High-Throughput Layered Decoder Implementation for Quasi-Cyclic LDPC Codes

Kai Zhang, *Student Member*, Xinming Huang, *Member, IEEE,* Zhongfeng Wang, *Senior Member, IEEE*

***Abstract*—This paper presents a high-throughput decoder design for the Quasi-Cyclic (QC) Low-Density Parity-Check (LDPC) codes. Two new techniques are proposed, including parallel layered decoding architecture (PLDA) and critical path splitting. PLDA enables parallel processing for all layers by establishing dedicated message passing paths among them. The decoder avoids crossbar-based large interconnect network. Critical path splitting technique is based on articulate adjustment of the starting point of each layer to maximize the time intervals between adjacent layers, such that the critical path delay can be split into pipeline stages. Furthermore, min-sum and loosely coupled algorithms are employed for area efficiency. As a case study, a rate-1/2 2304-bit irregular LDPC decoder is implemented using ASIC design in 90nm CMOS process. The decoder can achieve the maximum decoding throughput of 2.2Gbps at 10 iterations. The operating frequency is 950MHz after synthesis and the chip area is 2.9mm^2.**

***Index Terms*—Low-density parity-check codes, quasi-cyclic codes, parallel architecture, layered decoding, critical path splitting, min-sum algorithm, loosely coupled algorithm, VLSI.**

I. Introduction

INVENTED by Gallager in 1962 [1], Low-Density Parity-Check (LDPC) codes attracted much attention since they were rediscovered by MacKay in 1996 [2]. Compared with another kind of near Shannon limit Error Correction Codes (ECC), Turbo codes, LDPC codes have several advantages: more inherent parallelism, low hardware complexity, high throughput potentials and no error floors at high SNRs. LDPC codes are specified by sparse parity check matrices and can be fully represented by bipartite graphs (known as Tanner graph) with two sets of nodes, called the check nodes and the variable nodes. An LDPC code is called "regular" if the degrees of each set of the nodes are the same, while degrees of an irregular LDPC code vary according to some degree distributions. Irregular LDPC codes have higher decoding complexity but outperform regular LDPC codes by providing more powerful error-correction capability [3]. It is shown that irregular LDPC codes can approach theoretical limit within 0.06 dB [4]. Hence, LDPC codes are widely adopted by several recent communication standards such as 802.11n, DVB-S2, 802.15.3c (WPAN) [5] and 802.16e (WiMax) [6].

Manuscript received 1 October 2008; revised 15 January 2009. This work was supported in part by the National Science Foundation under Grants ECS-0725522.

K. Zhang and X. Huang are with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: kzhang@wpi.edu; xhuang@wpi.edu).

Z. Wang is with Broadcom Corp., Irvine, CA 92617 USA (e-mail: zfwang@broadcom.com).

Digital Object Identifier 10.1109/JSAC.2009.090816.

LDPC codes can be effectively decoded using the standard belief-propagation (BP) algorithm, also called sum-product algorithm (SPA). Two phases of messages, the check-to-variable (CTV) messages and the variable-to-check (VTC) messages, are transmitted along the edges of the Tanner graph to update each other iteratively. To simplify the BP algorithm, min-sum algorithm [7]–[9] is introduced to reduce the complexity of the check node operations. Also, in [10], [11], only variable (column) summations are stored during the BP algorithm to reduce memory size, which is called the loosely coupled algorithm in [12]. In particular, the aforementioned techniques can be employed to reduce the hardware complexity in message update units and to reduce the storage memories for the propagation messages.

Generally, the existing work in LDPC decoder architectures can be classified into three categories: fully parallel method [13], serial method [14] and partly parallel method [10], [15]–[17]. Implementation of an LDPC decoder is a tradeoff between error-correction performance, hardware complexity and decoding throughput. Partly parallel method has been popular, which exploits the parallelism for specially constructed architecture-aware LDPC codes. Among this class of LDPC codes, Quasi-Cyclic (QC) LDPC codes are well suited to hardware implementation because of the regularity in parity check matrices. In addition, QC-LDPC codes can provide comparable error-correction performance compared with random LDPC codes [17], [18].

With the increasing demand for high-data-rate wireless applications, many recent communication systems employ ultra-high throughput channel codes to match the data-rate requirements. For example, 802.15.3c standard is targeted for the data rate of multi-giga bits per second (Gbps), thus LDPC codes are preferred compared with convolutional codes and Turbo codes. However, it is a great challenge to design a high-through LDPC decoder due to the complexity of decoding algorithm. Since QC-LDPC codes are increasingly popular in emerging communication standards, we focus on the design and architecture of high-throughput QC-LDPC decoder in this paper.

The decoding throughput of an LDPC decoder can be calculated as

$$Throughput = \frac{Freq \times Block\,Length}{Cycles\,per\,Iter \times Num\,of\,Iter} \quad (1)$$

Therefore, three strategies can be attempted in order to improve the throughput: reducing the number of iterations required for convergence, reducing the decoding latency per iteration and improving the operating frequency. Correspondingly, three architecture-aware schemes are studied in this

paper, including layered decoding algorithm, parallel layered decoding architecture, and critical path splitting technique.

First, layered decoding algorithm (LDA) is adopted to reduce the required number of iterations by a factor of two for any given SNR, compared with the standard BP algorithm. Hence, the decoding throughput is supposed to be doubled without any bit error performance loss. Generally, there are two layered decoding methods: horizontal layered decoding [16], [19]–[23] and vertical layered decoding [24]. It has been proved these two methods are theoretically equivalent and both can converge twice as fast as the BP algorithm [25]. In this paper, we employ the horizontal layered decoding strategy because it is favorable for the min-sum algorithm.

Secondly, we propose a novel scheme, namely parallel layered decoding architecture (PLDA), that enables concurrent processing among all layers. In traditional LDA, the layers are processed sequentially which leads to longer decoding latency per iteration [19]. In PLDA, precisely scheduled message passing among different layers guarantees that all updated messages are passed to their designated locations in connected layers. The parity-check matrix optimization procedure adds specific offsets to each layer (row) in the base parity check matrix, making the idle time intervals between connected layers sufficiently large for message passing. Moreover, PLDA supports the decoding architecture in which check node updates unit (CNU) and variable node updates unit (VNU) are combined into a single functional unit. Thus, no extra clock cycles are needed to complete the variable node update as they have been merged into the CNU. As a result, the number of clock cycles per iteration in PLDA can be reduced by 75% compared with the existing architectures.

