Selection-based Low-cost Check Node Operation for Extended Min-Sum Algorithm

Kyeongbin Park and Ki-Seok Chung*
Hanyang University
222, Wangsimni-ro, Seongdong-gu, Seoul 133-791, Korea
[e-mail: lay1523@naver.com, kchung@hanyang.ac.kr]
*Corresponding author: Ki-Seok Chung

Received June 17, 2020; revised December 9, 2020; revised January 18, 2021; accepted February 11, 2021; published February 28, 2021

Abstract

Although non-binary low-density parity-check (NB-LDPC) codes have better error-correction capability than that of binary LDPC codes, their decoding complexity is significantly higher. Therefore, it is crucial to reduce the decoding complexity of NB-LDPC while maintaining their error-correction capability to adopt them for various applications. The extended min-sum (EMS) algorithm is widely used for decoding NB-LDPC codes, and it reduces the complexity of check node (CN) operations via message truncation. Herein, we propose a low-cost CN processing method to reduce the complexity of CN operations, which take most of the decoding time. Unlike existing studies on low complexity CN operations, the proposed method employs quick selection algorithm, thereby reducing the hardware complexity and CN operation time. The experimental results show that the proposed selection-based CN operation is more than three times faster and achieves better error-correction performance than the conventional EMS algorithm.

Keywords: Non-binary Low-density Parity Check, Extended Min-sum Algorithm, Hardware Optimization

This research was supported by Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government(MOTIE) (N0001883, The Competency Development Program for Industry Specialist)

http://doi.org/10.3837/tiis.2021.02.006 **ISSN : 1976-7277**

1. Introduction

Low-density parity-check (LDPC) codes have demonstrated powerful error-correction performance for data-communication systems. Their error-correction performance is close to the Shannon limit, if the code length is large. However, when the code length is small, their performance is not satisfactory. NB-LDPC codes over GF(q) with $q > 2$ have better error-correction capability than that of binary LDPC codes, especially when the code length is small. However, NB-LDPC codes perform satisfactorily at the expense of a significantly increased decoding complexity. Several studies were performed on the decoding algorithms for NB-LDPC codes. Although belief propagation [1] and the Fourier domain decoding algorithm [2] show high error-correction performances, their decoding complexities are significantly high. Symbol-flipping algorithms offer low decoding complexity but significantly degraded error-correction performance [3, 4]. To date, the extended min-sum (EMS) algorithm has achieved the best trade-off between decoding complexity and error-correction performance [5]. The belief information of the propagated messages in the EMS algorithm is represented by log-likelihood-ratio (LLR), which is a metric in the logarithm domain. It offers several advantages for reducing the computation complexity; this is because multiplications are replaced by additions in the logarithm domain, and the normalization step can be eliminated [6]. To further reduce the computation complexity, two methods have been proposed, i.e., truncating the messages and restricting the number of the most likely symbols to consider [5].

In a check node (CN) operation of the EMS algorithm, the incoming messages of length q from variable nodes (VNs) are truncated to the most reliable n_m ($n_m << q$) symbols; this step is called message truncation (MT). In addition to MT, the restriction on the number of the most likely symbols is also applied in the CN operation. Thus, the CN operation now comprises 1 to n_c message vectors ($n_c < d_c - 1$, where d_c denotes the number of VNs connected with CN, i.e., CN degree), and the rest $d_c - 1 - n_c$ message vectors are fixed to the most likely symbol of each message vector. For readability, the restriction on the number of the most likely symbols is denoted by SR in this study (see Section 2.3). Although these two methods may theoretically reduce the decoding complexity, they may not be appropriate to implement an NB-LDPC decoder. For example, SR incurs significant control and area overhead when an EMS-algorithm-based decoder is implemented in hardware (HW). These two complexity-reduction methods also require incoming message vectors to be sorted by the reliability of symbols in the CN operation. The sorting of message vectors accounts for most of the computational complexity of the CN operation.

To reduce the computational complexity of the CN operation, a Trellis-based EMS (T-EMS) algorithm has been proposed, which is appropriate for HW implementation [7, 8]. Although the T-EMS algorithm does not require message sorting, the amount of computation per symbol in each CN is high. Moreover, the decoding complexity of the T-EMS algorithm becomes significantly high when q increases ($q \geq 64$) [7]. Recently studied decoding algorithms, which revises T-EMS algorithm, even aims to reduce the amount of computation while minimizing the decrease in error-correction performance [9-12]. Meanwhile, Y. Hwang et al. proposed a heap-based message-sorting method, wherein a complete binary tree was used to reduce the computation complexity of message sorting [13]. By employing heap data structure, the messages were stored as nearly sorted; however, the error-correction performance degraded because less reliable symbols could also be included in the truncated message vectors.

To overcome the trade-off that error-correction performance and decoding complexity, we propose a low-cost CN (LC CN) operation to be implemented in the EMS algorithm. Different

with other studies that reduce decoding complexity at the cost of error-correction performance, the proposed algorithm has shown the result of improving error-correction performance while reducing decoding complexity. Compared with the original EMS algorithm, the proposed operation offers two advantages, i.e., significantly lower cost of the CN operation and more appropriateness for HW implementation. In the proposed CN operation, we decrease the computation complexity of MT using the quick selection algorithm, which can find the k^{th} largest or smallest element in O(N), where N denotes the number of elements. Unlike the previously reported partially sorted heap data structure [13], the quick selection algorithm can select exactly n_m reliable symbols. Experimental results show that the proposed LC CN operation simultaneously reduces the execution time and increases the decoding capability.

