P. 3508

ANALYSIS OF LDPC CODES FOR FUTURE WIRELESS STANDARDS

By

SARATH. R

Reg. No. 0920107019

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution affiliated to Anna University of Tech, Coimbatore)

COIMBATORE - 641049

A PROJECT REPORT

Submitted to the

FACULTY OF ELECTRONICS AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF ENGINEERING

IN

COMMUNICATION SYSTEMS

APRIL 2011

BONAFIDE CERTIFICATE

Certified that this project report entitled "**ANALYSIS OF LDPC CODES FOR FUTURE WIRELESS STANDARDS**" is the bonafide work of Mr.Sarath. R [Reg. no. 0920107019] who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Project Guide

Mrs.G.Amirtha Gowri

Head of the Department

Dr. Rajeswari Mariappan

The candidate with university Register no. 0920107019 is examined by us in the project viva-voce examination held on ....21/4/11.....

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

First I would like to express my praise and gratitude to the Lord, who has showered his grace and blessing enabling me to complete this project in an excellent manner.

I express my sincere thanks to our beloved Director **Dr.J.Shanmugam**, Kumaraguru College of Technology, for the kind support and necessary facilities to carry out the project work.

I express my sincere thanks to our beloved Principal **Dr.S.Ramachandran**, Kumaraguru College of Technology, who encouraged and supported me the project work.

I would like to express my sincere thanks and deep sense of gratitude to our HOD, **Dr.Rajeswari Mariappan**, Department of Electronics and Communication Engineering, for the valuable suggestions and encouragement which paved way for the successful completion of the project work.

In particular, I wish to thank and everlasting gratitude to the project coordinator **D.Mohanageetha(Ph.D).,** Associate Professor, Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success.

I am greatly privileged to express deep sense of gratitude to my guide **G.Amirtha Gowri (Ph.D).,** Associate Professor , Department of Electronics and Communication Engineering, for the valuable suggestions and support throughout the course of this project work. I wish to convey my deep sense of gratitude to all the teaching and non-teaching staff of ECE Department for their help and cooperation.

Finally, I thank my parents and my family members for giving me the moral support and abundant blessings in all of my activities and my dear friends who helped me to endure my difficult times with their unfailing support and warm wishes.

ABSTRACT

Low Density Parity Check Codes (LDPC) is a class of linear block codes best suited for forward error control in communication systems. The future wireless standards needs different scalable properties like multiple code rates, multiple code lengths, fixed code lengths, different throughputs depending on the applications. LDPC codes can be designed to meet the above requirements. In this paper, performance of LDPC codes is analysed for multi-rate and multiple block lengths. The strength of LDPC codes lies in its decoding algorithm and the fact that the parity-check matrix is a sparse matrix. The less number of nonzero elements in the paritycheck matrix will make the decoding comparatively easier. The analysis is done with iterative decoding algorithms (concept of belief propagation). The most efficient and commonly used algorithm is MAP algorithm. It has both hard decision and soft decision variants. The key idea of this algorithm is passing messages between bit node and check node for multiple times. These kinds of algorithm give better error correction properties and efficiency than any other kind of existing channel coding strategies. For analysing the scalability of there codes, different code rates and the number of iterations are used, which can be chosen according to the propagation channel and the available computational power. At the end of the analysis a method for generating Quasi cyclic LDPC codes with very less complexity is proposed and its performance is analysed.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

LDPC	-------	Low Density parity Check Codes
CN	-------	Check Node
BN	-------	Bit Node
MPA	-------	Message Passing Algorithm
BP	-------	Belief Propagation
BF	-------	Bit Filling
PEG	-------	Progressive Edge-Growth
BEC	-------	Binary Erasure Channel
ML	-------	Maximum Likelihood
DTMB	-------	Digital Terrestrial Multimedia Broadcast

CHAPTER 1

INTRODUCTION

Communication systems transmit data from source to destination through a channel or medium such as air, wire line and optical fibres. Reliability of the received data depends on the channel and external noises that could interface or distort the signal representing the data. Noise introduces errors in the received data. Error detection and correction is achieved by adding redundant symbols to the original data. FEC (Forward Error Correction) is used for error correction easily without data need to be retransmitted. Retransmission adds to delay, cost and wastage of system throughput. Several error correction codes have been developed to improve the reliability of data transfer. FEC includes Viterbi, convolution codes, BCH codes, RS codes, turbo codes and low density parity check (LDPC) codes.

1.1 PROJECT GOAL

The objective of the project is to analyse the LDPC codes using two different decoding schemes namely soft decision and hard decision algorithms for different code rates and block lengths. The different methods, both random and structured generation, are analysed. At the end of the analysis, a new method for generating regular LDPC codes is proposed which can claim a very low complexity compared to the basic conventional existing method.

1.2 OVERVIEW

Low-density parity-check (LDPC) codes are forward error-correction codes, first proposed in the 1962 PhD thesis of Gallager at MIT. At the time, their incredible potential remained undiscovered due to the computational demands of simulation in an era when vacuum tubes were only just being replaced by the first transistors. These codes remained largely neglected for over 35 years. In the mean time the field of forward error correction was dominated by highly structured algebraic block and convolutional codes. Despite the enormous practical success of these codes, their performance fell well short of the theoretically achievable limits set down by

Shannon. The relative quiescence of the coding field was utterly transformed by the introduction of turbo codes, proposed by Berrou, Glavieux and Thitimajshima in 1993, wherein all the key ingredients of successful error correction codes were replaced: turbo codes involve very little algebra, employ iterative, distributed algorithms, focus on average (rather than worst-case) performance, and rely on soft (or probabilistic) information extracted from the channel.

New generalizations of Gallager's LDPC codes by a number of researchers including Luby, Mitzenmacher, Shokrollahi, Spielman, Richardson and Urbanke, produced new irregular LDPC codes which offer certain practical advantages and an arguably cleaner setup for theoretical results. Today, design techniques for LDPC codes exist which enable the construction of codes which approach the Shannon's capacity to within hundredths of a decibel. The main research interests are low complexity encoding and efficient decoding schemes.

The future wireless standards need different scalable properties like multiple code rates, multiple code lengths, fixed code lengths, different throughputs depending on the application. LDPC codes can be designed to meet the above requirements. In this paper, performance of LDPC codes is analysed for multi-rate and multiple block lengths. In addition to the strong theoretical interest in LDPC codes, such codes have already been adopted in satellite-based digital video broadcasting and long-haul optical communication standards, are highly likely to be adopted in the IEEE wireless local area network standard, and are under consideration for the long-term evolution of third generation mobile telephony. The idea behind these codes dates back to the sixties, but recently such coding schemes has been given a fresh analysis and it has been shown that they can approach the information theoretical limits at unprecedented low complexity.

1.3 SOFTWARE USED

- MATLAB 7.6.0.324 (R2008a)

1.4 ORGANIZATION OF THE REPORT

- **Chapter 2** discusses about the basic concepts on LDPC codes
- **Chapter 3** deals about the encoding and decoding of the LDPC codes

- **Chapter 4** explains the proposed method for generating the regular LDPC codes with a very low complexity.
- **Chapter 5** discusses the simulation results and its interpretations.
- **Chapter 6** gives the Conclusion and Future scope of the project

CHAPTER 2

LOW DENSITY PARITY CHECK CODES

2.1 INTRODUCTION TO LDPC CODES

LDPC codes are block codes with parity-check matrices that contain only a very small number of non-zero entries. It is the sparseness of H which guarantees both a limited decoding complexity which increases only linearly with the code length and a minimum distance which also increases linearly with the code length. LDPC codes are designed by constructing a sparse parity-check matrix first and then determining a generator matrix for the code afterwards.[1][2]

Basically there are two different possibilities to represent LDPC codes. Like all linear block codes they can be described via

- Matrices.
- Graphical representation.

The matrix given below is a parity check matrix with dimension n ×m for a (6, 2) code. Two numbers are used to describing these matrixes. W_r for the number of 1's in each row and W_c for the columns. For a matrix to be called low-density the two conditions $W_c << n$ and $W_r << m$ must be satisfied. The matrix is called sparse matrix since the number of ones in the matrix will be less than the number of zeros.

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

Tanner introduced an effective graphical representation for LDPC codes. This way of representing the codes is called the Tanner graph. Tanner graph methods are very easy in implementing the LDPC decoding. Tanner graphs are bipartite graphs. A bipartite graph is a graph whose nodes can be divided into two sets such that each node is connected to a node in the other set. The two sets of nodes in a Tanner graph are called check nodes and variable nodes representing rows and eolumns respectively. That means that the nodes of the graph are separated into two distinctive sets and edges are only connecting nodes of two different types. The i^{th} check node is connected to the j^{th} variable node if and only if $H_{i,j} = 1$. Check nodes represent the six

rows of the matrix, whereas are the eolumns. The number of edges in each check node is equal to the row weight and the number of edges in each variable node is equal to the column weight. [3]

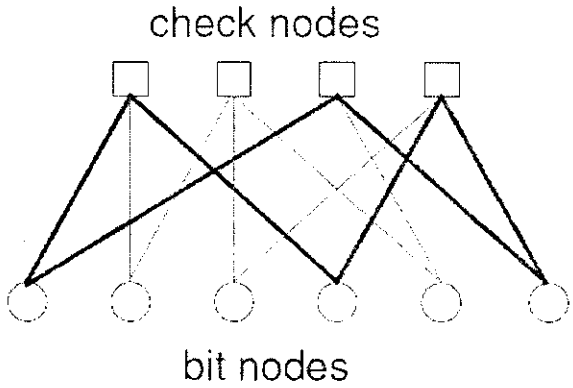

Figure 2.1: Tanner Graph Representation Of Parity Check Matrix

Figure 2.1 is an example for such a Tanner graph and represents the same code as the matrix in equation (1). The creation of such a graph is rather straight forward. It consists of m check nodes (the number of parity bits) and n variable nodes (the number of bits in a codeword). Check node c_i is connected to variable node b_j if the element hij of H is a 1.