Finally, the technique of reducing critical path delay is proposed to improve the operating frequency at circuit-level. The combination of CNU and VNU results in a long critical path in the decoder implementation [26], which limits the maximum clock speed. Fortunately, there are idle time intervals among different layers which allows the iterative messages to be processed and passed to the next layer within several clock cycles. Therefore, we can insert registers to split the CNU (including the VNU) into several pipeline stages. Consequently, the critical path is split into multiple stages and the clock speed can be improved dramatically, on condition that the number of stages does not exceed the idle time intervals. In practice, this critical path splitting method can increase the maximum frequency of the decoder by a factor of 3 or higher.

To demonstrate the aforementioned three techniques, a rate-1/2 2304-bit QC-LDPC code is selected from 802.16e standard as a case study. In addition, existing techniques such as min-sum algorithm and loosely coupled algorithm are also employed to simplify decoding complexity and to reduce the chip area of the decoder design.

The remainder of this paper is organized as follows. Section II introduces the background of LDPC codes, BP algorithm and other modified decoding algorithms. Three key strategies on improving throughput are discussed in Section III. The decoder architecture is presented Section IV based on the selected rate-1/2 2304-bit LDPC codes. Section V provides the ASIC implementation results and performance comparisons with existing designs. Conclusions and future work are presented in Section VI.

II. LDPC Codes and Decoding Algorithms

A. Quasi-Cyclic LDPC Codes

The LDPC codes can be described by a $M \times N$ sparse parity check matrix $\mathbf{H}$, in which most of the elements are 0's and only a few are 1's. M denotes the number of parity check equations, which is the number of check nodes. N is the block length, which is the number of variable nodes.

QC-LDPC codes are a special class of the LDPC codes with structured $\mathbf{H}$ matrix which can be generated by the expansion of an $m_b \times n_b$ base matrix. Each 1's element in the base matrix can be expanded by a circularly right-shifted $p \times p$ identity sub-matrix. The structure of the parity check matrix makes hardware designer easy to determine the locations of non-zero elements. Random connections on Tanner graph now become well-regulated and easy to handle. Therefore, QC-LDPC codes have been adopted by several communication standards, such as 802.11n, 802.15.3c and 802.16e.

In this paper, we exemplify a rate-1/2 2304-bit QC-LDPC code from 802.16e system, whose $\mathbf{H}$ matrix is shown in Fig. 1. The $\mathbf{H}$ matrix is extended from a 12×24 base matrix in which the size of sub-matrix p is set to be 96. An element of 0 in the base matrix can be expanded as a 96×96 identity matrix. A blank element can be expanded as a 96×96 all zero matrix. An element of a non-zero integer $s(i,j)$ can be expanded by circularly shifting a 96×96 identity matrix by $s(i,j)$ towards right.

B. Min-Sum Algorithm

The BP algorithm [1] provides a powerful method for decoding LDPC codes. Min-sum algorithm (MSA) [7]–[9] is introduced to improve the BP algorithm thus to reduce the complexity of the CNU. Before presenting the MSA, we first give some definitions as follows: Let c_n denote the n-th bit of a codeword and y_n denote the corresponding received value from the channel. Let $r_{mn}[k]$ be the check-to-variable (CTV) message from check node m to variable node n during the k-th iteration. Let $q_{mn}[k]$ be the variable-to-check (VTC) message from variable node n to check node m. Let $N(m)$ denote the set of variables that participate in check m and $M(n)$ denote the set of checks that participate in variable n. The set $N(m)$ without variable n is denoted as $N(m) \setminus n$ and the set $M(n)$ without check m is denoted as $M(n) \setminus m$.

1) **Initialization:**
Under the assumption of equal priori probability, compute the channel probability p_n (intrinsic information) of the variable node n, by:

$$p_n = log \frac{P(y_n \mid c_n = 0)}{P(y_n \mid c_n = 1)} \tag{2}$$

The CTV message r_{mn} is initially set to zero.

2) **Iterative Decoding:**
At the k-th iteration, for variable node n, calculate VTC message $q_{mn}[k]$ by

$$q_{mn}[k] = p_n + \sum_{m' \in \{M(n) \setminus m\}} r_{m'n}[k-1] \tag{3}$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1		94	73						55	83			7	0										
2		27				22	79	9				12		0	0									
3				24	22	81		33				0			0	0								
4	61		47						65	25						0	0							
5			39				84			41	72						0	0						
6					46	40		82				79	0					0	0					
7			95	53						14	18								0	0				
8		11	73				2			47										0	0			
9	12				83	24		43				51									0	0		
10						94		59			70	72										0	0	
11			7	65					39	49													0	0
12	43					66		41				26	7											0

Fig. 1. Parity-check matrix for the selected rate-1/2 LDPC code in 802.16e standard.

Meanwhile, the decoder can make a hard decision by calculating the APP (a-posterior probability) as follows

$$\Lambda_n [k] = p_n + \sum_{m' \in M(n)} r_{m'n} [k-1] \tag{4}$$

Decide the n-th bit of the decoded codeword $x_n = 0$ if $\Lambda_n > 0$ and $x_n = 1$ otherwise. The decoding process terminates when the entire codeword $x = [x_1, x_2, \cdots\cdots x_N]$ satisfy the parity check equations: $\mathbf{H}x = 0$, or the preset maximum number of iterations is reached.

If the decoding process does not stop, then calculate the CTV message r_{mn} for check node m by

$$\begin{aligned} r_{mn}[k] &= \left(\prod_{n' \in \{N(m) \backslash n\}} sign\left(q_{mn'}[k]\right) \right) \\ &\quad \times \left(\alpha \times \min_{n' \in \{N(m) \backslash n\}} \{|q_{mn'}[k]|\} \right) \end{aligned} \tag{5}$$

Here, a normalized factor α is introduced to compensate for the performance loss in the min-sum algorithm compared to BP algorithm [9]. In this paper, α is set to be 0.75.

Using the min-sum algorithm, LUTs which implement the intricate non-linear function in standard BP algorithm are now replaced by rather simple comparators, resulting in lower computational complexity of CNU.

C. Loosely Coupled Algorithm

In BP algorithm, messages are transmitted between check nodes and variable nodes iteratively to update each other. VTC messages are updated using channel probabilities and CTV messages from the check nodes. The CTV message are updated using the VTC messages received from the variable nodes. Every VTC and CTV message should be passed along the edges of the Tanner graph. Frequent message passing between check and variable nodes requires a large interconnection network which could occupy a significant part of the chip area in a decoder implementation.

Loosely coupled algorithm [12] was introduced to reduce the interconnection problem. The decoder does not exchange the CTV and VTC messages between check nodes and variable nodes. Instead, it only passes the check summations Δ_m and variable summations Λ_n (also called APP messages). At every variable node, the VNU first recovers individual CTV message r_{mn} from the received check summation Δ_m. Subsequently, it updates the variable summation Λ_n and pass it to the CNUs.

