IEIE Transactions on Smart Processing and Computing, vol. 8, no. 3, June 2019
https://doi.org/10.5573/IEIESPC.2019.8.3.236

IEIE Transactions on Smart Processing and Computing

Area-efficient Early Termination in Belief Propagation Decoders of Polar Codes

Soyeon Choi and Hoyoung Yoo

Department of Electronics Engineering, Chungnam National University, Daejeon 34134, Korea hyyoo.cnu@gmail.com

* Corresponding Author: Hoyoung Yoo

Received March 27, 2019; Accepted May 7, 2019; Published June 30, 2019

** Regular Paper*

** Extended from a Conference: Preliminary results of this paper were presented at the ICEIC 2019. This paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

***Abstract*:** Early termination has been widely adopted to decrease decoding latency and power consumption in a belief propagation (BP) decoder of polar codes. Although previous early termination schemes succeeded in improving the decoding iteration time, they suffer from the high hardware complexity required to detect the early termination condition. In this paper, area-efficient early termination methods are proposed by simplifying the previous termination conditions. We optimize two previous methods and propose simplified early termination algorithms, which are the simplified-G-matrix (SGM) and simplified minimum log-likelihood ratio (SML). While the previous early termination algorithms take both information and frozen bits into account, the proposed early termination algorithms use the condition associated with only the information. For (1024, 512) polar codes, the proposed SGM and SML achieve 25% and 50% area reductions, respectively, compared to their counterparts. The proposed early termination algorithms can be applicable for any BP decoders without sacrificing decoding performance.

***Keywords*:** Belief-propagation decoder, Early termination, Area-efficient, Low-latency, Polar codes

1. Introduction

Polar codes are the first error-correcting codes proven to achieve channel capacity with an efficient encoding and decoding algorithm [1]. Due to the superior error-correcting performance, many applications, including next-generation wireless communications and storage systems [5], are continuously trying to adopt polar codes. The two main decoding algorithms, successive cancellation (SC) [1-4] and belief propagation (BP) [6-9], are usually employed to bring polar codes into a practical implementation.

The SC decoding process [1-4] is based on the divide-and-conquer approach [1], which recursively divides a polar code of length N into two smaller polar codes of $N/2$, and recursively calculates estimated messages. A serial decoding process is inevitable in SC decoders, since decoding the later, smaller polar code is dependent on the former smaller polar code. While SC decoders [1-4] use a small amount of hardware resources, they suffer from long latency due to their serial nature. Recently, advanced SC algorithms, including list decoding [10, 11] and stack decoding [11, 12], have been studied so as to further improve the error correction performance using more hardware resources.

While a serial nature is inherent in SC decoding [1-4], a parallel nature is inherent in BP decoding [6-9]. During the decoding process in BP decoders, all messages are estimated simultaneously. Compared to SC decoders [1-4], BP decoders [6-9] have generally short decoding latency while demanding higher hardware complexity. In BP decoders, early termination techniques [14-19] are widely applied to decrease decoding latency further. Before the predetermined maximum number of iterations, the decoding process is terminated when a specific condition is satisfied. As the signal-to-noise ratio (SNR) increases, the effect on iteration saving becomes more remarkable.

Although previous early termination schemes [14-17] succeeded in improving the encoding iteration time, the additional circuits needed to detect the early termination condition are not negligible, especially for long polar codes.

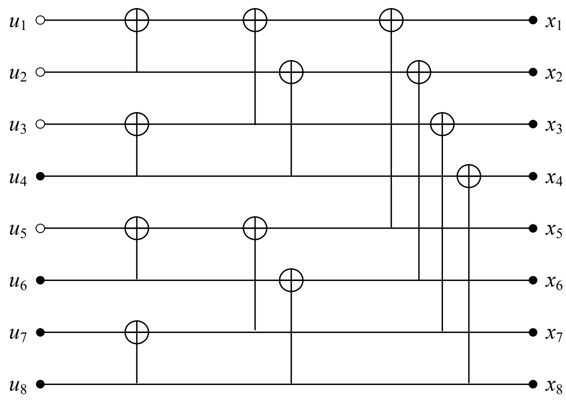

Fig. 1. The polar network for (8, 4) polar codes.

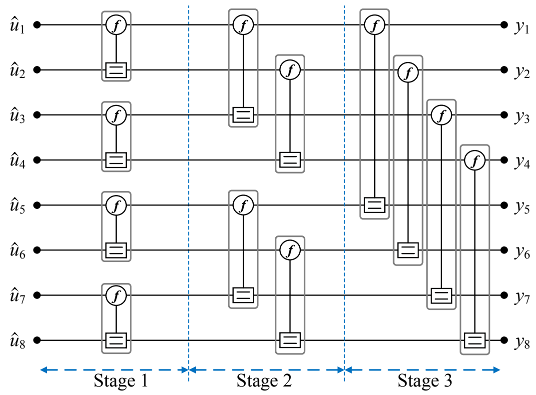

Fig. 2. The factor graph of the (8, 4) polar codes.

