

## ANALYSIS AND IMPLEMENTATION OF SOFT-DECISION DECODING ALGORITHM OF LDPC

M. M. Jadhav, Ankit Pancholi, Dr. A. M. Sapkal

Asst. Professor, E&TC, SCOE, Pune University  
Phase-I, D,-402, G.V 7, Ambegaon, Pune (INDIA).

**Abstract**—Low-density parity-check (LDPC) codes are freeware codes designed by Gallager in the year 1993 and used in most of the CDMA technology. An approach to study and analyze a mathematical model of belief propagation algorithm for decoding the encoding data stream with block size of 4096 using a binary symmetric channel to achieve good error probability of  $10^{-6}$  is simulated with 60 iterations. Due to large size of the matrix, the conventional block encoding method could require a significant number of computations. The algorithm will be implemented using an hardware platform of TMS320C6713 DSP processor based on the VLIW architecture with clock rate of 225MHz, well suited for arithmetic & logical calculations. A simulated algorithm adopts iterative approach with a code rate of one half where decoder operates alternatively on the bit nodes and the check nodes to find the most likely codeword giving a good error performance between  $10^{-4}$  and  $10^{-5}$  that is acceptable for video streaming on the internet. The experimental results shows that the dynamic range in computations is quite large resulting in numerical stability and intensive multiplication in the algorithm poses a challenge for implementation.

**Keywords**- Bit error rate, Belief propagation decoding, C6713 and BSC channel.

### I. INTRODUCTION

As the demand for video streaming services is constantly growing day by day which has motivated us to analyze and design algorithm to provide good error probability. The available bandwidth on mobile is limited and expensive thus we need some technology that utilizes the available spectrum more efficiently and providing good error correction capability. The implementation presented here is based on iterative decoding using belief propagation. The algorithm propagates soft probabilities of the bits between bit nodes and check nodes through the Tanner graph, thereby refining the confidence that the parity checks provide about the bits. A Tanner graph is a bipartite graph introduced to graphically represent codes. It consists of nodes and edges. The nodes are grouped into two sets. One set consists of  $n$  bit nodes and the other of  $m$  check nodes. The exchange of the soft probabilities is termed as belief propagation. The iterative soft-decision

decoding of code converges to true optimum decoding if the tanner graph contains no cycles. Therefore, we want LDPC code with as few cycles as possible.

An LDPC code is a linear block code defined by a very sparse parity check matrix, which is populated primarily with zeros and sparsely with ones. The LDPC code also showed improved performance when extended to non-binary code as well as binary code to define code words. The LDPC code yields a signal to noise ratio approaching a Shannon channel capacity limit, which is the theoretical maximum amount of digital data that can be transmitted in a given bandwidth in presence of certain noise interference.

#### A. LDPC Codes: Construction and Notation

To denote the length of the code we use  $N$  and  $K$  to denote its dimension and information bits  $M = N - K$ . Low density parity check codes are linear codes defined by a parity check matrix. We will consider binary codes, where all operations are carried out in the binary field. Since the parity check matrices we consider are generally not in systematic form, the symbol  $A$  is used to represent parity check matrices, reserving the symbol  $H$  for parity check matrices in systematic form. Following the general convention in the literature for LDPC codes, assume that vectors are column vectors. A message vector  $\mathbf{m}$  is a  $K \times 1$  vector; a codeword is a  $N \times 1$  vector. The generator matrix  $G$  is  $N \times K$  and parity check matrix  $A$  is  $(N-K) \times N$ , such that  $H.G = 0$ . The row of a parity check matrix as

$$A = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_M^T \end{bmatrix}$$

The equation  $a_i^T c = 0$  is said to be a linear parity-check constraint on the codeword  $\mathbf{c}$ . The notation  $z_m = a_m^T c$  where  $z_m$  is parity check or, a check. For a code specified by a parity check matrix  $A$ , it is necessary for encoding purposes to determine the corresponding generator matrix  $G$ . A systematic generator matrix may be found as follows. Using Gaussian elimination with column pivoting as necessary to determine an  $M \times M$  matrix  $A_p^{-1}$  so that

