A CLASS OF NON-BINARY LDPC CODES

A Thesis

by

DEEPAK GILRA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2003

Major Subject: Electrical Engineering

A CLASS OF NON-BINARY LDPC CODES

A Thesis

by

DEEPAK GILRA

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

K.R.Narayanan (Chair of Committee)	
C.N.Georghiades (Member)	R.Mahapatra (Member)
A.L.N.Reddy (Member)	C.Singh (Head of Department)

May' 2003

Major Subject: Electrical Engineering

ABSTRACT

A class of Non-Binary Low Density Parity Check Codes. (May 2003)
Deepak Gilra, B.Tech Indian Institute of Technology, Kharagpur
Chair of Advisory Committee: Dr. K.R.Narayanan

In this thesis we study Low Density Parity Check (LDPC) and LDPC like codes over non-binary fields. We extend the concepts used for non-binary LDPC codes to generalize Product Accumulate (PA) codes to non-binary fields. We present simulation results that show that PA codes over GF(4) performs considerably better than binary PA codes at smaller block lengths and slightly better at large block lengths. We also propose a trellis based decoding algorithm to decode PA codes and show that its complexity is considerably lower than the message-passing algorithm.

In the second part of the thesis we study the convergence properties of non-binary PA codes and non-binary LDPC codes. We use EXIT-charts to study the convergence properties of non-binary LDPC codes with different mean column weights and show why certain irregularities are better. Although the convergence threshold predicted by EXIT-charts on non-binary LDPC codes is quite optimistic we can still use EXIT-charts for comparison between non-binary LDPC codes with different mean column weights.

ACKNOWLEDGEMENTS

I would like to thank Dr. Narayanan for his continuous guidance, inspiration and enthusiasm during the course of my stay at Texas A&M University. I consider myself privileged to have been supervised by Dr. Narayanan. He has always been generous with his time, listening carefully and criticizing fairly. He has been a great source of encouragement and I will always remain indebted to him.

I would also like to thank Dr. Georghiades for his considering me for a Nortel Scholarship and giving me a wonderful opportunity to study at Texas A&M University. I would also like to thank Dr. Reddy and Dr. Mahapatra for being members on my committee.

I would like to thank all of the amazing persons I met during my stay here at Texas A&M University and making my grad life a memorable experience. I would especially like to thank Vivek for his help in correcting my thesis. Among other friends from KGP family I would like to thank Deba, Veera, Bhos, Singhal, Dali, Waqar, Parcy and whole lot of other people for making my graduate life a wonderful experience.

This thesis is dedicated to my mother, my father and my sisters, Kavita and Nisha, whose importance I shall not try to put into words. And lastly, I would like to thank the most beautiful person in my life, Nishu, for all her warmth and love, without whom I cannot imagine my life.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Low-density parity check (LDPC) codes are a class of linear error correcting codes defined by a very sparse parity check matrix. Gallagher [1] presented a decoding algorithm and a detailed performance analysis on regular LDPC codes in his dissertation in 1963. Recently Mackay and Neal [2] proved that LDPC codes are "very good" which means that there are sequences of LDPC codes with rates up to Shannon capacity, which achieve arbitrarily low probability of error using the optimal ML decoder. The re-discovery of these codes has sparked major research in the coding field because of their near Shannon limit performance and simple description.

Conventional LDPC codes have a low decoding complexity but may have high encoding complexity. The encoding complexity is typically of the order $O(n^2)$. Also high storage space may be required to explicitly store the generator matrix. For long block lengths the storage space required would be huge. The above factors make the implementation of the conventional LDPC codes less attractive.

Product Accumulate (PA) codes proposed by Li, Narayanan and Georghiades [3] are a class of LDPC codes that are linear time encodable and decodable at significantly low complexity and offer high rates. PA codes are essentially LDPC codes with two levels of checks. The encoding complexity of PA codes is low and also these codes do not require explicit storage of the generator matrix. The performance of binary PA codes has been shown to be a few tenths of a dB away from the Shannon limit for rates greater than or equal to 1/2. The decoding complexity of PA codes is similar to that of LDPC codes. The above factors make implementation of Product Accumulate Codes more attractive than LDPC codes.

The journal model is *IEEE Transactions on Automatic Control.*

Davey and Mackay [4], [5] have shown that non-binary LDPC codes defined over GF (q), q> 2, show significant improvement over binary LDPC codes. The main reason can be attributed to the dependence of the decoding algorithm on the mean column weight of the H matrix. Mackay has shown that, given an optimal decoder, LDPC codes can approach Shannon limit for long block lengths and high mean column weight. The parity check matrix for codes defined over GF(q) contain elements from GF(q). Hence the mean column weight of the equivalent binary parity check matrix increases, when moving from GF (2) to GF (8). Another way of increasing the mean column weight of the binary parity check matrix is to introduce more ones in the column of H matrix. But this introduces more cycles in the corresponding graph and it is known that the decoding algorithm of LDPC codes performs worse over graphs with cycles. On the other hand, by moving to GF(q) we increase the mean column weight of the parity check matrix without introducing cycles in the corresponding graph. Also moving onto GF(q), q> 2, increases the state space of each node in the decoding graph by decoding over GF(q). In other words increasing the field order is comparable to increasing the memory of convolutional code.

We see that codes defined over GF (4) and GF (8) perform better than codes defined over GF (2) for certain irregularities and certain mean column weights. For successful decoding, the average entropy of the messages passed in the graph of LDPC codes should fall below a certain threshold after a certain number of iterations. Mackay and Davey [5] show that the average entropy of messages passed in the graph for non-binary LDPC codes falls faster than the average entropy of messages passed in graph for binary LDPC codes only for certain mean column weight. Although the procedure provided in [5] finds the mean column weight where codes over GF(q) would outperform codes over GF(2) it does not give any insight into the convergence properties of the decoding algorithm.

The main objective of this thesis is to study LDPC and LDPC-like codes over non-binary fields. The main focus is on PA codes because of the advantages listed above. Also non-binary LDPC codes perform better than binary LDPC codes, so we generalize the PA codes to non-binary fields. We present simulation results for PA codes over GF(4) and

GF(8). We also propose a trellis based decoding algorithm for PA codes. We show that the decoding complexity of the new algorithm is considerably lower than the message-passing algorithm proposed in [3].

In order to explain the reason for better performance of non binary PA codes and the influence of mean column weight on the performance of LDPC codes over GF(q), we look at the convergence properties of the decoding algorithm. Ten Brink [6] showed that using mutual information as a measure for extrinsic information transfer (EXIT) charts, the convergence behavior of iteratively decodable schemes can be visualized. Each constituent decoder is represented by mutual information transfer characteristics, which describes the flow of extrinsic information through the soft in soft out decoder. However, it is tricky to extend the concept of EXIT-charts to non-binary fields because it is difficult to model the *apriori* information. So we use the approach suggested by Benedetto and Montorsi and Scanavino [7]. They show that if the decoder and the interleaver operate at the symbol level, we get a lower convergence threshold. We use this concept to study and compare the convergence thresholds for binary and non-binary PA codes. We also use the concept of EXIT-charts to show why certain irregularities in LDPC codes are better than others.

The organization of the thesis is as follows. In Chapter II, we give a detailed background of binary and non-binary LDPC codes. We also introduce Product Accumulate (PA) codes followed by decoding of binary and non-binary LDPC codes. A reader familiar with LDPC codes and LDPC-like codes can skip this chapter. In Chapter III we propose non-binary PA codes. We also propose trellis based decoding for non-binary PA codes and compare its complexity with the conventional message passing decoding. In Chapter IV we introduce EXIT-charts. We review Benedetto and Montorsi's approach for comparing bit- and symbol-interleaved serially concatenated codes [7]. We then extend the idea of EXIT-charts to non-binary fields. Next, the convergence threshold of binary and non-binary PA codes is compared. We also compare the convergence thresholds for non-binary LDPC codes with different mean column weights. In Chapter V we present the conclusions.

CHAPTER II
LOW DENSITY PARITY CHECK CODES

This chapter presents background material, which can be skipped by a reader familiar with LPDC codes. LDPC codes are a class of linear error correcting codes with very sparse parity-check matrices. These codes are usually decoded using the sum-product algorithm, which is a message passing algorithm working on the Tanner graph of the code. The sparseness of the parity check matrix is essential for attaining good performance with sum-product decoding. The time complexity of the sum- product algorithm is linear in code length. This property makes it possible to implement a practical decoder for long lengths.

Linear codes use a generator matrix G to map a message vector X of length k to a transmitted codeword Y of length n. All codewords satisfy HY=0, where H is the parity check matrix. Gallager defined (n, p, q) LDPC codes to have a block length n and a parity check matrix with exactly p ones per column and q ones per row, where p >=3. The rate of the code is $k/n = 1 – (p/q)$. Gallager proved that, for a fixed p, the error probability of the optimum decoder decreases exponentially for sufficiently low noise and sufficiently long block length. The parity check matrix is typically constructed randomly while constraining the distributions of the row and column vectors as uniform as possible. Since H is not in systematic form, we perform Gaussian elimination using row operations and reordering of columns. The resulting parity check matrix has the form $H^{/} = [-P \mid I_m]$, where the notation [A|B] indicates the concatenation of matrices A and B; and I_m is the mxm identity matrix. The corresponding generator matrix $G = [I_k \mid P]$, is not sparse. So the encoding complexity is O(nxn) per block.

The H matrix can be represented as a bipartite graph, which is defined as an undirected graph where vertices can be divided into two sets such that no edge connects vertices in the same set. Each **bit** (column of H) is represented by a variable (left) node and each **check** (row of H) is represented by a check (right) node. For binary codes the values in

the H matrix are either 1 or 0. A 1 denotes an edge between the corresponding variable node and the check node. If the *H* matrix has *N* columns and *M* rows, the corresponding bipartite graph has *N* bit nodes and *M* check nodes. An example of a parity check matrix is shown in Figure 1.

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

Figure 1: An example of H matrix (20,3,4)

In the above parity check matrix there are 4 ones per column and 3 ones per row. This means that a bit node participates in 3 checks and 4 bit nodes participate in a single check. The Tanner graph representation for the LDPC code described by the H matrix above is shown in Fig. 2.

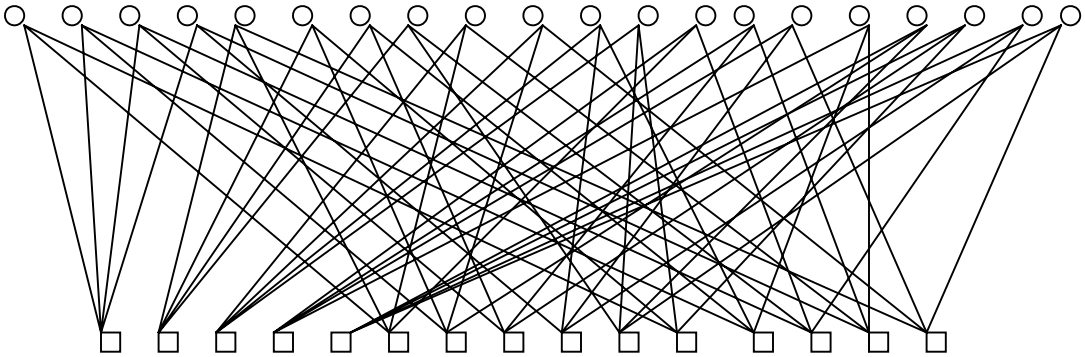

Figure 2: The Tanner graph of LDPC code described by H matrix in Figure 1

A **cycle** in a graph is a path that begins and ends at the same node. So, for a bipartite graph, the shortest possible cycle has length 4. If the Tanner graph of a code is cycle free, then message-passing algorithms like the min-sum algorithm and the sum-product algorithm, all converge to the optimal solution. Even if there are cycles in the graph, we see that the decoding algorithm still converges. The convergence is faster if there are no short cycles. In order to remove short cycles (of length 4), no two columns of the *H* matrix should overlap more than once.

2.1 Product Accumulate Codes: PA Codes are a class of "good" codes that offer high rates and are linear-time encodable and decodable. A "good" code is defined as a code for which arbitrarily low error rates can be achieved above a certain noise threshold as the block length goes to infinity. LDPC codes and Turbo codes are examples of such codes. Turbo codes have a high decoding complexity since the complexity of the maximum *aposteriori* probability (MAP) decoding of the constituent codes grows exponentially with the constraint length. Conventional LDPC codes, on the other hand, have a low decoding complexity but the encoding complexity is high. Also large storage space may be required to explicitly store the generator matrix. For long block lengths the storage space required would be huge. PA codes offer an advantage over LDPC codes in that they do not require explicit storage of the generator matrix and yet the decoding complexity is comparable to that of conventional LDPC codes. These advantages make

PA codes more attractive to implement than the conventional LDPC codes and Turbo codes.

