IPSJ Transactions on System LSI Design Methodology Vol. 3 292–302 (Aug. 2010)

Regular Paper

A High Parallelism LDPC Decoder with an Early Stopping Criterion for WiMax and WiFi Application

Zhixiang Chen,[†1] Xiongxin Zhao,[†1] Xiao Peng,[†1]
Dajiang Zhou[†1] and Satoshi Goto[†1]

In this paper we propose a synthesizable LDPC decoder IP core for WiMax and WiFi applications. Two new techniques are applied in the proposed decoder to improve the decoding performance. Firstly, a high parallelism permutation network (PN) is proposed to perform the circulant shift according to the parity check matrix (PCM) defined in WiMax and WiFi standards. By using the proposed PN, at most, four independent code frames with small code length are decoded concurrently, which largely improves the decoding throughput (2-4 times). Secondly, a fast early stopping criterion specialized for WiMax and WiFi LDPC code is proposed to reduce the average iteration number. Unlike the early works, by utilizing our proposed stopping criterion, the decoding will be stopped when all the information bits of a code frame are corrected even if there are still some errors in redundant part. Experiment results show that, it can reduce up to 20% iteration numbers compared to popular used stopping criterion.

1. Introduction

First invented by R. Gallager [1] in the 1960s and rediscovered by D.J.C. MacKay [2] in the 1990s, Low Density Parity Check Code (LDPC) as a forward error correcting code is now playing an important role in modern digital communication area because of its near-capacity error correction performance and potential for high-throughput decoding using iterative message-passing algorithm. Recently, LDPC code is adopted by wireless communication standards, WiMax (IEEE 802.16e) [3] and WiFi (IEEE 802.11n) [4] standards which are two promising standards for next-generation communication. LDPC codes defined in WiMax and WiFi standards are Quasi-Cyclic (QC) LDPC codes and support multiple code rates and code lengths for applications at different channel conditions. Especially, thanks to the structured parity check matrix (PCM), LDPC codes in WiMax and WiFi standards are very suitable for layered message passing (LMP) algorithm [5] which is proved to speed up the decoding convergence by a factor of two [6].

Based on LMP algorithm, an LDPC decoder architecture, called partially parallel decoder [7], was proposed to decode QC-LDPC code. Besides the advantage of high flexibility that supports multiple code lengths and code rates, partially parallel decoder does not have wire congestion problem which dominates the chip area in some conventional decoder architectures. Recently, based on partially parallel decoder architecture, research works on LDPC decoder [8)–10] have been done for WiMax or WiFi applications. For all of them, a flexible permutation network (PN) is needed to cyclically shift data of variable group sizes according to the QC-LDPC PCMs. Therefore, how to design PN becomes a hot topic in research area. Previous works have been done in Ref. 8) and recently some Benes network (BN) [11] based PNs have been reported in Refs. 12), 13). Except one PN in Ref. 14) all of the early works, even the latest state of the art work in Ref. 15), can only cyclically shift one group of data at one time even if the size of the data group is much smaller than the input size of the PN. This demerit leads to low throughput and low hardware utilization for LDPC decoder when the code length is small. In this paper, based on BN and the control algorithm proposed in Ref. 16), we propose a high-parallelism PN. Several groups (up to four) of data can be cyclically shifted concurrently by our proposed PN when the size of the data group is no bigger than half or quarter of the maximum input size of it.

Another technique proposed in this paper is an early stopping criterion (SC) specialized for LDPC code in WiMax and WiFi standards. Firstly, an observation on convergence speed of information bit and redundant bit of LDPC code in WiMax and WiFi standards is presented and discussed. Based on the observation and the particular structure of the PCM, an early SC is proposed to save the unnecessary decoding iterations for only correcting the errors in redundant part of a code frame. Compared to popular-used hard decision aided (HDA) early SC [17], our proposed early SC reduces average iteration number (AIN) by up to 20% with near zero bit error ratio (BER) performance loss.

With applying these two techniques, an improved partially parallel LDPC de-

†1 School of Information, Production and Systems, Waseda University

coder was designed and synthesized. Synthesis result shows that compared to the previous work, without additional hardware consumption our proposed decoder can achieve four-time throughput gain when the code length is small. The AIN is reduced by up to 20% by utilizing our proposed SC. The paper is organized as follows: Section 2 introduces necessary background knowledge including LDPC code in WiMax and WiFi standards, decoding algorithm, partially parallel decoder architecture and early stopping criterion. In Section 3, we propose our high-parallelism PN with synthesis result and comparisons. Our early SC is proposed in Section 4 with its performance and hardware overhead analysis. The proposed decoder and hardware implementation details are reported in Section 5. Section 6 draws the conclusion of this paper.

2. Preliminaries

2.1 LDPC code in WiMax and WiFi Standards

An LDPC code is defined by an $M \times N$ PCM where N and M equal to the code length and the number of redundant bits in a code frame, respectively. LDPC codes in WiMax and WiFi standards are quasi-cyclic (QC) codes, the PCM of which is an expansion of a $c \times d$ base matrix. The base matrix is expanded by $z \times z$ sub-matrices where z is called an expansion factor. It is composed of nonnegative integers and "-1" ("-" for WiFi) where the nonnegative integer represents a $z \times z$ cyclically shifted identity matrix and "-1" represents a $z \times z$ all-zero matrix. For the cyclically shifted identity matrix, the number of shift equals to its corresponding nonnegative integer. The code length N equals to $z \times d$, where d is the number of columns in a base matrix. An expansion example of $z = 4$ and $d = 3$ is shown in **Fig. 1**. For all base matrices in WiMax and WiFi, d equals to 24.