The rest of this paper is organized as follows. In Section II, we review the EMS algorithm for NB-LDPC codes. The proposed low-cost EMS algorithm is described in Section III. The experimental results on the decoding performance and computation complexity of the proposed algorithm are presented in Section IV. Finally, the conclusions are drawn in Section V.

2. Decoding Algorithm for Non-binary LDPC Codes

2.1 Preliminaries

An NB-LDPC code is defined using a sparse $M \times N$ parity-check matrix H, whose non-zero elements belong to a finite field, i.e., GF(q) [14]. Let $\boldsymbol{c}$ be the NB-LDPC code with an H-matrix, which is defined as follows:

$$\boldsymbol{c} = \{c \in GF(q)^N | Hc^T = 0\},$$

where T denotes transpose. The codeword $\boldsymbol{c}$, whose length is N, is of the form $\boldsymbol{c} = [c_0, \ldots, c_{N-1}]$, where c_j ($j \in [0, N-1]$) denotes a symbol in the finite field GF(q). Additionally, the code rate of $\boldsymbol{c}$ is defined as $R \leq (N-M)/N$. The i^{th} ($i \in [0, M-1]$) parity-check equation is written as follows:

$$\sum_{j:h_{i,j}\neq 0} h_{i,j} c_j = 0. \tag{1}$$

The H-matrix can be visualized using a Tanner graph, which has N varible nodes and M CNs.

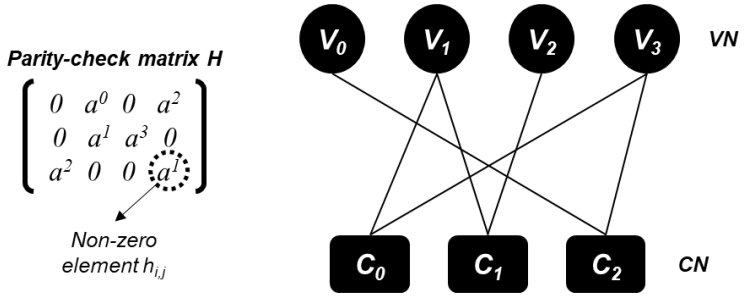


Fig. 1. Tanner-graph representation of parity-check matrix H with GF(4)

Fig. 1 shows an example of an *H*-matrix and its Tanner-graph representation. The positions of the non-zero elements in the *H*-matrix are represented by the edges between CNs and VNs in the Tanner graph, and the degree of a node is the number of edges incident on that node. The degree of a VN and CN is denoted by d_v and d_c, respectively.

The decoding of NB-LDPC codes is based on an iterative message-passing algorithm. In each decoding iteation, the message vectors that comprise the belief information for each symbol are propagated between CNs and VNs.

2.2 EMS Algorithm

In the EMS algorithm, which is a log-approximated version of a message-passing algorithm, the message vectors are updated and exchanged between CNs and VNs iteratively. The message vectors in the EMS algorithm are represented using a metric called LLR. The decoding iteration of the EMS algorithm initiates the VN *n* to CN *m* message vectors $U_{m,n}(a)$ ($m \in [0, M-1]$, $n \in [0, N-1]$, $a \in \mathrm{GF}(q)$) with the *a-priori information* of the channel LLR vectors, $L_n(a)$; the *a-priori information* includes the belief information of the received codeword and is defined as follows:

$$L_n(a) = ln\frac{P(c_n=0|channel)}{P(c_n=a|channel)}, \tag{2}$$

$$U_{m,n}(a) = L_n(a). \tag{3}$$

From (2), it is evident that the smaller is the value of $L_n(a)$, the higher is the reliability of symbol *a*. After initialization, the iterative process begins as follows :

1) CN operation: Update the CN *m* to VN *n* message vectors $V_{m,n}(a)$.

$$V_{m,n}(a) = \min_{\substack{a_{n'} \in H(m)\backslash\{n\} \\ \in C(m|a_n = a)}} \sum_{n' \in H(m)\backslash\{n\}} U_{m,n'}(a_{n'}), \tag{4}$$

where $C(m)$ denotes the set of incoming symbols to CN *m* that satisfies (1), and $C(m|a_n = a)$ denotes that the subset of $C(m)$ that the symbol from VN *n* is *a*. Additionally, $H(m)\backslash\{n\}$ denotes the set of VNs adjacent to CN *m*, excluding VN *n*. The cardinalities of $\boldsymbol{a}_n$ and $\boldsymbol{a}_{n'}$ are denoted by *q* and n_m, respectively, because of MT.

2) VN operation: Update the VN *n* to CN *m* message vectors $U_{m,n}(a)$.

$$U'_{m,n}(a) = L_n(a) + \sum_{m' \in H(n)\backslash\{m\}} U_{m',n}(a), \tag{5}$$

$$U_{m,n}(a) = U'_{m,n}(a) - U'_{m,n}(0), \tag{6}$$

where $H(n)\backslash\{m\}$ denotes the set of CNs adjacent to VN *n*, excluding CN *m*. Since (6) computationally stabilizes the decoding process, we need not normalize the exchanged messages [15].