A cycle in a parity check matrix is formed by a complete path through '1' entries with alternating moves between rows and eolumns. In a Tanner graph a cycle is formed by a path starting from a node and ending at the same node. The length of the cycle is given by the number of edges in the path. A cycle of six is shown in bold in the graph of Figure 2.1. The smallest cycle in a Tanner graph or parity check matrix is called its girth. The smallest possible girth is four. A bipartite graph has a minimum cycle of length four and has even cycle lengths.

Basically there are 2 classes of LDPC codes. They are:

- Regular Codes
- Irregular Codes

The regular codes are those codes in which the row weight and column weight distributed evenly through out the H matrix. In the case of irregular codes the row and column weights will differ through out the matrix. Basically design of regular LDPC codes is easy. But generally it is observed that the carefully designed irregular codes will give better performance for very large code lengths. [4]

The biggest difference between LDPC codes and classical block codes is how they are decoded. Classical block codes are generally decoded with ML (Maximum Likelihood) like decoding algorithms and so are usually short and designed algebraically to make this task less complex. LDPC codes however are decoded iteratively using a graphical representation of their parity-check matrix and so are designed with the properties of H as a focus.

2.2 ERROR CORRECTION USING PARITY CHECKS

The basic idea of forward error control coding is to augment these *message* bits with deliberately introduced redundancy in the form of extra *check* bits to produce a codeword for the message. These check bits are added in such a way that code words are sufficiently distinct from one another that the transmitted message can be correctly inferred at the receiver, even when some bits in the codeword are corrupted during transmission over the channel.

The simplest possible coding scheme is the single parity check code (SPC). The SPC involves the addition of a single extra bit to the binary message, the value of which depends on the bits in the message. In an even parity code, the additional bit added to each message ensures an even number of 1s in every codeword. More formally, for the 7-bit ASCII plus even parity code we define a codeword c to have the following structure:

$$c = [c1\ c2\ c3\ c4\ c5\ c6\ c7\ c8] \tag{2}$$

where each c_i is either 0 or 1, and every codeword satisfies the constraint

$$c1 \oplus c2 \oplus c3 \oplus c4 \oplus c5 \oplus c6 \oplus c7 \oplus c8 = 0 \tag{3}$$

Equation (3) is called a *parity-check equation*, in which the symbol $\oplus$ represents modulo-2 addition.

While the inversion of a single bit due to channel noise can easily be detected with a single parity check code, this code is not sufficiently powerful to indicate which bit, or indeed bits, were inverted. Moreover, since any even number of bit inversions produces a string satisfying the constraint in equation (3), patterns of even numbers of errors go undetected by this simple code. Detecting more than a single bit error calls for increased redundancy in the form of additional parity bits and more

sophisticated codes contain multiple parity-check equations and each codeword must satisfy every one of them.

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

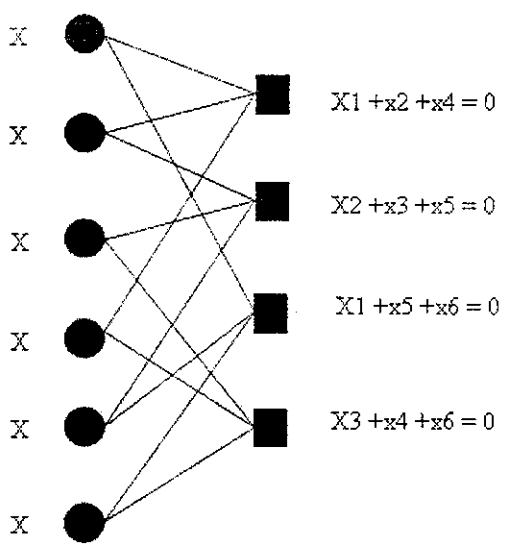

Figure 2.2: Parity Check Constrain for LDPC Codes

The matrix H is the *parity-check matrix*. Each row of H corresponds to a parity-check equation and each column of H corresponds to a bit in the codeword. Thus for a binary code with m parity-check constraints and length n codeword the parity-check matrix is an M×N binary matrix. In matrix form a string y = [c1 c2 c3 c4 c5 c6] is a valid codeword for the code with parity-check matrix H if and only if it satisfies the matrix equation.[5]

2.3 IMPORTANT DESIGN PARAMETERS

Design of LDPC codes involves many parameters which are often determined in consideration of the target application.[6]

Code size: The code size specifies the dimensions of the parity check matrix (M×N). Sometimes the term code length is used referring to N. Generally a code is specified using its length and row-column weights in the form (N, j, k). M can be deduced from the code parameters N, j and k. It has been shown that very long codes perform better than shorter ones. Long codes are therefore desirable to have good performance. However, their hardware implementation requires more resources (memory plus professing nodes).

Code Weights and Rate: The rate of a code, R, is the number of information bits over the total number of bits transmitted. It is expressed as $\left(\frac{N-M}{N}\right)$. Higher row and column weights result in more computations at each node because of many incoming messages. However, if many nodes contribute in estimating the probability of a bit the node reaches a consensus faster. Higher rates mean fewer redundancy bits. That is, more information data is transmitted per block resulting in high throughput. However, low redundancy means less protection of bits and therefore less decoding performance or higher error rate. Low rate codes have more redundancy with fewer throughputs. More redundancy results in more decoding performance. However, very low rates may have poor performance with a small number of connections. LDPC codes with column-weight of two have their minimum distance increasing logarithmically with code size as compared to a linear increase for codes with column weight of three or higher. As a result column-weight two codes perform poorly compared to higher column-weight codes. Column weights higher than two are usually used. Although regular codes are commonly used, carefully constructed irregular codes could have better error correcting performance.

Code Structure: The structure of a code is determined by the pattern of connections between rows and columns. The connection pattern determines the complexity of the communication interconnect between check and variable processing nodes in an

encoder and decoder hardware implementations. Random codes do not follow any predefined or known pattern in row-column connections. Structured codes on the other hand have a known interconnection pattern. Many methods have been developed for constructing those types of codes.

Number of iterations: The number of iterations is the number of times the received bits are estimated before a hard decision is made by the decoding algorithm. A large number of iterations may ensure decoding algorithm convergence but will increase decoder delay and power consumption. The number of corrected errors generally decreases with an increasing number of iterations. In performance simulations a large number of iterations, (about 100 to 200), can be used. For practical applications 20 to 30 iterations are commonly used.

2.4 CONSTRUCTION OF LDPC CODES

The two major classifications of the construction of LDPC codes are given by

- Random construction
    - MacKay Constructions
    - Bit filling Algorithm
    - Progressive Edge-Growth Algorithm
- Structured construction
    - Combinatorial Designs
    - Finite Geometry Designs
    - Algebraic Methods

2.4.1 Random Constructions

Random constructions connect rows and columns of a LDPC code matrix without any structure or predefined connection pattern. They are actually pseudo-random connections done by computer searches. Constructions could be done in the Tanner graph by connecting check to variable nodes with edges or in the parity-cheek matrix by connecting rows to columns with '1' entries where all other entries are '0's. Randomly adding edges to a Tanner graph or '1' entries in the parity-cheek

matrix will not produce a desired rate and will probably have cycles of four. However, the resulting code could be optimized by either post processing or by putting constraints on the random choices as the code is built. Post processing exchanges or deletes some connections in order to get a desired girth and rate. Random construction with constraints adds a connection in the code if it does not violate the desired girth or row and column weights.[7]

Random codes have good performance especially at long code lengths compared to structured codes. Random construction methods could be used to maximize performance (e.g. by girth) and rate for a given size as demonstrated by Campello. Heuristic algorithms are developed to search for good LDPC codes based on average girth distribution. While random codes show better performance compared to structured or constrained codes at code lengths of several thousands, there is no assurance that a particular code chosen at random will have good performance. Some random construction methods of importance are explained.

2.4.1.1 MacKay Constructions

MacKay showed that random LDPC codes have good performance approaching Shanon's limit. Some random construction methods for developing codes are suggested by him. A simple and easy to implement version of random construction in given below:

1. Matrix H is generated by starting with an all zero matrix and then randomly flipping bits in the matrix. Flipped bits are not necessarily distinct.
2. Matrix H is generated by randomly creating weight j columns.
3. Matrix H is generated with weight j per column and uniform weight per row and no two columns are connected to the same row more than once (avoiding four-cycles).
4. Matrix H is generated as in 3 with the girth condition further constrained so that the girth is larger than six.