2.1.1 Structure of PA Codes: PA codes are the serial concatenation of an inner rate-1 differential [1/(1+D)] encoder and an outer Turbo Product Code/Single Parity Check (TPC/SPC) code. The structure of a PA code is shown in Fig 3.

Conventional two-dimensional TPC/SPC codes are obtained by arranging the data in a $t \times t$ block and appending parity checks to each row and column. This is equivalent to an LDPC code where each row in each dimension satisfies a check, and hence message-passing decoding can be employed. Alternatively, the TPC/SPC may also be viewed as a parallel concatenation of two $(t+1, t)$ single parity check codes separated by a block interleaver. For the constituent code of a PA code, the block interleaver of the TPC/SPC is replaced by a random interleaver. The message-passing decoding can still be employed to decode this code. In Chapter III, we will present an alternative, trellis-based approach for the decoding of the TPC/SPC outer code. This approach enables us to exploit a wealth of research in trellis-decoding algorithms for use in the decoding of PA codes.

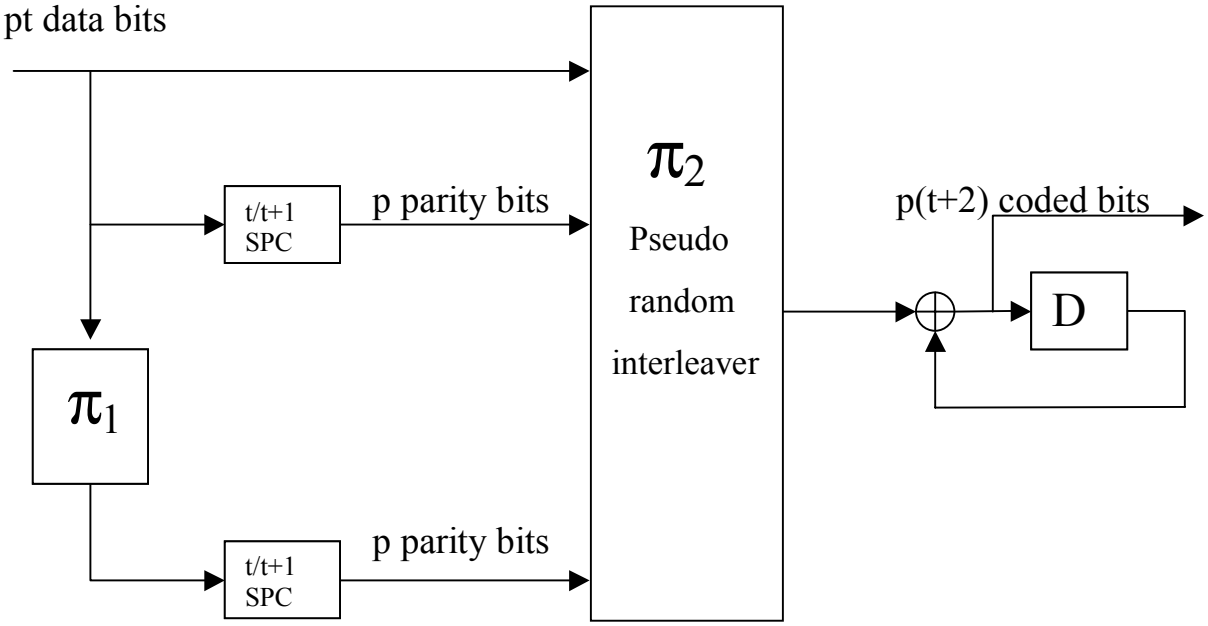

Figure 3: Structure of Product Accumulate code

The input block of $p*t$ bits is broken down into p blocks of t bits each. Each block of p bits is fed into the first SPC encoder to produce a parity bit per block. So each branch produces p parity elements. So $p*(t+2)$ bits are fed to the inner decoder. Since the inner encoder is rate-1 differential encoder the length of the output codeword is $p*(t+2)$. Hence the effective rate of the code is t/(t+2).

2.2 Low Density Parity Check Codes over GF(q): Davey and Mackay [4], [5] proposed low-density parity check codes over GF (q), q>2, called non-binary LDPC codes. They showed that LDPC codes over GF(q) achieve superior performance to that of binary LDPC codes. In case of non-binary codes the H matrix can take values from the finite field GF(q). Again $HY=0$ and the presence of elements from the GF(q) produces more stringent checks on the codewords. Any non-zero value at H(i,j) indicates that there is an edge existing between i_{th} row and j_{th} column.

An LDPC code over GF(q) can also be represented by a Tanner graph, with a weight on each edge of the graph. This weight is the matrix entry in the parity check matrix and is

chosen from the finite field GF(q). So now for bit nodes defined over GF(q), a check m would require

$$\sum_{j \in N(m)} a_{m_j} * x_j = 0, \qquad (1)$$

where N(m) is the set of variable nodes connected to the check m and a_{m_j} € GF(q), $a_{m_j} \neq 0$. Mackay showed that going from binary to non-binary field may reduce the number of cycles in the Tanner graph. If we associate a pxp matrix for every element in GF(q), $q=2^p$ then we can substitute the H matrix in GF(q) domain with an equivalent H matrix which is pxp times longer in each domain. This can be seen from the figure 4.

Let the H matrix in GF(4) be

1	1	0
3	0	2

Its equivalent binary H matrix is:

1	0	1	0	0	0
0	1	0	1	0	0
0	1	0	0	1	1
1	1	0	0	1	0

Where $0 = = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $1 = = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, $2 = = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, $3 = = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

The equivalent Tanner graph for both the cases would be:

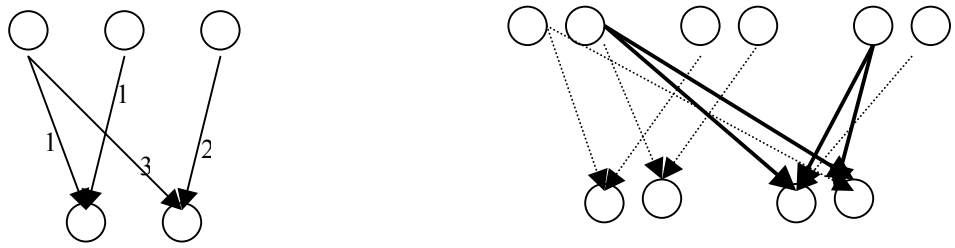

Figure 4: Comparison of Tanner graph at GF(4) and GF(2)

From the above graph we see that for the binary case there exists a short cycle of length=4 which is absent in the non-binary case. This is one of the main reasons why we expect LDPC codes over non-binary fields to perform better than LDPC codes over the binary field.

2.3 Sum-Product Algorithm: The decoding of the LDPC codes over GF(q) is done using the sum-product algorithm in an iterative fashion. The encoding is viewed as a set of bit nodes connected to the check node that satisfies the check as shown in (1). A bit node participates in a set of checks because there are more than 1 non-zero elements in the column. The H matrix determines the set of check nodes to which bit nodes are connected.

The decoder first receives p bits from the channel that make up a q–ary symbol. The prior distribution for that symbol is set to:

$$f^{a} = \Pi_{i=1}^{p} f_{x_i}^{a_i} \qquad (2)$$

Where $f_{x_i}^{a_i}$ is the likelihood that the i_{th} constituent bit is equal to a_i , an element from the finite field.

2.3.1 Update at Check Node : R_{ij}^{a} is the message that check *i* send to node *j*. It is the probability of the check being satisfied assuming the bit node to which it is connected is equal to symbol *a* where *a* is an element from the finite field GF(q). So the probability is calculated by summing over all configurations of x (the codeword) for which the check is satisfied and also the bit node is equal to symbol *a*.

$$R_{ij}^{a} = \sum_{x:x_j} P(z_i \mid x) \prod_{k \in N(i) \setminus j} Q_{i_k}^{x_k} \tag{3}$$

The probability $P(z_i| x)$ is the probability that the check is satisfied or not and hence is equal to zero or one. Q_{ij}^{a} 's are the messages that the bit node sends to the check node that is suppose to approximate the node's belief that it is equal to symbol '*a*' given messages received from all other check nodes. *N(i)* denotes the set of noise nodes connected to that check node and *N(i)/j* denotes the set of noise nodes connected to check node except *j*.

2.3.2 Update at Bit Node: The messages that bit node *j* sends to check *i* is the probability that the bit node is equal to symbol *a* according to the set of check nodes connected to that bit node.

$$Q_{ij}^{a} = \alpha_{ij} f_j^{a} \prod_{k \in M(j) \setminus i} R_{k_j}^{a} \tag{4}$$

Where M(j) is the set of check nodes connected to the bit node *j* and f_j^{a} is the prior probability that x_j is equal to symbol a. The normalization constant α_{ij} ensures that $\Sigma\, Q_{ij}^{a} = 1$.

2.3.3 Hard Decisions: At each iteration we compute the vector as:

$$Y_j = \arg\max f_j^a \prod_{k \in M(j)} R_{kj}^a \quad (5)$$

This vector should satisfy the parity check equation HY=0 for the decoding to be declared to be as successful. If the condition is not met then we iterate again. This continues until we reach a fixed number of iterations after which we declare a decoding failure.

2.4 Fourier Transform Decoding*:* The complexity of the message passing decoding algorithm scales as $O(q^2)$. Using Fourier transform decoding as described by Richardson and Urbanke [8] we reduce the complexity.

The update of check node is described in (3) is

$$R_{ij}^a = \sum_{x:x_j} P(z_i \mid x) \prod_{k \in N(i) \setminus j} Q_{i_k}^{x_k}$$

where N(i) is the set of bit symbols that participate in check i. The above equation represents a convolution of Q_{ik}^{xk} quantities. Using Fourier transform we can change the summation to a product of Fourier transform of Q_{ik}^{xk} and then later take the inverse Fourier transform. The Fourier transform is taken over GF (q). The Fourier transform F of a function f over GF(2) is given by $F^0 = f^0 + f^1$ and $F^1 = f^0 - f^1$ Similarly transforms over GF(4) can be viewed as :

$$F^0 = [f^0 + f^1] + [f^2 + f^3] \quad (6)$$
$$F^1 = [f^0 - f^1] + [f^2 - f^3] \quad (7)$$
$$F^2 = [f^0 + f^1] - [f^2 + f^3] \quad (8)$$
$$F^3 = [f^0 - f^1] - [f^2 - f^3] \quad (9)$$

The inverse transform is the same followed by division by 4. The transforms over GF (2^k) are viewed as a sequence of binary transforms in each of k dimensions and the inverse transforms are the same, followed by division 2^k.

