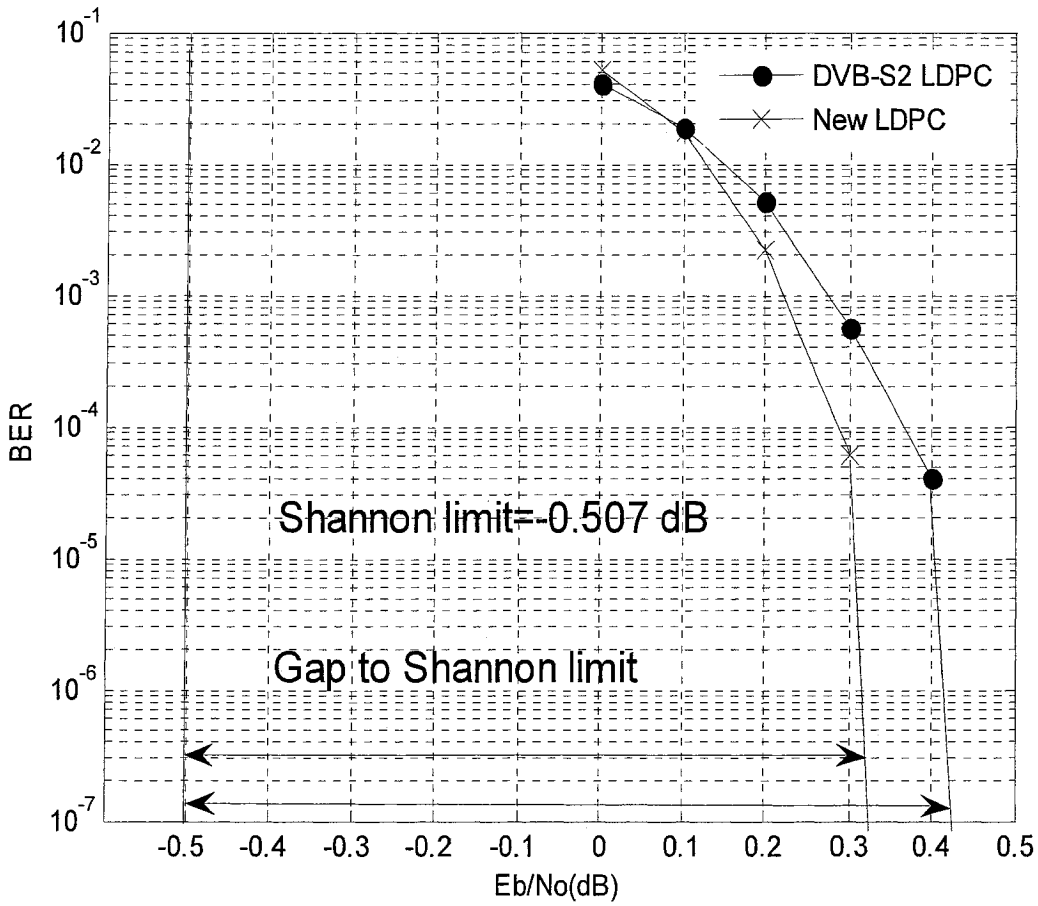


Figure 4.6: Error performance of LDPC codes with R=1/3

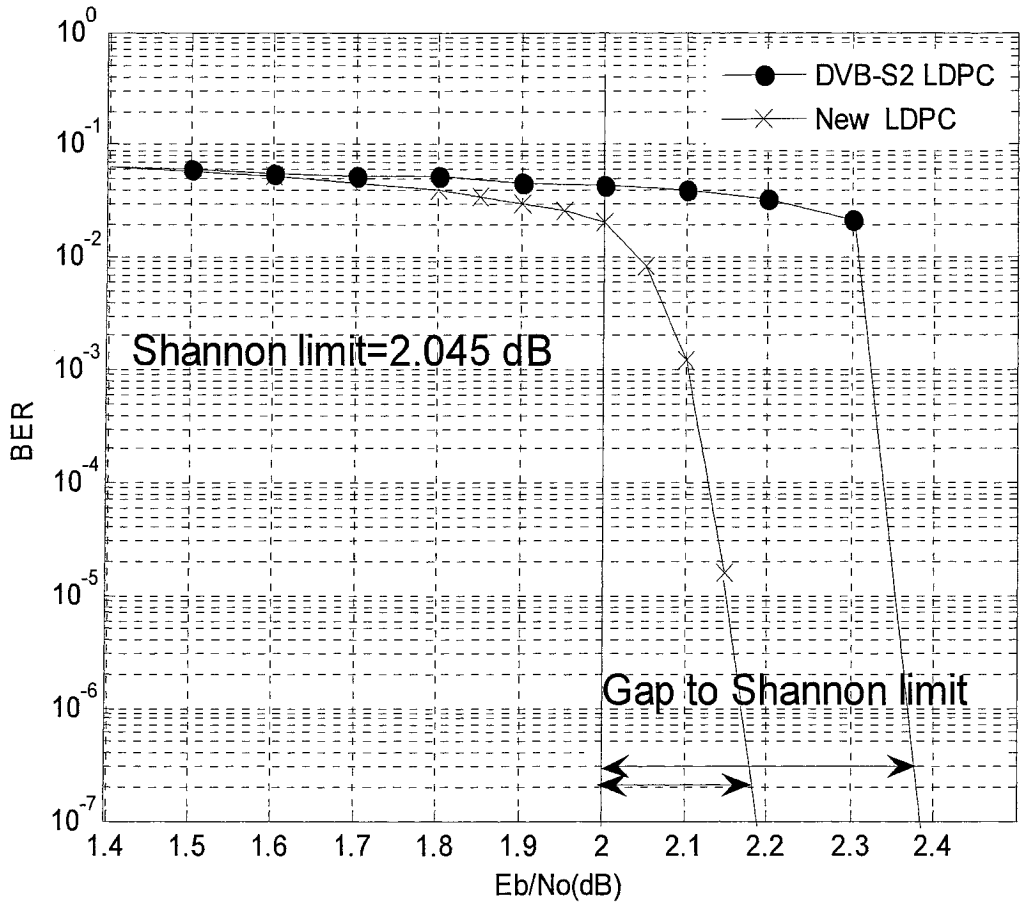


Figure 4.7: Error performance of LDPC codes with R=4/5

