Research Article Impact Factor: 0.621 ISSN: 2319-507X
Raksha Kamat, IJPRET, 2014; Volume 2 (9): 336-344 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

IMPLEMENTATION OF COLUMN LAYERED LDPC DECODER

RAKSHA R. KAMAT, SONIA KUWELKAR

Department of Electronics and Telecommunication Engineering, Goa College of Engineering, Farmagudi, Ponda-Goa.

Accepted Date: 27/02/2014 ; Published Date: 01/05/2014

Abstract: Error correcting codes (ECC) enable the communication systems to have a low-power, reliable transmission over noisy channels. Low Density Parity Check codes are the best known ECC code that can achieve data rates very close to Shannon limit. Layered decoding is well appreciated in Low-Density Parity-Check (LDPC) decoder implementation since it can achieve effectively high decoding throughput with low computation complexity. At first, the Min-Sum algorithm (MSA) is incorporated into the column-layered decoding. Then algorithmic transformations and judicious approximations are explored to minimize the overall computation complexity. Compared to the original column-layered decoding, the new approach can reduce the computation complexity in check node processing for high-rate LDPC codes by up to 90% while maintaining the fast convergence speed of layered decoding. Furthermore, a relaxed pipelining scheme is presented to enable very high clock speed for VLSI implementation. Equipped with these new techniques, efficient decoder architecture is developed.

Keywords: LDPC, MSA, Column-Layered, Decoding, Pipelining

PAPER-QR CODE

Corresponding Author: MS. RAKSHA R. KAMAT

Access Online On:

www.ijpret.com

How to Cite This Article:

Raksha Kamat, IJPRET, 2014; Volume 2 (9): 336-344

INTRODUCTION

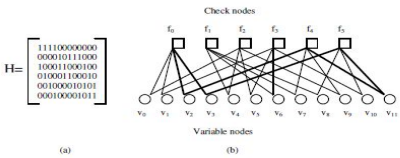
Low-density parity-check (LDPC) codes are a class of linear block codes. They were first introduced by Robert Gallager [1] in his PhD thesis in 1960s. They were largely ignored for a long time because their computational complexity was high for hardware technology at that time. In 1990s, MacKay and Neal [2],[3] rediscovered LDPC codes. A wide array of the latest communication and storage systems have chosen LDPC codes for forward error correction in applications including digital video broadcasting (DVB-S2), 10 Gigabit Ethernet (10GBASE-T) ,broadband wireless access (WiMax), wireless local area network (Wi-Fi),deep-space communications, and magnetic storage in hard disc drives. Currently high performance LDPC decoders are designed to be dedicated blocks within a System-on-chip (SOC).

LDPC Codes

Low Density Parity Check Codes are a class of linear block code defined by a sparse M×N parity check matrix H, [1], where N > M. The parity- check matrix has a small number of '1' entries compared to '0' entries, making it sparse. The number of '1' entries in parity -check matrix row is called the row - weight, k, and number of '1' in the column is the column - weight, j. A regular LDPC code is one in which both row and column weights are constant, otherwise the parity check matrix is irregular.

LDPC Representation

Basically there are two different possibilities to represent LDPC codes. Like all linear block codes they can be described via matrices. The second possibility is a graphical representation. A bipartite graph, also known as Tanner graph, [4], can be used to represent the code. 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 not residing in the same class. The two classes of nodes in a Tanner graph are "Bit Nodes" or "Variable 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 hji in H (parity check matrix) is a one."



LDPC code (a) matrix representation (b) Tanner graph representation.

LDPC Decoding

Conventionally, LDPC codes are decoded using the Sum-Product algorithm (SPA) [1] or the modified Min-Sum algorithm (MSA) [5]. In general, the SPA has the best decoding performance. The MSA is an approximation of the SPA aimed to reduce computation complexity. It is widely employed in LDPC decoder design [6][7]. Both algorithms are based on two-phase message-passing (TPMP) scheme.

Recently, layered LDPC decoding schemes [8][9][10][11] have attracted much attention in both academy and industry because they can effectively speed up the convergence of LDPC decoding and thus reduce the required maximum number of decoding iterations. Presently two kinds of layered decoding approaches, i.e., row-layered decoding [8][9][10] and column-layered decoding [10][11] have been proposed. In row-layered decoding, the parity check matrix of the LDPC code is partitioned into multiple row layers. The message updating is performed row layer by row layer. The column-layered decoding employs the similar idea except that the parity check matrix is partitioned into multiple column layers. The message computation is performed column layer by column layer.