3) Calculate the *a-posteriori* information and tentative decision as follows:

$$\hat{L}_n(a) = L_n(a) + \sum_{m \in H(n)} U_{m,n}(a), \tag{7}$$

$$s_n = argmin_{a \in GF(q)} \hat{L}_n(a). \tag{8}$$

The most reliable symbols of each VN ($\boldsymbol{s} = [s_0, \ldots, s_{N-1}]$) is determined using (8), and if $H \times \boldsymbol{s}^T = 0$, the decoding iteration is terminated. Otherwise, return to 1).

2.3 MT and SR of CN

Calculating the $V_{m,n}(a)$ in 2) consumes most of the execution time of the EMS algorithm. Therefore, MT and SR should be applied to reduce the complexity of CN update [5].

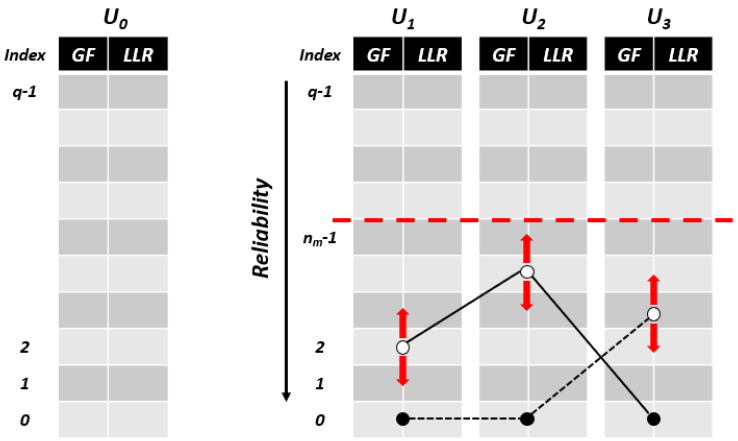


Fig. 2. Example of single CN update when $n_c = 2$ and $d_c = 4$

Fig. 2 shows an example of applying MT and SR in the CN update. To update a message vector U_0, the input message vectors U_1, U_2, and U_3 are sorted by the reliability of each symbol in ascending order. After the sorting, the only n_m reliable components in each message vector under the red dotted line in **Fig. 2** are considered to CN operation (as mentioned above, it is called MT). Subsequently, the most reliable symbol is located at the bottom of each message vector because each message vector has been sorted.

The CN update of single message vector operates with the combinations of symbols in n_c message vectors and the most reliable symbol in the other $(d_c - 1) - n_c$ message vectors. For example, when $n_c = 2$ and $d_c = 4$, the combinations of symbols from up to 2 message vectors and most reliable symbol from the rest are considered for the CN updates. Assume that the white circles in **Fig. 2** can move up and down to select a symbol, but the black circles, conversely, are fixed at the most reliable symbol at the bottom. The dotted line in **Fig. 2** is an example of CN operation with a combination of symbol in U_3 (1 white circle) and two most reliable symbols in U_1 and U_2 (2 black circles). The solid line shows another example that considers symbols in U_1 and U_2 (2 white circles) with one most reliable symbol in U_3 (1 black circle). Conversely, for a CN update, a combination of the fixed most reliable symbols (denoted by black circles) and n_c symbols in the other messages (denoted by white circles) are

considered, as shown in **Fig. 2**. In this paper, we define the combinations for computing a single message vector as a search area as follows:

$$\sum_{i=1}^{n_c} \binom{d_c-1}{i} * {n_m}^i \tag{9}$$

By including the most reliable symbol from some of the incoming messages, reliable results can be obtained, even if a limited number of combinations is considered. However, MT sorts all the message vectors incoming to a CN in ascending order according to the reliability of the symbols, and SR incurs the control or circuit area overhead.

The CN of the EMS algorithm is implemented in HW using a forward/backward (FWBW) strategy [6] or syndrome-based operation [16]. **Fig. 3** shows the CN with the recursive FWBW strategy based on a function block called elementary CN (ECN). In **Fig. 3**, the following six possible combinations exist for choosing the most reliable symbol from the incoming message vectors when updating U_0: U_1, U_2, U_3, (U_1, U_2), (U_1, U_3), and (U_2, U_3). Thus, CN has to update six times when it updates each outgoing message vector, and if d_c or n_c increases, performing a CN update will incur significant delay in control and computation. A syndrome-based CN operation that was appropriate for HW implementation has been reported previously [16], achieving a high degree of parallel processing of CN operations. However, they employed an additional processing unit for SR, and thus a costly area overhead was incurred when d_c or n_c increased.

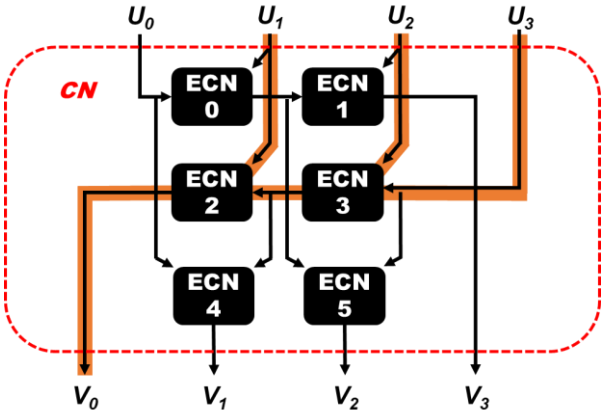


Fig. 3. CN update using the recursive FWBW strategy when $d_c = 4$

3. Proposed LC CN Operation

To reduce the SR overhead of a CN update, we discard SR, and MT is implemented via quick selection rather than sorting.

