Performance Analysis of Min-Sum LDPC Decoding Algorithm

S. V. Viraktamath[1], Girish Attimarad[2]

[1]Department of ECE, SDM College of Engineering and Technology, Dharwad, India
[2]Department of ECE, Dayanand Sagar College of Engineering, Bangalore, India

Abstract— *In this paper the performance of Min-Sum LDPC algorithm is analyzed. A parallel software implementation of low density parity check decoding algorithm is proposed, a modified version of Min-Sum algorithm (MSA) has been used for the decoding. Specifically, Open Multi-Processing (OpenMP) for parallelizing software on a multi-core processor. We process information on H-matrices using OpenMP pragmas on a multi-core processor and execute decoding algorithms in parallel using MATLAB EXecutable (MEX) function in MATLAB. We evaluated the performance of the proposed implementation with respect to single-core processor execution and verified that the proposed parallel execution reduces the execution time and yields better results compared to single-core processor execution.*

Keywords— Tanner graph, bit nodes, check nodes, LDPC Codes, code length.

I. INTRODUCTION

Nowadays everyone uses the electronic gadgets which support wireless communication. In order to have a reliable communication error correcting codes must be used. Error correcting codes basically introduce or insert redundancy into the transmitted data stream so that the receiver can detect and possibly correct the errors that occurred during the transmission. The LDPC codes are known for their performance.

The LDPC codes are basically linear block codes and are devised by Gallager in 1960 [1]. IT was difficult to implement LDPC codes using the technology available. The LDPC code was revised by Mackay and Neal [2]. The LDPC code can approach the Shannon limit [3]. Furthermore, iterative LDPC decoding schemes based on the Sum-Product Algorithm (SPA) can fully be parallelized, leading to high-speed decoding. For these reasons, LDPC coding is widely regarded as a very attractive coding technique for high-speed 4G wireless communications. LDPC codes are used in many standards, and they support multiple data rates for each standard. LDPCs are linear (N, K) block codes defined by parity-check sparse binary H matrices of dimension M *N, with M =N - K. They are usually represented by bipartite graphs formed by Bit Nodes (BNs) and Check Nodes (CNs) and linked by bidirectional edges, also called Tanner graph [2]. LDPC decoding is based on the belief propagation of messages between connected nodes as indicated by the Tanner graph, which demands very intensive computation running the Sum-Product Algorithm (SPA), or its simplified variants, namely the Logarithmic-SPA (LSPA) and the Min- Sum Algorithm [4]. More flexible solutions for LDPC decoding using Digital Signal Processors (DSPs) or Software Defined Radio (SDR) programmable hardware platforms [5] have already been proposed. OpenMP [6] provides an effective and relatively straightforward approach for programming general-purpose multi-cores and was selected under the context of this work.

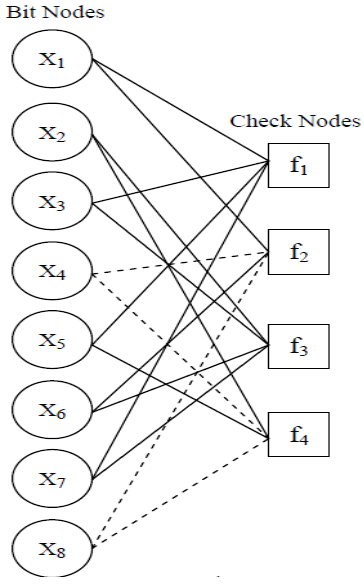
Figure 1. The performance of the LDPC-STBC is analyzed by using Density Evolution (DE). Also, the irregular LDPC codes for the LDPC-STBC optimized by using DE. The error rate performance of the optimized irregular LDPC codes and the regular LDPC codes for the LDPC-STBC has been analyzed in [7]. The estimation of Block-LDPC coding system implementation key metrics including the throughput and hardware complexity for both encoder and decoder are presented [8]. The use of low-density parity check (LDPC)-centric error correction coding (ECC) for magnetic recording read channel in the presence of significant burst errors is reported in [9]. Since an LDPC code by itself is severely vulnerable to burst errors due to its soft-decision probability-based decoding, they focused on LDPC-centric concatenated coding in which LDPC code is used as inner code. Min-Sum and Min-Sum with correction factor algorithms are reviewed [10] and adapted with TS-LDPC codes for future analog VLSI implementation. Three different

platforms for simulating the error performance of LDPC-CCs have been created [11]. The first two platforms are run on a Central processing Unit (CPU) while the third one involves the use of a Graphics Processing Unit (GPU). It has been shown that using GPU can improve the simulation speed substantially.