2.5 Simulation Results: For encoding we transmit an all zero sequence. Since the code is linear there is no loss of generality. Binary input Gaussian channel is simulated and we examine the success of decoding after several iterations on a block and several blocks. Each q-ary symbol is transmitted over various uses of channel. For example each symbol over GF(4) is transmitted over 2 uses of the binary channel and similarly each symbol over GF(8) us transmitted over 3 uses of the channel. The sum product algorithm is used to decode and the performance is observed by plotting the bit error rate with E_b/No. The figure 5 shows simulation results for rate 1/3 code with transmitted blocklength equal to 49152 binary bits where as the figure 6 shows the performance of the code with rate ¼ and transmitted block length=6000 bits:

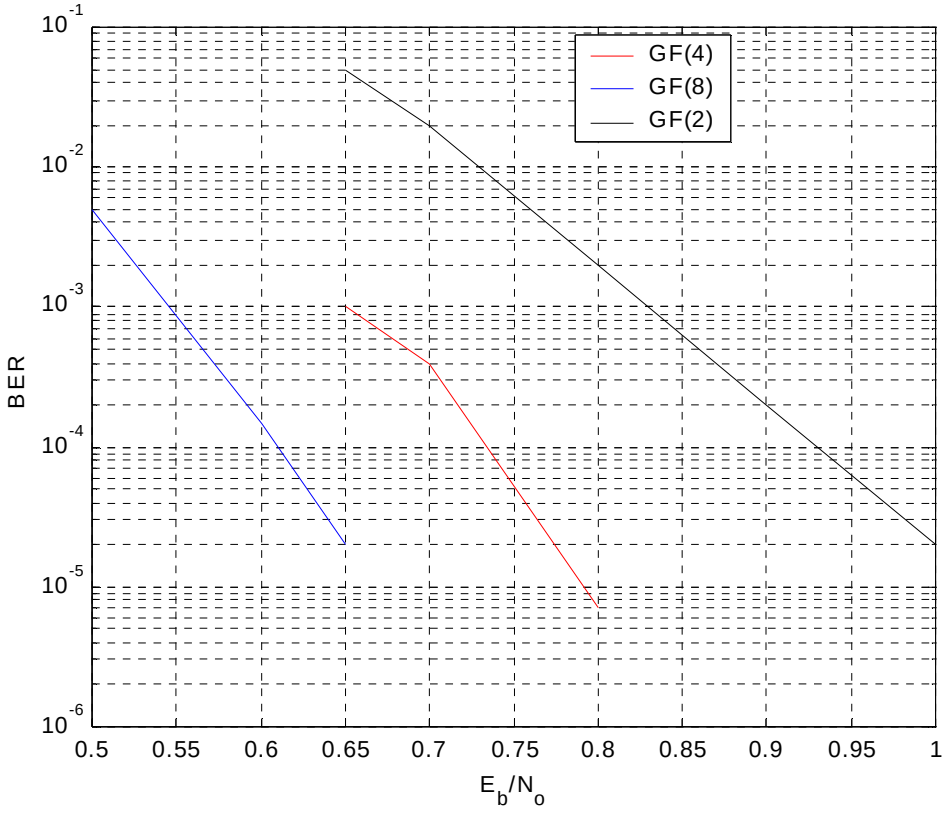

Figure 5: Performance of LDPC codes rate 1/3, transmitted block length=49152 bits over GF(2),GF(4),GF(8) with mean column weight=2.5

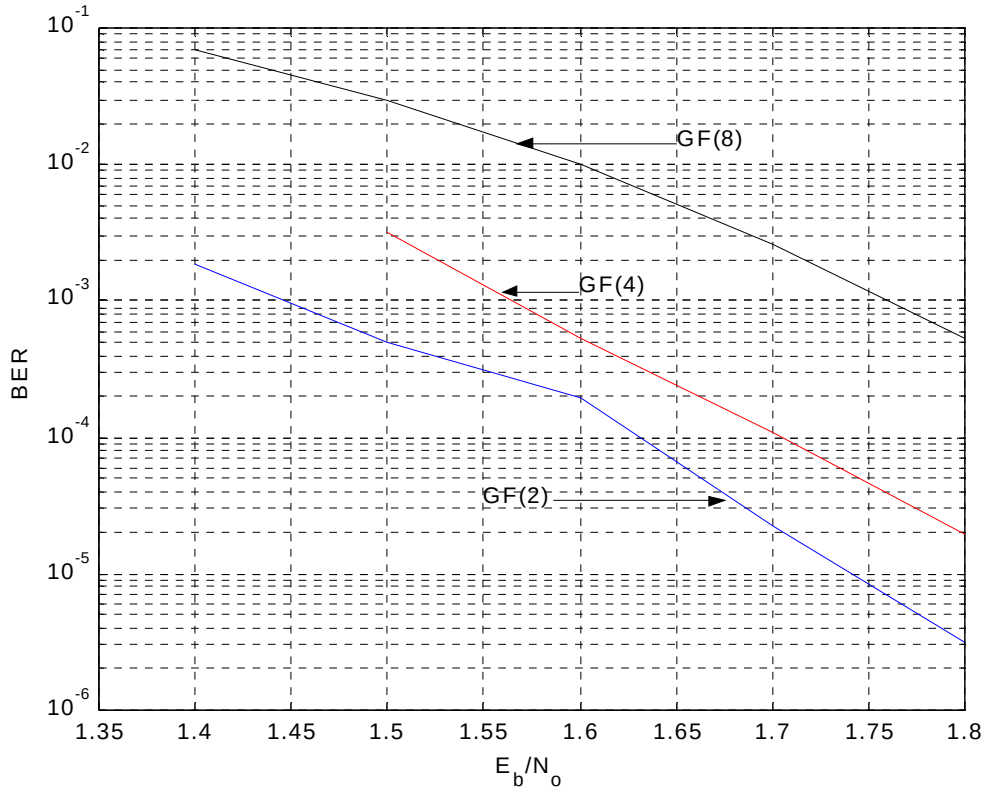

Figure 6: Performance of LDPC codes rate ¼ , block length=6000 transmitted bits

We see that for rate 1/3 codes there is a significant improvement in performance by going from GF (2) to GF (4) and then to GF (8). For rate ¼ the best codes are defined over binary field and the codes over GF (4) and GF (8) perform worse with the same binary block length. So we see that non-binary LDPC codes perform better than the binary codes only for certain mean column weight.

The reason can be attributed to the dependence of decoding algorithm on the matrix weight. As the matrix weight is increased, the number of neighbors for the check node increases and so the check node is less confident of its neighbors and the decoder performs worse. At the same time increasing the field order would produce similar effect because now the neighbor has more possible states. So going to higher order field with high matrix weight should give worse performance. But intuitively we also see that producing more stringent conditions on the check node reduces error. Davey shows the effect of changing the field order and mean column weight on the decoding algorithm.

Davey [5] used Monte Carlo methods to simulate LDPC codes of infinite length whose associated graphs have a tree structure and hence the decoding algorithm is known to be exact. First an ensemble S of S noise symbols is created according to the channel model. Then the messages, which go from the noise node to the check node, are updated after each iteration. The ensemble of noise symbols and associated Q messages are computed according to the channel model. Then the values of the check updates are calculated assuming that all the other noise nodes connected to it are coming from the same ensemble S. The number of noise nodes connected to a check node is determined by the rate and the mean column weight. After we have the check updates we can create a new ensemble from the updates of the check nodes. Similarly new ensembles are created after each iteration. The ensemble contains approximations to the distribution of Q messages in an infinite network after an arbitrarily number of iterations. For successful decoding the average entropy of the Q messages should become arbitrarily small as the decoding progress. The decoding is declared a success if the average entropy drops below a certain threshold after some iteration. So now Davey compares the decoding performance on the basis of rate and matrix column weight. He shows that for rate ½ and mean column weight 3 codes over GF (4) will perform the best followed by codes over GF (2) and then codes over GF (8), which is exactly what the results show. On the other hand for rate ¼ at mean column weight 3 codes over GF (2) will perform the best followed by codes over GF (4) and then GF (8). Davey shows that for rate ½ GF (8) would perform best at mean column weight of 2.8 followed by GF (4) and then GF (2) and for rate ¼ the mean column weight should be 2.6. The above approach does not give any insight into the convergence properties of the decoding algorithm. We explain these results using EXIT-charts in Chapter IV.

CHAPTER III

Non-Binary Product Accumulate Codes

From the previous chapter we see that non-binary LDPC codes perform better than binary LDPC codes. Also we see that binary Product Accumulate code offer advantages over binary LDPC codes. So we generalize Product Accumulate codes to non-binary fields using the same principles as used for non-binary LDPC codes. The transfer function for the inner code of a non-binary PA code is of the form 1/(1+aD), where 'a' is a weight randomly chosen from the finite field GF (q). A weight is chosen on the intuition that it would increase the robustness of the code. It has been later verified from the simulations that introducing weight indeed results in a better performance for short block lengths. The SPC in the outer code now has *t* random weights attached to it making the checks more stringent. The parity element is produced as:

$$\sum_{i=0}^{t} x(i) * a(i) + y * a(t+1) = 0. \qquad (1)$$

Where a(i)…a(t+1) are randomly chosen weights from the finite field GF(q). 'y' is the parity element produced as a result of the above equation.

3.1 Decoding of Non-Binary PA Codes: In general, the ML decoding of serially concatenated codes is prohibitively complex. Iterative decoding is usually employed, wherein the constituent decoders exchange "extrinsic" information in a turbo-like fashion. Since the decoding is performed in non-binary field the exchange of extrinsic information is in the form of actual probabilities or log of probabilities instead of log-likelihood ratios. From the figure 7 we see how the soft extrinsic information is exchanged between the two decoders.

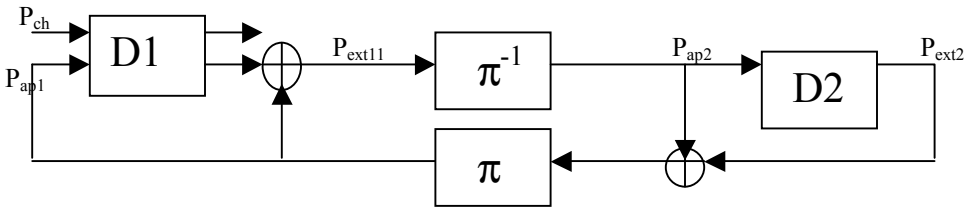

Figure 7:Block diagram depicting decoding for Serially Concatenated codes

The decoder D1 receives *apriori* information (p_{ap1}) for each input bit from the decoder D2. It also receives channel information p_{ch} from the demodulator. It then computes extrinsic information p_{ext1} for the input bits. This information is passed to the outer decoder D2 that treats the information as *apriori* information p_{ap2}. D2 then computes the extrinsic information p_{ext2} for the outputs and passes it to D1. The decoding proceeds until either all checks are satisfied (successful decoding) or a fixed number of iterations is reached. The idea behind extrinsic information is that the decoder D1 provides soft information to D2 using information not available not D1.

For a PA code the outer code is a parallel concatenation of SPC codes. The outer code can be decoded using message-passing algorithm. It can be represented on a graph with bit nodes and check nodes. 't+1' bits participate in a check and each bit participates in 2 checks except the 2*p parity elements which are produced by the single parity check equations, which participate in just one check. The inner code is decoded using BCJR algorithm [13] on non-binary fields.

3.2 Trellis Based Decoding: We propose to use the trellis based decoding to decode the outer code. The operations in the SPC encoder can be represented on an irregular trellis as shown in figure 8. Each path in the trellis shows a check being satisfied. Once we can draw the trellis we reduce the decoding complexity by using assumptions not so evident in the message passing decoding. For codes over GF(q) the corresponding trellis has q states. The paths in the trellis represents all possible codewords of the single parity check code. This is explained more clearly with an example for GF(4).

For GF(4) the trellis has 4 states. For t=2 the trellis has t+1=3 stages. The number of codewords equals to 16. The trellis in Fig 8 has 16 paths and each path corresponds to one of the codewords. For example the path *aaa* corresponds to codeword 000 where as the path *bbb* corresponds to codeword 303.

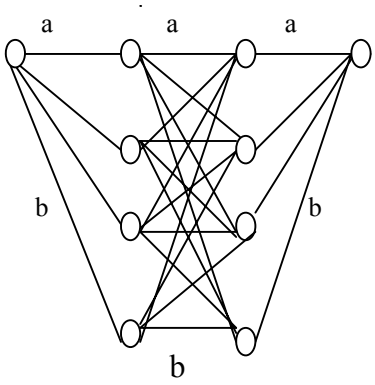

Figure 8: Trellis for t=2 SPC encoder over GF(4)

The first state is always 0. In the first stage there can be transition from state 0 to any state 0,1,2,3 depending on the input symbol. At the second stage there similarly there can be transition to any stages from any stage depending on the input symbol. At the third stage the transition is always to the state 0 denoting a check being satisfied. Once the trellis is drawn we apply BCJR decoding algorithm to get extrinsic information of the transitions on the trellis. Then we iteratively decode the outer TPC/SPC code in turbo fashion by passing soft information from one decoder to the other.

The sum-product and BCJR (forward-backward) algorithms are equivalent. The BCJR takes all paths into consideration before deciding the best path. Similarly in the sum-product algorithm, the update of check nodes is done by taking into consideration all possible combinations of values that bit nodes can take in order to satisfy the check. So the performance of both the decoders is the same except for the complexity. The sum product algorithm was discussed in the previous chapter and the BCJR algorithm is discussed below in section 3.2.1.

3.2.1 BCJR Algorithm: The Bahl Cocke Jelinek and Raviv algorithm [9] is used on Markov chains to produce *aposteriori* probabilities for the input and output symbols. The inner encoder is a differential encoder which is a convolutional code of the form 1/(1+aD) and can be viewed as Markov chain. Once the trellis is drawn, we can identify the states, the transitions, etc. For the inner code the number of states at each stage is 'q' the field order. For the outer code, the maximum number of states at any stage for each trellis is also 'q'.

