Design of a Quasi-Cyclic LDPC Decoder Using Generic Data Packing Scheme

Jinlei Chen, Yan Zhang and Xu Wang
Key Laboratory of Network Oriented Intelligent Computation
Shenzhen Graduate School, Harbin Institute of Technology
HIT Campus of Shenzhen University Town, Shenzhen, 518055
China
ianzh@foxmail.com http://www.hitsz.edu.cn

Abstract: -This paper proposed a generic data packing scheme for quasi-cyclic LDPC codes decoder which has two advantages. Firstly, in this scheme no shift network is needed, it greatly reduces the area of the LDPC decoder. Secondly, in this scheme, all the messages in a row can be accessed from memories in one cycle, it increase the throughput of LDPC decoder. A LDPC decoder architecture using the data packing scheme is proposed. Based on this architecture, a multi rate decoder that supports all code rates and all code lengths in IEEE 802.16e standard is implemented using a SMIC 0.18μm CMOS technology. Compared to the existing LDPC decoder which uses the same parallel check node process unit, the proposed decoder has nearly the same throughput and achieves saving of 16% in area. Synthesis results also show that the area and throughput of the proposed decoder increases linearly with the number of process units.

Key-Words: - LDPC decoder, message packaging, alignment unit, layered, parallel processing, QC codes.

1 Introduction

Low density parity check (LDPC) code was first proposed by Gallager in 1960 [1], and it was rediscovered by MacKay and Neal in 1996 [2]. Because of its excellent decoding performance, LDPC code has been widely adopted by many communication systems, such as wireless local area network (802.11n) [3], digital video broadcasting second generation (DVB-S2) [4] and world interoperability for microwave access (802.16e) [5][6]. Punctured LDPC codes used in coherent optical OFDM systems is also studied in [7].

Belief-propagation (BP), or sum-product [2], is one of the best algorithms in LDPC decoding, but it is not suitable for hardware implementation because the exponent computation will result in huge hardware complexity. Min-sum algorithm [8], which uses only comparison operation in check node process, greatly simplifies the decoding process with acceptable decoding performance loss. Moreover, modified min-sum algorithm [9-11], which uses normalized factor or offset factor in the min-sum algorithm, can improve the decoding performance of min-sum algorithm.

Layered decoding method was proposed in [12]. In layered decoding both check node and variable node update concurrently. It can reduce the decoding latency by about 50% without performance loss. In [13], a fast-convergence algorithm using layered decoding was proposed and about 1/6 iteration numbers is reduced for LDPC codes used in DVB-S2.

A parallel LDPC decoder was proposed in [14]. However, because of the typically large code length (more than 1 thousand bits) and a large number of complex routings between the processing units, fully parallel decoder suffers from high implementation complexity. For a 1024-b, rate-1/2 LDPC code, the total gate count of the decoder was 1750K [14].

Quasi-Cyclic (QC) LDPC code [15] can reduce the hardware implementation and achieve almost the same decoding performance as that of computer generated random codes. QC-LDPC code is constructed by a base matrix and many cyclic shifted identity matrixes, and its cyclic symmetry results in regular wiring and configurable structure.

Partially parallel decoder can achieve appropriate tradeoff between decoding throughput and implementation complexity. In partially parallel decoder, the messages in the same row or in the same column can be processed serially or in parallel. The serial partially parallel decoder has a limited decoding throughput because the messages in a row are processed in multi cycles. Most of the previously known partially parallel decoders use serial process unit [16-19], but they do not provide a generic architecture for LDPC decoder, which consist of any degree check node. In [20], a generic and scalable parallel check node process unit was proposed, and it supports both irregular and regular LDPC codes. The throughput of this architecture is 1.2Gb/s, by assuming 5 iteration and 96 min-sum process units. However, in this architecture, only messages corresponding to two neighbors of check node are read simultaneously from the RAM, so all the process units still need to wait several clocks to update all the messages in one row or in one column.

Packing multiple messages in a single memory word can take advantage of the configurable data-width and depth of embedded memory in an FPGA to maximize the decoding throughput. Until now, this method is only used in two phase LDPC decoder [21-23]. In two phase LDPC decoder, memory access conflicts will be caused since multiple memory words are accessed per cycle (three different memory words in worst case [22]). In [21], double buffering and alignment units are used to eliminate the memory access conflict, but it doubles both the size of memory and the clock frequency of memory access.

In this paper, a generic data packing scheme is proposed, and the scheme can be used in any QC-LDPC codes directly. A partially parallel decoder is also proposed, in which the parallel process units in [20] and the proposed data packing scheme are used. Layered decoding scheme is used to increase the decoding throughput, and by using the read and write alignment units, no memory access conflict will occur in the decoding process. The proposed decoder uses only 16 process units and 16% less area to obtain the same throughput as that of the decoder in [20], which uses 96 process units.