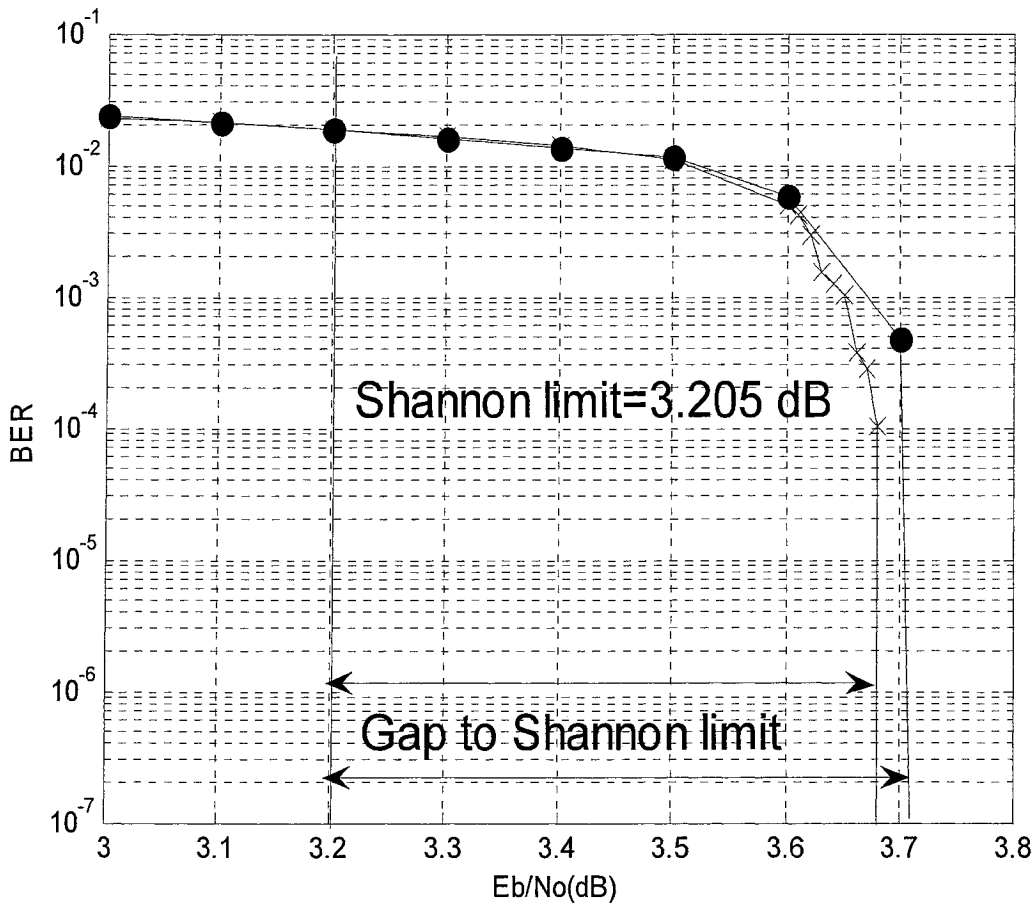


Figure 4.8: Error performance of LDPC codes with R=9/10

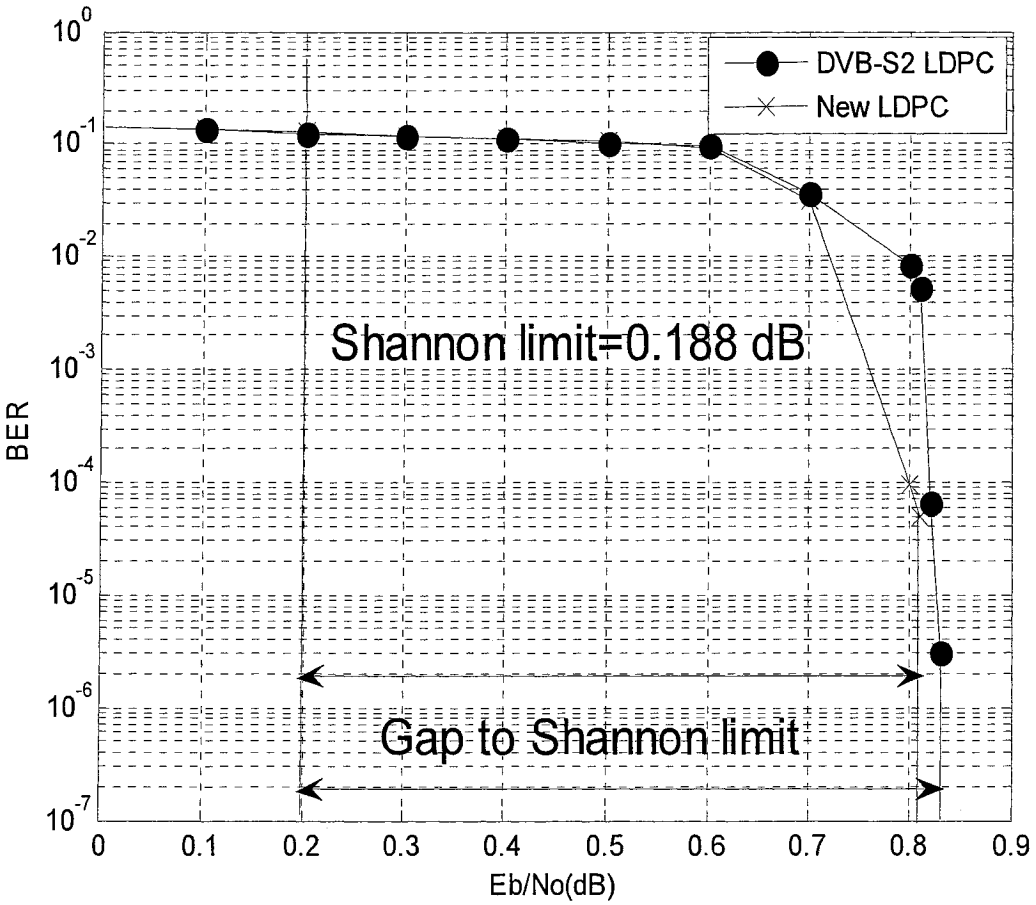


Figure 4.9: Error performance of LDPC codes with R=1/2

Chapter 5

Conclusion

5.1 Summary of Contributions

In this chapter, we briefly review the results of this thesis and then suggest some future work which can be considered as the immediate extension of this work.

