IPSJ Transactions on System LSI Design Methodology Vol. 2 122–130 (Feb. 2009)

Regular Paper

A High Throughput LDPC Decoder Design Based on Novel Delta-value Message-passing Schedule

Wen Ji,[†1] Xing Li,[†1,*1] Takeshi Ikenaga[†1] and Satoshi Goto[†1]

In this paper, we propose a partially-parallel irregular LDPC decoder for IEEE 802.11n standard targeting high throughput applications. The proposed decoder has several merits: **(i)** The decoder is designed based on a novel delta-value based message passing algorithm which facilitates the decoding throughput by redundant computation removal. **(ii)** Techniques such as binary sorting, parallel column operation, high performance pipelining are used to further speed up the message-passing procedure. The synthesis result in TSMC 0.18 CMOS technology demonstrates that for (648,324) irregular LDPC code, our decoder can achieve 8 times increasement in throughput, reaching 418 Mbps at the frequency of 200 MHz.

1. Introduction

Low-Density Parity-Check (LDPC) code is an error correcting code originally proposed by Gallager in 1963 [1], and rediscovered by Mackay and Neal [2] in 1996. After that irregular LDPC codes are constructed enabling data transmission rates close to the Shannon Limit [3)–5)]. On the other hand, the LDPC decoding algorithm is inherently parallel and is much easier to be implemented than its comparator turbo codes, thus making it more attractive to researchers [5]. As a result, they have been recently adopted for the 10GBase-T [6], the DVB-S2 [7], and the Mobile WiMAX standards [8].

LDPC code is very suitable for hardware implementation by utilizing a parallel decoding algorithm called Message-Passing (MP) algorithm [1]. In the last few years, researches have been done on designing specific decoder architectures for LDPC implementations, seeking for the best trade-off between area, power consumption and performance [9)–14)]. The LDPC decoders that have been designed share the same primitive processing elements: a check functional unit (CFU) performing row operations for check nodes and bit functional unit (BFU) performing column operations for bit nodes. These processing elements are connected according to the Tanner graph, a graph representing the relation between bit nodes and check nodes. The MP algorithm exchanges messages between check nodes and bit nodes by performing row and column operations iteratively.

The authors in Ref. 13) proposed an accelerated message-passing schedule, which only performs those column operations whose corresponding check nodes have been updated by the row operations, which is demonstrated to be more efficient than the basic algorithm. A partially-parallel LDPC decoder based on the accelerated MP schedule is introduced in Ref. 14) to support for irregular LDPC code. Although the partially-parallel irregular LDPC decoder proposed in Ref. 14) can improve the error correction performance, it suffers a main problem that the overall throughput is relatively low compared with the regular or fully-parallel irregular LDPC decoder (eg., Refs. 15), 16)). Moreover, the throughput of current partially-parallel irregular LDPC decoders designed, such as 54 Mbps of Ref. 14), is not sufficient for most applications of 802.11n stardard [17], which requires a throughput of upto 330 Mbps. From this observation, there is room for the improvement for partially-parallel irregular LDPC decoders. In this work, we propose an improved MP algorithm and decoder architecture for irregular LDPC decoder, in this paper, to achieve a nearly 8X speedup with almost the same BER performance.

The novelties of this decoder in terms of high throughput are as follows:

- The proposed LDPC decoder is based on a novel delta-value message-passing algorithm suitable for high throughput design.
- An improved binary sorting scheme is designed in row operation to reduce the computation time.
- A parallel structure of bit function unit is designed to speed up the column operation.
- A high performance pipeline structure is used to further speed up the message-passing procedure.

†1 Graduate School of Information, Production and Systems, Waseda University
*1 Presently with NEC Electronics Corporation

The rest of the paper is organized as follows. Section 2 introduces the preliminary concepts and background materials. Section 3 explains the problem of the existing message-passing algorithm and the motivation of this work. Section 4 discusses the improved message-passing algorithm proposed. Section 5 provides the strategies for high throughput design used in our LDPC decoder. Section 6 shows the detailed implementation and the synthesis results and finally Sect. 7 concludes.