3.1 CN Update Without RC

To reduce the SR overhead of a CN update, all the possible combinations in the truncated message vectors are considered in the proposed CN operation. **Fig. 4** shows the combination for calculating U_0. Because the proposed LC CN calculates U_0 by using all components in truncated message vectors U_1, U_2, and U_3, 3 white circles with solid line in **Fig. 4** move. Unlike in (8), the search area of the proposed method will be changed into (10).

$$n_m{}^{d_c-1} \tag{10}$$

Although the number of combinations to consider becomes higher than that in the conventional CN operation with SR, an overhead for SR is discarded from the perspective of HW implementation.

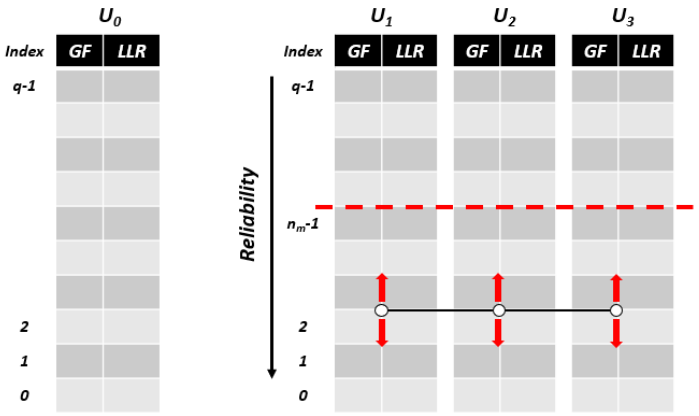


Fig. 4. Example of single CN update without RC when d_c = 4

To compare the execution times and hardware utilizations of conventional CN and proposed LC CN, we implemented a single CN with the FWBW strategy on FPGA using high-level synthesis (HLS). We assumed that the incoming message vectors were previously truncated, and the execution times including MT will be discussed in Section 4. Xilinx Zynq board (zc-104) was used, and all the CN units were equally optimized via loop unrolling and loop pipelining in all the experiments.

The implemented CN units with and without SR successfully operated with the clock frequency of 515 MHz. The experimental results in **Table 1** show that the CN operation without SR is 3.7 times faster than that with SR, when n_m = 8 and d_c = 4 with GF(64). The number of utilized HW resources is also reduced in the case of the CN operation without SR; this is because SR requires HW resources only for changing the addresses of the most reliable symbols.

Table 1. Synthesis results of a single CN unit for the compared EMS algorithms.

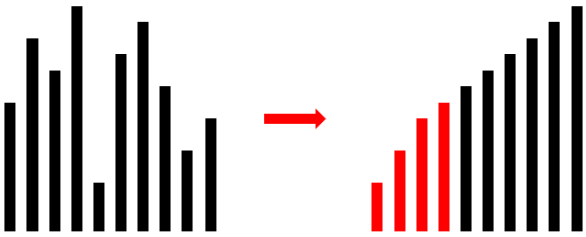
CN unit		**CN (n_c = 2)**	**LC CN**
Execution time (ms)		0.655	0.176
HW resources (Utilized / Total)	LUT	3552 / 230400	3318 / 230400
	FF	4478 / 460800	3979 / 460800
	BRAM	4 / 312	4 / 312

Another advantage of the proposed LC CN operation is its error-correction performance. Because SR discards the information of less reliable symbols, its error-correction capability is lower than that of the original EMS algorithm. However, the proposed operation does not exclude the information of less reliable symbols within the truncated message vector.

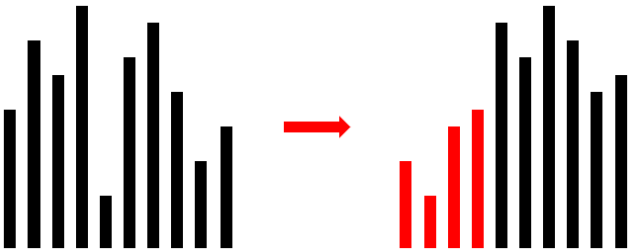
3.2 Quick Selection for MT

In the EMS algorithm, the most reliable symbol is located at the bottom of the message vector because MT is implemented via sorting, as shown in **Fig. 2**. Thus, it becomes easy to address the most reliable symbol because it will have index 0. Various sorting algorithms, including bubble, insertion, heap, merge, selection, and quick, have different complexities. Notably, the complexity of a sorting algorithm is O($n$log(n)) in the best case, when n denotes the number of elements to be sorted. There are no other works that focus on the message sorting for MT except [13]. To reduce the complexity of message sorting, a heap-based message-vector truncation has been proposed, wherein the heap was a partially sorted tree-based data structure [13]. However, the heap-based truncation had low error-correction performance because less reliable symbols could be included in the truncated message vectors.

