HIGH THROUGHPUT QUASI-CYCLIC LDPC DECODER

By

ANU JOSE

Reg. No: 1020106001

of

KUMARAGURU COLLEGE OF TECHNOLOGY

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

COIMBATORE - 641 049

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

APPLIED ELECTRONICS

APRIL 2012

BONAFIDE CERTIFICATE

Certified that this project report titled " **HIGH THROUGHPUT QUASI-CYCLIC LDPC DECODER**" is the bonafide work of **Ms.ANU JOSE [Reg.No: 1020106001]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report of dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Project Guide	**Head of the Department**
Ms.G.Amirtha Gowri	Dr.Rajeswari Mariappan

The candidate with university Register no. 1020106001 is examined by us in the project viva-voce examination held on ……

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**, **Ph.D**., Kumaraguru College of Technology, for his kind support and for providing necessary facilities to carry out the work.

I express my sincere thanks to our beloved Principal **Dr.S.Ramachandran**, **Ph.D**., Kumaraguru College of Technology, who encouraged me in each and every steps of the project work.

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

In particular, I wish to thank with everlasting gratitude to the project coordinator **Ms.R.HEMALATHA**, **M.E**., **Assistant 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 my heartfelt thanks to my project guide **Ms.G.AMIRTHA GOWRI**, **M.E**.,**(Ph.D)., Associate Professor**, Department of Electronics and Communication Engineering, throughout the course of this project work and I wish to convey my deep sense of gratitude to all the teaching and non-teaching staffs 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

Electronics communications have increasingly become more and more the centre of our day to day life. As a result methods that allow reliable transmission of information will become more and more important. Information Theory and Error Correction codes are the research areas that study how to achieve such a goal. Many error correction codes have been presented in the past but in recent years Low Density Parity Check Codes (LDPC) has imposed themselves as the best candidate to solve the problem. LDPC codes are a class of codes that can achieve reliable communication while keeping the complexity of encoder and decoder implementation tractable. If the generated LDPC codes have the property that when there is a single shifts in a codeword it will generate another codeword they are called Quasi-Cyclic LDPC codes. In this project an analysis on serial, parallel and partially parallel architectures of decoder has been done and based on it the partially parallel architecture optimizes area and speed throughput. It is proposed to design a high throughput partially parallel architecture for QC-LDPC decoder based on min sum algorithm. This approach reduces the complexity of the architecture and also reduces the storage memories for message passing. Regularity in parity check matrices using QC-LDPC makes them well suited for VLSI implementation especially in cases where there is heavy use of parallelism. They are used as error correcting code in the new DVB-S2 standard for satellite transmission of digital television and also in applications where reliable and high efficient information transfer is required.

TABLE OF CONTENTS

LIST OF FIGURES.

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Communication systems transmit data from source to destination through a channel or medium such as air, wire line and optical fiber. 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. Forward Error Correction Code (FEC) is used for error correction easily without data need to be retransmitted. Retransmission will result in delay, cost and wastes system throughput. Several error correction codes have been developed to improve the reliability of the data transfer. Forward Error Correction Codes (FEC) includes Viterbi, convolution codes, Bose Chaudhuri Hocquenghen (BCH) codes, Reed Solomon (RS) codes, turbo codes and low density parity check codes (LDPC).[1]

Low Density Parity Check (LDPC) codes are a class of linear block codes, shows good error correcting performance approaching Shannon□s limit. Good error correcting performance enables efficient and reliable communication. They were first introduced by Gallager in his Ph.D. thesis in 1960. But due to the computational complexity in implementing encoder and decoder for such codes and the introduction of Reed-Solomon codes, they were mostly ignored until about ten years ago. They 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 in his seminal 1948 paper. The relative quiescence of the coding field was utterly transformed by the introduction of turbo codes, proposed by Berrou, Glavieux and Thitimajshima, 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.[2]

The future wireless standards need 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 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. The name Low Density comes from the characteristic of their parity-check matrix which contains only a few 1□s in comparison to the number of 0□s. Their main advantage is that they provide a performance which is very close to the capacity for a lot of different channels and linear time algorithms for decoding. Furthermore they are suited for implementations that make heavy use of parallelism. They use parallel decoding and the simple computation operations are the main advantage of the LDPC codes.

1.2 PROJECT GOAL

The project aim is to design and implement a high throughput quasi-cyclic LDPC decoder in FPGA using min sum algorithm. In this project an analysis on serial, parallel and partially parallel architectures of decoder has been done and based on it the partially parallel architecture optimizes area and speed throughput. It is proposed to design a high throughput partially parallel architecture for QC-LDPC decoder. Pipelining is introduced and the power, throughput and delay are compared. [1]

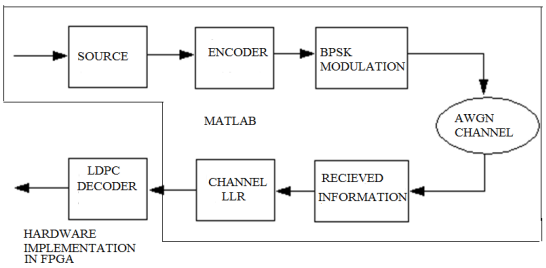

Figure 1.1.Overall block Diagram

1.3 SOFTWARES USED

- ModelSim SE 6.3f
- Xilinx ISE 9.2i
- Matlab R2008b

1.4 ORGANIZATION OF THE REPORT

- **Chapter 2** discusses about the LDPC codes.
- **Chapter 3** discusses the encoding and decoding of LDPC codes
- **Chapter 4** discusses the min sum algorithm
- **Chapter 5** discusses the decoder architecture design
- **Chapter 6** explains the applications
- **Chapter 7** explain the results and discussions
- **Chapter 8** discusses conclusion and future work

CHAPTER 2

LOW DENSITY PARITY CHECK CODES

2.1 PARITY CHECK CODES

A basic communication system is composed of three parts: a transmitter, channel, and receiver. Transmitted information is usually corrupted due to noise and channel distortion. To correct these errors, redundancy coding is intentionally introduced, and the receiver employs a decoder to make corrections based on the redundancy. [1]LDPC codes is the most popular error-correcting control codes (ECC) which shows results very close to the Shannon limit. It also has lower error floor and less computation requirement. As a matter of fact, LDPC code has been considered in many industrial standards, such as WLAN (802.11n), WiMAX (802.16e), DVB-S2, CMMB, and 10GBaseT (802.3an) systems. [3]- [5]

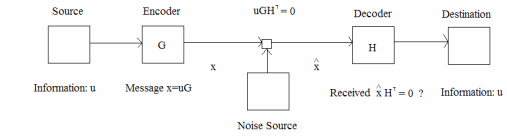

Figure 2.1: Transmitter Receiver model.

