IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 17, 2021, accepted September 11, 2021, date of publication September 14, 2021,
date of current version September 23, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3112738*

# Interleaved Local Sorting for Successive Cancellation List Decoding of Polar Codes

**WOOYOUNG KIM, (Student Member, IEEE), YUJIN HYUN, (Student Member, IEEE), JAEYOUNG LEE, 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, and in part by the Ministry of Science and ICT, South Korea, under Grant IITP-2020-0-01847.

**ABSTRACT** In the successive cancellation list decoding of polar codes, the metric sorting dominates the overall decoding latency. To reduce the latency of metric sorting, this paper proposes a new sorting method, called interleaved local sorting, which divides the metrics to be sorted into several groups and locally sorts each group independently. In addition, an interleaving scheme is proposed to recover the performance degradation caused by the local sorting. A hardware architecture effective in reducing the overall latency as well as the hardware complexity is also proposed based on the proposed metric sorting. The evaluation results show that the proposed interleaved local sorting architecture outperforms the state-of-the-art metric sorting architectures in terms of latency and hardware complexity when the list size is not small.

**INDEX TERMS** Polar codes, successive cancellation list decoding, metric sorting, interleaved local sorting.

## I. INTRODUCTION

The 3rd generation partnership project (3GPP) [1] has selected polar codes, a class of channel codes invented by Arikan [2], as a channel coding scheme of the emerging 5G, the latest standard of mobile communication systems. The polar codes can achieve the channel capacity by performing the successive cancellation (SC) decoding when the code length reaches infinity [3], [4]. To improve the error-correction performance of SC decoding for finite-length polar codes, the successive cancellation list (SCL) decoding that maintains $L$ most probable paths was proposed in [3]. Increasing the list size in general improves the error-correcting performance of SCL decoding, but makes the metric sorting take the longer processing time. If the list size is not small, therefore, the metric sorting dominates the overall latency of SCL decoding [5], [6].

The importance of metric sorting has led to many sorting algorithms and architectures in recent studies. The pruned bitonic sorting (PBS) architecture in [7] exploits some properties of metrics to reduce the number of comparisons and comparison stages. The simplified bubble sorting (SBS) architecture, which was also introduced in [7], requires fewer comparisons and comparison stages when the list size is small. In [8], the odd-even sorting (OES) architecture outperforms PBS, requiring a less number of comparison stages. The pairwise metric sorting (PMS) architecture in [9] reduced the hardware complexity while maintaining the same number of comparison stages. In [10], the pruned bitonic extractor (PBE) divided the extraction of the $L$ smallest metrics and the sorting of the $L$ metrics into two steps. Two improved PBEs, efficient PBE (EPBE) and OES-based PBE (OPBE), reduced the number of comparisons and comparison stages further by revising the sorters of the first and second groups in the PBE [11].

To further minimize the number of comparison stages and the computational delay taken for metric sorting, this paper proposes a new sorting architecture called interleaved local sorting, which is effective when the list size is not small. The proposed sorting is a class of approximate sorting, as it divides the entire metrics to be sorted into several groups and then sorts the metrics of each group independently of the other groups. As the number of metrics to be considered in the sorting is significantly reduced by the grouping, the proposed sorting architecture, which is designed based on the odd-even merging network [12], minimizes the comparison stages effectively. Due to its approximate nature, however, the error-correcting performance could be degraded

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

especially when the number of groups is large. To mitigate the performance degradation, group elements are interleaved before conducting the local sorting. Intensive simulations has revealed that the proposed interleaving is effective in maintaining the error-correcting performance. Compared to the state-of-the-art sorting architectures, the proposed sorting architecture achieves the fewest numbers of comparisons and comparison stages.

## II. BACKGROUND

In this Section, we briefly describe the SCL decoding, define the sorting problem and review the existing sorting architectures.

### A. SCL DECODING

When the SCL decoding procedure involving $L$ parent paths reaches an information bit, each path extends to a pair of two child candidates each of which decides the information bit to either 0 or 1. In the implementation based on the log-likelihood ratio (LLR) [5], which the recent studies are focused on, the metrics of $2L$ child candidates are calculated by using the LLR values of the information bit and the $L$ parent paths. Among $2L$ metrics of the child candidates, the $L$ candidates associated with the smallest metrics are selected to be the parent paths in the next decoding.

### B. PROBLEM DEFINITION

Let $\mathbf{m} = [m_0, m_1, \cdots, m_{2L-1}]$ denote the metrics of $2L$ child candidates, $\mathbf{n} = [n_0, n_1, \cdots, n_{L-1}]$ be the metrics of $L$ surviving paths, and $\mathbf{a} = [a_0, a_1, \cdots, a_{L-1}]$ represent the absolute values of $L$ LLRs corresponding to the information bit to be decoded. The hardware-friendly approximation [6] computes the elements in $\mathbf{m}$ as follows:

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

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

According to (1) and (2), $\mathbf{m}$ has the property expressed in (3). In addition, if the elements of $\mathbf{n}$ are already sorted, they have the property in (4).

$$m_{2l} \leq m_{2l+1} \quad l = 0, 1, \cdots, L-1 \tag{3}$$

$$m_{2l} \leq m_{2(l+1)} \quad l = 0, 1, \cdots, L-2 \tag{4}$$

In recent studies, the first property (3) and the second property (4) are intensively utilized to alleviate the computational complexity of the metric sorting by reducing the numbers of comparisons and comparison stages, and eventually to derive metric sorting methods suitable for LLR-based SCL decoders that decide one information bit in a decoding step [7]–[9].

To improve the maximum-likelihood (ML) bound of SCL decoding, the cyclic redundancy check (CRC) code has widely been adopted as an outer code to the polar code. The CRC code elongates the minimum distance between codewords, improving the ML bound dramatically at the cost of some additional complexity [13], [14]. The CRC-aided SCL (CA-SCL) decoding is thus regarded as a baseline in the 5G communication standard [1], and this paper assumes the CA-SCL decoding.

### C. EXISTING SORTING ARCHITECTURES

Recent sorting architectures appearing in the literature will be reviewed briefly in this sub-section. Some of them have utilized the known relations of metrics to reduce the computational complexity and sorting delay.