However, since the proposed LC CN operation does not require SR, the message vector need not be perfectly sorted for performing MT, and especially, the most reliable symbol does not have to have index 0. In this study, to reduce the message-sorting overhead while achieving the same error-correction performance as that of the EMS algorithm, we propose a method based on the quick selection algorithm, which finds the k smallest (or largest) element in an unordered list. Notably, the average complexity of the quick selection algorithm is O(n). The difference between sorting algorithms and quick selection algorithm is that the latter does not sort the input unordered list but selects exactly k smallest (or largest) elements in the list. **Fig. 5** shows the difference between sorting and selection. Assuming a situation where we need to truncate the 4 smallest elements, unordered list at left side in **Fig. 5(a)** is perfectly ordered by sorting and truncates 4 elements in **Fig. 5(a)**. In **Fig. 5(b)**, however, the unordered list at left side does not be ordered, but quick selection could select exactly 4 elements. Because the proposed CN operation considers all the combinations of the truncated message vectors, as shown in **Fig. 4**, it requires n_m reliable symbols in each truncated message vector irrespective of the order. Therefore, the quick selection algorithm is appropriate for the proposed CN operation.



(a) Sorting



(b) Selecting the four smallest elements

Fig. 5. Difference between sorting and selection.

We compared the execution time and HW utilization of the quick selection algorithm with those of quick sort and heap sort, respectively (see **Table 2**). Quick sort is a recursive algorithm that employs the divide-and-conquer approach. It is one of the fastest sorting algorithms, with the average time complexity of O($n$log(n)). Heap sort combines the advantages of merge sort and insertion sort, and its average time complexity is also O($n$log(n)). It exactly sorts an unordered list unlike the partial sorting used in the heap-based data-structure method [13].

Table 2. Synthesis results of the sorting and selection algorithms for MT

MT unit		Heap sort	Quick sort	Quick selection
Execution time (ms)		0.205	0.136	0.135
HW resources (Utilized / Total)	LUT	822 / 230400	1009 / 230400	827 / 230400
	FF	687 / 460800	891 / 460800	656 / 460800
	BRAM	2 / 312	3 / 312	1 / 312

Table 2 lists the evaluation results of the HW implementation of a single MT unit for both sorting and selection algorithms, respectively. Evidently, the quick selection algorithm utilized fewer HW resources than both heap sort and quick sort. Furthermore, the execution time of the quick selection algorithm was the shortest. Therefore, the proposed selection-based method is highly effective. In conclusion, the proposed LC CN operation truncates the message vector via the quick selection algorithm and updates the outgoing messages without using SR.

4. Experimental Results

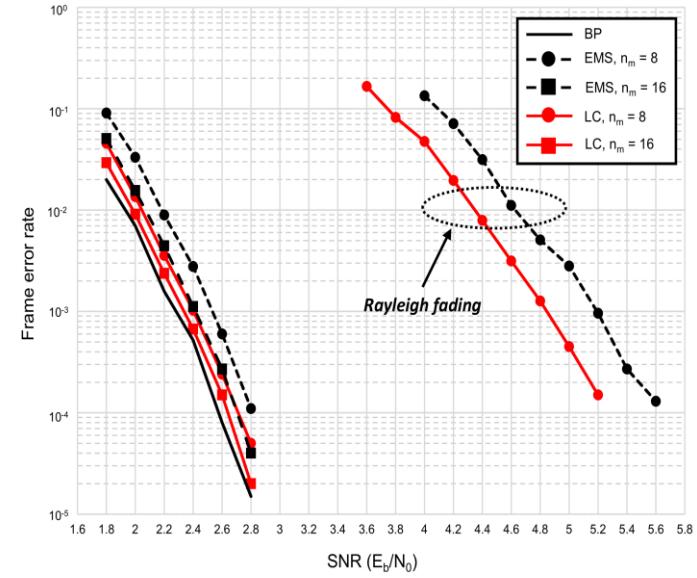
Various experimental results of the proposed selection-based LC CN operation are presented in this section. First, the error-correction capability of the proposed method is compared with that of belief propagation, which is state-of-the-art decoding algorithm from the viewpoint of error-correction performance, and that of the conventional EMS algorithm. Second, average iteration count of each decoding algorithm are compared. Third, hardware utilization of conventional CN and LC CN are compared. Because the error-correction performance of the proposed algorithm is independent of the decoding-scheduling method applied, we employed the flooding schedule approach in the experiments [17]. In all the experiments, we assumed additive white Gaussian noise channels when binary phase-shift keying (BPSK) or quadrature amplitude modulation (QAM) were employed, and with/without rayleigh fading was also employed. In the experimental result in QAM modulation, error floor region with channel fading is relatively longer than without fading. Accordingly, the experimental results of 256-QAM modulation with channel fading was excluded.

4.1 Frame Error Rate (FER) and Bit Error Rate (BER)

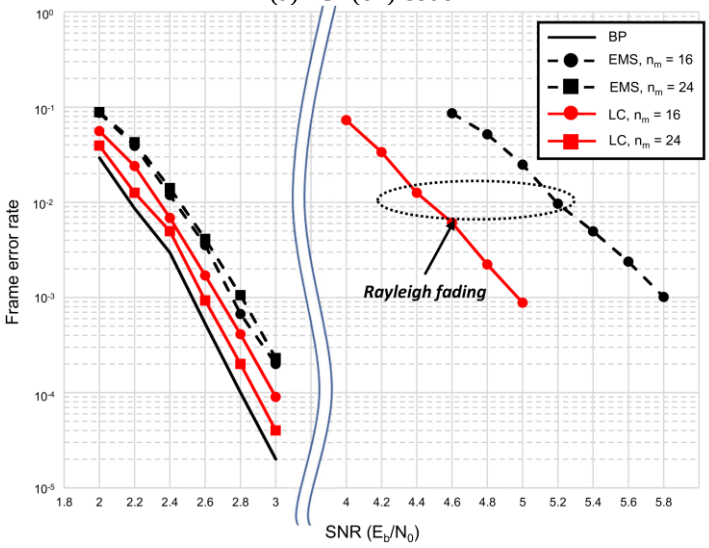
The decoders for (2, 4)-regular NB-LDPC codes over GF(64) with symbol length N = 96 and (2, 4)-regular NB-LDPC codes over GF(256) with symbol length N = 72 are designed using the concepts of optimized rows and short loops, which have been reported previously [18], [19]. The maximum number of decoding iterations is set to 20. The error-correction performance comparisons between both the codes with BPSK modulation are shown in **Fig. 6**. Notably, the lower is FER (BER), the better is the error-correction performance.