$$H = A_p^{-1} A = [I \quad A_2]$$

Having found  $H$ , form

$$G = \begin{bmatrix} A_2 \\ I \end{bmatrix}$$

Then  $HG = 0$ , so  $A_p HG = AG = 0$ , so  $G$  is a generator matrix for  $A$ . while  $A$  may be sparse, neither the systematic generator  $G$  nor  $H$  is necessarily sparse. A matrix is said to be sparse if fewer than half of the elements are nonzero. Parity check matrix should be such that no two columns have more than one row in which elements in both columns are nonzero.

B. Encoding of LDPC code

At this stage we will encode messages into codeword for LDPC codes which requires the generation of the parity check matrix  $H$ . The algorithms used in the construction of parity check matrix  $H$  will be discussed in the following subsections. The first encoding method is through the use of a generator matrix, denoted by  $G$ . The matrix  $G$  contains the set of constraints that form the parity check equations of the LDPC code. A codeword  $c$  is formed by multiplying a source message  $u$  by the matrix  $G$ . This is represented by the equation:  $c = u * G$ . For a binary code with  $k$  message bits and length  $n$  codeword the generator matrix  $G$  is a  $(k * n)$  binary matrix having the form  $= I_k | A^T$ . The row space of  $G$  will be orthogonal to  $H$  so that  $GH^T = 0$ . The process of converting  $H$  into the generator matrix  $G$  has the effect of causing  $G$  to lose the sparseness characteristic that was embodied in  $H$ .

The TMS320C6x family of processors is like fast special-purpose microprocessors with a specialized type of architecture and instruction sets suitable for signal processing. The TMS320C6713 DSK board is powerful and relatively cheap, having the necessary supporting tools for real-time data processing. The status of the four user dip switches on the DSK board can be read, which provides the user with a feedback control interface. The TMS320C6x are the first processors to use velocity architecture, having implemented the VLIW architecture. Figure 1 shows the system architecture and on board supports for practical implementation.

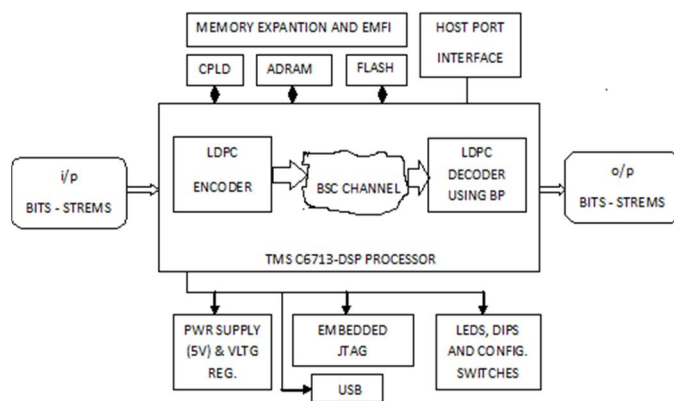


Figure1. System Architecture

The C6713 DSK is a low-cost standalone development platform that enables users to evaluate and develop applications for data manipulation, mathematical calculations,

and multiplies-accumulate operation called as MAC in a single instruction cycle. The MAC operation is useful in DSP algorithms that involve computing a vector dot product, such as digital filters, correlation and Fourier transforms. DSP processors often include several independent execution units that are capable of operating in parallel for FFT structure and reduce time to market.

Section-I of the paper contains basics of various terms regarding to LDPC code and requirements of efficient decoding algorithms. Literature survey is given in section-II. Section-III shows proposed belief propagation and it's significant. Mathematical calculations are also explained in IV section. Finally, Significance results are shown in section-V.

II. LITERATURE REVIEW

Basics of information theory, various encoding channel and decoding techniques are explained in [1] and [2] contains overview of LDPC code along with its structure and encoding schemes. In [3] Belief propagation for LDPC is described. Reference [6] represents the proposed architecture to achieve lowest error probability. Remaining papers describes the overview, architecture and programming part of TMS320C6713 DSP processor.