The SPA-based column-layered decoding approach [11] was proposed in 2005. In this work, we investigate and explore the benefits of the column-layered decoding approach. We first incorporate the Min-Sum algorithm into the column-layered message passing scheme because it has much lower complexity than the SPA. In addition, by deeply investigating the message updating process in the Min-Sum based column-layered decoding, we develop a simplified column-layered decoding scheme, which maximally eliminates the redundant computations in the original scheme and significantly reduces the computation complexity. For high-speed VLSI implementation, the proposed design has significant advantages over the conventional row-layered decoding. First, the column-layered decoding inherently has shorter critical path. For a row-layered decoder, the messages associated to multiple sub-blocks in a row layer can be processed in one cycle to increase throughput. However, it will increase the complexity of check node unit (CNU) and require serial concatenation of multiple comparison and selection stages in VLSI implementation. It has been reported that significant hardware overhead is required to optimize the corresponding circuitry for high clock speed. In the column layered decoding, the major implementation complexity is associated with variable node unit (VNU), particularly when the messages corresponding to multiple sub-blocks in a column layer are processed in parallel. Because only addition operations are performed in a VNU, it is very convenient to employ arithmetic optimization to minimize the critical path. Moreover, the intrinsic message loading latency is minimized in the column-layered decoding because the

decoding can start as soon as the intrinsic messages corresponding to one block column are available. In summary, the proposed design is well suited for very high decoding throughput and low power LDPC decoder implementation.

Min-Sum Algorithm Based Column-layered Decoding

Let *C* be a binary (N, K) LDPC code specified by a parity-check matrix ***H*** with *M* rows and *N* columns. Each row of the parity check matrix is associated with a check node, and each column is associated with a variable node. Let $N(c) = \{v : Hcv = 1\}$ denote the set of variable nodes that participate in check node *c*, and $M(v) = \{c : Hcv = 1\}$ denote the set of check nodes associated to variable node *v.* Let *Iv* denote the intrinsic message for variable node v and *Rcv* represent the check-to-variable message conveyed from check node *c* to variable node *v*, and *Lcv* represent the variable-to-check message conveyed from variable node *v* to check node *c*. Assume that the *N* bits of a codeword are divided into *G* groups of the same size, N_0, N_1, N_{G-1} . Accordingly, the parity-check matrix is divided into *G* block columns. The proposed column-layered decoding based on the Min-Sum algorithm is described with the pseudo code below.

Min-Sum-based column-layered decoding algorithm

Initialization:

Lcv =I v for *v=0, 1… N-1*, *c=0, 1, …, M-1*;

Iterative decoding:

For *iter* = 1, 2, …, maximum iteration number

{

For *g*=0, 1, …, *G*-1

{

Horizontal Step: For each check node *c* that is connected to variable node $v \in Ng$, computes

$$R_{cv} = \prod_{n \in N(c)\setminus v} \operatorname{sgn}(L_{cn}) \times \min_{n \in N(c)\setminus v} |L_{cn}|. \quad (1)$$

Vertical Step: For each variable node $v \in Ng$, updates *Lcv* and *Lv* as follows:

$$L_{cv} = I_v + \alpha \times \sum_{m \in M(v) \backslash c} R_{mv} , \qquad (2)$$

$$L_v = I_v + \alpha \times \sum_{m \in M(v)} R_{mv} . \qquad (3)$$

Hard decision and termination:

Make hard decision by using the sign of *Lv* ; Terminate the decoding if a valid codeword is found.

Low Complexity Decoding Scheme

Algorithm Reformulation

A significant amount of redundant computation is performed in the original column-layered decoding algorithm. To improve the computation efficiency, the computation in (1) for consecutive column layers can be incrementally performed. In LDPC decoding, each variable node sends a variable-to-check message to every neighboring check node. Assume that a check node *c* has *dc* variable node neighbors, and hence receives *dc* soft messages from its neighboring variable nodes.

Let $m_c^{(g)} = [m_1 \ m_2 \ ... m_{dc}]$ be a sorted vector of the magnitudes of the soft messages received by check node *c* in ascending order. The superscript *g* indicates the soft message was generated when the *gth* block column is processed. Similarly, let $Sc^{(g)}$ be

$$\left| \prod_{n \in N(c)} sign(L_{cn}) \right|$$

when the decoding for the layer *g* is completed. To reduce the computation complexity of Min-Sum based column-layered decoding, $\mathbf{m}_c^{(g)}$ can be computed from $\mathbf{m}_c^{(g-1)}$ in three steps.