The odd-even sorting (OES) architecture in [8] is illustrated in Fig. 1, where a vertical line with circles at both ends stands for a comparison operation between two inputs at the circles. The comparison operation switches the two inputs if the upper input is larger than the lower input, or outputs the two inputs without changing the order otherwise. The input metrics are divided into an odd group and an even group to process the sorted metrics and unsorted ones separately. In order to avoid redundant operations, all the comparisons related to the sorted metrics are eliminated. By exploiting the metric properties further, some comparisons at the middle stages are also removed. Such a pruned comparison is indicated with a dotted line in Fig. 1. The $S(L)$ and $C(L)$ required in the OES architecture are as follows,

$$S^{\mathrm{OES}}(L) = \frac{1}{2}(\log_2 L + 1)(\log_2 L + 2) - 1, \tag{5}$$

$$C^{\mathrm{OES}}(L) = \log_2 L(\frac{L}{4}\log_2 L - 1) + \frac{7L}{4} - 2. \tag{6}$$

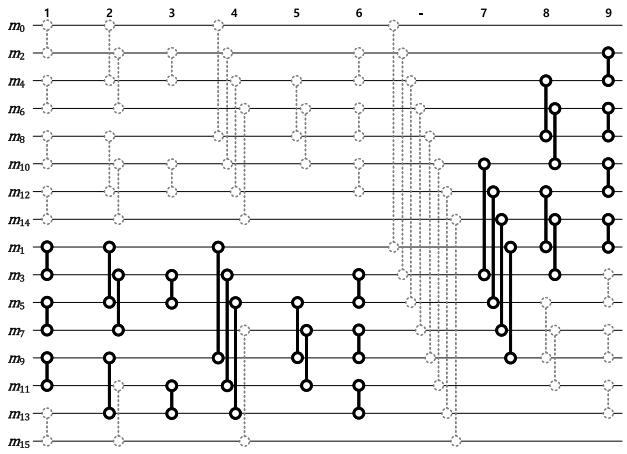


**FIGURE 1.** Odd-even sorting architecture for $2L = 16$ inputs [8].

The pairwise metric sorting (PMS) architectures have been studied based on the pairwise sorting network [15], two of which are shown in Fig. 2. The pairwise sorting process consisting of a group-dividing step and a group combination step requires the same complexity as the odd-even merge-sort network in terms of the numbers of comparisons and comparison stages. Assuming that the input metrics are sorted, however, one of the PMS method called full-sorted PMS (FS-PMS) has reduced the numbers of comparisons and comparison stages

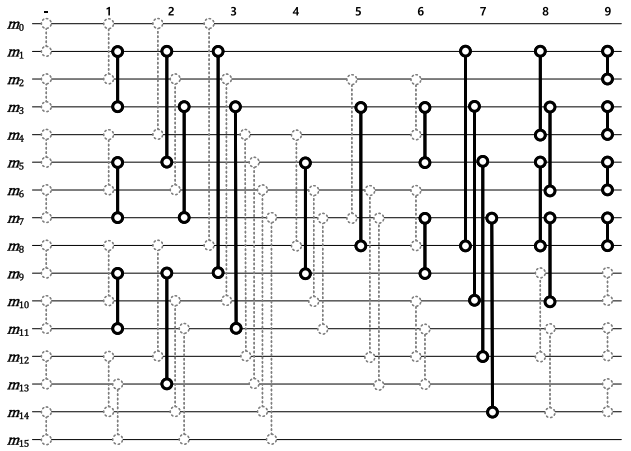


**FIGURE 2.** Sorting architectures for $2L = 16$ inputs. (a) Full-sorted pairwise metric sorting and (b) Half-sorted pairwise metric sorting [9].

significantly by fully exploiting the properties mentioned previously [9]. Another PMS method named half-sorted PMS (HS-PMS) extracts only $L$ smallest metrics among $2L$ metrics by considering the relations among $L$ smallest metrics. Both FS-PMS and HS-PMS reduced $S(L)$ and $C(L)$, which are given as follows,

$$S^{\text{FS-PMS}}(L) = \frac{1}{2}(\log_2 L + 1)(\log_2 L + 2) - 1, \tag{7}$$

$$C^{\text{FS-PMS}}(L) = \frac{\log_2 L}{4}((\log_2 L)(L-2) - L - 6) + 3L - 3, \tag{8}$$

$$S^{\text{HS-PMS}}(L) = \frac{1}{2}(\log_2 L)(\log_2 L + 1), \tag{9}$$

$$C^{\text{HS-PMS}}(L) = \frac{L}{8}\log_2 L(3\log_2 L - 5) + \frac{11L}{4} - \log_2 L - 3. \tag{10}$$

Another sorting method is the pruned bitonic extractor (PBE) based on a two-step sorting [10]. The first step is to extract the smallest $L$ metrics, and the second step is to sort the extracted $L$ metrics by running the PBE three times. A PBE consists of three different groups as shown in Fig. 3. The first group is a fully bitonic sorter (FBS) (the up-left-dotted box in Fig. 3). The $S(L)$ and $C(L)$ required by a FBS are $S^{\text{FBS}}(L) = 1/2(\log_2 L + 1)(\log_2 L + 2)$, and $C^{\text{FBS}}(L) = L/2(\log_2 L + 1)(\log_2 L + 2)$, repectively. The second group is a simplified bubble sorter (SBS) or a pruned bitonic sorter (PBS) (the bottom-left-dotted box in Fig. 3), and the third group is one stage PBS (the right-dotted-box in Fig. 3) that can extract the $L$ smallest metrics from the two sorted groups. The $S(L)$ and $C(L)$ required by a SBS are $S^{\text{SBS}}(L) = L - 1$, and $C^{\text{SBS}}(L) = L/2(L-1)$. This method reduces $S(L)$ and $C(L)$ significantly compared to the dedicated sorters, which are given as follows,

$$S^{\text{PBE}}(L) = S^{\text{SBS}}(\frac{L}{2}) + 1, \tag{11}$$

$$C^{\text{PBE}}(L) = C^{\text{FBS}}(\frac{L}{4}) + C^{\text{SBS}}(\frac{L}{2}) + \frac{L}{2}. \tag{12}$$