III. BELIEF PROPAGATION ALGORITHM

In 1981 Robert Tanner rediscovered LDPC codes in his work on the use of recursion to construct error correcting codes (Tanner, 1981). Tanner utilized bipartite graphs to describe the parity check matrix, which are now known as Tanner graphs, which display the incidence relationships between the variable codeword bits and the corresponding checksum tests. The graph  $G$  representing the parity check matrix  $H$  consists of two sets of vertices  $V$  and  $C$ . The set  $V$  consists of  $n$  vertices that represent the  $n$  codeword bits and are called variable nodes, denoted by  $v_0, v_1, \dots, v_{n-1}$ . Variable node index correspond to the column number of the parity check matrix. An edge is contained in the graph  $G$  if and only if the variable node  $v_n$  is contained in a parity check sum  $c_j$ . The Tanner graph for the parity check matrix is as shown in Figure 2.

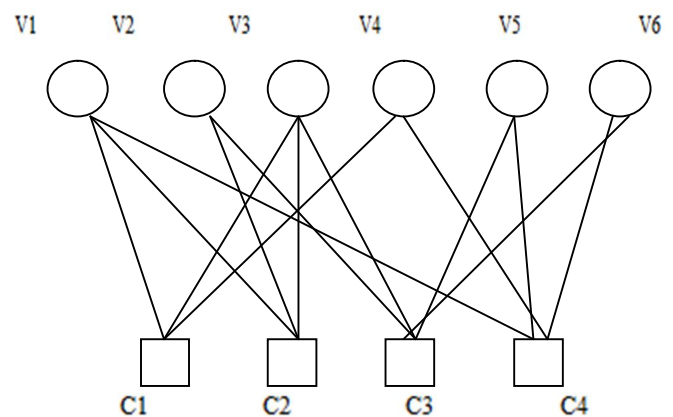


Figure 2. Tanner graph for LDPC code

Tanner graphs can be used to estimate codeword of an LDPC code  $C$  by iterative probabilistic decoding algorithms, based on either hard or soft decisions.

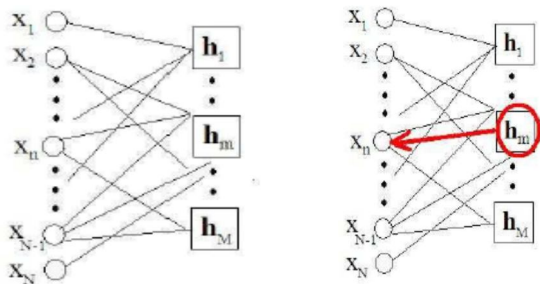


Figure 3. Message passing through Belief propagation [5]

It can be seen from above diagram that in right bound iteration messages are sent from variable nodes to check nodes and in left bound iteration messages are sent from check node to variable nodes.

BP decoding is an iterative process in which neighboring variables “talk” to each other, passing messages such as: “I (variable  $x$ ) think that you (check  $h$ ) belong in these states with various likelihoods”. After enough iteration, this series of conversations is likely to converge to a consensus that determines the marginal probabilities of all the variables. Estimated marginal probabilities are called beliefs. So BP algorithm is the process to update messages until convergence, and then calculate beliefs. Using only the sign bit of LLR ( $\lambda$ ), one can estimate the most probable value of  $C_n$ .

$$\lambda(X_n|y) = \log \frac{p(X_n=1|y)}{p(X_n=0|y)} = \log \frac{p(X_n=1|y_n, \{y_i: i \neq n\})}{p(X_n=0|y_n, \{y_i: i \neq n\})}$$

The Belief propagation algorithm is a soft decision algorithm which accepts the probability of each received bit as input. The input bit probabilities are called the *a priori* probabilities for the received bits because they were known in advance before running the LDPC decoder. The bit probabilities returned by the decoder are called the *a posteriori* probabilities. In the case of BP 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 matrix for a binary variable by a single value  $L(x) = \log\left(\frac{P(x=0)}{P(x=1)}\right)$ , where we use  $\log$  to mean  $\log_e$ . 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

$$P(x = 1) = \frac{e^{-Lx}}{1 + e^{-Lx}} \quad \text{and} \quad P(x = 0) = \frac{e^{Lx}}{1 + e^{Lx}}$$

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 aim of sum-product decoding is to compute the *maximum a posteriori probability* (MAP) for each codeword bit  $P_{i=P\{C_i=1|N\}}$ , which is the probability that the  $i^{\text{th}}$  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 BP 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.