MacKay's algorithms were used to find good performing codes with a variety of rates and length. These algorithms made the generation of random and semi random LDPC generation easier.[8]

2.4.1.2 Bit-Filling Algorithm

A bit-filling (BF) algorithm introduced in constructs a LDPC code by connecting rows and columns of a code one at a time provided a desired girth is not violated. The number of connections to rows and columns are kept almost balanced (about same number of connections) by connecting rows or columns with the least number of connections first. The algorithm obtains irregular codes with either a fixed row or column weight.[8-9]

The algorithm can obtain very-high rate and high-girth codes. It is extended in to get better girths and rates for a given code length. Although the algorithm produces high-rate and high-girth codes given a particular code size, resulting codes are not easily implementable in hardware. This is because the structure of row-column connections is not consistent enough to be an advantage in hardware implementation. The objective of the algorithm is to optimize girth or rate for a given code size.

Algorithm:

C is check node,

M is the number of check nodes

$\aleph_c$ is a set of check nodes sharing a variable node and

U_1 is a set of check nodes connected to the new or current variable node.

F_0 is a set of check node that can be connected

To U_1 without violating the girth

Set N = 0, A =| M |, and $U_1 = \emptyset$

for c $\in$| M |, set weight(e)=0 and $\aleph_c = \emptyset$

do {

$\forall c \in U_1$, set $H_{c,N} = 1$ and increment

deg(e) by 1 set i = 0, $U_1 = \emptyset$, and U=$\emptyset$

do {

Compute $F_0 = A \setminus U$

If ($F_0 = \emptyset$)

{

Choose c^* from F_0 according to some heuristic

$\forall c \in U_1$, update $\aleph_c = \aleph_c \cup \{c^*\}$ and update $\aleph_{c^*} = \aleph_{c^*} \cup U_1$

update

$U_1 = U_1 \cup \{c^*\}$, $U = U \cup V_{g/2-1}(c^*)$, and A}

$i = i + 1$

} while ((i < a) and ($F_0 = \emptyset$))

n=n+1

} while ((i = a) and ($F_0 = \emptyset$))

2.4.1.3 Progressive Edge-Growth Algorithm

The progressive edge-growth (PEG) algorithm is another simple non-algebraic algorithm that can be used to construct codes of arbitrary length and rate. It is similar to the bit- filling algorithm. In PEG node degrees are distributed according to some performance criteria (e.g. density evolution) before edges are added. The algorithm builds a Tanner graph by connecting the graph's nodes edge by edge provided the added edge has minimal impact on the girth of the graph. With this algorithm regular and irregular codes can be obtained with optimized performance. Codes obtained using this method is one of the best known performing codes at short lengths.

There are other variations of the algorithm that slightly improve performance of the obtained codes. Just as with bit-filling code, a major disadvantage of PEG codes is their high hardware complexity making them impractical at very large lengths. The unstructured interconnection results in routing complexity and congestion in decoder implementations. When row-column connections are done at random, there would be no general rule to define how a set of rows or columns are connected. Therefore the random connections need to be defined by a table (addressing) for each individual row or column or hardwired. Since codes are generally a few or more thousands in size, the tables would also be very large or the number of interconnections will be high and unstructured leading to long wire-lengths, large decoders with slow clock rates.[11]

2.4.2 Structured Constructions

Structured construction methods put constraints on row-column connections to get a desired or predefined pattern. The main objectives are to achieve good performance and at the same time have a connection pattern that is easier to implement in hardware. There are many structured methods already developed producing codes differing in structure (row-column interconnection pattern), performance and hardware implementation complexity. Developed methods include those based on graphs, combinatorial designs, algebra and heuristic searching techniques. Some existing structured constructions are explained briefly.[13]

2.4.2.1 Combinatorial Designs

A combinatorial design is an arrangement of sets of v points into b subsets called blocks. The inclusion or placement of points in blocks is according to some defined constraints.[12] The basic constraints are

1. A pair of points can appear together in A blocks for a defined value of A.
2. The number of points in each block is given by γ and the number of blocks in which a point appears.

2.4.2.2 Finite Geometry

Finite geometries are another approach similar to combinatories that can be used to design structured LDPC codes. A finite geometry is defined by n points and J lines with the following properties.

1. Every line consists of ρ points.
2. Any two points are connected by one and only one line.
3. Every point lies on γ lines.
4. Two lines are either parallel or they intersect at only one point.

2.4.2.3 Algebraic Methods

Parity check matrix connections could be constrained algebraically to obtain codes with a particular structure. Constraints could also be used to get a desired girth, rate or length. Fossorier presents algebraic constraints to obtain quasi-cyclic codes of a given girth. The code matrix is divided into shifted identity sub-matrices of equal sizes.[14]

2.4.3 Quasi Cyclic LDPC Codes

A QC LDPC codes are codes in which rows and columns in the sub matrix have similar and cyclic connections. They are a class of regular LDPC codes. They can be formed by a concatenation of circularly shifted sub matrices. Due to their quasi cyclic nature, QC-LDPC codes can be encoded efficiently with shift registers. The codes have general structure as:

$$H = [A_1, A_2, \ldots, A_k] \quad (4)$$

Where A_i is a circulant matrix. Such structures can be obtained with combinatorial construction and finite geometry methods. In the other construction method, the matrix is formed by isolated shifted identity sub matrices as shown in figure 2.3(a). In the fig. I_{xy} is a p× p shifted identity sub matrices and O is a p × p zero sub matrix, where p is a positive integer.

A shifted identity matrix is obtained by shifting each row of an identity matrix to the right or left by some amount. There is 'p' such identity matrices for a p × p matrix. In Figure 2.3(a) the number of sub matrices in row is equal to row weight and equal to column weight in column. But, in Figure 2.3(b). , the number of sub matrices greater than row and column weight. [15]

I_{11}	I_{12}	I_{13}	I_{14}
I_{21}	I_{22}	I_{23}	I_{24}
I_{31}	I_{32}	I_{33}	I_{34}

(a)

I_{11}	I_{12}	0	0	0	I_{13}	I_{14}
I_{21}	0	I_{22}	0	0	I_{23}	I_{24}
0	I_{31}	I_{32}	0	I_{33}	0	I_{34}
0	I_{41}	0	I_{42}	0	I_{43}	I_{44}
0	I_{51}	0	I_{52}	0	I_{53}	I_{54}

(b)

Figure2.3: Quasi Cyclic Code Arrangement (a) With all Non-zero Sub-matrices (b) With Zero Sub-matrices

2.4.4 Repeat Accumulate LDPC Codes

Repeat accumulate codes are "turbo-like" and are simple to design and understand. A repeat-accumulate (RA) code is an LDPC code with an upper triangular form already built into the parity-check matrix during the code design. An m × n RA code parity-check matrix H has two parts and it's given by $H = [H_1, H_2]$. The parity-check matrix of an RA code is called (q, a)-*regular* if the weight of all the rows of H_1 are the same, a, and the weight of all the columns of H_1 are the same, q. Note that a regular RA parity-check matrix has columns of weight 2, and one column of weight 1, in H_2 and so is not regular in the sense of (j, r)-regular LDPC codes. An irregular RA code will have an irregular column weight distribution in H_1, with H_2 the same as for a regular code. Generally RA codes are suitable for low and medium code rates.[16]

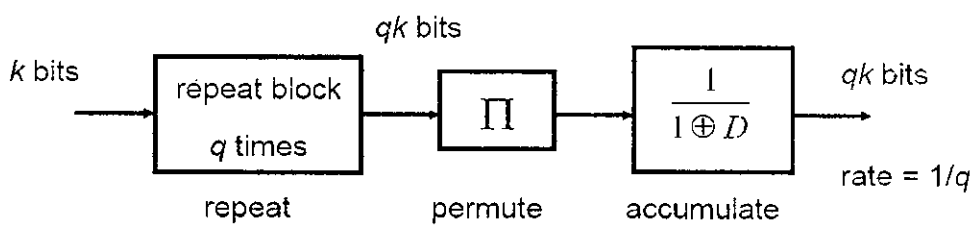

Figure2.4: Principle of RA Codes

Recently Irregular Repeat Accumulate codes (IRA) and Extended Irregular Repeat Accumulate Codes (eIRA) is of great interests for the researchers.

2.5 APPLICATIONS

DVB-S2: Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) is designed as a successor for the popular DVB-S digital television broadcast standard, and was developed in 2003 and ratified by ETSI (EN 302307) in March 2005. It is based on DVB-S and the electronic news-gathering (or Digital Satellite News Gathering) standard, used by mobile units for sending sounds and images from remote locations world-wide back to their home television stations. DVB-S2 is envisaged for

broadcast services including standard and HDTV, interactive services including Internet access, and (professional) data content distribution.[17]

Two new key features that were added compared to the DVB-S standard are

- Introduction of a powerful coding scheme based on modern LDPC code.
- VCM (Variable Coding and Modulation) and ACM (Adaptive Coding and Modulation) modes, which allow optimizing bandwidth utilization by dynamically changing transmission parameters.