In this paper, we modify the previous early termination conditions and propose simplified early termination schemes by eliminating unnecessary detection circuitry. According to the experimental results, the proposed early termination schemes save up to approximately half of the hardware resources while maintaining error correction performance and iteration savings. The rest of this paper is organized as follows. Section 2 provides the background to this work, and Section 3 describes the proposed early termination algorithms. Experimental results are discussed in Section 4, and concluding remarks are in Section 5.

2. Polar Codes

Polar codes are the first error-correcting code that can probably achieve channel capacity with efficient encoding and decoding algorithms. Polar codes are based on the phenomenon of channel polarization. According to the channel polarization [1], a part of the bit-channels become essentially error-free, and the others become completely noisy as the code length goes to infinity. The main concept of polar codes is to transmit informative messages through error-free bit-channels and to transmit fixed values through completely noisy bit-channels, respectively.

Let us consider an (N, K) polar code, where N and K denote code length and message length, respectively. Among the N bit-channels, the K most reliable bit-channels are used for the transmission of information bits, and $(N-K)$ unreliable bit-channels are used for the transmission of fixed values called frozen bits. The index set of information bits is denoted by A, and the index set of frozen bits is denoted by A^c. The input message vector, $\boldsymbol{u}$, with a length of N is composed of K information bits and $(N-K)$ frozen bits based on information set A and frozen set A^c. Since the polar codes belong to a type of linear block codes, codeword $\boldsymbol{x}$ is generated by matrix multiplication of input message vector $\boldsymbol{u}$ and generator matrix $\mathbf{G}$ as $\boldsymbol{x} = \boldsymbol{u}^{\mathrm{T}}\mathbf{G}$. Furthermore, generator matrix $\mathbf{G}$ is built by the n-th Kronecker power of $\mathbf{F}$, where $n = \log_2 N$, and $\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. For instance, Fig. 1 depicts an encoding structure for (8, 4) polar codes where A={4, 6, 7, 8} and A^c={1, 2, 3, 5}. The encoder is implemented with a total of 12 XOR gates, and operates during three stages. Note that the white and black circles on the left side indicate frozen and information bit-positions, respectively. A four-bit message assigned to the information bit-positions is encoded into an eight-bit codeword through the encoding network, as shown in Fig. 1.

2.1 Belief-propagation Decoding

In BP decoding [6-9], messages are decoded based on a factor graph, which efficiently computes the marginal probability at an individual node. Given the channel output, y, the BP decoder propagates probabilities alternating left to right and right to left. Fig. 2 depicts the factor graph for (8, 4) polar codes, which consists of 12 processing elements (PEs) and processes in three stages. It is important to note that the factor graph for decoding polar codes has a network similar to the encoding one, except for the structure of the PEs. To clarify input and output connection of PEs, the index for a PE is more precisely described in Fig. 3, where i, s, and t indicate node index, stage index, and time index, respectively. Distance between each input and output is described as 2^{s-1}, which leads to one, two, and four distances between each of two inputs and outputs at the first, second, and third stages, as shown in Fig. 2. During propagation on the factor graph, log-likelihood ratios (LLRs) are usually adopted, rather than probabilities, since the use of LLRs provides much less hardware complexity in practice. For alternately propagating LLRs on the factor graph, two LLR values propagated from right to left, $L^t_{i,s}$ and $L^t_{i+2^{s-1},s}$, are computed as follows:

$$L^t_{i,s} = f\left(L^{t-1}_{i,s+1}, L^{t-1}_{i+2^{s-1},s+1} + R^t_{i+2^{s-1},s}\right), \tag{1}$$

$$L^t_{i+2^{s-1},s} = f\left(L^{t-1}_{i,s+1}, R^t_{i+2^{s-1},s}\right) + L^{t-1}_{i+2^{s-1},s+1}, \tag{2}$$

and two LLR values propagated from left to right, $R^t_{i,s+1}$ and $R^t_{i+2^{s-1},s+1}$, are computed as

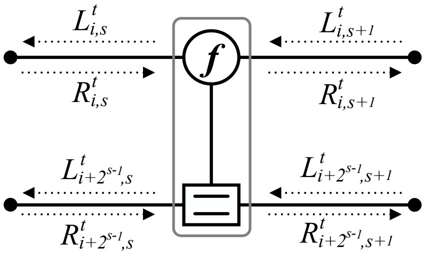

Fig. 3. The processing element for BP decodes.

$$R^t_{i,s+1} = f\left(R^t_{i,s},\, L^{t-1}_{i+2^{s-1},s+1} + R^t_{i+2^{s-1},s}\right), \tag{3}$$

$$R^t_{i+2^{s-1},s+1} = f\left(R^t_{i,s},\, L^{t-1}_{i,s+1}\right) + R^t_{i+2^{s-1},s}. \tag{4}$$

There are two types of LLR values, $L^t_{i,s}$ and $R^t_{i,s}$, which represent right-to-left and left-to-right LLR values at the i-index and s-stage, respectively, in the t-decoding iteration.