2.2 Decoding Algorithm and Partially Parallel Decoder

LDPC code can be decoded using iterative message passing algorithm. Research works on decoding algorithms mainly focus on two topics: message passing schedules including layered and flooding message passing algorithms [18] and implementation of check node operations such as belief propagation (BP), Min-Sum (MS) [19], Normalized Min-Sum (NMS) [20] and Offset-Min-Sum (OMS) algorithms. In this design, we apply layered message passing (LMP) and NMS

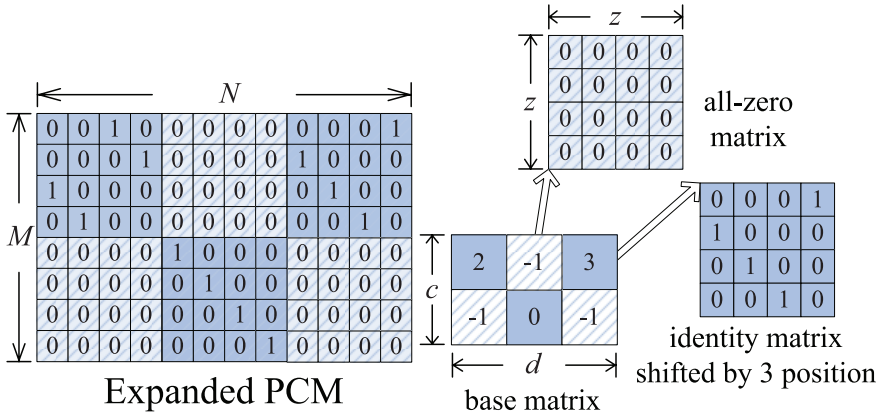


Fig. 1 QC-LDPC PCM expansion.

algorithms. Based on LMP algorithm and PCM structure of QC-LDPC code, a partially parallel decoder was proposed in Ref. 7). The architecture block diagram is shown in **Fig. 2**. A posterior probability (APP) messages are stored in APP memory and passed via permutation network (PN) to process engine (PE) array. As introduced above, PN module performs the circulant shift for a group of APP messages (z messages). All the message updating and calculations according to LMP and NMS algorithms are processed in PE array. Thanks to the independence of rows in one block of QC-LDPC code, z rows (one layer) can be processed in z PEs concurrently. Check node message are stored in Check Message Memory. Memory address, control signals and PCM information are generated or stored in Controller.

2.3 Early Stopping Criterion

Early stopping criterion (SC) was proposed to save unnecessary decoding iterations which account for more power consumption. During the iterative decoding process, if SC is satisfied before a predefined maximum iteration number is achieved, the decoding will be stopped. The most popular used SC is called hard decision aided (HDA) [21] SC which was originally invented for Turbo code decoding and applied in LDPC code recently. For HDA SC, if the hard decisions made in two successive decoding iterations do not change, the decoding

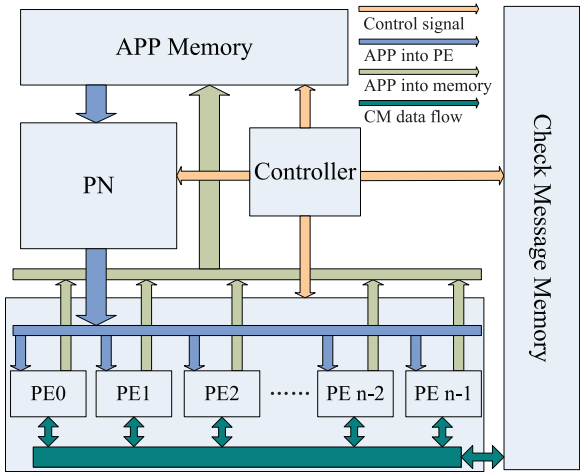


Fig. 2 Partially parallel decoder architecture.

will be stopped [22]. However, HDA SC does introduce some BER performance loss because no changes of hard decisions made in two successive decoding iterations does not guarantee a correct code word. To overcome this demerit, a threshold value of APP magnitude was introduced in Ref. 17). The decoding will not be stopped until the minimum value of APP magnitudes is larger than the pre-defined threshold value. With this extra constrain, the BER performance of HDA algorithm was improved a lot.

3. Proposed High-parallelism Permutation Network

In this section, firstly, a throughput and hardware utilization degradation problem at short code length of partially parallel decoder is discussed. Secondly, we introduce a class of switch network, called Benes Network (BN). Thirdly, based on symmetry property of BN and PN proposed in Ref. 16), we propose a high-parallelism permutation network (PN) and its control algorithm which can perform circulant shift for up to four independent data groups concurrently. Hardware implementation result and comparisons are given in the end of this section.

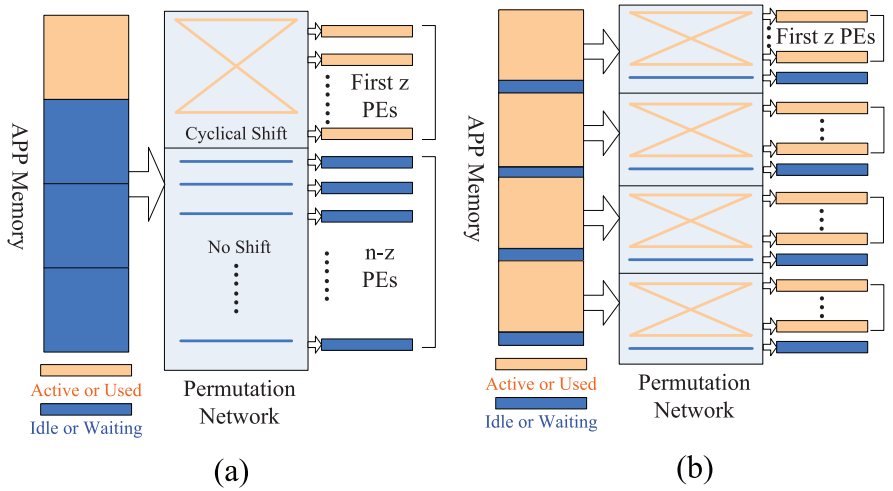


Fig. 3 A problem on partially parallel decoder.

