Self-Adaptive Termination Check of Min-Sum Algorithm for LDPC Decoders Using the First Two Minima

Keol Cho[1] and Ki-Seok Chung[1]
[1] Hanyang University
222, Wangsimni-ro, Seongdong-gu, Seoul 133-791, - Korea
[e-mail: keolman2@gmail.com, kchung@hanyang.ac.kr]
*Corresponding author: Ki-Seok Chung

Received January 29, 2016; revised December 14, 2016; accepted February 6, 2017; published April 30, 2017

Abstract

Low-density parity-check (LDPC) codes have attracted a great attention because of their excellent error correction capability with reasonably low decoding complexity. Among decoding algorithms for LDPC codes, the min-sum (MS) algorithm and its modified versions have been widely adopted due to their high efficiency in hardware implementation. In this paper, a self-adaptive MS algorithm using the difference of the first two minima is proposed for faster decoding speed and lower power consumption. Finding the first two minima is an important operation when MS-based LDPC decoders are implemented in hardware, and the found minima are often compressed using the difference of the two values to reduce interconnection complexity and memory usage. It is found that, when these difference values are bounded, decoding is not successfully terminated. Thus, the proposed method dynamically decides whether the termination-checking step will be carried out based on the difference in the two found minima. The simulation results show that the decoding speed is improved by 7%, and the power consumption is reduced by 16.34% by skipping unnecessary steps in the unsuccessful iteration without any loss in error correction performance. In addition, the synthesis results show that the hardware overhead for the proposed method is negligible.

Keywords: Low-density parity-check codes, LDPC decoder, min-sum algorithm, adaptive min-sum decoding, error correction codes

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (R7119-16-1009, Development of Intelligent Semiconductor Core Technologies for IoT Devices based on Harvest Energy).

https://doi.org/10.3837/tiis.2017.04.009 **ISSN : 1976-7277**

1. Introduction

Due to their outstanding error correction performance and strong parallelism potential of the decoding process, low-density parity-check (LDPC) codes, which were first introduced by Gallarger in 1962 [1], have received a great deal of attention in the past few decades. LDPC codes have been adopted in various communication systems such as mobile broadcasting, satellite and wireless communications [2]-[6]. LDPC codes also have been adopted for error correction for solid-state disks based on NAND flash memory [7]-[9]. Since these applications commonly require higher throughput with lower power consumption, implementing efficient LDPC decoders has been actively studied in various ways including scheduling schemes [6] [10]-[12], decoding algorithms [13]-[15], and efficient hardware implementations of the decoder [7] [16].

The LDPC codes are uniquely defined by a parity check matrix, H, and the H matrix of binary LDPC codes can be graphically described by a bipartite graph called a Tanner graph [17] of check nodes (CNs) and variable nodes (VNs). These codes are decoded using message-passing algorithms, such as the sum-product (SP) and min-sum (MS) algorithms, which exchange information between CNs and VNs iteratively. The most popular decoding algorithm is the MS algorithm because of its low computational complexity with slight loss of coding gain. Furthermore, even a simple modification of the MS algorithm can reduce the loss of coding gain. Thus, modified versions [13]-[15] of the MS algorithm have been widely adopted in modern LDPC decoders. Moreover, dynamic MS algorithm approaches have been discussed in a lot of research for better performance and throughput [18] [19]. M. Jiang *et al.* [18], adjusted the scaling and offset factors adaptively, and J. Y. Park *et al.* [19] utilized a look-up table of the minimum iteration number for adaptive scheduling. However, these conventional works are based on estimation of the signal-to-noise ratio (SNR), which requires either complex computation or an external estimator.

In this paper, an adaptive scheduling approach of the MS algorithm without SNR estimation is proposed. Finding the first two minima from variable node to check node (V2C) messages is usually adopted when MS-based LDPC decoders are implemented in hardware due to the efficiency in memory usage [20]. B. Xiang *et al.* [21] showed that instead of sending the two found minima, the interconnection complexity and memory usage could be reduced by compressing check node to variable node (C2V) messages using the difference in the two minima. In this paper, a study has been carried out to find the correlation between the differences in the two minima and the number of decoding iterations. Decoding is never successfully terminated when these differences are bounded by small values. Therefore, it is possible to dynamically skip the termination-checking step when the difference in the two found minima is relatively small.

The remainder of this paper is organized as follows. In Section 2, characteristics of MS decoding algorithms and several optimization techniques are briefly introduced. Section 3 presents analysis results of the differences in the first two minima. The proposed self-adaptive MS algorithm is explained in detail and the experimental results are summarized in Section 4. The analysis of the hardware cost of the proposed algorithm and the solutions to overcome the overhead are addressed in Section 5. Finally, our conclusions are presented in Section 6.

2. Related Work

2.1 Min-Sum Algorithm

The SP (or belief propagation) algorithm has the most powerful decoding capability for LDPC codes [1]; however, high decoder complexity is a serious concern with this algorithm. Thus, many implementations of LDPC decoders are based on the MS algorithm and its modified versions because satisfactory error correction performance and relatively low design complexity can be achieved [15].

