A Parallel-Layered Belief-Propagation Decoder for Non-layered LDPC Codes

Kun Guo, Yong Hei and Shushan Qiao
Asic and System Department, Institute of Microelectronics of Chinese Academy of Sciences, Beijing, China
Email: {guokun, heiyong, qiaoshushan}@ime.ac.cn

***Abstract*—In this paper, we proposed a Parallel-Layered Belief-Propagation (PLBP) algorithm first, which makes a breakthrough in utilizing the layered decoding algorithm on the "non-layered" quasi-cyclic (QC) LDPC codes, whose column weights are higher than one within layers. Our proposed PLBP algorithm not only achieves a better error performance, but also requires almost 50% less iterations, compared with the original flooding algorithm. Then we propose a low-power partial parallel decoder architecture based on the PLBP algorithm. The PLBP decoder architecture requires less area and energy efficiency than other existing decoders. As a case study, a multi-rate 9216-bit LDPC decoder is implemented in SMIC 0.13μm 1P6M CMOS technology. The decoder dissipates an average power of 87mW with 10 iterations at a clock frequency of 83.3 MHz. The chip core size is 7.59 mm^2, and the die area occupies 10.82 mm^2.**

***Index Terms*—Low-density parity-check codes, quasi-cyclic codes, layered decoding, parallel architecture, non-layered codes, VLSI**

I. INTRODUCTION

Low-density parity-check (LDPC) codes, are a kind of linear block codes, which were first introduced by Gallager in 1962[1], and were rediscovered by MacKay[2] in 1996. With the improving technology, LDPC codes and their efficient implementations have been receiving a lot of attention due to their excellent error-correcting performance closing to the Shannon limit. Hence, LDPC codes have been widely employed in most wireless communication systems, such as IEEE 802.11n[3], 802.16e[4], DVB-S2[5] and Chinese Mobile Multimedia Broadcasting(CMMB) [3] standard.

With the inherent parallelism in the decoding process, various decoder architectures (fully parallel [7], partially parallel [8], and completely sequentially [9]) have been proposed. Taking both throughput and hardware cost into consideration, the partially parallel method is the best choice for most applications. Recently, a growing attention has been given to different schedules of the elaborations of the Belief Propagation (BP) algorithm to speed up the decoder convergence with a smaller number of iterations. So far, there are two main decoding algorithms for the LDPC codes. The original flooding (or TPMP[10]) algorithm updates all the check-to-variable (CTV) messages first, then all the variable-to-check (VTC) messages in any iteration. Therefore, the estimation of all the variable nodes are updated only once with all the neighboring check-to-variable in one iteration. On the other hand, the layered algorithm [11] breaks up one iteration into several sub-iterations, called "layers". The CTV and VTC messages will be updated first in one layer, then the latest messages are used in the next layer, and so on, layer by layer. With more updated estimates, the layered schedule achieves faster convergent rate, and better error performance. However, the rows of parity check matrix can be grouped as a layer should have the feature that the layer column weight is one at most. So we can refer to the codes like these as "layered" codes, in opposition to "non-layered" codes, whose column weights are higher than one within layers.

As the original layered decoding algorithm can cause conflicts when used on the "non-layered" LDPC codes straightforwardly, two different strategies are proposed in [12]. But both the strategies are the approximations of the original layered algorithm and only work well with a small number of the overlapped blocks. When the number increases, the error performance can get worse and more iterations are required for convergence. Another solution based on the computation of an extra variation is presented in [13]. Such computation allows concurrent updates but requires more memory access or faster clock frequency. Instead of improving the message updating formulas, [14] proposed a reordering mechanism for the parity check matrix to reduce the number of conflicts. However, this approach lowers the level of parallelism and still can not achieve the same error performances as the original layered algorithm.

In this paper, we proposed a parallel-layered belief-propagation (PLBP) algorithm. The algorithm avoids the conflicts carefully by building direct paths among different layers for every code bit, when all the layers are processed in a parallel way. With such paths, every variable is able to be updated layer by layer. As a result, the PLBP algorithm can get the same error performance and the same convergent rate as the original layered algorithm, no matter how many conflicts appears. Moreover, a low-power PLBP architecture is proposed and implemented for 9216-bit LDPC codes in CMMB system as an example using 0.13μm CMOS technology.

The remainder of this paper is organized as follow. In Section II, we introduce the two popular decoding algorithms, and the "non-layered" QC-LDPC codes. The proposed PLBP algorithm and its architecture are

Manuscript received August 27, 2009; revised December 20, 2009; accepted January 15, 2010.

doi:10.4304/jcm.5.5.400-408

demonstrated in Section III and Section IV, respectively. The chip implementation is shown in Section V and the conclusions are presented in Section VI.

II. LOW-DENSITY PARITY-CHECK (LDPC) CODES

The decoding of the LDPC codes is an iterating process to refine the Log-Likelihood Ratio (LLR) of the received bits in the codeword, defined as (1), with x and y are the original codeword and its observation respectively. When decoding, the LLRs are propagated and updated between the variable nodes and the check nodes in the Tanner graph [13], until all the check equations are satisfied.

$$LLR(x)=\log\left(\frac{p(x\mid y=0)}{p(x\mid y=1)}\right) \tag{1}$$

A. Flooding decoding schedule

Flooding decoding algorithm is the most common message-propagating algorithm. In each iteration, the updating floods from one side of the Tanner graph (check nodes) to the other side (variable nodes). As a result, each variable node is updated only once, based on the message from all the check nodes connected to it.

At the *k*-th iteration, let $r_{mn}(k)$ and $q_{nm}(k)$ denote the message from check node *m* to variable node *n* and the message from variable node *n* to check node *m*, respectively. Assume N(m) is the set of variable nodes connecting to the check node m and M(n) is the set of check nodes connecting to the variable node n in the Tanner graph. One iteration is composed of two successive steps as follow.