We define the following quantities:

$$\alpha_t(m) = P(S_t = m, r_1^t) \text{ --------forward path metric} \quad (2)$$

$$\beta_t(m) = P(r_{t+1}^N \mid S_t = m) \text{ ---Backward path metric} \quad (3)$$

$$\gamma_t(m, m') = P(S_t = m, r_t \mid S_{t-1} = m') \text{–branch metric} \quad (4)$$

Here S_t denotes the state of the encoder at any time 't', r_t is the received sequence.
((m, m/) $\in$ (0,1,2,3) for q=4)

$$\gamma_t(m, m') = P(S_t = m, r_t \mid S_{t-1} = m')$$
$$= \sum_i P(a_t = i \mid S_t = m, S_{t-1} = m') * P(r_t \mid S_t = m, S_{t-1} = m') * P(S_t = m \mid S_{t-1} = m')$$

The first term in the equation is either '0' or '1' depending on whether the transition between states m and m' exists or not. The second term is the probability of the output symbol given the present state is m and the previous state is m'. This is the value obtained from the channel. The third term is the *apriori* probability of the input symbol for the transition from m' to *m*. In the absence of parallel transitions (which is the case for all examples considered here), the summation reduces to a single term:

$$\gamma_t(m.m') = P(r_t \mid S_t = m, S_{t-1} = m') * P(S_t = m \mid S_{t-1} = m')$$

3.2.1.1 Forward Recursion

$$\alpha_t(m) = P(S_t = m, r_1^t)$$
$$= \sum_{m'} P(S_{t-1} = m', r_1^{t-1}) * P(S_t = m, r_1^t \mid S_{t-1} = m', r_1^{t-1})$$
$$= \sum_{m'} \alpha_{t-1}(m') * \gamma_t(m', m) \quad (5)$$

.

The term $\alpha_t(m)$ is the probability of being in state m at time t given the received sequence until time 't'. When the encoding starts the encoder is always in state 0 and hence we initialize $\alpha_0(0) = 1$. Also $\alpha_0(m) = 0$ for $m \neq 0$. At each stage we normalize the $\alpha_t(m)$ to maintain numerical accuracy.

3.2.1.2 Backward Recursion

$$\beta_t(m) = P(r_{t+1}^N \mid S_t = m) = \sum_{m'} P(S_{t+1} = m', r_{t+1}^N \mid S_t = m)$$
$$= \sum_{m'} P(S_{t+1} = m', r_{t+1}, r_{t+2}^N \mid S_t = m)$$
$$= \sum_{m'} P(r_{t+2}^N \mid S_{t+1} = m', r_{t+1}, S_t = m) * P(S_{t+1} = m', r_{t+1} \mid S_t = m)$$
$$= \sum_{m'} \beta_{t+1}(m') * \gamma_t(m, m') \quad (6)$$

The term $\beta_t(m)$ is the probability of receiving the sequence r_{t+1}^N given that the current state is m. The trellis always ends in state 0 at t=N because of tail bits insertion in the encoder and so we initialize $\beta_N(0) = 1$. If we do not insert tail bits and the trellis ends in state m, we initialize $\beta_N(m) = 1$. For single parity check encoder the trellis ends in state 0 at every *(t+1)th* stage. We normalize the $\beta_t(m)$ at each 't' to preserve numerical accuracy.

The aposteriori probability of the input symbols is computed as:

$$P(a_t = 0 | r_1^N) = \sum_m \sum_{m'} \alpha_{t-1}(m) * \gamma_t(x_t = 0, m', m) * \beta_t(m) \quad (7)$$

$$P(a_t = 1 | r_1^N) = \sum_m \sum_{m'} \alpha_{t-1}(m) * \gamma_t(x_t = 1, m', m) * \beta_t(m) \quad (8)$$

$$P(a_t = 2 | r_1^N) = \sum_m \sum_{m'} \alpha_{t-1}(m) * \gamma_t(x_t = 2, m', m) * \beta_t(m) \quad (9)$$

$$P(a_t = 3 | r_1^N) = \sum_m \sum_{m'} \alpha_{t-1}(m) * \gamma_t(x_t = 3, m', m) * \beta_t(m) \quad (10)$$

where a_t is the input on the trellis at time 't' and r_1^N is the received sequence. x_t is the input/output associated with the transition from state m' to m.

The equations when expanded on GF(4) are as:

$P(a_t=0 \mid Y_1^N) = \alpha_{t-1}(0)*\gamma_t(0,0)*\beta_t(0) + \alpha_{t-1}(1)*\gamma_t(1,1)*\beta_t(1) + \alpha_{t-1}(2)*\gamma_t(2,2)*\beta_t(2) + \alpha_{t-1}(3)*\gamma_t(3,3)*\beta_t(3)$.

$P(a_t=1 \mid Y_1^N) = \alpha_{t-1}(0)*\gamma_t(0,1)*\beta_t(1) + \alpha_{t-1}(1)*\gamma_t(1,0)*\beta_t(0) + \alpha_{t-1}(2)*\gamma_t(2,3)*\beta_t(3) + \alpha_{t-1}(3)*\gamma_t(3,2)*\beta_t(2)$.

$P(a_t=2 \mid Y_1^N) = \alpha_{t-1}(0)*\gamma_t(0,2)*\beta_t(2) + \alpha_{t-1}(1)*\gamma_t(1,3)*\beta_t(3) + \alpha_{t-1}(2)*\gamma_t(2,0)*\beta_t(0) + \alpha_{t-1}(3)*\gamma_t(3,1)*\beta_t(1)$.

$P(a_t=3 \mid Y_1^N) = \alpha_{t-1}(0)*\gamma_t(0,3)*\beta_t(3) + \alpha_{t-1}(1)*\gamma_t(1,2)*\beta_t(2) + \alpha_{t-1}(2)*\gamma_t(2,1)*\beta_t(1) + \alpha_{t-1}(3)*\gamma_t(3,0)*\beta_t(0)$.

Once again the probabilities are normalized and clipped to maintain numerical accuracy.

3.2.2 Max-Log-MAP*:* The BCJR algorithm may be implemented in the log domain. The new definition of $\alpha_t(m)$ and $\beta_t(m)$ is as follows:

$$\alpha_t'(m) = \log \alpha_t(m) \quad (11)$$

$$\beta'_t(m) = \log \beta_t(m) \tag{12}$$

$$\gamma'_t(m,m') = \log(\gamma_t(m,m')) \tag{13}$$

The previous definition of gamma is given as:

$$\gamma_t(m,m') = P(S_t = m, r_t \mid S_{t-1} = m')$$
$$= P(r_t \mid S_t = m, S_{t-1} = m') * P(S_t = m \mid S_{t-1} = m')$$

$$\log(\gamma_t(m,m')) = \log(P(S_t = m, r_t \mid S_{t-1} = m'))$$
$$= \log(P(r_t \mid S_t = m, S_{t-1} = m') * P(S_t = m \mid S_{t-1} = m'))$$
$$= \log(P(r_t \mid S_t = m, S_{t-1} = m')) + \log(P(S_t = m \mid S_{t-1} = m')) \tag{14}$$

The forward and backward recursions are now defined in equation (15) and (16).

$$\alpha'_t(m) = \log \sum_{m'} \exp(\alpha'_{t-1}(m') + \gamma'_t(m',m)) \tag{15}$$

$$\beta'_t(m) = \log \sum_{m'} \exp(\beta'_{t+1}(m') + \gamma'_t(m',m)) \tag{16}$$

The initial conditions for the recursions are:

$\alpha'_0(0) = 0$ and $\alpha'_0(m) = -\infty$ for $m \neq 0$.

$\beta'_N(0) = 0$ and $\beta'_N(m) = -\infty$ for $m \neq 0$

From the above recursions it is clear that all the multiplications are converted to additions but the exponential and logarithmic terms still remain. In order to simplify the exponent terms we use the approximation given in equation (17).

$$\log(e^{x_1} + e^{x_2}) = \max(x_1, x_2) + \log(1 + e^{-|x_i - x_2|}) \tag{17}$$

The second term approaches zero as $|x_1 - x_2|$ increases. We can quantize the term $\log(1 + e^{-|x_1 - x_2|})$ and store the values of $\log(1 + e^{-|x_1 - x_2|})$ for 4 or 8 levels. If we use this

correction then the resulting algorithm is called Log-MAP algorithm. So each computation of the form $\log(e^{x_1} + e^{x_2})$ now required only a max operation and a look-up. There is no exponential calculation as well as no logarithm function calculation. The term $\log(1+e^{-|x_i - x_2|})$ may be dropped altogether with a small penalty in performance. This version of the algorithm is called the Max-Log-Map algorithm.

The MAP takes all paths through the trellis into calculation at each step and classifies them into q sets. Then afterwards at each step it groups the paths into respective sets. The Max-Log-MAP algorithm looks at only q paths at each step. The paths can change from step to step but one will always be maximum-likelihood (ML) path. This explains the difference in performance between Log-MAP and Max-Log-MAP.

The equations for the max-log-map algorithm are written in equation (18) and (19).

$$\alpha'_t(m) = \max_{m'}(\alpha'_{t-1}(m') + \gamma'_t(m', m)) \tag{18}$$

$$\beta'_t(m) = \max_{m'}(\beta'_{t+1}(m') + \gamma'_t(m, m')) \tag{19}$$

The normalization of α''s and β's is given in equation (20) and equation (21).

$$\alpha'_t(m) = \max_{m'}(\alpha'_{t-1}(m') + \gamma'_t(m', m)) \quad - \max_m(\max_{m'}(\alpha'_{t-1}(m') + \gamma'_t(m', m))) \tag{20}$$

$$\beta'_t(m) = \max_{m'}(\beta'_{t+1}(m') + \gamma'_t(m, m')) - \max_m(\max_{m'}(\beta'_{t+1}(m') + \gamma'_t(m, m'))) \tag{21}$$

Rewriting the equations 7-10 in terms of log probabilities we get

$$\log(P(a_t = i | r_1^N)) = \max_{m,m'}(\alpha'_{t-1}(m) + \gamma'_t(x_t = i, m', m) + \beta'_t(m)) \tag{22}$$

Once again we normalize as shown in equation (22).

$$\log(P(a_t = i | r_1^N)) = \max_{m,m'}(\alpha'_{t-1}(m) + \gamma'_t(x_t = i, m', m) + \beta'_t(m)) - \max_m \log(P(a_t = m \mid r_1^N)) \tag{23}$$

3.2.3 Complexity Comparisons: The complexity of message passing algorithm and Max-Log-Map can be tabulated as:

Table 1: Table comparing the complexity of different algorithms used to decode PA codes over GF(q)

	Max-Log-Map	Sum-Product	Fourier Transform
Additions	$5\ q^2 +2q$	$q^2 +3q$	5q
Multiplication	0	$q^2 +3q$	4q
Max ops	2q+3	0	0

In table 1 we also show the complexity of Fourier transform decoding that can be used to decode the outer code. Fourier transform decoding reduces the complexity of the update at check node. The update of check node is a convolution of messages coming from the bit nodes and by Fourier transform decoding we convert the convolution to a product of Fourier transforms. The details of Fourier transform decoding are given in section 2.4. From the above table we see that Max-Log-MAP has a much lower complexity as it does not involve any multiplications. Also when we are working in log domain and do not deal with actual probabilities then the algorithm is numerically much more stable.

3.3 Interleaver: When we generalize PA codes to non-binary fields we can use a bit interleaver or a symbol interleaver. Intuitively we would think that interleaver working at bit level would provide more interleaving gain than interleaver working at symbol level . This is because using a symbol interleaver is equivalent to using k bit interleavers that implement the same interleaving pattern where the field order is GF (2^k). Interleaving k bits separately allows the spreading of components of one error event to k times more error events. Thus a symbol level interleaver introduces a structure that reduces the interleaver gain. But still we work at symbol level. This is because working at bit level requires the projection of symbol extrinsic information onto bit extrinsic information in each passing of the extrinsic information between SISO's. This destroys the mutual

information between bits belonging to the same symbol, thereby reducing the effectiveness of the APP evaluation.

3.4 Results: In order to obtain the simulation results for non-binary PA codes we generate random symbols over GF(q), encode the symbols by the procedure described in the introduction section of this chapter. The symbols are then converted to binary and we use BPSK modulation at the transmitter. AWGN channel is considered for all the simulations. At the decoder we use Max-Log-Map algorithm or Message Passing algorithm to decode. Figure 9 shows the simulation results for rate ½ Product Accumulate Codes for a Block Length=2000 transmitted bits over different non-binary fields. Message passing algorithm is used to decode the outer code and BCJR algorithm decodes the inner code. The number of iterations considered is 50 after which we declare decoding failure. Figure 10 compares the performance of the Product Accumulate Code over GF(4) with two different decoding algorithms, Max-Log Map and Message Passing. Figure 11 shows the simulation results for rate ½ PA codes with transmitted block length=128000 bits over GF(2) and GF(4). The number of iterations we consider is 100 after which we declare decoding failure.