WiMAX: WiMAX (Worldwide Interoperability for Microwave Access) is a telecommunications protocol that provides fixed and mobile Internet access. The current WiMAX revision provides up to 40 Mbit/s with the IEEE 802.16m update expected to offer up to 1 Gbit/s fixed speeds. The name "WiMAX" was created by the WiMAX Forum, which was formed in June 2001 to promote conformity and interoperability of the standard. The forum describes WiMAX as "a standards-based technology enabling the delivery of last mile wireless broadband access as an alternative to cable and DSL".

DTMB: DTMB (Digital Terrestrial Multimedia Broadcast) is the TV standard for mobile and fixed terminals used in the People's Republic of China, Hong Kong and Macao. Besides the basic functions of traditional television service, the DTMB allows additional services using the new television broadcasting system. DTMB system is compatible with fixed reception (indoor and outdoor) and mobile digital terrestrial television. The DTMB standard uses many advanced technologies to improve their performance, for example, a pseudo-random noise code (PN-Pseudo-random Noise) as a guard interval that allows faster synchronization system and a more accurate channel estimation, LDPC (Low-Density Parity-Check) encoding for error correction, modulation TDS-OFDM (Time Domain Synchronization - Orthogonal Frequency Division Multiplexing) which allows the combination of broadcasting in SD, HD and multimedia services, etc.

Storage Applications: As peer-to-peer and widely distributed storage systems proliferate, the need to perform efficient erasure coding, instead of replication, is crucial to performance and efficiency. *Low-Density Parity-Check (LDPC)* codes have arisen as alternatives to standard erasure codes, such as Reed-Solomon codes, trading off vastly improved decoding performance for inefficiencies in the amount of data that must be acquired to perform decoding. Storage systems require very high code rates, low SNR (7-12 dB), and very high data rate in the Gbps range.

CHAPTER 3

ENCODING AND DECODING OF LDPC CODES

3.1 ENCODING PROCESS

The LDPC encoder transforms each input message block 'u' into a distinct *N*-tuple (*N*-bit sequence) code word 'c'. The codeword length *N*, where $N > K$, is then referred to as the *block-length*. And, there are 2^K distinct code words corresponding to the 2^K message blocks. This set of the 2^K code words is termed as a C(*N,K*) *linear block code*. The word *linear* signifies that the modulo-2 sum of any two or more code words in the code C(*N,K*) is another valid codeword. The number of non-zero symbols of a codeword 'c' is called the *weight*, while the number of bit-positions in which two code words differ is termed as the *distance*. The *minimum distance* of a linear code is denoted by d_{min}, and determined by the weight of that codeword in the code C(*N,K*), which has the minimum weight.[18]

The unique and distinctive nature of the code words implies that there is a one-to-one mapping between a *K*-bit information sequence 'u' and the corresponding *N*-bit codeword 'c' described by the set of rules of the encoder.

A generator matrix 'G' is determined by performing Gauss–Jordan elimination on 'H' to obtain it in the form:

$$[H' = [A, I_{N-K}]] \tag{5}$$

Where 'A' is a (N-K) × K binary matrix and I_{N-K} is the size N-K identity matrix. The generator matrix is then:

$$G = [A, I_{N-K}] \tag{6}$$

Since LDPC codes are linear block codes, a codeword is generated by multiplying the input vector with the generator matrix,

$$c = uG \tag{7}$$

Where 'c' is the code word and 'u' is the input vector bits. Since 'G' matrix is not sparse, the matrix multiplication at the encoder will have complexity in the order of n^2 operations.

3.2 DECODING OF LDPC CODES

LDPC code decoding tries to reconstruct the transmitted codeword, c, from the possibly corrupted received word, y. It is achieved by using the parity-check matrix, H. The condition that $cH^T = 0$ defines the set of parity-check constraints or equations that must be satisfied for the received codeword to be the same as the transmitted codeword. LDPC code decoding is achieved through iterative processing based on the Tanner graph, to satisfy the parity check conditions.

The different variants for the iterative decoding schemes based in the contest are given by

- Sum-Product Algorithm
- Min-Sum Algorithm
- Forward-Backward Algorithm, BCJR algorithm (Trellis Based Graphical Method)

A message passing algorithm (MPA) based on Pearl's belief algorithm describes the decoding iterative steps. The passed messages are probability estimations. Some of the decoding algorithms are given below. [19]

3.2.1 Message-Passing Decoding

The bit-flipping algorithm is a hard-decision message-passing algorithm for LDPC codes. The class of decoding algorithms used to decode LDPC codes are collectively termed message-passing algorithms since their operation can be explained by the passing of messages along the edges of a Tanner graph. Each Tanner graph node works in isolation, only having access to the information contained in the messages on the edges connected to it.

The message-passing algorithms are also known as iterative decoding algorithms as the messages pass back and forward between the bit and check nodes iteratively until a result is achieved (or the process halted). Different message-passing algorithms are named for the type of messages passed or for the type of operation performed at the nodes. In some algorithms, such as bit-flipping decoding, the messages are binary and in others, such as belief propagation decoding, the messages are probabilities which represent a level of belief about the value of the codeword

bits. It is often convenient to represent probability values as log likelihood ratios, and when this is done belief propagation decoding is often called sum-product decoding since the use of log likelihood ratios allows the calculations at the bit and check nodes to be computed using sum and product operations.

The steps of the message passing algorithm is given below

Step 1: Initialization – Initialise each variable node with received information from the source. Each variable node calculates the initial Log Likelihood Ratios (λ_m)

Step 2: Check-node update – Each check node calculate LLR and Check to variable node messages based on the incoming messages.

Step 3: Variable-node update – For each variable node, calculate LLR (λ_n) and outgoing messages along its edges to check nodes. LLR is the sum of all the incoming messages plus the initial value of the variable node. The outgoing message for each edge is given by the check node LLR minus the message received on that edge.

Step 4: Decision - Quantize the LLR of variable nodes such that $LLR_m = 0$ if $\lambda_n < 0$, and $LLR_m = 1$ if $\lambda_n \geq 0$. If $LLR \times H^T = 0$, then halt the algorithm with LLR at the decoder output. LLR gives the estimation of the codeword, c_n. Otherwise go to Step 2. If the algorithm does not halt within some maximum number of iterations, then declare a decoder failure.

3.2.1.1 Message-passing on the binary erasure channel

On the binary erasure channel (BEC) a transmitted bit is either received correctly or completely erased with some probability ε. Since the bits which are received are always completely correct the task of the decoder is to determine the value of the unknown bits. If there exists a parity-check equation which includes only one erased bit the correct value for the erased bit can be determined by choosing the value which satisfies even parity.

Since the received bits in an erasure channel are either correct or unknown (no errors are introduced by the channel) the messages passed between nodes are always the correct bit values or 'x'. When the channel introduces errors into the received word, as in the binary symmetric or AWGN channels, the messages in message-passing decoding are instead the best guesses of the codeword bit values based on the current information available to each node.

3.2.2 Bit Flipping Algorithm

The bit-flipping algorithm is a hard-decision message-passing algorithm for LDPC codes. A binary (hard) decision about each received bit is made by the detector and this is passed to the decoder. For the bit-flipping algorithm the messages passed along the Tanner graph edges are also binary: a bit node sends a message declaring if it is a one or a zero, and each check node sends a message to each connected bit node, declaring what value the bit is based on the information available to the check node.

The check node determines that its parity-check equation is satisfied if the modulo-2 sum of the incoming bit values is zero. If the majority of the messages received by a bit node are different from its received value the bit node changes (flips) its current value. This process is repeated until all of the parity-check equations are satisfied, or until some maximum number of decoder iterations has passed and the decoder gives up. [20]

The bit-flipping decoder can be immediately terminated whenever a valid codeword has been found by checking if all of the parity-check equations are satisfied. This is true of all message-passing decoding of LDPC codes and has two important benefits; firstly additional iterations are avoided once a solution has been found, and secondly a failure to converge to a codeword is always detected.

The bit-flipping algorithm is based on the principal that a codeword bit involved in a large number of incorrect check equations is likely to be incorrect itself. The sparseness of H helps spread out the bits into checks so that parity-check equations are unlikely to contain the same set of codeword bits. The bit-flipping algorithm is presented in Algorithm 3. Input is the hard decision on the received vector, $y = [y_1, \ldots y_n]$, and output is $M = [M_1, \ldots, M_n]$.