Check node updating is the process to update $r_{mn}(k)$ separately on signs and magnitudes:

$$\operatorname{sgn}\left(r_{mn}(k)\right)=\prod_{n'\in N(m)\backslash n}\operatorname{sgn}\left(q_{n'm}(k-1)\right) \tag{2}$$

$$\left|r_{mn}(k)\right|=\prod_{n'\in N(m)\backslash n}\Psi\left(\sum_{n'\in N(m)\backslash n}\Psi\left(\left|q_{n'm}(k-1)\right|\right)\right) \tag{3}$$

where

$$\Psi(x)=\log\left(e^{x}+1/e^{x}-1\right). \tag{4}$$

There are several low-complexity approximations of (4). In this paper, we use the Normalized Min-Sum algorithm with a normalized factor α of 0.8 [16].

$$\left|r_{mn}(k)\right|=\min_{n'\in N(m)\backslash n}\alpha\times\left(\left|q_{n'm}(k-1)\right|\right) \tag{5}$$

Variable node updating is to generate $q_{nm}(k)$ by summing the check message $r_{mn}(k)$ from its neighboring check nodes and the prior message λ_n from the channel.

$$q_{nm}(k)=\lambda_n+\sum_{m'\in M(n)\backslash m}r_{m'n}(k) \tag{6}$$

At the same time, a refined estimation on the transmitted bit $\Lambda_n(k)$ is computed, which is also referred to as the soft output:

$$\Lambda_n(k)=\lambda_n+\sum_{m\in M(n)}r_{mn}(k) \tag{7}$$

B. Layered decoding schedule

One disadvantage of the Flooding schedule is their slow convergence. To improve the convergence, the layered and shuffled decoding schedules are introduced, which splits parity check matrix horizontally or vertically into several sub matrices, called layers. Therefore, one iteration is broken up into the sequentially sub-iterations of these layers. With more than once updating of the variable nodes, both the layered and shuffled decoding schedules require up to 50% fewer iterations to converge and achieve better error performance than the original flooding schedule. As the horizontal layered decoding is more suitable for the implementation of the check nodes unit, it is more popular for practical implementations.

Let $\Lambda_n^{(p)}(k)$ denote soft output of variable node *n* at the *p*-th sub-iteration of the *k*-th iteration. As the check nodes updating is followed by the variable nodes updating in every sub-iteration, $\Lambda_n^{(p)}(k)$ is updated in every step by (8) and the value at the last step (the *pmax*-th step) represents the soft output of the *k*-th iteration.

$$\Lambda_n^{(p)}(k)=\lambda_n+\sum_{m'\in M(n)\backslash m}r_{m'n}(k-1)+r_{mn}(k) \tag{8}$$

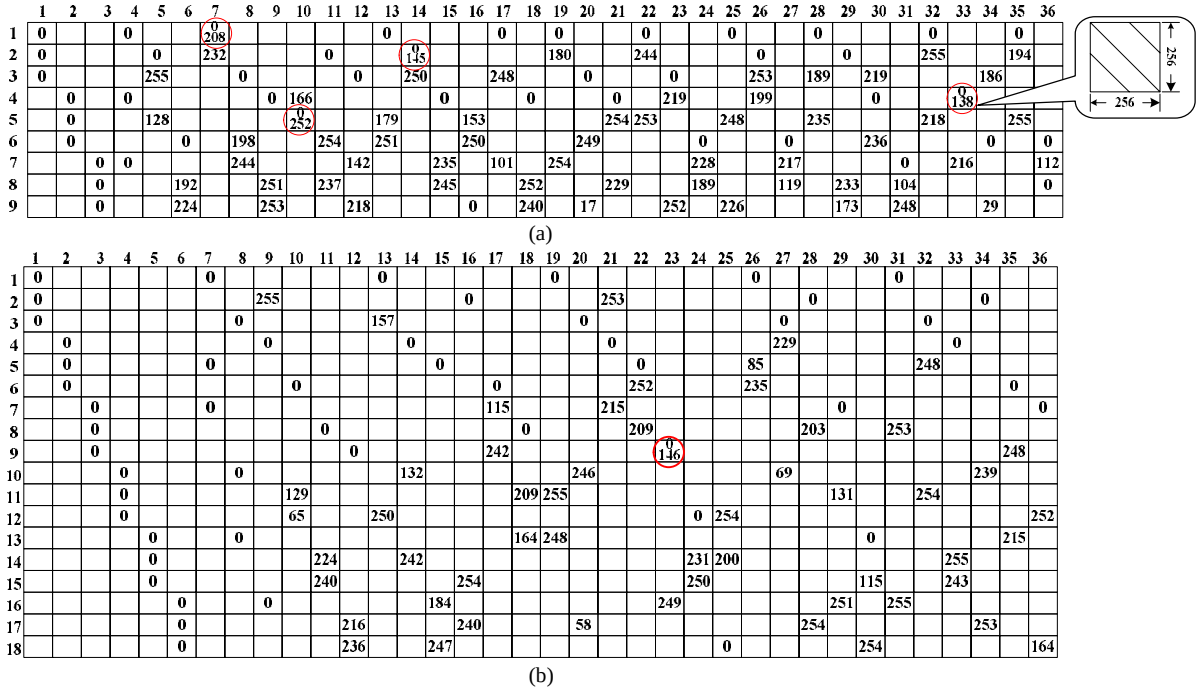
And the hard decision $\hat{X}_n(k)$ can be made as follow:

$$\hat{X}_n(k)=\begin{cases}0, & \Lambda_n^{(pmax)}(k)<0\\ 1, & \Lambda_n^{(pmax)}(k)<0\end{cases}. \tag{9}$$

C. Non-layered LDPC codes

Non-layered LDPC codes are a special kind of QC LDPC codes, whose elements in the base parity check matrices can be expanded as several p×p cyclic-shifted identity matrices overlapped with each other. This is the case of the codes used in CMMB system, as show in Fig.1 (a) and (b), and all the overlapped blocks are marked with circles. Such overlapping can give rise to the decoding conflicts, when the layered algorithm and the partial parallel architecture are both employed.