The rest of this paper is organized as follows. Section 2 describes the LDPC decoding algorithm, layered decoding scheme and parallel min unit. Section 3 proposes the generic data packing scheme. Section 4 proposes the LDPC decoder architecture, using the parallel check node process unit and the proposed data packing scheme. In section 5, the LDPC decoder using the proposed architecture is designed, and the implementation results are described. Section 6 concludes the paper.

2 Background

2.1 Min-Sum Algorithm

Min-sum algorithm is one of the most popular approaches of BP [9], and it greatly reduces the implementation complexity. Both normalized min-sum and offset min-sum algorithm are based on the min-sum algorithm. Min-sum algorithm is expressed as follows:

$$r_{ij} = \prod_{i' \in V_{j \setminus i}} a_{ji'} \cdot \min_{i' \in V_{j \setminus i}} \left| q_{ji'} \right| \quad (1)$$

$$q_{ji} = LLR_j + \sum_{j' \in C_{i \setminus j}} r_{j'i} \quad (2)$$

$$Q_j = LLR_j + \sum_{j' \in C_i} r_{j'i} \quad (3)$$

$$LLR_j = \log \frac{P(x=0 \mid y)}{P(x=1 \mid y)} \quad (4)$$

In (1), r_{ij} is the check-to-variable message passed from check node i to variable node j, $q_{ji'}$ is the variable-to-check message passed from variable node j to check node i', $a_{ji'}$ is the sign of $q_{ji'}$, $V_{j\setminus i}$ is the set of the variable nodes which connect to check node j without node i, in (2), $C_{i\setminus j}$ is the set of the check nodes which connect to variable node i without check node j, in (3), Q_j is the log likelihood ratio (LLR) for variable node j, C_i is the set of all the check nodes which connect to variable node i, in (4), x is the transmitted bit and y is the message received from channel.

At the beginning of decoding, all variable node messages q_{ji} are installed by (4). In each iteration, (1), (2) and (3) are processed serially, and a guess of the codeword is obtained by the sign of Q_j (0 for $Q_j > 0$, 1 for $Q_j < 0$) in (3), if the codeword fits all the parity check or the iteration exceeds the predefined maximum iteration time, the decoding stops.

2.2 Modified min-sum Decoding Algorithm

Although it is easy to implement the min-sum algorithm, it results in degradation in decoding performance. Both normalized min-sum and offset min-sum is modified min-sum algorithm, one with a normalized factor and the other with an additive correction factor, and these algorithms achieve almost the same performance as that of the BP algorithm.

In the normalized min-sum algorithm, check node updating operation uses normalization constant λ smaller than 1, and (1) is changed to:

$$r_{ij} = \prod_{i' \in V_{j \backslash i}} a_{ji'} \cdot \lambda \cdot \min_{i' \in V_{j \backslash i}} |q_{ji'}| \tag{5}$$

In offset min-sum algorithm, the check node updating operation is given as follows:

$$r_{ij} = \prod_{i' \in V_{j \backslash i}} a_{ji'} \cdot \max\left\{ \min_{i' \in V_{j \backslash i}} |q_{ji'}| - \varepsilon, 0 \right\} \tag{6}$$

In (5) and (6), both correction factors are used to decrease the magnitude of r_{ij}, and their performance is analyzed in [12]. Compared to offset min-sum, normalized min-sum algorithm has better decoding performance but the multiplication increases the implementation complexity. In this paper, normalized min-sum algorithm is used in the proposed LDPC decoder.

2.3 Layered Decoding Scheme

In layered decoding scheme, the parity check matrix can be viewed as horizontal layers and each layer can represent a component code [13]. The whole code is composed of all the layers and their intersections. In QC-LDPC code, the rows and columns are naturally blocked by the permutation matrix, so each block row can be indicated as a layer. As each layer starts decoding, its inputs are combined from the channel inputs and the newly calculated extrinsic probability messages from the previous layers. Iterations within a layer are called sub-iterations and the overall process is labelled as super-iterations.

In each sub-iteration, the variable-to-check message q_{ji}' is firstly computed by:

$$q_{ji}' = Q_j' - r_{ij}' \tag{7}$$

Where r_{ij}' is the check-to-variable message in the previous super-iteration of this layer, and Q_j' is the LLR result of variable node j from the previous sub iteration. The r_{ij}' is computed by (1) or (5) or (6), and Q_j is computed by:

$$Q_j = q_{ji}' + r_{ij} \tag{8}$$

When super-iteration is finished, The codeword is obtained by the sign of Q_j.