Algorithm: Bit-flipping Decoding

```
Procedure DECODE(y)
    I = 0                                   Initialization
    for i = 1 : n do
        Mi = yi
    end for
repeat
    for j = 1 : m do                        Step 1: Check messages
```

```
            for i = 1 : n do
                    Ej,i = ∑ i'∈Bj ,i'6=i(Mi' mod 2)
            end for
        end for
        for i = 1 : n do                          Step 2: Bit messages
            if the messages Ej,i disagree with yi then
                    Mi = (ri + 1 mod 2)
            end if
        end for
        for j = 1 : m do ◁ Test: are the parity-check
            Lj = ∑ i'∈Bj (Mi' mod 2)      Step 3:equations satisfied
        end for
        if all Lj = 0 or I = Imax then
            Finished
        else
            I = I + 1
        end if
    until Finished
end procedure
```

Simulated Example of Decoding (Bit Flipping Algorithm):

Each sub-figure indicates the decision made at each step of the decoding algorithm based on the messages from the previous step. A cross represents that the parity check is not satisfied while a tick indicates that it is satisfied. For the messages, a dashed arrow corresponds to the messages "bit = 0" while a solid arrow corresponds to "bit = 1".Thus by repeated message passing between the check nodes and the bit nodes we can finally able to tell the received code word is correct or nor .if there is any error in the code word then the algorithm will correct the errors. Since there is no connection with in the bit nodes and the check nodes and only connection between them the iterative decoding is easy in this case

Step 1: initialization

This is the first phase of MPA. In this phase in tanner graph the bit nodes are assigned the value of the received code word, this can or cannot be true. Then the bit nodes will send the information in to the corresponding check nodes to which they are connected. At the check node XOR operations are performed. If all the result of the XOR operation is zero then what ever code word we got is the actual code word or else there is an error in the code word which have to be corrected

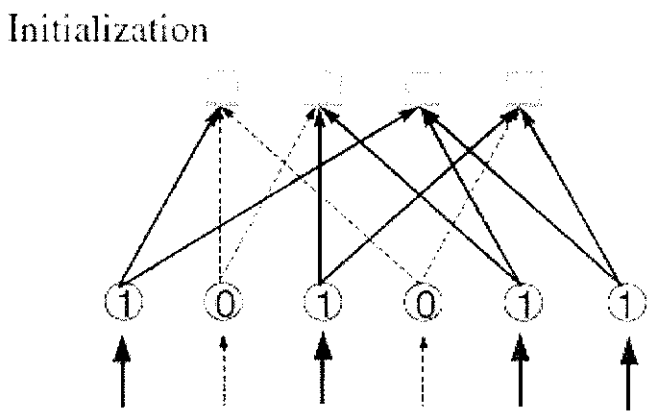

Figure 3.1: Initialization of the Bit Node

In the case of received bits [1 0 1 0 1 1] the value of the check bits are

$B_0 = 1$ $B_1 = 0$ $B_2 = 1$ $B_3 = 0$

Since all the bits in this case is not zero this is not satisfying the parity-check equations and this is not the actual code word

Step 2: Check-node update

This is the next step in decoding. The check nodes will send the values they hold to all the bit nodes to which they have connected. E_{ij} is the value passed from the j^{th} check node to the i^{th} bit node. Since one check node is connected to three bit nodes .it will take the incoming value from any two of the bit node and exor in and passed to the third one .this can be summarised in terms of all E_{ij}'s

$E_{11} = 0$ $E_{31} = 0$

$E_{22} = 0$ $E_{12} = 0$

$E_{23} = 1$ $E_{43} = 1$

$E_{14} = 1$ $E_{44} = 0$

$E_{25} = 1$ $E_{35} = 0$

$E_{36} = 0$ $E_{46} = 1$

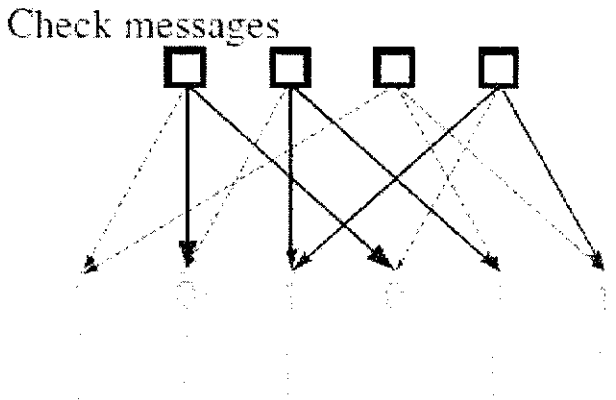

Figure 3.2: Check Node Message updates

Step 3: Variable-node update

The variable node values are up dated by looking the message from the check nodes .this will look maximum polling algorithm. That means each bit node will get messages from the two check nodes .That is two bits it can be of four different combinations they are {0,0} {0,1} {1,0} {1,1}.thus if the update from the check nodes are {1,0} or {0,1} whatever we received at the receiver is taken as the correct. But when the received information from the check node are {1,1} by maximum polling algorithm we will take the correct bit as '1' whatever we received. Similarly in the case of {0,0} we will take the error free received bit as '0' for whatever we received

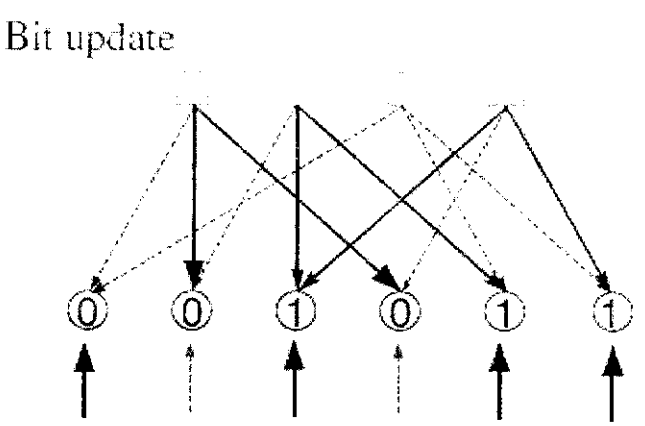

Figure 3.3: Variable-node Updates

Step 4: Decision

In this step the decisions will take. This is that by the new updated value of the received code word will be sending to check nodes again for the checking of correction .here we got that all the B values are zero so we represent it by the tic mark. Thus the error correction of the code is done

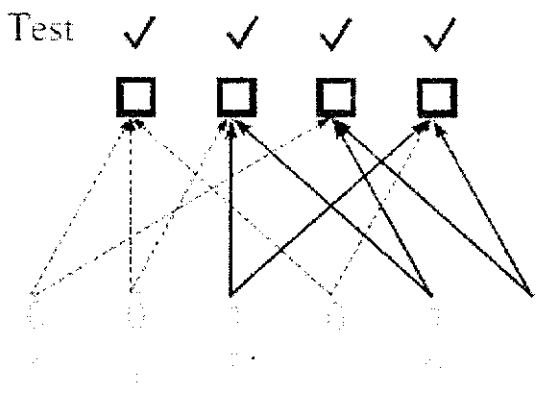

Figure 3.4 Check node Testing & Decision

Thus the error correction of the code is done and corrected code word is given as the states of the check node.

3.2.3 Sum-Product Decoding

The sum-product algorithm is a soft decision message-passing algorithm. It is similar to the bit-flipping algorithm described in the previous section, but with the messages representing each decision (check met, or bit value equal to 1) now probabilities. Whereas bit-flipping decoding accepts an initial hard decision on the received bits as input, the sum-product algorithm is a soft decision algorithm which accepts the probability of each received bit as input. The input bit probabilities are called the a priori probabilities for the received bits because they were known in advance before running the LDPC decoder. The bit probabilities returned by the decoder are called the a posteriori probabilities. In the case of sum-product decoding these probabilities are expressed as log-likelihood ratios.[21]

Log likelihood ratios are used to represent the metrics for a binary variable by a single value.

$$L(x) = \log\left(\frac{p(x=0)}{p(x=1)}\right) \tag{8}$$

The benefit of the logarithmic representation of probabilities is that when probabilities need to be multiplied log-likelihood ratios need only be added, reducing implementation complexity.

For an AWGN channel the a priori LLR is are given by,

$$r_i = 4\, y_i\left(\frac{E_s}{N_o}\right) \tag{9}$$

The term y_i represents the received code word and for the simulation purpose we are assuming the $E_s / N_0 = 1.25$. The extrinsic LLR and the total LLR calculation are done to find the codeword .

The sum-product algorithm iteratively computes an approximation of the MAP value for each code bit. However, the a posteriori probabilities returned by the sum-product decoder are only exact MAP probabilities if the Tanner graph is cycle free. Briefly, the extrinsic information obtained from a parity check constraint in the first iteration is independent of the a priori probability information for that bit (it does of course depend on the a priori probabilities of the other codeword bits). The extrinsic information provided to bit i in subsequent iterations remains independent of the original a priori probability for bit i until the original a priori probability is returned back to bit i via a cycle in the Tanner graph. The correlation of the extrinsic information with the original a priori bit probability is what prevents the resulting posteriori probabilities from being exact.