To figure out the conflict, let C1 and C2 denote the overlapped blocks. As mentioned in section II, the essential reason that the layered algorithm has a faster convergence is that the latest message is used by the next layer when decoding layer by layer in a single iteration. As the partial parallel architecture updates all the check nodes and variable nodes within a layer in a parallel way, C1 and C2 have to be processed at the same time and neither of them is able to benefit from the latest updated messages of each other. So such LDPC codes which do not allow the straightforward implementation of the original layered decoding, are referred to as "non-layered" codes. The most popular "non-layered" LDPC codes are the ones used in DVB-S2 system and CMMB system.



(a)

(b)

Figure 1. Parity check matrices of LDPC codes in CMMB system: (a) 3/4-rate matrix, (b) 1/2-rate matrix

III. THE PROPOSED PLBP ALGORITHM

In this section, we propose a parallel-layered belief-propagation (PLBP) algorithm for the non-layered LDPC codes. The decoding of this algorithm is quite different from the original layered algorithm, for it uses parallel updating among all the layers and serial updating within each layer. With such a decoding schedule, a variable node in different layers is updated at different time, which means that the message is able to be updated layer by layer. As a result, the PLBP algorithm can be employed by the "non-layered" LDPC codes, so as to improve their error performance and convergence speed just like the original layered decoding algorithm.

A. The PLBP algorithm

Let us take the LDPC codes for CMMB systems as an example to explain the detail of the PLBP algorithm. Fig. 2 illustrates message passing routes in the PLBP algorithm using the 33rd column of the parity check matrix in Fig. 1 (a). We denote the overlapped sub-matrices as Layer 4a and Layer 4b respectively. As the updating is processed sequentially within a layer, one iteration is broken up into 256 steps. The operation sequence of these three layers is described as following:

1) At step 0, all the layers update from the first row (Row 0). Column 0 in Layer 4a, Column 138 in Layer 4b and Column 216 in Layer 7 process the updating at the same time, according to (5),(6) and (8), as shown in Fig. 2(a).
2) At step 40, all the three layers are processed at Row 40, corresponding to Column 40 (in Layer 4a), Column 178 (in Layer 4b), Column 0 (in Layer 7). As Column 0 has already been updated in Layer 4a at the step 0, Layer 7 can use the latest message from Layer 4a, as shown in Fig. 2(b).
3) At step 78, all the three layers are processed at Row 78, Column 78 (in Layer 4a), Column 216 (in Layer 4b) and Column 38 (in Layer 7). Layer 4b can use the latest message of Column 216 from Layer 7 which has already been updated at step 0, as shown in Fig. 2(c).
4) At step 118, all the three layers are processed at Row 118, corresponding to Column 118 (in Layer 4a), Column 0 (in Layer 4b), Column 78 (in Layer 7). As Column 0 has already been updated in Layer 4a then in Layer 7, Layer 4b can get the latest message from Layer 7. So far, Column 0 has completed its iteration with three updating (Layer 4a → Layer 7→ Layer 4b), and stored the hard decisions in Layer 4b, as shown in Fig. 2(d).
5) As the steps go on, Column 216 and Column 138 also are able to get their latest messages from other layers, which have been updated at the earlier steps, as shown in Fig. 2(e)~(g).
6) At step 255, all the columns have finished their three layered-updating, but the updating sequences are different. Since the latest messages of all the columns are in different layers, as shown in Fig.2(h), the hard decisions of these columns in this iteration are distributed in different layers. Such distribution of the 33rd block column is shown is Table I.

The key point in the PLBP algorithm is that each variable node is updated in different layers at different time. Such differences make it possible to exchange the messages among different layers. However, there are several sub-matrices in the same block columns with the same shifting factors, which implies that all the three layers of one variable node are updated at the same time. To solve the problem, we can start the iteration with different rows for different layers, which is equivalent to