Fig. 6 compares the FER performances of (2, 4)-regular NB-LDPC codes over GF(64) and

GF(256) with BPSK modulation. The black dashed lines denote the error-correction performance of the conventional EMS algorithm (n_c = 2), and the solid red lines denote the error-correction performance of the proposed low-cost EMS algorithm (denoted by *LC*). Notably, the proposed algorithm shows better error-correction performance than that of the original one irrespective of the channel condition, provided the truncation size is the same. The error-correction performance improves with an increase in the size of the finite field. From **Fig. 6(b)**, it is evident that the error-correction performance of the proposed algorithm is better than that of the EMS algorithm, even if the truncation size is smaller for the former. With rayleigh fading, the proposed algorithm shows better error-correction performance than the EMS algorithm, and the gap between two algorithms was even greater. Therefore, the experimental results show that the proposed algorithm is more suitable for real world application.



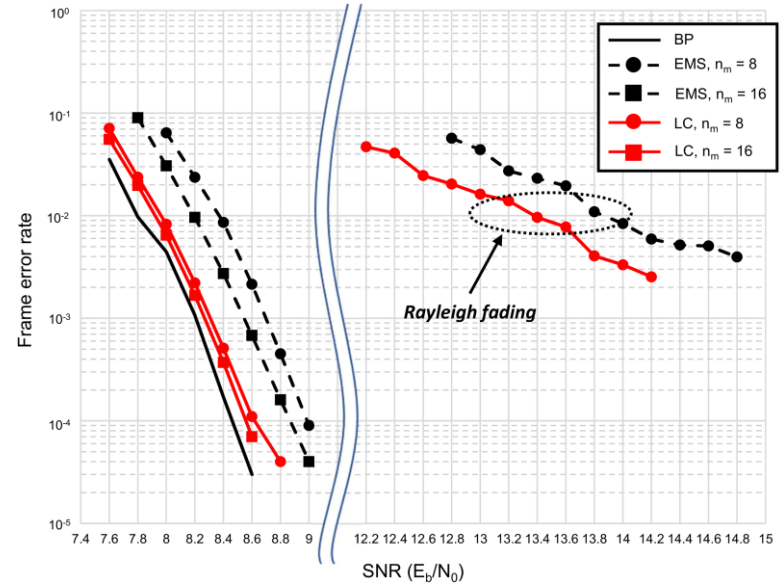
(a) GF(64) code



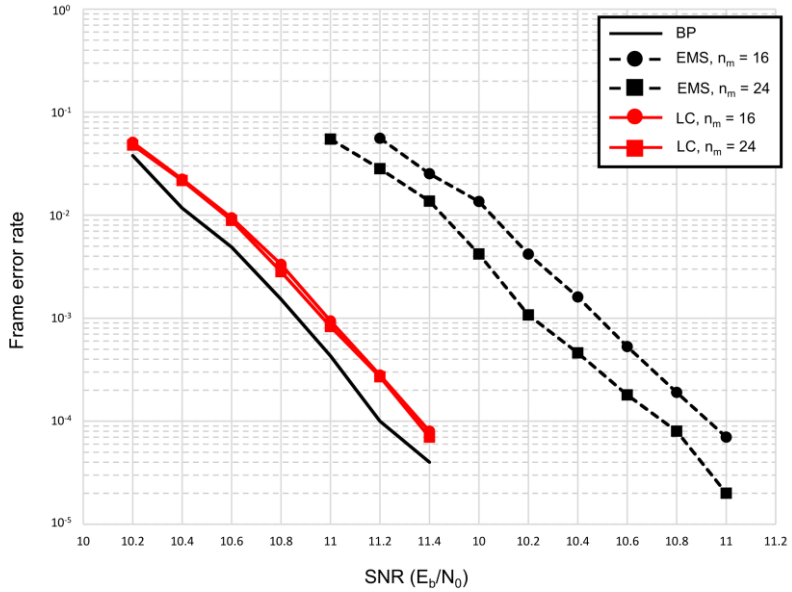
(b) GF(256) code

Fig. 6. FER performance comparisons with BPSK modulation.

Fig. 7 shows the FER performances of (2, 4)-regular NB-LDPC codes over GF(64) and GF(256) with QAM modulation. The improvement in the error-correction performance of the proposed method over conventional EMS algorithm was higher with QAM modulation than that with BPSK modulation. This is because QAM modulation is more susceptible to noise than BPSK modulation, and also the information loss due to SR deteriorates the error-correction performance. As shown in **Fig. 7(b)**, compared with the EMS algorithm, the proposed algorithm achieves an FER smaller by more than 1.0 dB. The proposed algorithm shows better error-correction performances with rayleigh fading channel same as BPSK modulated cases.