In communication systems the noise are added when the messages are passed over the channel. So different error correcting methods are introduced. Parity check method of error correction is one of the simplest methods. In parity check method we will only consider binary messages and so the transmitted messages consist of strings of 0□s and 1□s. The essential 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:

$$c = [c1\ c2\ c3\ c4\ c5\ c6\ c7\ c8] \quad (2.1)$$

Where each ci is either 0 or 1, and every codeword satisfies the constraint

$$c1 \oplus c2 \oplus c3 \oplus c4 \oplus c5 \oplus c6 \oplus c7 \oplus c8 = 0 \quad (2.2)$$

Equation (2.2) is called a parity-check equation, in which the symbol ⊕ 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 (2.2), 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.

2.2 LDPC CODES

Low Density Parity Check (LDPC) codes are error checking and correcting codes, which show good error correcting performance approaching Shannon□s limit. The matrix „H□ is called a parity-check matrix. In LDPC codes the H matrix is sparse. The number of ones in the matrix is lesser compared to the number of zeroes. 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

code with k

2.3 REPRESENTATION OF LDPC CODES

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

- Matrix representation.
- Graphical representation.

2.3.1 MATRIX REPRESENTATION

The parity check constrain can be represented in the form of matrix with 1□s and 0□s. The matrix given in eqn (2.3) is a parity check matrix with dimension n × m for a (6, 2) code. We can now define two numbers describing these matrixes. W_r, (row weight) represent the number of 1□s in each row and W_c (column weight) represent number of ones in each column. For a matrix to
be called low-density the two conditions $W_c << n$ and $W_r << m$ must be satisfied. [6], [7].

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (2.3)$$

The matrix H is called a 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□s the parity-check matrix is an m × n binary matrix. The matrix is called sparse since the number of ones in the matrix is less compared to the number of zeroes. In matrix form a string y = [c1 …….cn] is a valid codeword for the code with parity-check matrix H if and only if it satisfies the matrix equation $Hy^T = 0$.

Matrix G is called the generator matrix of the code. The message bits are conventionally labeled by u = [u1, u2, …, uk], where the vector „u□ holds the „k□ message bits. Thus the codeword c corresponding to the binary message u = [u1u2u3] can be found using the matrix equation c = uG. For a binary code with „k□ message bits and length „n□ code words the generator matrix, G, is a k×n binary matrix. The ratio k/n is called the rate of the code. A

message bits contains 2^k code words. These code words are a subset of the total possible 2^n binary vectors of length n.

2.3.2 GRAPHICAL REPRESENTATION

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 message passing between the nodes and the LDPC decoding

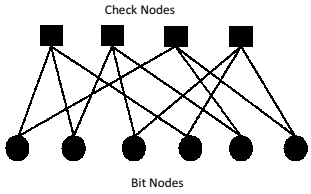

Figure 2.2: Tanner graph representation of parity check matrix

Tanner graphs are bipartite graphs. 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 two types of nodes in a Tanner graph are called variable nodes (v-nodes) and check nodes (C-nodes). Figure 2.2 is an example for such a Tanner graph and represents the same code as the matrix in (2.3). The creation of such a graph is 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 fi is connected to variable node c_j if the element h_{ij} of H is a 1.

The graph representation is analogous to a matrix representation by looking at the adjacency matrix of the graph, let H be a binary m x n matrix in which the entry (i; j) is 1 if and only if the i^{th} check node is connected to the j^{th} message node in the graph. Then the LDPC code defined by the graph is the set of vectors c = (c1…. cn) such that $H*c^T = 0$. The matrix H is called a parity check matrix for the code. Conversely, any binary m x n matrix gives rise to a

bipartite graph between „n message and „m check nodes, and the code defined as the null space of H is precisely the code associated to this graph. Therefore, any linear code has a representation as a code associated to a bipartite graph (note that this graph is not uniquely defined by the code). However, not every binary linear code has a representation by a sparse bipartite graph if it does, then the code is called a low-density parity-check (LDPC) code. The sparsity of the graph structure is key property that allows for the algorithmic efficiency of LDPC codes

2.4 REGULAR AND IRREGULAR LDPC CODES

An LDPC code is called regular if W_c is constant for every column and $Wr = W_c * (n/m)$ is constant for every row. The example matrix from equation (2.3) is regular with $W_c = 2$ and $W_r = 3$. It s also possible to see the regularity of this code while looking at the graphical representation. There is the same number of incoming edges for every v-node and also for all the c-nodes. If the numbers of 1 s in each row or column aren t constant, then the code is called an irregular LDPC code. The parity check matrix of LDPC codes is either regular or irregular. But the irregular LDPC parity check matrix gives better performance [8], [9].

Advantages of LDPC codes are as follows:

- LDPC codes have error floor at very lower BER.
- The LDPC codes are not trellis based.
- LDPC codes can be made for any arbitrary length according to the requirements.
- Rate can be changed for LDPC codes by changing different rows.
- They have low decoding complexity.
- LDPC codes are easily implementable.
- LDPC codes have good block error performance.

2.5 QUASI-CYCLIC LDPC CODES

An (n, k) linear block code of dimensions $n = mn_o$ and $k = mk_o$, is called Quasi-Cyclic if every cyclic shift of a codeword by n_o symbols yields another codeword. Quasi-Cyclic LDPC (QC-LDPC) codes have gained the most attention due to the regularity in their parity check matrices which is well suited for VLSI implementation. Moreover, QC-LDPC codes can provide comparable error correction performance compared with random LDPC codes. As an example, consider the following generator matrix of an (8, 4) binary linear code [10].

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \qquad (2.4)$$

This code is Quasi-Cyclic with no = 2, since every row of G is the same as the previous with a cyclic shift of two positions.

QC-LDPC code is a kind of LDPC codes whose parity check matrices consist of the blocks of circularly shifted identity matrices. A codeword is directly generated by a parity check matrix. Thus, QC-LDPC code is convenient to choose the parity check matrix ***H*** as an array of shifting fact (q × q) circulant permutation sub matrices, where some of the sub matrices are given an identity matrix or a zero matrix.