The MS algorithm consists of initialization, CN operation, VN operation, and termination check steps. Let the block length of the LDPC code be N, and $L^{(l)}_{i\to j}$ and $L^{(l)}_{j\to i}$ denote the log-likelihood ratio (LLR) information from VN i to CN j and that from CN j to VN i, respectively, at the l-th iteration. The set of VNs neighboring CN j is denoted as V_j, and the set of CNs neighboring VN i is denoted as C_i. At first, $L^{(l)}_{i\to j}$ of the VNs is initialized as follows:

$$L^{(0)}_{i\to j} = F_i = \frac{2y_i}{\delta^2} \tag{1}$$

where F_i is the initial LLR value (a priori LLR) derived from the received vector, y_i, and δ^2 is the noise variance. In some implementations, received vectors are directly used as the initial LLR. After initialization, CNs update C2V messages as follows:

$$L^{(l)}_{j\to i} = \prod_{i^{'}\in V_j\setminus i} \operatorname{sign}\left(L^{(l)}_{i^{'}\to j}\right)\cdot \min_{i^{'}\in V_j\setminus i}\left|L^{(l)}_{i^{'}\to j}\right| \tag{2}$$

where $V_j \setminus i$ represents the subset of VNs excluding the i-th VN. In modified versions of the MS algorithm, (2) is scaled by k which makes the error correction capability close to that of the SP algorithm [15]. In the VN operation, V2C messages are updated as follows:

$$L^{(l)}{}_{i\to j} = F_i + \sum_{j'\in C_i\setminus j} L^{(l)}{}_{j^{'}\to i} \tag{3}$$

where $C_j \setminus i$ represents the subset of CNs excluding the j-th CN. After VNs are updated, a posteriori LLR values, z_i, are calculated as in

$$z_i^{(l)} = F_i + \sum_{j\in C(i)} L^{(l)}{}_{j\to i} \tag{4}$$

The next step is termination checking. The termination-checking step consists of two operations: tentative decision and parity check operation. In the tentative decision, estimated codeword $\hat{\boldsymbol{c}} = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_N\}$ is constructed based on z_i by

$$\hat{\boldsymbol{c}} = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_N\}, \quad \hat{c}_i = \begin{cases} 0, & z_i^{(l)} \geq 0 \\ 1, & z_i^{(l)} < 0 \end{cases} \tag{5}$$

With these estimated codewords, the parity check operation is processed. If $\boldsymbol{H} \cdot \hat{\boldsymbol{c}}^T = 0$ or the number of iterations reaches the predefined maximum count, the estimated codeword, $\hat{\boldsymbol{c}}$, becomes the decoded word. When the termination check is not satisfied, the decoder goes back to the CN operation again with the updated LLR information.

2.2 Hardware Implementation of MS Algorithms

In a hardware implementation of the CN operation unit of an MS-based LDPC decoder, the sign calculation part in (2) can be implemented only with exclusive-OR gates. However, the minimum finding operation in (2) is not simple because the minimum should be found for all of the neighboring VNs excluding the *i*-th VN. Thus, the first two minimum values are found instead. C. K. Liau *et al*. [20] showed that the memory space of an LDPC decoder was reduced significantly with this method. Thus, the minimum finding part in (2) is replaced by

$$\min_{i' \in V_j \backslash i} \left| L_{i' \rightarrow j}^{(l)} \right| = \begin{cases} min2, & \text{if index of } min1 = i \\ min1, & \text{otherwise} \end{cases} \tag{6}$$

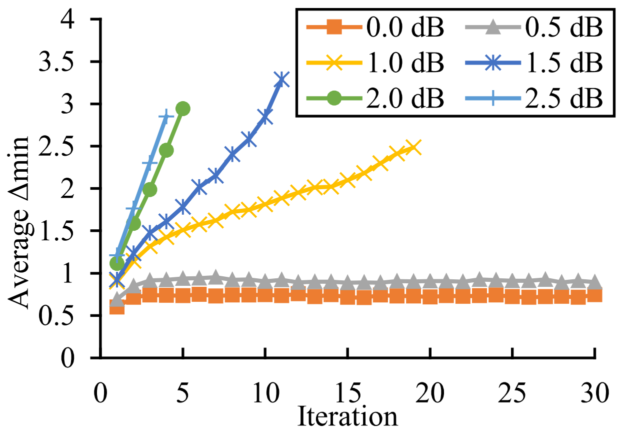
where *min1* and *min2* are the first and the second minimum, respectively. Therefore, C2V messages are formatted with four components: {signs of all output values, index of *min1*, *min1*, *min2*}.

2.3 C2V Message Compaction

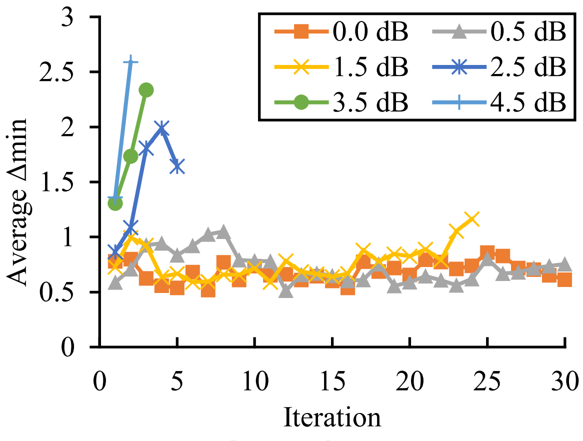
The block length of modern LDPC code applications ranges up to tens of thousands of bits, so LDPC decoders suffer from high interconnection complexity and a large memory space requirement. B. Xiang *et al.* [21] proposed a method which compressed C2V messages using the difference in the first two minimums to address these concerns. Instead of sending *min2*, C2V messages are compacted {signs of all output values, index of *min1*, *min1*, Δmin}, where Δmin is the difference between *min1* and *min2*. Using this compression, the memory space for C2V messages is reduced by 5.64% with negligible performance loss [21].

