IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 30, 2018, accepted September 26, 2018, date of publication October 12, 2018, date of current version November 9, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2875756

Efficient Sorting Architecture for List-Fast-SSC Decoding of Polar Codes

XIAOJUN ZHANG, RONGQUAN SUI, JIANMING CUI, DEXUE ZHANG, AND QINGTIAN ZENG
College of Electronic, Communication and Physics, Shandong University of Science and Technology, Qingdao 266590, China

Corresponding author: Xiaojun Zhang (zhangxiaojun@sdust.edu.cn)

This work was supported in part by the Natural Science Foundation of China under Grant 61472229, Grant 61471224, and Grant 61701284, in part by the China Postdoctoral Science Foundation under Grant 2016M592216 and Grant 2017M622233, in part by the Qingdao Postdoctoral Research Project under Grant 2016125, and in part by the Science and Technology Development Fund of Shandong Province of China under Grant 2016ZDJS02A11.

ABSTRACT Polar codes have been proven to achieve the symmetric capacity of memoryless channel. Compared with a successive cancellation list decoder, list-Fast simplified-successive cancellation generates more candidate paths, which leads to more resource costs and higher decoding latency. To remedy this drawback, we present a simplified sorting architecture. An $M*L$ ordered candidate path matrix is constructed by preliminary sorter, where M and L denote the number of candidate path expanded by one constituent code and the list size of the decoder, respectively. Then, we eliminate the candidate paths that are definitely not in the L best paths by the proposed lossless pruning algorithm. Finally, a compatible sorting network combining the advantages of bitonic sorter and odd–even sorter is proposed. Numerical results show that for $L = 32$ and $M = 8$, the proposed architecture can reduce 66.7% of candidate paths and save 52.3% of compare and swap units (CASUs) and 25% of CASU stages compared with the odd–even sorter.

INDEX TERMS List-Fast-SSC, ordered candidate path matrix, pruning algorithm, compatible sorting network.

I. INTRODUCTION

Polar codes are the first code family that have been proven to achieve the symmetric capacity on binary-input discrete memoryless channels [1]. The successive cancellation (SC) decoding is a low-complexity algorithm for polar codes, which can achieve the channel capacity when the code length tends to infinity. The error-correction performance is not reasonable for polar codes with short and moderate lengths. The successive cancellation list (SCL) [2], [3] algorithm provides a better performance, but it must traverse each node of the binary tree like SC, and lead to high latency. To lower the block error rate (BLER), the researchers provided the cyclic redundancy check aided SCL decoding (CA-SCL) algorithm, which outperforms LDPC and turbo codes [4]–[7]. Amin Alamdar-Yazdi proposed the simplified successive cancellation (SSC) decoding algorithm [8], which divides the leaf nodes into rate-0, rate-1 and rate-r. To reduce the computational complexity, the estimates of rate-0 and rate-1 nodes can obtained by hard detections directly at the root of the sub-tree. Gabi Sarkis and Warren J. Gross consider the rate-r nodes of SSC algorithm still need to traverse their corresponding sub-trees. Therefore, they proposed the ML-SSC [9] and Fast-SSC [10] algorithms. The estimates of repetition (REP) and single-parity check (SPC) can also be calculated directly like rate-0 and rate-1 nodes instead of traversing their sub-trees. The deeply pipeline architecture for Fast-SSC [11] has high throughput. In order to yield further performance improvements, [12] proposed the list decoding algorithm based on Fast-SSC decoding (list-Fast-SSC). But the candidate paths of it are far more than that of SCL algorithm, which results in more hardware costs and higher decoding latency. Reference [13] proposed simplified successive-cancellation list (SSCL) that provides a good trade-off between error-correction performance and complexity. But SSCL decoding has some redundant path forks for rate-1 nodes. So [14] provides a faster decoding method for rate-1 node in SSCL decoding, which require fewer number of time-steps than SSCL. Then, [15] and [16] provide new decoding algorithms called Fast-SSCL and Fast-SSCL-SPC based on SSCL and SSCL-SPC, which remove redundant calculations when encountering rate-1 and SPC nodes, respectively.

Although the list decoders have better decoding performances, they suffer from much more resource consumption.

What's more, the metric sorter in the decoder architecture will be the crucial path while the list size L is large. In recent literatures, Lin Jun designed a general bitonic sorter (GBS) for SCL decoder [17], [18], which does not exploit the properties of the $2L$ paths. Reference [19] proposed the pruned bitonic sorter (PBS) and a bubble sorter that can sort the paths generated by SCL decoder efficiently. PBS exploits part of the characters of the paths, which removes one stage and many compare and swap units (CASUs). PBS is not good for $L \leq 16$, while it is a better option when $L \geq 32$. According to [19], the bubble sorter and the radix-$2L$ [20]–[22] can make a good trade-off between speed and area while $L \leq 16$. Reference [23] proposed the simplified odd-even sorter (SOES) based on odd-even sorter (OES) [24] for SCL, which separates the odd-indexed and the even-indexed elements to reduce unnecessary sorting operations and has a better performance than PBS. Meanwhile, the half-clear (HC) network in SOES will be eliminated, which reduces the number of CASUs. Two-Step Metric Sorter reduces hardware cost and gets better performance in terms of area efficiency and latency reduction [25]. Recently, full-sorted pairwise metric sorting (FS-PMS), half-sorted pairwise metric sorting (HS-PMS), and M-bit parallel decoding (M-PMS) architecture were proposed in [26]. The FS-PMS architecture is based on the general pairwise metric sorting (PMS) proposed in [27], which effectively reduces the CASUs by making best use of the properties of the paths. To further reduce the latency, HS-PMS combines the PMS architecture and the HC of GBS, which can save both CASUs and CASU stages. The M-PMS architecture was presented for the M-bit parallel decoding [28], [29], the number of CASU stages declines significantly. The aforementioned sorting networks are designed for the SCL algorithm except M-PMS, but they do not take into account the difference of new paths generated by constituent nodes of the list-Fast-SSC. Thus, they will cause large hardware complexity if applied to list-Fast-SSC directly.