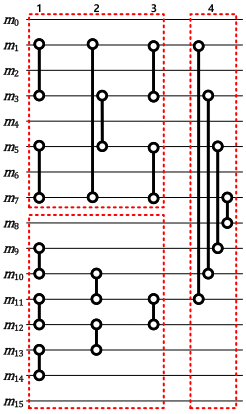

**FIGURE 3.** Pruned bitonic extractor architecture for $2L = 16$ inputs [10].

An efficient PBE (EPBE) and an OES-based PBE (OPBE) were proposed in [11] to further improve the PBE [10]. It was observed that the first stage of the FBS in the first group has the same functionality as the PBS in the third group. Based on this, the EPBE eliminates the first stage of the FBS to reduce CASUs as shown in Fig. 4(a). On the other hand, the OPBE replaces a FBS in the first group with a full OES (FOES) since the number of CASUs required by the FBS is larger than that of the FOES as depicted in Fig. 4(b). The $S(L)$ and $C(L)$ of FOES are $S^{\text{FOES}}(L) = \frac{1}{2}(\log_2 L + 1)(\log_2 L + 2)$ and $C^{\text{FOES}}(L) = 2L + L/2\log_2 L(\log_2 L + 1) - 1$, respectively. The $S(L)$ and $C(L)$ for the EPBE and OPBE are given in (13), (14) and (15).

$$S^{\text{EPBE}}(L) = S^{\text{OPBE}}(L) = \begin{cases} S^{\text{SBS}}(\frac{L}{2}) + 1, & L < 32. \\ S^{\text{FS-PMS}}(\frac{L}{2}) + 1, & L \geq 32. \end{cases} \tag{13}$$

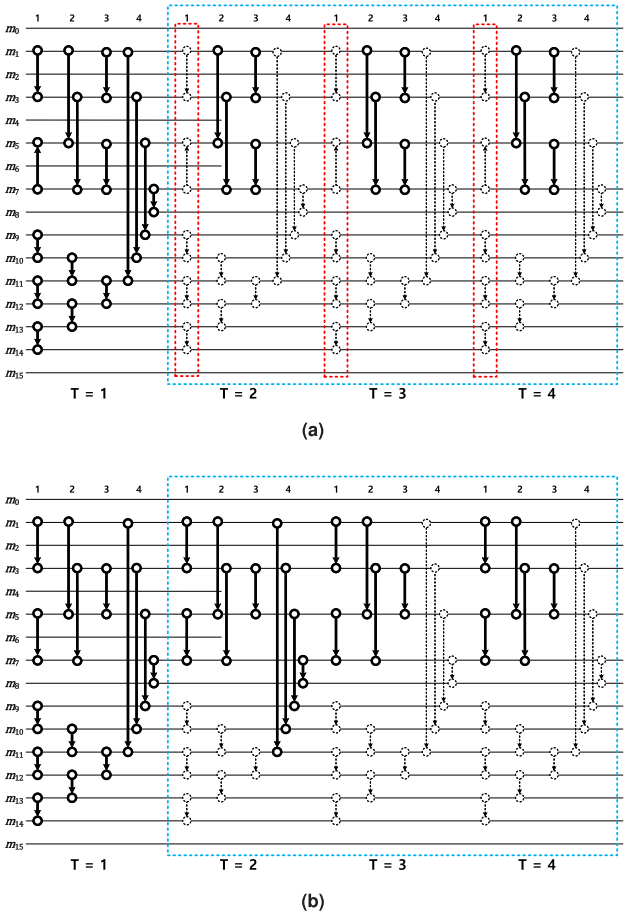


**FIGURE 4.** (a) Efficiently pruned bitonic extractor architecture and (b) odd-even sorting based pruned bitonic extractor architecture for $2L = 16$ inputs [11].

$$C^{\mathrm{EPBE}}(L) = \begin{cases} C^{\mathrm{FBS}}(L/4) + C^{\mathrm{SBS}}(L/2) + L/2, & L < 32. \\ C^{\mathrm{FBS}}(L/4) + C^{\mathrm{FS\text{-}PMS}}(L/2) + L/2, & L \geq 32. \end{cases} \quad (14)$$

$$C^{\mathrm{OPBE}}(L) = \begin{cases} C^{\mathrm{FOES}}(L/4) + C^{\mathrm{SBS}}(L/2) + L/2, & L < 32. \\ C^{\mathrm{FOES}}(L/4) + C^{\mathrm{FS\text{-}PMS}}(L/2) + L/2, & L \geq 32. \end{cases} \quad (15)$$

## III. PROPOSED METHOD AND ARCHITECTURE

While the previous sorting architectures have reduced the hardware complexity significantly, the sorting latency is still not sufficient in achieving a high-throughput decoder when the list size is not small. Though the previous works have focused on exact sorters and have aimed at reducing the hardware complexity and the sorting latency by fully utilizing the properties of path metrics stated in (3) and (4), the proposed interleaved local sorting (ILS) is to reduce the sorting latency and complexity further by applying approximate sorting instead of exact sorting. If the proposed ILS does not degrade the performance severely, it is effective in reducing the hardware complexity and processing latency of SCL decoding. This section also presents a hardware architecture for the proposed ILS.

Suppose that the list size is $L$ and we want to find $L$ smallest elements from $2L$ candidates. The local sorting divides the $2L$ elements into $G$ groups with $2k$ elements per group, i.e., $2L = 2k \times G$. Then, $2k$ elements in a group are sorted independently of the other groups, and the resulting $k$ smallest elements in each group constitute the final output, which means that the overall processing time of the local sorting is determined by the number of elements in a group, $2k$, rather than $2L$ of the exact sorter. However, the local sorting leads to inexact results if the $L$ smallest elements are not uniformly distributed to the groups, and usually induces a severe degradation of error-correcting performance. To maximize the probability that the $L$ smallest metrics will survive the local sorting, the metrics in a group should be distributed uniformly to different groups. In addition, the metrics in a spread group should be diverse enough to make the small metrics survive there. Note that the desired interleaver is completely different from the conventional interleaver of which objective is to distribute the data randomly.

