Computationally Simpler And Fast Convergence Algorithm For Neural Network Based Ldpc Encoder/ Decoder

Rajasekar.B, Logashanmugam.E, Nandhitha. N.M

Abstract- Soft computing technique for computationally less complex encoder / decoder for LDPC is proposed. The novel learning technique is computationally less complex than Ordinary Gradient Learning (OGLN) and highly accurate than AGL. Performance of the proposed technique is compared with the conventional techniques in terms of maximum error, minimum error and computational complexity. In order to emulate a codec Artificial Neural Network based LDPC encoder/decoder is developed. The performance of the proposed LDPC codec is compared with that conventional codecs in terms of learning algorithm. The proposed learning algorithm has X-L multiplication in contrast to X2 and X multiplication of the conventional techniques.

KEYWORDS: Back propagation, CODEC, computational complexity, LDPC, Perceptron.

1 INTRODUCTION

LDPC is an example for block code in which message bits are coded with a Parity-Check Matrix (H matrix). Parity-Check Matrix is generated with a large number of zeros and less number of ones (Gallager R. G. (1962), Chu-Hsiang Huang et al,2014, Daisuke, M.,(2014)), . Number of zeros in the matrix affects the decoding complexity and determines the minimum distance of the code. Degree distribution refers to the number of non-zero entries in each of the rows and columns of the H matrix (Guohua Zhang et al, 2013). If the degree of distribution of both rows and columns are uniform then it is a regular H matrix, otherwise it is called irregular H-matrix. The pictorial representation of the H-matrix is called the Tanner graph (Alexios Balatsoukas, Andreas Burg (2014)). A cycle is defined as the sequence of connected nodes that start and end at the same node and girth refers to the smallest cycle in Tanner graph. The encoded message that consists of the message and redundant bits is called code word. A novel modified SALT technique for LDPC encoders is described. The proposed technique used lesser number of computations in order to encode the message bits. However in the case of transceivers, it will be most appropriate if the same circuit could perform both encoding and decoding in LDPC. Reviriego et al (2013) studied the decoding logic for a class of Euclidean geometry low density parity check (EG-LDPC) codes using a similar technique to that of one step majority logic decodable. Majority logic decoding was implemented serially with simple hardware but it requires a lengthy decoding time. Imemory applications, this increased the memory access time. The method detects whether a word had errors in the first iterations of majority logic decoding, and when there are no errors in the decoding it ended without completing the rest of the iterations. Since most words in the memory will be error-free, the average decoding time is severely reduced.

- *1Associate Professor, 2,3Professor & Dean*
- *1,2,3Sathyabama Institute of Science and Technology, Deemed to be University,Chennai,india*
- *Email : 1rajasekar.ece@sathyabama.ac.in,*

Performance of Quasi-Cyclic Low-density parity check codes could be further accelerated using enhanced logic decoding like majority logic decoding. Hanghang Qi and Norbert Goertz (2013) proposed an efficient encoding technique with greedy permutation algorithm. For changing the H-matrix they used Approximate Lower Triangular (ALT) form by two ways namely Systematic Encoding by Gaussian elimination and Encoding with a Complexity that grows approximately linear with the block size. Then the Systematic ALT form (SALT form) is used to change the H-matrix. The greedy permutation is used for encoding and the performance is compared with the results of Gaussian and greedy permutation (Hua Xiao and Mehdi Karimi (2013)). In this work, a successful attempt has been made to develop a neural network based technique for both encoding and decoding. The pilot idea was borrowed from Ordinary Gradient Learning Network (OGLN) and Adaptive Natural Gradient learning (AGLN) algorithms proposed by Michael R Bastian in his Ph.D dissertation (2009). However the proposed technique outperforms the two algorithms cited in his dissertation in terms of both computational simplicity and convergence. BASTIAN'S OGLN AND AGLN TECHNIQUES Bastian proposed OGLN and AGLN for supervised learning in Multi Layered Perceptron (MLP) network. This neural network is used for prediction and classification of output variables from a set of input variables. Neural work cited in the thesis has three layers namely input layer, one hidden layer and an output layer. Number of neurons in the input and the output layers are dependent on the type of the LDPC encoder. In this case, a 5 input-10 output LDPC encoder is considered. Hence number of neurons in the input and the output layers are 5 and 10 respectively. Neurons in the hidden layer increase the accuracy of prediction. But it increases the area complexity of the system. Hence in this case, number of neurons in the hidden layer is chosen as 10. The existing neural network was simulated and codes are predicted using both OGL and AGL techniques. The minimum, maximum and average errors in predicting the 10 bit code from a 5-bit input message are shown in Table 1. The number of multiplications is also shown in the last column of the Table. It is found that AGLN provides lesser average