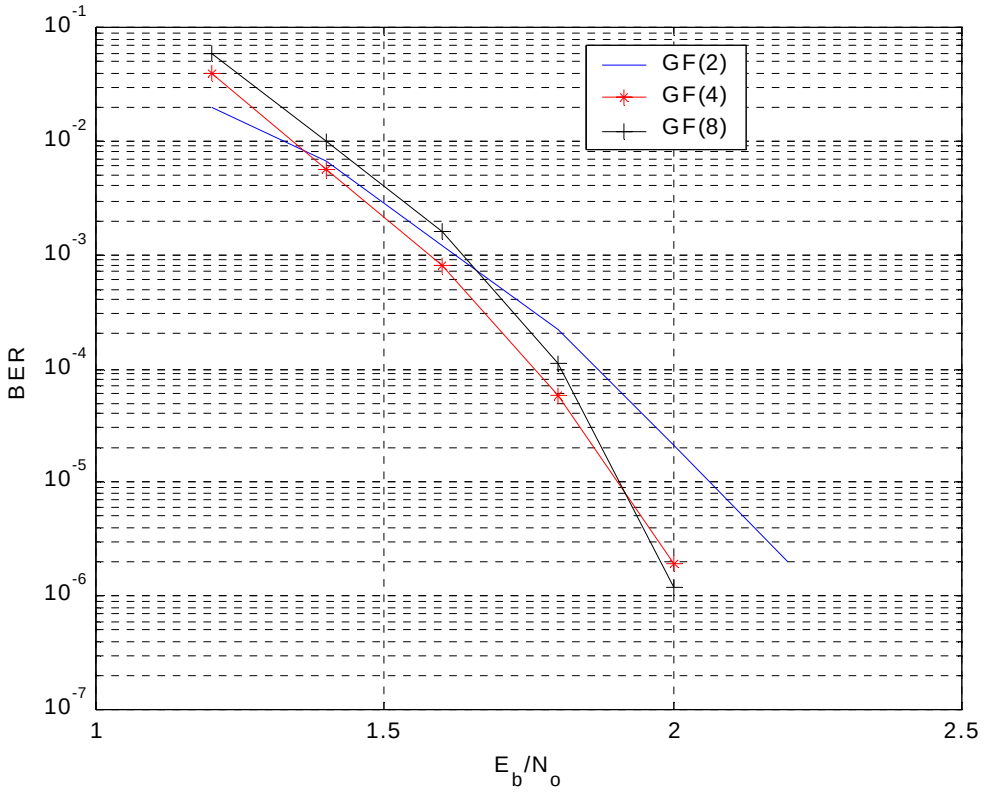

Figure 9: Simulation Results for rate ½ PA codes with transmitted block lengths =2000 bits over GF(2),GF(4) and GF(8)

From the results it can be seen that at very low SNRs code defined over GF(2) performs better than GF(4) and also GF(8) but at higher SNRs code defined over GF(8) performs better than GF(4) and GF(2) . We know that a serial concatenated code performs worse at lower SNRs if the inner encoder has more memory. Increasing the field order is equivalent to increasing the memory. So binary PA codes perform better than non-binary PA codes at low SNRs.

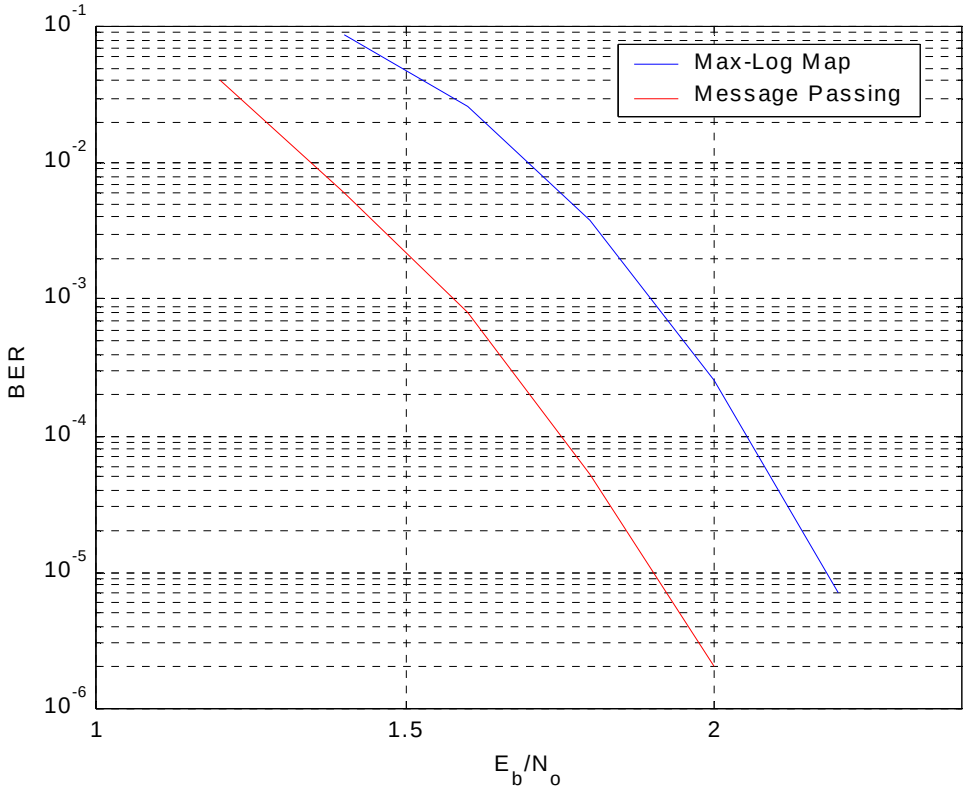

Figure 10: Simulations using Max-Log MAP decoding and Message passing algorithm for rate ½ PA codes over GF(4) with transmitted block lengths= 2000 bits

From the above plot we see that the loss in the performance is approximately equal to 0.3dB when we use Max-Log-MAP algorithm. This is the trade off we achieve between complexity and performance.

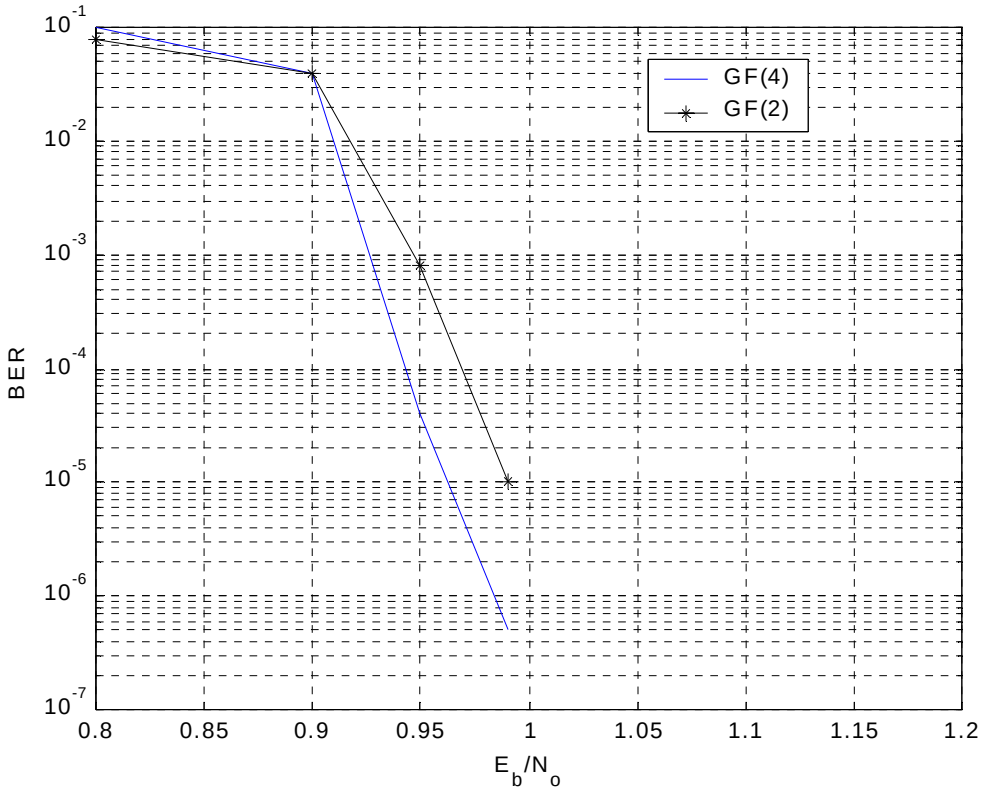

Figure 11: Simulation results for PA codes with transmitted block lengths =128000 bits over GF(2) and GF(4)

The above plot shows we achieve a gain of 0.03dB at higher block length and BER=10^{-5} which is quite small. We thus conclude that codes over GF(q) with higher block length do not show significant improvement in performance than binary codes but for small block lengths the improvement in performance is quite significant.

CHAPTER IV
EXIT-CHART TECHNIQUE

4.1 Motivation: From the simulation results shown in the previous chapter several questions arise. We would like to see, asymptotically if there is an improvement in performance between binary and non-binary codes. Also we would like to design non-binary codes without going into actual BER simulations. We use Extrinsic Information transfer chart (EXIT-chart) to address all the above issues. EXIT-charts are known to accurately predict the behavior of iterative decoding for binary codes. But the extension of this technique to non-binary fields is non-trivial.

4.2 Introduction: Extrinsic Information transfer chart [6] is a tool to predict the behavior of iterative decoding by looking at the input/output relations of the individual decoders. EXIT-charts are used to describe the behavior of iterative decoding without performing actual BER simulations. EXIT-chart is a technique designed on the lines of density evolution technique introduced by Richardson and Urbanke [8]. Similarly SNR measures developed by Hesham El Gamal and A.Roger Hammons Jr [10] have also been used to study the convergence of iterative decoding of turbo codes. It has been shown that the iterative decoder converges to zero probability of error as the number of iteration increases if and only if E_b/N_o exceed a certain threshold. In other words there exists a threshold which characterizes the convergence of the iterative decoder.

In order to plot the EXIT-charts we use the following assumptions.

1) The extrinsic information passed from one sub-decoder to other is a Gaussian random variable.
2) The extrinsic information entering the decoder are jointly and pair wise independent.

These assumptions can be validated by experimental data. These assumptions are reasonable since the channel is Gaussian and the computation of extrinsic information involves summation of various random variables.

Extrinsic Information transfer characteristics are based on mutual information to describe the flow of extrinsic information through constituent decoders. Based on the EXIT-chart techniques constituent encoders can be designed to give low convergence threshold or give better turbo cliff position.

Figure 12: A SISO decoder showing its input and output

From the above figure 12 we see that a soft in soft out (SISO) decoder gets *apriori* information as the input and it outputs the extrinsic information. The *apriori* information entering the decoder is the extrinsic information passed from the other decoder. The *apriori* information entering the decoder is modeled as a Gaussian random variable based on the assumptions described above. In order to characterize the *apriori* information we use mutual information between the *apriori* information and the coded bits. Mutual information is used as a measure to describe the flow of extrinsic information between the constituent decoders.

The mutual information between extrinsic information and coded bits for inner decoder is a function of its input mutual information and E_b/N_o.

$$I_{E_1} = f(I_{A_1}, E_b / N_o) \qquad (1)$$

E_1 is the output extrinsic information of inner decoder and A_1 is the input *apriori* information to the inner decoder.

For the outer decoder the mutual information between the output extrinsic information and coded bits is only a function of its input mutual information since it does not receive any input from the channel.

$$I_{E_2} = f(I_{A_2}) \tag{2}$$

where E_2 is the output extrinsic information and A_2 is the corresponding input *apriori* information.

The function I_{E_1} is plotted with respect to I_{A_1} as an EXIT-curve. On the same figure we also plot I_{E_2} with respect to I_{A_2} but with the axes of this curve reversed. The figure now is called EXIT-chart and we can study the evolution of iterative decoding by visualizing the flow of extrinsic information between the SISO decoders. The charts are characterized by different values of E_b/N_o.

Mutual information is defined in equation 3.