3. Analysis of Δmin

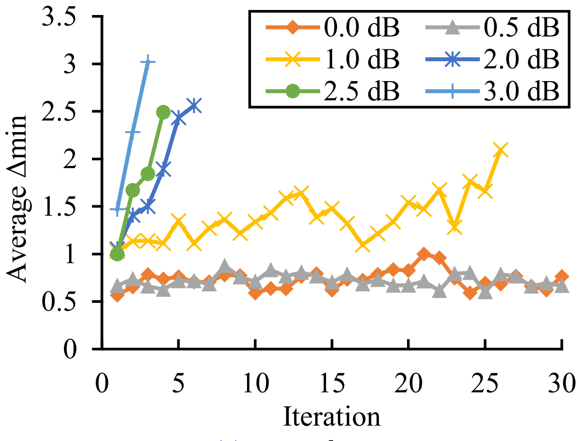
In this paper, the distribution of Δmin values in the C2V messages in each iteration was investigated with a varying SNR. LDPC codes with block lengths of 96 and 204 with a rate of 1/2 obtained from [22] and a block length of 9216 with a rate of 1/2 adopted by a mobile broadcasting system [3] were used for investigation. In this article, for the sake of brevity, these codes are named after their block length, such as 96 code, 204 code, and 9216 code. The maximum iteration count was set to 30, and the scaling factor, *k*, was set to 0.75, which showed the best error correction performance while an additive white Gaussian noise (AWGN) channel with various SNRs was chosen. In each decoding iteration, the average of all Δmin values from CNs was calculated for analysis, and 10,000 frames of LDPC codes were analyzed.



(a) 9216 code



(b) 96 code



(c) 204 code

Fig. 1. Average of Δmin values with various SNRs

Fig. 1 shows the analysis results of the 9216 code, 96 code, and 204 code in each iteration with various SNRs. In each iteration, the average of all Δmin values from overall CNs was calculated for analysis. Clearly, in **Fig. 1(a)**, the Δmin values are bounded when the decoding is unsuccessful as in the case of 0.5 dB, whereas the averages increase as the iterative decoding nears successful completion. As shown in **Fig. 1(b)** and **(c)**, the average of the Δmin values of the 96 code and that of the 204 code show similar characteristics to the 9216 code. The Δmin values are low-bounded values in a poor channel condition, as in the case of 0.5 dB, but these values are relatively high when decoding is successful. Furthermore, Δmin values increase as the SNR increases in successful decoding and iteration counts to complete the decoding process are much lower than the maximum iteration number in successful decoding.

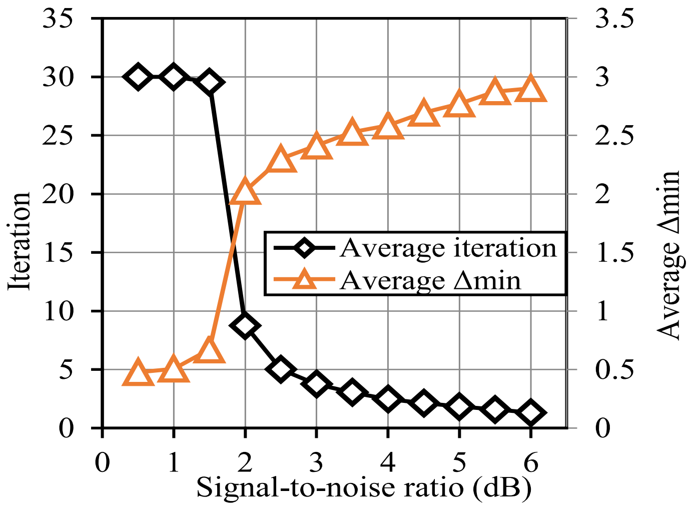


Fig. 2. Overall average Δmin values and average iteration numbers of 9216 code in various SNRs

Based on these results, the correlation between Δmin values and the iteration count was analyzed. By varying the SNR, the overall average of the Δmin values of the 9216 code was calculated in order to determine the correlation between the average difference and the average iteration count. The results of the analysis are depicted in **Fig. 2**. The Δmin values in **Fig. 2** are closely related to the iteration count of the LDPC decoder. Decoding in a low SNR region, which can be recognized by low Δmin values, leads to a high iteration number, while decoding under a good channel condition, which can be identified by high Δmin values, successfully terminates with a low iteration count. As stated in [23], the magnitude of V2C messages does not increase for undecodable codewords, whereas the magnitude grows for decodable codewords. Since C2V messages are determined by the magnitude of V2C messages (in equation (2)), the difference of the first two least C2V messages stays low despite of continuous iterative steps for unsuccessful decoding.

J. Y. Park *et al.* [19] reported that the average iteration number in successful decoding is much less than the maximum iteration count and utilized this characteristic for adaptive MS decoding. The minimum iteration count until successful decoding for various SNRs was stored in a look-up table. Utilizing a built-in SNR estimator, the LDPC decoder skips the termination check until the iteration count reaches the minimum iteration counter stored in the look-up table [19].