Recently, several quasi-cyclic (QC) LDPC codes are proposed to overcome these problems, and they can further be separated into two categories: one is based on finite geometries and the other is based on circulant permutation matrices .Both of them can produce medium length codes with good minimum distance. Encoding can be done efficiently by using shift registers as far as hardware implementation is concerned.

The results show that these codes perform quite well compared to random LDPC codes at moderate block lengths. Furthermore, the required memory for storing these QC-LDPC codes can be reduced by a factor 1/p, while p×p circulant permutation matrices are employed. One new method to construct the QC-LDPC codes based on the circulant matrices is proposed, and these circulant matrices are permuted according to the one-coincidence sequences (OCSs).

This simple method can be used to construct the (j, k)-regular QC-LDPC codes in variety of block lengths and rates with no 4-cycles. Since the cycles of short length may degrade the performance of LDPC codes, it is necessary to ensure that the Tanner graph of the LDPC codes is free of cycles of length 4 and hence has girth at least 6.

The algebraically constructed QC LDPC codes perform quite well compared to random regular LDPC codes at short to moderate block lengths, while for long block lengths, a randomly constructed regular LDPC code typically performs somewhat better. Moreover, random regular LDPC codes of column weight j>3 are asymptotically good.

2.6 CONSTRUCTION OF QC-LDPC CODES FROM CIRCULANT PERMUTATION MATRICES

A simple method to design a (j, k) regular QC-LDPC code is to construct the preliminary matrix Y by constructing the two sequences $\{a_0, a_1, \ldots a_{j-1}\}$ and $\{b_0, b_1, \ldots b_{k-1}\}$ with elements randomly selected from GF (p)(p is prime and p>2).The matrix Y is represented as

$$Y = \begin{bmatrix} y_{0,0} & y_{0,1} & \cdot & \cdot & y_{0,k-1} \\ y_{1,0} & y_{1,1} & \cdot & \cdot & y_{1,k-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ y_{j-1,0} & y_{j-1,1} & \cdot & \cdot & y_{j-1,k-1} \end{bmatrix} \qquad (2.5)$$

where the (u, v)- th element of Y can be calculated by the following quadratic congruential equation for fixed parameter d:

$$y_{u,v} = \left[d(a_u + b_v)^2 + e_u + e_v \right] \bmod(p) \qquad (2.6)$$

where , d ∈ { 1,2,…p-1} and e_u, e_v {0,1,….p-1}.

Then the proposed parity check matrix H can be constructed by using the following equation,

$$H = \begin{bmatrix} I(y_{0,0}) & I(y_{0,1}) & \cdot & \cdot & I(y_{0,k-1}) \\ I(y_{1,0}) & I(y_{1,1}) & \cdot & \cdot & I(y_{1,k-1}) \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ I(y_{j-1,0}) & I(y_{j-1,1}) & \cdot & \cdot & I(y_{j-1,k-1}) \end{bmatrix} \qquad (2.7)$$

Where I(x) is a p × p identity matrix with rows cyclically shifted to the right by x positions.

For example I (1) is represented as follows:

$$I(1) = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdot & \cdot & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \cdot & \cdot & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \cdot & \cdot & \mathbf{0} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{0} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdot & \cdot & \mathbf{0} \end{bmatrix} \qquad (2.8)$$

Hence, the resulting H, which has j ones in each column and k ones in each row, represents a (j, k)-regular LDPC code. This LDPC code is also an [N, K] regular LDPC code, where N = kp is the block length of the QC-LDPC code and K is the number of message bits.

Since the cycles of short length may degrade the performance of LDPC codes, it is necessary to ensure that the Tanner graph of the LDPC codes is free of cycles of length 4 and hence has girth at least 6. It is easy to prove that the parity check H constructed by the proposed method can satisfy this.

Example 1: A [155, 64] QC-LDPC code (p = 31).

Let j=3 and k=5.First construct sequences and then by assuming d=1 and σ_w, σ_y =0 the following parity check matrix can be formed by substituting the above parameters in eqns (2.5) and (2.6).Therefore the matrix can be represented as

$$H = \begin{bmatrix} I(7) & I(4) & I(20) & I(28) & I(16) \\ I(5) & I(5) & I(2) & I(18) & I(0) \\ I(18) & I(25) & I(19) & I(2) & I(1) \end{bmatrix} \quad (2.9)$$

where I(x) is a 31 × 31 identity matrix with rows shifted cyclically to the right by x positions.

Example 2: A [305, 64] QC-LDPC code (p = 61).

$$H = \begin{bmatrix} I(38) & I(34) & I(12) & I(47) & I(46) \\ I(17) & I(12) & I(15) & I(13) & I(27) \\ I(25) & I(32) & I(40) & I(55) & I(11) \end{bmatrix} \quad (2.10)$$

Example 3: A [905, 64] QC-LDPC code (p = 181).

$$H = \begin{bmatrix} I(133) & I(99) & I(144) & I(145) & I(148) \\ I(168) & I(165) & I(82) & I(176) & I(172) \\ I(46) & I(62) & I(29) & I(180) & I(125) \end{bmatrix} \quad (2.11)$$

2.7 IMPORTANT DESIGN PARAMETERS

Design of QC-LDPC codes involves many parameters which are often determined in consideration of the target application:

Code size:

The code size specifies the dimensions of the parity check matrix (M x 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 then shorter ones. Long codes therefore are desirable to have good performance. However, their hardware implementation requires more resources (memory plus professing nodes).[12]

Code Weights and Rate:

The rate of the code R, is the number of information bits over the total number of bits transmitted. It is expressed as (N – M / N).Higher row and column weights results 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 in block resulting in higher 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 rate codes may have poor performance with a small number of connections. LDPC codes with minimum 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 when 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.

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.

Girth:

Cycles in the Tanner graph lead to correlations in the marginal probabilities passed by the sum-product decoder; the smaller the cycles the fewer the number of iterations that are correlation free. Thus cycles in the Tanner graph affect decoding convergence, and the smaller the code girth, the larger the effect. Definite performance improvements can be obtained by avoiding 4-cycles and 6-cycles from LDPC Tanner graphs but the returns tend to diminish as the girth is increased further.

Expansion:

A related concept to the graph girth is the graph expansion. In a good expander every subset of vertices has a large number of neighbors that are not in the subset. More precisely, any subset S of bit vertices of size m or less is connected to at least ϱ |S| constraint vertices, for some defined m and ϱ. If a Tanner graph is a good expander then the bit nodes of a small set of erroneous codeword bits will be connected to a large number of check nodes, all of which will be receiving correct information from an even larger number of the correct codeword bits. Using only a simple hard decision decoding algorithm they proved that a fixed fraction of errors can be corrected in linear time provided that the Tanner graph is a good enough expander.