III. High Throughput Strategies

In this section, we discuss three techniques that can be implemented to improve the throughput of QC-LDPC decoders. First, we adopt the LDA which reduces the number of iterations into half. Both min-sum and loosely coupled algorithms are integrated with LDA to reduce the decoding complexity. Secondly, parallel layered decoding architecture (PLDA) is proposed that enables concurrent decoding among all layers. In PLDA, messages are passed through articulately-defined fixed message-passing paths and therefore it eliminates the crossbar interconnections. PLDA can provide the same bit error performance as of normal LDA, but the decoding latency per iteration is reduced significantly. Finally, critical path splitting technique is proposed to pipeline the CNU design into multiple stages, which improves the system clock frequency considerably.

A. Layered Decoding Algorithm

In BP algorithm [1], the two-phase mutual messages, namely VTC messages q_{mn} and CTV messages r_{mn}, are updated by separate processing units and passed to each other iteratively. q_{mn} updates will not start until all of the r_{mn} are prepared and vice versa. In horizontal layered decoding, the CTV message from the current layer will be passed vertically to all other unprocessed layers that belong to the same variable node. In each iteration, the horizontal layers are processed sequentially from the top to the bottom layer.

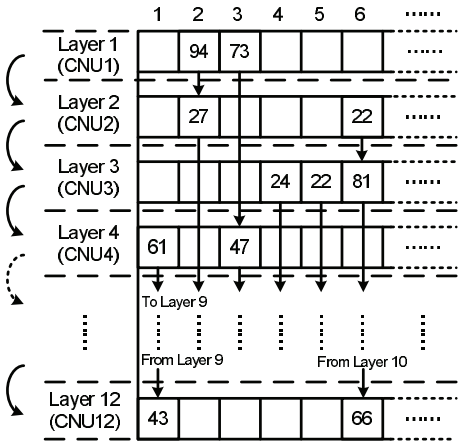


Fig. 2. Message passing flow in horizontal LDA.

As an example, the $\mathbf{H}$ matrix of our selected LDPC code from 802.16e standard is quasi-cyclic, which consists of sub-matrices that are generated by circularly shifting an identity matrix, as shown in Fig. 1. Such structure is well suited for horizontal LDA as each sub-matrix has the column weight of one and we can treat each row of the base matrix as a layer. Message passing flow of LDA for the selected $\mathbf{H}$ matrix is illustrated in Fig. 2.

Loosely coupled algorithm is also adopted in this work to reduce the interconnection complexity. As illustrated in Fig. 3, at the j-th layer, the VTC messages are first recovered from the variable summations Λ_n and the CTV message from memories, as in (6). At the output of CNUj, the updated CTV messages $\left\{r_{j,n_j} \mid n_j \in N(j)\right\}$ in (7) are stored back to the CTV memories. Variable summation $\left\{\Lambda_{n_j} \mid n_j \in N(j)\right\}$ are renewed as in (8) and sent back to the APP memory. The entire flow can be expressed as follows:

$$q_{j,n_j} = \Lambda_{n_j} - r_{j,n_j} \tag{6}$$

$$\begin{aligned} \overline{r_{j,n_j}} &= \left(\prod_{n'_j \in \{N(j) \backslash n_j\}} sign\left(q_{j,n'_j}\right) \right) \\ &\quad \times \left(\min_{n'_j \in \{N(j) \backslash n_j\}} \left\{ \left| q_{j,n_j} \right| \right\} \times \alpha \right) \end{aligned} \tag{7}$$

$$\overline{\Lambda_{n_j}} = q_{j,n_j} + \overline{r_{j,n_j}} \tag{8}$$

Compared with conventional LDA, it can be observed that loosely coupled algorithm does not require variable node operations. In other words, VNUs can be eliminated since their main function of updating the variable summations Λ_n can be done by CNUs using VTC messages $r_{j,n_j}[k]$ from previous layers, as indicated in (8).

B. Parallel Layered Decoding Architecture

As discussed above, the essential reason that LDA can reduce the number of iterations is that the latest extrinsic messages are passed to and employed by the subsequent layers within the current iteration. Therefore, LDA requires layers to be processed sequentially, which results in a large decoding latency per iteration. A method of increasing parallelism inside a layer is proposed in [16], [20], but all layers are still processed in series. Although decoding throughput can be improved, this method introduces crossbar-based interconnection networks that increase the hardware complexity.

Motivated by the partly parallel mechanism mentioned in [15], we propose the PLDA that allows all layers to be processed concurrently. Each layer generates and sends updated messages and at the same time it also receives the updated messages from other layers. Unlike the method proposed in [16], [20], PLDA uses parallel processing among all layers and serial processing within each layer. Detailed message processing flow at the j-th layer (CNU) can be summarized as follows:

1) Fetch the corresponding variable summations $\left\{\Lambda_{n_j} \mid n_j \in N(j)\right\}$ from the APP memory and CTV messages $\left\{r_{j,n_j} \mid n_j \in N(j)\right\}$ from local CTV memory.
2) Calculate the VTC messages $\left\{q_{j,n_j} \mid n_j \in N(j)\right\}$ in the same row using (6).
3) Calculate horizontally to obtain new CTV messages $\left\{\overline{r_{j,n_j}} \mid n_j \in N(j)\right\}$, as in (7).
4) Immediately update the variable summations $\left\{\overline{\Lambda_{n_j}} \mid n_j \in N(j)\right\}$ using (8).
5) Deliver the new variable summation $\overline{\Lambda_{n_j}}$ to the same location at another layer.

Hereby, we explain how to pass $\overline{\Lambda_{n_j}}$ messages in PLDA. Instead of passing $\overline{\Lambda_{n_j}}$ to all unprocessed layers as in conventional LDA, PLDA only sends $\overline{\Lambda_{n_j}}$ to the layer that will use it next. Let us take the first column of the $\mathbf{H}$ base matrix as an example. Non-zero entries are at the 4th, 9th and 12th layer whose permutation numbers are 61, 12 and 43, respectively. Now we suppose that at cycle 0 each of these three corresponding CNUs starts to process from the first row of the sub-matrix, corresponding to the 62th, 13th and 44th column indices. In general, corresponding row and column indices of the entry being processed at cycle l can be calculated as

$$r_{index} = l \tag{9}$$

$$c_{index} = mod\left(l + 1 + s\left(i,j\right), 96\right) \tag{10}$$

In Fig. 4, it illustrates the message passing routes in PLDA using the the first column of the parity check matrix in Fig. 1 as an example. The operation sequence in time of these three layers is described as the following:

1) At cycle 0, CNUs at layers 4, 9 and12 start to process simultaneously from row 1 of each sub-matrix, corresponding to column 62, column 13 and column 44 according to (10).
2) At cycle 18, all three layers are processing at row 19, corresponding to column 80 (in layer 4), column 31 (in layer 9) and column 62 (in layer 12). Layer 12 calls for the latest summation message of column 62 which has already been updated by layer 4. Therefore, the updated variable summations at layer 4 should be sent to layer 12 (Layer 4 →Layer 12).