Here a modified form of sum-product algorithm can be used to reduce the implementation complexity of the decoder.

$$E_{i,j} = 2\tanh^{-1}\left(\prod_{j\in B_{i,i'}} \tanh\left(\frac{M_{j,i}}{2}\right)\right) \tag{10}$$

First $M_{j,i}$ can be factored as

$$M_{j,i} = \alpha_{j,i'}\beta_{j,i'} \tag{11}$$

Where,

$$\alpha_{j,i'} = sign(M_{j,i'}) \text{ and}$$

$$\alpha_{j,i'} = |M_{j,i'}| \tag{12}$$

The reduced equation for $E_{j,i}$ is given by,

$$E_{j,i} = \left(\prod_{i'} \alpha_{j,i'} \right) \phi \left(\sum_{i'} \phi \left(\beta_{j,i'} \right) \right) \tag{13}$$

$\phi(x)$ is a function and it is given by,

$$\phi(x) = -\log \tanh \left(\frac{x}{2} \right) = \log \frac{e^x + 1}{e^x - 1} \tag{14}$$

The product of the signs can be calculated by using modulo 2 addition of the hard decisions on each $M_{j,i}$ while the function $\phi(x)$ can be easily implemented using a lookup table.

Algorithm: Sum-Product Decoding

procedure DECODE(r)

I = 0

for i = 1 : n do

  for j = 1 : m do

    $M_{j,i} = r_i$

  end for

end for

repeat

for j = 1 : m do

  for i ∈ Bj do

$$E_{j,i} = \left(\prod_{i'} \alpha_{j,i'} \right) \phi \left(\sum_{i'} \phi(\beta_{j,i'}) \right)$$

  end for

end for

for i = 1 : n do

  $L_i = \sum j \in A_i \cdot E_{j,i} + r_i$

$$z_i = \begin{cases} 1, L_i \le 0 \\ 0, L_i \rangle 0 \end{cases}$$

end for

for i = 1 : n do

  for j ∈ A_i do

$Mj,i = \sum j' \in A_i, j' \neq j \; E_{j',i} + r_i$

end for

end for

I = I + 1

end if

until Finished

end procedure

CHAPTER 4

PROPOSED METHOD FOR REGUALR LDPC CODE GENERATION

4.1CONVENTIONAL QUASI CYCLIC METHOD

A QC LDPC codes are codes in which rows and columns in the sub matrix have similar and cyclic connections. They can be formed by a concatenation of circularly shifted sub matrices. Due to their quasi cyclic nature, QC-LDPC codes can be encoded efficiently with shift registers. The codes have general structure as:

$$H = [A_1, A_2, \ldots, A_k] \quad (15)$$

Where A_i is a circulant matrix. Such structures can be obtained with combinatorial construction and finite geometry methods [6]. In the other construction method, the matrix is formed by isolated shifted identity sub matrices as shown in fig.4.1. [22]

A shifted identity matrix is obtained by shifting each row of an identity matrix to the right or left by some amount. In fig.4.a. the number of sub matrices in row is equal to row weight and equal to column weight in column.

I	I	I	I	I	I
I	I_1	I_2	I_3	I_4	I_5
I	I_2	I_3	I_4	I_5	I_6

Figure 4.1: Conventional Quasi Cyclic Generation

4.2 PROPOSED METHOD

The proposed method is a variant of the conventional method of generation. Here instead of shifting the rows of the I matrix to generate the I_N matrix, we will use the mirror image of the I matrix, denoted by I_M and place those matrices accordingly to form bigger H matrices.

We can arrange the I and I_M in many ways so that girth conditions are avoided. Two possible ways of arrangements are given in Figure 4.2 and Figure 4.3.

I	I	I	I	I	I
I	I_M	I_M	I_M	I_M	I_M
I	I	I_M	I_M	I_M	I_M

Figure 4.2: Proposed Method: Arrangement 1

I	I	I	I	I	I
I	I_M	I	I_M	I	I_M
I_M	I	I_M	I	I_M	I

Figure 4.3: Proposed Method: Arrangement 2

In the proposed method the number of shifts needed to build the sum-matrices is considerably low when compared to the conventional method. Here only we need to construct a mirror image I_M of the identity matrix and we can use it over and over. So in the case of I of size 10, in the above example, the proposed method needs 10 number of shifts to either left or right. But the conventional method needs $5 \times 10 = 50$ (Each sub-matrix needs 10 shifts) shifts to construct the full H matrix.

When coming to the case of performance, the proposed method gives a closely comparable performance which is slightly less than the performance of the conventional method, by a very small margin but can be achieved by considerably less complexity of generation. Since the number of shifts required are minimised, the hardware implementation becomes easier. And also the speed of operation of the encoder increases. The performance analysis for the proposed method is given in the chapter 5.

CHAPTER 5

SIMULATION RESULTS AND DISCUSSIONS

The platform used here to simulate the performance of the LDPC codes is MATLAB (2008a). Here the parameter focused on is BER (Bit error rate) for a specific modulation scheme over a specified channel. The BER is given by the ratio of the number of bits in error to the total number of bits transmitted.

BER = Number of bits in error Total number of bits transmitted.

$$BER = \frac{\text{Number of bits in error}}{\text{Total number of bits transmitted}} \tag{16}$$

First we are analyzing the performance of the soft decision decoding algorithm and the hard decision decoding algorithm.

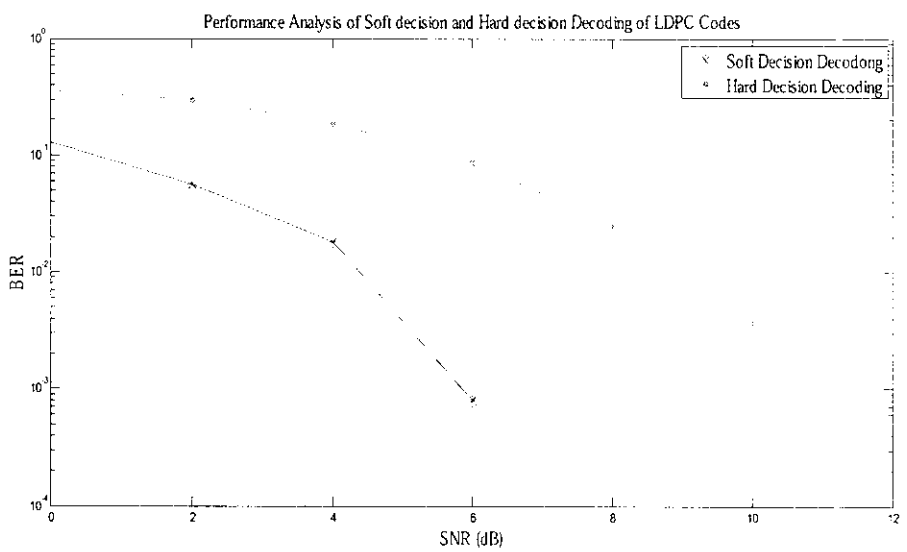

Figure 5.1: Performance Analysis of Soft Decision and Hard Decision Decoding

Figure 5.1 shows the simulated bit error rate (BER) versus Signal to Noise Ratio (SNR) for hard decision decoding algorithm and soft decision decoding algorithm. Soft decision decoding algorithm gives better performance compared to hard decision algorithm.

Next the performance of convolutional codes and LDPC codes are analysed. Here we have used a (12,9) LDPC codes and its performance is compared with

convolutional codes with PSK modulation schemes and QAM modulation schemes and the result is given Figure 5.2.

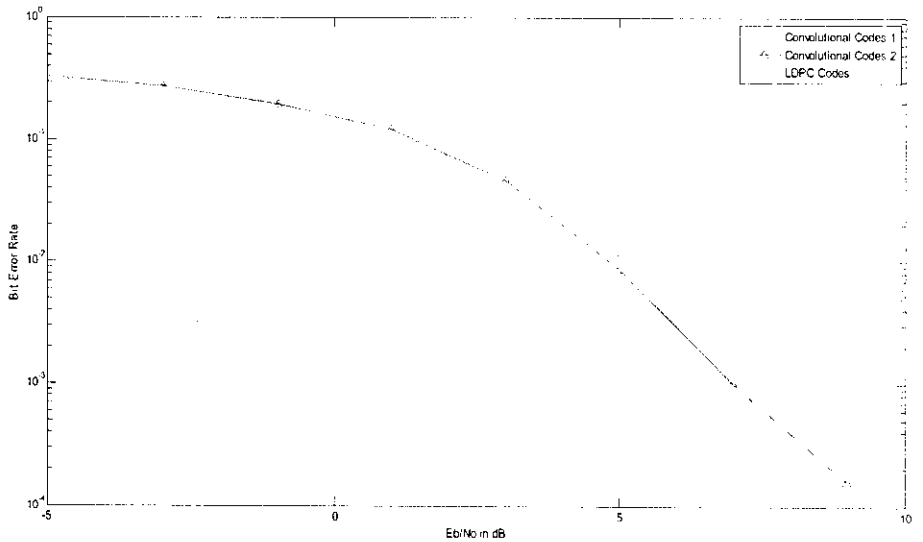

Figure 4.2: Performance Comparison of Convolutional Codes and LDPC Codes

Next the analysis for different code rates is given in Fig5.3. Here we are taking 50,000 message bits are transmitting. Here the soft decision sum product algorithm is used. Low code rate gives good BER.