Recently, low density parity check codes have been under high interest in the Forward Error Correcting field. In this thesis, a background on LDPC has been presented which includes designing, encoding and decoding of the LDPC codes, as well as their applications in DVB-S2 standard. LDPC codes used in DVB-S2 approach Shannon limits to within 0.7 dB for a wide range of throughput.

The core work done in this thesis is to design a huge sparse parity check matrix to be used in the LDPC encoder and decoder of DVB-S2 system. We used the density evolution method for the analysis and design of the parity check matrix with the special weight distributions.

We verified that message passing iterative decoding, offers very good performance, compared to other decoding types.

For encoding, we used a parity check matrix. In this method, staircase lower triangular sub-matrix of the parity check matrix has been chosen. With this type of parity check matrix the problem of deriving a generator matrix has been eliminated. This method provides the advantage of having linear encoding complexity.

We showed that by increasing the codeword block length, it is possible to move closer to the Shannon limit criteria. In this situation the main problem is to alleviate the complexity associated with encoding and decoding.

We showed that irregular LDPC codes have better performance compared to regular ones, but the main problem here is the derivation complexity of the parity check matrix. In this thesis we showed that it is possible to have an irregular parity check matrix in a regular manner, which reduces the complexity and also makes it possible to have a straightforward encoding process.

5.2 Possible Future Work

In continuation of this thesis, there are a number of problems that can be the subject of the future research area.

Implementation complexity was the problem which prevented using LDPC codes for approximately three decades. This complexity was mainly due to the huge parity check matrix. Therefore, another interesting future work is trying to find a good rule to map the space of (1)'s in the parity check matrix, which will reduce the necessary memory space.

In some applications we need to use short block length with high rate LDPC codes. As another future research area, we suggest finding a parity check matrix with simple encoding, decoding, and good performance for this special case.

Generally, in some cases, it is possible to decompose a huge matrix to some small blocks, called the submatrices. It would be interesting to investigate the possibility of

evaluating a rule, or sets of rules, in order to design the parity check matrix with the help of some previously defined submatrices. Clearly, with the help of this matrix decomposition, the complexity and the necessary memory space for decoding and encoding process will drastically decrease.

References

[1] John G. Proakis, *Digital Communications.* McGraw-Hill, 4th edition, 2000.

[2] A. Glavieux C. Berrou and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo-Codes," *Proceedings of International Communications Conference (ICC'93),* pages 1064-1070, 1993.

[3] A. Viterbi, "Orthogonal tree codes for communication in the presence of white gaussian noise," *IEEE Transactions on Communications,* pp. 238 -242, Apr 1967.

[4] C. E. Shannon, "A mathematical theory of communication," *Bell system Technical Journal, vol. 27, pp. 379-423, 1948.*

[5] S.B. Wicker, *Error Control Systems for Digital Communication and Storage.* Prentice Hall, New Jersey, 1995.

[6] R. Gallager, *Low-Density Parity-Check Codes.* PhD thesis, Massachusetts Institute of Technology, 1963.

[7] D.J.C. MacKay and R.M. Neal, "Near shannon-limit performance of low-density parity-check codes," *Electronics Lett.,* 32, pp.1645-1646, August 1996.

[8] D.J.C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Transaction on Information Theory, pp. 399-431, Mar 1999.*

[9] S.-Y. Chung, G.D. Forney, T.J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0:0045 dB of the shannon limit," *IEEE Commun. Lett.,* pp. 58-60, February 2001.

[10] L.N. Lee, "LDPC Code, Application to the Next Generation Wireless Communication Systems," 2003. Fall VTC, Panel Presentation by Hughes Network.

[11] R.M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory, pp. 533-547, Sep 1981.*

[12] A. Shokrollahi T.J. Richardson and R. Urbank, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Trans. Inform. Theory, vol. 47, pp. 619-637, Feb 2001.*

[13] H.Jin, A. Khandekar, and R. Mceliece, "Irregular Repeat Accumulate codes," *In proc. 2 nd International Symposium on Turbo Codes & Related Topics, Pages 1-8 Brest, France, Sept 2000.*

[14] European Telecommunications Standards Institute. Digital Video Broadcasting (DVB); second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2); EN 302 307, 2004.