The elements to be sorted are interleaved across groups in order to collect suitable elements into a group, which mitigates the performance degradation. There are many possible interleaving patterns, and an interleaving pattern that is appropriate for a sequence of metrics may be not good for other sequences. To achieve almost no degradation of error-correcting performance for general cases, a systematic interleaving method is proposed in this paper which is exemplified in Fig. 5, where $L = 8$ and $G = 2$, $G = 4$, $G = 8$. First, the metrics of child candidates are divided into $G$ groups as

$$m_l = m_{\lfloor \frac{l}{2k} \rfloor, l - \lfloor \frac{l}{2k} \rfloor \times 2k} \quad l = 0, 1, \cdots, 2L-1. \quad (16)$$

The first subscript denotes the group index, and the second one is the index in a group. Then, the metrics in a group, $G_i$, are rotated by $i\%2k$ where $\%$ represents the modulo operation. After shifting, the $j$-th element in the $i$-th group is reorganized as

$$m_{i, i+j-\lfloor \frac{i+j}{2k} \rfloor \times 2k}, \quad j = 0, 1, \cdots, 2k-1. \quad (17)$$

Finally, the elements in a rotated group are spread across all the groups. In general, the metric located at the $j$-th element of the $i$-th rotated group is relocated to the $(2k \times \lfloor \frac{i}{2k} \rfloor + (j\%G))$-th group. If $2k$ is equal or larger than $G$, the metric is simply moved to the $(j\%G)$-th group as shown in Figs. 5(a) and (b), since the term $(2k \times \lfloor \frac{i}{2k} \rfloor)$ is removed to zero. Otherwise, if $2k$ is smaller than $G$, the metric is relocated according to the general form as exemplified in Fig. 5(c).

In the proposed interleaver, the first step of rotation changes the index of metrics for each group, and the second step of regular interleaving distributes the metrics uniformly to different groups. The first step is needed to make a

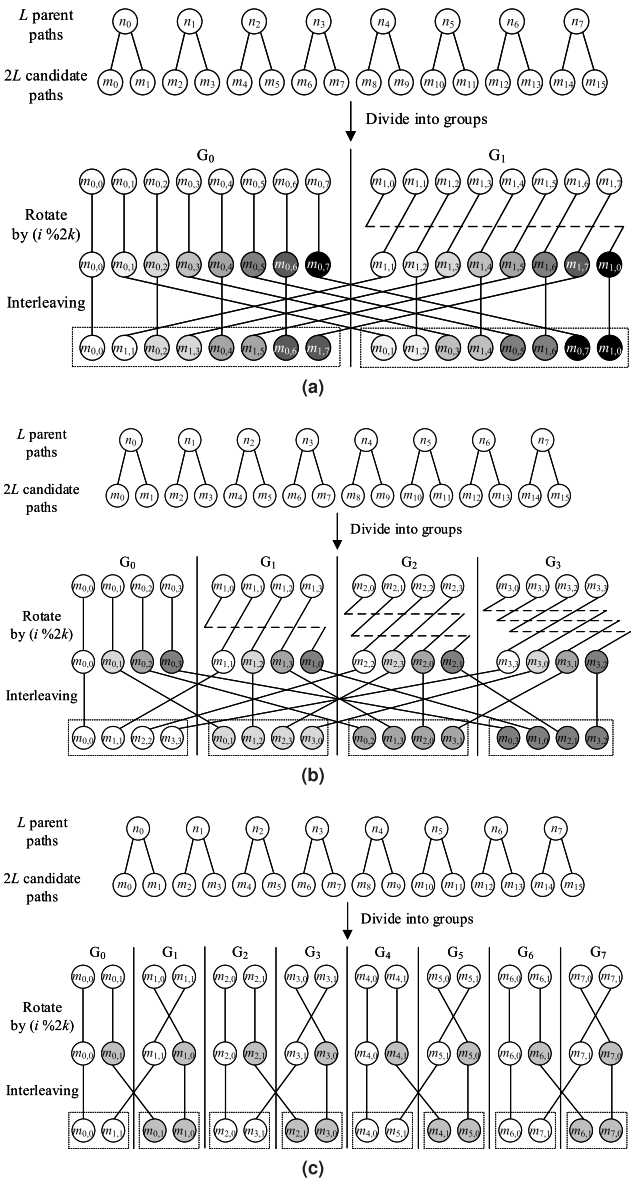


**FIGURE 5.** Interleaving process for $L = 8$, (a) $G = 2$, (b) $G = 4$, and (c) $G = 8$.

spread group consisting of diverse metric values, increasing the probability that small metrics survive there. In Fig. 5, the first and second subscripts of the metrics in a spread group are all different in a group, which ensures that the interleaving scheme shuffles the metrics uniformly. $L$ and $G$ are usually powers of two, because the list size is usually a power of two in polar codes and such a number of groups leads to an efficient sorting architecture in hardware implementation.

Figs. 6 and 7 show the frame-error rate (FER) performances obtained for the polar codes defined in the 5G communication standard [1]. The first is a half-rated 512-bit code and the second is a half-rated 1024-bit code. The two polar codes are both simulated over the binary-input AWGN channels. To make every message have a 11-bit CRC code as specified in the 5G standard [1], it is precoded with a generator polynomial $g_{CRC11}(D) = [D^{11}+D^{10}+D^{9}+D^{5}+1]$, and the information-bit locations for the precoded message are determined by taking into account the reliability sequence reported in [1]. Different polar-code construction methods can be applied to determine the location of information bits [16]. In the FER simulations, channel LLR values, internal LLR values, and path metrics are uniformly quantized to 4 bits, 7 bits, and 8 bits, respectively. In Figs. 6 and 7, the ILS-SCL($L$, $G$) denotes the FER obtained for taking a list size of $L$ and the interleaved local sorting with $G$ groups, and the LS-SCL($L$, $G$) represents the FER achieved with no interleaving for the same number of groups and the same list size. The simulation results indicate that the ILS-SCL decoding almost recovers the performance degradation caused by the local sorting. Regardless of the list size and the channel condition, the resulting FER degradation is negligible for all the polar codes experimented when the number of elements in a group is not less than 8.