The original BP decoding computes the function $f(a, b)$ as $2\tanh^{-1}\{\tanh(a/2)\times\tanh(b/2)\}$. However, the hyperbolic function leads to serious burdens in both computation and implementation, and thus, Gallager proposed min-sum approximation [9], which approximates $f(a, b)$ as $\text{sign}(a)\times\text{sign}(b)\times\min(|a|,|b|)$. Min-sum approximation [9] can save remarkable hardware resources with negligible degradation of the correcting performance. Furthermore, the scaled min-sum approximation in [20] compensates for Gallager's min-sum approximation [9] by multiplying by scaling factor α to further decrease degradation of the correcting performance.

To sum up, the conventional BP decoding algorithm is described in Fig. 4. First, $L^t_{i,s}$ and $R^t_{i,s}$ in each node are initialized. Given channel output y, $L^t_{i,n+1}$ at the n+1 stage is initialized as $\text{LLR}(y_i) = \log(\Pr(y_i=0)/\Pr(y_i=1))$. In stage 1, $R^t_{i,1}$ for $i \in A^c$ is initialized to infinity to provide the information from the frozen set. The others are initialized to zero, since they have no prior information. After the initialization, each node on the factor graph is alternately updated based on left-to-right LLRs and right-to-left LLRs from Eqs. (1)-(4). The node update continues until the predetermined number of decoding iterations, T. Lastly, the messages are estimated when time index t reaches maximum decoding iteration T by making a hard decision in stage 1, as follows:

$$\hat{u}_i = \begin{cases} 0, \left(L^T_{i,1} + R^T_{i,1}\right) \geq 0 \\ 1, \left(L^T_{i,1} + R^T_{i,1}\right) < 0 \end{cases}. \tag{5}$$

Although the conventional BP decoding [6-9] provides good error-correcting performance, a large number of computations is inevitable, since the hard decision is performed in predetermined maximum iteration T. The huge computational complexity also results in long decoding latency and large power consumption. To resolve the problem, early termination schemes were employed [14-17] by terminating the decoding iteration before

```
Initialization:
   If (s == n+1)
      L^t_{i,n+1} = LLR (y_i)
   Else if (s == 1) & (i ∈ A^c )
      R^t_{i,1} = ∞
   Else
      L^t_{i,s} = R^t_{i,s} = 0
   End if
   t = 0
Decoding Iteration:
   While (t ≤ T)
      Update L^t_{i,s} and R^t_{i,s} based on (1)-(4)
      If (t==T)
         û_i = sign (L^t_{i,1} + R^t_{i,1})
      End if
      t = t + 1
   End while
```

Fig. 4. The conventional BP decoding algorithm.

maximum iteration T. By introducing negligible error-correction degradation, the early termination schemes [14-17] succeeded in the reduction of average decoding iterations.

2.2 Previous Early Terminations

Among the previous early termination algorithms for the polar BP decoder, the G-matrix-based (GM) early termination algorithm [14] provides a simple termination condition by re-encoding the results from an intermediate estimation. Based on the fact that $\boldsymbol{x} = \boldsymbol{u}^{\mathrm{T}}\mathbf{G}$ for the encoding process, as depicted in Fig. 1, $\hat{\boldsymbol{u}}$ is likely valid if $\hat{\boldsymbol{u}}^{\mathrm{T}}\mathbf{G}$ is equal to $\hat{\boldsymbol{x}}$. In the GM early termination algorithm [14], $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{x}}$ are estimated at the first and last stages, as follows:

$$\hat{u}_i = \begin{cases} 0, & (L^t_{i,1} + R^t_{i,1}) \geq 0 \\ 1, & (L^t_{i,1} + R^t_{i,1}) < 0 \end{cases}, \tag{6}$$

$$\hat{x}_i = \begin{cases} 0, & (L^t_{i,n+1} + R^t_{i,n+1}) \geq 0 \\ 1, & (L^t_{i,n+1} + R^t_{i,n+1}) < 0 \end{cases}. \tag{7}$$