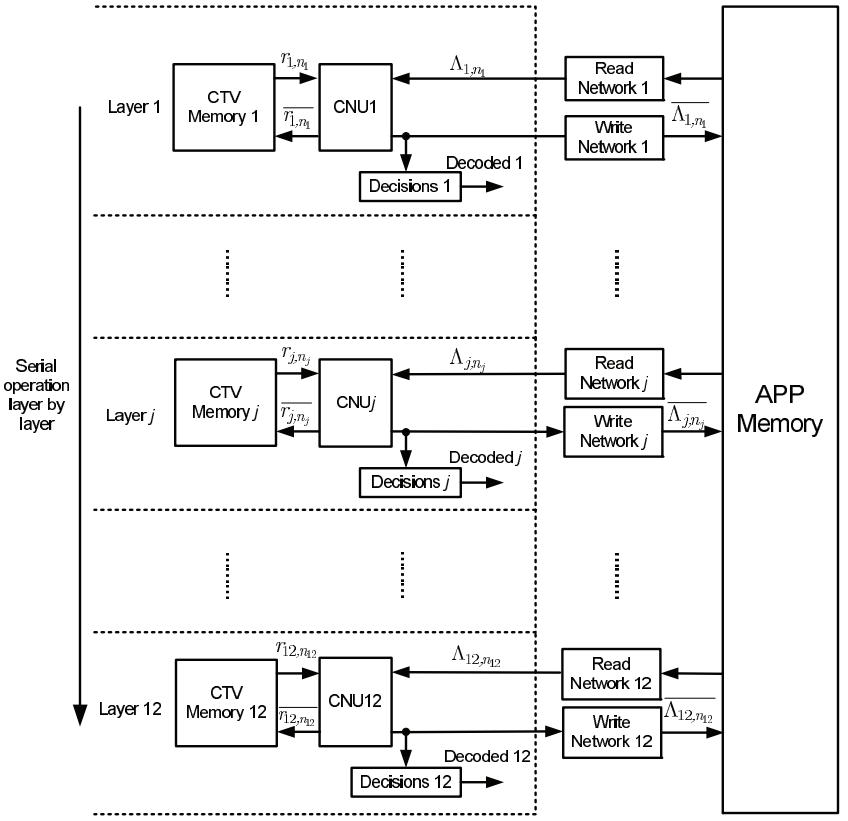


Fig. 3. Architecture of horizontal LDA with loosely coupled algorithm.

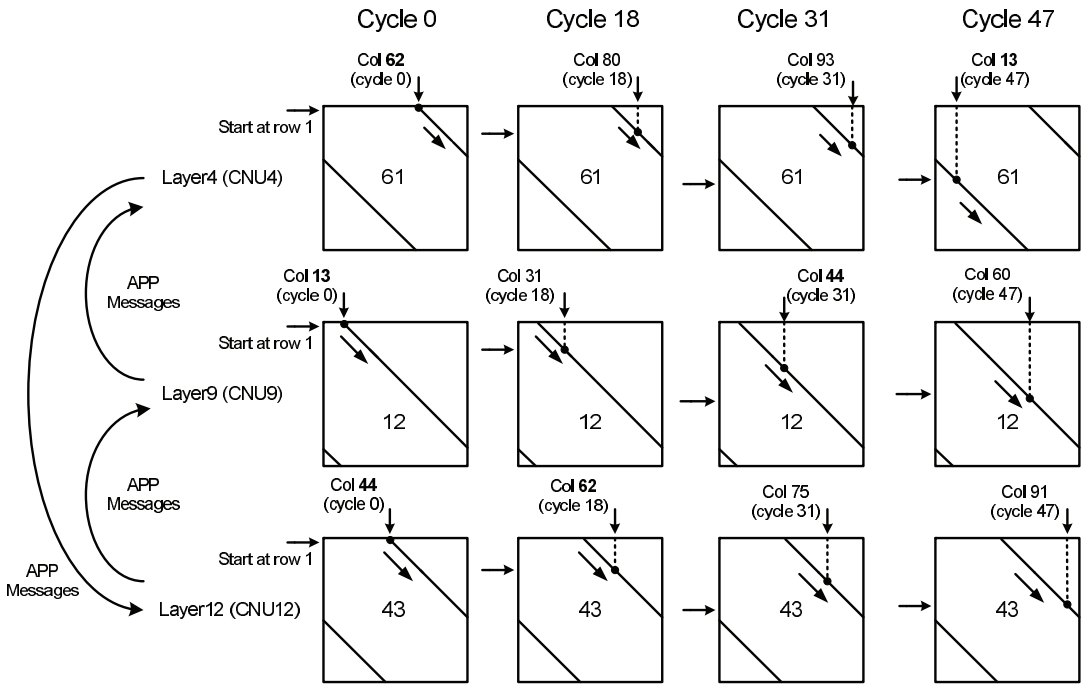


Fig. 4. Processing status at four different clock cycles.

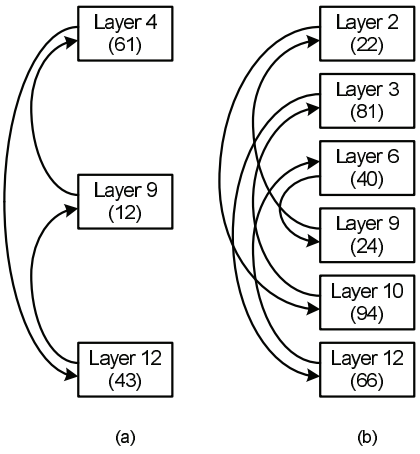


Fig. 5. Varible summations passing directions of the $\mathbf{H}$ base matrix: (a) for column 1 (b) for column 6.

3) At cycle 31, all three layers are processing at row 32, corresponding to column 93 (in layer 4), column 44 (in layer 9) and column 75(in layer 12). Layer 9 calls for the latest summation message of column 44 which has already been updated by layer 12. Therefore, the updated variable summations at layer 12 should be sent to layer 9 (Layer 12 →Layer 9).
4) Similarly, at cycle 47, all three layers are processing row 48, corresponding to column 13 (in layer 4), column 60 (in layer 9) and column 91 (in layer 12). Layer 4 calls for latest summation message of column 13 which has already been updated by layer 9. Therefore, the updated variable summations at layer 9 should be sent to layer 4 (Layer 9 →Layer 4).