2. Decoding of LDPC Codes

In this section, we present the preliminary concepts, specifically, irregular LDPC code, and message-passing algorithm. The irregular LDPC codes can be defined by a parity check matrix H containing $M \times N$ sub-blocks, where M and N are the number of row-blocks and column-blocks respectively. Each sub-block matrix is a $b \times b$ square matrix, obtained through right shifting the identity matrix $I_{b\times b}$ by a specific number. The parity check matrix used in this work is a 12×24 matrix (each sub-block is 27 bits $\times$ 27 bits) defined in IEEE 802.11n standard [17] with code rate of 1/2 and block length of 648 bits, as shown in **Fig. 1** (a), where the virgule lines in the sub-blocks mean the positions where exit "1"s after shifting according to the shift value provided by the parity check matrix.

LDPC code can be decoded iteratively using the MP algorithm described in Ref. 1). Each iteration of the algorithm is composed of four phases: row operation, column operation, error correction and parity check. The row operation updates message α of all check nodes using message β, and sends the message to bit nodes. The column operation updates message β of all bit nodes based on message α and initial message λ. The error correction calculates the tentative decision and the parity check operation decides whether another decoding iteration is necessary or not according to the tentative decision.

The calculations of α and β value in row and column operation are shown in Eq. (1) and Eq. (2), where $\alpha(m,n)$ and $\beta(m,n)$ denote the check node and bit node message in m^{th} row and n^{th} column of the parity check matrix H, respectively. $\lambda(n)$ denotes the initial message of n^{th} column. And $A(m)$ and $B(n)$ are defined as $A(m) = \{n|H_{mn} = 1\}$ and $B(n) = \{m|H_{mn} = 1\}$ respectively.

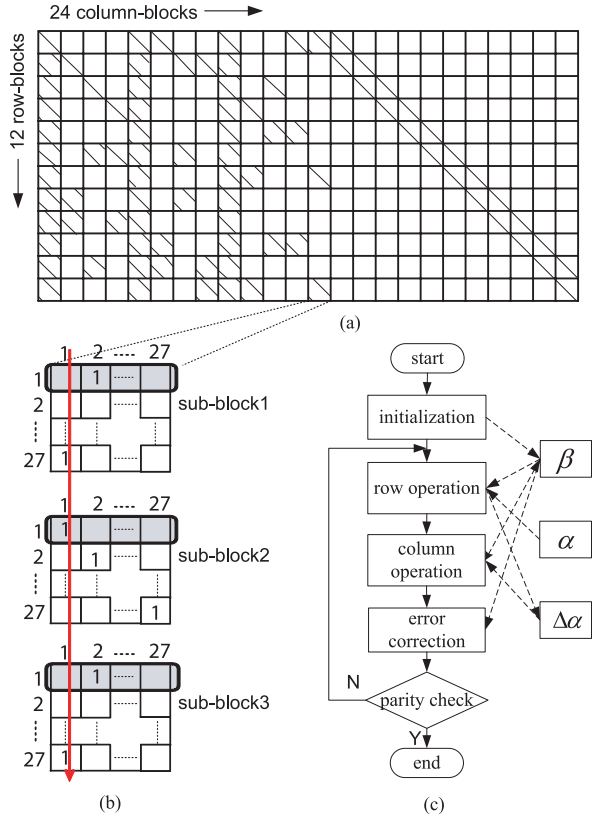

Fig. 1 Delta-value based message-passing algorithm: (a) targeting parity check matrix (24×12 sub-blocks, each is 27 bits×27 bits) (b) 13^{th} column-block, and (c) algorithm flowchart.

$$\alpha(m,n) = \prod_{n' \in A(m)\backslash n} \mathrm{sign}(\beta(m,n')) \times \min_{n' \in A(m)\backslash n} |\beta(m,n')| \quad (1)$$

$$\beta(m,n) = \lambda(n) + \sum_{m' \in B(n)\backslash m} \alpha(m',n) \quad (2)$$

In the accelerated MP schedule proposed in Refs. 13), 14), column operations are performed right after a row operation only for those bit nodes, which are in

connection with the updated check nodes. This schedule is demonstrated to be more efficient than the basic algorithm [13].

3. Motivation

In this section, we use an example to demonstrate the problem of the accelerated message-passing algorithm.

