Received April 21, 2020, accepted May 14, 2020, date of publication May 20, 2020, date of current version June 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2996016

Large-Small Sorting for Successive Cancellation List Decoding of Polar Codes

KYUNGPIL LEE, (Student Member, IEEE), AND IN-CHEOL PARK, (Senior Member, IEEE)

School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 305-701, South Korea

Corresponding author: In-Cheol Park (icpark@kaist.edu)

This work was supported in part by the National Research Foundation of Korea under Grant NRF-2017R1E1A1A01076992, in part by the Center for Integrated Smart Sensors under Grant CISS-2012M3A6A6054192, and in part by the IC Design Education Center (IDEC).

ABSTRACT The successive cancellation list (SCL) decoding is used to achieve good error-correcting performance for practical finite-length polar codes. However, the metric sorting that is repeatedly performed in SCL decoding increases the overall decoding latency as the list size increases, making it difficult to apply the SCL decoding to the applications with limited processing time. To reduce the latency of the metric sorting, this paper proposes a new sorting method derived by analyzing path extension cases encountered in SCL decoding. The proposed method can avoid unnecessary sorting operations by adaptively determining the sorting size, and can be combined with other metric sorting methods. Simulation results show that the proposed method significantly reduces the decoding latency for various list sizes and code rates without degrading the error-correcting performance. The proposed method also becomes more effective as the code rate or the list size increases. Combined with the state-of-the-art sorting method, the proposed method reduces the average SCL decoding latency for a (1024, 512) polar code by 33.1% when the list size is 8.

INDEX TERMS Polar codes, successive cancellation list decoding, metric sorting, large-small sorting, low latency.

I. INTRODUCTION

Polar codes are the first class of error-correcting codes that can probably achieve the channel capacity for any symmetric binary-input discrete memoryless channel (B-DMC), and also have efficient encoding and decoding algorithms [1]. Therefore, polar codes have been selected as one of the channel coding methods for the 5G wireless communication system [2]. Successive cancellation (SC) decoding, the first polar-code decoding algorithm, can achieve the channel capacity when the code length goes to infinity. However, the error-correcting performance of SC decoding is degraded for practical polar codes of which length is finite. To overcome the issue, the successive cancellation list (SCL) decoding presented in [3] maintains the most probable L paths during the decoding. For a non-frozen bit, the SCL decoding extends L paths surviving in the previous decoding step to $2L$ path candidates. Then, it sorts the extended $2L$ path candidates to select L paths with the smallest metrics.

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

Though the error performance of SCL decoding improves as the list size increases, the $2L$-to-L metric sorting increases the overall decoding latency significantly [4] and may deteriorate the operating clock frequency [5]. In order to reduce the decoding latency of SCL decoding, many studies have been conducted. A multi-bit decoding scheme was proposed in [7], [8] by extending [6], and another class of decoding schemes was proposed in [10], [11] by extending the fast simplified successive cancellation (fast-SSC) [9]. However, these works were developed by reformulating the SC decoding algorithm and did not consider the metric sorting latency. In addition, tree pruning and list pruning methods were proposed in [12] and [13] to reduce the computational complexity of SCL decoding. On the other hand, many studies have been conducted to reduce the latency of metric sorting in SCL decoding, which will be briefly introduced in Section II-C.

In this paper, we propose a new sorting method called large-small (LS) sorting to reduce the average latency of metric sorting without sacrificing the error-correcting performance noticeably. While the previous sorting methods focus mainly on speeding up $2L$-to-L metric sorting, the proposed

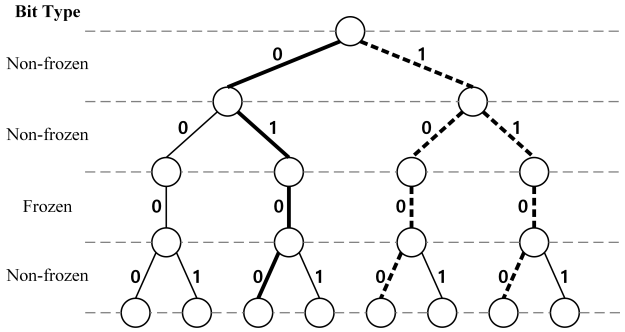

FIGURE 1. Decoding tree representing SC and SCL ($L = 2$) decoding. Thick solid lines correspond to the SC path, and thick dotted lines to the SCL paths.

LS sorting adaptively replaces the $2L$-to-L metric sorting with a number of small-sized sortings that can be conducted in parallel. Simulation results show that the proposed method reduces the average decoding latency of SCL decoding effectively by reducing the metric sorting latency regardless of the code rate and the list size, and becomes more effective as the code rate or the list size increases. In addition, the proposed method can be combined with other state-of-the-art sorting methods to reduce the average latency further.

The rest of the paper is organized as follows. In Section II, we briefly introduce the SCL decoding, define the metric sorting problem, and review the previous works to reduce the latency of the metric sorting. Then the proposed large-small (LS) sorting method is described in Section III. Section IV presents simulation results and analyzes them in terms of error performance and average latency. Finally, concluding remarks are made in Section V.