When the result of re-encoding for $\hat{\boldsymbol{u}}$ is equal to $\hat{\boldsymbol{x}}$, GM [14] early termination finishes the decoding iterations under the assumption that $\hat{\boldsymbol{u}}$ is a valid estimation. As a result, the GM [14] can save a remarkable number of decoding iterations, and the average number of decoding iterations decreases as the channel condition improves. However, a GM [14] necessitates circuitry for the termination check, including adders, comparators, and the encoding structure. More precisely, $2N$ adders are necessary to generate the estimate of $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{x}}$, and $(N/2)\cdot\log_2 N$ XORs are used for re-encoding, and N comparators are necessary to check that $\hat{\boldsymbol{x}} = \hat{\boldsymbol{u}}^{\mathrm{T}}\mathbf{G}$. For instance, a GM [14] early termination for (1024, 512) polar codes demands a total of 2048 adders, 5120 XOR gates, and 1024 comparators. The additional circuits for GM [14] early termination become a serious burden, especially for long polar codes.

```
Initialization:
   If (s == n+1)
      L^t_{i,n+1} = LLR (y_i)
   Else if (s == 1) & (i ∈ A^c )
      R^t_{i,1} = ∞
   Else
      L^t_{i,s} = R^t_{i,s} = 0
   End if
   t = 0
Decoding Iteration:
   While (t ≤ T)
      Update L^t_{i,s} and R^t_{i,s} based on (1)-(4)
      Update û_i and x̂_i based on (6)-(7)
      If (ûG̃ == x̂)
         iteration termination
      End if
      t = t + 1
   End while
```

Fig. 5. The proposed SGM early termination algorithm.

To assuage the huge hardware complexity, the minimum log-likelihood ratio (min-LLR)-based (ML-based) early termination algorithm was presented in [14]. Whereas the GM [14] algorithm employs only hard information for the termination condition, the ML-based algorithm employs soft information [14]. Since the LLR value represents the ratio of $\hat{u}_i$ being 0 or 1 at the i-th bit position, $\hat{u}_i$ is more likely to be valid if the magnitude of LLR for $\hat{u}_i$ is larger. The ML [14] computes the magnitude of LLR values at the first stage and determines if the magnitudes are large enough. More precisely, ML [14] calculates the magnitudes of $|L^t_{i,1}+R^t_{i,1}|$ for $1 \le i \le N$ and finds the minimum among them. When the minimum of $|L^t_{i,1}+R^t_{i,1}|$ is larger than a threshold value, β, ML considers all the estimated messages as valid and terminates the decoding iteration [14]. Otherwise, the decoding iteration continues until the predetermined maximum iteration, T. It is clear that the selection of an appropriate threshold value for β is crucial to achieving a decoding iteration without error-correction degradation. When β is large, early termination would be excessively activated, even when the estimation is not reliable. In addition, when β is small, the iteration savings obtained from ML [14] decreases, since more iterations are needed to get a higher magnitude. To implement the ML [14] early termination algorithm, N adders and N comparators are necessary in order to compute $|L^t_{i,1}+R^t_{i,1}|$ and to find the minimum among N values. For instance, ML [14] early termination for (1024, 512) polar codes demands a total of 1024 adders and 1024 comparators. Although the decoding iteration savings obtained from ML [14] vary, depending on threshold value β, ML [14] definitely saves decoding iterations, on average, with relatively few hardware resources, compared to GM [14] early termination.

3. Proposed Early Termination Algorithms

Although the previous GM [14] and ML [14] early termination schemes succeeded in saving decoding iteration time, they demanded more hardware resources to detect the early termination condition. In this section, we first optimize two previous schemes and propose simplified early termination algorithms and their structures: the simplified-G-matrix (SGM) and the simplified-minLLR (SML). While the previous early termination algorithms take both information and frozen bits into account, the proposed early termination algorithms use the condition associated only with the information.

3.1 Simplified G-matrix-based Algorithm

The GM [14] early termination algorithm checks for the termination condition by re-encoding estimate $\hat{\boldsymbol{u}}$. In the GM [14], $\hat{u}_i$ for $1 \le i \le N$ is calculated with Eq. (6), which requires N adders. However, it is clear that the calculations associated with frozen set A^c are unnecessary.

According to (6), $\hat{u}_i$ for $i \in A^c$ is always zero, since $R^t_{i,1}$ for a frozen bit is initialized to infinity. Thus, we can completely remove the adders associated with frozen set A^c. Furthermore, an XOR network where leaves are all frozen bits can also be eliminated since the result of the XOR network is guaranteed to be zero. As a result, the SGM decreases hardware complexity by eliminating the condition check associated with frozen bits. The proposed SGM optimizes the previous GM [14] early termination by using simplified generator matrix $\tilde{\mathbf{G}}$ that extracts the computation associated with the frozen bits in generator matrix **G**. Fig. 5 summarizes the proposed SGM early termination algorithm. Note that the SGM employs simplified generator matrix $\tilde{\mathbf{G}}$ instead of the conventional generator matrix **G**, leading to less detection circuitry in practice.