Important Equations and Notations of BP in Logarithm domain

A. LLR for the BSC channel is given by

$$r_i = \log \frac{p}{1-p} \quad \text{if } y_i = 1$$

$$r_i = \log \frac{1-p}{p} \quad \text{if } y_i = 0$$

and for AWGN Channel

$$r_i = 4Y_i \frac{E_s}{N_0}$$

B. To begin decoding we set the maximum number of iterations and pass in  $H$  and  $r$ . Initialization is  $M_{i,j} = r$ . The extrinsic information from check node to bit node is given by

$$E_{j,i} = \log \frac{1 + \prod_{i \in B_j, i \neq i} (\tanh M_{j,i} / 2)}{1 - \prod_{i \in B_j, i \neq i} (\tanh M_{j,i} / 2)}$$

C. Each bit has access to the input a priori LLR,  $r_i$ , and the LLRs from every connected check node. The total LLR of the  $i$ -th bit is the sum of these LLRs:

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

However, the messages sent from the bit nodes to the check nodes,  $M_{i,j} = r$ , are not the full LLR value for each bit. To avoid sending back to each check node information which it already has, the message from the  $i$ th bit node to the  $j$ th check node is the sum in above equation without the component  $E_{j,i}$ , which was just received from the  $j$ -th check node:

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

Finally decision of received code word bits is made from the equation of  $L_i$ . If  $c.H^T = 0$ , then decoding is successfully completed at  $i$ th iteration because all the parity checks are satisfied otherwise it goes to maximum numbers of iterations.

#### IV. MATHEMATICAL CALCULATIONS

A. Problem Statement

Using sum-product perform the decoding and correct the error with  $C = [0 \ 0 \ 1 \ 0 \ 1 \ 1]$  is sent through a BSC with crossover probability  $p = 0.2$  and received word  $r = [1 \ 0 \ 1 \ 0 \ 1 \ 1]$ .

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

B. Thus the a priori probabilities for the BSC are  $r_i = \log \frac{p}{1-p}$  if  $y_i=1$  and  $r_i = \log \frac{1-p}{p}$  if  $y_i=0$

So,  $\text{Log} \frac{p}{1-p} = \log \frac{0.2}{0.8} = -1.3863$  and

$$\text{Log} \frac{p}{1-p} = \log \frac{0.8}{0.2} = 1.3863$$

And so the a priori log likelihood ratios are

$$r = [-1.3863, 1.3863, -1.3863, 1.3863, -1.3863, -1.3863].$$

C. To begin decoding we set the maximum number of iterations. Initialization is  $M_{i,j} = r$ , the 1st bit is included in the 1st and 3rd checks and so  $M_{1,1}$  and  $M_{3,1}$  are

$$M_{1,1} = r_1 = -1.3863 \text{ and } M_{3,1} = r_1 = -1.3863.$$

Repeating for the remaining bits gives:

- i = 2:  $M_{1,2} = r_2 = 1.3863$   $M_{2,2} = r_2 = 1.3863$
- i = 3:  $M_{2,3} = r_3 = -1.3863$   $M_{4,3} = r_3 = -1.3863$
- i = 4:  $M_{1,4} = r_4 = 1.3863$   $M_{4,4} = r_4 = 1.3863$
- i = 5:  $M_{2,5} = r_5 = -1.3863$   $M_{3,5} = r_5 = -1.3863$
- i = 6:  $M_{3,6} = r_6 = -1.3863$   $M_{4,6} = r_6 = -1.3863$