error, minimum and maximum error. However the number of computations is high in the case of AGLN based network.

Table 1 *The minimum, maximum and average error for Input message*

Method	Sum Error	Average error	Minimum error	Maximum error	No. of computations
OGL	14.3392	0.0448	0.0014	0.9716	X multiplications
AGL	8.8542	0.0277	5.4360e-005	0.9986	X multiplications

MODIFIED AGLN In order to compare the performance of the proposed algorithm with the conventional techniques, initially the architecture of the neural network is retained. Also the feasibility of using the learning techniques for decoding the message bits from the encoded bits is studied. Performance of the proposed learning technique (decoder) is also compared with the conventional learning algorithms. The proposed multi layered perceptron has an input layer, a hidden layer and an output layer.

3.1 NEURAL NETWORK BASED LDPC ENCODER

In the case of encoder, number of neurons in the hidden layer is chosen as 10. Number of neurons in the input and output layers are 5 and 10 respectively. The architecture of proposed neural network based encoder is shown in Figure The proposed perceptron based LDPC encoder is capable of generating a 10 bit code for any of the 32 combinations of the input message stream ranging from 00000B to 11111B. The Parity Check Matrix is again 32 combinations of five bits. In this particular case, the 10 bit codeword is obtained by appending the message bits with the code bits. However the code words can also be generated by considering a 10×5 Parity Check Matrix and perform modulo 2 addition on the ended message and corresponding code bits.

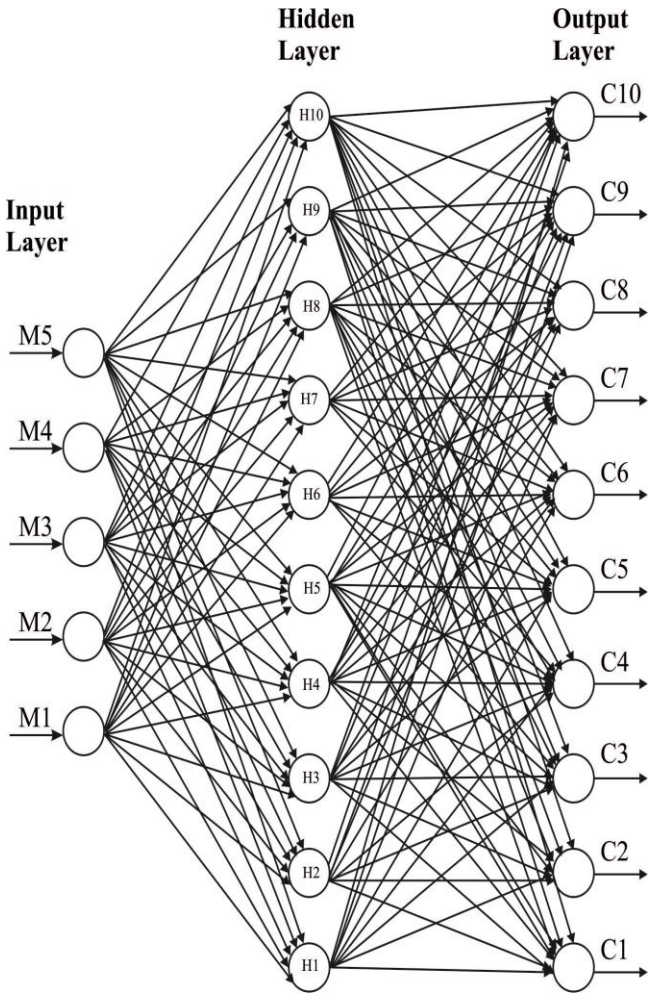

Figure 1 *Proposed Neural network based LDPC encoder*

The learning parameter is 0.0001 and the momentum parameter is 0.01. A bias term is included for all the neurons in the hidden and the output layers. The network is designed in such a way to receive the message block and the Parity Check Matrix also as input parameters. The codewords are the net outputs of the neurons in the output layer. An example of the exemplar set (message bits and the corresponding codewords) is shown in Tables 2.