To improve the hardware efficiency of sorting networks, the key is to reduce the number of elements in the metrics sorter. In this paper, we investigate the characters of the candidate paths for REP, rate-0, rate-1 and SPC nodes, and find that some of them are ordered while others are not. Therefore, the candidate paths are sorted into an ordered candidate path matrix (OCPM) based on part of ordered candidate paths in advance. To avoid redundant sorting operations, a pruning algorithm is developed to remove the candidate paths grounded on the nature of OCPM. We prove that the pruned ones are definitely not in the L best paths. The remaining paths are combined into a newly generated path matrix by merge sorter between rows, which is proved to be still ordered and can be also eliminated with the pruning algorithm. Hence, the candidate paths to be sorted in later modules are greatly reduced. To lower the hardware complexity, we combine the rows of OCPM by the pruned rear network (PRN) of OES and adopts HC of GBS to select the L possible paths in the last stage. Finally, we propose a sorting architecture compatible with four kinds of constituent nodes to save numbers of CASUs and CASU stages.

The remainder of this paper is organized as follows. Section II provides a brief review of the list-Fast-SSC algorithm and analyzes the properties of the candidate paths that generated by constituent nodes. In Section III, we propose a simplified sorting network for the four constitute nodes. A compatible sorting network is presented in Section IV. Section V shows the performance analysis on different sorting networks. Finally, conclusions are drawn in Section VI.

II. PRELIMINARIES

A. POLAR CODES

A polar code can be represented by $P(N, K)$, where N denotes the code length and K/N is the code rate. The polar encoding can be denoted by $x = uF^{\otimes n}$, where $u = \{u_0, u_1, \cdots, u_{N-1}\}$ is the data sequence, and $x = \{x_0, x_1, \cdots, x_{N-1}\}$ denotes the encoded sequence. $F^{\otimes n}$ is the n-th Kronecker power of the generator matrix, where $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. Polar codes select the K most reliable channels to transmit the information bits, and the other N-K channels are set to frozen bits. Let A present the information set, and A^c be the frozen set. As indicated in Fig. 1(a), $A = \{u_7, u_9, u_{10}, u_{11}, u_{12}, u_{13}, u_{14}, u_{15}\}$, $A^c = \{u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_8\}$. The white circles are frozen bits and the black ones are information bits. The gray nodes represent the concatenation of the two sub-trees. Soft messages α and hard estimate β are transmitted along the edge in the SC decoding tree. Fig. 1(b) shows that the soft messages α_v is passed to node v from its parent node p_v, α_l is passed from node v to the left child node v_l, while the hard-decision estimates β_l returns to its parent from the left child. Then, α_r of the right child v_r can be calculated. When β_r is known, β_v can be obtain through combining operations.

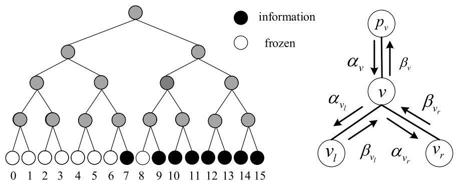


FIGURE 1. **(a) The decoding tree of a $P(16, 8)$ polar code. (b) Local decoder.**

B. FAST-SSC

Fig. 2 shows the decoding tree of SSC and Fast-SSC algorithms that are trimmed from SC decoding tree

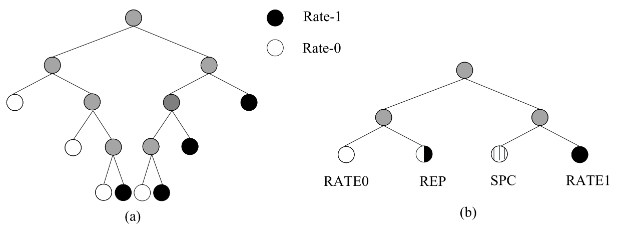


FIGURE 2. **Decoding tree of the SSC and Fast-SSC decoding algorithm.**

(See Fig. 1(a)). When all the leaf nodes are frozen bits, the constituent node is a rate-0 node. The leaf nodes of rate-1 are groups of information bits. They can be directly evaluated at the root of the their sub-trees. The outputs of rate-0 node are an all-zero vector. Rate-1 node can be estimated by hard decisions as

$$\beta_v[i] = h(\alpha_v[i]) = \begin{cases} 0 & \text{when } \alpha_v[i] \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

where i denotes the i-th bit of rate-1 node. The REP node is composed by a single information bit in the rightmost bit of the subtree. The Fast-SSC algorithm uses a low-complexity maximum likelihood (ML) algorithm instead of depth-first traversal of the sub-tree to decode the REP nodes. The ML-decision for REP is

$$\beta_v[i] = \begin{cases} 0 & \text{when } \sum_j \alpha_v[j] \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

The SPC decoder performs threshold detections firstly according to (3). Parity of the hard decision is calculated using addition modulo-two as (4), and the least reliable bit is selected based on (5).

$$h(\alpha_v[i]) = \begin{cases} 0 & \text{when } \alpha_v[i] \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

$$parity = \sum_j h(\alpha_v[i]) \tag{4}$$

$$j = \arg\min_j |\alpha_v[j]| \tag{5}$$

The outputs of SPC node are calculated as

$$\beta_v[i] = \begin{cases} h(\alpha_v[i]) \oplus parity & \text{when } i = j \\ h(\alpha_v[i]) & \text{otherwise} \end{cases} \tag{6}$$

C. LIST-FAST-SSC

Similar to SCL decoding algorithm, list-Fast-SSC decoder also generates new paths and chooses the L most reliable ones. But a significant difference with SCL decoder is that the length of constituent nodes is larger than one, and the constituent decoder can generate multiple candidates for each source path. List-Fast-SSC decoding has four types of constituent nodes: rate-0, rate-1, REP and SPC. These nodes will generate different number of candidate paths, such as rate-0 does not generate new paths, rate-1 node generates four candidates, REP node generates two candidates, and the SPC node generates eight ones [12]. Let M denote the number of the candidate paths generated by the source path, and M equals to 1, 2, 4 and 8 for rate-0, REP, rate-1 and SPC, respectively. There will be $M * L$ admissible candidates with list size L. The key step of list-Fast-SSC decoding is to choose L most reliable paths from them. Let p_s^l be the source path metric retained from previous step, where l is the l-th path and s denotes the source path. Different from SCL, p_s^l in list-Fast-SSC decoder composed of different candidate paths. Let p_i^l denote the i-th new path metric generated by the l-th source path at previous stage. Then, the $M * L$ new path metrics of each constituent nodes are computed as follows. Let min_1, min_2, min_3, min_4 represent the index of the minimum, the second, the third and the fourth minimum of absolute value of α_v, respectively.