Based on the description above, message passing routes for column 1 of base matrix is shown in Fig. 5(a). An additional example is illustrated in Fig. 5(b) for column 6 of the base matrix. In general, we can determine the message passing routes among layers based on the their permutation values. For each column, we can sort the permutation values of all layers in descending order. Each layer then passes messages to the next layer with a smaller permutation value. Moreover, the layer with the smallest permutation value loops back and connects to the layer with the largest value. This message passing scheme guarantees the updated messages among all layered are processed progressively within each iteration.

C. Critical Path Splitting

According to the message passing mechanism explained above, we suppose CNUs in all layers start at row 1 of each sub-matrix. Actually, the CNUs can start to operate from different rows, which is equivalent to adding an offset to the permutation value of each layer. For example, Fig. 6 shows a modified $\mathbf{H}$ base matrix with a set of offset values added for different layers. The offset values are carefully selected, such that the difference of modified permutation values between any two layers is at least 5. It means each layer needs to read, update, store and pass the message to the next connected layer within 5 clock cycles. The detailed steps for the processing in a layer are shown in in Fig. 7. In other words, each CNU has a period of 4 cycles to complete the task of reading old message and update new message, as indicated in (6) to (8). The message passing rule remains the same, except that the updated permutation values should be used to determine the connections in PLDA.

From Fig. 6, we can see that the row weight of each layer is either 6 or 7, which requires 6 or 7 messages to be compared in the CNU. Combined with other necessary functions such as adding, rounding and memory read/write, CNU becomes the critical path that limits the maximum operating frequency of the decoder. Taking advantage of the 4-cycle time intervals, we can split the critical path of CNU into 4 pipeline stages by inserting 3 levels of registers. An optimal splitting yields balanced delays among the pipeline stages. The implementation of critical path splitting will be given in Section IV-B.

IV. Proposed Decoder Architecture

In order to prove the concept of our proposed PLDA architecture and high-throughput strategies, a rate-1/2 2304-bit QC-LDPC code selected from 802.16e standard is designed based on the offset-modified $\mathbf{H}$ matrix. The size of each sub-matrix is chosen to be 96×96. Before constructing the functional units of the decoder, we first perform fixed-point analysis to quantize the word-length of messages. In fact, word-length quantization is a tradeoff between memory resources and bit-error-rate performance. Fig. 8 shows the BER performance of the selected rate-1/2 LDPC code using message word-length $(3, 2)$. As demonstrated in [27], 5 bits (3 bits for the integer part and 2 bits for the fractional part) are adequate for representing the absolute values of the extrinsic messages. Considering one additional sign bit, we choose 6 bits fixed-point representation for messages in this implementation.

A. Overall Decoder Architecture

The overall architecture of the QC-LDPC decoder is shown in Fig. 9. It consists of check node processing units (CNUs), APP memory banks, CTV memory banks and hard-decision units. The entire $\mathbf{H}$ matrix is divided into 12 layers and each layer employs a dedicated CNU. APP memory banks and CTV memory banks are used to store variable summations and CTV messages. Each non-zero sub-matrix corresponds to one APP memory unit and one CTV memory unit. In layered decoding, each APP memory exports a summation message to the CNU and imports a new summation message from the CNU of another layer. Similarly, each CTV memory exports a CTV message to the CNU and imports a new CTV message from the same CNU for each layer. As a result, single-port memories that support concurrent read and write operations are used in the design. Since there are 76 sub-matrices in total, the APP memory bank and the CTV memory bank each consists of 76 small single-port memory units.

Each layer in the PLDA architecture corresponds to 6 or 7 APP memory units, which means that 6 or 7 updated variable summations of the j-th layer $\overline{\Lambda_{j,n_j}}$ will be delivered to APP memory units in other layers. The passing routes are based on the message passing scheme described in Section III-B and Fig. 5. Therefore, fixed connection wires are used to

Layer	Offset	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	+0		94	73						55	83			7	0										
2	+8		35				30	87	17				20		8	8									
3	+0				24	22	81		33				0			0	0								
4	+12	73		59						77	37						12	12							
5	+84			27				72			29	60						84	84						
6	+0					46	40		82				79	0					0	0					
7	+88			87	45						6	10								88	88				
8	+8		27	81				10			55										8	8			
9	+0	12				83	24		43				51									0	0		
10	+16						14		75			86	88										16	16	
11	+0			7	65					39	49													0	0
12	+80	27					50		25				10	87											80

Fig. 6. Offset-modified parity-check matrix for rate-1/2 LDPC code in 802.16e.

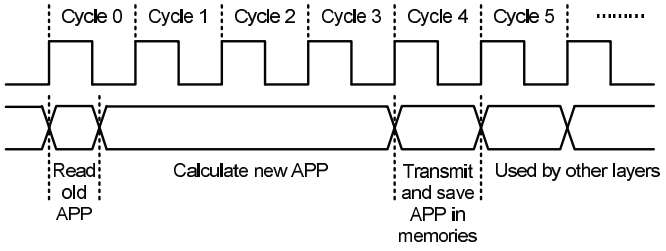


Fig. 7. Timing diagram for parallel layered decoding architecture.

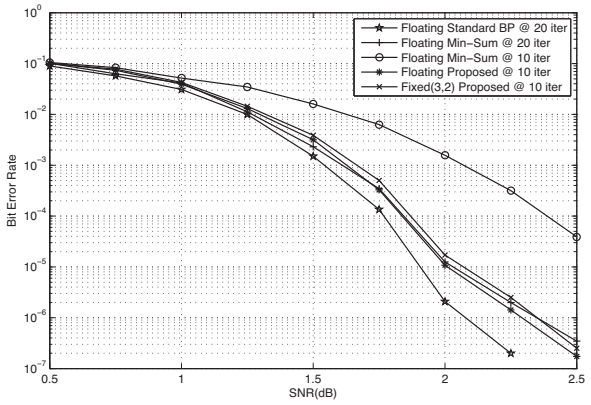


Fig. 8. BER performance comparison of different decoding algorithms.

connect the two phases of messages, instead of crossbar-based networks.

B. Pipeline Architecture for CNU

For check node update, each layer employs a CNU to perform a series of functions including subtraction, comparison and addition. Consequently, the CNU becomes the longest delay path in timing. In min-sum algorithm, a CNU needs to compare 6 or 7 numbers to find the minimum value and its location as well as the second minimum value. Thus, the comparator becomes a key component in a CNU. A 2-input comparator consists of an adder and a multiplexer. A 3-input comparator can be implemented with three adders, five multiplexers and some basic logic gates. Based on comparison units of 2-input and 3-input comparators, 6-input or 7-input comparator can be constructed by combining three levels of 2-input or 3-input comparators, as shown in Fig. 10.