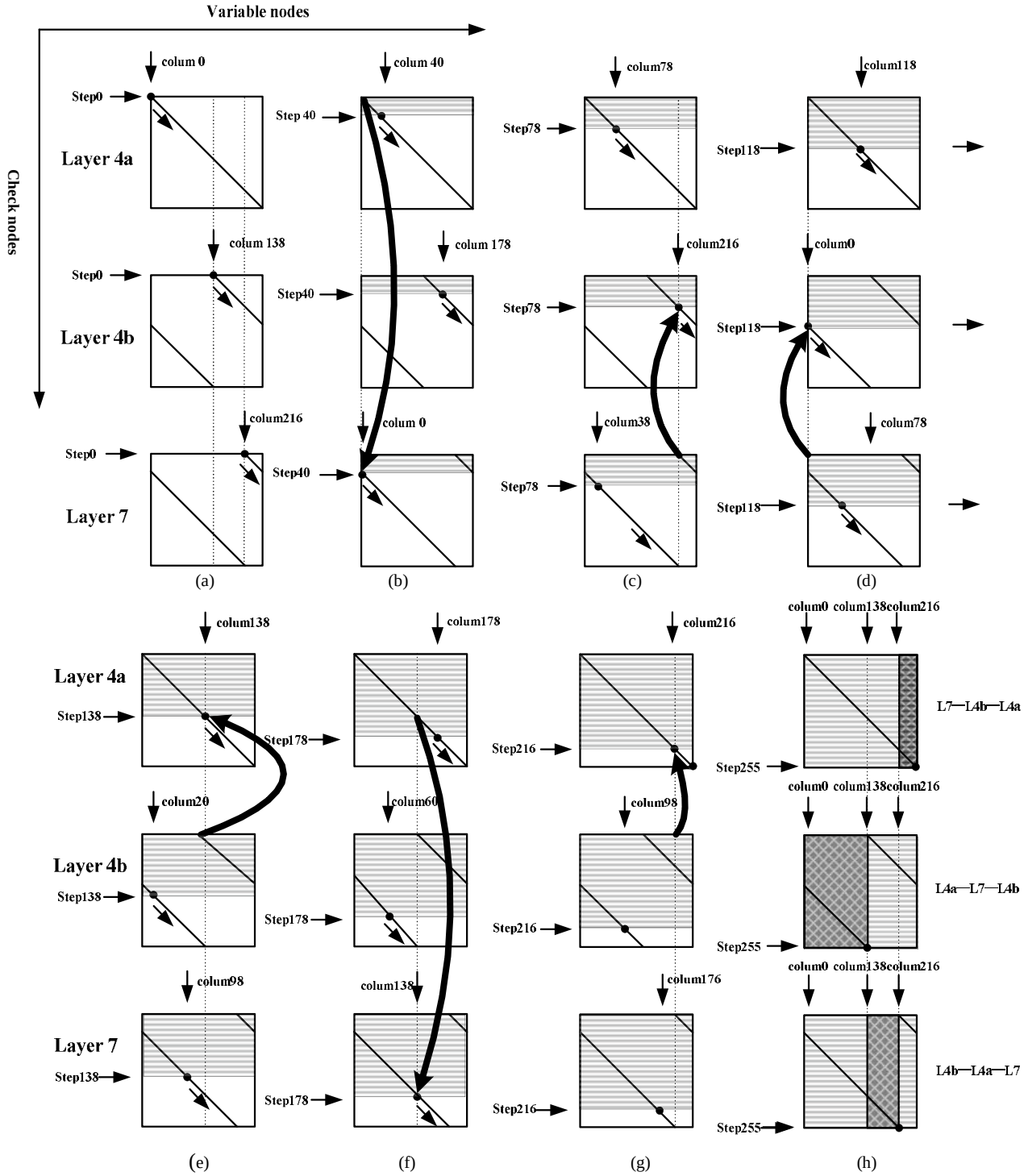


Figure 2. The decoding schedule of the 33th block column

adding an offset to the shifting factor of each layer, as show in Fig. 3(a) and (b). The offset values are carefully selected, so that the difference of the modified shifting factors between any two layers is at least four.

B. Simulation result

Both the flooding and PLBP algorithms are simulated using the codes in Fig. 1, with 6-bit quantized LLRs in BPSK modulation mode over AWGN channel. The check nodes updating algorithm is the "Normalized Min-Sum" for both algorithms.

The BER performance comparison is plotted in Fig. 4 (a) and (c) for 3/4-rate code and 1/2-rate code, respectively. The maximum iteration number is set to 5,

TABLE I.
UPDATING SEQUENCE AND HARD DECISION DISTRIBUTION OF THE 33TH BLOCK COLUMN

	Col.0~137	Col.138~215	Col.216~255
Sequence	L4a-L7-L4b	L4b-L4a-L7	L7-L4b-L4a
Hard Decision	L4b	L7	L4a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	
+190	1	190			190			192 142						190				190		190			190			190			190				190			190		+190
+66	2	66				66		42				66			66 211					246			246				66			66			65			4		+66
+194	3	194				193			194				194		188			186			194			194			191		127		157			124			+94
+30	4		30		30					30	196					30			30			30		249			229				30			30 168			+30
+72	5		72			200					72 68			251			225					70	78			64			51			34			71		+72
+5	6		5				5		203			3		0			255				254				5			5			241			5		5	+5
+80	7			80	80			68					222			59		181		78					52			41				80		40		192	+80
+213	8			213			149			208		194				202			209			186			146			76		190		61				213	+213
+104	9			104			72			101			66				104		88		121			100		74				21		96			133			+104

(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	
+215	1	215						215						215						215							215					215							+215
+87	2	87							87								168					219						168							168			+87
+219	3	219								219				224							87						87					87							+219
+90	4		90					90							219							165					192						219						+90
+168	5		168							167						90							90				175					82							+168
+40	6		40								40							40					36				19									40			+40
+221	7			221				221										80				180							221								221	+221
+93	8			93							93								93				46						40			90							+93
+214	9			214								214						200						214 104													206			+214
+240	10				240				240						116						230						53								223			+240
+11	11				11						129								220	10									142				9						+11
+115	12				115						65			109											115	113												111	+115
+120	13					120			120										28	112										120						79			+120
+50	14					50					18			36											25	250							49						+50
+187	15					187					171					185									181						46			174					+187
+43	16						43			43					227									36						38		42							+43
+70	17						70					30				54					128								68						67				+70
+176	18						176					156			167											176					174							84		+176

(b)

Figure 3. The parity check matrices with offset for the CMMB system: (a) 3/4-rate matrix, (b) 1/2-rate matrix