For REP nodes, the path metrics are computed as

$$\begin{aligned} p_1^l &= p_s^l - \sum_i |\min(\alpha_v[i], 0)|, \\ p_2^l &= p_s^l - \sum_i |\max(\alpha_v[i], 0)| \end{aligned} \tag{7}$$

Rate-1 node generates four candidates as the following.

$$\begin{aligned} p_1^l &= p_s^l, \\ p_2^l &= p_s^l - |\alpha_v[min_1]|, \\ p_3^l &= p_s^l - |\alpha_v[min_2]|, \\ p_4^l &= p_s^l - |\alpha_v[min_1]| - |\alpha_v[min_2]| \end{aligned} \tag{8}$$

The reliabilities of the candidates in SPC decoder can be expanded by

$$\begin{aligned} p_1^l &= p_s^l - (1-q)|\alpha_v[min_1]|, \\ p_2^l &= p_s^l - q|\alpha_v[min_1]| - |\alpha_v[min_2]|, \\ p_3^l &= p_s^l - q|\alpha_v[min_1]| - |\alpha_v[min_3]|, \\ p_4^l &= p_s^l - q|\alpha_v[min_1]| - |\alpha_v[min_4]|, \\ p_5^l &= p_s^l - |\alpha_v[min_2]| - |\alpha_v[min_3]|, \\ p_6^l &= p_s^l - |\alpha_v[min_2]| - |\alpha_v[min_4]|, \\ p_7^l &= p_s^l - |\alpha_v[min_3]| - |\alpha_v[min_4]|, \\ p_8^l &= p_s^l - q|\alpha_v[min_1]| - |\alpha_v[min_2]| - |\alpha_v[min_3]| \\ &\quad - |\alpha_v[min_4]| \end{aligned} \tag{9}$$

Where q is an indicator, q is set to one when the parity check of SPC is satisfied. Otherwise, q is equal to zero.

It is worth noting that at most two candidates should be retained when $L = 2$ for any given source paths. Thus, only the two most reliable paths corresponding to p_1^l and p_2^l need to be evaluated for each source path, regardless of the constitute node is SPC or rate-1.

D. SORTING NETWORKS

Reference [17] proposed GBS which is optimal if the number of inputs is a power of two. As shown in Fig. 3(a), it is a 16-input GBS. The CASU compares the two inputs, and outputs the smaller one to the upper line and the larger one to the lower line. The numbers on the top of the sorter in Fig. 3(a) are the stage index. The CASUs in each stage can work in parallel. Let c_{2L}^{GBS} and s_{2L}^{GBS} denote the number of CASUs and CASU stages of $2L$-input sorter, respectively. They can be calculated via

$$c_{2L}^{GBS} = = \frac{L}{2}(\log L + 1)(\log L + 2) \tag{10}$$

$$s_{2L}^{GBS} = \frac{1}{2}(\log L + 1)(\log L + 2) \tag{11}$$

$$m_{2l} \leq m_{2(l+1)}, \quad \text{where } 0 \leq l < L - 1 \tag{12}$$

$$m_{2l} \leq m_{2l+1}, \quad \text{where } 0 \leq l \leq L - 1 \tag{13}$$

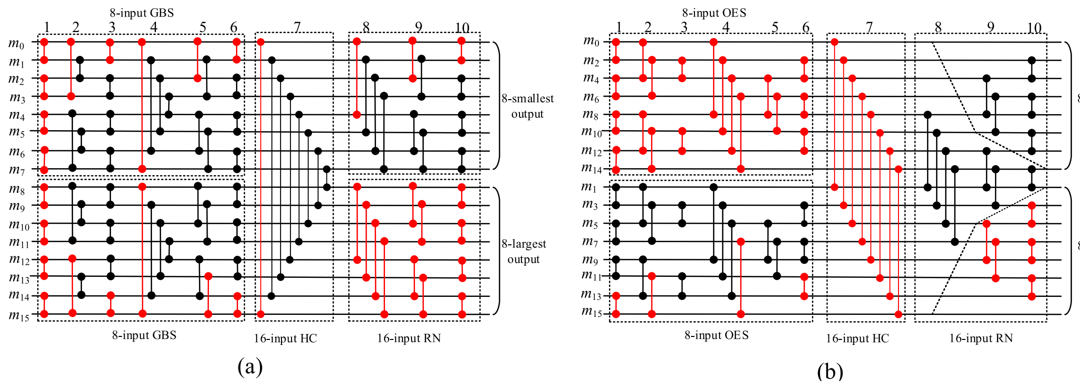


FIGURE 3. (a) Bitonic sort network for $2L = 16$, (b) Odd-even sort network for $2L = 16$.