Tables 2 *An exemplar set of message bits and the corresponding codewords*

Input message bits					Output encoded bits									
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	1
0	0	0	1	0	1	1	0	1	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	0	0	0	0	1	1
0	0	1	0	0	1	0	1	0	1	0	0	1	0	0
0	0	1	0	1	0	1	0	0	1	0	0	1	0	1
0	0	1	1	0	0	1	1	1	1	0	0	1	1	0
0	0	1	1	1	1	0	0	1	1	0	0	1	1	1
0	1	0	0	0	0	0	1	1	1	0	1	0	0	0
0	1	0	0	1	1	1	0	1	1	0	1	0	0	1
0	1	0	1	0	1	1	1	0	1	0	1	0	1	0
0	1	0	1	1	0	0	0	0	1	0	1	0	1	1
0	1	1	0	0	1	0	0	1	0	0	1	1	0	0
1	0	0	1	1	0	1	1	0	1	1	0	0	1	1
1	0	1	0	0	1	1	1	1	0	1	0	1	0	0
1	0	1	0	1	0	0	0	1	0	1	0	1	0	1

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

In the Table 2, the first few columns denote the message bits and the next ten columns denote the code bits. The first five bits of every column is fed as input and the target is set as the corresponding codeword. The learning techniques are then used to minimize the error between the predicted and desired outputs of every output layer neuron. Training is performed for a predetermined number of iterations. Relationship between the actual and the desired output of every output layer neuron for all the 32 combinations is shown in Table 3 for the proposed learning algorithm. From the Table 3, it is found that there is a slight mismatch between the actual and the desired values. Performance is measured in error sum, Mean absolute Error and maximum error. Also computational complexity is measured in terms if the number of multiplications during the execution of the program. These values are shown in Table 4. From the Table 4, it is found that the Error sum, Mean Absolute Error and the maximum error are lesser than the AGL techniques. Also the computational complexity has reduced by L multiplications when compared to OGL technique which in turn involves lesser computation than AGL.

***Table** 3Relationship between the Desired (D) and the Actual (A) values*

C10		C9		C8		C7		C6		C5		C4		C3		C2		C1	
D	A	D	A	D	A	D	A	D	A	D	A	D	A	D	A	D	A	D	A
0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	1
1	0	1	0	1	1	0	0	0	0	0	1	0	0	0	1	0	1	1	0
1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	1	0	0	1
0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0
1	0	0	0	1	1	0	1	1	0	0	0	0	0	1	0	0	1	0	1
0	0	1	1	0	1	0	1	1	0	0	0	0	1	1	1	0	0	1	1
0	1	1	1	1	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0
1	1	0	1	0	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0
1	0	1	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1
1	0	1	1	1	1	0	0	1	1	0	1	1	0	0	0	1	1	0	1
0	1	0	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	1	0
1	1	0	0	0	0	1	0	0	1	0	1	1	0	1	0	0	1	0	0
0	0	1	0	1	0	1	1	0	0	0	1	1	0	1	1	0	0	1	1
0	1	1	1	0	0	0	1	0	1	0	0	1	1	1	0	1	0	0	1
1	1	0	0	1	1	0	1	0	0	0	1	1	1	1	0	1	1	1	0
0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	1	0
1	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	1
0	1	1	0	1	0	0	1	1	0	1	0	0	1	0	1	1	0	1	0
1	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	0	1	0	1
0	1	0	1	0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	0
0	1	0	0	1	1	0	1	0	1	1	1	0	0	1	0	1	0	0	1
1	1	1	0	0	1	0	0	0	0	1	0	0	1	1	1	1	1	1	1
0	0	1	0	1	1	0	0	0	1	1	1	1	1	0	1	0	0	0	1
1	0	0	1	0	1	0	1	0	1	1	0	1	0	0	1	0	1	1	0
1	1	0	1	1	0	1	1	0	0	1	0	1	0	0	0	1	1	0	0
0	0	1	0	0	1	1	1	0	1	1	0	1	1	0	0	1	0	1	0
1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1
0	1	0	0	1	0	0	0	1	0	1	1	1	1	1	0	0	0	1	1