Nowadays, LDPC block codes (LDPC-BCs) have been adopted by several standards such as IEEE 802.16 (WiMAX), IEEE 802.3an (10GBASE-T), and IEEE 802.15.3c. As the counterpart of LDPC-BCs, LDPC convolutional codes (LDPC-CCs) [12] are more suitable for streaming video and variable length packets based on following features: 1) simple encoding process, 2) regular decoder architecture, 3) powerful decoding performance, 4) flexible code rates. The stochastic computation makes the decoding of LDPC-Convolutional Codes more efficient, but the boundary effect of sliding window causes poor performance [13].

II. REVIEW OF LDPC CODES

Low Density Parity Check codes are a class of linear block codes corresponding to the parity check matrix H. Parity check matrix H(N-K)xN consists of only zeros and ones and is very sparse which means that the density of ones in this matrix is very low.



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

Figure 1. Tanner Graph representation.

LDPC codes can be represented effectively by a bi-partite graph called a “Tanner" graph. A bi-partite graph is a graph (nodes or vertices are connected by undirected edges) whose nodes may be separated into two classes, and where edges may only be connecting two nodes that residing in the same class. The two classes of nodes in a Tanner graph are “Variable Nodes or Bit Nodes” and “Check Nodes”.

The Tanner graph of a code is drawn according to the following rule: “Check node fj, j=1,…..N-K is connected to bit node xi, i=1,……N whenever element hij in H (parity check matrix) is a one”. Fig. 1 shows a Tanner Graph made for a simple parity check matrix H. In this graph each Bit node is connected to two check nodes (Bit degree = 2) and each check node has a degree of four.

As mentioned by Gallager [1], the H-matrix should be very sparse. It also determines the complexity of the encoder/decoder. Depending on the platform which is going to do the encoding/decoding process, this matrix can be random or structured. The generator matrix is shown in equation 1. Then the code will be equal to c=mG.

$$G = [I|P] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad \text{--(1)}$$

A. A. Steps for LDPC encoding

- Loading the H matrix
- Finding check-node_ones and variable-node ones
- Enter SNR rnage
- Conversion from dB to decimal
- Defining random data input for encoding
- Generation of code word
- BPSK Modulation
- Addition of noise to the random data
- Transmission for the signal

B. B. Steps for LDPC decoding

- Demodulation
- Decoding the demodulated signal using MSA
- Check node processing
- Variable node processing
- Update APP LLR
  (a posteriori probability, log likelihood ratio)
- Hard decision
- If code word valid then calculate the BER

III. PROPOSED LDPC DECODER USING OPENMP

Parallelizing an application by using OpenMP resources often consists in identifying the most costly loops and, provided that the loop iterations are independent, parallelizes them via the #pragma omp parallel for directive. If we analyze the algorithm, we observe that the check node update can be done in parallel since each row has no correlation with each other. Also, the bit node update on each column can be processed in parallel. The reason we can perform in parallel is that an LDPC decoding algorithm does not have dependencies in memory access among the four types of operations. The MSA algorithm implements the OpenMP for parallelizing the decoder by following steps

1. Include header file <omp.h>
2. Initialization
3. Compute all the messages associated to all CN‟s and BN‟s by parallelizing the code by adding OpenMP Pragma directive and which forks the N threads specified by the environment.
4. After computation, Master thread joins all child threads.
5. Perform decoding operations with N threads specified by the environment using FOR loop.
6. After decoding computation, BER and Frame error rate (FER) are plotted.
7. End.

IV. RESULTS AND DISCUSSIONS

The simulation study has been carried out for block lengths 128, 256, 512 and 1024 (lowest, intermediate and maximum) for constant code rate ½. This section presents the simulations to demonstrate the performance of Min-sum LDPC decoder for the different iterations for the given code length, for different code length. The impact of the selection of number of iterations has been shown in the Fig. 2. It is observed that for the fixed block size as the number of iterations increases the BER decreases. The impact of selection of code length for five iterations has been shown in Fig 3. It may be observed that as the code length increases the performance also improves.