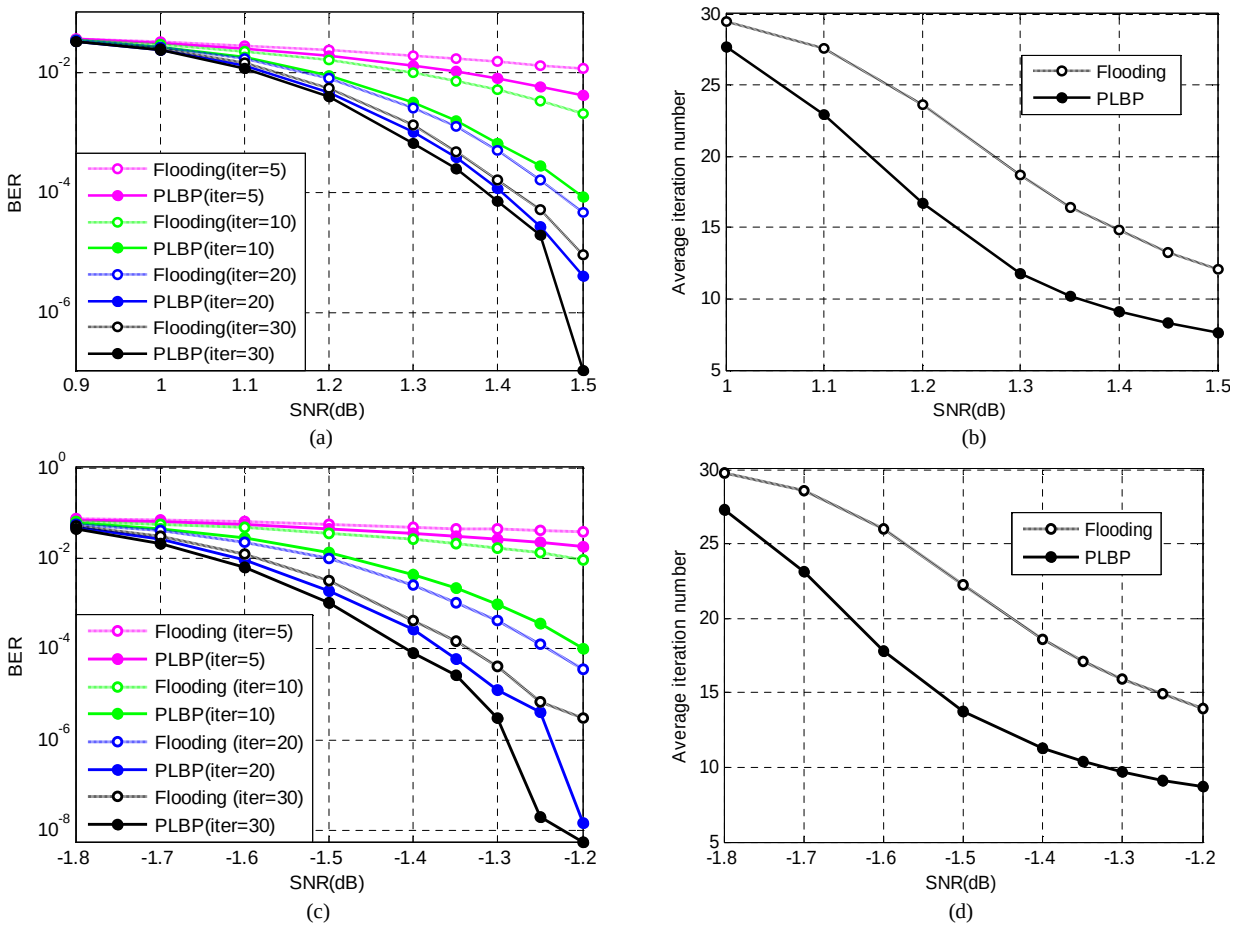


Figure 4. Error performance and convergence speed of PLBP algorithm vs. Flooding algorithm: (a) and (b) for 3/4-rate codes of CMMB system, (c) and (d) for 1/2-rate codes of CMMB system.

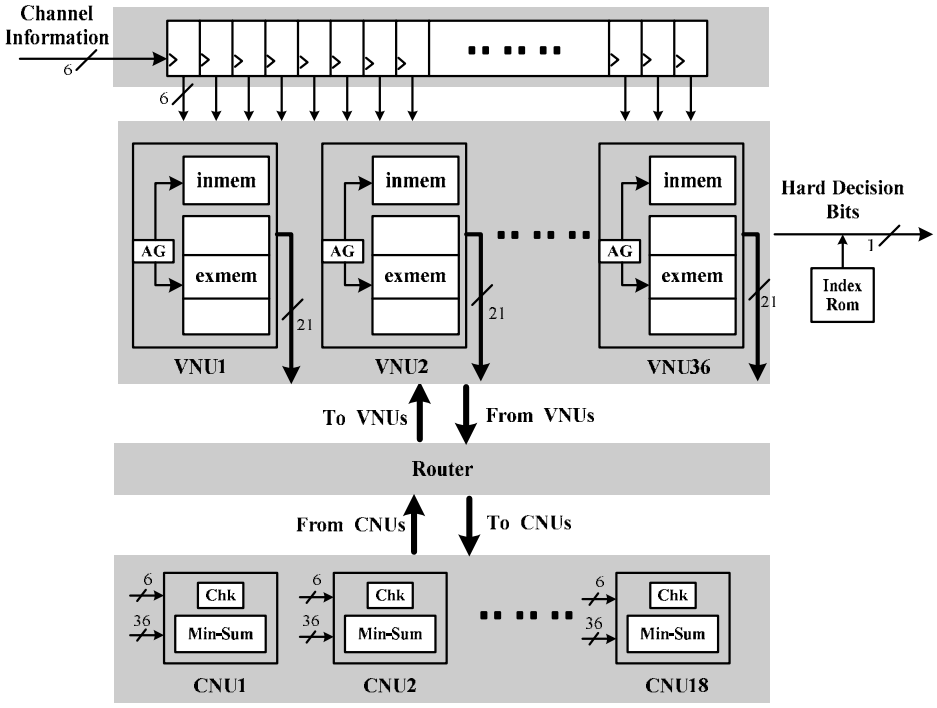


Figure 5. The architecture of the PLBP decoder

10, 20 and 30. The figure demonstrates that the proposed the PLBP algorithm achieves much better BER at the same SNR for the same number of iterations. In particular, with 10 iterations, 1/2-rate code at -1.2dB (Eb/No = 1.8dB) and 3/4-rate code at 1.5dB (Eb/No = 2.75dB), the PLBP algorithm provides an order of magnitude improvement in BER.