Fig. 10 shows detailed architecture of CNU and its functional block inside. The *subtractor* and *adder* blocks fulfill the functions defined in (6) and (8), respectively. Two *quantizers* are inserted to prevent the CTV and variable summations from overflow during computation. The *abs* block calculates the absolute values of VTC messages and the *compare & select* block determines the values of CTV messages.

As explained in Section III-C, in order to reduce the critical path delay and improve clock speed, a CNU can be split into 4 pipeline stages by inserting 3 set of registers. Registers are carefully inserted such that all 4 pipeline stages have approximately the same delay, since maximum frequency is determined by the longest delay path. Delays of the functional units and the pipeline stages are also shown in Fig. 10.

C. Decision Units

The output bits of the LDPC decoder are decided by the signs of the variable summations. In traditional horizontal LDA, decisions can be made during the variable node processing at the bottom layer. PLDA operates in a slightly different way in which every layer generates variable summations during each iteration, as CNUs in different layers are working concurrently. As illustrated in Fig. 11, let us take the first column of the modified $\mathbf{H}$ base matrix with offset as an example. CNU 4, CNU 9 and CNU 12 start to operate from row 13, row 1 and row 81, respectively, as defined by the offset values shown in Fig. 6, which correspond to column 74, column 13 and column 28, respectively. According to the message passing rule derived in Section III-B, variable summations passing routes of the first column is Layer 4→Layer 12→Layer 9→Layer 4, also shown in Fig. 11. After completion of each iteration, the final variable summations for columns 13 through 27 are stored at layer 12. Variable summations for columns 28 through 73 are stored at layer 4. Similarly, variable summations for column 74 through 12

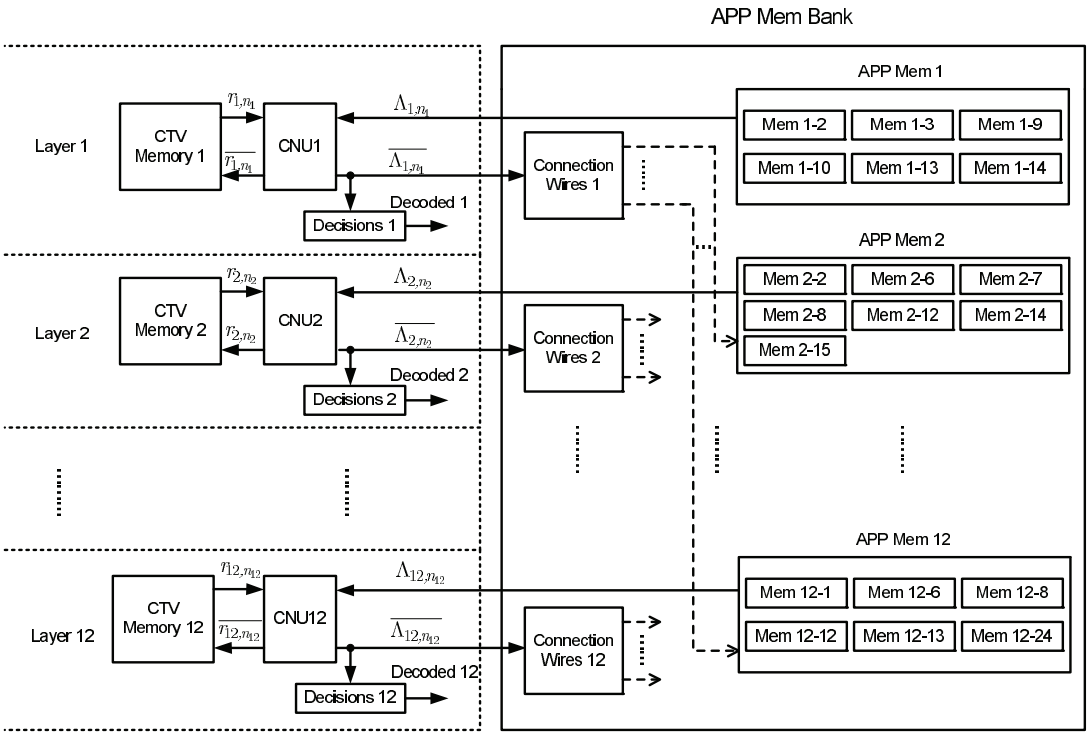


Fig. 9. Overall PLDA architecture for QC-LDPC codes.

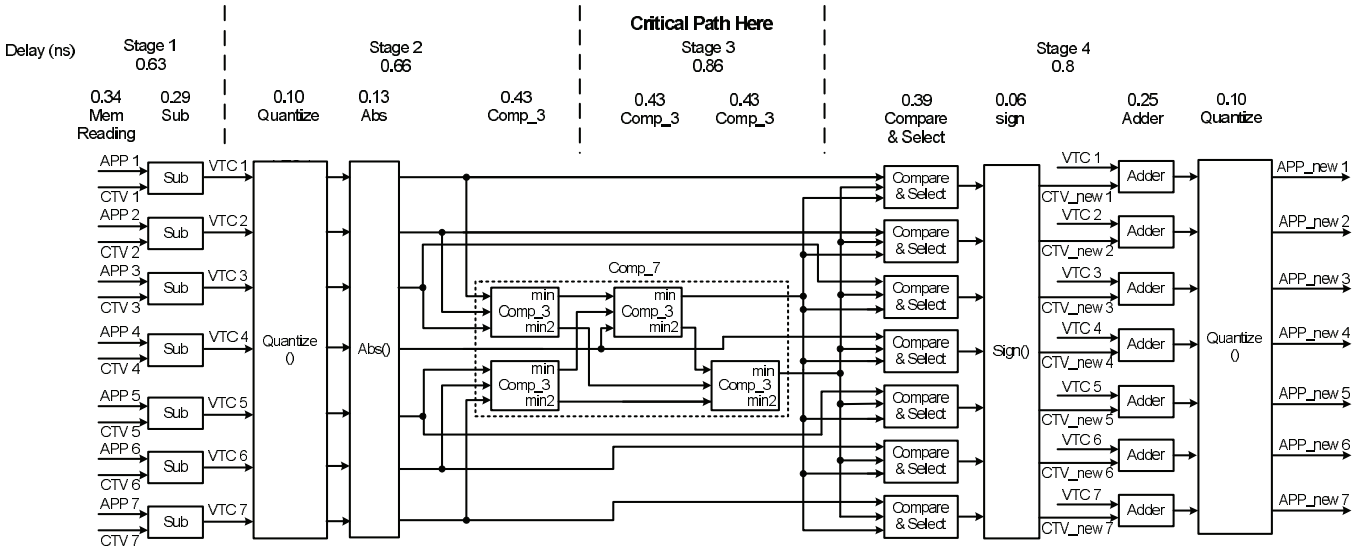


Fig. 10. CNU architecture with critical path splitting into 4 pipeline stages.

are stored at layer 9. Therefore, the hard decision bits for columns 1 through 96 are stored in the memories distributed in 3 different layers.