3.1 Problems of Partially Parallel Decoder at Short Code Length

To support different code lengths, PN in decoder has to be flexible to variable size of data group which equals to different z. However, for almost all the early works, only one group of data can be shifted at one time even if the size of the data group (equal to z) is much smaller than the maximum input size. Because of this demerit, the throughput of the decoder will become very low when it is processing a short code. Note that throughput is proportional to the sub-matrix size which equals to the size of the data group fed into the permutation network. As shown in **Fig. 3** (a), if the expansion factor is z, only z APP messages from memory are shifted and only z PEs are actively working at that time resulting low throughput and low hardware usage. For example, assume there are n PEs, if the code length is 576 ($z = 24$), the throughput will be almost one-fourth of that the code length is 2,304 ($z = 96$) and the PE usage will be only 25% ($n = 96$). By concurrently decoding 3 more short-length code frame which are stored in the rest of the APP memory, we can solve both throughput and hardware utilization problems. One way to realize this idea is to add 3 more PNs for additional 3 independent code frames. However, since one PN occupies almost 10%-15% area

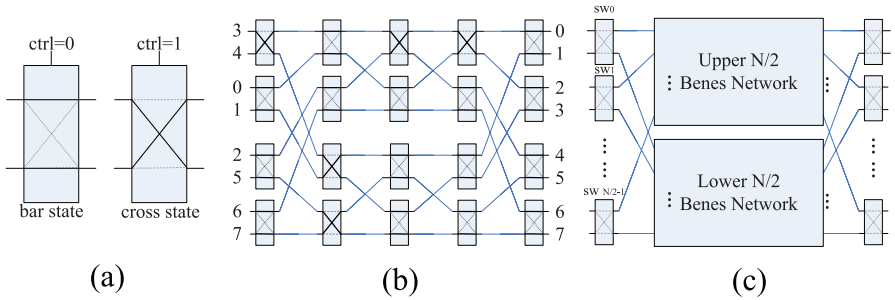


Fig. 4 Benes network.

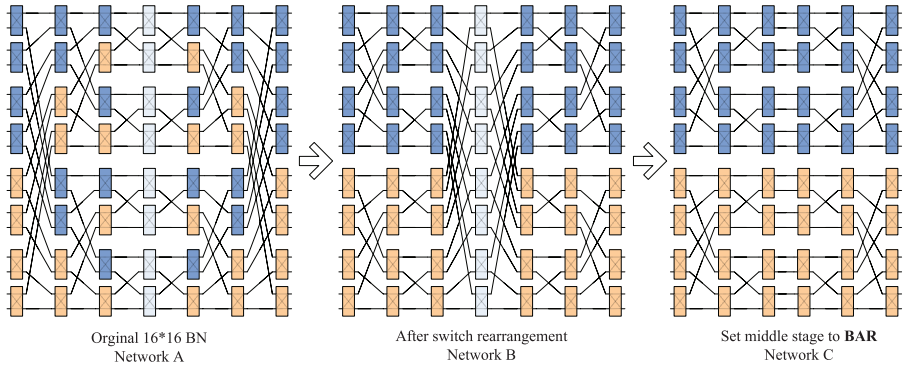


Fig. 5 Symmetry property of Benes network.

or gate count of the whole decoder, it is not reasonable or practical for us to place 4 PNs in one decoder. Our solution is to improve the parallelism of PN, making it possible to cyclically shift several groups of data (up to 4) from memory to PEs at one time, shown in Fig. 3 (b).

3.2 Benes Network

Benes Network (BN) [11] is a network constructed by cross-bar switch, the state of which is controlled by a one-bit signal shown in **Fig. 4** (a). BN as a switch network is capable of performing arbitrary data permutation by appropriately controlling the cross-bar switches in it. A permutation example of an 8×8 BN is shown in Fig. 4 (b). Carefully watch the 8×8 BN in Fig. 4 (b), it can be found that an $N \times N$ BN can be constructed by two stages (input and output) and two $N/2 \times N/2$ BNs as shown in Fig. 4 (c). By exploiting this recursion property, a PN based on BN and an implementable cross-bar switch control algorithm were proposed in Ref. 16) to perform circulant shift for one data group of variable sizes. Since space limitation, we omit the details of Ref. 16) here.

3.3 Proposed High-parallelism Permutation Network

Besides the recursion property mentioned above, without loss of generality, a symmetry property of BN can be explained using a 16×16 BN shown in **Fig. 5**. With the partition of input and output signals into two groups (upper part and lower part), except the middle stage, all the cross-bar switches in Network A can be categorized into two groups, colored blue and orange, shown in the left of Fig. 5. After switch rearrangement, moving blue ones to upper and orange ones to lower part we get Network B shown in the middle of Fig. 5. Shown in the right of Fig. 5, two independent networks referred as Network C can be obtained by setting the switches of the middle stage in Network B to **BAR**. Moreover, if the two middle stages of Network C are combined together, Network C becomes an 8×8 BN. This is called symmetry property of BN. Therefore, same permutation can be performed on two independent groups of input data by setting the same control signals for cross-bar switches at symmetric positions and setting all the switches in the middle stage to **BAR**. This property can be deduced to 4, 8,..., N/2 groups.

Now we propose our high-parallelism permutation network. The architecture is shown in **Fig. 6**. Generally, the PN consists of BN and cross-bar switch control signal generator. Control Signal Generator K (CSG-K) generates control signals for K switches in the input and output stages of BN with K inputs (BN-K). Each BN-K is constructed by two BN-K/2s with CSG-K/2. And each BN-2 is a single cross bar switch.