II. BACKGROUND

Based on the channel polarization phenomenon, an (N, K) polar code with code length $N = 2^n$ and rate $R = K/N$ divides N bit-channels into K reliable ones and $N - K$ unreliable ones. A message is transmitted through the K reliable channels, and the bits corresponding to these channels are called information bits or non-frozen bits. The bits corresponding to the remaining $N - K$ unreliable channels are called frozen bits and are usually set to zero. Throughout this paper, the set of non-frozen bits is denoted as $\mathcal{A}$.

A. SCL DECODING

Fig. 1 shows a tree representation of decoding paths appearing in SC and SCL decoding. The SC decoding is serial and keeps only one path that is the most reliable in each decoding step. Due to its serial nature, if an error occurs in the middle of the decoding process, the error propagates into the subsequent decoding process, causing a decoding fail. On the other hand, the SCL decoding maintains the most probable L paths in every decoding step and finally outputs the most reliable one among them. As shown in Fig. 1, when the SCL decoding reaches a non-frozen bit, every parent candidate is extended to a pair of child candidates corresponding to bit 0 and bit 1. Then, it sorts the metrics of the extended $2L$ child candidates to select L paths with smallest metrics. Although the error-correcting performance of SCL decoding improves as the list size increases, the complexity of $2L$-to-L metric sorting increases drastically, which in turn increases the overall decoding latency.

In order to improve the error performance of SCL decoding further, the cyclic redundancy check (CRC) code is widely adopted as an outer code of the polar code. We will only consider the CRC-aided SCL (CA-SCL) decoding [14], which is also employed in the 5G communication standard [2]. When an r -bit CRC code is used for an (N, K) polar code, the set $\mathcal{A}$ is extended by adding r more non-frozen bits to K information bits, so its cardinality $|\mathcal{A}|$ becomes $K + r$.

B. PROBLEM STATEMENT

In the implementation based on the log-likelihood ratio (LLR) [15] that this paper focuses on, the path metrics of $2L$ child candidates are calculated from the LLR values of parent candidates. Only L candidates with the smallest metrics are selected and maintained as parent candidates in the next decoding step. Let $\mathbf{m} = [m_0, m_1, \cdots, m_{2L-1}]$ denote the metrics of $2L$ child candidates to be sorted, $\mathbf{n} = [n_0, n_1, \cdots, n_{L-1}]$ indicates the metrics of L parent candidates surviving in the previous decoding step, and $\mathbf{a} = [a_0, a_1, \cdots, a_{L-1}]$ is the absolute LLR values that represent the incremental reliability values to be considered in computing the path metrics of child candidates. By the hardware-friendly approximation in [15], $\mathbf{m}$ is calculated as follows:

$$m_{2l} = n_l, \quad \text{for } l = 0, 1, \cdots, L-1 \tag{1}$$

$$m_{2l+1} = n_l + a_l, \quad \text{for } l = 0, 1, \cdots, L-1 \tag{2}$$

Based on (1) and (2), $\mathbf{m}$ has a property of (3).

$$m_{2l} \leq m_{2l+1} \tag{3}$$

Moreover, if $\mathbf{n}$ is already sorted, $\mathbf{m}$ has another property of (4).

$$m_{2l} \leq m_{2(l+1)} \tag{4}$$

In recent studies, these two properties are used to reduce the sorting complexity such as the number of comparisons and that of comparison stages [16]–[21].

C. EXISTING 2L-TO-L METRIC SORTING METHODS

In [5] and [15], a parallel radix-$2L$ sorter [22] is used to sort $2L$ admissible path candidates and select L ones with smallest metrics. However, the radix-$2L$ sorter lies on the critical path that determines the maximum clock frequency of the decoder when the list size is not small [16], and eventually deteriorates the operating frequency. Thus, this paper does not focus on it. To maintain a reasonable clock frequency, various metric sorters have been presented that perform the $2L$-to-L metric sorting with employing multiple comparison stages [4], [16]–[21]. However, their complexity

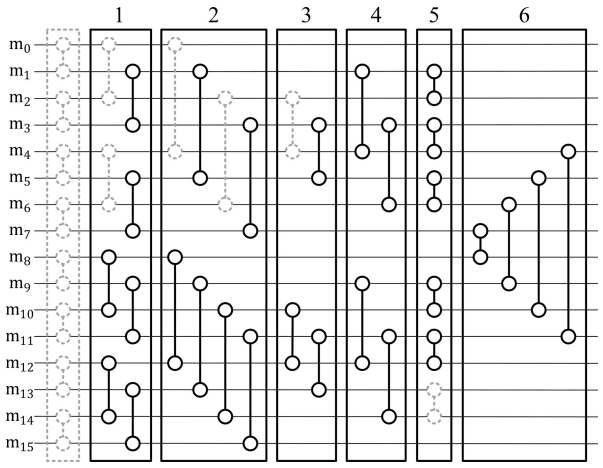

FIGURE 2. Half-sorted pairwise metric sorting (HS-PMS) architecture [20] for $L = 8$. A rectangular box represents a comparison stage and the dotted line denotes a pruned comparison.

is still depending on the list size, which in turn increases the overall decoding latency.