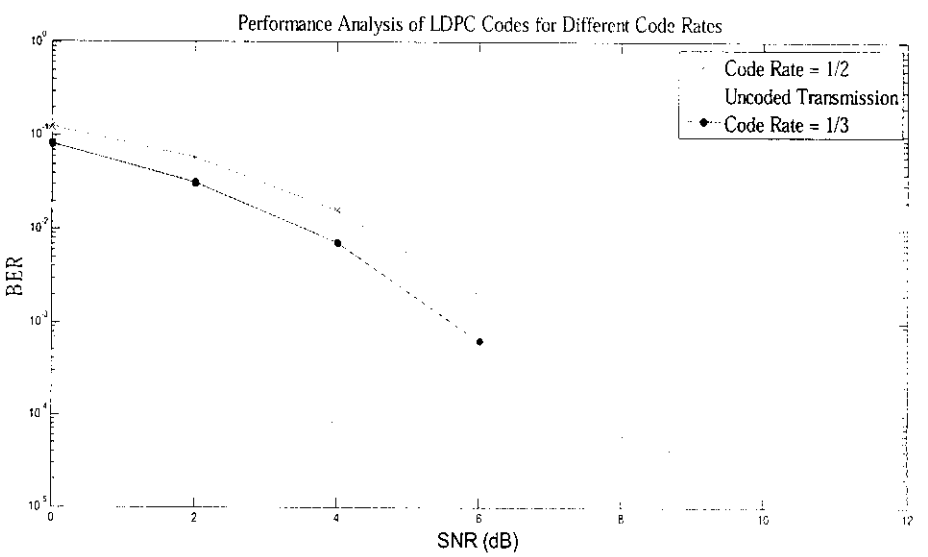

Figure 5.3: BER Performance of LDPC Codes for Various Code Rates

The performance of LDPC codes also depends up on the number of iterations that we are using in the message passing algorithm. The performance analysis based

on the number of iterations, is given in Figure 5.4. We can choose a specific number of iterations based on the hardware, power and computational resources available.

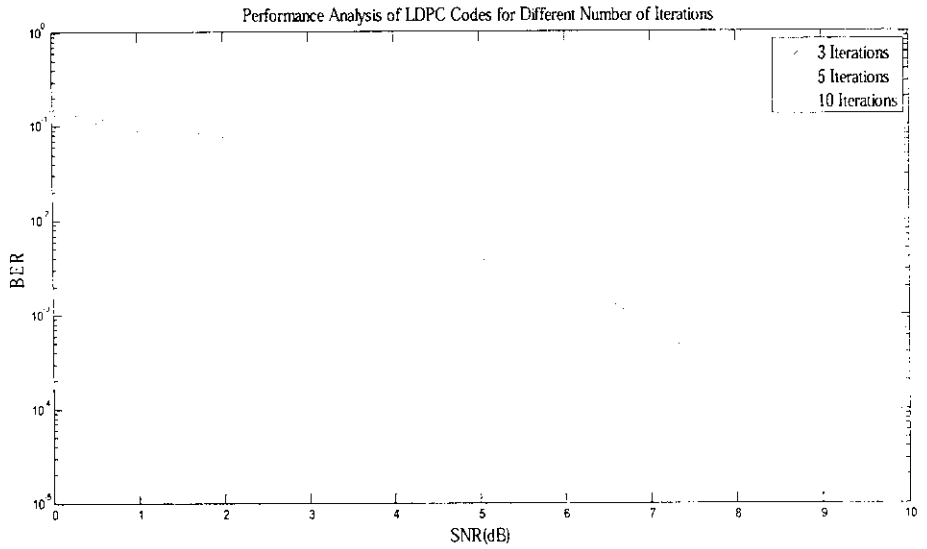

Figure 5.4: Decoding Performance of LDPC Codes with Various Number of Iterations

Performance of the decoding algorithm for various iterations is shown in Fig.5.5 shows the BER performance for various methods of generation of H matrix. Quasi cyclic codes and repeat accumulate codes have good BER performance.

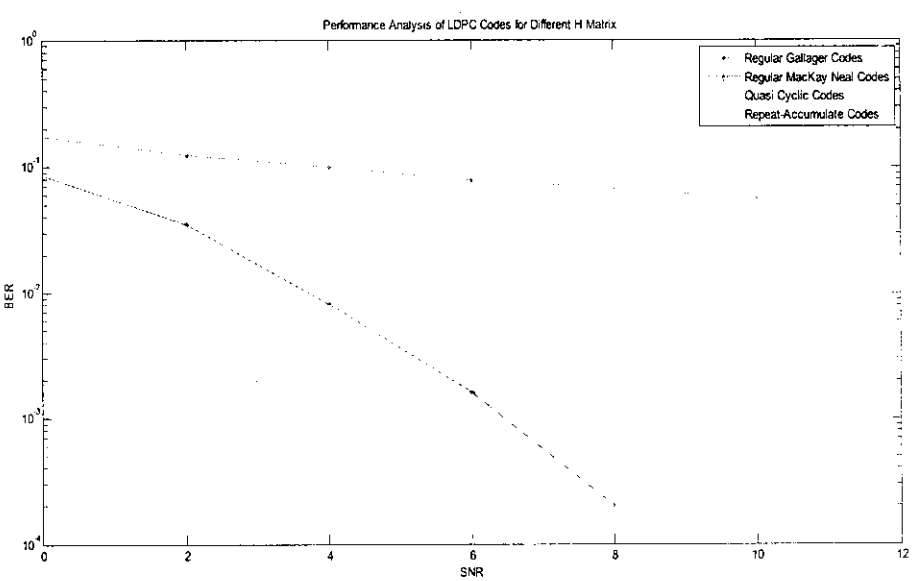

Figure 5.5: Decoding performance of LDPC codes for different H matrix generation

Next the effect of changing the code length N is analysed. According to the results in the figure 5.6, it's evident that when the code length N is increased, the performance of the code is increasing.

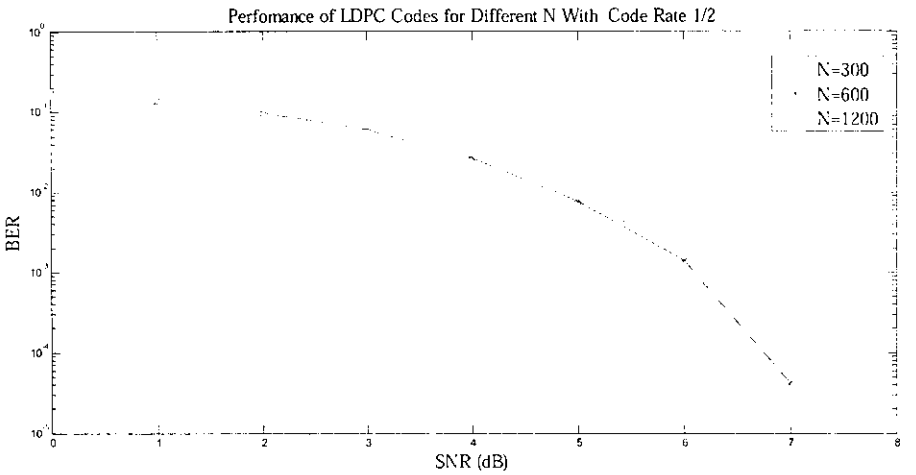

Figure 5.6: Performance of LDPC Codes for Different N with Code Rate 1/2

Finally the simulation result for the proposed method of generating LDPC codes is given. Here we are comparing the performance of the conventional quasi-cyclic method and the proposed method for generating the H matrix. As the simulation results indicated in figure 5.7, the conventional method provides slightly better performance by a small margin. But considering the reduction in the implementation complexity provided by the proposed method, we can use the method for generation of LDPC codes with a very small trade off in BER performance. Here we are taking 2 arrangements for the mirror image of I matrix. Both are giving almost the same performance. So we can choose the convenient one for the generation.

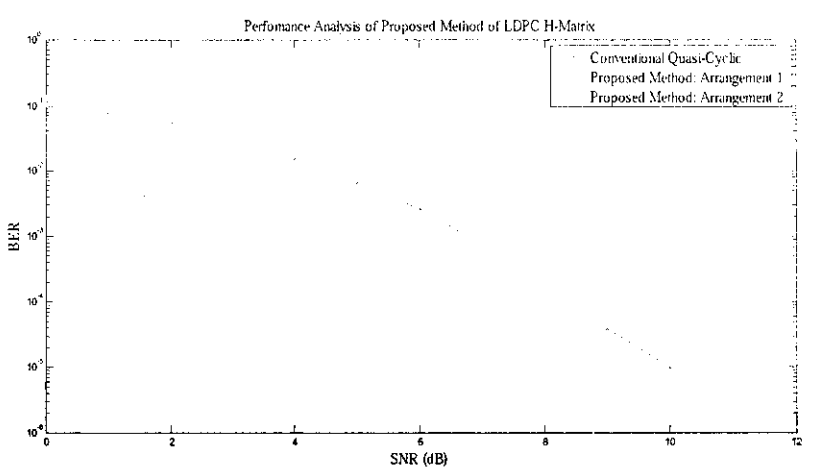

Figure 5.7: Performance of Proposed Method for LDPC Codes Generation

The Figure 5.8 the performance of the new method with respect to different code length N is shown. As the code length increases the performance increases slightly in the case of medium length codes.