The control signal generating algorithm is shown in **Fig. 8**. Parameter z, s, N and m denote the size of data group, number of shift positions, size of network input and number of data groups respectively. Function name CSG is standing for control signal generating. CSG(z, s, N, m) sets the switches in the input (output) stage in $N \times N$ BN by step 1 (step 3) and recursively calls function CSG for the smaller BN. Expression "$\lfloor a \rfloor$" denotes the biggest integer which is not larger

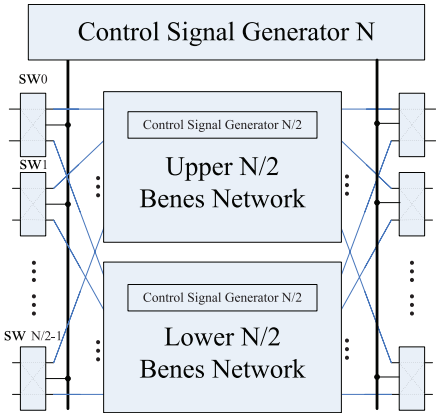


Fig. 6 Architecture of proposed PN.

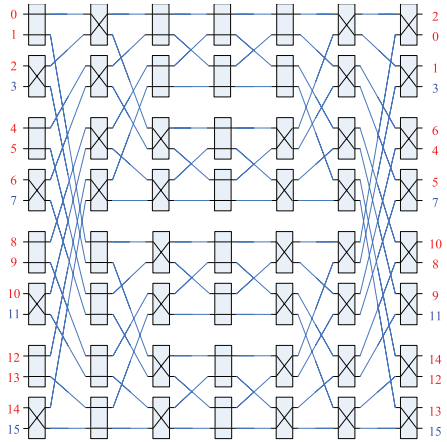


Fig. 7 An permutation example.

than number a. The basic idea of this algorithm is to divide a big shift task to two small ones and combine their results together. For example if we want to cyclically shift one size-2K data group by $2s$ positions, we divide it to two size-K data groups and shift them by s positions respectively. The final result can be obtained by combining two shifted size-K data groups together. For an $N \times N$ BN, data group division is achieved by setting the proper status of the cross-bar switches in the input stage according to step 1 in function CSG(z, s, N, m). Two data groups are combined through the output stage which is set by the step 3 of function CSG(z, s, N, m). Step 2 recursively calls step 1 and step 3 to decrease N. Based on the symmetric property of BN, two or four groups of data can be shifted concurrently by appropriately copying the control signals of cross bar switches that is listed in the algorithm. A permutation example is given in **Fig. 7** where $N = 16$, $z = 3$, $s = 1$ and $m = 4$. Input and output stages in 16×16 BN are set by step 1 and step 3 in CSG(3, 1, 16, 4), respectively. CSG(1, 0, 8, 4) and CSG(2, 1, 8, 4) are called in step 2 to set the signals in upper and lower 8×8 BNs. Recursively, with the decreasing of N all the switches in the BN are set. Totally, the hardware of control signal generating logic occupies about 10% gate count over the whole PN when data width is 9 bits. Comparison of implementation result with previous works is given in **Table 1**. It shows that our proposed

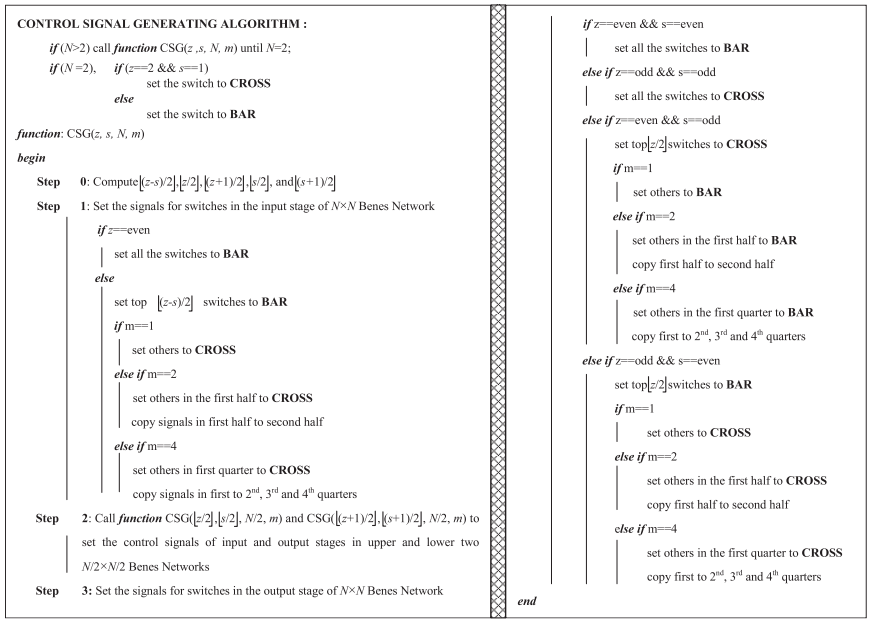


Fig. 8 Cross-bar switch control signal generating algorithm.

Table 1 Comparison of implementation result of PN.

	Proposed	Ref. 8)	Ref. 12)	Ref. 13)	Ref. 14)		Ref. 15)
Technology (nm)	65	130	180	180	130	130	130
Area (mm^2)	–	0.511	2.171	0.722	–	–	0.1358
Gate Count	47.4k	–	–	–	21.9k	37.4k	27.1k
Data Width	9 bits	6 bits	8 bits	8 bits	6 bits	6 bits	9 bits
Max Freq. (Mhz)	311	333	85	94	384	384	500
Network Size	128×128	128×128	128×128	96×96	96×96	96×96	–
Parallelism ($z = 64$)	Two	One	One	One	One	One	One
Parallelism ($z = 32$)	Four	One	One	One	One	Two	One

permutation network can achieve high parallelism data permutation (Two data groups at $32 < z \leqslant 64$ and four at $z \leqslant 32$) while maintaining the acceptable hardware overhead, which is 47.4k gate count for 9-bit input data width.