The average numbers of iterations required to converge using both algorithms are plotted in Fig. 4(b) and (d). As shown, the PLBP algorithm requires significantly less iterations to converge. Where in some cases it requires close to half the number of iterations to converge compared with the flooding algorithm. Hence, the decoder power consumption can be reduced when the PLBP algorithm achieves the same performance as the flooding algorithm.

IV. THE PROPOSED ARCHITECTURE

The overall architecture of the PLBP decoder for the LDPC codes in CMMB system is shown in Fig. 5. It mainly contains two edge node processor clusters. The first one is the variable node units (VNUs), which generate the sum of extrinsic messages for the neighboring check nodes, and the second one is check node units (CNUs) which check the hard decision and generate the check message for the VNUs. There are 36 VNUs and 18 CNUs in total. In each VNU, there are one block of Inmem memory and 3 blocks of Exmem memory. The 256×6 single-port Inmem block is used to store the intrinsic message from the channel, while the 256×6 dual-port Exmem blocks are used to store the extrinsic message from the CNUs. Each memory block associated with an address generator (AG) to provide reading and writing address. In particular, the code bits of the CMMB system are not transmitted in its natural order as encoded [3], therefore, a ROM to reorder the output bits is needed at the receiver, as shown in Fig. 5.

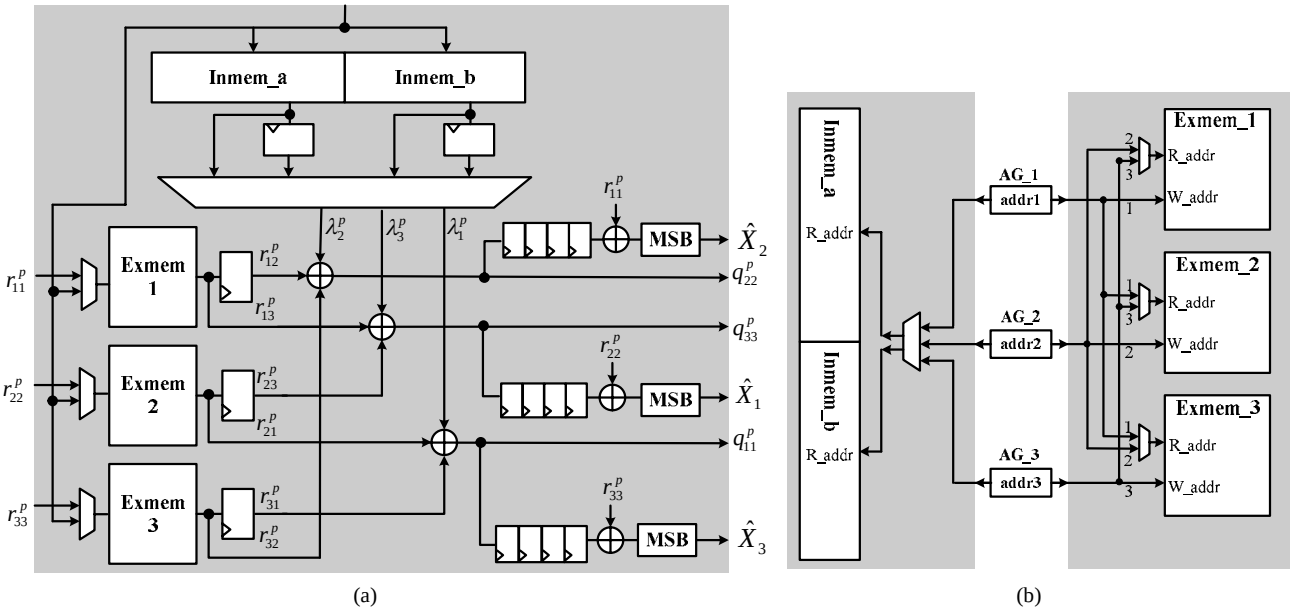
A. The VNU block

The VNU architecture is shown in Fig. 6(a). Each Exmem blocks storage the extrinsic messages for one of the three layers in the same block column. In this architecture, it takes two clock cycles to complete one step mentioned in Section III. At each step, every Exmem block reads out two extrinsic messages for the other two layers, and writes in one new updated message for its own layer from the corresponding check nodes, on the other hand, the Inmem block composed of two 128×6 single-port memory, processes three reading operations providing an intrinsic messages for each layer. The three AG generate three different addresses for the three layers and control the reading/writing operations for the memory blocks, as shown in Fig. 6(b). The address for the memories at step *p* can be calculated as

$$addr_i = (p + s_i) \bmod 256 \quad (i = 1,2,3) \qquad (10)$$

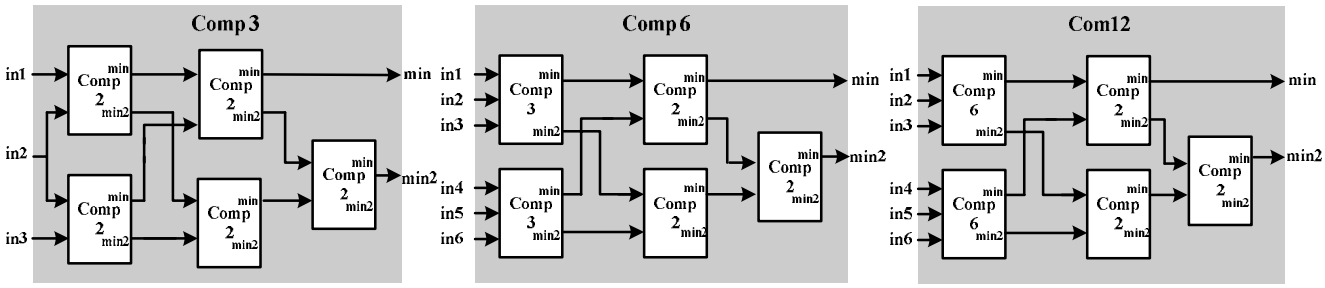
where s_i represents the shifting factor of the *i*-th cyclic-shifted identity matrix from the top of the block column.