In [19] and [21], a different sorting approach was proposed based on the pruned bitonic extractor (PBE), which divides the conventional metric sorting into two steps: the first step extracts L paths to survive and the second step sorts the metrics of the surviving paths for the next pruning process. The approach is effective in that the second step can be performed in parallel with the next extension process. However, the extension takes a few cycles, usually one cycle, in the decoding process, while the second step takes multiple cycles when the list size is not small. The number of additional clock cycles taken for the second step is usually not negligible as the list size increases. Fig. 2 shows an example of the half-sorted pairwise metric sorting (HS-PMS) [20], one of the state-of-the-art metric sorters. The HS-PMS method utilizes the properties of path metrics actively to lower the sorting latency, but a large number of comparison stages is still required if the list size is not small, making it difficult for the decoder to achieve high throughput. Since the HS-PMS is one of the sorting methods associated with the lowest latency, it is used as the baseline sorting method in evaluating the proposed method.

In addition, a sorting method called double thresholding strategy (DTS) has been presented in [23] to efficiently approximate the conventional $2L$-to-L metric sorting. The DTS method applies two threshold values that are generated in runtime: an acceptance threshold (AT) and a rejection threshold (RT). The $2L$ path metrics are compared in parallel with these two threshold values to select L paths to survive. Allowing a degradation of error-correcting performance, the method is suitable when the list size is large. Although the properties of path metrics expressed in (1)-(4) are no longer valid for the DTS method, the proposed method can use the DTS method to conduct the $2L$-to-L metric sorting.

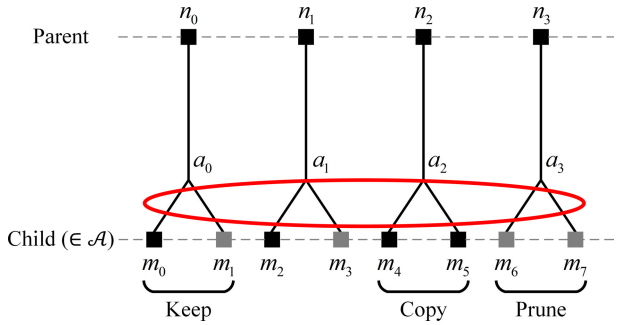

FIGURE 3. Path extension of SCL decoding when $L = 4$. Surviving candidates are denoted as black squares and pruned candidates as gray squares. A large sorting is required in this situation.

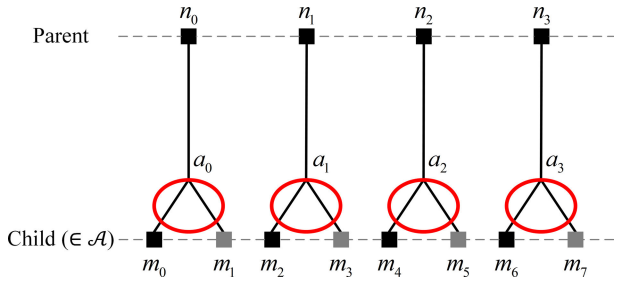

FIGURE 4. All-keep case that can be replaced with a parallel small sorting.

III. PROPOSED METHOD

The proposed large-small (LS) sorting method developed for SCL decoding is described in this section. The objective is to reduce the overall sorting latency by adaptively replacing the $2L$-to-L metric sorting with 2-to-1 metric sortings that can be performed in parallel. The $2L$-to-L sorting and the 2-to-1 sorting are referred to as a large sorting and a small sorting, respectively.

A. PATH EXTENSION CASES IN SCL DECODING

Fig. 3 shows the extension of four paths, causing a large sorting. Each path surviving in the previous decoding step has its own path metric. When the SCL decoding reaches a non-frozen bit, it performs a large sorting to select 4 child candidates with the smallest metrics among the extended 8 child candidates. In the case, the path extensions are classified into three cases. As shown in Fig. 3, a case that only one child candidate survives is called a *keep* case, another case that two child candidates survive is called a *copy* case, and the last case that two child candidates are removed is called a *prune* case. A large sorting is in general required because of the *copy* and *prune* cases. As shown in Fig. 4, when all the paths are *keep* cases, the large sorting of $2L$ child candidates is equivalent to multiple small sortings each of which is conducted for a pair of the child candidates extended from a parent path.

B. LARGE-SMALL SORTING METHOD FOR SCL DECODING

Performing a parallel small sorting is equivalent to performing only the first stage of a multi-staged sorting architecture.