The accelerated message passing schedule, proposed in Ref. 13), is well suitable for irregular LDPC decoder design. The idea is illustrated through an example demonstrated in Fig. 1 (b). This is a column block extracted from the 13^{th} column-block of the targeting parity check matrix,with sub-block1 and sub-block3 right shifting one position, and sub-block2 remaining at the identical position. After the row operations for all the first rows in each sub-block are executed, three α values are updated, which are $\alpha_1(1,2)$, $\alpha_2(1,1)$ and $\alpha_3(1,2)$ respectively. In the accelerated MP algorithm, the calculation of β values will be executed right after the first row operation. The computations of β values at the first column are shown as follows:

$\beta'_1(27,1) = \lambda(1) + \alpha_2(1,1) + \alpha_3(27,1)$
$\beta'_2(1,1) = \lambda(1) + \alpha_1(27,1) + \alpha_3(27,1)$
$\beta'_3(27,1) = \lambda(1) + \alpha_1(27,1) + \alpha_2(1,1)$

If we consider these column operations that calculate the updated message β, only a small part of the operands have been changed since last computation of the same message β. For example, only $\alpha_2(1,1)$ is changed for calculation of $\beta_1(27,1)$ and $\beta_3(27,1)$ and no operand is changed for calculation of $\beta_2(1,1)$. A considerable part of addition operations in column operation, in fact are a repetition of former computations. And for the targeting parity check matrix, when $|B(n)|$ is large, this problem becomes more significant as more useless additions are operated. The above observation demonstrates that accelerated MP scheme still has computational redundancies, which degrade the efficiency of hardware implementation.

4. Delta-value Based Message Passing Schedule

In accelerated MP algorithm, the computation of β for columns with a weight of 12 in the targeted parity check matrix ($|B(n)| = 12$), requires the addition of 11 β values and one λ value according to Eq. (2) [13),14)]. This results in a five-depth addition and thus becomes the bottleneck to get high throughput designs. However, we propose an improved MP schedule called Delta-Value based Message-Passing (DVMP) schedule to help solve this bottleneck problem and thus increases the throughput.

We first introduce the concept of a new message $\Delta\alpha$, as follows.

$$\Delta\alpha(m,n) = \alpha'(m,n) - \alpha(m,n) \tag{3}$$

Back to example in Fig. 1 (b), the calculation can be simplified by adding only the updated delta-value of message α shown as follows.

$\beta'_1(27,1) = \beta_1(27,1) + \Delta\alpha_2(1,1)$
$\beta'_2(1,1) = \beta_2(1,1)$
$\beta'_3(27,1) = \beta_3(27,1) + \Delta\alpha_2(1,1)$

This new calculation method of message β can remove redundant computations and reduce the total number of additions in the bottleneck column operation, especially when $|B(n)|$ becomes larger. For the targeted parity check matrix [17)], at most two message α are updated after certain row operation. Based on this observation, only a small number of updated α value is sufficient to generate a correct message β. Therefore, we proposed our DVMP algorithm by only calculating the delta value of updated α in the row operation to improve the decoding efficiency. In such scheme, the resulting β can be obtained by at most 2 levels of addition in DVMP rather than 5 levels in the accelerated MP scheme [13),14)].

The proposed DVMP algorithm is shown in Fig. 1 (c), with the solid line showing the flowchart of the algorithm stages and the dotted line showing the data transmission of the storages. In the initialization step, message β is initialized as message λ. Then the row operation is executed to update message α according to Eq. (1) and generate corresponding $\Delta\alpha$ according to Eq. (3) at the meantime. Next, the column operation calculates message β by the updated $\Delta\alpha$ values as Eq. (4). The error correction calculates the tentative decision, based on which a parity check is done after each iteration to determine the termination of the decoding process.

$$\beta'(m,n) = \beta'(m,n) + \sum_{m' \in D(n) \setminus m} \Delta\alpha(m',n) \tag{4}$$

In Eq. (4), D(n) are those row numbers of which the message α has been updated since the last computation of $\alpha(m, n)$ in column n. The total number of possible addition is reduced from $|B(n)|$ to $|D(n)|$. And in the situation of the targeted irregular decoding matrix, this redundancy removal can improve the column operation by a saving of three level additions, which reduce both the computation time and area of hardware implementation.

5. High Throughput Design

Apart from the high throughput DVMP schedule, we integrate three different strategies into the proposed decoder to further speed up the decoding process. In row operation module, a binary sort scheme is used to shorten the comparison time. In column operation module, parallel DVMP schedule is used to save the processing time. Furthermore, a pipeline structure is utilized to speed up the whole processing procedure.