Let v_i^p (i=1, 2, 3; p=0, 1, …, 255) denote the variable node which is going to be updated in the *i*-th layer at the *p*-th step and λ_i^p denote its corresponding intrinsic message. Let r_{ji}^p (i, j=1, 2, 3, $i \neq j$) represent the extrinsic message of the v_i^p from the *j*-th Exmem. The message of to the CNUs is denoted as q_{ii}^p, which



(a) (b)

Figure 6. The architecture of VNU block: (a) the data path of the VNU memory blocks, (b) the address generator for the memory block



(a) (b) (c)

Figure 7. The comparison part of the CNU block: (a) 3-input, (b) 6-input, (c) 12-input

can be computed by (6), and the newly updated message of v_i^p from the CNUs is denoted as r_{ii}^p, which can be computed by (5). Four or three clock cycles (for different code rates) after q_{ii}^p generated, r_{ii}^p is available for the computation of $\hat{X}_i$ using (9), so q_{ii}^p is delayed for four or three clock cycles as shown in Fig. 6(a).

B. The CNU block

To perform the check node updating in Min-Sum Algorithm, we should search for the minimum and second minimum among the receive data. As demonstrated in Fig. 7(a), the 3-input comparison unit (Comp3) can be built by the 2-input comparison unit (Comp2). Base on the construction of Comp3 and Comp2, the design of 6-input comparison unit (Comp6) and 12-input comparison unit (Com12) can be realized in hierarchal method[17], as shown in Fig. 7(b) and Fig. 7(c). It takes 3 clock cycles for 1/2-rate codes and 4 clock cycles for 3/4-rate codes to complete the CNU computation. As shown in Fig. 3, the difference of the shifting factor among different layers in a same block column is more than 4, so that the messages for the next layer have already been prepared in the Exmem block.

V. IMPLEMENTATION RESULTS

In order to evaluate the performance of the proposed PLBP architecture, we implement the multi-rate LDPC decoder for CMMB system in SMIC 0.13μm 1P6M CMOS technology.

The storage elements are implemented by 144 memory banks, which consist of 36 single-port and 108 dual-port rams. Each of the memory bank has 256 entries, which one entry consists of 6-bit data. To reduce the routing complexity, we use the checkboard [17] layout scheme to as shown in Fig. 8, where "+" represents the dual-port memory, "-" represents the single-port memory and "O" represents the ROM, which is used to storage the reordering index list as mentioned in Section IV.

In order to compare with other state of art, the normalized area and power are derived as follows:

$$\text{Normalize Area} = \frac{\text{Area}}{\text{code\_length}^2 \times (1-\text{code\_rate}) \times \text{technology}^2} \quad (11)$$

$$\text{Normalize Power} = \frac{\text{Power}}{(\text{core\_power\_supply})^2} \quad (12)$$

TABLE II.

OVERALL COMPARISON BETWEEN THE PROPOSED CMMB LDPC DECODER AND THE EXISTING LDPC DECODERS

	JSSC'06 [18]	JSSC'08 [19]	JSSC'02 [20]	LDPC decoder IP [21]	This work
Code Length	2304	576~2304	1024	576~2304	9216
Gates	220(logic)	420	1750k	NA	900k
Parallelism	Partial	Partial	Fully	Partial	Partial
Frequency(MHz)	125	150	64	333	83.3
Iterations	10	20	64	10	10
Throughput (Mbps)	640	105	1000	619	135
Area (mm^2)	14.3	6.25	52.5	3.84	10.82 (with index Rom)
Normalized Area（10^{-3} mm^2）	0.166	0.291	3.9	0.045	0.015
Power (mW)	787	264	690	NA	87
Energy Efficiency (pJ/Bit/Iter)	123	125	10.9	NA	118
Normalized Power	243	264	307	NA	60
Technology	0.18 μm,1.8 V	90nm,1.0V	0.16μm, 1.5V	0.18 μm,1.8V	0.13μm,1.2V
Rate	8/16:1/16:14/16	1/2,2/3,3/4,5/6	1/2	1/2,2/3,3/4,5/6	1/2, 3/4

Table II shows the decoder implementation results compared with other existing QC-LDPC decoders. Note that the proposed LDPC decoder is much smaller than the other research works in normalized area. Moreover, the normalized power and Energy Efficiency are also smaller than other partial parallel decoder chips. In other words, the propose decoder with superior characteristics of low area cost and low power dissipation is quite suitable for the wireless communication systems, especially CMMB and DVB system.