In this paper, more aggressive and flexible scheduling can be achieved by utilizing Δmin values than by utilizing SNR values. The magnitude of the Δmin value is not only a

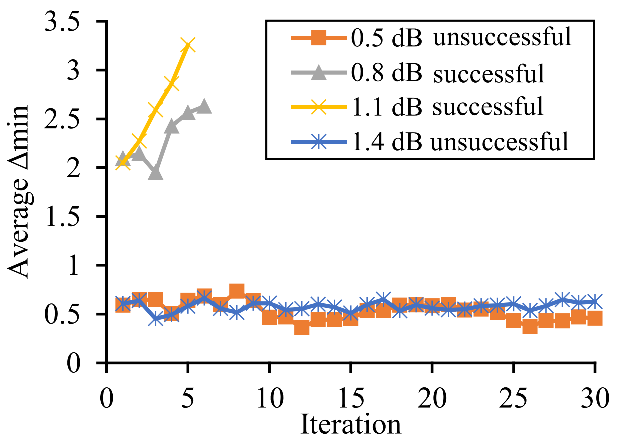


Fig. 3. Analysis of Δmin values of the 96 code with unsuccessful decoding

replacement of the SNR estimator in an adaptive low-power LDPC decoder [19], but it is also an indicator for success of decoding of the received message. Due to the random characteristics of channel impairment, a decoding failure may happen even in relatively good channel conditions. This situation cannot be handled by a static approach such as a look-up table, but the Δmin values can reflect the dynamically changing situation. **Fig. 3** shows the average of the Δmin values of the 96 code with varying SNRs. In the case of 1.4 dB, which is a relatively good channel condition, decoding was unsuccessful until the iteration reached the maximum iteration count. The average Δmin values of 1.4 dB are similar to those of 0.5 dB, which is the worst channel condition in this simulation. This result strongly implies that the Δmin value is an accurate criterion for predicting successful decoding.

4. Proposed Decoding Method with Experimental Results

4.1 Self-adaptive Min-sum Algorithm

Based on the analysis of the Δmin values, a self-adaptive MS algorithm without SNR estimation is proposed. Pseudocode for the proposed algorithm is shown in **Algorithm 1**. The proposed algorithm utilizes Δmin values, so C2V messages are formatted as devised by B. Xiang *et al*. For adaptive scheduling, an additional metric called '*delta-minima*' is devised in this paper. The *delta-minima* is a refined value of the Δmin values. The refinement of Δmin values can be implemented in various ways, such as straightforward summing of the Δmin values, saturation checking of the summation, or averaging of the Δmin values. In this article, the average of the Δmin values from all CNs was chosen as *delta-minima* (line 7 in **Algorithm 1**) to prove the validity of the proposed decoding algorithm.

In the proposed algorithm, *delta-minima* is calculated from C2V messages while the VN update operation is processed. When *delta-minima* is lower than $\Delta\text{min}_{\text{bound}}$ (line 8 in **Algorithm 1**), which is determined through extensive simulations, the decoder skips the termination check because LDPC decoding is very unlikely to be successfully terminated in that iteration. The termination check (line 12 and line 13 in **Algorithm 1**) consists of two steps: a hard decision to generate an estimated codeword and a parity check operation using the estimated codeword. Since the block lengths of modern LDPC code applications are very long, the termination check is a computationally demanding process in an LDPC decoder [19].

By skipping the unnecessary termination check, the proposed approach reduces power consumption. In addition, it does not affect the error correction performance of LDPC decoders.

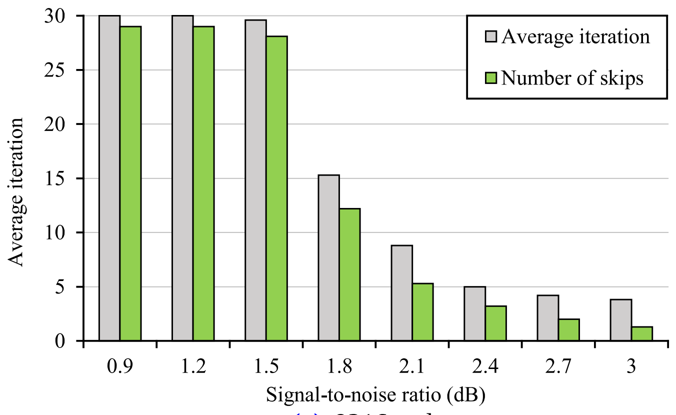
```
Algorithm: Self-adaptive MS algorithm

1: Initialize VNs, $L^{(0)}_{i\to j}$, with initial LLRs, $F_i$, derived from received vector $y_i$
        $L^{(0)}_{i\to j} = F_i = y_i$
2: for $l$ from 1 to max_iteration do
3:     {Check node update and construct C2V message}
        $L^{(l)}_{j\to i} = \prod_{i' \in V_j \setminus i} \mathrm{sign}\left(L^{(l)}_{i'\to j}\right) \cdot \min_{i' \in V_j \setminus i} \left|L^{(l)}_{i'\to j}\right|$
        $\mathrm{C2V}_{\mathrm{message}} = \{\mathrm{signs},\ min1_{\mathrm{index}},\ min1,\ \Delta\mathrm{min}\}$
4:     {Variable node update with delta-minima computation}
5:          $L^{(l)}_{i\to j} = F_i + \sum_{j' \in C_i \setminus j} L^{(l)}_{j'\to i}$
6:          $z_i^{(l)} = F_i + \sum_{j \in C(i)} L^{(l)}_{j\to i}$
7:          compute delta-minima using , $\Delta$min values of $\mathrm{C2V}_{\mathrm{message}}$
8:     if ( delta-minima < $\Delta\mathrm{min}_{\mathrm{bound}}$ and $l$ < max_iteration )
9:              $l = l + 1;$,
10:             Go to line 3;
11:    else
12:             $\hat{\boldsymbol{c}} = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_N\}, \quad \hat{c}_i = \begin{cases} 0, & z_i^{(l)} \ge 0 \\ 1, & z_i^{(l)} < 0 \end{cases}$
13:     if ( $\boldsymbol{H} \cdot \hat{\boldsymbol{c}}^T = 0$ or $l$ = max_iteration )
14:          Output $\hat{\boldsymbol{c}}$ as decoded bits
15:     else
16:          $l = l + 1;$
17:          Go to line 3;
```