I(X;Y)=H(X)-H(X|Y)

(3)

The first term is simple to compute and we use Monte Carlo simulations to compute the second term.

$$\begin{aligned}
\mathrm{I(X;Y)} &= \sum_i - p(i) * \log_2(p(i)) - \frac{1}{N}\sum_N \sum_i \; - p(y \mid x = i) * \log_2(p(y \mid x = i)) \\
&= \sum_i - 0.25 * \log_2(0.25) - \frac{1}{N}\sum_N \sum_i \; - p(y \mid x = i) * \log_2(p(y \mid x = i)) \\
&= 2 + \frac{1}{N}\sum_N \sum_i \; p(y \mid x = i) * \log_2(p(y \mid x = i))
\end{aligned}$$

The mutual information between a random variable and the information symbol can also be calculated as:

$$I_{\Pi} = \frac{1}{2} \cdot \sum_{a=-1,1} \int_{-\infty}^{\infty} p_{\Pi}(\xi \mid A = a) * \log_2 \frac{2.p_{\Pi}(\xi \mid A = a)}{p_{\Pi}(\xi \mid A = -1) + p_{\Pi}(\xi \mid A = 1)} d\xi$$

We transmit an all zero sequence and so the above equation takes the form as:

$$I_{\Pi} = \int_{-\infty}^{\infty} p_{\Pi}(\xi \mid A = -1) * \log_2 \frac{2.p_{\Pi}(\xi \mid A = -1)}{p_{\Pi}(\xi \mid A = -1) + p_{\Pi}(\xi \mid A = 1)} d\xi$$

For GF(4) the above function is defined as:

$$I_{\Pi} = (1/4) \sum_{a=0,1,2,3} \int_{-\infty}^{\infty} p_{\Pi}(\xi \mid A = a) * \log_2 \frac{4.p_{\Pi}(\xi \mid A = a)}{p_{\Pi}(\xi \mid A = 0) + p_{\Pi}(\xi \mid A = 1) + p_{\Pi}(\xi \mid A = 2) + p_{\Pi}(\xi \mid A = 3)} d\xi$$

If we transmit all zeros sequence the above equation takes the form as:

$$I_{\Pi} = \int_{-\infty}^{\infty} p_{\Pi}(\xi \mid A = 0) * \log_2 \frac{4.p_{\Pi}(\xi \mid A = 0)}{p_{\Pi}(\xi \mid A = 0) + p_{\Pi}(\xi \mid A = 1) + p_{\Pi}(\xi \mid A = 2) + p_{\Pi}(\xi \mid A = 3)} d\xi$$

An example of EXIT-chart is given in figure 13. EXIT-chart for PA codes at GF(4) is drawn at E_b/N_o=1.2dB.

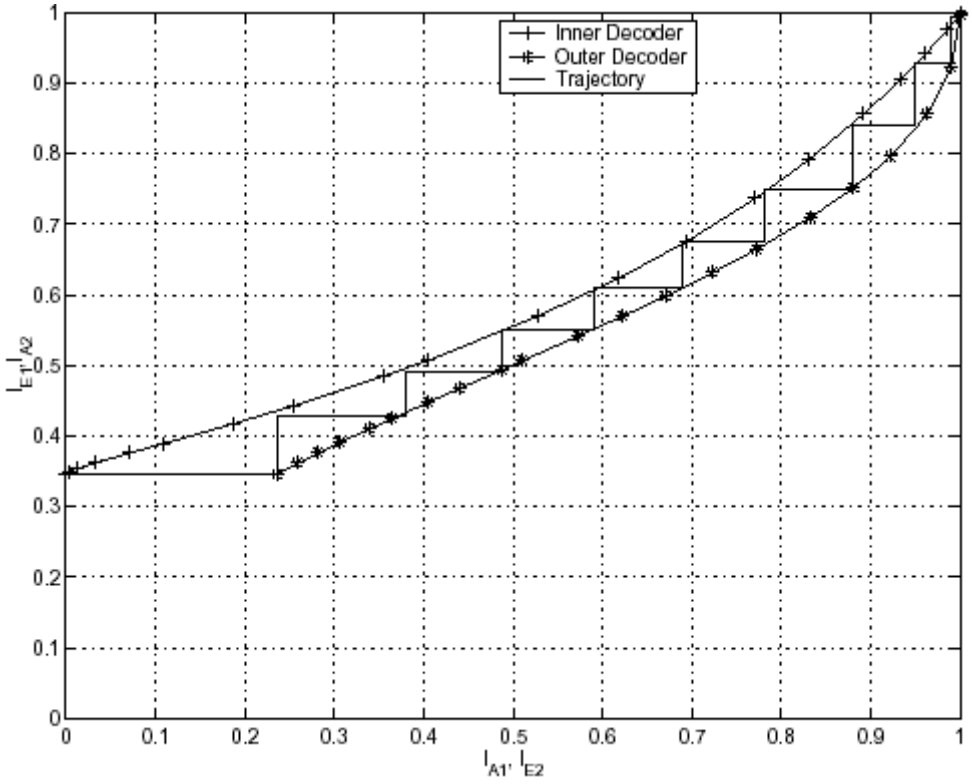

Figure 13: EXIT-chart showing evolution of decoders at E_b/N_o =1.2dB

From figure 13, we can explain the evolution of the iterative decoding. The x-axis represents the input mutual information of the inner decoder and output mutual information of the outer decoder with respect to the coded bits. The y-axis represents the output mutual information of the inner decoder and input mutual information of outer decoder with respect to the coded bits.

From the stepwise curve shown we can study the evolution of the iterative decoding. At iteration 0 the SISO inner decoder receives zero input mutual information and outputs mutual information I_1. This mutual information is fed to the outer decoder, which outputs mutual information O_1. Then at next iteration O_1 is fed to SISO inner that produces I_2. Interleaving and de-interleaving do not change the mutual information. The iterations continue as long as both the curves do not meet. The iteration stops when there is an intersection between the two EXIT-curves. This indicates the presence of a fixed point. If this happens the decoder does not converge.

The convergence threshold is defined as that E_b/N_o when the step curve just manages to sneak through the EXIT-curves. A large gap between the EXIT-curves indicates fast convergence. If the gap is small then the convergence occurs after a large number of iterations.

For the case of parallel-concatenated codes, both the encoders are same. So we plot the transfer curve of one decoder and the transfer curve of the other decoder is obtained by taking a mirror image along the x=y line.

4.3 EXIT-chart for Non-Binary Codes: The problem in plotting EXIT-charts for non-binary case is modeling the *apriori* information. The apriori information for non-binary codes cannot be taken as a product of k Gaussian random variables, where GF(q)=GF(2^k), as that would amount to destroying the mutual information between bits belonging to the same symbol. Instead, we assume this product of k Gaussian random variables is the input to the previous decoding stage and the corresponding output is what is input to the decoder under consideration.

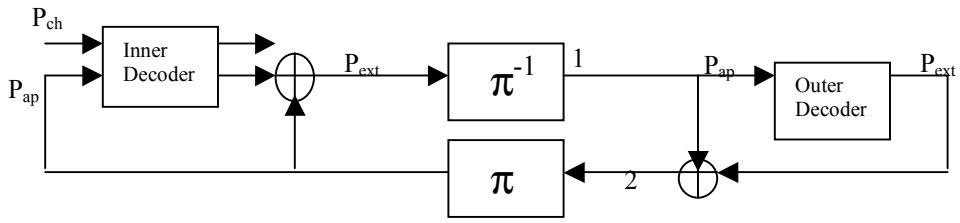

Figure 14:A Serial Concatenated Decoder

With reference to Fig. 14, we cut the loop at section 1 and input k Gaussian random variables (with the appropriate mutual information) to the outer decoder. The (extrinsic) mutual information generated by this decoder is treated as the *apriori* input to the inner decoder and forms the basis of computing the transfer characteristics of the inner decoder. In a similar fashion, the loop is cut at 2 (Fig. 14) next, and *k* Gaussian random variables are input to the inner decoder. The (extrinsic) mutual information generated by the inner decoder is then passed to the outer decoder as *apriori* information and forms the basis of computing the transfer characteristics of the outer decoder.

Similar to the binary case, the two transfer characteristics so obtained can be plotted on the same graph to obtain the EXIT chart for a non-binary concatenated code.

4.4 EXIT-Chart for Non-Binary LDPC Codes: The main objective is to study why certain irregularities are better for non-binary LDPC codes. We use EXIT-charts to study the convergence behavior of non-binary LDPC codes with different mean column weight. The decoding algorithm for LDPC codes was described in section 2.3. The iterative decoding algorithm proceeds with an update at the bit node followed by an update at the check node. We can interpret an LDPC code as a serial concatenated code. The inner decoder gives the update at the bit nodes and the outer decoder gives the update at the check nodes. The interpretation is a bit loose because a serial concatenated decoder exchanges only one extrinsic message per bit of the code word whereas an LDPC decoder exchanges several messages per bit of the code word. Inspite of this difference, the algorithm of the previous section can still be used to compute the EXIT chart for a non-binary LDPC code.

In figures 15 and 16 we show EXIT-chart for binary and non-binary LDPC codes at mean column weight=2.8 and $E_b/N_{o=}$ 1.0.

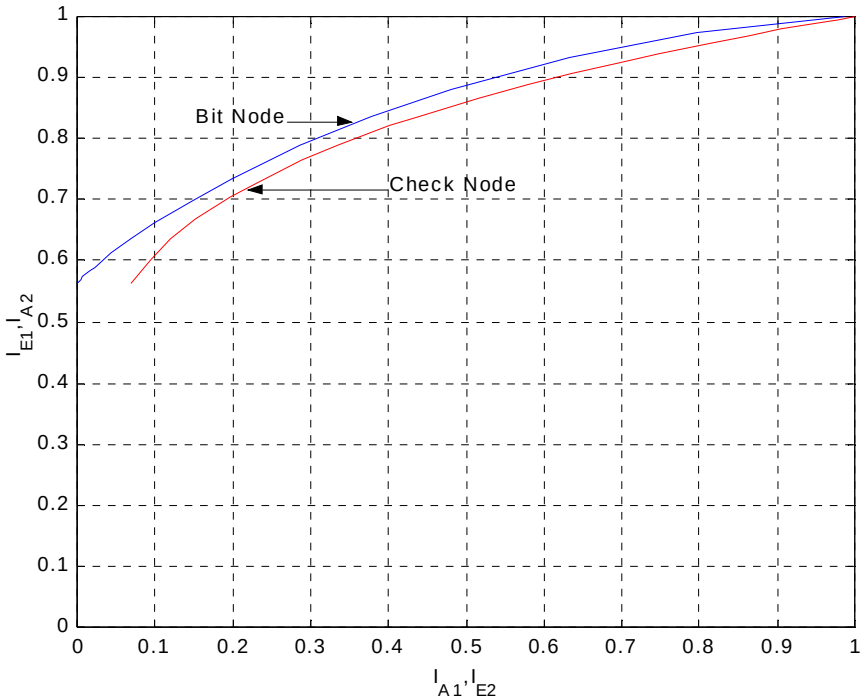

Figure 15: EXIT-chart for non-binary LDPC codes with mean column weight=2.8, rate=1/2 and $E_b/N_{o=}$ 1.0 dB

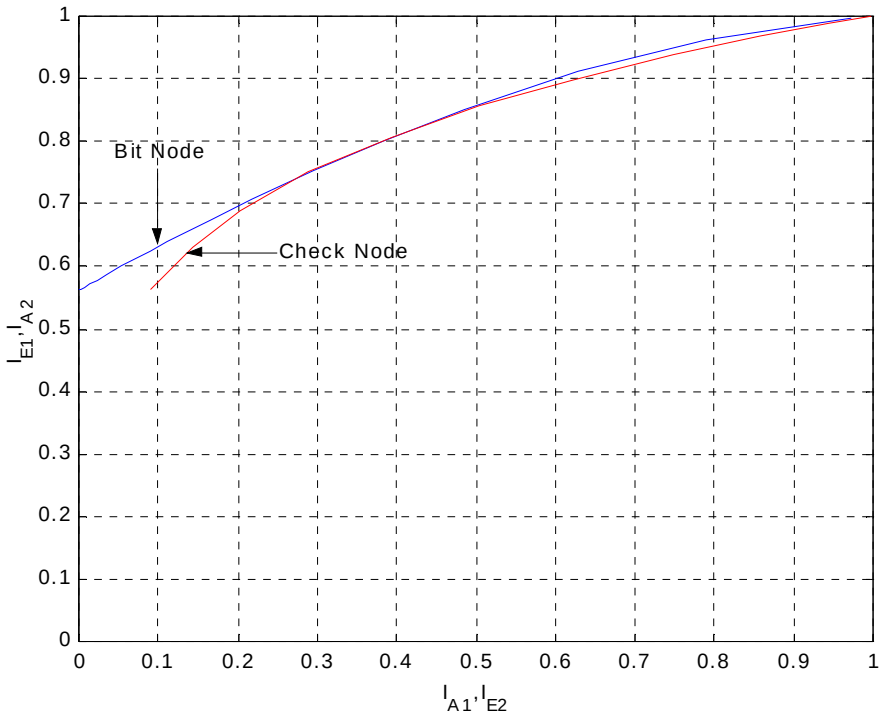

Figure 16: EXIT-chart for binary LDPC codes with mean column weight=2.8, rate=1/2 and $E_b/N_{o=}$ 1.0 dB

In the figures 15 and 16 I_{A1}, I_{A2} represent the input mutual information to the bit node and check node respectively and I_{E1}, I_{E2} represent the output mutual information of the bit node and check node respectively. From the figure 16 we see that for the binary LDPC codes there is a fixed point at 1.0dB and from figure 15 we see that the decoding algorithm converges easily for non-binary LDPC codes at E_b/N_o =1.0dB. After plotting EXIT-charts for several E_b/N_o we find that the convergence threshold for non-binary LDPC codes is equal to 0.4dB. This would mean that the non-binary LDPC codes would perform much better than the binary LDPC codes at very large block length and lower E_b/N_o.