5.1 High-throughput Row Operation Module

In the calculation of α value according to Eq. (1), the minimum and second minimum β value among a total of $|A(m)|$ values in the same row should be obtained through a proper comparison scheme. In previous designs [13),14)], simple sorting algorithms are used to obtain the minimum values, such as the implementation of bubble sort comparison scheme. Two registers are used to store the minimum value and second minimum value respectively. Each time a new α is read in to compare with the current minimum one and replace it if the new one is smaller. These serial comparison steps, however, requires considerable computational time and becomes the most time consuming part in row operation when the number of message β increases greatly in the targeted matrix.

In order to improve the hardware performance, a tree structure is proposed, together with a binary sort to obtain two minimum values in parallel. The modified sorting scheme requires less area and can generate the result in only five comparison steps.

The detailed comparison procedure in row operation module is shown in **Fig. 2**. The left tree and the right tree represent respectively the sorting for the minimum and the second minimum β value among a total of eight values. In the figure, we assume that the values of eight β at the same row (β_1, β_2, ..., β_8) comply with

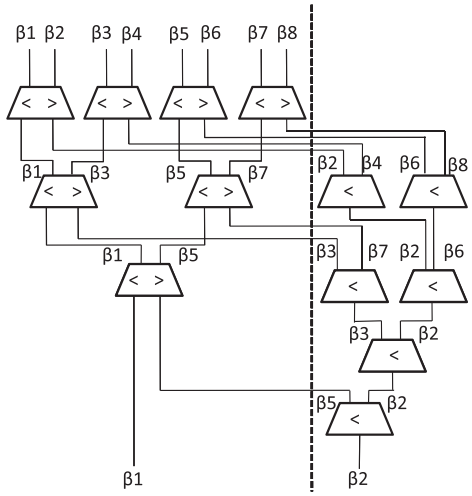

Fig. 2 Comparison schedule in row operation module.

the relationship of $\beta_1 < \beta_2 < \ldots < \beta_7 < \beta_8$ and the resulting values are labelled on each data path. In the first step, eight β values are compared in pairs. The smaller one (e.g., β_1) is remained in the sorting tree for the minimum value while the larger one (e.g., β_2) is eliminated to the sorting tree for the second minimum value. Then the values in both trees will continue the comparison process in pairs with the larger one in the left tree eliminated to the right tree. In this manner, we can get the minimum and the second minimum value at the third and the fifth step of the operation. Compared with a total 9 clock cycles in Ref. 14), our proposed scheme can complete the comparison in only 2 clock cycles under the frequency of 200 MHz, a 4.5X speedup.

5.2 High-throughput DVMP-based Column Operation Module

As discussed in Sect. 4, the DVMP-based column operation can update β values by the addition of its original β and at most two updated $\Delta\alpha$ values according to Eq. (4). The number of updated $\Delta\alpha$ values and the exact position of each updated $\Delta\alpha$ in the column operation are determined by the parity check matrix. Therefore, Equation (4) is simply achieved by two addition steps during implementation. In the first addition step, at most two updated $\Delta\alpha$ values is added

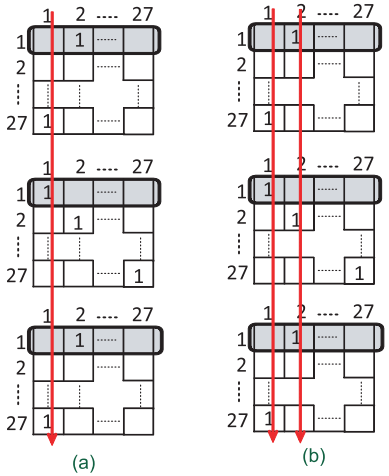

Fig. 3 Example of column operation: (a) column after column scheme (b) parallel scheme.

together. Then we compute the new β value by adding the original one and the sum of $\Delta\alpha$ in the second step. By adapting the DVMP schedule, not only the computation time of each column operation is minimized, but also the area of hardware implementation is reduced. Based on this observation, we can further improve the throughput by applying a parallel column operation here.