The hardware structures for the GM [14] and the proposed SGM are compared in Fig. 6. For (8, 4) polar codes, the previous GM [14] requires 16 adders, 12 XORs, and eight comparators, and the proposed SGM demands 12 adders, eight XORs, and eight comparators. Whereas the previous GM [14] demands the detection circuits, regardless of the code rate, $R(R = K/N)$, the complexity of the detection circuits for the proposed SGM varies depending on code rate R. For the (N, K) polar codes with a code rate of R, the proposed SGM demands $(1+R)N$ adders, $\gamma(N/2)\cdot\log_2 N$ XOR gates, and N comparators, where γ is a reduction ratio resulting from XOR network simplification. Since code rate R and reduction ratio γ are always less than one, it is guaranteed that the proposed SGM always has less hardware complexity, compared to the previous GM [14]. As the code rate becomes lower, more hardware savings are expected.

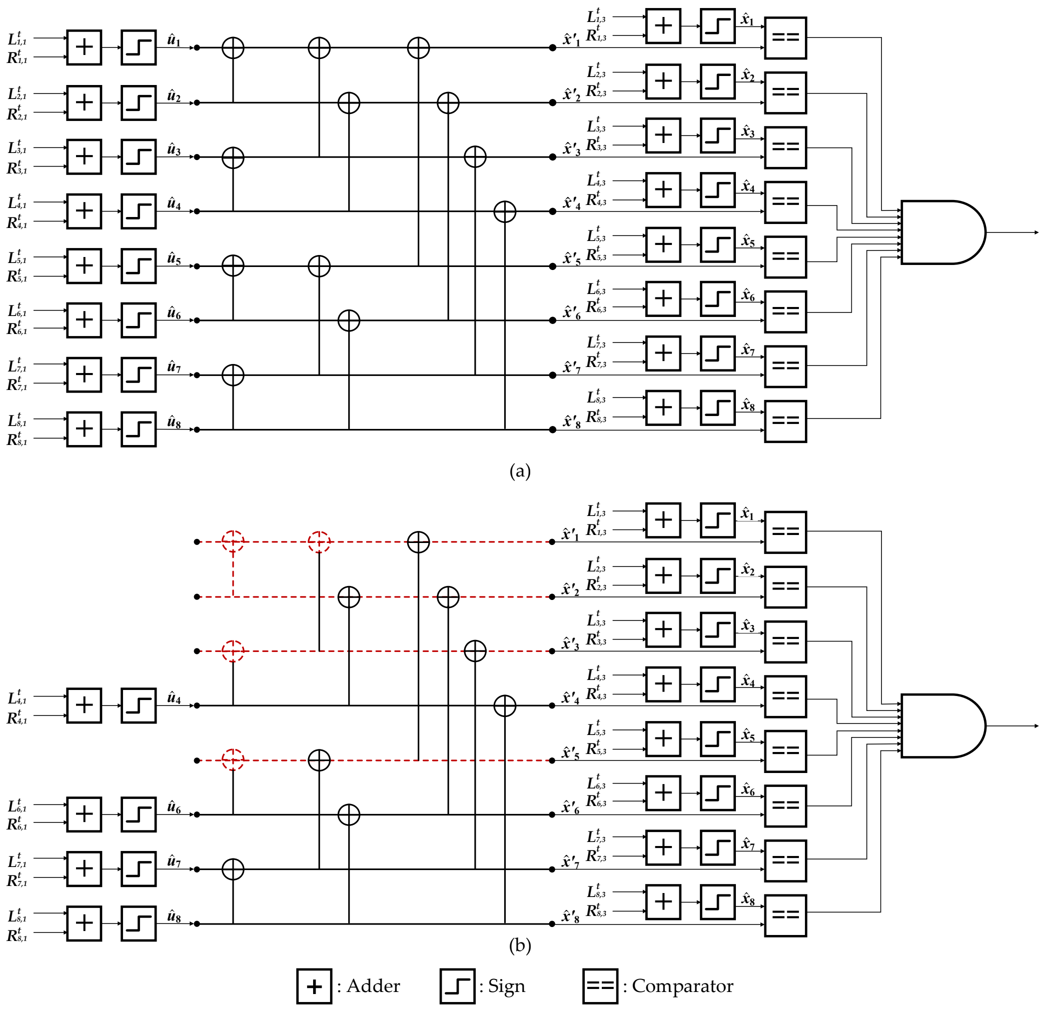

Fig. 6. Structure of (a) the GM and (b) the SGM for (8, 4) polar codes.