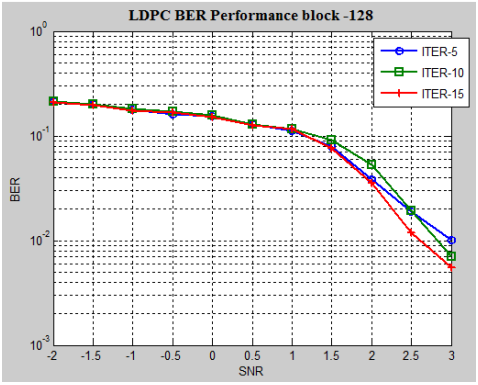


Figure 2.BER performance for block length 128

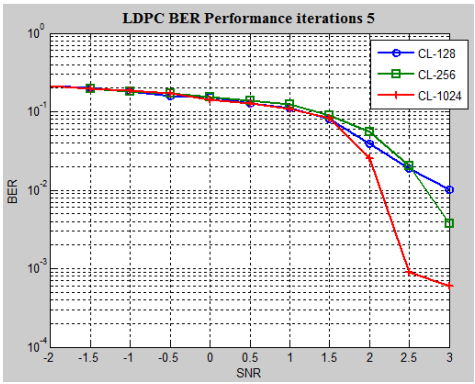


Figure 3.BER performance for 5 iterations

The performance of min-sum LDPC decoder for different code length and for different block sizes is shown in Fig 4. It may be observed that for lower SNRs there is no significant difference for different code lengths as well as for the different iterations. For the SNRs above 2.5 it may be observed that as the code length and iterations increases the performance also improves significantly. The average of all the three different iterations for three different code lengths has been plotted in Fig 5. It may be observed that as the number of iterations increases and as the code length increases there is an improvement in the decoding performance.

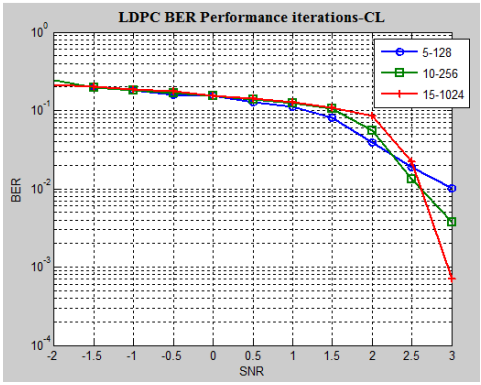


Figure 4.BER performance for different iterations & CL

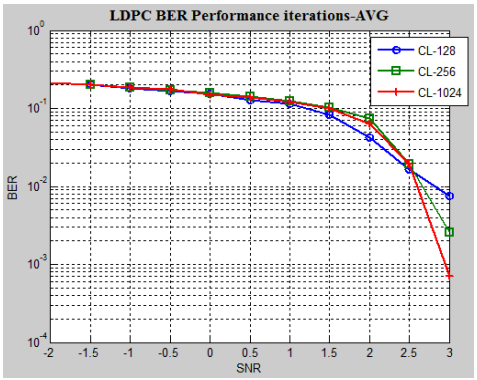


Figure 5.BER performance –average

Fig.6 shows with and without OpenMP, it may be observed that with open amp the BER performance is also good compared to serial processing. With OpenMP parallelism is possible and the time required will be less as shown in Fig.7. As the number of iterations increases the performance also improves. As the number of iterations increases the time required using OpenMp is less, hence the processing and decoding takes less time. Fig. 8 shows the execution time of the MATLAB and OpenMP. It may be observed that using the OpenMP the time taken is very less. Hence parallel processing helps to finish the decoding faster.

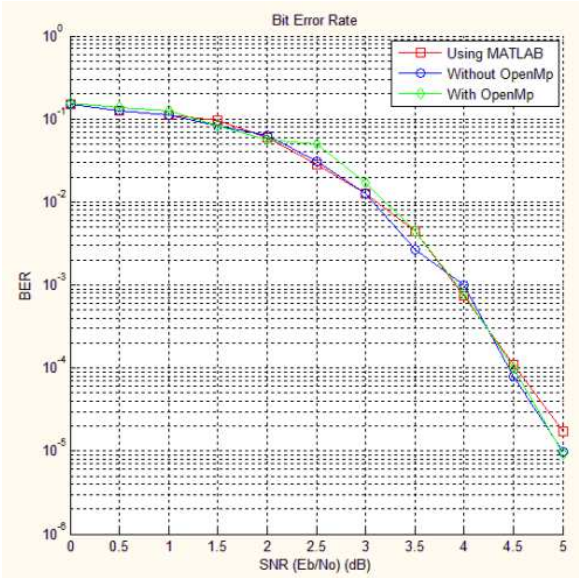


Figure 6.BER versus Eb/No for code length 128 bits and r=1/2B for SNR ranging from 0 dB to 5dB with step size 0.5dB and for iteration= 10

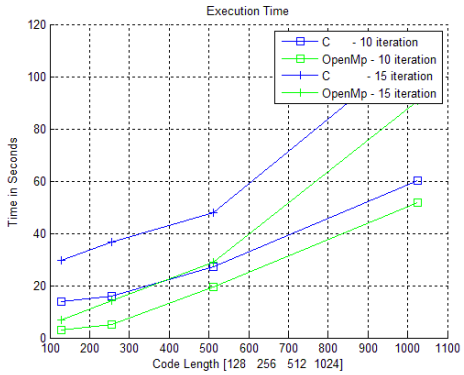


Figure 7. Execution time plot for different code length versus time taken in execution of C and OpenMP platform