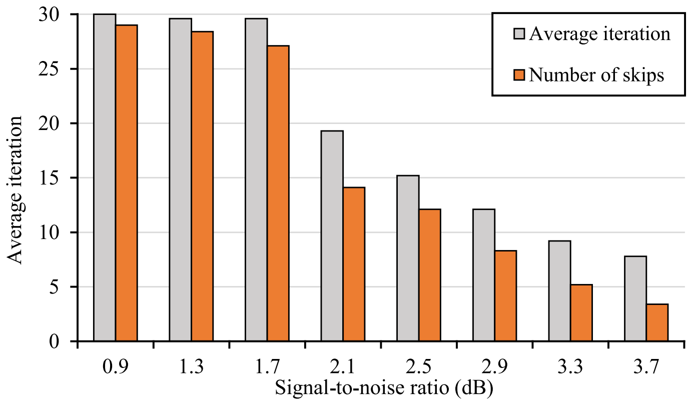
Algorithm 1. The proposed self-adaptive MS algorithm

4.2 Experimental Results of Termination Check Skipping

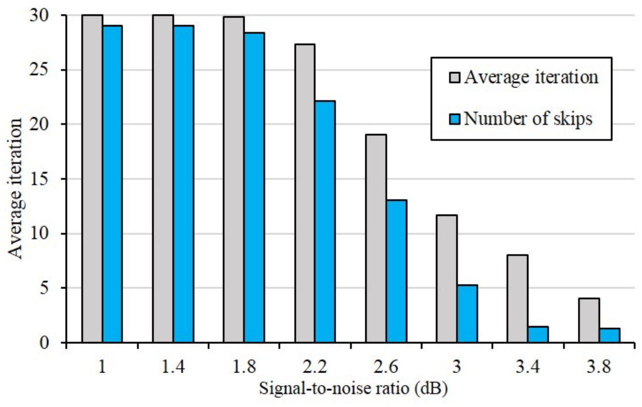
The proposed decoding algorithm was employed as a part of an LDPC decoder with the block length of 9216 and the code rate of 1/2. For the simulation, the average of the Δmin values is used as *delta-minima*, and $\Delta\mathrm{min}_{\mathrm{bound}}$ was chosen based on the Δmin analysis in Section 3. The choice of $\Delta\mathrm{min}_{\mathrm{bound}}$ is crucial for the proposed algorithm because this threshold affects both the average iteration count to complete the decoding and the number of skipped termination



(a) 9216 code



(b) 204 code



(c) 1296 code

Fig. 4. The number of skipped termination checks in various SNRs

checks. The results of Δmin analysis show that Δmin values lie between 0.5 and 1.0. Therefore, simulations have been carried out varying Δmin_{bound} from 0.5 to 1.0 to find the best Δmin_{bound}. As a result, Δmin_{bound} is set to 0.75 which merely affects the average iteration count to completion of decoding while achieving high termination check skipping rate.

By varying the SNR, the average iteration count to complete decoding a frame compared to the average skipped number of termination checks in case of *delta-minima* was lower than 0.75. The maximum iteration count was set to 30, and 1,000,000 frames were simulated. The results are summarized in **Fig. 4**. When the SNR was less than 1.5 dB in 9216 code, **Fig. 4(a)**, most termination checks except for the last iteration (line 13 in **Algorithm 1**) were skipped. Decoding messages transmitted through a poor channel condition was hardly successful; therefore *delta-minima* values below 1.5 dB were mostly lower than 0.75 as shown in **Fig. 1**. In comparison, almost half of the termination checks over 2.1 dB which is a waterfall region of the 9216 code were skipped, because *delta-minima* values were much larger than 0.75 and the iteration counts to complete the decoding were small in successful decoding. The simulation results when the proposed algorithm was employed to the 204 code and 1296 code (the length of 1296 with rate of 3/4 which is defined in 802.11n standard [4]) where $\Delta\text{min}_{\text{bound}}$ for both case was set to 0.75 are summarized in **Fig. 4(b)**, and the results verify that our algorithm is also effective in short codes and irregular code with different rate.

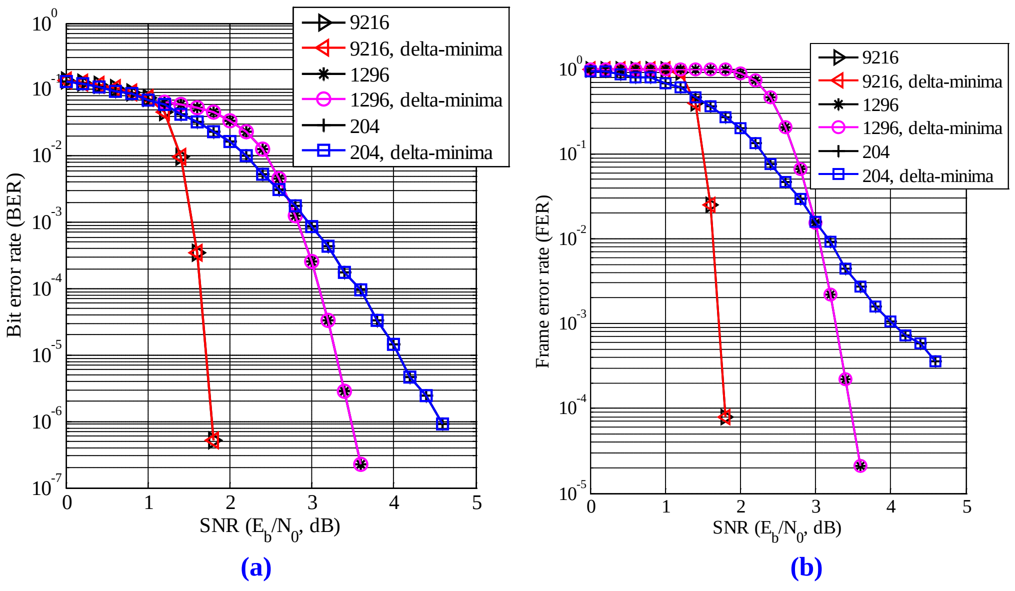


Fig. 5. Error correction performance comparison: (a) BER performance and (b) FER performance