3.2 Simplified Min-LLR-based Algorithm

Although ML [14] early termination compares the magnitude of all bit-channels with threshold value β, the minimum magnitude is always in frozen set A. Because $R^t_{i,s}$ for a frozen bit is initialized to infinity, the magnitude $|L^t_{i,1}+R^t_{i,1}|$ of a frozen bit is also infinity and is not the minimum at all. Thus, the proposed simplified ML algorithm checks the magnitude of LLRs related only to information set A. Fig. 7 summarizes the proposed SML early termination algorithm. Unlike the previous ML [14] algorithm, the proposed SML finds the minimum magnitude among only information set A. The proposed SML can reduce hardware complexity by eliminating unnecessary computations related to frozen set A^c. The hardware structures for the ML [14] and the proposed SML are compared in Fig. 8. For (8, 4) polar codes, eight adders and eight absolute operators are used to compute the magnitudes, and eight comparators are used to generate a termination control signal. On the other hand, the proposed SGM demands four adders, four absolute operators, and four comparators by eliminating the circuitry associated with the frozen bits. Whereas the detection circuits for the previous ML [14] do not depend on the code rate, those for the proposed SML depend on code rate R. For (N, K) polar codes with a code rate of R, the proposed SML demands RN adders, RN absolute

Initialization:
  If $(s == n+1)$
    $L^t_{i,n+1}$ = LLR (y_i)
  Else if $(s == 1)\ \&\ (i \in A^c)$
    $R^t_{i,1} = \infty$
  Else
    $L^t_{i,s} = R^t_{i,s} = 0$
  End if
  $t = 0$
Decoding Iteration:
  While $(t \le T)$
    Update $L^t_{i,s}$ and $R^t_{i,s}$ based on (1)-(4)
    Find min among $|L^t_{i,1}|$ for $i \in A$
    If (min $\ge \beta$)
      iteration termination
    End if
    $t = t + 1$
  End while

Fig. 7. The proposed SML early termination algorithm.

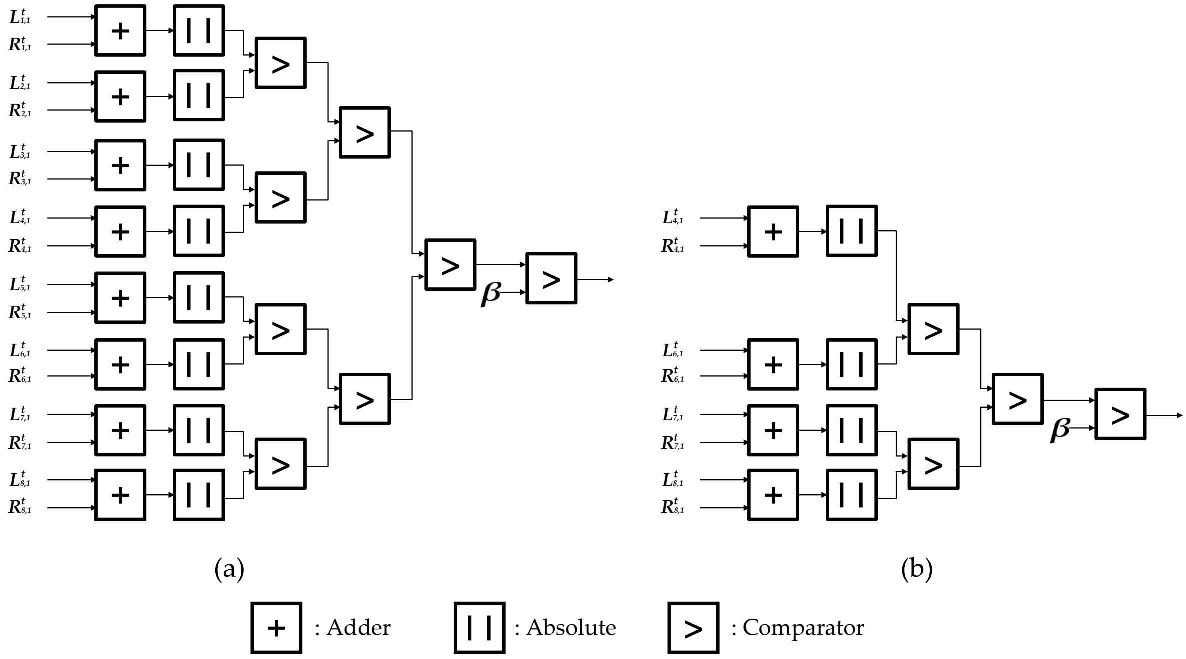

Fig. 8. Structure of (a) ML and (b) SML for the (8, 4) polar codes.

operators, and RN comparators. Similar to the proposed SGM, the hardware savings are more significant as the code rate becomes lower.