D. The extrinsic information from check node to bit node is given by

$$E_{j,i} = \log \frac{1 + \prod_{i \in B_j} \tilde{r}_i \neq i (\tanh M_{j,i} / 2)}{1 - \prod_{i \in B_j} \tilde{r}_i \neq i (\tanh M_{j,i} / 2)}$$

$$E_{1,1} = \log \left( \frac{1 + \tanh(M_{1,2} / 2) \tanh(M_{1,4} / 2)}{1 - \tanh(M_{1,2} / 2) \tanh(M_{1,4} / 2)} \right)$$

So,  $E_{1,1} = \log \left( \frac{1 + \tanh(1.3863/2) \tanh(1.3863/2)}{1 - \tanh(1.3863/2) \tanh(1.3863/2)} \right)$

$$E_{1,1} = \log \left( \frac{1+0.6*0.6}{1-0.6*0.6} \right) = 0.7538$$

Similarly, the extrinsic probability from the 1st check to the 2-nd bit depends on the probabilities of the 1st and 4th bits

$$E_{1,2} = \log \left( \frac{1 + \tanh(M_{1,1} / 2) \tanh(M_{1,4} / 2)}{1 - \tanh(M_{1,1} / 2) \tanh(M_{1,4} / 2)} \right)$$

So,  $E_{1,2} = \log \left( \frac{1 + \tanh(\frac{-1.3863}{2}) \tanh(\frac{1.3863}{2})}{1 - \tanh(\frac{-1.3863}{2}) \tanh(\frac{1.3863}{2})} \right)$

$$E_{1,2} = \log \left( \frac{1-0.6*0.6}{1+0.6*0.6} \right) = -0.7538$$

$$E_{1,4} = \log \left( \frac{1 + \tanh(M_{1,1} / 2) \tanh(M_{1,2} / 2)}{1 - \tanh(M_{1,1} / 2) \tanh(M_{1,2} / 2)} \right)$$

So,  $E_{1,4} = \log \left( \frac{1 + \tanh(-1.3863/2) \tanh(1.3863/2)}{1 - \tanh(-1.3863/2) \tanh(1.3863/2)} \right)$

$$E_{1,4} = \log \left( \frac{1 - 0.6 * 0.6}{1 + 0.6 * 0.6} \right) = -0.7538$$

Similarly, information from other check nodes can be counted and matrix E becomes:

$$\begin{bmatrix} 0.7538 & -0.7538 & . & -0.7538 & . & . \\ . & 0.7538 & -0.7538 & . & -0.7538 & . \\ 0.7538 & 0 & . & . & 0.7538 & 0.7538 \\ . & 0 & -0.7538 & 0.7538 & . & -0.7538 \end{bmatrix}$$

To test the intrinsic and extrinsic probabilities for each bit are combined. The 1st bit has extrinsic LLRs from the 1st and 3rd checks and an intrinsic LLR from the channel. The total LLR for bit one is their sum:  $L_1 = r_1 + E_{1,1} + E_{3,1} = -1.3863 + 0.7538 + 0.7538 = 0.1213$ .

Thus even though the LLR from the channel is negative, indicating that the bit is a one, both of the extrinsic LLRs are positive indicating that the bit is zero. The extrinsic LLRs are strong enough that the total LLR is positive and so the decision on bit one has effectively been changed. Repeating for bits two to six gives:

- $L_2 = r_2 + E_{1,2} + E_{2,2} = 1.3863 - 0.7538 + 0.7538 = 1.3863$
- $L_3 = r_3 + E_{2,3} + E_{4,3} = -1.3863 - 0.7538 - 0.7538 = -2.8938$
- $L_4 = r_4 + E_{1,4} + E_{4,4} = 1.3863 - 0.7538 + 0.7538 = 1.3863$
- $L_5 = r_5 + E_{2,5} + E_{3,5} = -1.3863 - 0.7538 + 0.7538 = -1.3863$
- $L_6 = r_6 + E_{3,6} + E_{4,6} = -1.3863 + 0.7538 - 0.7538 = -1.3863$

The hard decision on the received bits is given by the sign of the LLRs,  $Z = [0 0 1 0 1 1]$ .