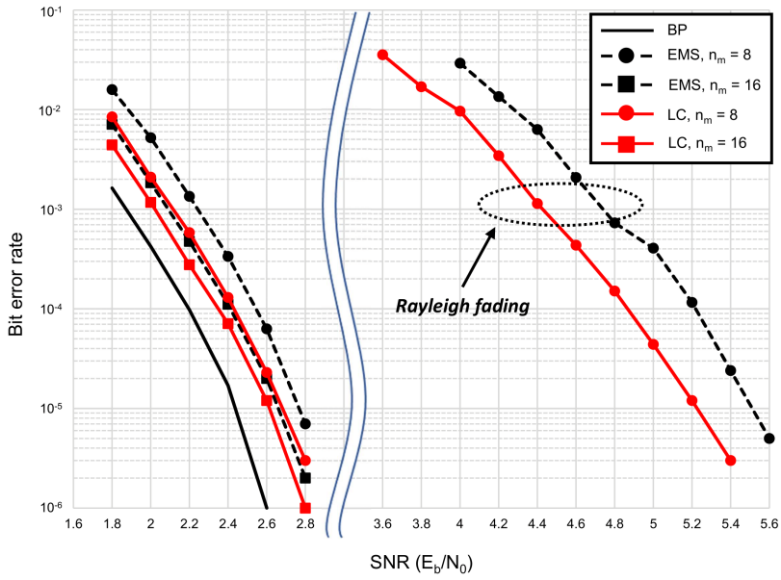
(a) GF(64) code with 64-QAM modulation



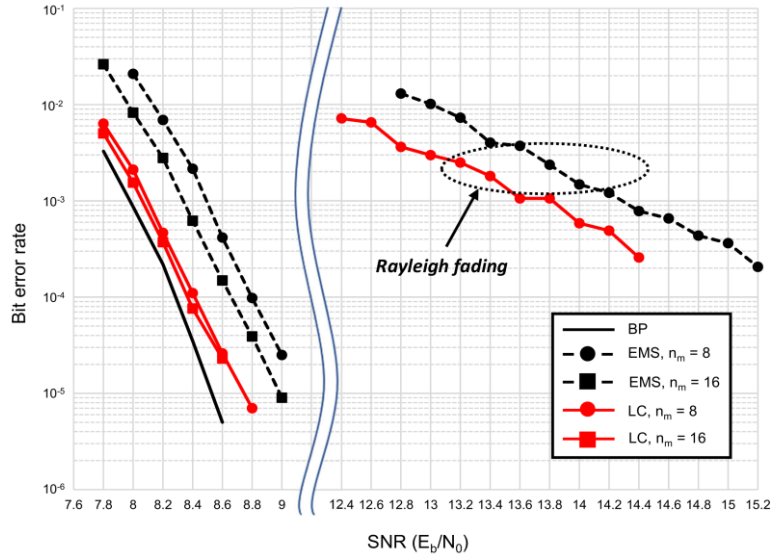
(b) GF(256) code with 256-QAM modulation

Fig. 7. FER performance comparisons

Not only the FER performances, we also compared the BER performances under the simulated codes over GF(64) and it is shown in **Fig. 8**. Same with the results of FER performances, the proposed algorithm with LC CN shows improved BER performances.



(a) BPSK modulation



(b) 64-QAM modulation

Fig. 8. BER performance comparisons of GF(64) code

4.2 Average Iteration Count

We compared the average iteration count of the proposed EMS algorithm with that of the conventional EMS algorithm. The same experimental environment as the one presented in Section 4.1 was used. From **Table 3**, it is evident that the proposed algorithm achieves a

smaller average iteration count by 0.46. From **Table 1** and **Table 2**, it is clear that the execution of a single CN operation of the proposed method with the selection method is substantially faster than the EMS algorithm with SR and sorting. Additionally, because of its small average iteration count, the proposed algorithm is significantly faster than the conventional EMS algorithm.

Table 3. Comparison of the average iteration counts of BPSK modulated (2, 4)-regular NB-LDPC code

SNR	EMS (64, 8)	EMS (64, 16)	LC (64, 8)	LC (64, 16)
1.8	8.39	7.33	7.36	6.69
2	6.76	5.89	5.88	5.48
2.2	5.55	5.01	4.96	4.7
2.4	4.73	4.34	4.29	4.11
2.6	4.12	3.84	3.77	3.65
2.8	3.66	3.44	3.38	3.29

4.3 HW Implementation of Single CN

We compared the amounts of utilized HW resources and execution times of the implemented single CN between the proposed LC CN and conventional CN, respectively. Unlike **Table 1**, MT is included in the implemented CNs, and quick sort is applied for truncation in the conventional CN. We implemented each CN with the FWBW strategy on FPGA by using HLS. Compared with EMS CN, the HW utilization of the proposed LC CN unit is smaller and execution time more than 3 times faster (see **Table 4**).