In general, the decoding convergence speed of the layered decoding scheme is two times faster than that of the two-phase scheme, and in layered decoding, only Q_j and r_{ij} is stored in memories because the variable-to-check message q_{ji} can be computed by (7).

2.4 Parallel Check Node Units

In serial partially parallel decoder, all the p messages in a sub matrix are packaged in one memory word. In one cycle, only one message of a row or a column can be updated in one process unit, so serial partially parallel decoder usually has low throughput.

Parallel process units can update the messages of a row or a column in just one cycle. A generic and scalable parallel min unit is proposed in [20], it supports both irregular and regular LDPC codes with check node of any degree d. A fully parallel LDPC decoder architecture which employs 96 min units is also proposed and it can achieve a throughput of 1.2 Gbps at a clock frequency of 91M.

3 Generic Data Packing Scheme

The proposed generic data packing scheme can be used in any QC-LDPC codes decoder, it includes straightforward data packing method and memory access control logic.

The $M \times N$ parity matrix of a QC-LDPC code is always given as an $m \times n$ array of $p \times p$ sub matrices, each sub matrix is either a $p \times p$ zero matrix or a cyclic-shifted identity matrix, so the parity matrix is divided into n block columns and m block rows.

In layered decoding scheme, only variable node messages Q_j and check-to-variable messages r_{ij} are needed to be stored in LLR RAM and Check RAM, and the Check RAM (which stores r_{ij}) will be introduced in section 4.

Let k denotes the number of parallel process units used in the decoder. In order to enable the reading of all the Q_j of k rows simultaneously, n memory units (corresponding to the n block columns) are needed and each memory unit must support read and write of k locations simultaneously.

In the proposed generic data packing scheme, the straightforward message packing method is used, in which k adjacent messages are stored sequentially in each memory word. Fig. 1 shows an example of data packing of $p = 16$ and $k = 4$, all the

3	d15	d14	d13	d12
2	d11	d10	d9	d8
1	d7	d6	d5	d4
0	d3	d2	d1	d0

Figure 2. An example of data packing, with $p=16$ and $k=4$.

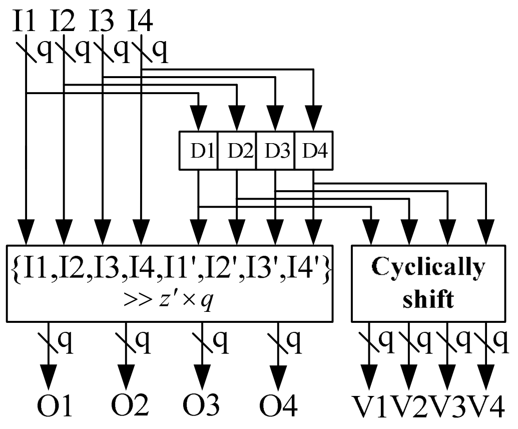

Figure 3. Architecture of the read alignment unit and right shifter for the example in Fig. 1

Table 1: Transition Table for the Alignment Unit and Left Shifter in Fig. 3

Symbol	Cycle1	Cycle5
I1, I2, I3, I4	d10, d9, d8, d7	-
D1,D2,D3,D4	-	d6, d5, d4, d3
V1,V2,V3,V4	d6', d5', d4', d7'	d6',d5',d4', d7
l1, l2, l3, l4	V1, V2, V3, V4	D1, D2, D3, D4
h1, h2, h3, h4	I1, I2, I3, I4	V1, V2, V3, V4
O1,O2,O3,O4	d7, d6', d5', d4'	d7, d6, d5, d4

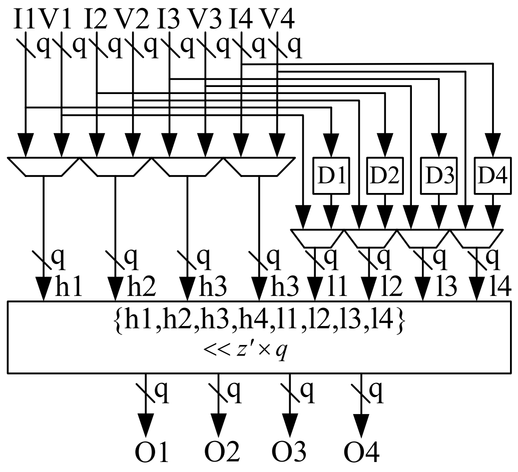

Figure 1. Architecture of the write alignment unit and left shifter for the example in Fig. 1

p messages are directly saved in $\lceil p/k \rceil = 4$ locations. Let z denotes the cyclic-shifted parameter of the $p \times p$ sub matrix, because k messages are located in one word in the memory, the shifted value for each shifter is changed to $z' = z \bmod k$, and the address of d_i is $\lfloor i/k \rfloor$, where $0 \le i \le p-1$.