In previous implementation of the accelerated MP schedule [13),14)], all N column-blocks are computed in parallel while different columns in one column-block are calculated in serial as shown in **Fig. 3** (a). After the first row operation is calculated, which is shown as gray rectangulars, column operations are calculated one by one. This processing procedure requires at most 11 clock cycles for column computation in the real implementation of the targeting LDPC code [14)]. In the proposed module, message β of all columns, in which there exists an updated α, are computed in parallel, as shown in Fig. 3 (b). A hardware implementation for computing k columns in parallel is shown in **Fig. 4**.

The resulting parallel DVMP-based column operation can be computed in one clock cycle, which is 11 times faster than the implementation in Ref. 14).

5.3 High-throughput Pipeline Schedule

In the proposed decoder, pipeline structure is utilized to achieve further speed

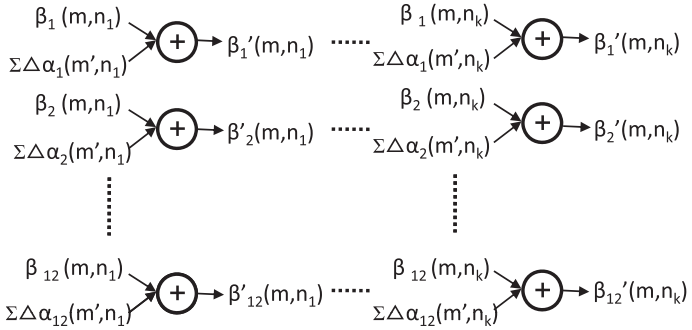

Fig. 4 Architecture of column operation module.

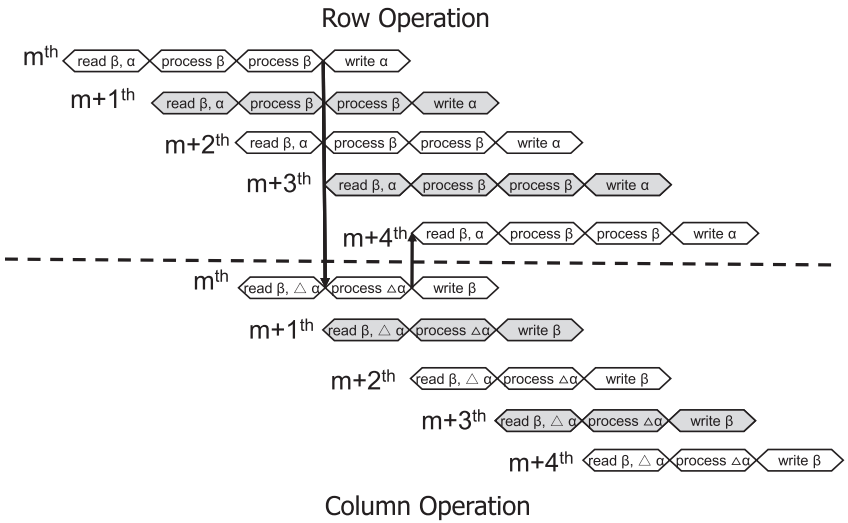

Fig. 5 Timing schedule of the decoder.

up of the procedure, as shown in **Fig. 5**. The row operation and column operation (including message read and write) are divided into four and three pipeline stages respectively to balance the computation time of each stage. After a further overlap of the row and column operation based on data dependency information, the message of a particular row operation and corresponding column operation

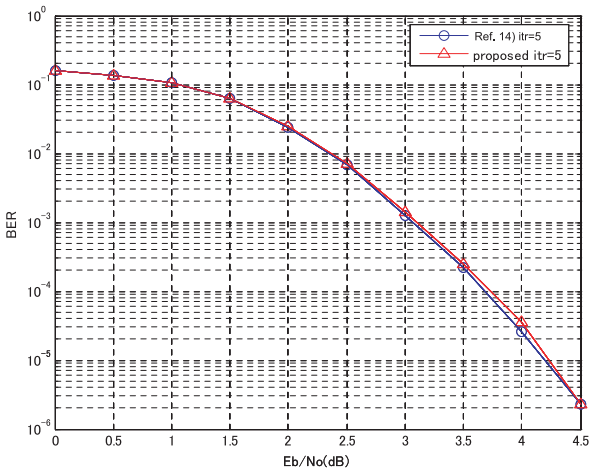

Fig. 6 Bit error rate performance.

can be updated after four clock cycle.