The paths generated by SCL decoding are satisfied with (12) and (13), where m_{2l} and m_{2l+1} denote the even-indexed and odd-indexed path metrics, respectively. The first stage of GBS in SCL can be removed based on (13). Moreover, m_0 is the smallest path that must be in the L smallest ones, and m_{2L-1} is the largest path that is never among the L smallest ones, then all CASUs involving m_0 and m_{2L-1} can be removed. And the $L/2$ CASUs of the final stages can be deleted since they are irrelevant to the results. Thus, PBS can reduce one stage and many CASUs. The CASUs in red can be removed according to PBS in Fig. 3(a). The number of CASUs and CASU stages in PBS can be evaluated by

$$s_{2L}^{PBS} = \frac{1}{2}(\log L + 1)(\log L + 2) - 1 \tag{14}$$

$$c_{2L}^{PBS} = (\frac{L}{2} - 1)(\log L)(\log L + 2) + 1 \tag{15}$$

Fig. 3(b) illustrates an odd-even sorter for $2L$-input proposed in [24]. Compared to GBS, it costs fewer CASUs. The number of CASUs and CASU stages in OES can be

$$c_{2L}^{OES} = ((\log 2L)^2 - \log 2L + 4)2^{\log 2L - 2} - 1 \tag{16}$$

$$s_{2L}^{OES} = \frac{1}{2}(\log L + 1)(\log L + 2) \tag{17}$$

SOES separates the odd-indexed and even-indexed elements to avoid performing redundant sorting operations for the elements that are already sorted according to (12) and (13). The number of CASUs and CASU stages in SOES can be calculated by (18) and (19), respectively.

$$c_{2L}^{SOES} = c_{2L}^{OES} - \{c_L^{OES} + H_{2L} + R_L + R_{\frac{L}{2}} + (\log 2L - 1)\} \tag{18}$$

$$s_{2L}^{SOES} = \frac{1}{2}(\log L + 1)(\log L + 2) - 1 \tag{19}$$

Where $R_{2L} = \sum_{i=2}^{\log 2L}(i-2)2^{i-2}$ for $L \geq 2$ and $R_2 = 0$. $H_{2L} = L$ for $L \geq 2$ with $H_2 = 0$.

III. PROPOSED SORTING NETWORK

The proposed sorting network for list-Fast-SSC decoder includes preliminary sorter(PS), pruning module (PM) and merge sort between rows module (MSR). We sort the paths preliminarily, and obtain an OCPM. PM minimizes the elements to be sorted. Then MSR chooses the L best paths from the remaining candidate paths.

Remark 1: Because the path metrics are all non-positive real numbers, in order to save hardware resources, the sorter only needs to compare the absolute values of corresponding metrics. Let $m_j^i = |p_j^i|$.

Lemma 1: The four paths generated by rate-1 decoder hold: $m_1^l \leq m_2^l \leq m_3^l \leq m_4^l$, where $1 \leq l \leq L$.

Proof: Assume that x is an arbitrary non-positive real number and $b \geq a \geq 0$. Then $|x-b| \geq |x-a|$. It is easy to see that $0 \leq |\alpha_v[min_1]| \leq |\alpha_v[min_2]| \leq |\alpha_v[min_1]| + |\alpha_v[min_2]|$, thus there exist $m_1^l \leq m_2^l \leq m_3^l \leq m_4^l$ based on (8).

According to Lemma 1 and the nature of list-Fast-SSC, we can get the following properties.

1) Rate-0 nodes only generate one candidate path, m_1^l.

2) The candidate paths generated by rate-1 nodes satisfy $m_1^l \leq m_2^l \leq m_3^l \leq m_4^l$.

3) There is no obvious relationship between m_1^l and m_2^l generated by REP nodes.

4) The candidates generated by SPC nodes satisfy $m_1^l \leq m_2^l \leq m_3^l \leq m_4^l$ and $m_5^l \leq m_6^l \leq m_7^l \leq m_8^l$, where $m_3^l \leq m_5^l$, $m_4^l \leq m_6^l$.

According to the above discussion, it shows that rate-0 nodes can retain the paths to the next step directly, the rate-1 decoder needs a $4L$-to-L sorter, REP decoder needs a $2L$-to-L sorter, and SPC nodes need an $8L$-to-L sorter to select the L best paths from the generated paths.

A. PRELIMINARY SORTER

The matrix, composed of candidate paths satisfying (20) and (21), is referred to an ordered candidate path matrix. For instance, Fig. 4(c) shows an OCPM for $L = 8$ and $M = 8$.

$$m_i^l \leq m_{i+1}^l, \quad \text{where } 1 \leq l \leq L, 1 \leq i \leq M-1 \tag{20}$$

$$m_i^l \leq m_i^{l+1}, \quad \text{where } 1 \leq l \leq L-1, 1 \leq i \leq M \tag{21}$$

For the properties of the generated paths, we can design a simplified sorting network. But if the paths

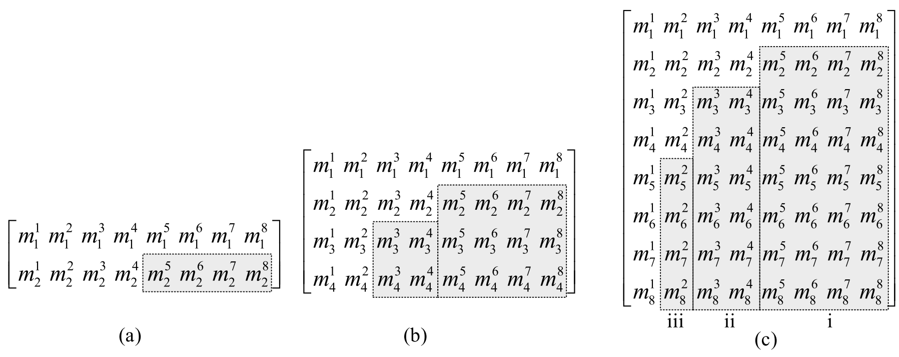


FIGURE 4. Ordered candidate path matrix for (a) $L = 8, M = 2$, (b) $L = 8, M = 4$, (c) $L = 8, M = 8$.

satisfy (20) and (21), the sorter will be more efficient, and save more redundant CASUs. To meet the above requirements, the following is performed.

1) RATE-1 NODES

The paths generated by rate-1 nodes are already satisfied with (20). An L-input OES is adopted to sort $\{m_i^1, m_i^2, \cdots, m_i^L\}$, where $1 \le i \le 4$, and the outputs act as the i-th row of the OCPM in Fig. 4(b).