In the decoding process, k data is needed from each memory unit in one cycle, and in most cases, these k data is not packed in the same location, in order to read and write data sequentially, a read alignment unit and a write alignment unit are needed for each memory unit, and a right shifter and left shifter is also needed in read and write alignment unit.

The architecture of the read alignment unit and right shifter (RA&RS) for the example in Fig. 1 is shown in Fig. 2. All the 4 input data are buffered in *D*, so in the current cycle, the values of *D* are the input data of the previous cycle. All the input data of the current cycle and the previous cycle are concatenated and right shifted by $z' \times q$ bits, then the lower 4 data are chosen and connected to the 4 outputs *O1*, *O2*, *O3*, *O4*, and they are the input data of the 4 process units. Outputs *V1*, *V2*, *V3*, *V4* are the cyclically shifted results of *D1*, *D2*, *D3*, *D4*, and they will be used in the corresponding write alignment unit.

For example, let $z=7$, so the shift parameter for packing messages is $z'=3$, in the first cycle, the reading address is 1, the 4 input data are *d7*, *d6*, *d5* and *d4*, in the second cycle, the reading address is 2, the 4 input data are *d11*, *d10*, *d9* and *d8*, all these 8 data are concatenated and right shift by $3 \times q$ bits, at last, the lower 4 data are chosen and the output data are *d7*, *d8*, *d9* and *d10*. The data of outputs *V1*, *V2*, *V3* and *V4* are *d7*, *d4*, *d5* and *d6*.

Fig. 3 shows the write alignment unit and left shifter (WA&LS). This unit has 2 groups of inputs and 2 groups of multiplexers. *Ii* is the result of (8) from process unit, *Vi* is the output data of the read alignment unit, where $1 \le i \le 4$. In the first