During the first clock cycle, α and β values are read in to prepare for a row operation. In the second and third clock cycles, a tree structure comparison is conducted for the minimum and second minimum β value, and corresponding α and $\Delta\alpha$ values are obtained. In the fourth clock cycle, all related column operations are computed based on $\Delta\alpha$ values, and α values are written into memories at meantime.

The BER performance is simulated by calculating the average bit error rate of 2,000 randomly generated input values and C++ program is used to simulate the hardware process. Based on this proposed pipeline structure, the updated message of a specific row can be used after four clock cycles, in other words, the updated messages of m^{th} row can be used by $(m+4)^{th}$ row operation. Although in Ref. 14), the updated message of the same row is used by $(m+3)^{th}$ row operation, our pipeline structure incurs nearly no performance degrading compared to Ref. 14) under the same decoding iterations as illustrated in **Fig. 6**. Further more, the proposed pipeline structure is more compact, which improves both the hardware utilization and throughput.

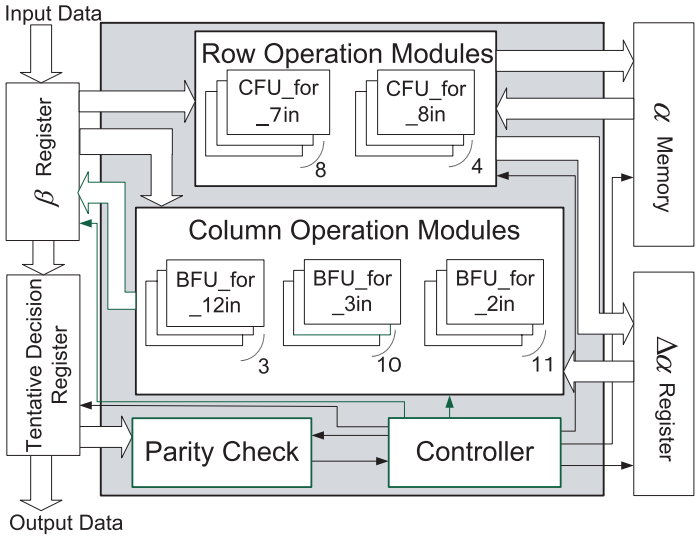

Fig. 7 Block diagram of the decoder.

6. Implementation and Results

In this section, we present the hardware implementation of the proposed partially-parallel irregular LDPC decoder and the synthesis results.

6.1 Block Diagram

The proposed decoder is mainly composed of five parts: row operation modules, column operation modules, parity check, controller and storage parts (memory for α value, registers for β value, $\Delta\alpha$ value and tentative decision value), as shown in **Fig. 7**.

In the proposed decoder, we design 12 CFUs and 24 BFUs in order to execute the computation of the same row or column in each sub-block in parallel. Since the targeted LDPC decoding matrix is irregular, two different row operation modules (CFU_for_7in and CFU_for_8in), and three different column operation modules (BFU_for_12in, BFU_for_3in, and BFU_for_2in) are designed for rows and columns with different $|A(m)|$ and $|B(n)|$. The controller module generates the control signals for storage and operation modules, while parity check

module does the error correction and the parity check at the end of each iteration. Message α is only used in the row operation and is stored in memory with corresponding row address. Other messages like β, which have multiple-access problem, are stored in registers.

The detailed composition of the proposed decoder core is listed in **Table 1**, under the column with DVMP. The result is compared to the implementation without DVMP, and we can see from the table, the computation-intensive column operation modules are reduced because of the use of DVMP schedule. And the total number of gates of the decoder core is reduced.

6.2 Implementation result

The proposed decoder is implemented under TSMC 0.18 um CMOS technology. The synthesis results are listed in **Table 2**. The power consumption of the proposed decoder is simulated using Synopsys Power Compiler under the worst case. Reference 10) is a partially-parallel irregular decoder whose code length is 8088 bit. Reference 9) is a fully-parallel decoder targeting regular codes. And we compare our work with Refs. 13), 14) under the same LDPC code and the same design rule. Under column Ref. 14) is the synthesis result of a partially-parallel irregular LDPC decoder designed in Ref. 14), which is the only design known, targeting the same newly proposed LDPC code in Ref. 17). We also modified the design in Ref. 13) to support the same irregular code with detailed synthesis result under column Ref. 13)*.