4. Proposed Early Stopping Criterion

In this section, firstly, we present an observation on convergence speed of information and redundant bits of LDPC code in WiMax and WiFi standards. It implies that a lot of decoding iterations are performed for only correcting the errors in redundant part of a code frame, named Type II. Secondly, an early SC is proposed to terminate the decoding if the Type II code frame is observed. Thirdly, hardware overhead and decoding performance is given in the end of this section.

4.1 Observation on Convergence Speed

For LDPC PCMs defined in WiMax and WiFi standards, the column weights of right most $M - z$ columns in redundant part of the PCM are two which is the smallest number among all. This feature leads to an interesting phenomenon that averagely, APP messages of information bits in a code frame converge to a reliable value much faster than redundant bits which is shown in **Fig. 9**. We explain this phenomenon as follows: The bit node corresponding to the bigger column weight is connected to more check nodes and has more constrains from parity check equations. Consequently, it will get more messages and be updated more frequently. More importantly, because the information bits converge much faster, errors among them will be corrected earlier during the iterative decoding. Therefore, the code frames which are incorrectly decoded due to lack of decoding iterations always contain more errors in redundant part than information part. To demonstrate this, a simulation was run where a rate-0.5 length-2304 code in WiMax was used. The incorrectly decoded code frames are categorized into two types: Type I and Type II. Code frame of Type I contains errors in information part or left z redundant bits while Type II code frame only contains errors in right $M - z$ redundant bits. Proportions of two types of error codes at different channel conditions are depicted in **Fig. 10**. It can be found that the Type II error happens much more frequently than Type I, which implies that a lot of decoding iterations are performed just for correcting Type II errors which can be ignored. If a decoder can detect a Type II frame and stop the decoding process, decoding iterations can be saved to some extent.

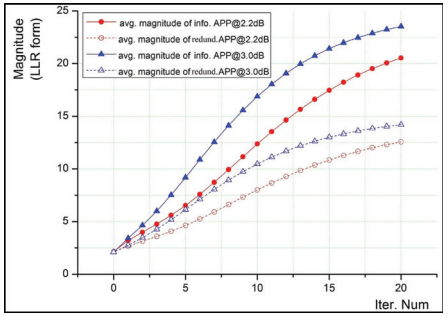


Fig. 9 APP message convergence.

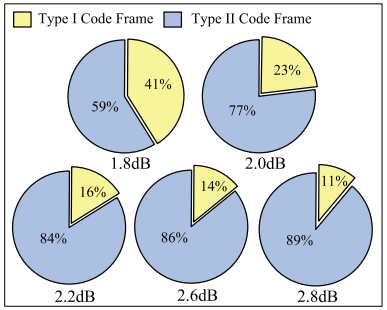


Fig. 10 Proportion of error frame.

4.2 Proposed Early Stopping Criterion

The key idea of our proposed early SC is to save the unnecessary decoding iterations for correcting the Type II error. By exploiting the particular structure of PCMs in WiMax and WiFi standards, a mechanism is devised to find the Type II code frames. Before further discussion, some notations and mathematic definitions are given below. A syndrome vector is defined as $\mathbf{s}$ (1) where M is the number of rows of a PCM and s_i (0 or 1) is the result of the parity-check equation corresponding to the ith row, indexed from 0, of PCM. A syndrome accumulation vector (SAV) $\mathbf{a}$ (2) is defined as a mapping from $\mathbf{s}$ according to (3) where z is an expansion factor of a PCM and c equals to the number of rows in the base matrix. Element a_i of SAV is the sum of results of parity check equations with the same offset index i in c row blocks, where each row block contains z rows.

$$\mathbf{s} = [s_0, s_1, \ldots, s_{M-2}, s_{M-1}]^T \tag{1}$$

$$\mathbf{a} = [a_0, a_1, \ldots, a_{z-2}, s_{z-1}]^T \tag{2}$$

$$a_i = \sum_{k=0}^{c-1} s_{i+kz} \tag{3}$$

According to the standards, for a base matrix with expansion factor z and code rate R, generally, the right most $M - z$ columns of the expanded PCM forms a $M \times (M - z)$ two-diagonal matrix. A bit in a code word is labeled as b_{kz+j}, where k is the index of a column block from 0 to 23 and j is an offset in a column

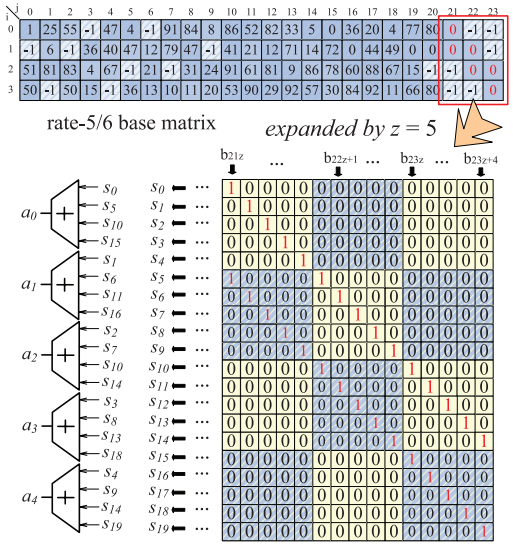


Fig. 11 An example of notation and definition.

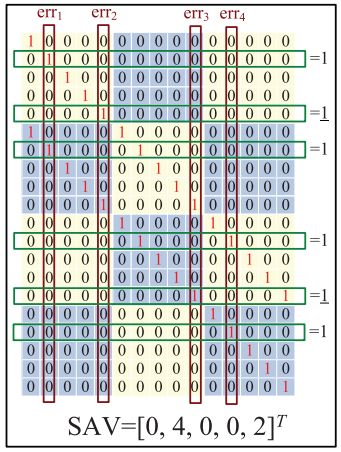


Fig. 13 An example of SAV of Type II code frame.