4. Experimental Results

First, we simulated BP decoders equipped with various early termination techniques for difference SNR conditions to verify the fact that the proposed SGM and SML do not introduce any degradation in error-correcting performance. All BP decoders take maximum decoding iteration T at 40, and (1024, 512) polar codes with a code rate of 1/2 are employed. In addition, threshold β for the previous ML [14] and the proposed SML is set at 3.5 to eliminate error-correcting obtained from the conventional BP decoder without any early termination. According to Fig. 9, all BP decoders adopted with the previous and proposed early terminations have no error-correcting degradation, compared to the conventional BP decoder. Moreover, Fig. 10 shows the average number of iterations for various SNR conditions. Unlike early termination, all BP decoders equipped with early termination decrease the average number of iterations as the SNR condition improves. Since both the GM and the SGM demand more hardware resources, they save more iterations, compared to the ML and the SML early terminations.

More importantly, for various early termination algorithms, we summarize the use of hardware resources in Table 1. Note that the use of limiters in Fig. 7 is not counted, since they can be implemented as hard-wiring without any hardware resources. Since code rate R and reduction ratio γ are less than one, the use of hardware resources for the proposed algorithms is always less than for each of the previous ones. For instance, when (1024, 512) polar codes with a code rate of R=1/2 are considered,

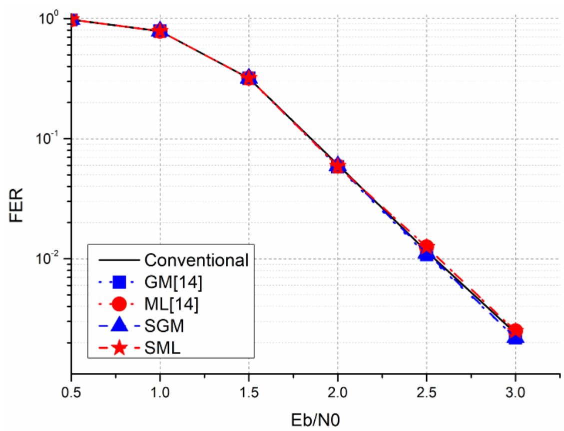

Fig. 9. FER performance for (1024, 512) polar codes when BP adopts GM, ML, SGM, and SML early termination algorithms.

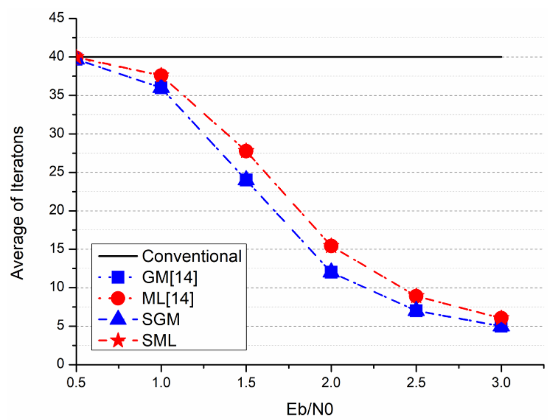

Fig. 10. Average number of iterations for (1024, 512) polar codes when BP adopts GM, ML, SGM, and SML early termination algorithms.

Table 1. Experiment Parameters.

Computational Units	GM[14]	SGM	ML[14]	SML
Adders	$2N$	$(1+R)N$	N	RN
Comparators	N	$RN+2N$	N	RN
Absolute operator	–	–	N	RN
XOR gates	$(N/2)\log_2 N$	$\gamma(N/2)\log_2 N$	–	–
AND gates	$N-1$	$N-1$	–	–

the proposed SGM and SML can achieve virtually 25% and 50% area reductions, respectively. According to Table 1, the proposed early termination algorithms can achieve more improvements for a lower code rate. Since the proposed SGM and SML simplify the conventional BP decoders, taking no detection circuits of the previous GM and ML, the proposed algorithms always have less hardware compared to their counterparts.

5. Conclusion

In this paper, we proposed two early termination algorithms to reduce unnecessary computations and the corresponding hardware resources. Whereas the previous early termination algorithms [14] for polar BP decoders take both information and frozen bits into account, the proposed SGM and SML consider only the information bits. According to the experimental results, the proposed schemes significantly reduce the detection circuitries for the termination condition, compared to their counterparts, without any degradation in correcting performance and decoding iterations. Since the proposed SGM and SML completely eliminate the circuits related to frozen bits, area reduction becomes more significant as the code rate decreases.

Acknowledgement

This work was supported by the research fund of Chungnam National University in 2017.