4.3 Error Correction Performance

The error correction performance of the proposed decoding algorithm was investigated in the presence of AWGN with 30 maximum iteration count. Performance comparisons with the modified MS algorithm were conducted with respect to bit error rate (BER) and frame error rate (FER) were conducted when the proposed method was employed to the 204 code, the 1296 code, and the 9216 code. In both algorithms, scaling factor k was set to 0.75 which showed the best performance. As shown in **Fig. 5**, the proposed algorithm does not affect error correcting performance.

It is reported that the termination check would account for 17% of the power consumption of the total LDPC decoder in each iteration [19]. Thus, our proposed method reduced the power consumption of the LDPC decoder by up to 16.43% under bad channel conditions. By skipping unnecessary termination checks, the decoding time was also reduced by 7% in the case of low SNRs.

To figure out the advantage of the proposed algorithm in terms of hardware implementation, operational units of the LDPC decoder for the 9216 code were implemented and synthesized using a 0.18-µm CMOS cell library. The operational units include the VN operation unit, the termination check unit, and the CN operation unit designed with a tree structure [24]. The degrees of the CNs and VNs are 6 and 3, respectively, in the 9216 code with a rate of 1/2, and the termination check unit processes parity check operations with 6 inputs. The synthesis results are summarized in **Table 1**. The size of the termination check unit is quite small because it mostly consists of exclusive-OR gates, and its execution time is about 8.61% and 13.26% of those of the CN and the VN units, respectively. However, the termination check is quite a heavy operation considering that it includes memory accesses and parity check operations equal to the number of CNs, which can be up to 60,000 in the case of NAND flash applications. Skipping unnecessary termination checks becomes more advantageous when the block length is longer.

Table 1. Synthesis results of operational unit in LDPC decoder

	CN Unit	**VN Unit**	**Term. Check Unit**
Area (µm^2)	264,774.78	22,280.22	1,017.87
Delay (ns)	1.51	0.98	0.13

5. Consideration of Hardware Implementation of the Proposed Algorithm

Calculating *delta-minima* by averaging all of the Δmin values from all CNs requires excessive hardware costs and time considering that a single VN unit calculates the sum of dozens of operands. To reduce the hardware cost of the *delta-minima* computation, Δmin values are randomly taken from 4, 8, 16, 32, and 64 CNs, and the average of these values is computed as the approximated *delta-minima*. From the experiments, it is found that the average of more than eight sampled Δmin values provides a reasonable approximation to *delta-minima* and $\Delta\text{min}_{\text{bound}}$ for the proposed decoding scheme. **Fig. 6(a)-(c)** depict the 16, 32, and 64 sampled *delta-minima* values for the 9216 code, respectively, with various SNRs. Although the *delta-minima* values from fewer samples tend to reveal some inconsistencies compared to those from more samples, all of the sampled *delta-minima* values show obvious bounds in unsuccessful decoding.