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

***Table 4** Performance evaluation of the proposed technique*

Method	Error Sum	Average error	Maximum error	No. of computations
OGL	14.3392	0.0448	0.9716	X multiplications
AGL	8.8542	0.0277	0.9986	X2 Highly complex
Proposed	7.0177	0.0219	0.9969	X-L multiplications

3.2 **NEURAL NETWORK BASED DECODER**

In order to realize the decoder, the number of neurons in the input layer is 10 and the number of neurons in the output layer is 5. Number of neurons in the hidden layer is 10. The proposed architecture for the neural network based decoder is shown in Figure 2.

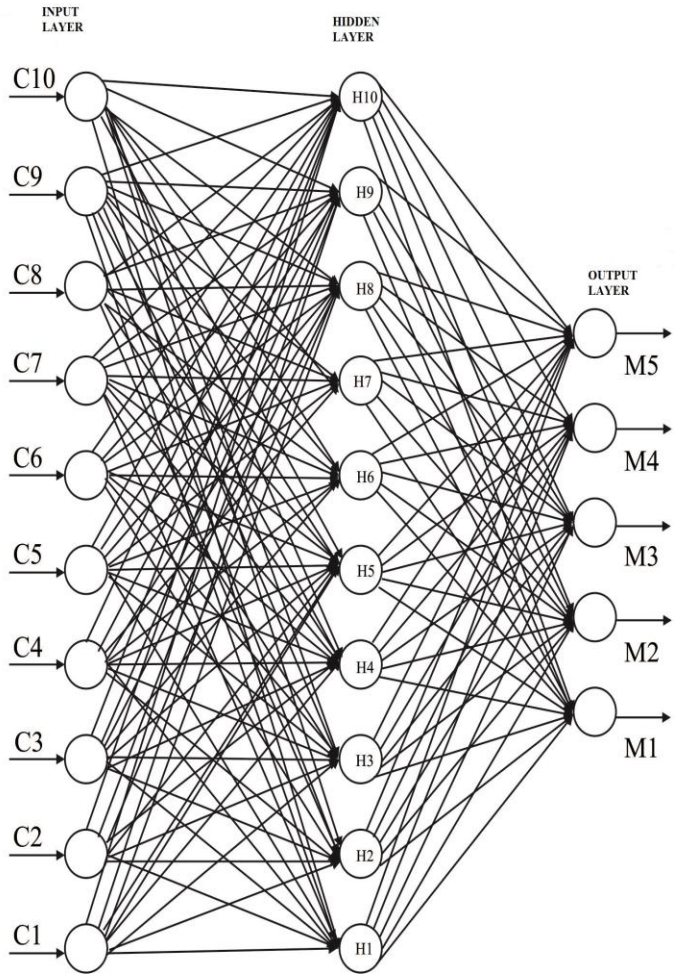

***Figure 2** Proposed neural network based LDPC decoder*

A set of exemplars used for training the neural network is shown in Table 5. Ten bit codeword is used as the input and the 5 bit message is used as the target for training the neural network based decoder. Similar to the encoder, the algorithm is executed for a predetermined number of times determined by the iterations. Learning and the momentum parameters are left unchanged. Relationship between the desired and the actual values are shown in Table 6.