1. For each variable node *v* in N_g, remove the old $| L_{CV} |$ from $\mathbf{m}_c^{(g-1)}$ *to* obtain a temporary sorted Vector ~ **m***c* (*g*). In addition, remove the old *sign* (*Lcv)* from $Sc^{(g-1)}$ and obtain a temporary sign-product ~ $Sc^{(g)}$. Since the smallest value, m_1, in ~$\mathbf{m}_c^{(g)}$ is

$$\min_{n \in N(c) \backslash v} (| L_{cn} |)$$

and ~ $Sc^{(g)}$ is

$$\prod_{n \in N(c) \setminus v} sign(L_{cn})$$

The value of *Rcv* can be computed as ~ $Sc^{(g)} \times m_1$. Send the *Rcv* message to the variable node *v*.

2. Perform variable-to-check message computations for all variable nodes belonging to N_g. The new *Lcv* messages are sent back to corresponding check nodes.

3. For each check node, insert the updated | L_{CV} | into ~ **m***c* (*g*) in a sorted order to obtain $\mathbf{m}_c^{(g)}$.

The reformulated column-layered decoding procedure is summarized as follows:

The Reformulated Column-layered Decoding

Initialization:

Let *Lcv* =*I v* for all variable nodes. For each check node, sort the magnitudes of the *Lcv* messages from its neighboring variable nodes. Compute the sign product for each check node

$$S_c = \prod_{n \in N(c)} sign(L_{cn})$$

Iterative decoding:

For *iter* = 1, 2, ..., maximum iteration number

{

For g=0, 1, ..., *G-1*

{

***Horizontal Step-A*:** for each check node *c* that connects to variable node $v \epsilon N^{(g)}$,compute ~ $\mathbf{m}c^{(g)}$ by removing the old from $\mathbf{m}_c^{(g-1)}$, (4)

And

$RCV^{(g)} = \sim Sc^{(g)} \times m_1$ (5)

Vertical Step: For each variable node $v \epsilon Ng$, compute $L_{CV}^{(g)}$ and $L_V^{(g)}$ using (2) and (3).

Horizontal Step-B: for each check node *c* that connects to variable node $v \epsilon N$ (*g*) ,

compute $\mathbf{m}c^{(g)}$ using ~ $\mathbf{m}c^{(g)}$ and | $L_{CV}^{(g)}$ | , (6)

and

$Sc^{(g)}=\sim Sc^{(g)}\times sign(Lcv^{(g)})$
(7)

}

Hard decision and termination:

Make hard decision by using the sign of *Lv* ; Terminate decoding if a valid codeword is found or the maximum decoding iteration is reached.

The Pipelined Column-layered Decoding

Pipelining is a common practice in VLSI implementation to increase clock speed and thus to speed up data processing throughput. In the original column-layered decoding, the change between $\mathbf{m}c^{(g-1)}$ and $\mathbf{m}c^{(g-1+P)}$ is very small if the value of *P* is not large. Thus, $\mathbf{m}c^{(g-1)}$ can be used as the estimation of $\mathbf{m}c^{(g-1+P)}$. An approximation of $Rcv^{(g+P)}$ can be calculated from $\mathbf{m}c^{(g-1)}$ before $\mathbf{m}c^{(g-1+P)}$ are obtained. Then, it is immediately used for computing $Lcv^{(g+P)}$. The approximation allows *P* clock cycles to complete the message computation for each layer. When | $Lcv^{(g+P)}$ | is obtained, the $\mathbf{m}c^{(g-1+P)}$ is already available. Thus, the horizontal step for layer *g+P* can be undertaken with $\mathbf{m}c^{(g-1+P)}$ and | $Lcv^{(g+P)}$ | . The updating method of $\mathbf{m}c^{(g+P)}$) is the same as before.