Start of the Proof

1) For case n=1, the error bit $b_{k_1 z+j_1}$ makes only two parity-check equations corresponding to row $(k_1\text{-}24R)z+j_1$ and row $(k_1\text{-}24R)z+j_1$ of the PCM equal to ones where $24R<k_1<24$, and $0 \leqslant j_1 < z$. Therefore, the SAV contains z-1 zeros and 1 two for a_{j_1}. Thus, the property is true for n=1;

2) Assume that for case n=q the property is true which means if there are q errors, $b_{k_1 z+j_1}$, $b_{k_2 z+j_2}$,…, $b_{k_q z+j_q}$, among the right most M-z bits of the code word, the SAV contains only even numbers;

3) For case n=q+1, the (q+1)th error bit $b_{k_{q+1} z+j_{q+1}}$ only affects two parity-check equations corresponding to row $(k_{q+1}\text{-}24R)z+j_{q+1}$ and row $(k_{q+1}\text{-}24R)z+j_{q+1}$ of the PCM by either flipping them from zero to one or from one to zeros. Therefore, only $a_{j_{q+1}}$ in SAV of case n=q will be added or subtracted by two or unchanged. Based on the assumption of case n=q, for case n=q+1 the SAV still contains only even numbers. Thus, the property is true for n=q+1;

Above all, we have proven that the property is true for every possible n.

End of the Proof

Fig. 12 Proof of SAV property.

block from 0 to $z-1$. An example of the notation and definition above is shown in **Fig. 11** with $R = 5/6$, $z = 5$ (for space saving). If a code frame is of Type II with n errors in redundant part, the SAV contains only even numbers. This is called SAV property which can be proven by the mathematic induction, shown in **Fig. 12**. With SAV property, a Type II code frame can be easily verified like the example shown in **Fig. 13**. For this example, $z = 5$, $R = 5/6$, $n = 4$ and the SAV equals to $[0, 4, 0, 0, 2]^T$.

Reversely, if SAV contains only even numbers, the corresponding code frame is of Type II for almost all the circumstances. This is because an error among information or first z redundant bits will at least affect three parity-check equations corresponding to three rows in random positions of a PCM, making the SAV disorderly. Other than strict proof, we run a simulation to test all the codes in WiMax and WiFi. Simulation result shows the miss detection rate for Type II code frame using SAV property is smaller than 10^{-6} for all kinds of codes in WiMax and WiFi standards. In the worst case, our proposed early stopping algorithm introduces a BER performance degradation of 10^{-8} which can be ignored for WiMax and WiFi applications. Therefore, it is reasonable and effective for us to claim a Type II code word if SAV contains only even numbers. Based on the discussion above, now we propose our new early stopping criterion as follows.

For each decoding iteration, calculate the syndrome accumulation vector (SAV). If SAV contains only even numbers, stop the decoding.

4.3 Performance and Hardware Overhead

Simulation was run to test the performance of our proposed SC. Two codes, a length-2304 rate-1/2 code in WiMax and a length-1944 rate-2/3 code in WiFi standards, were used. Maximum iteration number is set 15 and quantization effect corresponding to hardware design was considered in the simulation. BER and FER curves (solid for BER and dash for FER) are drawn in **Fig. 14** and **Fig. 15** for the two codes, respectively. It is shown that our proposed early stopping algorithm achieves almost the same BER and FER performance as that

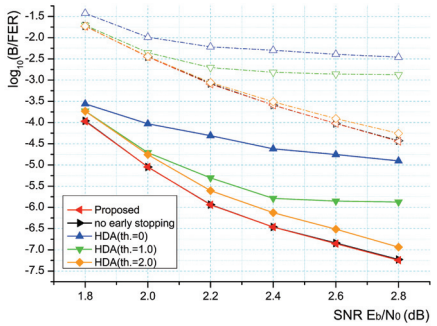


Fig. 14 Comparison of error correction performance (WiMax).

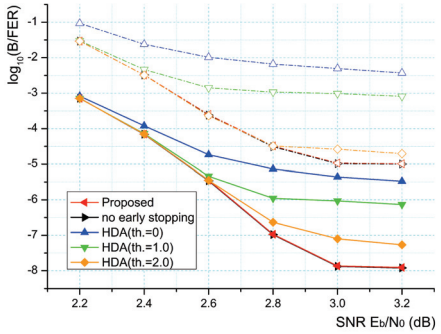


Fig. 15 Comparison of error correction performance (WiFi).

Table 2 Comparisons of AIN.

Code Type	SNR (dB)	Average Iteration Number (AIN)				AIN Reduction to	
		Proposed	HDA(0)	HDA(1.0)	HDA(2.0)	HDA(1.0)	HDA(2.0)
WiMax	2.0	7.09	6.68	8.06	8.98	12.0%	21.0%
$N = 2{,}304$	2.4	5.42	5.11	6.04	6.63	10.3%	18.3%
$R = 1/2$	2.8	4.44	4.13	4.88	5.25	9.1%	15.4%
WiFi	2.4	6.26	5.78	6.83	7.27	8.4%	17.1%
$N = 1{,}944$	2.8	4.70	4.32	5.10	5.36	7.9%	14.4%
$R = 2/3$	3.2	3.88	3.56	4.2	4.37	7.6%	12.6%

without early stopping. The main purpose of SC is to reduce the unnecessary decoding iterations which account for additional power consumption or throughput decrease. To evaluate this functionality, average iteration numbers (AIN) were monitored in the simulations which are listed in **Table 2**.