Table 5 *Exemplar used for training the decoder*

Input codeword										Output message				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	0	0	0	0	1
1	1	0	1	0	0	0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0	0	0	1	1
1	0	1	0	1	0	0	1	0	0	0	0	1	0	0
0	1	0	0	1	0	0	1	0	1	0	0	1	0	1
0	1	1	1	1	0	0	1	1	0	0	0	1	1	0
1	0	0	1	1	0	0	1	1	1	0	0	1	1	1
0	0	1	1	1	0	1	0	0	0	0	1	0	0	0
1	1	0	1	1	0	1	0	0	1	0	1	0	0	1
1	1	1	0	1	0	1	0	1	0	0	1	0	1	0
0	0	0	0	1	0	1	0	1	1	0	1	0	1	1
1	0	0	1	0	0	1	1	0	0	0	1	1	0	0
0	1	1	1	0	0	1	1	0	1	0	1	1	0	1
0	1	0	0	0	0	1	1	1	0	0	1	1	1	0
1	0	1	0	0	0	1	1	1	1	0	1	1	1	1
0	1	0	1	1	1	0	0	0	0	1	0	0	0	0
1	0	1	1	1	1	0	0	0	1	1	0	0	0	1
1	0	0	0	1	1	0	0	1	0	1	0	0	1	0
0	1	1	0	1	1	0	0	1	1	1	0	0	1	1
1	1	1	1	0	1	0	1	0	0	1	0	1	0	0
0	0	0	1	0	1	0	1	0	1	1	0	1	0	1
0	0	1	0	0	1	0	1	1	0	1	0	1	1	0
1	1	0	0	0	1	0	1	1	1	1	0	1	1	1
0	1	1	0	0	1	1	0	0	0	1	1	0	0	0
1	0	0	0	0	1	1	0	0	1	1	1	0	0	1
1	0	1	1	0	1	1	0	1	0	1	1	0	1	0
0	1	0	1	0	1	1	0	1	1	1	1	0	1	1
1	1	0	0	1	1	1	1	0	0	1	1	1	0	0
0	0	1	0	1	1	1	1	0	1	1	1	1	0	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 6 *Relationship between the desired and the actual outputs*

M5		M4		M3		M2		M1	
D	A	D	A	D	A	D	A	D	A
0	0	0	0	0	1	0	1	0	1
0	1	0	0	0	1	0	1	1	0
0	0	0	0	0	1	1	0	0	1
0	0	0	1	0	1	1	1	1	0
0	0	0	0	1	0	0	1	0	1
0	0	0	1	1	1	0	0	1	1
0	0	0	1	1	0	1	1	0	0
0	1	0	0	1	1	1	1	1	1
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1
0	1	1	0	0	0	1	1	0	1
0	0	1	0	0	1	1	0	1	0
0	1	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	1	1
0	0	1	1	1	0	1	0	0	1
0	1	1	1	1	0	1	1	1	0
1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	1	0
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0
1	0	0	1	1	0	0	1	0	1
1	1	0	0	1	1	0	0	1	0
1	1	0	0	1	0	1	0	0	1
1	0	0	1	1	1	1	1	1	1
1	1	1	1	0	1	0	0	0	1

From the corresponding columns of Table 6, it is found that there is a slight variation between the desired and the actual message bits.

NEURAL NETWORK BASED LDPC CODEC

Having proved that the proposed learning technique is less complex in terms of both computation and area and highly accurate, a successful attempt has been made to develop prototype architecture for LDPC encoder/decoder using neural network. The prototype architecture is shown in Figure 3.