Algorithm 1 Large-Small Sorting for SCL Decoding

Input:
The metrics of L parent candidates: $\mathbf{n}$
The L incremental reliability values for each parent candidate: $\boldsymbol{a}$
Output:
The metrics of surviving L child candidates: $\mathbf{n}'$

```
for l = 0, 1, ..., L − 1 do
    m_{2l} ← n_l
    m_{2l+1} ← n_l + a_l
end for
if a_l > Ω for ∀l then
    n' ← parallel small sorting for n
else
    n' ← large sorting for n
end if
```

TABLE 1. Optimal threshold values for various code rates and SNRs. $L = 8$ and $N = 1024$.

SNR [dB]	Code Rate			
	1/3	1/2	2/3	5/6
1.0	7.0	6.0	7.5	5.0
1.5	7.5	6.5	7.5	5.0
2.0	8.0	8.0	8.0	5.0
2.5	8.0	9.0	8.0	5.0
3.0	5.5	9.5	9.5	7.0
3.5	-	4.5	7.5	7.0
4.0	-	-	-	8.5
4.5	-	-	-	7.5

In other words, the small sorting plays a crucial role in reducing the overall sorting latency of SCL decoding. In addition, the *copy* case requires to copy the path information of the corresponding parent to its child, which further increases the computational complexity as well as the latency [3], [15]. It is worth noting that the small sorting case does not need the copy operation, being effective in reducing the overall latency.

In a SCL decoding step corresponding to a non-frozen bit, only L child candidates with the smallest metrics survive, so that metric differences among the surviving L candidates are assumed to be not large. Based on the assumption, if the decoding reliability values of all the paths are relatively large, then all the paths are assumed to be *keep* cases, which induces a parallel small sorting. For example, in Fig. 4, when $[n_0, n_1, n_2, n_3] = [10, 13, 15, 18]$ and $[a_0, a_1, a_2, a_3] = [13, 10, 9, 23]$, the large sorting is equivalent to four-parallel small sorting applied to two child candidates of every parent. To decide whether to apply a large sorting or not, we introduce a threshold value Ω of decoding reliability. The proposed sorting method is described in Algorithm 1, where only copy cases are considered for the first $\log_2 L$ non-frozen bits so as to make L paths.

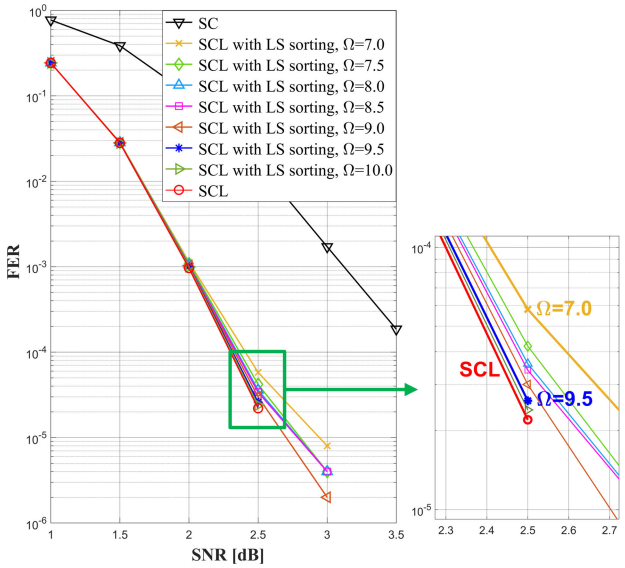

FIGURE 5. The effect of threshold values on the error-correcting performance of a (1024, 512) polar code. $L = 8$.

TABLE 2. Average small-sorting ratio obtained for 1024-bit polar codes.

List size (L)	Code Rate			
	1/3	1/2	2/3	5/6
2	0.90	0.91	0.91	0.95
4	0.79	0.82	0.84	0.90
8	0.69	0.76	0.79	0.87
16	0.59	0.69	0.74	0.84

IV. SIMULATION RESULTS

All simulations have been performed with the binary phase-shift keying (BPSK) modulation and the additive white Gaussian noise (AWGN) channels. To demonstrate the error-correcting performance of the proposed SCL decoder, various 1024-bit polar codes with 16-bit CRC codes, adopted in the 5G standard [2], have been simulated. Note that the CRC code is not used for the SC decoder. The channel LLR values are quantized with 4 integer bits and 1 fractional bit, while the internal LLR values with 6 integer bits and 1 fractional bit. In addition, the path metrics are quantized with 8 integer bits and 1 fractional bit.

Fig. 5 shows how the error-correcting performance varies according to the threshold value. As shown in Algorithm 1, the small sorting is conducted when the incremental decoding reliability values of all the paths exceed the threshold value. As shown in Fig. 5, the error-correcting performance of the proposed SCL decoder improves according to the increase of the threshold value, which is natural as the large sorting is more likely applied in the case.

We can take a trade-off between the error-correcting performance and the small-sorting ratio by changing the threshold value, and the optimal threshold value would be depending on the SNR, the code rate, and the list size.