In Table 2, HDA(x) represents the HDA early stopping with threshold x. It is shown from the simulation result that compared to HDA SC with threshold our proposed SC has much better error correcting performance. It achieves 10% to 20% AIN reduction compared to HDA with threshold 2.0. This technique provides a good potential for decoder power reduction. The hardware overhead is depicted in **Fig. 16**. In each PE, there is an SC module, which is detailed in the red dash block. For an SC module, one D flip-flop and one 1-bit binary adder (*XOR* gate) are needed to calculate the least significant bit of ai of SAV. A

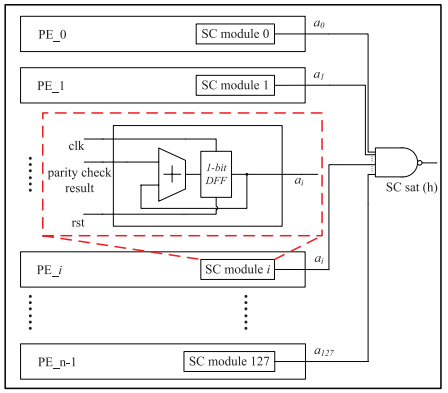


Fig. 16 Hardware overhead of proposed SC.

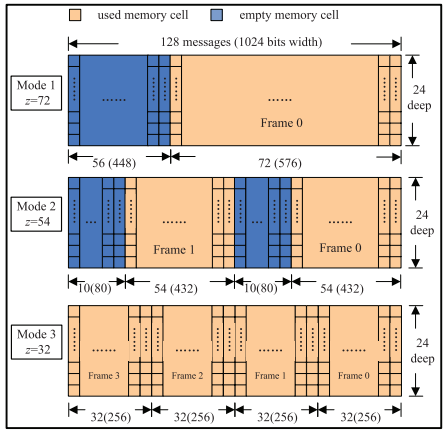


Fig. 17 APP memory organization.

SC satisfaction signal (high effective) is obtained at the output of a *NAND* gate whose inputs are least significant bits of SAV. Totally, the hardware overhead of proposed SC is around 2k gate count which can be ignored over the entire design (650k).

5. Proposed Decoder and Performance Comparisons

In this section, we present the implementation details of our proposed decoder. Synthesis result and performance comparisons are also reported. The basic architecture of our design is based on partially parallel decoder [7]. APP message is stored as two's complement form and quantized by 8 bits, one for sign, five for integer and two bits for fraction. A memory with 1,024-bit (128 messages) width and 24-word depth is used to store APP messages. To support high-parallelism decoding at short code length, the decoder can work under three modes, where one, two ($z \leq 64$) and four ($z \leq 32$) code frames are decoded simultaneously for mode 1, 2 and 3, respectively.

The memory organization under these three work modes are depicted in **Fig. 17**. A 128×128 high-parallelism PN proposed in Section 3 and 128 PEs are implemented in the decoder to perform layered iterative decoding. Check node messages are stored in a compressed form in check message memory. For

Table 3 Synthesis result of proposed decoder and performance comparisons.

		Proposed	Ref. 8)	Ref. 9)	Ref. 10)	Ref. 15)
Technology (nm)		65	130	180	90	90
Gate Count		651.2k	–	349.5k	970k	947k
Area (mm^2)		–	3.834	3.39	6.25	6.22
Support STD.		16e&11n	16e	11n	16e	16e&11n
Operating Frequency (Mhz)		264	333	208	150	300
Max Iteration #		15	15	5	20	20
Throughput (Mb/s)	$z = 32$ (27)	356	111 ⋆1	134	105	212
@Operating Fre./Iter.	$z = 64$ (54)	356	222 ⋆1	286	($z = 96$	($z = 96$
$z = \{16e(11n)\}$	$z = 96$ (81)	266	333	415	$R = 5/6$)	$R = 5/6$)

each PE, a 16-bit wide 12-word deep memory bank is used to store the magnitude of check messages for 12 rows in 12 different layers. For each row, 5 and 6 bits are used to quantize the first minimum and the second minimum magnitudes of check messages of this row, respectively. Another 5 out of 16 bits are used to encode the position of the first minimum value. Signs of the check messages are stored in a 128-bit wide 96-word deep memory. Therefore, 36,864 ($12 \times 16 \times 128 + 128 \times 96$) memory bits are used for check messages. Sum with the APP memory (24,576 bits) 61,440 memory bits are needed in the decoder. For this design, we synthesize the memory using registers which occupy 361k gate count. Synthesis result and performance comparisons to previous work are given in **Table 3**.

Throughput at different code lengths (different z) are shown in Table 3. Except for the design in Ref. 9) all the code for throughput comparisons are WiMax codes. It is obvious that unlike the previous designs the throughput at short code length does not decrease for our proposed decoder. The BER performance of our proposed decoder with proposed early SC is simulated through PC based software and the results are shown in Fig. 14, Fig. 15 and Table 2.

⋆1 These two numbers were calculated based on Eq. (6) and the information in Table 2 of Ref. 8).

6. Conclusion

In this paper, two new techniques are proposed to improve the performance of LDPC decoder applied in WiMax and WiFi standards. By using a high-parallelism permutation network, up to 4 code frames at short length can be processed in the decoder concurrently, thus to increase the decoding throughput by up to 4 times. An early stopping criterion is proposed to improve the decoding efficiency of WiMax and WiFi LDPC codes. Simulation shows it can reduce up to 20% average iteration number compared to most popular used SC. Based on these two techniques a performance improved partially parallel decoder is proposed with synthesis result and performance comparisons.

Acknowledgments This research was supported by "Ambient SoC Global COE Program of Waseda University" of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and by the CREST project of Japan Science and Technology Agency.