[15] European Telecommunications standards Institute. Digital Video Broadcasting (DVB); Framing structure, Channel coding and modulation for 11/12 GHZ Satellite Services; EN 300 421, 1994.

[16] Shu Lin and Daniel J.Castello,Jr, *Error Control Coding*. Prentice Hall, Second Edition, 2004.

[17] Li Ping, W.K. Leung, and Nam Phamdo, "Low density parity check codes with semi-random parity check matrix," *Electronics Lett.,* vol. 35, pp. 38-39, January 1999.

[18] T.J. Richardson and R.L. Urbanke, "Efficient encoding of low density parity check codes," *IEEE Trans. Inform Theory,* vol. 47, pp. 638-656, February 2001.

[19] R. Echard and S. Chang., "The π-rotation low density parity check codes," *Proc IEEE GLOBECOM 2001,* pp. 980-984, 2001.

[20] Hughes Network System. DVB-S2 coding standard proposal, Technical report, DVB Standardization Committee, January 2003.

[21] T. J. Richardson and R. L. Urbanke, "The capacity of low density parity check codes under message-passing decoding," *IEEE Trans. Inform. Theory, vol* 47, pp. 599-628, February 2001.

[22] Frank Kienle,Torben Brack and Norbert When, " A Synthesizable IP Core for DVB-S2 Code Decoding," *Design, Automation and Test in Europe, pp. 100-105, 2005.*

[23] Mustafa Eroz , Feng-Wen Sun and Lin-Nan Lee, "DVB-S2 low density parity check code with near Shannon limit performance," *International Journal of Satellite Communications and Networking, pp.* 269-279, 2004.

[24] A. Ventura and M. Chiani, "Design and evaluation of some High Rate Low Density Parity Check Codes," *Proc. IEEE Globecome, vol 2, pp. 990-994, Nov 2001.*

[25] M. Yang, Y. Li, and W. E. Ryan, " Design of Efficiently Encodable moderate Length High Rate Irregular LDPC Codes," *IEEE Trans.communications, pp. 564-571, April 2004.*

Appendix A

A.1 Base addresses of code rate 1/3:

31413 18834 28884 947 23050 14484 14809 4968 455 33659
16666 19008;13172 19939 13354 13719 6132 20086 34040 13442
27958 16813 29619 16553;1499 32075 14962 11578 11204 9217
10485 23062 30936 17892 24204 24885;32490 18086 18007 4957
7285 32073 19038 7152 12486 13483 24808 21759;32321 10839
15620 33521 23030 10646 26236 19744 21713 36784 8016
12869;35597 11129 17948 26160 14729 31943 20416 10000 7882
31380 27858 33356;14125 12131 36199 4058 35992 36594 33698
15475 1566 18498 12725 7067;17406 8372 35437 2888 1184
30068 25802 11056 5507 26313 32205 37232;15254 5365 17308
22519 35009 718 5240 16778 23131 24092 20587 33385;27455
17602 4590 21767 22266 27357 30400 8732 5596 3060 33703
3596;6882 873 10997 24738 20770 10067 13379 27409 25463
2673 6998 31378;15181 13645 34501 3393 3840 35227 15562
23615 38342 12139 19471 15483;13350 6707 23709 37204 25778
21082 7511 14588 10010 21854 28375 33591;12514 4695 37190
21379 18723 5802 7182 2529 29936 35860 28338 10835;34283
25610 33026 31017 21259 2165 21807 37578 1175 16710 21939
30841;27292 33730 6836 26476 27539 35784 18245 16394 17939
23094 19216 17432;11655 6183 38708 28408 35157 17089 13998
36029 15052 16617 5638 36464;15693 28923 26245 9432 11675
25720 26405 5838 31851 26898 8090 37037;24418 27583 7959
35562 37771 17784 11382 11156 37855 7073 21685 34515;10977
13633 30969 7516 11943 18199 5231 13825 19589 23661 11150
35602;17287 27292 19033;25796 31795 12152;12184 35088
31226;38263 33386 24892;23114 37995 29796;34336 10551
36245;35407 175 7203;14654 38201 22605;28404 6595
1018;19932 3524 29305;31749 20247 8128;18026 36357
26735;7543 29767 13588;13333 25965 8463;14504 36796
19710;4528 25299 7318;35091 25550 14798;7824 215 1248;30848
5362 17291;28932 30249 27073;13062 2103 16206;7129 32062
19612;9512 21936 38833;35849 33754 23450;18705 28656
18111;22749 27456 32187;28229 31684 30160;15293 8483
28002;14880 13334 12584;28646 2558 19687;6259 4499
26336;11952 28386 8405;10609 961 7582;10423 13191
26818;15922 36654 21450;10492 1532 1205;30551 36482
22153;5156 11330 34243;28616 35369 13322;8962 1485 21186;