EXIT-charts are known to accurately predict the convergence thresholds for binary codes but for the non-binary codes the accuracy is yet to be investigated. In order to verify the

accuracy of the convergence threshold at non-binary level we plot the actual decoding trajectory for one block with a large block length on the EXIT-chart.

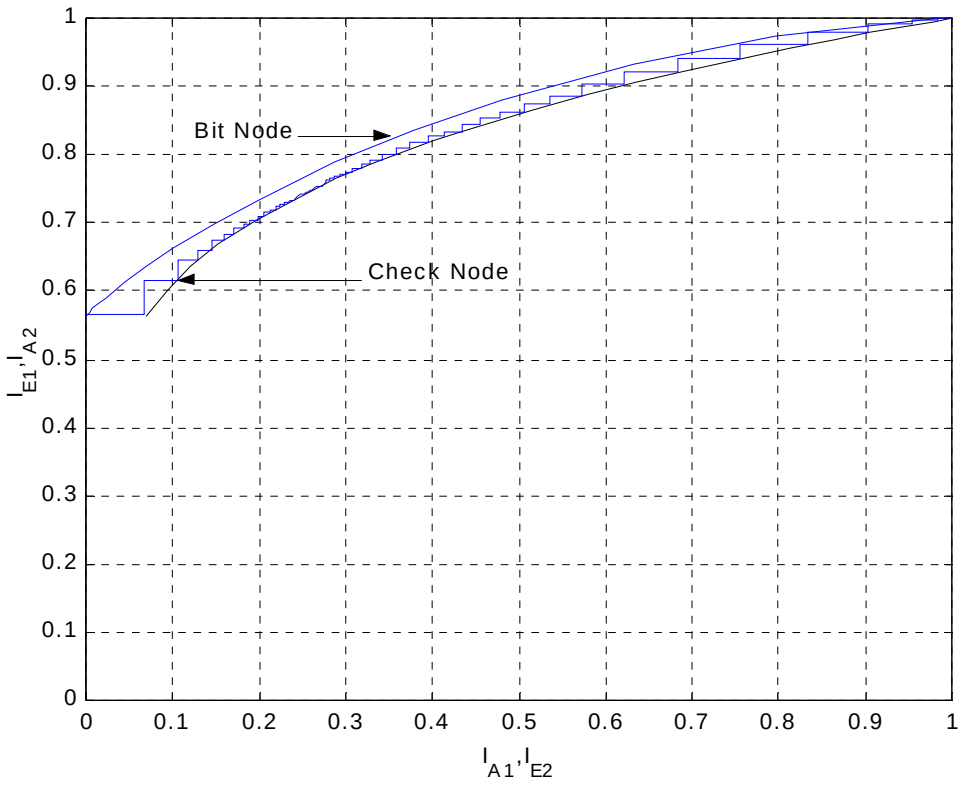

Figure 17: EXIT-chart showing the actual trajectory for one block for rate ½ LDPC code at mean column weight=2.8 and $E_b/N_{o=}$ 1.0 dB

From figure 17 above we see that there is a discrepancy between the actual decoding trajectory and the one predicted by the EXIT-charts. This discrepancy comes possibly from the Gaussian approximation we use for the extrinsic information at non-binary fields. At first iteration the bit node receives mutual information equal to 0 and outputs mutual information equal to 0.57. The check node receives mutual information equal to 0.57 and outputs mutual information equal to 0.07. Until this point the actual decoding trajectory and the one predicted by EXIT-chart seem to match. At the next iteration the bit node receives mutual information equal to 0.07 and outputs mutual information equal to 0.62 but the EXIT-chart predicts mutual information equal to 0.65. In order to understand this discrepancy we plot the histogram of the actual extrinsic information and

extrinsic information with Gaussian approximation at the output of the bit node at second iteration. Since we transmit all zeros we only plot the histogram for probabilities that the symbol equals to zero.

One curve in figure 18 shows the distribution of actual extrinsic information at second iteration. In order to obtain second curve we cut the loop at 1 in figure 14 and feed the check node with k Gaussian random variables approximating the extrinsic information with the mutual information obtained at the output of bit node. These random variables are then fed to the check node and the output from check node is then fed to the bit node. The second curve shows the histogram of the extrinsic information obtained by using this Gaussian approximation.

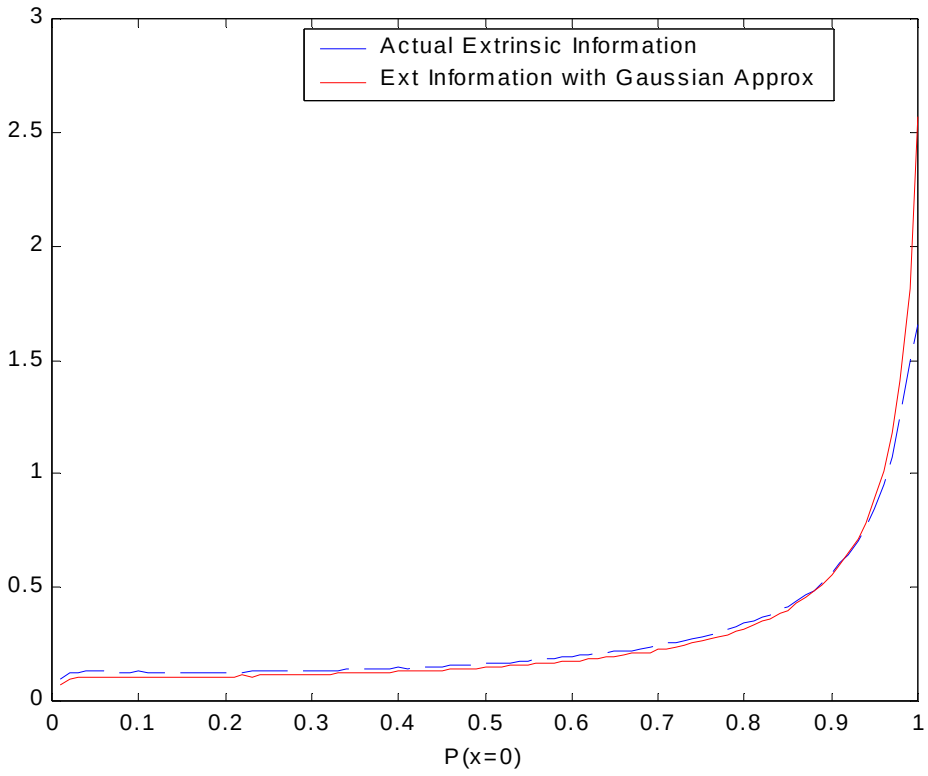

Figure 18: Histogram of actual extrinsic information and extrinsic information using Gaussian approximation

From the plot we see that the extrinsic information using Gaussian approximation is more optimistic than the actual extrinsic information. The erroneous region can be magnified to get a better view of the above curve as shown in figure 19.

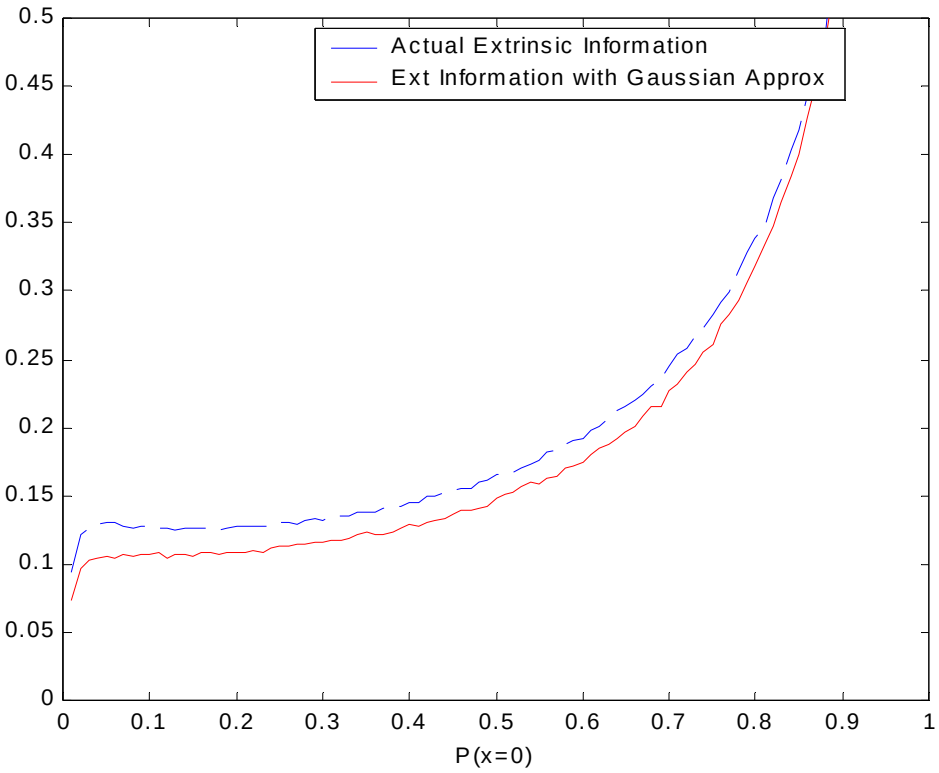

Figure 19: Histogram of actual extrinsic information and extrinsic information using Gaussian approximation in the erroneous region

For the erroneous regions, we see that the area under the curve for actual extrinsic information is more than that for the extrinsic information obtained using Gaussian approximation. Hence we get less output mutual information for given input mutual information during actual decoding in comparison to the output mutual information predicted by EXIT-charts.

So the convergence threshold predicted by EXIT-charts for non-binary LDPC codes is optimistic. Although the convergence threshold of non-binary LDPC codes is not an accurate measure of the decoding algorithm we can still use it as a tool to compare the behavior of the decoding algorithm at different mean column weights. We plot EXIT-

charts for non-binary LDPC codes with different mean column weight at several E_b/N_o. For mean column weight 2.8 and 2.6 we take the profile given in [17]. For the other mean column weight we randomly generate the profile by using column weights of weight 2 and weight 3. The convergence threshold is tabulated in Table 2.

Table 2: Convergence threshold of rate ½ non-binary LDPC codes with different mean column weights.

Mean Column Weight	Convergence threshold (dB)
2.6	0.5
2.8	0.4
3.0	0.65
3.2	0.7
3.4	0.8

From the above table we see that the best codes for the LDPC codes over GF(4) are the ones with mean column weight 2.8. This is exactly what we see from the results provided in [3]. We conclude that there is a consistency in the results obtained from the EXIT-charts and the procedure provided in [5], although the convergence threshold predicted by EXIT-charts is optimistic.

4.5 EXIT-Chart for PA Codes: The EXIT-charts for PA codes are plotted following the procedure described above for serial concatenated codes. We plot the EXIT curves for both the decoders on the same plot with axes swapped for the outer decoder.