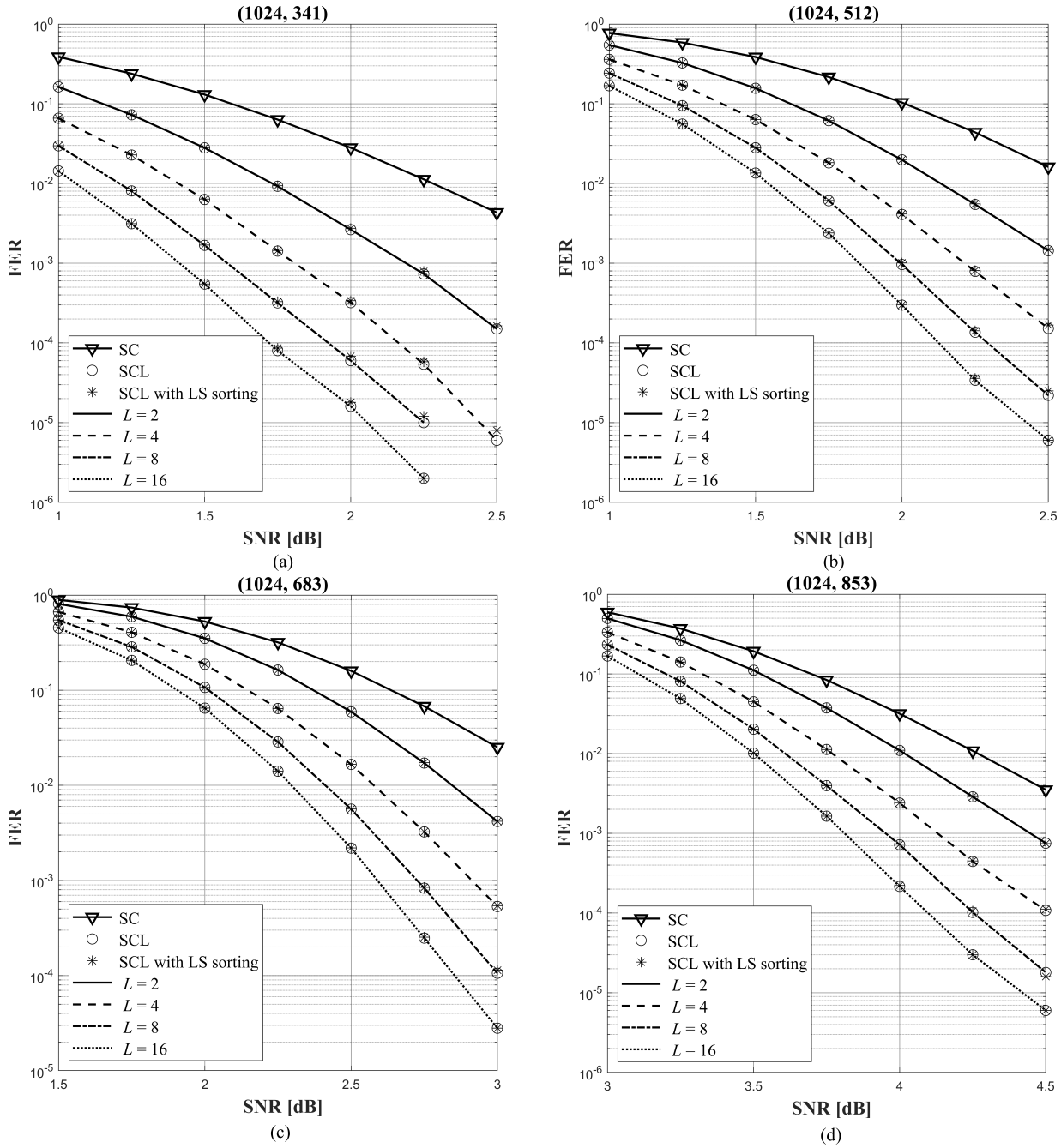

FIGURE 6. Error-correcting performance for code rates of (a) 1/3, (b) 1/2, (c) 2/3, and (d) 5/6.

Let 1024-bit polar codes with various code rates be decoded by the proposed SCL decoder with a list size of 8. In that case, the optimal threshold values obtained from intensive simulations are shown in Table 1.

As hardware design prefers a fixed threshold value applicable for all SNRs and code rates, the proposed SCL decoder has been simulated with fixed threshold values of 8.0, 9.5, 9.5, and 10.5 for $L = 2$, 4, 8, and 16, respectively.

Fig. 6 shows the error-correcting performances achieved for various code rates and list sizes, where we can see that the error-correcting performance of the proposed SCL decoder is not degraded at all for all the list sizes and code rates simulated.