2) REP NODES

The paths generated by REP nodes neither satisfy (20) nor (21). It can be easily selected with a $2L$-to-L sorter directly. In order to save hardware resources and make better use of the compatible sorter proposed in next section. Firstly, sort $\{m_1^l, m_2^l\}$ to satisfy (20) and obtain an ordered sequence $\{m_1^{l'}, m_2^{l'}\}$. Then sort $\{m_i^{1'}, m_i^{2'}, \cdots, m_i^{L'}\}$ by OES to satisfy (21), where $i = 1, 2$. The remaining paths can be written as Fig. 4(a).

3) SPC NODES

We sort $\{m_4^l, m_5^l\}$ firstly, the paths will satisfy (20), and obtain the ordered sequence $\{m_4^{l'}, m_5^{l'}\}$. Then sort $\{m_i^1, m_i^2, \cdots, m_i^L\}$ and $\{m_i^{1'}, m_i^{2'}, \cdots, m_i^{L'}\}$, where i=1, 2, 3, 6, 7, 8 and i=4, 5, respectively. After the two sorting operations, the sorted paths can be formulated as Fig. 4(c).

B. PRUNING MODULE

This module aims to eliminate some candidate paths on the basis of the properties of OCPM. To begin with, we prove that candidate paths of the ordered candidate path matrix definitely not in the L best paths can be pruned.

Lemma 2: If there are at least L paths smaller than m_i^j in the $M * L$ OCPM, m_i^j will not be in the final L candidate paths and can be pruned, where $1 \le i \le M$, $1 \le j \le L$.

Proof: Assume that all candidate paths of the OCPM are sorted ascendingly, let $k + 1$ denote the index of m_i^j. It is easy to see that k candidate paths are less than or equal to m_i^j. If $k \ge L$, then m_i^j is not in the smallest L candidate paths, so we can prune m_i^j.

In Fig. 4(c), since m_1^1 to m_1^5 and m_2^1 to m_2^4 are less than or equal to m_2^5 according to (20) and (21), we can delete m_2^5 by Lemma 2. For m_2^5 is the smallest candidate path in region i, thus all the others in region i can be removed. Similarly, m_3^3 and m_5^2 are the smallest candidate paths in region ii and iii, respectively. Accordingly, paths in regions ii and iii can be eliminated according to Lemma 2. Now, we eliminate the redundant paths at each row. Let $\mathcal{A}_i$ be the number of retained paths in the i-th row. When the list size is a power of two, PM prunes candidate paths according to Alg. 1 losslessly, and assure that it does not affect the error-correcting performance. The shaded part in Fig. 4 can be removed according to Alg. 1.

Algorithm 1 The Lossless Pruning Algorithm

```
Step 1:
If($M \ge 2$) then
Elements of the first row are all retained, and $\mathcal{A}_1 = L$.
The second row retains the $\frac{L}{2}$ smallest elements, and
$\mathcal{A}_2 = \frac{L}{2}$. goto step 2.
else stop.
Step 2:
If($M \ge 4$) then
The third row retains the $\sum_{\lambda=1}^{\lfloor \frac{1}{2}\log_2 L \rfloor} \frac{L}{2^{2\lambda}}$ smallest elements.
The fourth row only retains the $\frac{L}{4}$ smallest elements. goto
step 3.
else stop.
Step 3:
If($M == 8$) then
The fifth row only retain the smallest $\sum_{\lambda=0}^{\lfloor \frac{1}{4}\log_2 L-3 \rfloor}(\frac{L}{2^{4\lambda+3}}$
$+\lfloor \frac{L}{2^{4\lambda+4}} \rfloor)$ elements. And the sixth, seventh and eighth
rows retain $\sum_{\lambda=0}^{\lfloor \frac{1}{4}\log_2 L-3 \rfloor}(\frac{L}{2^{4\lambda+3}}+\lfloor \frac{L}{2^{4\lambda+5}} \rfloor)$, $\sum_{\lambda=1}^{\lfloor \frac{1}{3}\log_2 L \rfloor} \frac{L}{2^{3\lambda}}$
and $\frac{L}{8}$ smallest elements, respectively.
else stop.
```

C. MERGE SORT BETWEEN ROWS

MSR sorts the remaining paths, and finally outputs the L best paths. Let merge-stage denote the operation that mergesorts the $2i$-th and $(2i$-1)-th rows of the matrix and outputs $\mathcal{A}_i$ smallest paths, where i is smaller than or equal to half of the

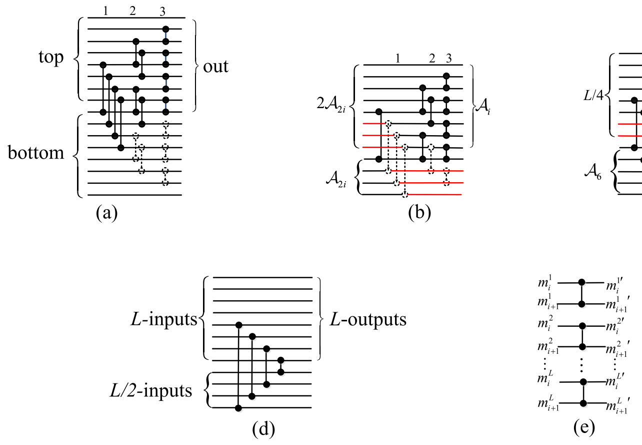


FIGURE 5. **(a) Merge sorter for 2L-inputs, (b) PRN for $\mathcal{A}_3$ and $\mathcal{A}_4$, (c) PRN for $\mathcal{A}_5$ and $\mathcal{A}_6$ when $L = 32$, (d) HC network for $L = 8$, (e) Compare the m-th and (m+1)-th paths generated by previous step, where $1 < m < M$.**

number of rows in the matrix. Hence, MSR consists of $\log_2 M$ merge-stages, and the outputs of each merge sorter are viewed as the i-th row of a new generated path matrix. By Lemma 3, the newly generated path matrix is still ordered. Therefore, we can eliminate some paths according to Alg. 1.

Lemma 3: If the original matrix is an OCPM, then the newly generated matrix after the j-th merge-stage is still an OCPM, where $j = 1, 2, \cdots, \log_2 M$.