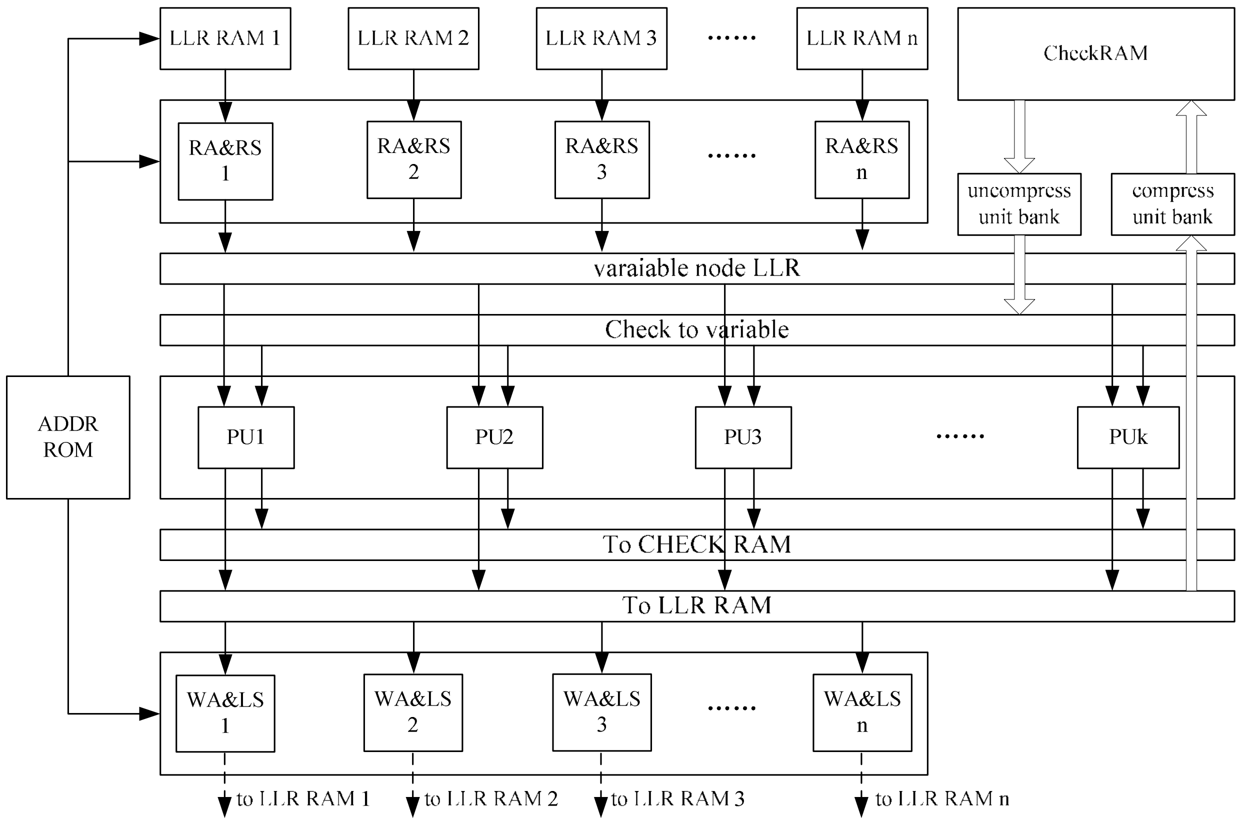

Figure 4. Partially parallel decoding architecture using proposed message packing and alignment unit.

processing cycle, the new LLRs updated by the process unit may not be in the same location of the memory. In the previous example, *d7*, *d8*, *d9* and *d10* are updated in the first process cycle, they are located in address 1 and 2 of the data memory, and the writing process cannot be finished in one cycle. In the first writing cycle, only *d7* is written to the memory at address 1 together with *d6'*, *d5'*, *d4'* (*d6'*, *d5'*, *d4'* are the LLRs of the previous iteration). In this cycle, *hi=Ii*, *li=Vi*, and the output data of the alignment unit is *d7*, *d6'*, *d5'* and *d4'*. The detail of this process is shown in the second column of Table 1. In the fifth writing cycle, the outputs of the alignment unit are *d7*, *d6*, *d5* and *d4*, it is shown in the third column of Table 1.

In this generic message packing scheme, N variable node messages are divided into n groups, each group has p variable node messages and stored sequentially in n memory units. $\lceil p/k \rceil + 1$ cycles are needed for both memory reading and writing, and in layered decoding, both check node and variable node update concurrently, no conflict will occur.

4 Partially Parallel LDPC Decoder Architecture for QC LDPC codes

The proposed partially parallel decoding architecture for QC LDPC codes is shown in Fig. 4. This architecture has two advantages:

1) The proposed generic data packing scheme and layered decoding are used, no shift network is needed, it greatly reduces the area of the LDPC decoder, and no memory access conflict occur.
2) Only a few parallel process units are used, all the messages in a process unit can be accessed from memories in one cycle, and the decoder provides a high throughput.

In this decoder, N variable node messages are divided into n groups and stored in n LLR RAMs, each memory word contains k variable massages, so $k \times d_i$ (d_i denote the hamming weight of the *i*th row) messages can be transferred to k process units in one cycle. In the first iteration of decoding, the LLR RAMs store the channel messages LLR_j, in

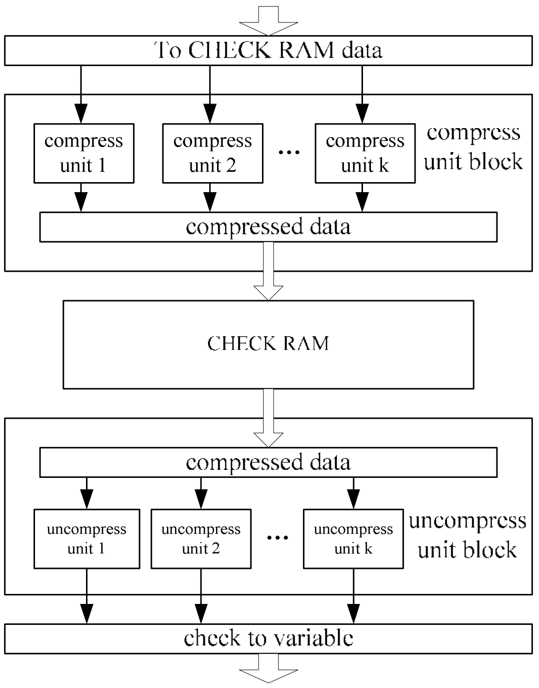

Figure 7. Data flow of CHECK RAM.

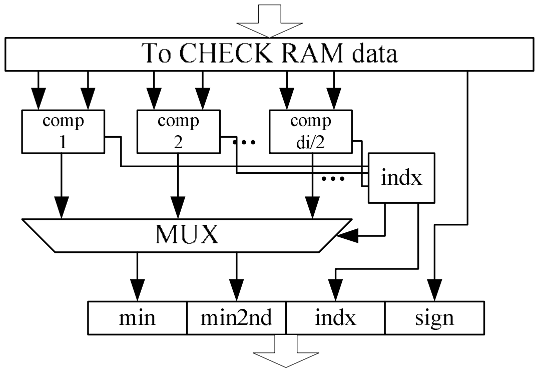

Figure 8. Architecture of compress unit.