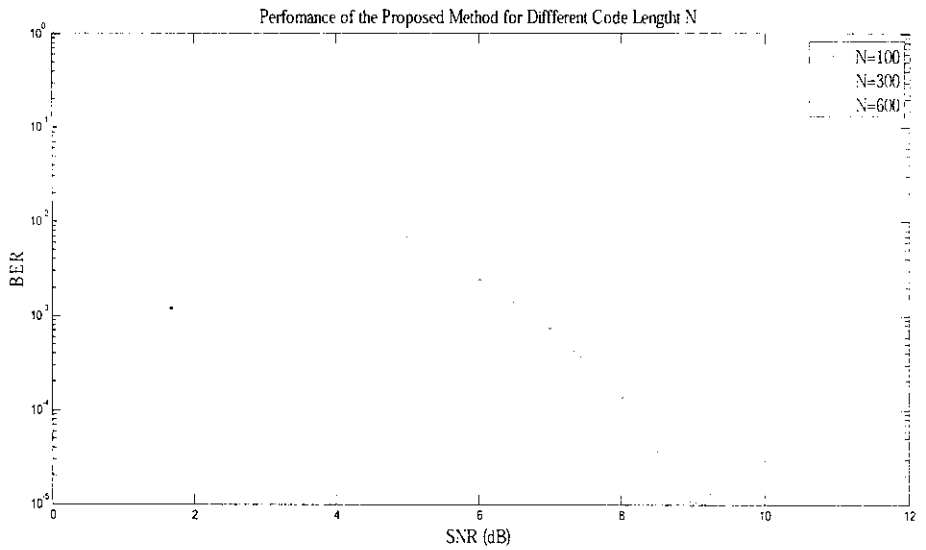

Figure 5.8: Variation of Performance of the Proposed Method With Code Length N

At the end of the brief analysis we can see that repeat accumulate codes give better performance compared to other codes. Quasi cyclic codes give a comparable performance with a very less complexity. The variation in performance with respect to change in codelength N is very small for low and average code lengths.

CHAPTER 6

CONCLUSION

In this paper, some important design parameters of LDPC codes are analyzed in the point of view of future wireless standards. Then the fact that the parity check matrix of the LDPC codes are sparse matrices, allows the hardware implementation of the decoder to be less complex. The process of passing extrinsic information between bit node and check node allows the implementation of the concept of parallel processing. Since the future wireless standards are designed for a wide band of application in a single hardware platform, it needs to be adapted to different kinds of decoding algorithm, different code rates, different code lengths and different number of iterations according to the available computational capability. The QC-LDPC can be effectively used for the multi-rate and multi-length wireless applications. At the end of the analysis, a modified approach for the generation of regular LDPC codes is proposed. The proposed method is a variant of the conventional QC-LDPC codes. The modified method gives comparable performance which is less than the conventional QC-LDPC by a slight margin but can be implemented with considerable amount of reduction in complexity. The analyses and the simulation results will give an insight to the design of LDPC codes for the future wireless standards.

FUTURE SCOPE

The future wireless standards demands for very high code rates and data rates at large code lengths. So LDPC codes can plan an important role. The carefully constructed irregular LDPC codes can produce very good BER. So this area needs to be attended with more research. And also new methods of generating less complex but efficient regular codes can be explored more.

REFERENCES

[1] Gallager, "Low-density parity-check codes," IRE Transactions on Information Theory, vol. IT-8, no.1, pp. 21–28, January 1962.

[2] Seo Sangwon; Mudge Trevor; Zhu Yuming; Chakrabarti, Chaitali, "Design and Analysis of LDPC Decoders for Software Defined Radio", Signal Processing Systems, 2007 IEEE Workshop on Digital ObjectIdentifier:10.1109/SIPS.2007.4387546 Publication Year: 2007 , Page(s): 210 - 215

[3] T. Richardson and R. Urbanke, "An introduction to the analysis of iterative coding systems", *Codes, Systems, and Graphical Models*, IMA Volume in Mathematics and Its Applications, pages 1-37. Springer, 2001.

[4] Todd K. Moon, "Error Correction Coding, Mathematical Methods and Algorithms", A John Wiley & Sons, Inc, Publication, New Delhi, India, Ed.2006.

[5] Y.Kou, S.Li and M.P.C. Fossier, Low Density Parity check codes: construction based on finite Geometries, IEEE Globe Com 2000, San Fransisco, CA,2(7):825 – 829, Nov.2000.

[6] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity approaching low-density parity-check codes," IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[7] 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, no. 2, pp. 599–618, February 2001.

[8] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," Electron. Lett, vol. 32, no. 18, pp. 1645–1646, March 1996, reprinted Electron. Lett, vol. 33(6), pp.457–458, March 1997.

[9] D. J. C.Mackay, "Good error correcting codes basedon very sparse matrices," IEEE Trans. Inform. Theory, vol. 45, pp. 399-431, March 1999.

[10] R. M. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inform. Theory, vol. IT-27, no. 5, pp. 533–547, September 1981.

[11] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," IEEE Trans. Inform. Theory, vol. 47, no. 2, pp. 638–656, February 2001.

[12] M. Yang, Y. Li, and W. E. Ryan, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," IEEE Trans. Commun., vol. 52, pp. 564-571, Apr. 2004.

[13] Sarah J. Johnson.; School of Electrical Engineering and Computer Science, The University of Newcastle, Australia.

[14] F.R. Kschischang, B.J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm", *IEEE Trans. Inform. Theory*, vol 47, pp. 498-519, Feb. 2001.

[15] Chiurtu N.; Gasser L.; Roud P.; Rimoldi B.; "Software-defined radio implementation of multiple antenna systems using low-density parity-check codes", Wireless Communications and Networking Conference, 2005 IEEE , Volume: 1 Digital Object Identifier: 10.1109/WCNC.2005.1424556 Publication Year: 2005 , Page(s): 527 - 531 Vol. 1

[16] "Design of repeat-accumulate codes for iterative detection and decoding," IEEE Trans. Signal Processing, vol. 51, pp. 2764–2772, Nov.2003.

[17] N. Wiberg, "Codes and decoding on general graphs", Ph.D. Dissertation, Linkoping University, Sweden, 1996.

[18] R. Narayanaswami, "Coded modulation with low-density parity-check codes," Master's thesis, Dept. Elect. Eng., Texas A&M Univ., College Station, TX, 2001.

[19] T. Richardson and R. Urbanke, "The renaissance of Gallager's lowdensity parity-check codes," IEEE Commun. Mag., vol. 41, no. 8, pp.126–131, Aug. 2003.

[20] M.P.C. Fossorier and M. Mihaljevic, "Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, May 1999.

[21] Jin sha,"Multi Gb/s LDPC Code design and implementation" vol.17,no 2 February 2009

[22] M. G. Luby, M. Mitzenmacher, A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation", Technical Report TR-97-044, Berkeley, CA, 1997.

[23] David J. C. MacKay, Simon T. Wilson, and Matthew C. Davey, "Comparison of constructions of irregular gallager codes", IEEE Transactions on Communications, vol. 47, pp. 1449-1454, Oct. 1999.

[24] S. Ten Brink, G. Kramer and A. Ashikhmin. "Design of low-density parity-check codes for modulation and detection", IEEE Trans. Commun., vol. 52, pp. 670-678, April 2004.

[25] M. Chiani and A. Ventura, "Design and performance evaluation of some high-rate irregular low-density parity-check codes," in IEEE GLOBECOM, 2001, vol. 2, pp. 990-994.

ICCOS '11

International Conference on "Communication and Signal Processing"

Karunya UNIVERSITY

(Karunya Institute of Technology and Sciences)
Declared as Deemed to be University Under sec. 3 of the UGC Act, 1956
Karunya Nagar, Coimbatore 641 114. India

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that Dr/Mr/Ms/Mrs. Sarath. R of Kumaraguru college of Technology, Coimbatore has ~~participated~~ /presented a paper titled "Performance analysis of iterative LDPC decoders" in the International Conference on "Communication and Signal Processing (ICCOS '11)" on 17th & 18th March 2011 organized by the School of Electrical Sciences, Department of Electronics and Communication Engineering, Karunya University, Coimbatore, India.

Dr. A. Ravi Sankar
Convener

Dr. (Mrs.) Anne Mary Fernandez
Patron

Dr. Paul P. Appasamy
Patron

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE - 641049, TAMIL NADU, INDIA.

CERTIFICATE

CITEL 2011

This is to certify that Mr./~~Ms~~. R. SARATH , ME [COMMUNICATION SYSTEMS] KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE has ~~attended~~ / presented a paper titled ANALYSIS OF LDPC CODES FOR FUTURE WIRELESS STANDARDS in the 3rd National Conference on **COMMUNICATION, INFORMATION AND TELEMATICS (CITEL 2011)** on 3rd & 4th March 2011, organized by the Department of Electronics and Communication Engineering, Kumaraguru College of Technology, Coimbatore.

3/3/11

Dr. Rajeswari Mariappan
HOD - ECE

Dr. S. Ramachandran
Principal

Dr. J. Shanmugam
Director