Proposed column layered decoder architecture

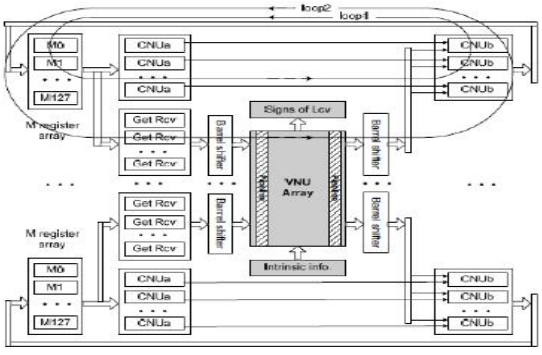


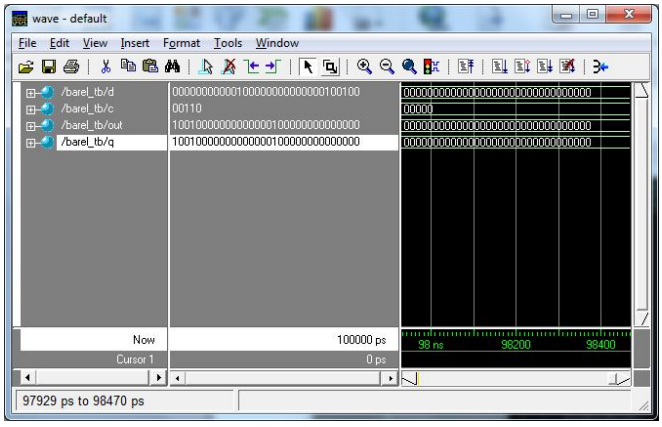
Figure 3.1: pipelined column-layered decoding

Each M component in an M-register array on the left represents a 25-bit register for storing a sorted vector associated with a row of the H matrix. The Get Rcv component gets check-to-variable messages for layer g+P, where P=2. The CNUa component (the portion of check node unit for horizontal Step-A) removes the old variable-to-check message associated to layer g from the sorted vector. The total numbers of M-register, Get Rcv, CNUa, and CNUb are all 512. The number of variable node unit (VNU) in the VNU array is 128. A VNU is required to perform the vertical step (2) and (3) for a variable node. The barrel shifters in the left side of the VNU array align the check-to-variable message from row order to column order. Similarly, the barrel shifters in the right side of VNU array align the variable-to-check message from column order to row order. Two types of loops are formed in column-layered decoding. The first type of loop consists of a CNUa and a CNUb components. The second type of loop is composed of a Get Rcv, a CNUb, a VNU, and two barrel shifters. With the pipelined column-layered decoding approach, the critical path in the second type loop is drastically reduced when applying two-stage pipelining.

Implementation

Based on the architectures proposed before, the column layered decoder will be modeled in Verilog hardware description language and simulated using Xilinx ISE 10.1 software package. It will be implemented on Xilinx Virtex5 FPGA.

Simulation results



Barrel shifter simulation results

REFERENCES

1. R. G. Gallager, "Low-density parity-check codes," IRE Trans. Inform. Theory, vol. IT-8, pp. 21-28, Jan. 1962.

2. D.J.C. MacKay and R.M. Neal. Near Shannon Limit Performace of Low Density Parity Check codes. In Electronics Letters, volume 32,pages 16451646, Aug 1996.

3. D.J.C. MacKay. Good Error-Correcting Codes Based on Very Sparse Matrices. IEEE Transaction on Information Theory, 45(2):399431,Mar 1999.

4. R.M. Tanner. A recursive approach to low complexity codes. IEEE Transactions on Information Theory, 27(5): 533547, Sep 1981.

5. J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, X. –Y. Hu, "Reduced-Complexity Decoding of LDPC Codes," IEEE Transactions on Communications, vol. 53, issue 8, pp. 1288 – 1299, Aug. 2005.

6. Z. Wang and Z. Cui, "A memory efficient partially parallel decoder architecture for QC-LDPC codes," in Proc. of the 39th Asilomar Conference on Signals, Systems & Computers, pp. 729-733, 2005.

7. C. Lin, K. Lin, H. Chan, and C. Lee, "A 3.33 Gb/s (1200, 720) low-density parity check code decoder," in Proc.31st Eur. Solid-State Circuits Conf., pp. 211–214, Sep. 2005.

8. E. Sharon, S. Litsyn, and J. Goldberger, "An efficient message-passing schedule for LDPC decoding," in Proc. ofthe 23rd IEEE Convention of Electrical and Electronics Engineers in Israel, pp. 223-226, Sept., 2004.

9. D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," IEEE Workshop on Signal Processing Systems, pp. 107 - 112, 2004

10. P. Radosavljevic, A. Baynast, and J. R. Cavallaro, "Optimized Message Passing Schedules for LDPC Decoding," in Proc. of 39th Asilomar Conference on Signals, Systems and Computers, pp. 591 – 595, 2005.

11. J. Zhang, M.P.C. Fossorier, "Shuffled iterative decoding," IEEE Transactions on Communications, vol. 53, Issue 2, pp. 209 – 213, Feb. 2005.