The small-sorting ratio, which is the number of small sortings over the total number of sortings, is shown in Fig. 7, and the average small-sorting ratio is numerically summarized in Table 2, which clearly indicate that the small sorting occurs quite frequently for diverse configurations. Moreover, the small-sorting ratio increases as the code rate increases, which means that the proposed method becomes more

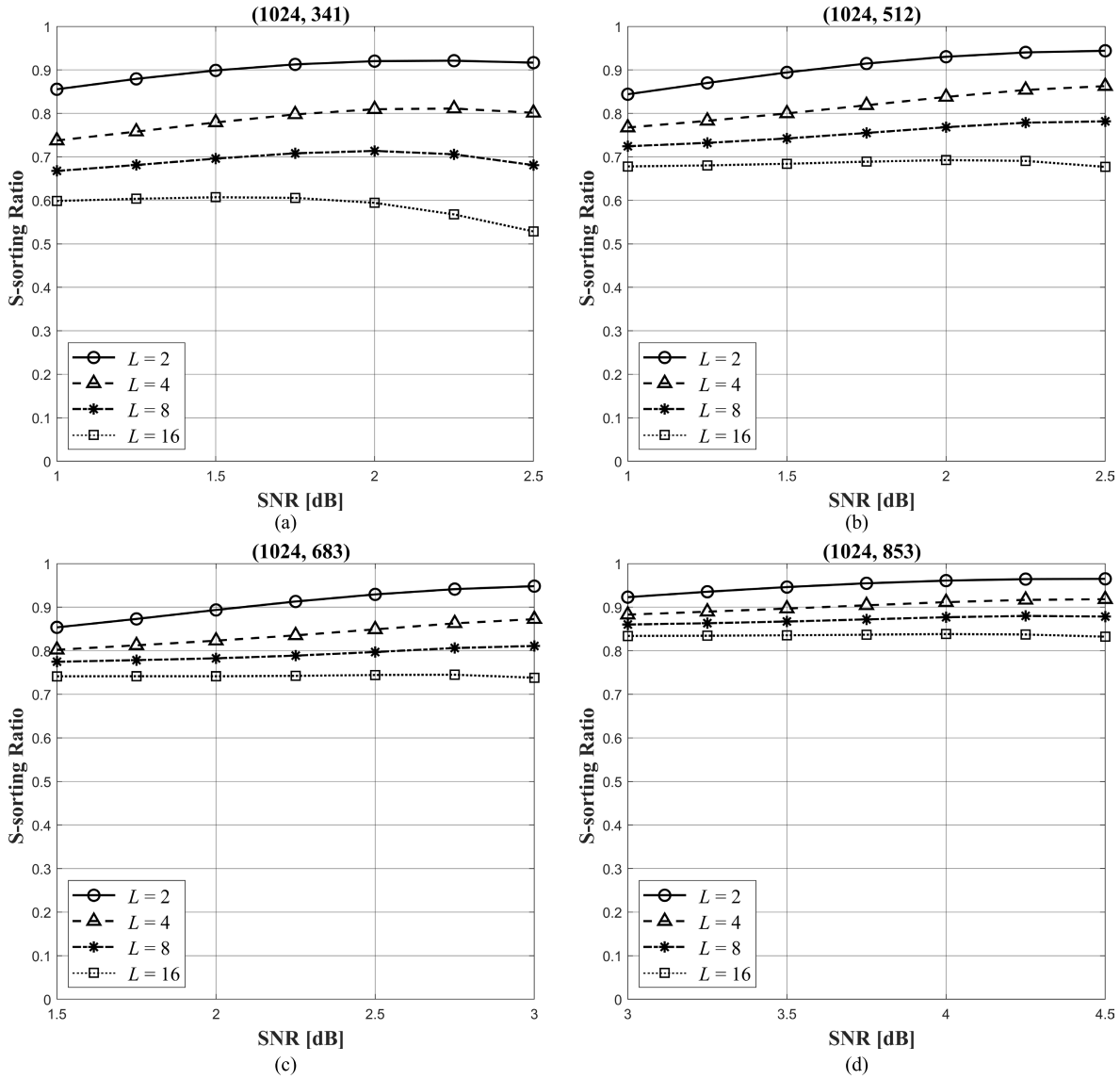

FIGURE 7. Small-sorting ratio for code rates of (a) 1/3, (b) 1/2, (c) 2/3, and (d) 5/6.

effective according to the increase of the code rate. Although the small-sorting ratio decreases gradually as the list size increases, it is still significant. In addition, the sorting complexity increases as the list size increases, so the proposed method is also effective for polar decoders with large lists. For a (1024, 512) polar code, the proposed SCL decoder with a list size of 8 reduces the overall sorting latency significantly by replacing 76% of the large sortings with the parallel small sortings.

The average sorting latency and the overall decoding latency of the proposed SCL decoder are compared with those of the conventional SCL decoder in Table 3. As mentioned in Section II-C, the HS-PMS [20] is used for the large sorting. To emphasize the contribution of this work, we consider a full-parallel SC decoder [1]. The maximum operating frequency of the decoder is determined by considering the logic delays of the representative operations needed in SC decoding: f and g operations [15], path-metric update, a comparison stage in metric sorting, and the copy of path information. Most of the previous works have assumed that the f and g operations and the metric sorting operation take one clock cycle. The f operation chooses the minimum value from two inputs, which can be achieved by performing subtraction, and the g operation performs either addition or subtraction depending on the bit value determined previously. The metric sorting consists of a number of comparison stages, each of which conducts subtraction to realize the comparison. If the cycle time is mainly determined by the delay of either f or g operation, the metric sorting must take as many cycles as the number of comparison stages. Based on the analysis above, a comparison stage is assumed to take one cycle in the metric sorting architecture. In the conventional SCL decoder, the

TABLE 3. Average SCL decoding latency with and without LS Sorting. The latency is measured in the number of cycles.