Because of the pipeline structure along with the improved row and column operation modules, our proposed decoder requires only 31 clock cycles for a single iteration and five iterations for codeword correction under SNR of 3.0 dB, which can achieve 8 times throughput than Refs. 13), 14). Although it consumes more power, it is about one-fifth of the area of Ref. 9) according to the gate numbers, and achieves almost the same throughput. Our synthesis result shows that the design of partially-parallel approach can also overcome the low throughput problem.

Table 1 Composition of the decoder core.

Module	with DVMP		without DVMP	
	Number of Gates	Percentage (%)	Number of Gates	Percentage (%)
Row Operation Modules	53042	53.00	49902	41.17
Column Operation Modules	36619	36.59	60903	50.24
Parity Check	9285	9.28	9285	7.66
Controller	1124	1.13	1124	0.93
LDPC decoder core	100070	100	121214	100

7. Conclusion

In this paper, a novel high throughput partially-parallel irregular LDPC decoder is proposed. Row operations and column operations are speeded up by a

Table 2 Implementation result.

	Ref. 10)	Ref. 9)	Ref. 13)*	Ref. 14)	**Proposed**
Design rule	TI 0.11 μm	0.16 μm	TSMC 0.18 μm		
LDPC Code	8,088 bit rate 1/2 irregular	1,024 bit rate 1/2 regular	802.11n 648 bit rate 1/2 irregular		
LDPC decoder	partially-parallel	fully-parallel	partially-parallel		
Throughput	188 Mbps (itr=25, SNR=NA)	1 Gbps×1/2 (itr=64, SNR=3.0 dB)	54 Mbps	54 Mbps (itr=5, SNR=3.0 dB)	**418 Mbps**
Frequency	212 MHz	64 MHz	200 M	200 M	**200 M**
Memory area	407 K(gates)	N/A	708 K(gates)	502 K(gates)	**170 K(gates)**
Area w/o wiring	742 K(gates)	1750 K(gates)	832 K(gates)	611 K(gates)	**423 K(gates)**
Area w/. wiring	N/A	52,500,000(μm^2) (fabricated)	13,090,549(μm^2)	9,004,366(μm^2) (synthesis)	**12,930,443(μm^2)**
Power (mW)	N/A	690 (@64 MHz, 1.5 V)	765.85	486.44 (@200 MHz, 1.6 V)	**893.18**

modified binary searching scheme and delta-value based message-passing schedule respectively. Moreover a pipeline structure is utilized to further compact the procedure. The synthesis result demonstrates that our decoder can achieve a much higher throughput and almost the same bit error rate performance compared to other partially-parallel irregular LDPC decoders. It also demonstrates that for partially-parallel structure, throughput can be improved close to fully-parallel structure with less area cost.

Acknowledgments This research is supported by CREST, JST and a grant of Knowledge Cluster Initiative, MEXT.