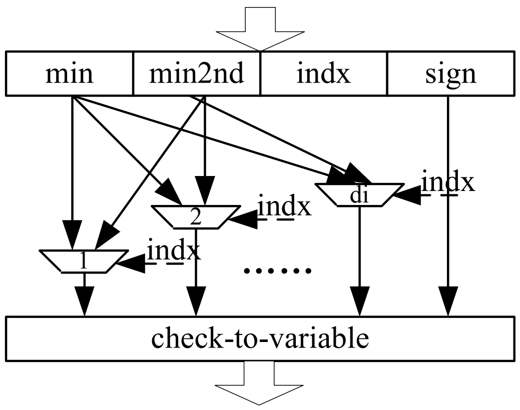

Figure 5. Arechitecture of uncompress unit.

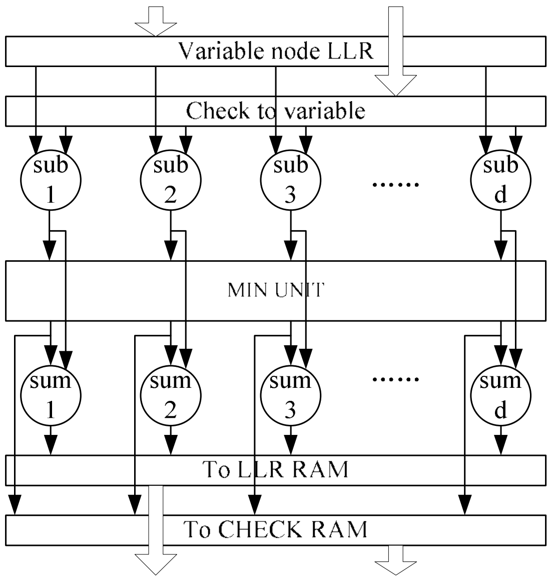

Figure 6. Architecture of a PU.

the other iteration, the LLR RAMs store Q_j.

The Check RAM is used to store the check-to-variable messages. In this design, the check-to-variable messages are stored in a compressed mode [25] to reduce the size of memory. In the compressed mode, only the minimum message, the second minimum message, the index of minimum message and the sign of all the check-to-variable messages r_{ij} are stored. In the proposed decoder, k parallel check node process units are used, so all the d_i check-to-variable messages in one row are updated in one cycle. So in this design, as shown in Fig. 5, k compressed units are used to transform the k groups of d_i check-to-variable messages into the compressed mode, and k uncompressed units are used to do the inverse job. Fig. 6 and Fig. 7 show the architecture of compressed unit and uncompressed unit. In the compress unit, the two adjacent data are compared and the compare result is used to compute the index of the minimum value, and the uncompress unit contains d_i multiplexers which are controlled by the index.

All the LLR RAMs and CHECK RAM are dual-port RAM and can be read and written independently in a single cycle, which results in very high decoding throughput.

The process unit in the proposed decoder is shown in Fig. 8. It contains a min unit in [20] to process (1) and two groups of adders to process (7) and (8).

Table 2: Implementation Results and Comparison with Existing LDPC Decoders

Features	Chih-Hao[23]	Venkata[18]	Proposed			
			K=4	K=8	K=12	K=16
CLK (MHz)	150	91	91	91	91	91
Quantization (bits)	6	6	6	6	6	6
Iteration	20	5	5	5	5	5
Technology	0.09μm	0.18 μm	0.18 μm	0.18 μm	0.18 μm	0.18 μm
Maximum Throughput	150Mbps	1.2Gbps	388Mbps	700Mbps	953Mbps	1.16Gbps
Area (K gates)	970	925	324	475	643	778