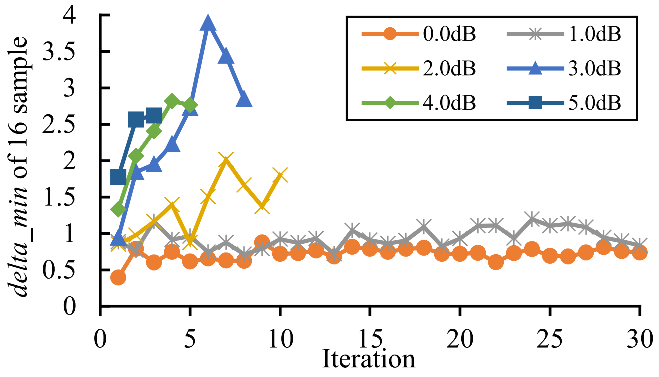


Fig. 6(a). 16 samples

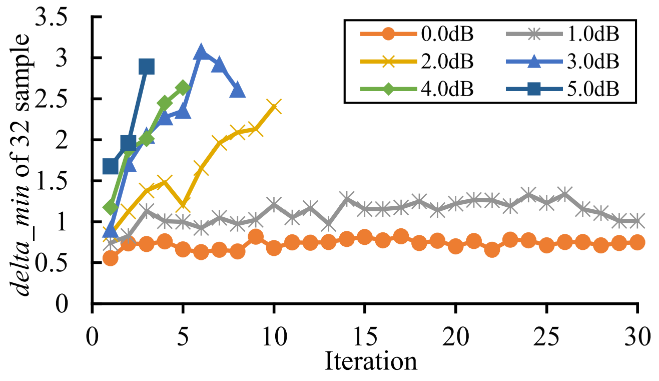


Fig. 6(b). 32 samples

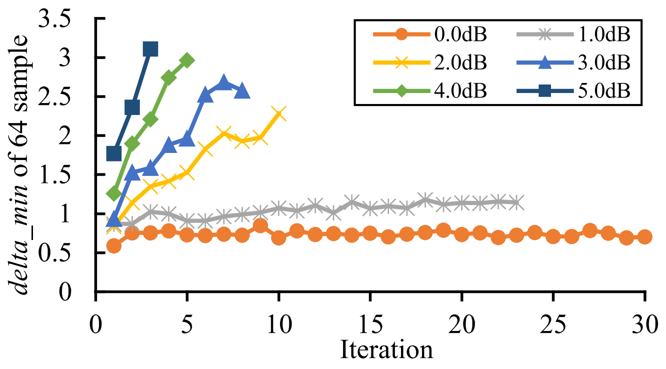


Fig. 6(c). 64 samples

Fig. 6. *delta-minima* based on Δmin values of various samples

The architecture of LDPC decoders can be classified into three architectures: fully-parallel, serial, and partially-parallel architectures. The fully parallel architecture, which has the same number of CN and VN units as the number of CNs and VNs in the Tanner graph, provides the best throughput, but suffers from the highest hardware cost. The minimal number of CN and VN unit can be achieved with the serial architecture, but the degradation of throughput is incurred. The partially-parallel architecture provides the competitive solution in terms of the area and the throughput of the LDPC decoder. In the partially-parallel architecture, a group of CNs is processed by a CN processing unit and a group of VNs is processed by a VN processing unit. Thus the numbers of CN and VN units in a partially-parallel LDPC decoder are much smaller than fully-parallel architecture ranging from 8 to 128 or more [16] [20] [21].

Implementing the computation unit of the *delta-minima* values out of sampled CNs is straightforward when an LDPC decoder is designed with the partially-parallel architecture. As shown in **Fig. 7**, the Δmin values of each CN unit are combined in the *delta-minima* computation module in the LDPC controller. The *delta-minima* computation module calculates the average of the sampled Δmin values, and then LDPC controller compares the calculated *delta-minima* with $\Delta\text{min}_{\text{bound}}$. After the comparison, LDPC decoder decides whether the termination check will be skipped or not. It means that only some additional wires and *delta-minima* calculation modules are required to implement the proposed algorithm.

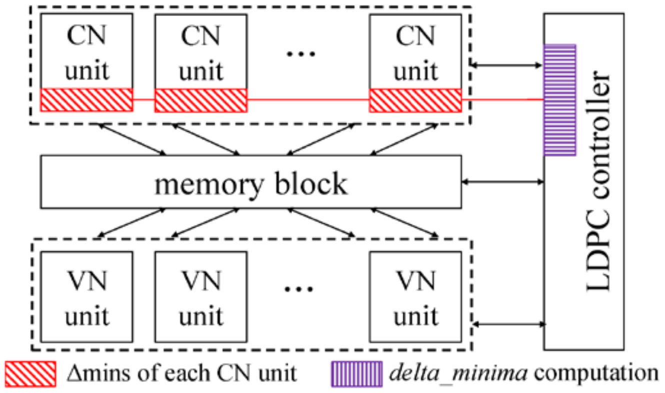


Fig. 7. Architecture of the self-adaptive MS-based LDPC decoder

For a more precise evaluation of the implementation hardware cost of the proposed algorithm, the *delta-minima* computation module, CN units, and VN units for a 16-level partially parallel LDPC decoder are implemented and synthesized using a 0.18-μm CMOS cell library. **Table 2** reports the area of the synthesized designs in terms of a NAND2 gate count. Considering that the SNR estimator takes up 12.1% of the area of the LDPC decoder [19], the hardware cost of the *delta-minima* computation module is negligible.

Table 2. Synthesis results of the operational unit and delta-minima module of the 16-level parallel LDPC decoder

	CN Unit	**VN Unit**	**delta-minima Unit**
Area (in NAND2 gate)	41975.71	3545.979	94.23

6. Conclusion

This paper proposed a self-adaptive approach for the MS algorithm using the difference between two minima (Δmin) that should be found in the MS algorithm. It was observed that the size of the Δmin values has a strong correlation with the success of decoding the received frame. Utilizing this characteristic, the proposed LDPC decoder dynamically decides whether the termination check will be skipped or not based on the Δmin value without external information, such as channel estimation. The proposed method is much more aggressive and accurate than the previous research and does not affect the error correction capability. Simulation results show that the proposed method improves the speed by 7% and reduces the power consumption by 16.43%. The synthesis results show that the additional hardware cost is negligible. Thus, the proposed self-adaptive algorithm can be very useful when a system is not able to accurately estimate channel conditions.

References

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Information Theory*, vol. 8, no. 1, pp. 21-28, Jan. 1962. Article (CrossRef Link)

[2] ETSI, "Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and other Broadband Satellite Applications," *EN 302 307, V1. 1. 1*, Jun. 2004. Article (CrossRef Link)

[3] W. Liang, W. Zhang, D. He, Y. Guan, Y. Wang, and J. Sun, "Digital Terrestrial Television Broadcasting in China," *IEEE MultiMedia*, vol. 14, no. 3, pp. 92-97, July-Sept, 2007. Article (CrossRef Link)

[4] IEEE P802.11n/TM-2009, "IEEE standard for information technology part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications," Oct. 2009. Article (CrossRef Link)

[5] D. Qu, L. Li, and T. Jiang, "Invertible subset LDPC code for PAPR reduction in OFDM systems with low complexity," *IEEE Transactions on Wireless Communications*, vol. 13, no. 4, pp. 2204 - 2213, Apr. 2014. Article (CrossRef Link)