Code structure:

The structure of the 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 interconnections. Structured codes on the other hand have a known inter connection pattern. Many methods have been developed for constructing those type of codes.

In this chapter, various LDPC code construction approaches have been proposed. All these approach construct a low-density parity-check matrix $\boldsymbol{H}$. Although the parity check matrices of LDPC codes are sparse by code construction, the generator matrices are usually high density matrices. Therefore, the direct encoding approach, $\boldsymbol{c} = \boldsymbol{uG}$, has the encoding complexity of $O(N^2)$, where N is the block length of an LDPC code. To reduce the encoding complexity, various efficient encoding method has been proposed. Each one is usually only suitable for a specific class of LDPC codes. The encoding details for QC-LDPC codes are presented in the coming chapter.

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.[13],[14].

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:

$$H' = [A, I_{N-K}] \tag{3.1}$$

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:

$$G = [A, I_{N-K}] \tag{3.2}$$

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

$$c = uG \tag{3.3}$$

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

The class of decoding algorithms used to decode LDPC codes is collectively termed message-passing algorithms (MPA) 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 code word 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 decoding algorithms are normally classified in to two they are hard decision algorithm and Soft decision algorithms. Soft decision algorithm which is based on the concept of belief propagation gives better decoding performance and therefore is a preferred method. The decoding can be done iteratively since the parity check matrix is sparse the LDPC codes have less complexity compared with Turbo codes [16]

3.3 MESSAGE PASSING ALGORITHM

The messages passed along the Tanner graph edges are straightforward: a bit node sends the same outgoing message M to each of its connected check nodes. This message, labeled Mi. for the i-th bit node, declares the value of the bit „1" or „0" . The check nodes send back

different messages to each of their connected bit nodes. This message, labeled $E_{j,i}$ for the message from

the j-th check node to the i-th bit node, declares the value of the i-bit „1" or „0" as determined by
the j-th check node. If the bit node of an erased bit receives an incoming message which is „1" or „0" the bit node changes its value to the value of the incoming message. This process is until some maximum number of decoder iterations has passed and the decoder gives up.

The advantages of LDPC decoding algorithms are that they will use tanner graph and iterative decoding methods. They consist of two sets of nodes check nodes and bit nodes. They both are in different levels, Connected each other. Within one level there is no connection so the parallel processing can be done easily. This will speed up the decoding process

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

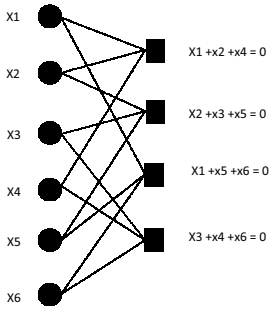

Figure 3.1: Representation of parity check constrain of LDPC codes

The notation B_j is used to represent the set of bits in the j^{th} parity-check equation of the code. So for the parity check constrain shown in figure 3.1 we have

$B_1 = \{1, 2, 4\}$ $B_2 = \{2, 3, 5\}$ $B_3 = \{1, 5, 6\}$ $B_4 = \{3, 4, 6\}$.

Similarly, we use the notation A_i to represent the parity-check equations which check on the i^{th} bit of the code. So for the parity check constrain shown in figure 3.1 we have

$A_1 = \{1, 3\}$ $A_2 = \{1, 2\}$ $A_3 = \{2, 4\}$
$A_5 = \{1, 4\}$ $A_5 = \{2, 3\}$ $A_6 = \{3, 4\}$.

Algorithm outlines message-passing decoding on the BEC. Input is the received values from the detector, y = [y1, . . . , yn] which can be „1" , „0" , and output is M = [M1, . . . ,Mn] which can also take the values „1" , „0" .

The data are sometime send over the erasure channel. The MPA algorithm will help to find the erased bit. Thus the message passing algorithm (MAP) helps to find out the reassured bits at the decoder. This message passing algorithm can be used in erasure channel where the received bits can be „0" ," 1" or x the unknown bit. The specialty of the channel is that it will receive either a receive either a true bit or bit x. It will not produce an error bit. The unknown bit x can be found out by the message passing algorithm by passing of messages between bit nodes and check nodes

3.4 HARD DECISION DECODING

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.

The bit-flipping decoder can be immediately terminated whenever a valid code word 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 code word is always detected. The bit-flipping algorithm is based on the principle that a code word 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 code word bits. The bit-flipping algorithm applies the hard decision on the received vector, y = [y1, . . . , yn], and output is M = [M1, . . . , Mn].

The steps of the message passing algorithm is given below

Step 1: Initialization
Step 2: Check-node update
Step 3: Variable-node update
Step 4: Decision

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 exor operations are performed. If all the result of the exor 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. So messages are passed between the bit nodes and the check nodes

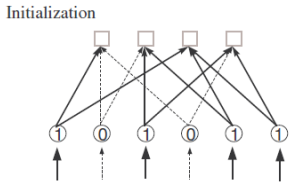

Figure 3.2: Initialization of the bit node

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

$E_{11} = 0$ E31 = 0
$E_{22} = 0$ E12 = 0
$E_{23} = 1$ E43 = 1
$E_{14} = 1$ E44 = 0
$E_{25} = 1$ E35 = 0
$E_{36} = 0$ E46 = 1

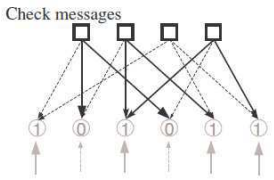

Figure 3.3: Check node message updates

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

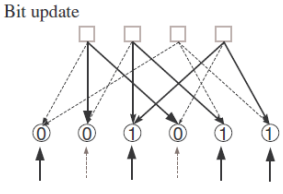

Figure 3.4: Variable-node update

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

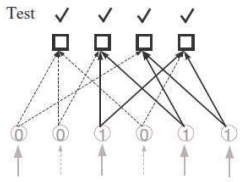

Figure 3.5: Decision making

Bit-flipping decoding of the received string y = [1 0 1 0 1 1]. 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 not. 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.