Fig. 8 shows the FER performances of 5G 1024-bit polar codes with code rates of 1/3 and 1/2 which were specified in [18]. The simulations were conducted in two versions: floating-point simulations in which 32-bit floating-point values are used for LLR values and path metrics, and fixed-point simulations in which LLR values and path metrics are quantized as mentioned above. All the codes have a 11-bit CRC code, and the list size and the number of elements in a group are 16 and 4, respectively. It can be seen that the FER performances of the proposed ILS-SCL decoding are very similar to those of the conventional SCL decoding regardless of the code rates.

Fig. 9 shows the FER performances of shortened polar codes with code rates of 1/3 and 1/2. The codelength is shortened to 1014 bits by not transmitting the first 10 bits from a codelength of 1024 bits. The puncturing specified in the 5G standard [1] was applied to make the shortened polar codes. The simulation results in Fig. 9 shows that the proposed ILS has almost the same error-correcting performances as the conventional SCL decoding.

Based on the interleaving of group elements, the proposed metric sorting architecture is depicted in Fig. 10. In the path metric extension unit, the metrics of $L$ parent paths are extended to $2L$ metrics by considering the LLRs corresponding to the child candidates. After the extension, the $2L$ metrics are interleaved. Note that the interleaving can be realized with only wire connections, not requiring any additional hardware resources. The $2k$ metrics in a group are fed into a $2k$-to-$k$ sorter that sorts the incoming metrics and outputs the smallest $k$ metrics. The selected $L$ metrics are stored into the path metric memory and serve as the parent metrics in decoding the next bit.

Let us investigate the $S(L)$ and $C(L)$ of the proposed sorting architecture. Unlike the previous architecture, the input elements of the proposed $2k$-to-$k$ sorter do not have the relations addressed in (3) and (4). Therefore, the $2k$-to-$k$ sorter is designed based on the odd-even merge-sort network [12] that requires the lowest number of

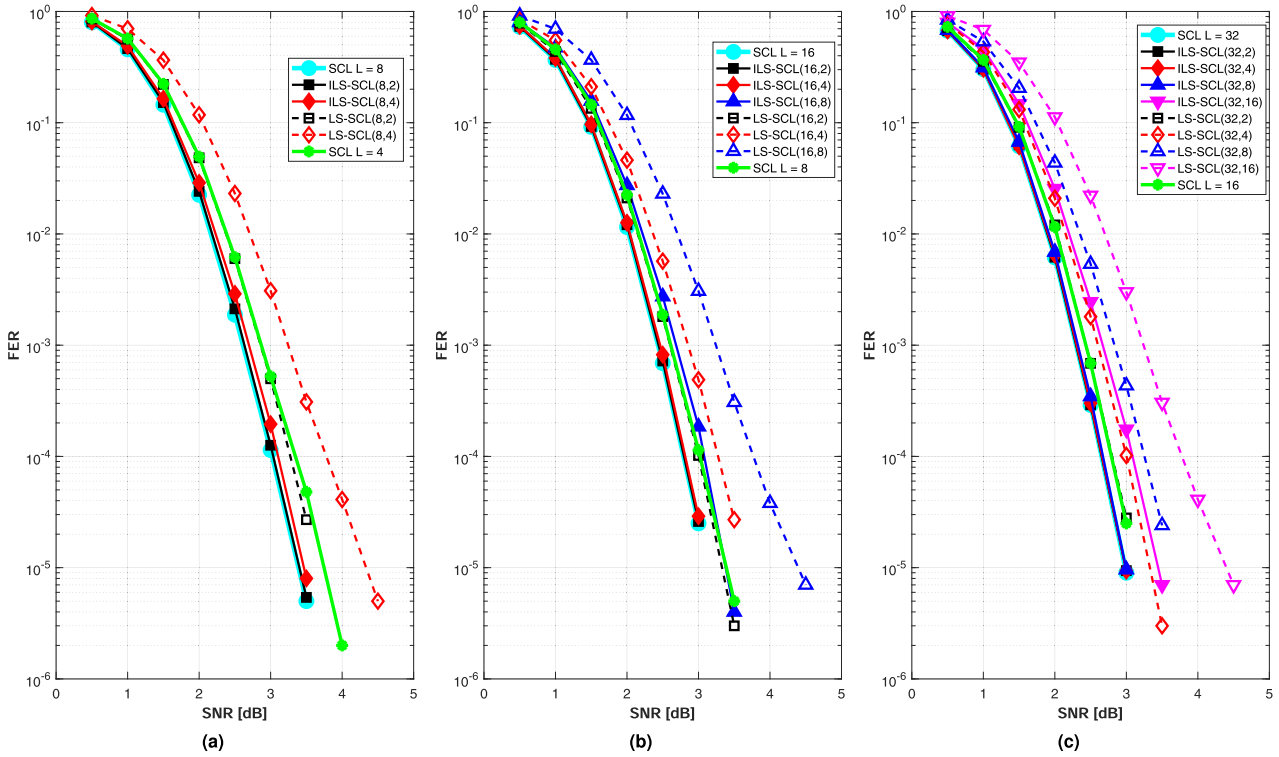


**FIGURE 6.** Frame error rates of fixed-point SCL decoding when codelength = 512, coderate = 1/2 (a) $L = 8$, (b) $L = 16$, and (c) $L = 32$.