[6] L. Li, D. Qu, and T. Jiang, "Partition optimization in LDPC-coded OFDM systems with PTS PAPR reduction," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 8, pp. 4108-4113, Oct. 2014. Article (CrossRef Link)

[7] J. Kim and W. Sung, "Rate-0.96 LDPC decoding VLSI for soft-decision error correction of NAND flash memory," *IEEE Trans. VLSI*, vol. 22, no. 5, pp. 1004–1015, May 2014. Article (CrossRef Link)

[8] L. Kong, J. Wen, G. Han, S. Zhao, M. Jiang, and C. Zhao, "Quantization and reliability-aware iterative majority-logic decoding algorithm for LDPC code in TLC NAND flash memory," in *Proc. of 2016 8th International Conference on Wireless Communications & Signal Processing (WCSP)*, pp. 1–5, Oct. 2016. Article (CrossRef Link)

[9] K.-C. Ho, C.-L. Chen, and H.-C. Chang, "A 520k (18900, 17010) array dispersion LDPC decoder architectures for NAND flash memory," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 4, pp. 1293–1304, Apr. 2016. Article (CrossRef Link)

[10] J. Zhang, Y. Wang, M. Fossorier, and J. S. Yedidia, "Replica shuffled iterative decoding," *IEEE Int. Symp. on Information Theory*, Adelaide, Australia, pp. 454–458, Sep. 2005. Article (CrossRef Link)

[11] J. H. Kim, M. Y. Nam, and H. Y. Song, "Variable-to-check residual belief propagation for LDPC codes," *IET Electronics Letters*, vol. 45, no. 2, pp. 117-118, Jan. 2009. Article (CrossRef Link)

[12] A. Anand and P. S. Kumar, "An efficient non binary LDPC decoder using layered dynamic scheduling," *J. Comput. Theor. Nanosci.*, vol. 12, no. 12, pp. 5066–5070, Dec. 2015. Article (CrossRef Link)

[13] K. Zhao, Y. Xu, D. He, Y. Guan, and W. Zhang, "Variable LLR scaling in LDPC min-sum decoding under horizontal shuffled structure," in *Proc. of 2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–7, Jun. 2016. Article (CrossRef Link)

[14] M. K. Roberts and R. Jayabalan, "An improved self adaptive min-sum decoding algorithm for flexible low-density parity-check decoder," *Natl. Acad. Sci. Lett.*, pp. 1–5, Nov. 2016. Article (CrossRef Link)

[15] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low density parity check codes," *IEEE Trans. Commun.*, vol. COM-50, no. 3, pp. 406-414, Mar. 2002. Article (CrossRef Link)

[16] S. Kim, C. Park, and S. Hwang, "A novel partially parallel architecture for high-throughput LDPC decoder for DVB-S2," *IEEE Trans. Consumer Electron.*, vol. 56, no. 2, pp. 820-825, May. 2010. Article (CrossRef Link)

[17] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, Sept. 1981. Article (CrossRef Link)

[18] M. Jiang, C. Zhao, L. Zhang, and E. Xu, "Adaptive offset min-sum algorithm for low-density parity check codes," *IEEE Commum. Lett.*,vol. 10, no. 6, pp. 483–485, June 2006. Article (CrossRef Link)
[19] J. Y. Park and K. S. Chung, "An adaptive low-power LDPC decoder using SNR estimation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1-9, July 2011. Article (CrossRef Link)
[20] C. K. Liau, S. Y. Lin, T. H. Tsai, and C. L. Wey, "A partially parallel low-density parity check code decoder with reduced memory for long code-length," in *Proc. of 18th VLSI Des./CAD Symp.*, Hua-Lien, Taiwan, Aug. 2007. Article (CrossRef Link)
[21] B. Xiang, R. Shen, A. Pan, D. Bao, and X. Zeng, "An area-efficient and low-power multirate decoder for quasi-cyclic low-density parity-check codes," *IEEE Trans. Very Large Scale Integration Systems*, vol. 18, no. 10, pp. 1447-1460, Sep. 2010. Article (CrossRef Link)
[22] D. J. C. MacKay "Encyclopedia of sparse graph codes," *Cavendish Laboratory*, University of Cambridge, 2005. Article (CrossRef Link)
[23] F. Kienle and N. Wehn, "Low complexity stopping criterion for LDPC code decoders," in *Proc. of IEEE 61st Vehicular Technology Conference*, Stockholm, Sweden, May 2005. Article (CrossRef Link)
[24] C. L. Wey, M. D. Shieh, and S. Y. Lin, "Algorithms of finding the first two minimum values and their hardware implementation," *IEEE Trans. Circuits Syst. I,* vol. 55, no. 11, pp. 3430–3437, Dec. 2008. Article (CrossRef Link)

Keol Cho received his B.S. degree in Media Communication Engineering from Hanyang University, Seoul, Korea in 2009. Since 2009, he has been enrolled in a unified M.S. and Ph.D. course at Hanyang University, Seoul, Korea. His research interests include system-on-chip architecture, error-correction codes, and hardware implementation of error-correction codes.

Ki-Seok Chung received his B.E. degree in Computer Engineering from Seoul National University, Seoul, Korea, in 1989 and his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 1998. He was a Senior R&D Engineer at Synopsys, Inc. in Mountain View, CA, from 1998 to 2000 and was a Staff Engineer at Intel Corp. in Santa Clara, CA, from 2000 to 2001. He also worked as an Assistant Professor at Hongik University, Seoul, Korea, from 2001 to 2004. Currently, he is a Professor at Hanyang University, Seoul, Korea. His research interests include low-power embedded system design and system-on-chip designs for communication and multi-media applications.