Proof: Let $\{x_s\}$ and $\{y_t\}$ present the sequence in i-th and $(i+1)$-th row of the newly generated path matrix respectively, where s and t denote the length of sequence. Because the outputs of the merge sorters are in ascending order, thus $\{x_s\}$ and $\{y_t\}$ satisfy (21). For any $l = 1, 2, 3, \cdots, min(s, t)$, there exists l elements in $\{y_t\}$ which are smaller than or equal to y_l. According to the definition of the original matrix, each of these l elements in $\{y_t\}$ has at least one corresponding element in $\{x_s\}$ is smaller than or equal to itself. It guarantees that at least l elements in $\{x_s\}$ are smaller than or equal to y_l. Let k denote the index of the largest one that smaller than or equal to y_l among $\{x_s\}$, and hence $k \geq l$. Therefore, $x_l \leq x_k \leq y_l$, and the newly generated matrix is an OCPM after merge-stage.

To illuminate Lemma 3, we give an example as follows. Let $B = \{b_1, b_2, b_3, b_4\}$, $C = \{c_1, c_2, c_3\}$, $D = \{d_1, d_2\}$, and $E = \{e_1, e_2\}$ denote the first, second, third, and fourth rows in the pruned OCPM, respectively. We combine B and C, D and E, and obtain $B^* = \{b_1, b_2, c_1, c_2, b_3, c_3, b_4\}$, $D^* = \{d_1, e_1, d_2, e_2\}$. Since the two new rows are the outputs of sorters, they satisfy (21). It is known that B, C, D, and E satisfy (20). By the definition of OCPM, we can infer that $b_1 \leq d_1$, $b_2 \leq c_1 \leq e_1$, $c_1 \leq e_1 \leq d_2$, $c_2 \leq e_2$, then B^* and D^* satisfy (20) and they can also construct an OCPM.

To sort the remaining paths more efficiently, we propose a merge sorter based on SOES and GBS. The first to ($\log_2 M$-1)-th merge-stages use the PRN of SOES. The last merge-stage uses HC network of GBS, which can separate the L smaller elements from the two ordered inputs. As shown in Fig. 5(a), PRN is evolved from the general 2L-input merge sorter, in which the HC network can be pruned according to the property of the OCPM, and the CASUs at RN that are not associated with the outputs can be pruned as well. The pruned CASUs are indicated by dotted lines.

IV. COMPATIBLE SORTING NETWORK

This section presents a sorting network compatible with rate-0, rate-1, REP and SPC nodes. Fig. 6 shows that the architecture comprises PS, PM and MSR modules. The first stage of PS contains two 2L-input sorting networks. One is for REP to sort $\{m_1^l, m_2^l\}$ and another is for SPC to sort $\{m_4^l, m_5^l\}$ as indicated in Fig. 5(e), where $1 \leq l \leq L$. Then the paths are imputed into an 8L-input OES. After the PM, we can get the pruned OCPM. MSR is composed of two parts. The first one is a PRN that outputs $3L/2$ smallest candidate paths. The other is a $3L/2$-input HC network, after which we can obtain the L smallest paths. The proposed compatible architecture is devised for $L \geq 8$. When $L = 2, 4$, the sorter can be simplified based on the proposed architecture and it does not need to sort all candidate paths according to section III-A. The costs of each part are evaluated as follows.

A. PS MODULE

The 8L-input OES in this part is actually composed of eight L-input OESs. The number of CASUs used by L-input OES can be calculated as

$$\mathcal{N}_{CL} = [(\log_2 L)^2 - \log_2 L + 4]2^{\log_2 L - 2} - 1 \tag{22}$$

The 2L-input sorter is one stage composed of L CASUs. The number of CASUs used by PS can be given by

$$\mathcal{N}_{CP} = 8\mathcal{N}_{CL} + 2L \tag{23}$$

The number of retained candidate paths is

$$\mathcal{N}_{re} = \sum_{i=1}^{M} \mathcal{A}_i \tag{24}$$

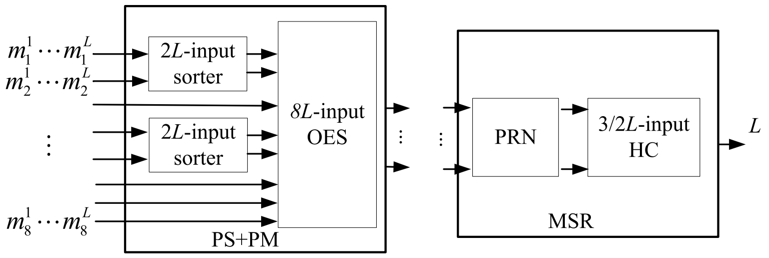


FIGURE 6. Compatible sorting architecture.

B. MSR MODULE

In this module, PRN has two merge-stages. Each of them can be divided into the following cases.

(a) input $\mathcal{A}_1$ and $\mathcal{A}_2$ paths, output the L smallest ordered elements, where $\mathcal{A}_1 = L$, $\mathcal{A}_2 = L/2$;

(b) input $\mathcal{A}_3$ and $\mathcal{A}_4$ paths, output the $L/2$ smallest ordered elements, where $L/4 \leq \mathcal{A}_3 \leq L/2$, $\mathcal{A}_4 = L/4$;

(c) input $\mathcal{A}_5$ and $\mathcal{A}_6$ paths, output the $\mathcal{A}_3$ smallest ordered elements, where $L/8 \leq \mathcal{A}_5 < L/4$, $L/8 \leq \mathcal{A}_6 < L/4$;

(d) input $\mathcal{A}_7$ and $\mathcal{A}_8$ paths, output the $L/4$ smallest ordered elements, where $L/8 \leq \mathcal{A}_7 < L/4$, $\mathcal{A}_8 = L/8$.