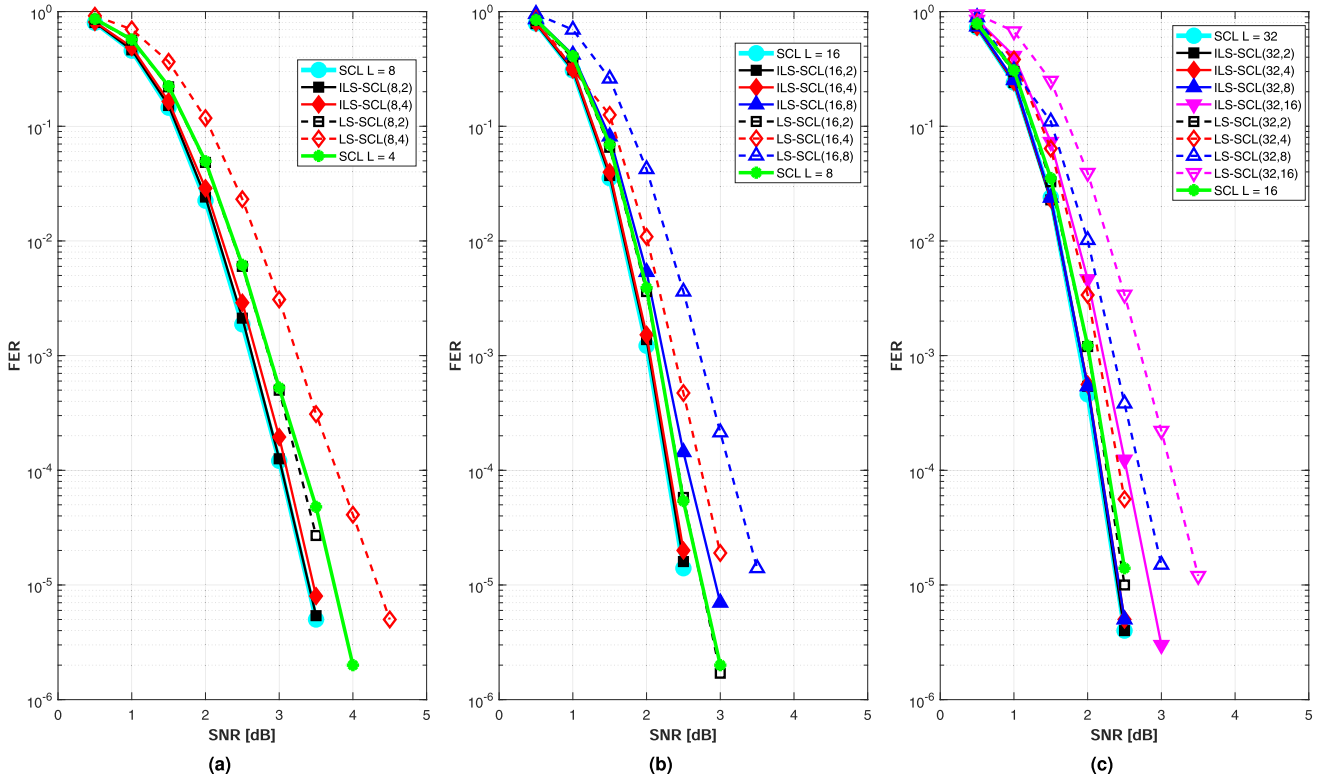


**FIGURE 7.** Frame error rates of fixed-point SCL decoding when codelength = 1024, coderate = 1/2 (a) $L = 8$, (b) $L = 16$, and (c) $L = 32$.

comparisons and comparison stages under the condition mentioned above.

Fig. 11 depicts a 8-to-4 sorter. The dotted comparison at the right side is pruned as its inputs are in the 4 biggest

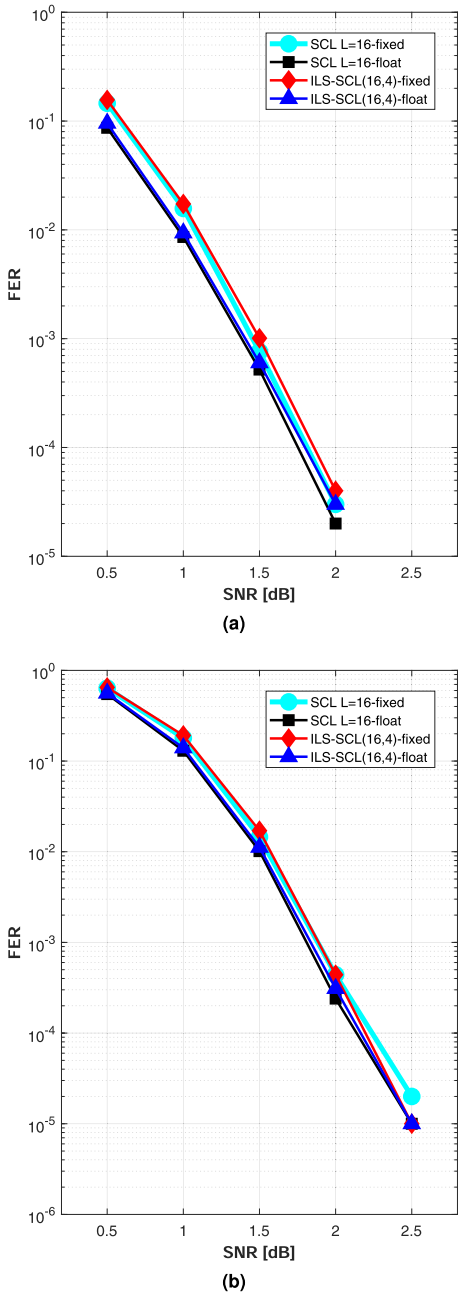


**FIGURE 8.** Frame error rates of floating-point SCL decoding and fixed-point SCL decoding when codelength = 1024, $L = 16$, $G = 4$, (a) coderate = 1/3, and (b) coderate = 1/2.