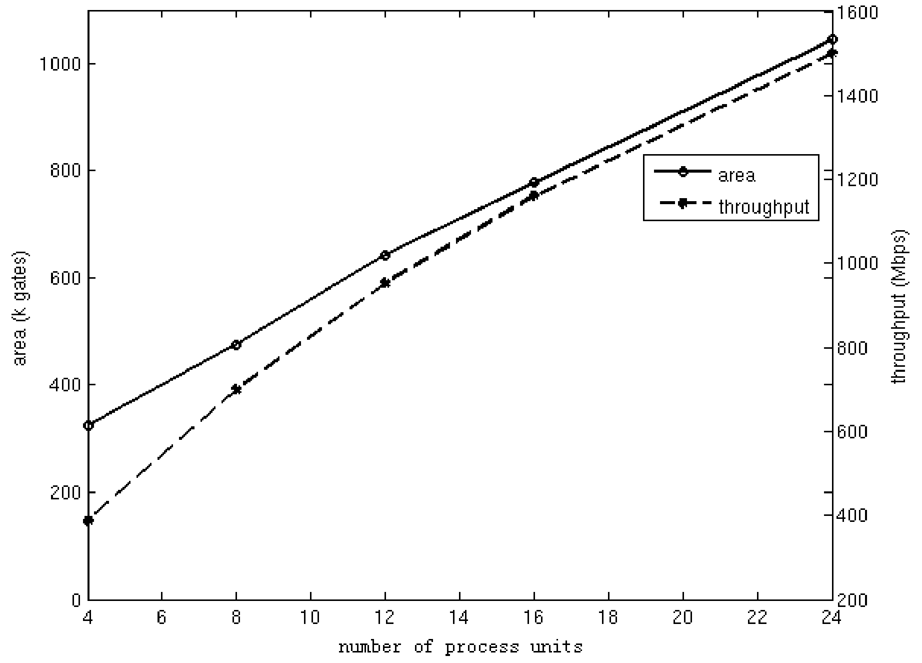

Figure 9. LDPC decoder area versus the number of process units.

5 Implementation and Results

A multi rate QC LDPC decoder is implemented in Verilog HDL, it uses the proposed architecture and supports all code rates and all code lengths of IEEE 802.16e standard. Cadence NCSim is used for simulation and verification. We synthesized the proposed decoder using Synopsys design flow and mapped it to a SMIC 0.18-μm standard-cell library.

Table 2 summarizes the implementation results of the proposed decoder and some other published implementations, the proposed decoder includes of various numbers of process units (k=4, 8, 12 and 16). The throughput results of the proposed architecture are obtained by assuming five iterations and block length of 2304 bits, the peak throughput is given in rate 5/6. Simulation results show that the proposed decoder achieves a similar throughput as in [20] when k=16, with the area is 16% smaller. Compared to [26], in which a serial partially parallel LDPC decoder is presented, the proposed decoder has higher throughput but less gate count when k=8, 12 and 16.

Fig. 9 shows the area and throughput of proposed decoder when the number of process unit is 4, 8, 12, 16 and 24. From the result, it is concluded that the area and throughput of the proposed decoder increases linearly with the number of process unit.

6 Conclusion

In this paper, a generic data packing scheme for QC LDPC codes decoder is proposed, it greatly reduces the area and increases the throughput of LDPC decoder. A QC LDPC decoder architecture is also proposed, it uses the parallel min-sum unit and the proposed data packing scheme. An implementation example that supports all code rates and all code lengths in IEEE 802.16e is demonstrated, and the simulation result shows that the proposed QC LDPC codes decoder reduces the area by 16% when has nearly the same throughput of the existing LDPC decoder.

References:

[1] R.G.Gallager, Low-density parity-check codes, *IEEE Transactions on Information Theory*, Jan.1962, pp.21-28.

[2] D. J. C. MacKay and R. M. Neal, Near Shannon limit performance of low-density parity-check codes, *Electronic Letter*, vol. 32, Aug 1996, pp. 1645-1646

[3] IEEE Std. 802 11n-2009, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Enhancements for Higher Throughput, *IEEE P802.11n*, Sep. 2009.

[4] Digital video broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broad-band satellite applications, *ETSI*, EN 302 307 V1. 1.1, 2005.

[5] IEEE Std. 802 16e, Air Interface for Fixed and Mobile Broadband Wireless Access Systems, *IEEE P802.16e*, Feb. 2006.

[6] M. Khan, R. Carrasco and I. Wassell, application of high rate LDPC codes for IEEE 802.16d or WiMAX (fixed broadband) systems, *WSEAS Transactions on Communications*, vol. 6, iss. 1, Jan. 2007, pp. 92-97.

[7] J. Chang, K. Takeshi, F. Hisato and H. Kazuhisa, Influence of laser linewidth and modulation level for coherent optical OFDM with punctured LDPC codes, *WSEAS Transactions on Communications*, vol. 10, iss. 6, pp. 175-181.

[8] J. Chen and M. P. C. Fossorier, Near optimum universal belief propagation based decoding of low-density parity check codes, *IEEE Transactions on Communications*, vol. 50, Mar. 2002, pp. 406-414.

[9] J. Chen and M. P. C. Fossorier, Decoding low-density parity-check codes with normalized APP-based algorithm, *Proceedings of IEEE Globecom, San Antonio*, TX, Nov. 2001, pp. 1026–1030.