When $\mathcal{A}_{2i-1} = \mathcal{A}_{2i}$, we can use a PRN shown in Fig. 5(a). *max-val* is the maximum value of the sorting network and indicated by red lines in Fig. 5. If $\mathcal{A}_{2i-1} \leq 2\mathcal{A}_{2i}$, we can construct a $3\mathcal{A}_{2i}$-input PRN which needs to add $2\mathcal{A}_{2i} - \mathcal{A}_{2i-1}$ *max-val*s to the top part. For $\mathcal{A}_3 = 5$ and $\mathcal{A}_4 = 4$, we should add three *max-val*s to the top for constructing an 12-input PRN as depicted in Fig. 5(b). In particular, if $\mathcal{A}_5$ and $\mathcal{A}_6$ are not a power of two, we will add $L/4 - \mathcal{A}_5$ *max-val*s to construct a PRN as shown in Fig. 5(c). In Fig. 5(d), the third merge-stage is a $3L/2$-input HC network, which outputs the L smallest elements. It needs one CASU stage and $L/2$ CASUs.

Let S_C be the total number of CASUs used by the proposed sorting network, where $\mathcal{N}_{C(i,j)}$ denotes the CASUs used by the PRN for $\mathcal{A}_{2i-1}$ and $\mathcal{A}_{2i}$ in the j-th merge-stage of MSR.

$$S_C = \mathcal{N}_{CP} + \sum_{j=1}^{2} \sum_{i=1}^{8/2^j} \mathcal{N}_{C(i,j)} + \frac{L}{2} \tag{25}$$

The CASU stages of the sorting network is given by

$$\mathcal{N}_{CSP} = \frac{1}{2}(\log_2 L)(\log_2 L + 1) + 1 \tag{26}$$
$$\mathcal{N}_{CSM} = 2\log_2 L + 1 \tag{27}$$

Where $\mathcal{N}_{CSP}$ and $\mathcal{N}_{CSM}$ present the number of CASU stages of PS and MSR, respectively.

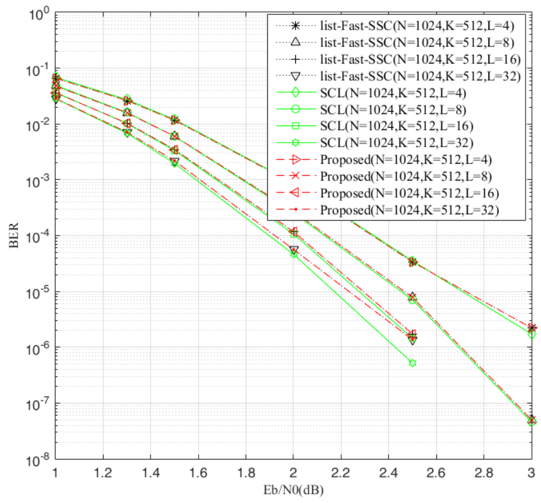


FIGURE 7. The comparison of BER performance among sorters.

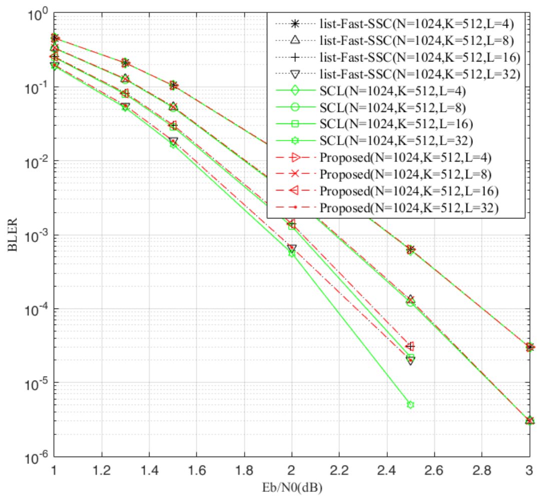


FIGURE 8. The comparison of BLER performance among sorters.

V. PERFORMANCE ANALYSIS

Assume a (1024, 512) polar code is transmitted over the binary-input additive white Gaussian noise channel with binary phase shift keying (BPSK) modulation. Fig. 7 and Fig. 8 illustrate the bit error rate (BER) and BLER performance among List-Fast-SSC, SCL and the proposed, respectively. It can be seen that the decoding performance of the proposed is identical to that of list-Fast-SSC. When the code length of SPC node is limited to 4, the error-correcting performance of list-Fast-SSC is slightly degraded compared with SCL algorithm.

We analyze the number of candidate paths in different search widths and different node types. Let "Origin" represent the original candidate paths and "Pruned" denote the

TABLE 1. The number of candidate paths.

L	$M=2$		$M=4$		$M=8$	
	Origin	Pruned	Origin	Pruned	Origin	Prune
2	4(1.00)*	3(0.75)	4(1.00)	3(0.75)	4(1.00)	3(0.75)
4	8(1.00)	6(0.75)	16(1.00)	8(0.50)	32(1.00)	8(0.25)
8	16(1.00)	12(0.75)	32(1.00)	16(0.50)	64(1.00)	20(0.31)
16	32(1.00)	24(0.75)	64(1.00)	33(0.52)	128(1.00)	42(0.33)
32	64(1.00)	48(0.75)	128(1.00)	66(0.52)	256(1.00)	85(0.33)

*The numbers in parentheses are normalized with respect to the original data.

TABLE 2. The number of CASUs and CASU stages.

L	OES		GBS		SOES		PBS		Proposed	
	CASUs	Stages	CASUs	Stages	CASUs	Stages	CASUs	Stages	CASUs	Stages
2	5(1.00)*	3(1.00)	6(1.20)	3(1.00)	5(1.00)	3(1.00)	5(1.00)	3(1.00)	5(1.00)	3(1.00)
4	191(5.62)	15(1.50)	240(7.06)	15(1.50)	76(2.24)	10(1.00)	91(2.68)	13(1.30)	34(1.00)	10(1.00)
8	543(2.76)	21(1.50)	672(3.41)	21(1.50)	252(1.28)	16(1.14)	320(1.62)	19(1.36)	197(1.00)	14(1.00)
16	1471(2.34)	28(1.40)	1792(2.85)	28(1.40)	760(1.21)	23(1.15)	973(1.55)	26(1.30)	628(1.00)	20(1.00)
32	3839(2.10)	36(1.33)	4608(2.52)	36(1.33)	2152(1.17)	31(1.15)	2726(1.49)	34(1.26)	1832(1.00)	27(1.00)

*The numbers in parentheses are normalized with respect to the proposed sorter.

candidate paths after PS. As illustrated in Table 1, compared to original candidate path, the number of candidate paths after pruning can be reduced greatly. For $L = 8$ and $M = 8$, 31% of the candidate paths is retained. When $L = 32$ and $M = 8$, 66.7% of original candidate paths can be eliminated. Particularly, when $L = 2$, at most two candidate paths need to be evaluated for any given source paths, thus two paths should be selected from the four candidate ones, so the reduction rate is still 25% regardless of M. For PM deletes the more paths, less resources will be consumed at the following stages.