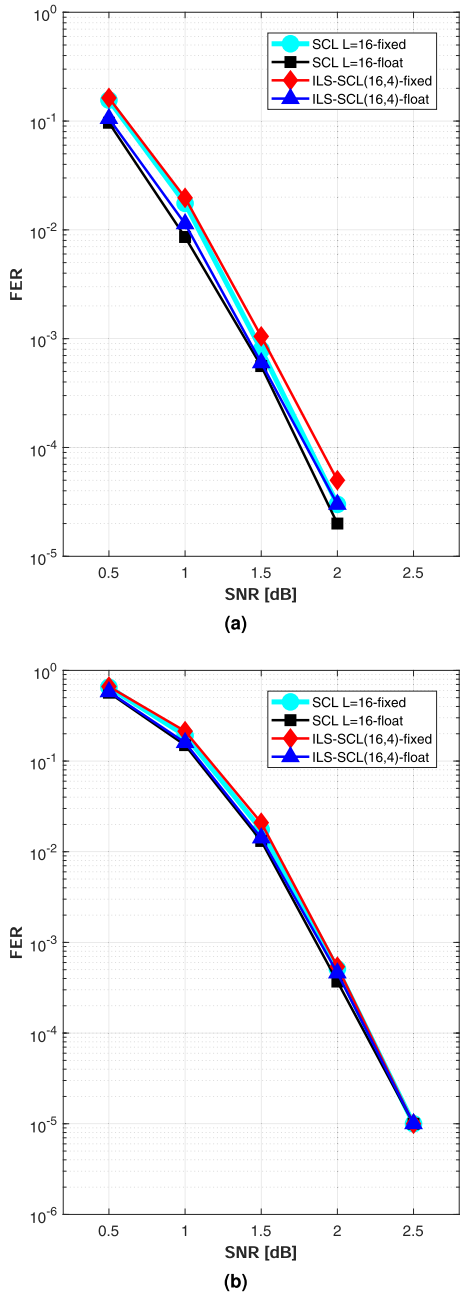


**FIGURE 9.** Frame error rates of shortened polar codes, with (a) coderate = 1/3, and (b) coderate = 1/2, where codelength = 1014, $L = 16$, $G = 4$.

elements. The $S(L)$ and $C(L)$ of a $2k$-to-$k$ sorter are computed with considering a general odd-even merge sorter. Let $C_{2L}$ be the number of comparisons of a general odd-even merge sorter dealing with $2L$ inputs, $C'_{2L}$ be the number of pruned comparisons, and $S_{2L}$ be the number of comparison stages. By the recurrent relation of the merge sorting architecture, $C_{2L}$ and $C'_{2L}$ can be computed as follows:

$$C_{2L} = \frac{L}{2}(\log_2 2L)(\log_2 2L - 1) + 2L - 1, \quad L \geq 1, \quad (18)$$

$$C'_{2L} = L \log_2 2L - 2L + 1, \quad L \geq 1. \quad (19)$$

where $C_2 = 1$ and $C'_2 = 0$ as the initial condition. Since the number of comparison stages is proportional to $\log_2 2L$,

$$S_{2L} = \frac{1}{2}(\log_2 2L)(\log_2 2L + 1), \quad 2L \geq 2 \quad (20)$$

The number of comparisons of a $2k$-to-$k$ sorter is $C_{2L} - C'_{2L}$, the number of comparison stages is the same as $S_{2L}$, and the number of $2k$-to-$k$ sorters included in the proposed sorter is $G$. Finally, the number of comparison stages $S(L, G)$ and comparisons $C(L, G)$ required in

**TABLE 1. Number of comparisons.**

| List size ($L$) | SBS [7] | OES [8] | FS-PMS [9] | HS-PMS [9] | PBE [10] | EPBE [11] | OPBE [11] | ILS[a] |
|---|---|---|---|---|---|---|---|---|
| 16 | 120 | 86 | 79 | 93 | 60 | 60 | 55 | 72 |
| 32 | 496 | 249 | 233 | 280 | 216 | 175 | 158 | 144 |
| 64 | 2016 | 680 | 660 | 791 | 768 | 505 | 456 | 288 |

[a]2L / G = 2k = 8

**TABLE 2. Number of comparison stages.**

| List size ($L$) | SBS [7] | OES [8] | FS-PMS [9] | HS-PMS [9] | PBE [10] | EPBE [11] | OPBE [11] | ILS[a] |
|---|---|---|---|---|---|---|---|---|
| 16 | 15 | 14 | 14 | 10 | 8 | 8 | 8 | 6 |
| 32 | 31 | 20 | 20 | 15 | 20 | 15 | 15 | 6 |
| 64 | 63 | 27 | 27 | 21 | 27 | 21 | 21 | 6 |

[a]2L / G = 2k = 8

**TABLE 3. Normalized equivalent gate counts (EGCs) [b].**

| List size ($L$) | SBS [7] | OES [8] | FS-PMS [9] | HS-PMS [9] | PBE [10] | EPBE [11] | OPBE [11] | ILS[a] |
|---|---|---|---|---|---|---|---|---|
| 16 | 1.78 | 1.27 | 1.17 | 1.38 | 0.89 | 0.89 | 0.81 | 1 |
| 32 | 3.68 | 1.85 | 1.73 | 2.08 | 1.60 | 1.30 | 1.17 | 1 |
| 64 | 7.47 | 2.52 | 2.44 | 2.93 | 2.84 | 1.87 | 1.69 | 1 |

[a]2L / G = 2k = 8

[b]Equivalent gate counts (EGCs) are measured by counting a two-input NAND gate as one.

[c]All values are normalized by ILS values.

**TABLE 4. Normalized latency of sorters.**

| List size ($L$) | SBS [7] | OES [8] | FS-PMS [9] | HS-PMS [9] | PBE [10] | EPBE [11] | OPBE [11] | ILS[a] |
|---|---|---|---|---|---|---|---|---|
| 16 | 2.63 | 2.46 | 2.46 | 1.76 | 1.40 | 1.40 | 1.40 | 1 |
| 32 | 5.44 | 3.51 | 3.51 | 2.63 | 3.51 | 2.63 | 2.63 | 1 |
| 64 | 11.06 | 4.74 | 4.74 | 3.69 | 4.74 | 3.69 | 3.69 | 1 |

[a]2L / G = 2k = 8

[b]All values are normalized by ILS values.

the ILS architecture is given by

$$S^{\mathrm{ILS}}(L,G) = \frac{1}{2}(\log_2 2k)(\log_2 2k+1) \tag{21}$$

$$C^{\mathrm{ILS}}(L,G) = G(\frac{L}{2}(\log_2 2k)(\log_2 2k-3)+4k-2) \tag{22}$$

which is defined for $k \geq 2$. Note that the number of comparison stages is independent of the list size, and the number of comparisons is proportional to the list size.

## IV. EVALUATION

In this section, we evaluate the proposed sorting architecture and the previous sorting architectures in terms of comparisons and comparison stages. The number of comparisons is related to the hardware complexity, while that of comparison stages plays a significant role in determining the processing latency.