References

1) Gallager, R.G.: *Low-Density Parity-Check Codes*, MIT Press, Cambridge, MA (1963).
2) MacKay, D.J.C.: Good error-correcting codes based on very sparse matrices, *IEEE Trans. Inform. Theory*, Vol.45, No.2, pp.399–431 (2001).
3) Shannon, C.: A mathematical theory of communication, *Bell Syst. Tech. J.*, Vol.27, pp.379–423, 623–656 (1948).
4) Richardson, T.J., Sholrollahi, M.A. and Urbanke, R.L.: Design of capacity-approaching low-density parity-check codes, *IEEE Trans. Inform. Theory*, Vol.47, No.2, pp.619–637 (2001).
5) Chung, S.Y., Forney, G.D., Richardson, T.J. and Urbanke, R.L.: On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit, *IEEE Communication Letters*, Vol.5, No.2, pp.58–60 (2001).
6) *World Wide Web*, *IEEE 802.3an Task Force*, http://www.ieee802.org/3/an/index.html (2004).
7) ETSI, D.V.B.: *Second generation framing structure, channel coding and modulation systems for Broadcasting, Interative Services, News Gathering and other broadband satellite applications*, draft EN 302 307 V1.1.1 edition (2004).
8) *IEEE Standard for Local and Metropolitan Area Networks*, *IEEE 802.16e Standard*, http://standards.ieee.org/getieee 802/download/802.16e-2005.pdf (2006).
9) Blanksby, A. and Howland, C.: A 690-mW 1-Gbps 1024-b, rate-1/2 low-density parity-check code decoder, *J. Solid State Circuits*, Vol.37, No.3, pp.404–412 (2002).
10) Chen, Y. and Hocevar, D.: A FPGA and ASIC implementation of rate 1/2 8088-b irregular low density parity check decoder, *IEEE Global Telecommunications Conf.*, pp.113–117 (2003).
11) Liao, E., Yeo, E. and Nikolic, B.: Low-density parity-check code constructions for hardware implementation, *Proc. IEEE Conf. Communications*, pp.2573–2577 (2004).
12) Mansour, M. and Shanbhag, N.: Low power VLSI decoder architectures for LDPC codes, *Proc. Int. Symp. Low Power Electronics & Design*, pp.284–289 (2002).
13) Shimizu, K., Ishikawa, T., Togawa, N., Ikenaga, T. and Goto, S.: Power-efficient LDPC decoder architecture based on accelerated message-passing schedule, *IEICE Trans. Fundamentals*, Vol.E89-A, No.12, pp.3602–3612 (2006).
14) Li, X., Abe, Y., Shimizu, K., Qiu, Z., Ikenaga, T. and Goto, S.: Cost-efficient parallel irregular LDPC decoder with message passing schedule, *Int. Symp. Integrated Circuits*, pp.548–551 (2007).
15) Chien, Y.H. and Ku, M.K.: A high throughput H-QC LDPC decoder, *IEEE Int. Symp. Circuits & System*, pp.1648–1652 (2007).
16) Wang, Q., Shimizu, K., Ikenaga, T. and Goto, S.: A power-saved 1 Gbps irregular LDPC decoder based on simplified min-sum algorithm, *VLSI Design, Automation and Test*, pp.95–98 (2007).
17) *IEEE P802.11 Wireless LANs Joint Proposal: High throughput extension to the 802.11 Standard: PHY*, IEEE 802.11-05/1102r4 (Jan. 2006).

(Received May 23, 2008)
(Revised August 22, 2008)
(Accepted October 9, 2008)
(Released February 17, 2009)

(Recommended by Associate Editor: *Masaya Yoshikawa*)

Wen Ji received her B.S. degree in Electrical Engineering from Shanghai Jiao Tong Univeristy in 2006. She is currently a graduate student in the Graduate School of Information, Production and Systems, Waseda University. Her current research interest is VLSI design, especially LDPC decoder. She is a student member of IEEE.

Xing Li received his B.S. degree in Electrical engineering in Tsinghua University in 2005 and the M.E. degree in the Graduate School of Information, Production and Systems, Waseda University in 2007. He is now working in NEC Electoronics Corporation.

Takeshi Ikenaga received his B.E. and M.E. degrees in electrical engineering and the Ph.D. degree in information and computer science from Waseda University, Tokyo, Japan, in 1988, 1990, and 2002, respectively. He joined LSI Laboratories, Nippon Telegraph and Telephone Corporation (NTT) in 1990, where he has been undertaking research on the design and test methodologies for high-performance ASICs, a real-time MPEG2 encoder chip set, and a highly parallel LSI and System design for image-understanding processing. He is presently an associate professor in the system LSI field of the Graduate School of Information, Production and Systems, Waseda University. His current interests are application SOC for image, security and network processing. Dr. Ikenaga is a member of the IPSJ and the IEEE. He received the IEICE Research Encouragement Award in 1992.

Satoshi Goto was born in Hiroshima, Japan, 1945. He received the B.E. and the M.E. Degrees in Electronics and Communication Engineering from Waseda University in 1968 and 1970 respectively. He also received the Dr. of Engineering from the same University in 1981. He joined NEC Laboratories in 1970 where he worked for LSI design, Multimedia system and Software as GM and Vice President. Since 2003, he has been Professor at Graduate school of Information, Production and Systems of Waseda University at Kitakyushu. His main research interest is now on VLSI design methodologies for multimedia and mobile applications. He has published 7 books, 38 journal papers, 67 international conference papers with reviews. He served as GC of ICCAD and ASPDAC and was a board member of IEEE CAS society. He is IEEE Fellow, IEICE Fellow and Member of Academy Engineering Society of Japan.