References

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes," in *Proc. 2008 IEEE International Symposium on Information Theory*, Toronto, ON, 2008, pp. 1173-1177. Article (CrossRef Link)

[2] K. Niu, K. Chen, J. Lin and Q. T. Zhang, "Polar codes: Primary concepts and practical decoding algorithms," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 192-203, July 2014. Article (CrossRef Link)

[3] N. Hussami, S. B. Korada and R. Urbanke, "Performance of polar codes for channel and source coding," in *Proc. 2009 IEEE International Symposium on Information Theory*, Seoul, 2009, pp. 1488-1492. Article (CrossRef Link)

[4] I. Tal and A. Vardy, "How to Construct Polar Codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562-6582, Oct. 2013. Article (CrossRef Link)

[5] Guangyi Liu, Dajie Jiang, "5G: Vision and Requirements for Mobile Communication System towards Year 2020," *Chinese Journal of Engineering*, vol. 2016, Article ID 5974586, 8 pages, 2016. Article (CrossRef Link)

[6] E. Arikan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Communications Letters*, vol. 12, no. 6, pp. 447-449, June 2008. Article (CrossRef Link)

[7] Arıkan, E, "Polar codes: A pipelined implementation," *in Proc. 4th Int. Symp. Broadcast. Commun.* (ISBC), Jul. 2010, pp. 11-14. Article (CrossRef Link)

[8] Y. Zhang, A. Liu, X. Pan, Z. Ye and C. Gong, "A Modified Belief Propagation Polar Decoder," *IEEE Communications Letters*, vol. 18, no. 7, pp. 1091-1094, July 2014. Article (CrossRef Link)

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

[10] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. 2011 IEEE International Symposium on Information Theory Proceedings*, St. Petersburg, 2011, pp. 1-5. Article (CrossRef Link)

[11] B. Li, H. Shen and D. Tse, "An Adaptive Successive Cancellation List Decoder for Polar Codes with Cyclic Redundancy Check," *IEEE Communications Letters*, vol. 16, no. 12, pp. 2044-2047, December 2012. Article (CrossRef Link)

[12] K. Niu and K. Chen, "Stack decoding of polar codes," *Electronics Letters*, vol. 48, no. 12, pp. 695 -697, 7 June 2012. Article (CrossRef Link)

[13] K. Chen, K. Niu and J. Lin, "Improved Successive Cancellation Decoding of Polar Codes," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3100-3107, August 2013. Article (CrossRef Link)

[14] B. Yuan and K. K. Parhi, "Early Stopping Criteria for Energy-Efficient Low-Latency Belief-Propagation Polar Code Decoders," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6496-6506, Dec.15, 2014. Article (CrossRef Link)

[15] C. Simsek and K. Turk, "Simplified Early Stopping Criterion for Belief-Propagation Polar Code Decoders," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1515-1518, Aug. 2016. Article (CrossRef Link)

[16] Y. Zhang, Q. Zhang, X. Pan, Z. Ye and C. Gong, "A simplified belief propagation decoder for polar codes," in *Proc. 2014 IEEE International Wireless Symposium*, X'ian, 2014, pp. 1-4. Article (CrossRef Link)

[17] Q. Zhang, A. Liu and X. Tong, "Early stopping criterion for belief propagation polar decoder based on frozen bits," *Electronics Letters*, vol. 53, no. 24, pp. 1576-1578, 23 11 2017. Article (CrossRef Link)

[18] S. M. Abbas, Y. Fan, J. Chen and C. Tsui, "Low complexity belief propagation polar code decoder," in *Proc. 2015 IEEE Workshop on Signal Processing Systems* (SiPS), Hangzhou, 2015, pp. 1-6. Article (CrossRef Link)

[19] J. Lin, C. Xiong and Z. Yan, "Reduced complexity belief propagation decoders for polar codes," in *Proc. 2015 IEEE Workshop on Signal Processing Systems* (SiPS), Hangzhou, 2015, pp. 1-6. Article (CrossRef Link)

[20] B. Yuan and K. K. Parhi, "Architecture optimizations for BP polar decoders," in *Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, 2013, pp. 2654-2658. Article (CrossRef Link)

Hoyoung Yoo received a BSc in Electrical and Electronic Engineering from Yonsei University, Seoul, Korea, in 2010, and an MSc and a PhD in Electronics Engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2012 and 2016, respectively. Since September 2016, he has been an assistant professor in the department of Electronics Engineering at Chungnam National University. Special areas of interest include VLSI design for error correction codes and 5G communications systems.

Soyeon Choi received her BSc in Electronics Engineering from Chungnam National University (CNU), Daejeon, Korea, in 2018. Since 2018, she has been in the MSc program for Electronics Engineering at CNU, Daejeon, Korea. Her current research interests include the algorithms and VLSI for error-correcting code.