Tables 1 and 2 summarize the number of comparisons and comparison stages required in various metric sorting architectures when the list size $L$ varies from 16 to 64. More precisely, we compare the proposed interleaved local sorting (ILS) with 7 different sorters:simplified bubble sorting (SBS) [7], odd-even sorting (OES) [8], full-sorted pairwise metric sorting (FS-PMS), half-sorted pairwise metric sorting (HS-PMS) [9], pruned bitonic extractor (PBE) [10], efficient PBE (EPBE) and OES-based PBE [11]. The second group of the PBE is considered as the SBS, and the FS-PMS is adopted for the EPBE and OPBE. For the proposed ILS architecture, the numbers are computed by (21) and (22) with fixing k to 4, i.e., fixing the number of input metrics in a group to 8. As the proposed ILS architecture divides the metrics into groups and then each group is sorted independently of the other groups, it achieves the lowest number of comparison stages when the

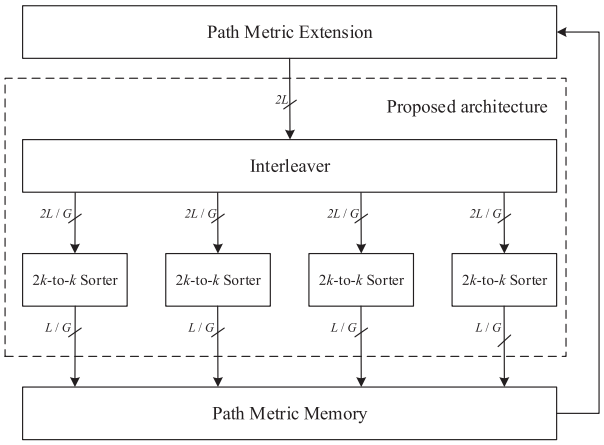


**FIGURE 10.** Block diagram of the proposed sorting architecture.

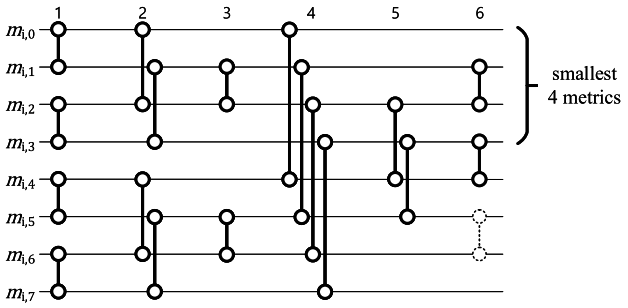


**FIGURE 11.** The *i*-th 8-to-4 sorter of the proposed sorting architecture.

list size is 32 or 64. The proposed ILS architecture reduces the numbers of comparisons and comparison stages a lot, and the reduction becomes larger as the list size increases.

The various sorting architectures have been synthesized in a 65-nm CMOS technology in order to compare their hardware complexities and latencies directly, as summarized in Tables 3 and 4. For all the architectures, the path metrics are represented in an unsigned 8-bit format. All the equivalent gate counts and sorting latencies are normalized by those of the ILS. The proposed ILS architecture is faster than the other architectures for all the list sizes experimented, and has the smallest hardware complexity when the list size is 32 or 64.

## V. CONCLUSION

The proposed architecture divides the metrics to be sorted into several groups and then sorts each group independently in order to reduce comparisons and comparison stages. To mitigate the performance degradation caused by the independent group sorting, an interleaving scheme is adopted to shuffle the metrics systematically. For various list sizes, simulation results have shown that the proposed metric sorting results in almost no degradation of error-correcting performance in the SCL decoding of polar codes. Compared to the state-of-the-art PMS architectures, the proposed architecture achieves the smallest numbers of comparisons and comparison stages, significantly lowering the processing latency of the SCL decoding of polar codes when the list size is not small.

## ACKNOWLEDGMENT

The authors would like to thank the IC Design Education Center (IDEC), South Korea, for supporting the EDA tool.

## REFERENCES

[1] *Multiplexing and Channel Coding*, document 38.212, 3GPP, 2021.
[2] 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.
[3] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.
[4] K. Chen, K. Niu, and J. R. Lin, "List successive cancellation decoding of polar codes," *Electron. Lett.*, vol. 48, no. 9, pp. 500–501, Apr. 2012.
[5] 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.
[6] 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.
[7] 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)*, Lisbon, Portugal, May 2015, pp. 1993–1996.
[8] B. Y. 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.
[9] 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.
[10] 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, May 2017.
[11] 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.
[12] K. E. Batcher, "Sorting networks and their applications," in *Proc. AFIPS SJCC*, Apr. 1968, pp. 307–314.
[13] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, Oct. 2012.
[14] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2044–2047, Dec. 2012.
[15] I. Parberry, "The pairwise sorting network," *Parallel Process. Lett.*, vol. 2, nos. 2–3, pp. 205–211, Sep. 1992.
[16] H. Vangala, E. Viterbo, and Y. Hong, "A comparative study of polar code constructions for the AWGN channel," Jan. 2015, *arXiv:1501.02473*. [Online]. Available: http://arxiv.org/abs/1501.02473
[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[18] *Evaluation on Channel Coding Candidates for eMBB Control Channel*, document TSG RAN WG1 #87 R1-1611109, 3GPP, Reno, NV, USA, Nov. 2016.

**WOOYOUNG KIM** (Student Member, IEEE) received the B.S. degree in semiconductor systems engineering from Sungkyunkwan University, Suwon, South Korea, in 2017. 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 VLSI architectures for error-correcting codes and communication systems.

**YUJIN HYUN** (Student Member, IEEE) received the B.S. degree in information and communication engineering from Inha University, Incheon, South Korea, in 2020. She is currently pursuing the M.S. degree with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. Her current research interests include algorithms and VLSI architectures for error-correcting codes and communication systems.

**JAEYOUNG LEE** received the B.S. degree in electrical and electronics engineering from Konkuk University, Seoul, South Korea, in 2021. 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 VLSI architectures for error-correcting codes and deep neural networks.

**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 Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1988 and 1992, respectively.

Since June 1996, he has been an Assistant Professor with the School of Electrical Engineering, KAIST, where he is currently a Professor. Prior to joining KAIST, he was with IBM T. J. Watson Research Center, Yorktown, NY, USA, from May 1995 to May 1996, where he researched high-speed circuit design. His current research interests include computer-aided design algorithms for high-level synthesis and VLSI architectures for general purpose microprocessors.

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