In the previous case the hard decision algorithm is done with 4 x6 parity check matrix. The code word length is 6 bits. Hard decision decoding is extended up to 32 bit code word; the corresponding parity check matrix dimension is 16 x 32. In the case of LDPC codes as the dimension of the H, the parity check matrix increases, the performance shown by the code is better.[19]

where we use log to mean loge. If p(x = 0) > p(x = 1) then L(x) is positive and the greater the difference between p(x = 0) and p(x = 1), i.e. the more sure we are that p(x) = 0, the larger the positive value for L(x). Conversely, if p(x = 1) > p(x = 0) then L(x) is negative and the greater the difference between p(x = 0) and p(x = 1) the larger the negative value for L(x). Thus the sign of L(x) provides the hard decision on x and the magnitude |L(x)| is the reliability of this decision. To translate from log likelihood ratios back to probabilities we note that

$$p(x=0)=\frac{p(x=1)/p(x=0)}{1+p(x=1)/p(x=0)}=\frac{e^{-L(x)}}{1+e^{-L(x)}} \tag{3.5}$$

And

$$p(x=1)=\frac{p(x=0)/p(x=1)}{1+p(x=0)/p(x=1)}=\frac{e^{L(x)}}{1+e^{L(x)}} \tag{3.6}$$

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 .The sum-product algorithm iteratively computes an approximation of the MAP value for each code bit. Input is the log likelihood ratios for the a priori message probabilities. The a priori probabilities for the Binary Symmetric channel (BSC) are:

$$ri=\log p/(1-p) \text{ if } y_i=1 \quad \text{Or}$$

$$ri=\log(1-p)/(p) \text{ if } y_i=0 \tag{3.7}$$

In sum-product decoding the extrinsic message from check node j to bit node i, E_{ji}, is the LLR of the probability that bit i causes parity-check j to be satisfied. [22]

The hard decision decoding algorithm, also known as bit-flipping, does not perform as well as the soft decision counterpart, with performance loses that can reach several dB but it has the advantage of having a low complexity that makes it an interesting option for future very-high-speed communication systems. The idea underlining the algorithm is based on the observation that if a high number of parity check equations containing a certain variable are violated then that variable is probably wrong. The message decoder computes all the check parity equations and then changes any variable node that is contained in a fixed number, or more, of unsatisfied check nodes. The equations are then recomputed with the new values.

3.5 SOFT DECISION DECODING

It is 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. Soft decision algorithm which is based on the concept of belief propagation gives better decoding performance and therefore is a preferred method.

The sum-product algorithm is a soft decision message-passing algorithm. It is similar to the bit-flipping algorithm. But with the messages representing each decision (check met, or bit value equal to 1) now probabilities. 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 posterior probabilities. In the case of sum-product decoding these probabilities are expressed as log-likelihood ratios.

For a binary variable x it is easy to find p(x=1) given p(x=0), since p(x=1) = 1−p(x=0) and so we only need to store one probability value for x. Log likelihood ratios are used to represent the metrics for a binary variable by a single value

$$L(x)=\log\left(\frac{p(x=0)}{p(x=1)}\right) \tag{3.4}$$

$$E_{j,i}=\log\left(\frac{1+\prod_{i'\in B_j,i'\neq i}\tanh Mj,i/2}{1-\prod_{i'\in B_j,i'\neq i}\tanh(Mj,i'/2)}\right) \tag{3.8}$$

The intrinsic message from check node j to bit node i, M_{ji}, is given by,

$$M_{j,i}=\sum_{j'\in Ai,j'\neq j}E_{j',i}+r_i \tag{3.9}$$

The total LLR of the bit stream is

$$Li=\sum_{j\in A_j}E_{j,i}+r_i \tag{3.10}$$

The total LLR can be either positive or negative number. The hard decision is taken. When total LLR is positive the decision is „0" else „1" . The code word is z. Then syndrome calculation is done by s=zH" . When s is zero then z is a valid codeword, and the decoding stops,
returning z as the decoded word. For an AWGN channel the a priori LLRs are given by

$$r_i=4y_i(E_s/N_o) \tag{3.11}$$

The extrinsic LLR and the total LLR calculation are done to find the codeword .

Various steps of Sum Product Decoding algorithm is given below

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

$$E_{j,i} = \log\left(\frac{1+\prod_{i' \in B_j, i' \neq i} \tanh\left(Mj, i'/2\right)}{1-\prod_{i' \in B_j, i' \neq i} \tanh\left(Mf\ i/2\right)}\right)$$

end for

end for

for i = 1 : n do

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

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

end for

for i = 1 : n do

for j ∈ A_i do

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

end for

end for

I = I + 1

end if

until Finished

end procedure

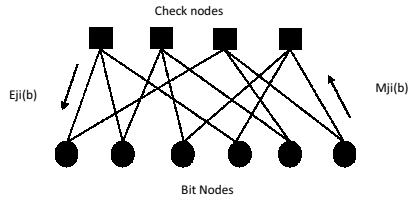

Figure 3.6: Intrinsic and Extrinsic information transfer between bit nodes and the check nodes

The figure 3.6 shows the intrinsic and extrinsic information transfer between bit nodes and the check nodes. Finally the total LLR is calculated and the decision is made.

The aim of sum-product decoding is to compute the maximum a posterior probability (MAP) for each codeword bit., which is the probability that the i-th codeword bit is a 1 conditional on the event N that all parity-check constraints are satisfied. The extra information about bit i received from the parity-checks is called extrinsic information for bit i. 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. Sum Product decoding algorithm results in complexity in computations and much execution time. So we go for Min sum algorithm which is a low complexity approximation of sum product algorithm.

CHAPTER 4

MIN SUM ALGORITHM

LDPC codes can be typically defined by a parity check matrix H. The symbol N represents the length of the block (i.e. the number of bits in the codeword), while the symbol M represents the number of parity checks in the code. The rate of such a code is thus (N-M)/N. The typical LDPC decoding algorithm is the sum product algorithm which has two phases. In the first phase, the variable nodes compute updated information which is sent to adjacent check nodes. In the second phase, the check nodes compute updated information based on the new messages from the variable nodes. This update information is then sent back to adjacent variable nodes and the process is repeated over and over again. The modified min-sum decoding algorithm is similar to the sum-product algorithm, with an approximation of check node process. It has some advantages in implementation against the sum-product algorithm, such as less computation and estimation of noise power is unnecessary for an AWGN channel.[24]

Min-sum algorithm reduces the complexity of the check node operations. Also, it converges much faster and also reduces the storage memories for message passing. Minsum decoding algorithm is a low complexity approximation of Sum Product algorithm and it is also computed in two phases: the check node processing and the variable node processing.