Code rate		1/3				1/2				2/3				5/6			
$\lvert\mathcal{A}\rvert$		357				528				699				869			
List size (L)		2	4	8	16	2	4	8	16	2	4	8	16	2	4	8	16
SCL [20]	S.T[a]	357	1071	2142	3570	528	1584	3168	5280	699	2097	4194	6990	869	2607	5214	8690
	C.T[b]	357	357	357	357	528	528	528	528	699	699	699	699	869	869	869	869
	D.T[c]	4299	5013	6084	7510	4641	5697	7281	9391	4983	6381	8478	11272	5323	7061	9668	13142
	N.T[d]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SCL with LS sorting	S.T	357	507	910	1674	528	718	1162	2001	699	923	1433	2335	869	1043	1434	2120
	C.T	36	75	111	146	48	95	127	164	63	112	147	182	43	87	113	139
	D.T	3978	4167	4606	5403	4161	4398	4874	5748	4347	4620	5165	6100	4497	4715	5132	5842
	N.T	1.08	1.20	1.32	1.39	1.12	1.30	1.49	1.63	1.15	1.38	1.64	1.85	1.18	1.50	1.88	2.25
Reduction in S.T (%)		0.0	52.7	57.5	53.1	0.0	54.7	63.3	62.1	0.0	56.0	65.8	66.6	0.0	60.0	72.5	75.6
Reduction in C.T (%)		89.9	79.0	68.9	59.1	90.9	82.0	75.9	68.9	91.0	84.0	79.0	74.0	95.1	90.0	87.0	84.0
Reduction in D.T (%)		7.5	16.9	24.3	28.1	10.3	22.8	33.1	38.8	12.8	27.6	39.1	45.9	15.5	33.2	46.9	55.5

[a] The total time for sorting operations
[b] The total time for copying operations
[c] The total time for SCL decoding
[d] The normalized throughput based on the SCL decoding without LS sorting

TABLE 4. Average SCL decoding latency obtained with two different baseline sorting methods.

List size	SCL with LS sorting based on HS-PMS [20]				SCL with LS sorting based on EPBE [21]			
	Code Rate				Code Rate			
	1/3	1/2	2/3	5/6	1/3	1/2	2/3	5/6
2	3978	4161	4347	4497	3978	4161	4347	4497
4	4167	4398	4620	4715	4134	4356	4573	4672
8	4606	4874	5165	5132	4949	5325	5663	5533
16	5403	5748	6100	5842	7508	8209	8757	8050

total number of clock cycles taken for metric sorting can be expressed as $\omega \times |\mathcal{A}|$, where ω is the number of comparison stages that depends on the list size. It is assumed that the L path metric values calculated at the frozen bit located in front of each non-frozen bit are already sorted. Since a small sorting takes only one clock cycle, the average number of clock cycles taken for the entire metric sortings is reduced to $(1-\eta)\times\omega \times |\mathcal{A}| + \eta \times |\mathcal{A}|$ in the proposed SCL decoder, where η is the small-sorting ratio in Table 2.

If the list size is 2, only one clock cycle is taken for the large sorting, as one comparison stage is sufficient for the sorting. In the case, the proposed decoder has no advantage in sorting, but can reduce the overall processing time by omitting the copy of path information. As the list size or the code rate increases, the latency of metric sorting increases and dominates the overall decoding latency. The proposed SCL decoder reduces the overall decoding latency by reducing the average latency of metric sorting significantly. Table 3 shows the detailed processing latencies taken for critical operations, which are measured with varying the code rate and the list size. To apply the proposed sorting method, additional operations are required to compare the incremental reliability values and the threshold value. As the operations can be performed in parallel with the path-metric update operation, it does not affect the overall decoding latency. In Table 3, it is clear that the proposed sorting method plays a more significant role as the list size increases, and becomes more effective as the code rate increases. If the proposed SCL decoder with a list size of 8 is used to decode a (1024, 512) polar code, for example, it reduces the sorting latency by 63.3% and improves the overall decoding throughput by 1.49 times, compared to the conventional SCL decoder.

The decoding latencies of the proposed SCL decoder are measured once more by changing the baseline large-sorting architecture to the EPBE method [21], which are summarized in Table 4. Among PBE-based sorting methods, the EPBE is the most effective one in lowering the latency. If the second step of PBE sorting is not hidden by the next expansion process, the number of additional cycles taken for the second step is considered in calculating the overall latency. As indicated in Table 4, the latency resulting from the EPBE is similar to that obtained from the HS-PMS when the list size is small, but it becomes much longer as the list size increases.