In this design, we employ a convenient and robust "early termination" strategy, similar to [26], [28]. The pivot of the early termination strategy lies in that the hard decision bits from previous iteration are stored and compared with the decision bits from current iteration. If all decoded bits are identical, then the decoder indicates successful decoding of a codeword and the iterative decoding process terminates. Otherwise, the decoding process continues until the maximum number of iterations is reached.

V. Implementation Results

In order to evaluate the performance of the proposed PLDA architecture, we implement a rate-1/2 2304-bit QC-LDPC decoder in TSMC 90nm 1.0V CMOS technology with 8-layer metals. We complete synthesis and core area place and route using Synopsys tools.

Implementation results show that the decoder can operate at a maximum frequency of 950MHz after synthesis, corresponding to 2.2Gbps decoding throughput using 10 iterations. A total of 152 single-port memory units each with size of 96×6 bits (including one sign bit for each message) are employed, which sums up to $87,752$ bits of memory use and occupies more than 75% of the core area. PLDA only needs

TABLE I
OVERALL COMPARISON BETWEEN PROPOSED DECODER AND OTHER EXISTING LDPC DECODERS.

	C. Liu [20]	X. Shih [26]	T. Brack [21]	G. Gentile [22]	Y. Ueng [29]	M. Karkooti [30]	Proposed Decoder
Code Length	576~2304	576~2304	576~2304	576~2304	2304	1944	2304
Frequency	150MHz	83.3MHz	333MHz	400MHz	200MHz	412MHz	950MHz
Iterations	20	2~8	10, 15	15	4.6 (average)	15	10
Throughput	105Mbps	60~220Mbps	133-928Mbps	128-746Mbps	106Mbps	736Mbps	2.2Gbps
Technology	90nm	130nm	130nm	65nm	180nm	130nm	90nm
Area	$6.25mm^2$	$8.29mm^2$	$3.83mm^2$	$0.59mm^2$	-	$2.4mm^2$	$2.9mm^2$
Power	264mW	52mW	-	-	-	502mW	870mW

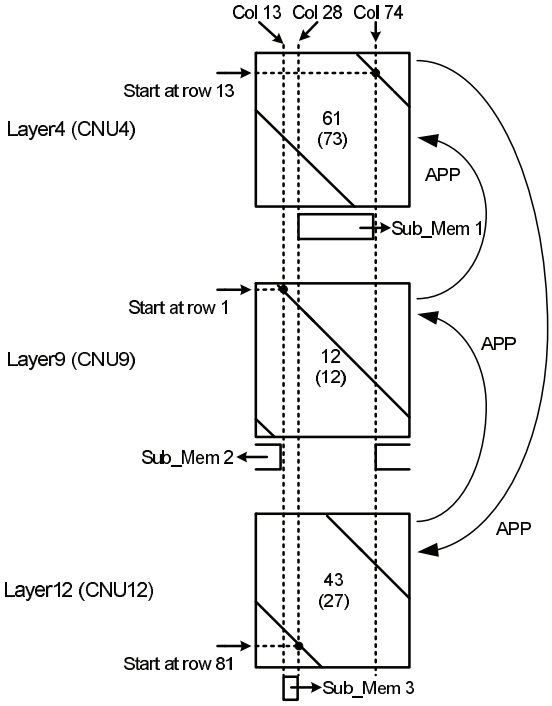


Fig. 11. Register allocation in each section of the hard decision bits.

Fig. 12. Layout of the decoder core area.

$(p+3)\times Iter$ clock cycles for the decoding process and p is the size of the sub-matrix. In [26] and [29], this number rises to $p\times(4\times Iter+1)$ and $p\times(5\times Iter)+12$, respectively. Hence, under the same number of iterations, PLDA could reduce the decoding latency by approximately 75%. Moreover, the implementation results show that the maximum operating frequencies with and without critical path splitting method are 950MHz and 305MHz, respectively, which demonstrates an improvement of the decoder speed by a factor of 3. Combined with LDA that doubles the convergence speed, the proposed architecture can significantly improve the throughput of the QC-LDPC decoder up to multi-Gbps.

Fig. 12 shows the layout view of the decoder with core area of $1.8mm\times1.6mm$ and logic density of 70%. The design does not have a crossbar interconnection network, since PLDA employs fixed message passing paths. Single-port memories are generated by Synopsys DesignWare tool and thus flattened during synthesis and place and route design flow. The core area of the decoder is $2.9mm^2$ and the estimated power consumption is 870 mW.

Table I shows the decoder implementation results compared with the existing QC-LDPC decoders. Note that the throughput values from [21], [22] are recalculated based on the decoded bits for fair comparison to other implementations listed in Table I. We show that the proposed decoder can achieve higher decoding throughput with comparable or smaller chip area. The decoder consumes more power mainly due to its high operating frequency.

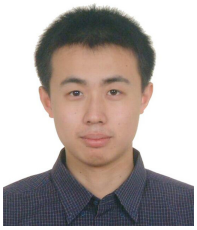
VI. CONCLUSIONS

LDPC codes are widely used in recent communication systems due to their superior error-correction performance. In this paper, we proposed a new architecture to improve the throughput of QC-LDPC decoders. With parallel layered decoding architecture and critical path splitting technique, the decoder implementation, which uses 90nm CMOS technology, can achieve 2.2Gbps decoding throughput for selected rate-1/2 irregular QC-LDPC codes. In addition, min-sum and loosely coupled algorithms are employed for area efficiency and the core size is $2.9mm^2$. The proposed PLDA architecture is very suitable for high data rate communication systems employing LDPC codes for channel coding.

REFERENCES

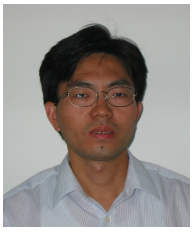
[1] R. Gallager, "Low-Density Parity-Check Codes," *IRE Tans. Inf. Theory*, vol. 7, pp. 21–28, 1962.
[2] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645–, Aug 1996.
[3] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 585–598, Feb 2001.