Variable Node Operation:

$$L_v = \sum_{m \in M(v)} R_{mv} + I_v \tag{4.1}$$

$$L_{vc} = L_v - R_{cv} \tag{4.2}$$

Check Node Operation:

$$R_{cv} = \prod_{n \in N(c)/v} sign(L_{nc}) \times \alpha \times \min_{n \in N(c)/v} |L_{nc}| \tag{4.3}$$

where, I_v is the input to variable node v, also known as Log Likelihood ratio (LLR), L_{vc} is the output of variable node v going to check node c, M(v)\c denotes the set of check nodes connected to variable node v excluding the check node c, R_{mv} is the output of check nodes going to variable node v.

From the perspective of implementation, the Min-Sum algorithm requires less computation and the noise power estimation is unnecessary for an additive white Gaussian noise (AWGN) channel. Furthermore, the Min-Sum algorithm can help reducing the message storage requirement because the messages transmitted from a check node to adjacent variable nodes have only two possible magnitudes per iteration. However, this advantage is not easy to take in hardware implementation. The memory requirement reduction is achieved, but the hardware is complex, and the decoding latency is extremely high. The efficient use of Min-Sum algorithm in decoding LDPC codes still remains unresolved.

CHAPTER 5

DECODER ARCHITECTURE DESIGN

5.1TYPES OF DECODER ARCHITECTURE

LDPC decoder architecture can be classified into three types: fully parallel, serial and partially parallel. In fully parallel architecture, every check node or variable node corresponds to an individual node processor unit, which usually leads to large hardware cost, complicated routing, and less flexible. The serial architecture applies just one check node processor unit to perform the computation of all check nodes serially. As a result, it will be too slow for most practical applications. For partially parallel architectures, multiple processing units are used to allow proper trade-off between hardware cost and throughput. Also, partially parallel method can exploit the matrix regularity of specially constructed LDPC codes, such as QC-LDPC codes.Besides hardware cost and speed requirement, power consumption is another challenge of LDPC decoder design, especially for mobile applications. In this paper, a hardware efficient low-power LDPC decoder design for QC-LDPC codes based on min-sum algorithm is presented. The partially parallel architecture efficiently reduces the hardware cost and leads to energy-efficient design compared with other architectures.

5.2 PARTIALLY PARALLEL DECODER ARCHITECTURE

A typical partially parallel decoder architecture for a (3,5)-regular QC-LDPC codes is shown in below figure where totally 3×5=15 memory banks are used to store the soft message symbols exchanged between two decoding phases, memory banks {*Z*1, *Z*2,…, *Z*5 are used to store the intrinsic information, and memory banks {*C*1, *C*2,.., *C*5} are used to store the decoded data bits. For QC-LDPC codes, the address generator for each memory bank can be realized with a simple binary counter, which not only simplifies the hardware design, but also improves the circuit speed. In general, each node processing unit takes one clock cycle (assuming dual port memories are used, otherwise two cycles are needed) to complete message updating for one row (or column) of the parity check matrix.

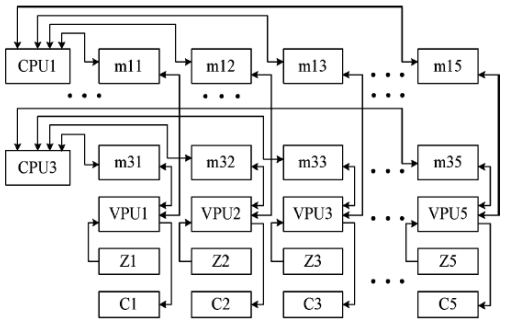

Figure 5.1.The structure of a partially parallel decoder for (3, 5) QC-LDPC codes

There are three major units in this decoder design.

- The posterior Log Likelihood Ratio (LLR) memory which accumulates the check-to-variable messages,
- The extrinsic memory which stores the check-to-variable messages in a shift register, and
- The prior LLR memory.

Each Variable Node (VN) participates in the operations in rows. In each operation, a variable-to-check message is computed by subtracting the corresponding check-to-variable message (of the previous iteration) from the posterior Log Likelihood Ratios (LLR) (of the previous iteration) .The variable-to-check message is converted to the sign-magnitude form before it is sent to the Variable Node Unit (VNU) routers destined for a Check Node(CN). The

returning messages to the VN could be from one of the six CNs. A multiplexer selects the appropriate message based on a schedule. The check node operation is completed in two steps:

1) The CN computes the minimum (min_1) and the second minimum ($min_2 \geq min_1$) among all the variable-to-check messages received from the neighboring VNs, as well as the product of the signs(prd) of these messages;

2) The VN receives min_1, min_2 and computes the marginals, which is followed by the conversion to the two" s complement. The resulting check-to-variable message is accumulated serially to form the posterior LLR Hard decisions are made in every iteration.

To increase the parallelism, we can enable each node processing unit to process the data corresponding to 1-components at multiple rows (or columns) of the parity check matrix at the same cycle. However, this will generally cause memory access conflicts since multiple data accesses per cycle are required for each memory bank. Two efficient approaches proposed were:

I) Partition each memory bank into **p** sub-banks (or called memory segments) where all the soft symbols corresponding to 1-components at **p** adjacent rows of a sub-matrix are stored in **p** different segments.

II) Store soft messages corresponding to **p** adjacent rows of a sub-matrix in one memory entry while utilizing extra buffers to solve the memory access conflict [25].

5.3 CHECKNODE PROCESSOR ARCHITECTURE

The below figure is the CNP architecture according to the min-sum algorithm. It finds the smallest two inputs and the index of the minimum one. Function of the sub-module MIN is to record the minimum, 2nd-Min and the index of the minimum . This check node process can be time consuming for a big row weight matrix. It has log2(t) levels of comparators. Plus the time needed in the variable node process, the critical path is long. To increase the clock speed, the data paths are cut by two level pipelining.

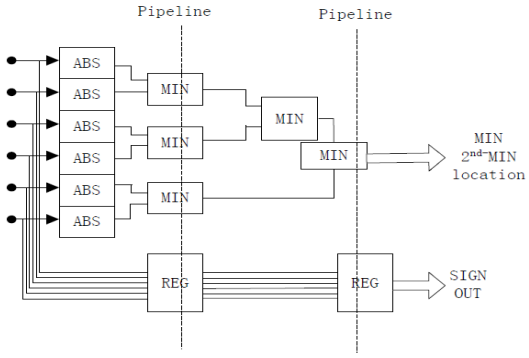

Figure 5.2: Check Node Processor

Also, during the computations of the current iteration, CNU checks the code bits resulting from the previous iteration to check if the code bits satisfy the corresponding parity check equation. After the first half of the iteration is complete, the result of all parity checks on the codeword will be ready too. With this strategy, computations in Check nodes and Variable nodes can be done continuously without the need to wait for checking the codeword resulting from the previous iteration. This increases the speed of the decoding.