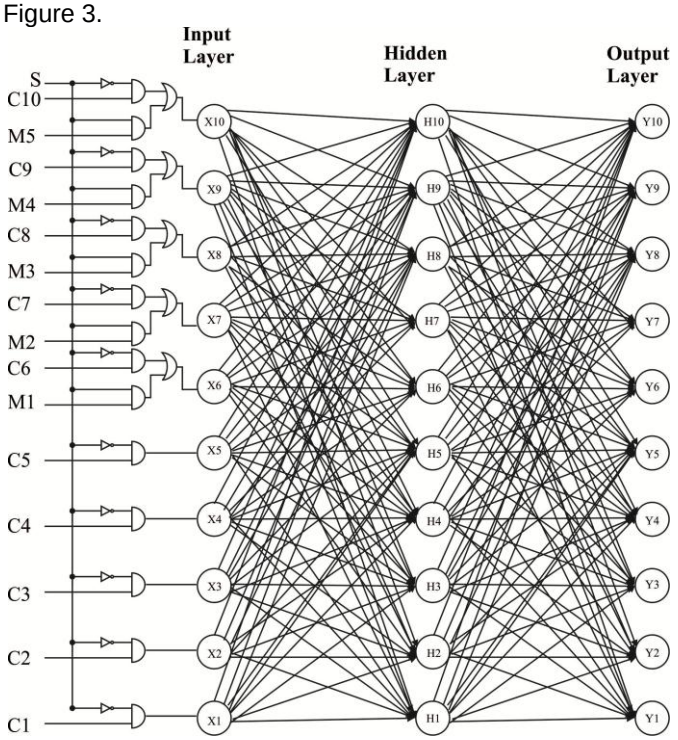

Figure 3 *Prototype architecture*

A select bit (s) is used for deciding the proposed encoder/decoder for a particular function i.e. either as encoder or decoder. If s=0, it acts as an encoder and if s=1, it acts as a decoder. In this work, a modified less complex, highly accurate learning algorithm is developed for LDPC decoding. Also a prototype model has been successfully developed for LDPC encoding/decoding. The accuracy of the proposed system is much higher than the conventional learning algorithms.

CONCLUSION AND FUTURE WORK

In this paper, a novel learning algorithm has been proposed for the LDPC codec. The proposed algorithm is computationally less complex than Adaptive Natural Gradient learning (AGLN) and highly accurate than Ordinary Gradient Learning (OGLN). An efficient architecture is also proposed for the neural network to act

as a universal LDPC codec by selecting either 0/1 for the set bit. However this work has stopped at the simulation stage itself. A dedicated hardware can be developed for the universal LDPC codec in future. Also simulated annealing can be used instead of Gradient Descent Rule.

REFERENCES

[1] Alexios Balatsoukas, Andreas Burg (2014), Density Evolution for Min-Sum Decoding of LDPC Codes Under Unreliable Message Storage", IEEE Communications Letters, Accepted For Publication, IEEE Communications Letters, pp.1-4.

[2] Chu-Hsiang Huang, Yao Li and Lara Dolecek Gallager B. (2014), "LDPC Decoder with Transient and Permanent Errors", IEEE Transactions on Communications, Vol. 62, No. 1, pp.15-28.D

[3] aisuke, M., Kazunori, H. and Ryo, Y. (2014), "An LDPC Decoder With Time-Domain Analog and Digital Mixed-Signal Processing", IEEE Journal of Solid-State Circuits, Vol. 49, No. 1, pp. 73-83.

[4] Gallager R. G. (1962), "Low-density parity-check codes," IRE Trans. Inform. Theory, Vol. IT-8, pp. 21-28.

[5] Rajasekar, B., Logashanmugam, E, (2014), Design and development of an improved split row.

[6] decoding algorithm with reduced BER , Research Journal of Applied Sciences, Engineering and Technology,

[7] Hanghang Qi and Norbert Goertz (2013), "Low-Complexity Encoding of LDPC Codes: A New Algorithm and its Performance". available at publik. tuwien. ac. at/files/PubDat_166941. pdf,(06. 04. 2011) (2013).

[8] Guohua Zhang, Rong Sun and Xinmei Wang (2013), "Construction of girth-eight QC-LDPC codes from greatest common divisor", IEEE Communications Letters, Vol. 17, No. 2, pp.369-372.

[9] Rajasekar, B., Logashanmugam, E. (2014), Euclidean geometry LDPC codes for error.

[10] correction in memory devices, International Journal of Applied Engineering Research.

[11] Hua Xiao and Mehdi Karimi (2013), "Error Rate Estimation of Low-Density Parity-Check Codes Decoded by Quantized Soft-Decision Iterative Algorithms", IEEE transactions on communications, Vol. 61, No. 2, pp.474.

[12] Reviriego Pedro, Juan A. Maestro and Mark F. Flanagan (2013), "Error detection in majority logic decoding of Euclidean geometry low density parity check (EG-LDPC) codes", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, Vol.21, No. 1, pp. 156-159.

[13] Michael R. Bastian,(2009),"Neural Networks And The Natural Gradient, Ph.D Dissertation Utah State University, Logan, Utah.