Table 4. Synthesis results of a single CN unit

CN units		**LC CN**	**EMS CN (Quick sort)**
Execution time (ms)		0.322	1.192
HW resources (Utilized / Total)	LUT	6626 / 230400	7588 / 230400
	FF	6603 / 460800	8042 / 460800
	BRAM	4 / 312	6 / 312

5. Conclusions

We proposed a selection-based LC CN operation for an NB-LDPC decoder. From the viewpoint of HW implementation cost, restricting the number of the most reliable symbols in the CN operation of the EMS algorithm incurs a significant amount of control and sorting overhead, resulting in high latency. The proposed method discarded SR and employed the quick selection algorithm for achieving a low HW complexity and short CN operation time with increased decoding capability. Because the quick selection algorithm selects the k smallest (or largest) elements significantly faster than the sorting methods, the proposed selection-based method operated substantially fast even with a small circuit size. The experimental results verified that the proposed CN operation achieved more than three times faster latency than that of the conventional EMS algorithm. Moreover, because of its simple operation, the FPGA implementation of the proposed method utilized fewer HW resources than those utilized by the conventional EMS method, while achieving a better error-correction performance.

References

[1] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *Proc. of 2003 IEEE Information Theory Workshop*, pp. 70-73, Mar. 2003. Article (CrossRef Link)

[2] T. Lehnigk-Emden and N. Wehn, "Complexity evaluation of non-binary Galois field LDPC code decoders," in *Proc. of 2010 6th IEEE International Symposium on Turbo Codes and Iterative Information Processing*, pp. 53-57, Sep. 2010. Article (CrossRef Link)

[3] Y. Lu, N. Qiu, Z. Chen, and S. Goto, "An efficient majority-logic based message-passing algorithm for non-binary LDPC decoding," in *Proc. of 2012 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 479-482, Dec. 2012. Article (CrossRef Link)

[4] F. Garcia-Herrero, D. Declercq, and J. Valls, "Non-Binary LDPC decoder based on symbol flipping with multiple votes," *IEEE Communications Letters*, vol. 18, no. 5, pp. 749-752, May 2014. Article (CrossRef Link)

[5] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633-643, Apr. 2007. Article (CrossRef Link)

[6] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc. of 2004 IEEE International Conference on Communications*, vol. 2, pp. 772-776, June 2004. Article (CrossRef Link)

[7] E. Li, K. Gunnam, and D. Declercq, "Trellis based extended min-sum for decoding nonbinary LDPC codes," in *Proc.* of the *8th International Symposium on Wireless Communication Systems*, pp. 46-50, Nov. 2011. Article (CrossRef Link)

[8] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary LDPC codes and its hardware structure," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2600-2611, July 2013. Article (CrossRef Link)

[9] J. O. Lacruz, F. García-Herrero, J. Valls, and D. Declercq, "One minimum only trellis decoder for non-binary low-density parity-check codes," *IEEE transactions on circuits and systems I: regular papers*, vol. 62, no. 1, pp. 177-184, 2014. Ariticle (CrossRef Link)

[10] J. O. Lacruz, F. García-Herrero, M. J. Canet, and J. Valls, "Reduced-complexity nonbinary LDPC decoder for high-order Galois fields based on trellis min–max algorithm," *IEEE Transactions on very large scale integration (VLSI) Systems*, vol. 24, no. 8, pp. 2643-2653, 2016. Ariticle (CrossRef Link)

[11] H. P. Thi and H. Lee, "Two-extra-column trellis min–max decoder architecture for nonbinary LDPC codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1787-1791, 2017. Ariticle (CrossRef Link)

[12] T. X. Pham, T. N. Tan, and H. Lee, "Minimal-Set Trellis Min-Max Decoder Architecture for Nonbinary LDPC Codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, 216-220, 2020. Ariticle (CrossRef Link)

[13] Y. Hwang, K. Yang, and K. Cheun, "Low-latency low-complexity heap-based extended min-sum algorithms for non-binary low-density parity-check codes," *IET Communications*, vol. 9, no. 9, pp. 1191-1198, June 2015. Article (CrossRef Link)

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

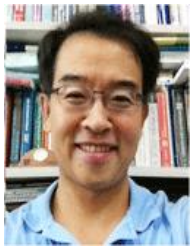
[15] E. Boutillon and L. Conde-Canencia, "Bubble check: A simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *Electronics Letters*, vol. 46, no. 9, pp. 633-634, Apr. 2010. Article (CrossRef Link)

[16] P. Schläfer, N. Wehn, M. Alles, T. Lehnigk-Emden, and E. Boutillon, "Syndrome based check node processing of high order NB-LDPC decoders," in *Proc. of the 22nd International Conference on Telecommunications (ICT)*, pp. 156-162, Apr. 2015. Article (CrossRef Link)

[17] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498-519, Feb. 2001. Article (CrossRef Link)

[18] G. Haboub, "Entwicklungen verteilter Bildercodierungsmethoden basierend auf LDPC," Doctoral Dissertation, Berlin Institute of Technology, Berlin, Germany, 2011.
[19] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, d_c)-LDPC codes over GF(q) using their binary images," *IEEE Transactions on Communications*, vol. 56, no. 10, pp. 1626-1635, Oct. 2008. Article (CrossRef Link)

Kyeongbin Park received his B.S. in Electronics & Communication Engineering from Hanyang University, Seoul, Korea in 2014, and He is currently working toward Ph.D. degree in Electronics and Computer Engineering from Hanyang University, Seoul, Korea. His interest research includes algorithms and hardware implementation of error correction codes.



Ki-Seok Chung received his B.S. in Computer Engineering from Seoul National University, Seoul, Korea in 1989, and Ph.D. in Computer Science from 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. Since 2004, he has been a professor at Hanyang University, Seoul, Korea. His research interests include low power embedded system design, multi-core architecture, image processing, reconfigurable processor and DSP design, SoC-platform based verification and system software for MPSoC.