E. To check if z is a valid codeword

$$S = Z.H^T = [0 0 1 0 1 1] \cdot \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [0 0 0 0 0 0].$$

This indicates that since S is zero Z is a valid codeword, and the decoding stops, returning Z as the decoded word. The above procedure is continued until the decided bits form a valid codeword or until a maximum number of iterations are reached, whichever occurs first.

### V. RESULTS

It can be seen from the simulated graph that error probability of  $10^{-4}$  to  $10^{-5}$  is obtained with iteration count of 60 and code rate of one half using Binary Symmetric Channel. From the



graph it is seen that the minimum distance of an LDPC code increases with increasing code length and at the same time the error probability decreases exponentially. In other words a parallel architecture where computations can be performed independently for all bit nodes (or check nodes) can be the best choice to achieve the highest throughput and where there is no need for large memory space.

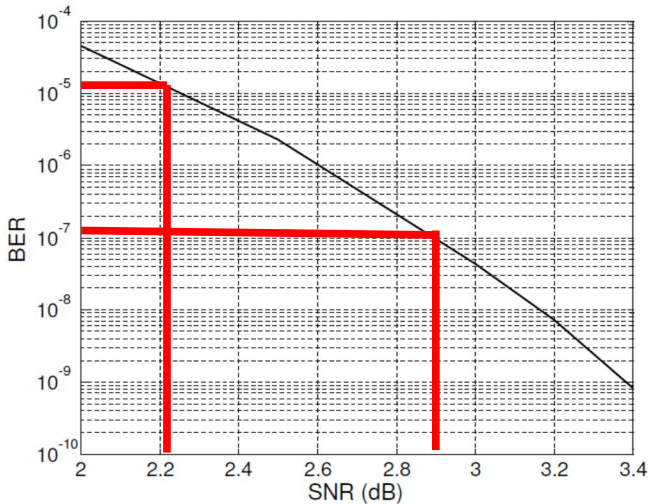


Figure 4. BER performance of LDPC code over BP decoding

#### VI. CONCLUSIONS

In many cases these LDPC codes are designed with a higher code rate or with a lower error rate. In this paper by using Belief propagation decoding algorithm is simulated and implemented on hardware platform TMS320C6713 DSP processor. An error probability in the range of  $10^{-4}$  to  $10^{-5}$  is achieved using BPSK Modulation for the signal to noise ratio of 2.2 dB to 2.9 dB as compare to other conventional codes. LDPC codes will be utilized more often in future in all forms of wireless communications.

#### ACKNOWLEDGEMENT

The authors would like to thank the author Yuan Jiang for providing us a good book with solved examples on Belief Propagation which was helpful in understanding the algorithms, as well as H. M. Jadhav for her careful reading and detailed suggestions that helped to improve the paper.

#### REFERENCES

- [1] R. G. Gallager, "Low density parity check codes," IEEE Trans. Inform. Theory, vol. IT-8, pp. 21–28, Jan. 1962.
- [2] Saeedi and Hamid, "Performance of Belief Propagation for Decoding LDPC Codes in the Presence of Channel Estimation Error ". Communication IEEE transaction on Dept. of Syst. & Computer. Eng., Carleton Univ., Ottawa, Ont. vol. IT-55, pp. 83–87, Jan. 2007.
- [3] TMS320C6000 Code Composer Studio Tutorial, SPRU301C, Texas Instruments, Dallas, TX, 2000.
- [4] TMS320C6713 DSK Technical Reference, 506735-0001 Rev.A, May,2003.

[5] M.P.C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," IEEE Trans. on Comm., vol. 47,no. 5, pp. 673 –680, May. 1999.

[6] "Introducing Low-Density Parity-Check Codes", by Sarah J. Johnson School of Electrical Engineering and Computer Science The University of Newcastle, Australia-2007.

[7] TMS 320C6713- A Floating point digital signal processor Lab Manual, SPRS186L – DECEMBER 2001 – REVISED NOVEMBER 2005.