5.4 VARIABLE NODE PROCESSOR ARCHITECTURE

Figure 5.3 shows the architecture of variable node processor. The input messages are firstly transferred to two" s complement format and then do the add operation. Finally they are transferred back to sign and magnitude format. At each clock cycle it performs *read-compute-write* operations. They are summarized as follows

i) Read: Reads check-to-variable extrinsic messages from the memory location associated in its PE.

ii) Compute: Performs the variable node computation and generates hybrid variable-to-check extrinsic message with one bit hard decision contained in it.

iii) Write: The variable-to-check messages are written back to the same memory location from where it was read.

There are some methods to improve the performance of Min-Sum product algorithm to scale the variable to check node message or to minus an offset. The algorithm is developed to suit for hardware implementation. Here gives the pseudo code:

if input >= 8 output = 3" b111;

else if input >= 4 output = input-1;

else output = input (unchanged);

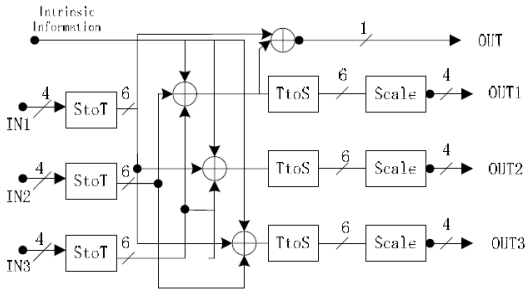

Figure5.3: Variable node processor

Figure 5.4 shows the circuit for scale module architecture [4]. According to performance simulation, it is quite clear that this quantized scaling method is accurate and performs well with QC-LDPC code.

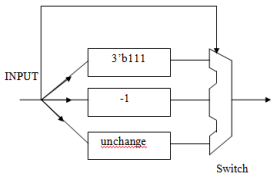

Figure 5.4: Scale module architecture

CHAPTER 6

APPLICATIONS

LDPC codes are fast becoming the new standard in communication in various applications, including digital video broadcasting in the DVB-S2 standard, Wi-Fi, and deep space satellite communications. LDPC codes allow for higher performance in regions of low SNR. Oftentimes, one cannot simply increase the power of a transmitted signal so it can be detected easily without compromising on practicality and cost. In one respect, coding helps to save on power consumption. Devices that communicate via satellite on the DVB-S2 standard will be able to decode faster, perhaps from longer distances where the SNR would drop, or even in environments that are heavily shielded that would attenuate the power of a received signal. Increased performance means that a wider variety of applications can be developed for more massive amounts of information content.

Additionally, LDPC codes are excellent choices of codes in deep space communications. Even in the next mission to Mars, the Jet Propulsion Laboratory is researching LDPC codes for use proximity-link-relay communications. In deep space communications as well as these link relay systems, the SNR will be quite low considering that power will attenuate over such long distances and through different atmospheres if transmitted from the surface of another planet. Thus, communications in low SNR conditions require powerful error detection and correction.

Another consequence of LDPC codes increasing the performance of transmission means that communication can be extended even further to more remote places. Markets such as broadband wireless and mobile networks that operate in noisy environments need powerful error correction in order to improve reliability for their customers.

CHAPTER 7

RESULTS AND DISCUSSION

A LDPC decoder is modelled using VHDL. Binary Phase Shifted Keying (BPSK) modulated message is passed through an Additive White Gaussian Noise (AWGN) Channel. LLR" s are calculated for the message information received from the AWGN channel. These LLR" s are the input to the designed LDPC decoder. BPSK modulation ,transmission of messages through an AWGN channel and estimation of LLR" s are done using MATLAB
7.6.0.324(R2008a).

A 155-bit (3, 5)-regular quasi-cyclic LDPC code decoder is implemented using Xilinx Virtex-4 XC4VIX15 device with package SF363. The entire decoder is designed in VHDL. Xilinx ISE 9.2i is used for developing, simulating and synthesizing the decoder model. The variable node unit (VNU) and check node unit (CNU) architectures are designed using VHDL procedures and functions. After its implementation pipelining is introduced and the delay,power consumption and resource utilization are being compared. The timing summary, power consumption summary and resource utilization summary for a 155-bit (3, 5)-regular LDPC code are shown in the tables below.

7.1 TIMING SUMMARY

TIMING CONSTRAINTS	WITHOUT PIPELINING	WITH PIPELINING
Mnimum period	2.601ns	2.421ns
Minimum input arrival time before clock	2.428ns	2.428ns
Maximum output required time after clock	3.921ns	3.921ns

Table 7.1 Timing Summary

The timing summary from the synthesis results for the partially parallel architecture and the pipelined partially parallel architecture are being compared above.The minimum period is less for the pipelined architecture.But the minimum inpur arrival time before clock and maximum output required time after clock are the same.

7.2 POWER SUMMARY

POWER SUMMARY	WITHOUT PIPELINING	WITH PIPELINING
Total Power Consumption	176mW	176mW

Table 7.2 : Power Consumption Summary

From the power consumption summary it is clear that both the architectures consume the same power.

7.3 RESOURCE UTILIZATION SUMMARY

RESOURCES	WITHOUT PIPELINING	WITH PIPELINING
Total Number Of Slice Registers	32/6144 = 1%	32/6144 = 1%
Number Of 4 Input Luts	60/12288 = 1%	60/12288 = 1%
Number Of Bounded Iobs	25/240 = 10%	25/240 = 10%
Total Equivalent Gate Count For Design	1376	1376

Table 7.3 : Resource Utilization Summary

The resource utilization is same for both the architectures.