[10] E. Eleftheriou, T. Mittelholzer, A. Dholakia, Reduced-complexity decoding algorithm for low-density parity-check codes, *IEEE Electronic Letters*, vol. 37, Jan. 2001, pp. 102–104.

[11] M.P.C. Fossorier, M. Mihaljevic, H. Imai, Reduced complexity iterative decoding of low density parity check codes based on belief propagation, *IEEE Transactions on Communications*, vol. 47, no. 5, May 1999, pp. 673–680.

[12] D.E. Hocevar, A Reduced Complexity Decoder Architecture via Layered Decoding of LDPC Codes, *IEEE workshop on Signal Processing Systems*, Oct. 2004, pp: 107-112.

[13] J. Xie, L. Yin, N. Ge and J. Lu, Fast Convergence Algorithm for Decoding of Low Density Parity Check Codes, *WSEAS Transactions on Communications*, volume.8, issue7, July 2009, pp. 598-607

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

[15] L. Lan, L. Zeng, Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach, *IEEE Transactions on Information Theory*, vol. 53, no. 7, Jul. 2007, pp. 2429–2458.

[16] M. Karkooti, P. Radosavljevic, and J. R. Cavallaro, Configurable, high throughput, irregular LDPC decoder architecture: Tradeoff analysis and implementation, *Proceedings of IEEE Int. Conf. Application-Specific Syst., Arch., and Process. (ASAP)*, 2006, pp. 360–367.

[17] T. Brack, F. Kienle, and N.Wehn, Disclosing the LDPC code decoder design space, *Proceedings of Design, Automation and Test in Europe (DATE)*, Mar. 2006, pp. 1-6.

[18] K. Gunnam, G. Choi, W. Wang, and M. Yeary, Multi-rate layered decoder architecture for block LDPC codes of IEEE 802.11n wireless standard, *Proceedings of IEEE Int. Symp. Circuits and Syst. (ISCAS)*, May 2007, pp. 1645–1648.

[19] Y. Sun, M. Karkotti, and J. R. Cavallaro, High throughput, parallel, scalable, LDPC encoder/decoder architecture for OFDM

systems, *Proceeding of IEEE Dallas/CAS Workshop*, Oct. 2006, pp. 39–42.

[20] V. K. K. Srinivasan, C. K. Singh, P. T. Balsara, A Generic Scalable Architecture for Min-Sum/Offset-Min-Sum Unit for Irregular/Regular LDPC Decoder, *IEEE Transactions on VLSI*, vol. 18, no. 9, Aug., 2010, pp. 1372-1376.

[21] X. H. Chen, J. Y. Kang, S. Lin V. Akella, Memory System Optimization for FPGA-Based Implementation of Quasi-Cyclic LDPC Codes Decoders, *IEEE Transactions on Circuit and System*, vol. 58, Issue: 1, Jan. 2011, pp. 98-111.

[22] Z. Wang and Z. Cui, Low-complexity high-speed decoder design for quasi-cyclic LDPC codes, *IEEE Transactions on Very Large Scale Integr. (VLSI) Systems*, vol. 15, no. 1, Jan. 2007, pp. 104–114.

[23] M. Gomes, G. Falcao, V. Silva, V. Ferreira, A. Sengo, and M. Falcao, Flexible parallel architecture for DVB-S2 LDPC decoders, *Proceeding on IEEE Global Telecommunication Conference*, Nov. 2007, pp. 3265–3269.

[24] Y. Dai, Z. Yan, and N. Chen, Optimal overlapped message passing decoding of quasi-cyclic LDPC codes, *IEEE Transactions on Circuit and Systems*, vol. 16, no. 5, May 2008, pp. 565–578.

[25] J. Sha, Z. Wang, M. Gao, and L. Li, Multi-Gb/s LDPC Code Design and Implementation, *IEEE Transactions on Circuit and Systems*, vol. 17, no. 2, Feb. 2009, pp. 262-268.

[26] 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 Journal of Solid-State Circuits*, vol. 43, no. 3, March, 2008, pp. 684-694.

Biographies

Jinlei Chen born in 1982. He received the B.S. degree from Jilin University in microelectronics in 2005 and M.S. degree in microelectronics from the Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China, in 2007. Since 2008, he has been a PhD candidate in microelectronics. His main research interests include LDPC codes and VLSI hardware design.

Yan Zhang born in 1969. He has been professor of the Shenzhen Graduate School, Harbin Institute of Technology since 2002. His main research interests are application specific instruction set processor design, including medical image processing chips and wireless communication baseband chip.

Wang Xu born in 1980. Received the M.A's. degrees in microelectronics from the Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China, in 2007. Since 2008, he has been a PhD candidate in microelectronics. His main research interests include image processing and embedded DSP processor design.