[4] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619–637, Feb 2001.
[5] *[Online]: http://www.ieee802.org/15/pub/TG3c.html.*
[6] *[Online]: http://www.ieee802.org/16/tge.*
[7] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.
[8] E. B. Guilloud and J. Danger, "λ-Min Decoding Algorithm of Regular and Irregular LDPC codes," in *Proc. 3rd Int. Symp. Turbo Codes and Related Topics*, Sep 2003, pp. 451–454.
[9] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-Complexity Decoding of LDPC Codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.
[10] Y. Chen and D. Hocevar, "A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder," in *Proc. IEEE GLOBECOM*, vol. 1, Dec. 2003, pp. 113–117.
[11] Z. Wang and Z. Cui, "A Memory Efficient Partially Parallel Decoder Architecture for QC-LDPC Codes," in *Proc. 39th Asilomar Conf. Signals, Syst., Comput.*, Nov. 2005, pp. 729–733.
[12] S.-H. Kang and I.-C. Park, "Loosely coupled memory-based decoding architecture for low density parity check codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 5, pp. 1045–1056, May 2006.
[13] A. Blanksby and C. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar 2002.
[14] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "VLSI architectures for iterative decoders in magnetic recording channels," *IEEE Trans. Magn.*, vol. 37, no. 2, pp. 748–755, Mar 2001.
[15] T. Zhang and K. Parhi, "VLSI implementation-oriented (3,k)-regular low-density parity-check codes," in *Proc. IEEE Workshop on Signal Process. Syst. (SiPS)*, 2001, pp. 25–36.
[16] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.
[17] Z. Wang and Z. Cui, "Low-Complexity High-Speed Decoder Design for Quasi-Cyclic LDPC Codes," *IEEE Trans. VLSI Syst.*, vol. 15, no. 1, pp. 104–114, Jan. 2007.
[18] M. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inform. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.
[19] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop on Signal Process. Syst. (SiPS)*, Oct. 2004, pp. 107–112.
[20] C.-H. Liu, S.-W. Yen, C.-L. Chen, H.-C. Chang, C.-Y. Lee, Y.-S. Hsu, and S.-J. Jou, "An LDPC Decoder Chip Based on Self-Routing Network for IEEE 802.16e Applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 684–694, March 2008.
[21] T. Brack, M. Alles, F. Kienle, and N. Wehn, "A synthesizable IP core for WiMAX 802.16e LDPC code decoding," in *Proc. IEEE 17th Int. Symp. Personal, Indoor and Mobile Radio Communications*, Sept. 2006, pp. 1–5.
[22] G. Gentile, M. Rovini, and L. Fanucci, "Low-complexity architectures of a decoder for IEEE 802.16e LDPC codes," in *Proc. Euromicro Conf. Digital System Design (DSD)*, Aug. 2007, pp. 369–375.
[23] K. Gunnam, G. Choi, M. Yeary, and M. Atiquzzaman, "VLSI architectures for layered decoding for irregular LDPC codes of WiMax," in *Proc. IEEE Int. Conf. Commun. (ICC)*, June 2007, pp. 4542–4547.
[24] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.
[25] E. Sharon, S. Litsyn, and J. Goldberger, "Efficient Serial Message-Passing Schedules for LDPC Decoding," *IEEE Trans. Inform. Theory*, vol. 53, no. 11, pp. 4076–4091, Nov. 2007.
[26] X.-Y. Shih, C.-Z. Zhan, C.-H. Lin, and A.-Y. Wu, "An 8.29 mm^2 52 mW Multi-Mode LDPC Decoder Design for Mobile WiMAX System in 0.13 μm CMOS Process," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 672–683, March 2008.
[27] T. Zhang, Z. Wang, and K. Parhi, "On finite precision implementation of low density parity check codes decoder," in *Proc. IEEE ISCAS*, vol. 4, May 2001, pp. 202–205 vol. 4.
[28] R. Shao, S. Lin, and M. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug 1999.
[29] Y.-L. Ueng, C.-J. Yang, Z.-C. Wu, C.-E. Wu, and Y.-L. Wang, "VLSI decoding architecture with improved convergence speed and reduced decoding latency for irregular LDPC codes in WiMAX," in *Proc. IEEE ISCAS*, May 2008, pp. 520–523.
[30] M. Karkooti, P. Radosavljevic, and J. Cavallaro, "Configurable, High Throughput, Irregular LDPC Decoder Architecture: Tradeoff Analysis and Implementation," in *Proc. IEEE 17th Int. Conf. Application specific Systems, Architectures and Processors (ASAP)*, Sept. 2006, pp. 360–367.



Kai Zhang (S'09) received the B.E. degree in Information Engineering and the M.S. degree in Microelectronics, both from Xi'an Jiaotong University, Xi'an, China in 2004 and 2007 respectively. He is currently working toward the Ph.D. degree in Electrical and Computer Engineering at Worcester Polytechnic Institute, MA. His research interests are VLSI architectures for error correction codes and cooperative communications.

Xinming Huang (M'01) is an Assistant Professor in the Department of Electrical and Computer Engineering at Worcester Polytechnic Institute (WPI), Worcester, MA. He received the Ph.D. degree in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA in 2001. He was a member of technical staff with the wireless advanced technology laboratory, Bell Labs of Lucent Technologies, from 2001 to 2003. He was also an Assistant Professor with the Department of Electrical Engineering at the University of New Orleans from 2003 to 2006. He was among the recipients of the DARPA/MTO young faculty award in 2007, the IBM faculty fellowship award in 2004, and the central Bell Labs annual excellence and teamwork award in 2002. His research interests are in the areas of circuit design and system architecture, with emphasis on reconfigurable computing, wireless communications, and networked embedded systems.



Zhongfeng Wang (M'00-SM'05) received B.S. and M.S. degrees, both from the Department of Automation at Tsinghua University, Beijing, China. He obtained the Ph.D. degree from the Department of Electrical and Computer Engineering at the University of Minnesota, Minneapolis in 2000. In the past, he has worked for Beijing Hua-hai New Technology Development Co., Beijing, CHINA, Morphics Technology Inc. (now a part of Infineon Technology), Campbell, CA, USA, National Semiconductor Co., Longmont, CO, and School of EECS at Oregon State University, Corvallis, OR. Since 2007, he has been with Broadcom Corp., Irvine, CA, as Senior Principle Scientist.

Wang was the recipient of the Best Student Paper award (1st prize) at the 1999 IEEE Workshop on Signal Processing Systems (SiPS'99) and the IEEE Circuits and Systems Society VLSI Transactions Best Paper Award in 2007. He has served as Associate Editor for the IEEE Trans. on Circuits and Systems: I (TCAS-I), TCAS-II and IEEE Trans. on VLSI Systems. He has also served in numerous technical committees in the IEEE Circuits and Systems society. His current research interests are in the area of Low Power/High Speed VLSI Design for Digital Communication Systems.