V. CONCLUSION

This paper has proposed a new sorting method called LS sorting to reduce the SCL decoding latency of polar codes. The proposed method is to adaptively replace the large sorting with a parallel small sorting, which is derived by analyzing path extension cases. To determine which sorting method is applied, a threshold value is introduced; the incremental reliability values of all L paths are greater than the threshold value, the small sorting is used to reduce the overall latency without degrading the error-correcting performance noticeably. The simulations performed for various list sizes and code rates have shown that the proposed method can reduce the metric sorting latency with almost no loss of error-correcting performance regardless of the list size and code rate. Furthermore, the proposed method reduces the copy of intermediate path information, which is usually conducted after the metric sorting is completed. In our experiments, the SCL decoding latency of a (1024, 512) polar code is reduced by 33.1% on the average when the list size is 8. The proposed method becomes more significant as the code rate or list size increases.

The proposed method is to reduce the SCL decoding latency by replacing the complex $2L$-to-L metric sorting with simple 2-to-1 sorting when a certain condition is met. Though the proposed method is applied to the leaf node of the SCL decoding tree, it can be applied to the internal node of the decoding tree, as was done in the simplified SCL (SSCL) decoding scheme [10], [11]. As the LLR values at the leaf nodes and those at the internal nodes may be different in their ranges, the threshold value to be compared should be determined by performing intensive simulations.

REFERENCES

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
[2] *Multiplexing and Channel Coding*, document TS 38.212, 3GPP, 2018.
[3] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2011, pp. 1–5.
[4] J. Lin and Z. Yan, "An efficient list decoder architecture for polar codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2508–2518, Nov. 2015.
[5] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 8, pp. 609–613, Aug. 2014.
[6] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-Bit decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 4, pp. 1241–1254, Apr. 2014.
[7] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation list decoders for polar codes with multibit decision," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2268–2280, Oct. 2015.
[8] B. Yuan and K. K. Parhi, "LLR-based successive-cancellation list decoder for polar codes with multibit decision," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 1, pp. 21–25, Jan. 2017.
[9] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 946–957, May 2014.
[10] S. A. Hashemi, C. Condo, and W. J. Gross, "A fast polar code list decoder architecture based on sphere decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 12, pp. 2368–2380, Dec. 2016.
[11] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast and flexible successive-cancellation list decoders for polar codes," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5756–5769, Nov. 2017.
[12] K. Chen, B. Li, H. Shen, J. Jin, and D. Tse, "Reduce the complexity of list decoding of polar codes by tree-pruning," *IEEE Commun. Lett.*, vol. 20, no. 2, pp. 204–207, Feb. 2016.
[13] J. Chen, Y. Fan, C. Xia, C.-Y. Tsui, J. Jin, K. Chen, and B. Li, "Low-complexity list successive-cancellation decoding of polar codes using list pruning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6.
[14] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, Oct. 2012.
[15] A. Balatsoukas-Stimming, M. Bastani Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.
[16] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "On metric sorting for successive cancellation list decoding of polar codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 1993–1996.
[17] B. Yong Kong, H. Yoo, and I.-C. Park, "Efficient sorting architecture for successive-cancellation-list decoding of polar codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 7, pp. 673–677, Jul. 2016.
[18] B. Y. Kong and I.-C. Park, "Hybrid sorting architecture for low-latency successive cancellation list decoding of polar codes," *J. Semicond. Technol. Sci.*, vol. 18, no. 5, pp. 593–601, Oct. 2018.
[19] V. Bioglio, F. Gabry, L. Godard, and I. Land, "Two-step metric sorting for parallel successive cancellation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 456–459, Mar. 2017.
[20] H. Li, "Enhanced metric sorting for successive cancellation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 664–667, Apr. 2018.
[21] X. Wang, T. Wang, J. Li, L. Shan, H. Cao, and Z. Li, "Improved metric sorting for successive cancellation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 23, no. 7, pp. 1123–1126, Jul. 2019.
[22] L. G. Amaru, M. Martina, and G. Masera, "High speed architectures for finding the first two Maximum/Minimum values," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 12, pp. 2342–2346, Dec. 2012.
[23] Y. Fan, J. Chen, C. Xia, C.-Y. Tsui, J. Jin, H. Shen, and B. Li, "Low-latency list decoding of polar codes with double thresholding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1042–1046.

KYUNGPIL LEE (Student Member, IEEE) received the B.S. degree in electronic engineering from Kwangwoon University, Seoul, South Korea, in 2019. He is currently pursuing the M.S. degree with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. His current research interests include algorithms and very-large-scale integration architectures for error-correction codes and communication systems.

IN-CHEOL PARK (Senior Member, IEEE) received the B.S. degree in electronic engineering from Seoul National University, Seoul, South Korea, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1988 and 1992, respectively.

From 1995 to 1996, he was with IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, where he was involved in the research of high-speed circuits design. Since 1996, he has been an Assistant Professor and is currently a Professor with the School of Electrical Engineering, KAIST. His current research interests include very-large-scale integration architectures for digital signal processing, neural networks, and general-purpose microprocessors.

Dr. Park was a recipient of the Best Design Award at ASP-DAC, in 1997, and the Best Paper Award at ICCD, in 1999.