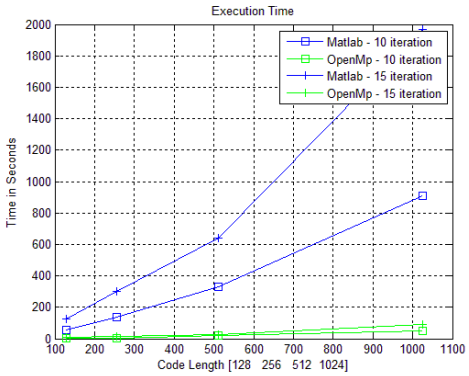


Figure 8. Execution time plots for different code length versus time taken in execution of MATLAB and OpenMP platform

To evaluate the performance of the LDPC decoder, we compared the performance of three cases: (1) where no parallelization technique was applied, (2) where only parallelization utilizing OpenMP was applied, and (3) Where MATLAB technique is employed.

When the iteration count increased with low SNR values, the speedup became greater. This is mainly because of the fact that as the iteration count increases, the amount of check and bit nodes operation will increase. Thus, more parallelization can be done, and accordingly the speedup will also increase. Fig. 7 and Fig. 8 show the execution time plot for different code length executed in different platforms. These Figures give clear information that decoding with OpenMP speedup the execution and increases throughput of data then compared no parallelization is applied.

V. CONCLUSION

Owing to the multiple standards and diverse device function needs of current digital communications, hardware only implementation may not be cost-effective. Instead, software implementation of communication protocols using CPU's or GPU's are rapidly being adopted in digital communication system designs. In this paper we have described a software design that

implements parallel processing of LDPC decoding algorithm. The decoding algorithm is implemented (simulated) using combination of MATLAB, C and OpenMP platform to achieve both flexibility and high performance. Specifically, using of OpenMP for parallelizing software on a multi-core processor. Test results shows that parallel software implementation of LDPC algorithms reduces the execution time thus speeding up of data processing and thereby increasing the throughput of the data.

REFERENCES

[1] R. G. Gallager, "Low-density Parity-check Codes," M.I.T.Press, Cambridge, Massachusetts, 1963.

[2] D. J. MacKay and R. M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," Elect. Lett., vol. 32, pp. 1645–1646, July, 1996.

[3] Chung, S., Forney, G., Richardson, T., and Urbanke, R. (2001), "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit", IEEE Communications Letters, 5(2):58–60.

[4] J. Chen and M.P.C. Fossorier, "Near Optimum Universal Belief Propagation Based Decoding of Low-Density Parity Check Codes," IEEE Trans. Comm., vol. 50, no. 3, pp. 406-414, Mar. 2002.

[5] S. Seo, T. Mudge, Y. Zhu, and C. Chakrabarti, "Design and Analysis of LDPC Decoders for Software Defined Radio," Proc. IEEE Workshop Signal Processing Systems, pp. 210-215, Oct. 2007.

[6] B. Chapman, G. Jost, and R. Van Der Pas, Using OpenMP: Portable Shared Memory Parallel Programming. The MIT Press, 2008.

[7] Akinori Ohhashi and Tomoaki Ohtsuki "Performance Analysis and Code Design of Low-Density Parity-Check (LDPC) Coded Space-Time Transmit Diversity (STTD) System", IEEE Communications Society, Globecom 2004, 0-7803-8794-5/04/$20.00 © 2004 IEEE, Page 3118-3122.

[8] Hao Zhong and Tong Zhang, "Block-LDPC: A Practical LDPC Coding System Design Approach", IEEE Transactions on Circuits and Systems -I: Regular Papers, VOL. 52, NO. 4, APRIL 2005.

[9] Ningde Xie, Tong Zhang, and Erich F. Haratsch, "Improving Burst Error Tolerance of LDPC-Centric Coding Systems in Read Channel", IEEE Transactions on Magnetics, VOL. 46, NO. 3, March 2010.

[10] Alireza Rabbani Abolfazli, Yousef R. Shayan and Glenn E.R. Cowan, "TS-LDPC Analog Decoding Based on the Min-Sum Algorithm", 26th Biennial Symposium on Communications (QBSC) 978-1-4673-1114-4/12/$31.00 ©2012 IEEE.

[11] Chi H. Chan and Francis C. M. Lau, "Parallel decoding of LDPC convolutional codes using OpenMP and GPU", 2012 IEEE Symposium on Computers and Communications (ISCC),978-1-4673-2713-8/12/$31.00 ©2012 IEEE

[12] A. J. FelstrLom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," IEEE Trans. On Inform. Theory, vol. 45, no. 6, pp. 2181-2191, Sep. 1999.

[13] Xin-Ru Lee, Chih-Lung Chen, Hsie-Chia Chang, and Chen-Yi Lee, "Stochastic Decoding for LDPC Convolutional Codes", 2012 IEEE International Symposium on Circuits and Systems (ISCAS) DOI: 10.1109/ISCAS.2012.6271843.