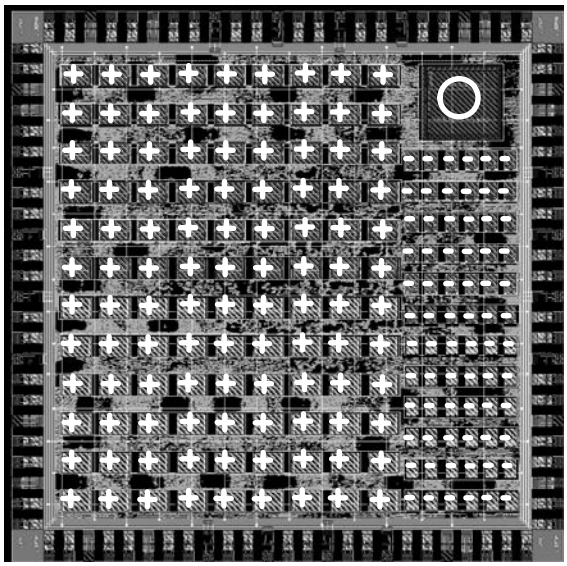


Figure 8. The layout photo of the proposed decoder

VI. CONCLUSION

As layered algorithm cannot be used in the decoding of the "non-layered" LDPC codes, we proposed a parallel-layered belief-propagation (PLBP) algorithm and its partial parallel architecture in this paper. The PLBP algorithm establishes a path for messages propagating among different layers, which makes every code bit updated layer by layer without any loss in error performance and convergence speed, compared with the original layered algorithm. Additionally, the proposed PLBP architecture is implemented for the codes in CMMB system as a study case, using SMIC 0.13μm 1P6M CMOS technology. The die area is only 10.82mm^2 and the power consumption is 87mW at 83.3MHz, which are both smaller than the other designs in the normalized way. In summary, the proposed PLBP algorithm and the corresponding architecture are very suitable for low-power communication systems employing the "non-layered" QC LDPC codes.

REFERENCES

[1] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 7, pp. 21–28, Jan. 1962.

[2] D. J. C. MacKay," Good error-correctiong codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp.399–431, Jan. 1999.

[3] IEEE P802.11n/D1.05 October 2006, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications – Enhancements for Higher Throughput (Draft).

[4] IEEE 802.16e: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, IEEE, 2004.

[5] European Telecommunications Standards Institute (ETSI). Digital Video Broadcasting (DVB) Second generation, framing structure for broadband satellite applications; EN 302 307 V1.1.1. 2005.

[6] GY-T200.1-2006, "Mobile Multimedia Broadcasting Part 1: Framing Structure Channel Coding and Modulation for Broadcasting Channel", 2006.

[7] A. J. Blanksby and C. J. 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, pp. 404–412, Mar. 2002.

[8] C. J. Howland and A. J. Blanksby, "Parallel decoding architectures for low density parity check codes," in *Proc. IEEE ISCAS*, vol. 4, pp. 742–745, May 2001.

[9] Z. Cui and Z. Wang, "Area-efficient parallel decoder architecture for high rate QC-LDPC codes," in *Proc. IEEE ISCAS*, pp. 5107–5110, May 2006.

[10] M. M. Mansour and N. R. Shanbhag, "high-throughput LDPC decoders," *IEEE Trans.Very Large Scale Integration Systems*, vol.11, no.6, Dec.2003

[11] M. M. Mansour and N. R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE J. Solid-State Circuits*, vol. 41, pp. 684–698, Mar. 2006.

[12] Massimo Rovini, Francesco Rossi, Pasqale Ciao, Nicola L'Inslata, Luca Fanucci."Layered Decoding of Non-Layered LDPC codes" in *Proc.the 9th Euromicro comference on Digital System Design*, pp.537-544, Sep, 2006.

[13] E. Boutillon and F. Guilloud, "LDPC decoder, corresponding method, system and computer program," US patent 7,174,495 B2, Feb. 2007.

[14] C. Marchand, J.-B. Dor´e, L. Conde-Canencia, and E. Boutillon, "Conflict resolution by matrix reordering for DVB-T2 LDPC decoders," in *Global Telecommunications Conference*. Honolulu, USA, Oct. 2009.

[15] R.M.Tanner, "A recursive arpproach to low complexity codes," *IEEE Trans. Inform. Theory*, IT-27, pp.533-547, September 1981.

[16] 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.

[17] Xin-Yu Shih,Cheng-Zhou Zhan, Cheng-Hung Lin, and AN-Yeu(Andy) Wu, "An 8.29mm^2 52mW Multi-Mode LDPC Decoder Design for Mobile WiMAX System in 0.13μm CMOS Process," *IEEE J. Sold-State Circuits*, vol43, No.3, pp.672-683, March 2008.

[18] M. M. Mansour and N. R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 634–698, Mar. 2006.

[19] Chih-Hao Liu, Shau-Wei Chen, Chil-Lung Chen,Hsie-Chia Chang, Chen-Yi Lee and et al. "An LDPC decoder chip based on self-routing network for IEEE 802.16e application," *IEEE J. Solid-State Circuits*, vol.43, no.3, pp.684-693, Mar.2006.

[20] A. J. Blanksby and C. J. 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, pp. 404–412, Mar. 2002.

[21] T. Brack, M. Alles, F. Kienle, and N. When, "A synthesizable IP core for WIMAX 802.16E LDPC code decoding," in *Proc. IEEE 17th Int. Symp. Personal, Indoor and Mobile Radio Communications*, pp. 1–5, Sep. 2006.

Kun Guo was born in 1982 in Dalian, China. She received the B.S. degrees in Electrical and Electronic Engineering from Tianjin University, Tianjin, China, in 2005.

She is currently pursuing the Ph.D. degree in the Institute of Microelectronics Chinese Academy of Sciences (IMECAS), Beijing, China. Her research interests include wireless communication and low power design.

Yong Hei was born in 1974 in Heibei, China. He received B.S. and M.S. degrees from Beijing Broadcasting Institute, Beijing, China, in 1996 and 1999, respectively.

He received Ph.D. degree in 2002 and is a professor in the Institute of Microelectronics Chinese Academy of Sciences. Presently, he is senior visiting scholar in University of California, Los Angeles. His research interesting includes DSP and VLSI application, reconfigurable processor, wireless communication and low power design methodology. He has published over 40 technical papers in referred conference and journals.

Shushan Qiao was born in 1981 in Shanxi, China. He received B.S. degree in Automation from Hunan University, Changsha, China in 2003. He received the Ph.D. degree in IMECAS, in 2007.

At present, he is an assistant professor of IMECAS. His scope of activity comprises wireless communication, DSP and VLSI application. He has published over 10 technical papers in refereed conference and journals.