A.2 Base addresses of code rate 1/2:

22422 10282 11626 19997 11161 2922 3122 99;25087 16218
17015 828 20041 25656 4186 11629;11049 22853 25706 14388
5500 19245 8732 2177;16581 22225 12563 19717 23577 11555
25496 6853;16529 14487 7643 10715 17442 11119 5679
14155;5340 8636 16693 1434 5635 6516 9482 20189;18506 22236
20912 8952 5421 15691 6126 21595;8433 4694 5524 14216 3685
19721 25420 9937;21500 24814 6344 17382 7064 13929 4004
16552;22517 2429 19065 2921 21611 1873 7507 5661;1770 4636
20900 14931 9247 12340 11008 12966;6635 14556 18865 22421
22124 12697 9803 25485;19982 23963 18912 7206 12500 4382
20067 6177;756 11158 14646 20534 3647 17728 11676
11843;9306 24009 10012 11081 3746 24325 8060 19826;7575
7455 25244 4736 14400 22981 5543 8006;3482 9270 13059 15825
7453 23747 3656 24585;15627 15290 4198 22748 5842 13395
23918 16985;24896 16365 16423 13461 16615 8107 24741
3604;3729 17245 18448 9862 20831 25326 20517 24618;16455
17646 15376 18194 25528 1777 6066 21855;1400 8135 23375
20879 8476 4084 12936 25536;25119 23586 128 4761 10443
22536 8607 9752;377 21160 13474 5451 17170 5938 10256
11972;13075 9648 24546 13150 23867 7309 19798 2988;20526
3553 11525 23366 2452 17626 19265 20172;18839 21132 20119
15214 14705 7096 10174 5663; 4127 2971 17499 16287 22368
21463 7943 18880;10651 24471 14325 4081 7258 4949 7044
1078;21374 13231 22985 5056 3821 23718 14178 9978;6199
22056 7749 13310 3999 23697 16445 22636;7978 12177 2893
20778 3175 8645 11863 24623;20473 11294 9914 22815 2574
8439 3699 5431;8208 5755 19059 8541 24924 6454 11234
10492;16264 11275 24953 2347 12667 19190 7257 7174;3627
5969 13862 1538 23176 6353 2855 17720;0 18539 18661;1 10502
3002;2 9368 10761;3 12299 7828;4 15048 13362;5 18444
24640;6 20775 19175;7 18970 10971;8 5329 19982;9 11296
18655;10 15046 20659;11 7300 22140;12 22029 14477;13 11129
742;14 13254 13813;15 19234 13273;16 6079 21122;17 22782
5828;18 19775 4247;19 1660 19413;20 4403 3649;21 13371
25851;22 22770 21784;23 10757 14131;24 16071 21617;25 6393
3725;26 597 19968;27 5743 8084;28 6770 9548;29 4285
17542;30 13568 22599;31 1786 4617;32 23238 11648;33 19627
2030;34 13601 13458; 35 13740 17328;36 25012 13944;37 22513
6687;38 4934 12587;39 21197 5133;40 22705 6938;41 7534
24633;42 24400 12797;43 21911 25712;44 12039 1140;45 24306
1021;46 14012 20747;47 11265 15219;48 4670 15531;49 9417
14359;50 2415 6504;51 24964 24690;52 14443 8816;53 6926
1291;