The EXIT-chart for binary PA code at E_b/N_o =1.0 dB is shown below in figure 20

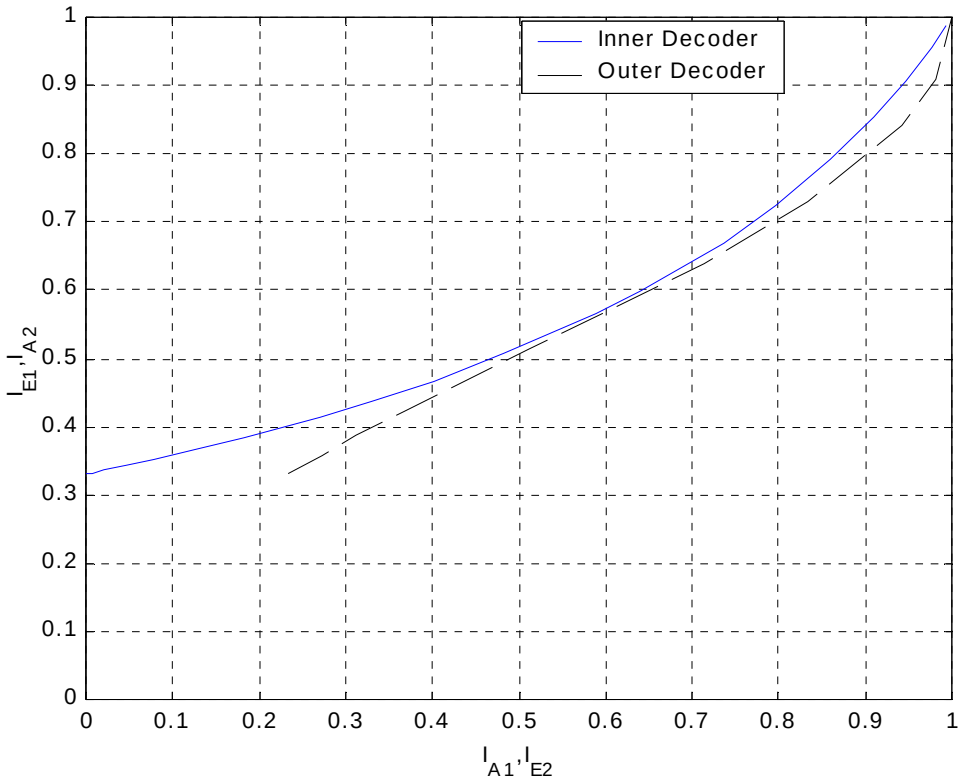

Figure 20: EXIT-chart for binary PA codes at E_b/N_o =1.0 dB

Figure 20 shows that the convergence threshold for binary PA codes is about 1.0 dB. This is in excellent agreement with the simulation results we see in figure 10. So we verify the claim that EXIT-charts accurately predict the convergence threshold for the binary codes. For non-binary case the EXIT-chart is plotted in figure 21.

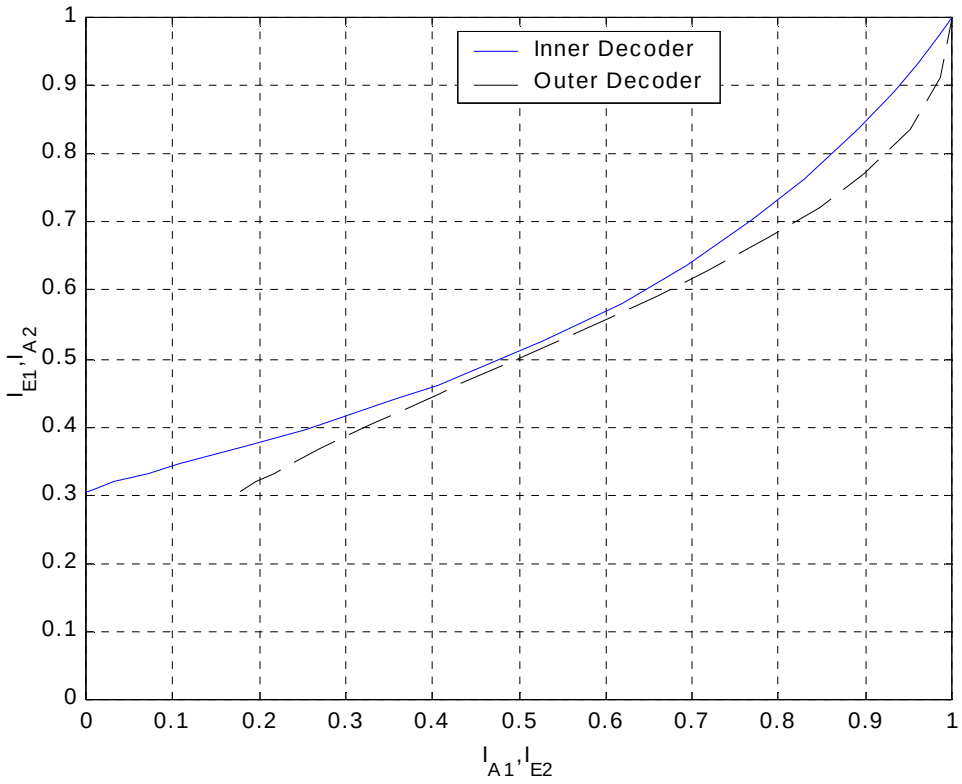

Figure 21: EXIT-chart for PA codes over GF(4) at E_b/N_o =0.7dB

From the above plot we see that there is a very small tunnel through which the stair case plot can still pass through. So the convergence threshold for non-binary PA code is about 0.7 dB. This would mean that there is a gain of 0.3 dB between binary PA codes and non-binary PA codes. But from the simulations we know that the convergence threshold of non-binary PA codes is about 0.92dB. So, once again we see that the convergence threshold predicted by EXIT-charts for the non-binary is optimistic. This can also be verified by plotting an actual decoding trajectory of a large block on the EXIT-chart.

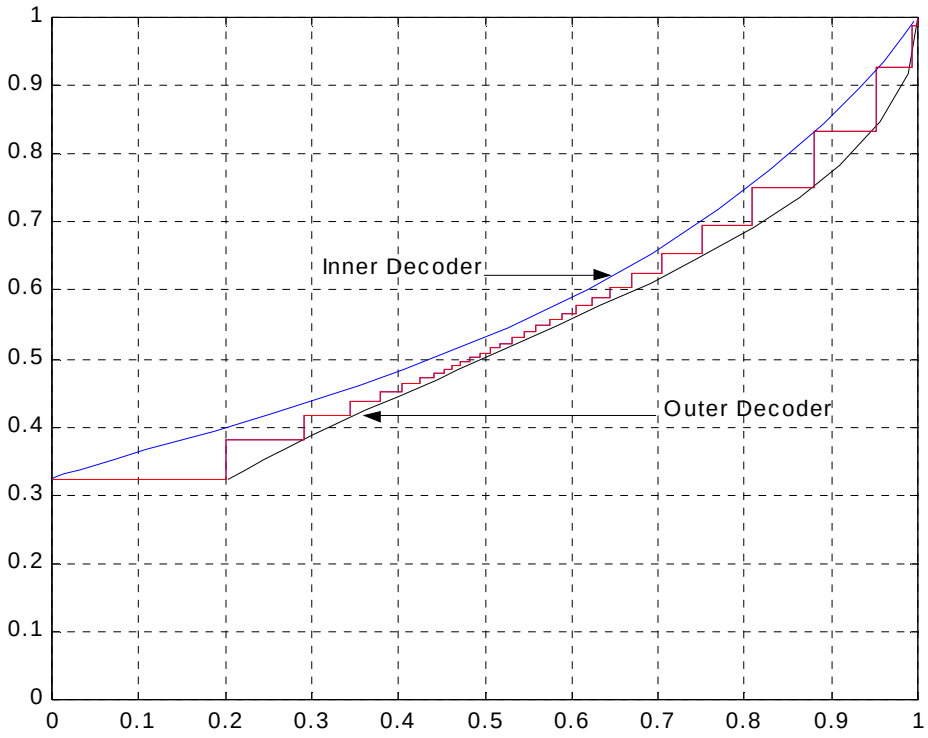

Figure 22: EXIT-chart with the actual decoding trajectory for rate ½ non-binary PA code at E_b/N_o =1.0dB

CHAPTER V
CONCLUSION

Product Accumulate code over GF(4) show a performance improvement of 0.2dB at short block length=2000 transmitted bits and rate =1/2 over binary Product Accumulate code. For long block lengths the performance improvement is quite small. The reason can be attributed to the absence of cycles at large block length for both the binary case and non-binary case. But for short block length there are cycles and by moving to non-binary fields the cycles in the corresponding graph is reduced. This explains the significant performance improvement of the non-binary PA codes.

The trellis based decoding proposed for PA codes reduces the complexity of decoding further by eliminating the need for multiplication used in message passing algorithm. Also the new decoding algorithm is numerically more stable than the message-passing algorithm.

EXIT-charts are used to explain why certain irregularities in non-binary LDPC codes perform better. We compare the convergence threshold of non-binary LDPC codes with different mean column weights. The Gaussian approximation used is optimistic and it does not predict the convergence threshold accurately. However, we can still use this technique to search for the non-binary LDPC code with the best mean column weight. We see that the best LDPC codes over GF(4) exists for mean column weight =2.8. We also use EXIT-chart to show that the convergence threshold for non-binary PA codes lie 0.3dB below the convergence threshold of binary PA codes.

REFERENCES

[1] R.G. Gallagher, "*Low-density parity-check codes*", MIT Press, Cambridge, MA 1963

[2] D. MacKay and R. Neal, "Near Shannon limit performance of low density parity check codes*" Electronics Letters ,* vol. 32, 18 , pp. 1645, Aug. 1996.

[3] J. Li, K.R. Narayanan and C.N. Georghiades, "Product accumulate codes: A class of capacity approaching codes with low decoding complexity", submitted to *IEEE Trans. Info Theory,* June 2001

[4] M. Davey and D. MacKay, "Low-density parity check codes over GF (q)" *IEEE Communications Letters,* vol. 2 , 6 , pp. 165 –167, June 1998.

[5] M. Davey and D. MacKay, "Low density parity check codes over GF (q)**"** *Information Theory Workshop,* Killarney, Ireland, pp.70 –71,1998.

[6] S. Ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes" *, IEEE Transactions on Communications,* vol. 49, 10, pp. 1727-1737, Oct. 2001.

[7] B. Scanavino, G. Montorsi and S. Benedetto, "Convergence properties of iterative decoders working at bit and symbol level" in *Global Telecommunications Conference*, San Antonio, *GLOBECOM '01. IEEE,* vol. 2, pp. 1037-1041, 2001.

[8] T.Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding", *IEEE Transactions on Information Theory,* vol. 47, 2 , pp. 599-618, Feb. 2001.

[9] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)"*, IEEE Transactions on Information Theory,* vol. 20, 2 , pp 284-287, May 1974.

[10] H. El Gamal, and Jr. A. Hammons, "Analyzing the turbo decoder using the Gaussian approximation", *IEEE Transactions on Information Theory ,* vol. 47, 2 , pp. 671-686, Feb. 2001.

Supplemental Sources Consulted

S. Benedetto and G. Montorsi, "Serial concatenation of interleaved codes: Analytical performance bounds" in *Global Telecommunications Conference,. GLOBECOM '96. 'Communications: The Key to Global Prosperity*, vol. 1, pp. 106-110, 1996.

C. Berrou and M. Jezequel, "Non-binary convolutional codes for turbo coding" *Electronics Letters,* vol: 35, 1, pp. 39-40, Jan. 1999.

C. Berrou, M. Jezequel, C. Douillard and S. Kerouedan, " The Advantages of non-binary turbo codes" , in Proc. *Information Theory Workshop,* Cairns, Australia, 2001, pp 61 –63.

G.C.Clark, Jr and J.B.Cain, *Error-correction coding for digital communications*, Plenum Press, New York,1981

P. Elias, "Error-Free coding," *IRE Transactions on Information Theory* , PGIT-4, pp 29-37, 1954.

T.J.Richardson, A.Shokrollahi and R.Urbanke, "Design of capacity approaching irregular low-density parity-check codes," *IEEE Trans.Inform Theory*, vol.47,pp.619-637, 2001.

VITA

Deepak Gilra was born May 22, 1977 in Cuttack, India. He received his Bachelor of Technology in electronics and electrical communication engineering from the Indian Institute of Technology (IIT), Kharagpur in May 2000. In fall of 2000 he started his masters program in the department of electrical engineering at Texas A&M University. His research was focused on studying LDPC codes and LDPC like codes. He may be contacted through email at d_gilra@yahoo.com, and his permanent address is Paper Palace, B.K.Road, Cuttack 753001,Orissa, India