7.4 DECODING THROUGHPUT

a) WITHOUT PIPELINING

The throughput of a partially parallel quasi-cyclic LDPC decoder is given as

$$\text{Throughput} = \frac{\text{blocklength} \times \text{Maximumfrequency}}{\text{No.ofiterations} \times \text{Clockcycles required to complete one iteration}}$$

No of clock cycles required for CNU processing=31

No of clock cycles required for VNU processing=31

No of clockcycle required for decoding = 1

No of iterations = 10

DecodingThroughput=155×384.460MHz/63×10

= **94Mb/s**

b) WITH PIPELINING

No of clock cycles required for CNU processing=31

No of clock cycles required for VNU processing=31

No of clockcycle required for decoding = 1

No of iterations = 10

DecodingThroughput=155×413.052MHz/63×10

= **101.62Mb/s**

7.5 COMPARISON WITH PREVIOUS WORK

PARAMETERS	PROPOSED	Reference [5]	Reference [6]
THROUGHPUT	101.62Mbps	40Mbps	1Gbps
POWER	176mW	-	690mW

Table 7.4.Comparison of Proposed with previous works

The throughput and power of the proposed architecture is compared with 2 previous work in this area viz."A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder"[] and "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity check code decoder"[].It is seen that the throughput of the proposed architecture has been improved to a great extend and also the power consumption has also reduced significantly.

7.6 DISCUSSIONS

From the above results, we find that throughput is high for a pipelined version of partially parallel decoder . A throughput of 101.62 Mbps for a 155-bit (3, 5)-regular code is an appreciable one. Also the proposed pipelined architecture reduces power to a great extend compared to the previous works in this area.

CHAPTER 8

CONCLUSION AND FUTURE WORK

Low Density Parity Check (LDPC) Codes has wide range of applications in the wireless field due to its high capacity performance. Thus the (3, 5)-regular quasi-cyclic LDPC code decoder is implemented on FPGA targeting Xilinx Virtex -4 XC4VLX12 device with package SF363. The decoder was designed using VHDL in Xilinx ISE 9.2i EDA tool.The detailed decoder design and architecture are presented in previous chapter and the results are also explained. This research helped us to gain knowledge in three different fields namely, digital wireless communication, coding theory and VLSI design.

Our current and future work is the FPGA implementation of LDPC decoder for wireless standard 802.16e. At present we are facing the memory conflict problem in Xilinx ISE 9.2 while synthesizing the 768x1280 parity check matrix of wireless standard 802.16e. Techniques such as block partitioning of generator matrix and utiliziation of internal block memory of FPGA device are to be analysed to avert the problem of memory conflict in future.

CHAPTER 9

REFERENCES

1. VLSI Design for Low-Density Parity-Check Code Decoding by Zhongfeng Wang, Zhiqiang Cui, and Jin Sha, *IEEE Trans. Commun., vol. 52,* pp. 1038–1089,Jan 2011
2. Z. Wang and Z. Cui, "Low complexity, high speed decoder design for quasi-cyclic LDPC codes," *IEEE Trans. VLSI Syst., vol. 15, no. 1, pp.*
3. Z. Cui and Z. Wang, "Efficient message passing architecture for high throughput LDPC decoder," in *Proc. 2007 IEEE Int.*
4. Z. Wang and Z. Cui, "A memory efficient partially parallel decoder architecture for QC-LDPC codes," in *Proc. 39th Asilomar Conf. Signals,Systems and Computers, 2005, pp. 729–733*
5. Y. Chen and D. Hocevar, "A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder," in *Proc. IEEE GLOBECOM '03, Dec. 2003, vol. 1, pp. 113–117*
6. A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity check code decoder," *IEEE J. Solid-State Circuits, vol.* 37, pp. 404–412, Mar. 2002
7. J. Sha, M. Gao, Z. Zhang, L. Li, and Z. Wang, "Efficient decoder implementation for QC-LDPC codes," in *Proc. IEEE Int. Conf. Communica- tions, Circuits and Systems (ICCCAS)*, June 2006, vol. 4, pp. 2498 –2502.
8. Z. Wang and Z. Cui, "Low complexity, high speed decoder design for quasi cyclic LDPC codes," *IEEE Trans. VLSI Syst.*, vol. 15, no. 1, pp.104 –114, Jan. 2007.
9. A. J. Blanksby and C. J. Howland, "A 690 -mW 1- Gb/s 1024 -b, rate-1/2 low-density parity check code decoder," *IEEE J. Solid-State Circuits*, vol.37, pp. 404 –412, Mar. 2002.
10. Z. Wang and Z. Cui, "A memory efficient partially parallel decoder architecture for QC-LDPC codes," in *Proc. 39th Asilomar Conf. Signals, Systems*

and Computers, 2005, pp. 729–733.

11. T. Zhang and K. K. Parhi, "A 54 Mbps (3, 6) -regular FPGA LDPCdecoder," in *Proc. IEEE SiPS'2002*, May 2003, pp. 127–132.
12. Z. Wang, Y Chen, and K Parhi, "Area-efficient quasi-cyclic LDPC code decoder architecture," in *Proc. ICASSP 2004*, May 2004, vol. 5, pp.49–52..
13. Z. Wang and Q. Jia, "Low complexity, high speed decoder archi- tecture for quasi-cyclic LDPC codes," in *Proc. ISCAS 2005*, Kobe, Japan, May 23 –27, 2005
14. Gallager, "*Low-density parity-check codes,*" IRE Transactions on Information Theory, vol. IT-8, no.1, pp. 21–28, January 1962
15. R. G. Gallager, "*Low-Density Parity-Check Codes*". Cambridge, MA:MIT Press, 1963.
16. Daesun,"*Low Complexity Switch Network for Reconfigurable LDPC Decoder*" IEEE Trans. vol.18, no.1, January 2010
17. Jin sha, "*Multi Gb/s LDPC Code design and implementation*" IEEE Trans vol.17, no 2 February 2009
18. .Y. Chen and D. Hocevar, "A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder," in *Proc. IEEE GLOBECOM '03, Dec. 2003, vol. 1, pp. 113–117*
19. A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity check code decoder," *IEEE J. Solid-State Circuits, vol.* 37, pp. 404–412, Mar. 2002
20. VLSI Design for Low-Density Parity-Check Code Decoding by Zhongfeng Wang, Zhiqiang Cui, and Jin Sha, *IEEE Trans. Commun., vol. 52,* pp. 1038–1089,Jan 2011
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, no. 2, pp. 599–618, February 2001.
22. 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.
23. 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.
24. 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
25. 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.
26. 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.
27. 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.
28. 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.
29. R. Narayanaswami, "*Coded modulation with low-density parity-check codes*," Master" s thesis, Dept. Elect. Eng., Texas A&M Univ., College Station, TX, 2001.
30. R. M. Tanner, "*A recursive approach to low complexity codes,*" IEEE Trans. Inform. Theory, vol. IT-27, no. 5, pp. 533–547, September 1981.