References

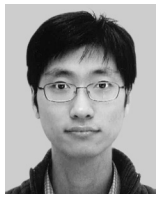
1) Gallager, R.: *Low-Density Parity-Check Codes*, MITPress, Cambridge, MA (1963).
2) MacKay, D.J.C.: Good codes based on very sparse matrices, *IEEE Trans. Inform. Theory*, Vol.45, No.2, pp.399–431 (1999).
3) *IEEE Std 802.16e-2005 and IEEE Std 802.16-2004 /Cor 1-2005* (*Amendment and Corrigendum to IEEE Std 802.16-2004*), IEEE (2006).
4) *IEEE P802.11 Wireless LANs WWiSE Proposal: High throughout extension to the 802.11 Standard*, IEEE (2004).
5) Mansour, M.: High-Throughput LDPC Decoders, *IEEE Trans. VLSI*, Vol.11, No.6 (2003).
6) Sharon, E.: Efficient Serial Message-Passing Schedules for LDPC Decoding, *IEEE Trans. Inform. Theory*, Vol.53, pp.4076–4091 (2007).
7) Sun, Y., Karkooti, M. and Cavallaro, J.R.: VLSI decoder architecture for high throughput, variable block-size and multi-rate LDPC codes, *IEEE 2007 ISCAS's Proceedings*, pp.2104–2107 (2007).
8) Brack, T.: A Synthesizable IP Core for WIMAX 802.16E LDPC Code Decoding, *IEEE 17th Inter. Symp. on Personal, Indoor and Mobile Radio Communications Proceedings*, pp.1–5 (2006).
9) Studer, C.: Configurable high-throughput decoder architecture for quasi-cyclic LDPC codes, *42nd ACSSC's Proceedings*, pp.1137–1142 (2008).
10) Liu, C.: An LDPC Decoder Chip Based on Self-Routing Network for IEEE 802.16e

Applications, *IEEE Journal of Solid-State Circuits*, Vol.43, pp.684–694 (2008).
11) Benes, V.E.: Optimal rearrangeable multistage connecting networks, *Journal of Bell Syst. Tech.*, Vol.43, No.4, pp.1641–1656 (1964).
12) Tang, J.: Reconfigurable shuffle network design in LDPC decoder, *2006 Int. Conf. ASAP's Proceedings*, pp.81–86 (2006).
13) Oh, D. and Parhi, K.K.: Low-Complexity Switch Network for Reconfigurable LDPC Decoders, *IEEE Trans. VLSI System*, Vol.18, No.1, pp.85–94 (2010).
14) Liu, C.: Multimode message passing switch networks applied for QC-LDPC decoder, *IEEE 2008 ISCAS's Proceedings*, pp.752–755 (2008).
15) Liu, C.: Design of a Multimode QC-LDPC Decoder based on Shift-Routing Network, *IEEE Trans. Circuits and Systems II: Express Briefs*, Vol.56, No.9, pp.734–738 (2009).
16) Oh, D. and Parhi, K.K.: Area efficient controller design of barrel shifters for reconfigurable LDPC decoders, *IEEE 2008 ISCAS's Proceedings*, pp.240–243 (2008).
17) Sun, Y.: A low-power 1-Gbps reconfigurable LDPC decoder design for multiple 4G wireless standards, *IEEE 2008 ISOCC's Proceedings*, pp.367–370 (2008).
18) Rovini, M.: Layered Decoding of Non-Layered LDPC Codes, *9th EUROMICRO DSD's Proceedings*, pp.537–544 (2006).
19) Fossorier, M.P.C.: Reduced complexity iterative decoding of low-density parity check codes based on belief propagation, *IEEE Trans. Commun.*, Vol.47, No.5, pp.673–680 (1999).
20) Chen, J.: Reduced-complexity decoding of LDPC codes, *IEEE Trans. Commun.*, Vol.53, No.8, pp.1288–1299 (2005).
21) Shao, R.Y., Lin, S. and Fossorier, M.P.C.: Two simple stopping criteria for Turbo decoding, *IEEE Trans. Commun.*, Vol.47, pp.1117–1120 (1999).
22) Shih, X., Zhan, C., Lin, C. and Wu, A.: An 8.29 mm2 52 mW Multi-Mode LDPC Decoder Design for Mobile WiMAX System in 0.13 um CMOS Process, *IEEE Jounal of Solid-State Circuites*, Vol.43, pp.672–683 (2008).

(Received December 1, 2009)
(Revised February 21, 2010)
(Accepted April 14, 2010)
(Released August 16, 2010)

(Recommended by Associate Editor: *Masanori Hariyama*)

Zhixiang Chen received his B.S. degree in Electronic Engineering from Shanghai Jiaotong University in 2007. In 2010, he received his M.E. degree from graduate school of Information, Production and Systems, Waseda University, Japan and currently he is working toward a Ph.D. degree there. His research interests include digital VLSI architecture for error correction systems in communication.



Xiongxin Zhao received his B.S. degree in Electronic Science and Technology from Tsinghua University in 2006. He received his M.E. degree from graduate school of Information, Production and Systems, Waseda University in 2010, and currently he is pursing his Ph.D. degree there. His research interests include wireless communication and error correcting codes.

Xiao Peng received his B.S. and M.S. degrees in Electronic Science and Technology from Tsinghua University in 2005 and 2008, respectively. He is currently a Ph.D. candidate in Graduate School of Information, Production and Systems, Waseda University, Japan. His research interests include wireless communication and error correcting code, especially the LDPC codes.

Dajiang Zhou received his B.S. and M.S. degrees from the Department of Electronic Engineering of Shanghai Jiaotong University, China, in 2005 and 2008, respectively. Since April 2008, he has been working toward a Ph.D. degree at the Graduate School of Information, Production and Systems of Waseda University, Japan. Since April 2009, he has been a research fellow of Japan Society of the Promotion of Science (JSPS). Currently, his main interest is in algorithms and VLSI architectures for multimedia and communication technologies.

Satoshi Goto was born on January 3rd, 1945 in Hiroshima, Japan. He received his B.E. and M.E. degrees in Electronics and Communication Engineering from Waseda University in 1968 and 1970, respectively. He also received his Dr. of Engineering from the same university in 1981. He is an IEEE fellow, Member of Academy Engineering Society of Japan and professor of Waseda University. His research interests include LSI System and Multimedia System.