Table 2 summarizes the number of CASUs and CASU stages for various sorting networks with different L while $M = 8$. The GBS and OES sorting networks are implemented by an $8L$-input sorter, thus the number of CASUs and CASU stages is larger than other sorters. Since the OCPM outputs from PS in this paper satisfy the input condition of SOES. Therefore, the SOES in Table 2 is based on the PS without pruning the candidate paths. For $L = 8$, $M = 8$, the SOES in Table 2 uses eight CASUs to make the paths satisfy (20) firstly, then uses eight OES sorters to satisfy (21). Secondly, it uses seven $2L$-to-L pruned rear networks in SOES to select the L most reliable paths. Similar to the SOES, the PBS is also processed by PS firstly, then uses the PBS to obtain the L smallest elements. These sorting methods do not eliminate the paths after PS. But the proposed architecture prunes many candidates after PS, and employs PRN of SOES and HC of GBS in the last stage. Table 1 shows that the number of paths is greatly reduced after PS for different list sizes. Although PS module occupy a large portion resources of the entire sorting architecture, pruning module can separate the unnecessary candidate paths in advance with its help. Hence proposed architecture uses less resources than other sorters. For $L = 2$, there are four candidate paths need to be sorted and to select the two smallest ones, the traditional OES is same as the SOES sorter, the PBS prunes one CASU in the last stage based on GBS, the proposed sorter uses traditional OES or SOES method. However, when $L > 2$, the proposed sorter consumes least CASUs and CASU stages. It shows that, when $M = 8$, $L = 32$, the proposed sorter reduces 52.3% of CASUs and 25% of CASU stages than the OES, 60.2% of CASUs and 25% of CASU stages compared with the GBS. Since the proposed architecture costs less resources and CASU stages compared to other sorters, it can obtain lower latency. When the list size is large, the sorting network may be in the crucial path of the decoder, which will bring in a large delay. We can pipeline the sorter to improve the working frequency.

VI. CONCLUSION

In this work, an efficient sorting architecture for list-Fast-SSC decoder is presented. We sort the candidate paths to construct an ordered candidate path matrix. To avoid redundant sorting operations, we develop a lossless pruning algorithm to eliminate for the elements that are definitely not in the L best paths, so the number of candidate paths to be sorted can be reduced efficiently. The compatible architecture combines advantages of SOES and GBS, and saves a lot of resources. Numerical results show that the proposed sorting network reduces the number of CASUs and CASU stages compared with other sorters.

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] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[3] 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.

[4] Y. Fang, G. Bi, Y. L. Guan, and F. C. M. Lau, "A survey on protograph LDPC codes and their applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1989–2016, 4th Quart., 2015.

[5] G. M. Kraidy, "On progressive edge-growth interleavers for turbo codes," *IEEE Commun. Lett.*, vol. 20, no. 2, pp. 200–203, Feb. 2016.

[6] Y. Fang, G. Han, G. Cai, F. C. M. Lau, P. Chen, and Y. L. Guan, "Design guidelines of low-density parity-check codes for magnetic recording systems," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1574–1606, 2nd Quart., 2018.

[7] Y. Fang, S. C. Liew, and T. T. Wang, "Design of distributed protograph LDPC codes for multi-relay coded-cooperative networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7235–7251, Nov. 2017.

[8] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, Dec. 2011.

[9] G. Sarkis and W. J. Gross, "Increasing the throughput of polar decoders," *IEEE Commun. Lett.*, vol. 17, no. 4, pp. 725–728, Apr. 2013.

[10] 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.

[11] X. Zhang, X. Yan, Q. Zeng, J. Cui, N. Cao, and R. Higgs, "High-throughput fast-SSC polar decoder for wireless communications," *Wireless Commun. Mobile Comput.*, vol. 2018, Jul. 2018, Art. no. 7428039.

[12] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 318–328, Feb. 2016.

[13] S. A. Hashemi, C. Condo, and W. J. Gross, "Simplified successive-cancellation list decoding of polar codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 815–819.

[14] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast simplified successive-cancellation list decoding of polar codes," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, San Francisco, CA, USA, Mar. 2017, pp. 1–6.

[15] 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.

[16] 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.

[17] J. Lin and Z. Yan, "Efficient list decoder architecture for polar codes," in *Proc. IEEE Int. Symp. Circuits Syst.*, Jun. 2014, pp. 1022–1025.

[18] 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.

[19] 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.

[20] 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.

[21] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Florence, Italy, May 2014, pp. 3903–3907.

[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] 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.

[24] K. E. Batcher, "Sorting networks and their applications," in *Proc. AFIPS Spring Joint Comput. Conf. (SJCC)*, Apr. 1968, pp. 307–314.

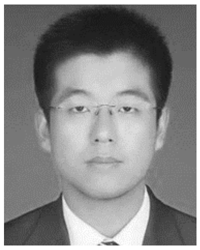
[25] V. Bioglio, F. Gabry, L. Godard, and I. Land, "Two-step metric sorting for parallel successive cancellation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 456–459, Mar. 2017.

[26] 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.

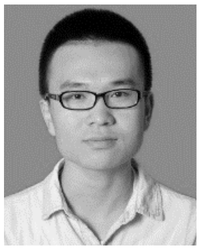
[27] I. Parberry, "The pairwise sorting network," *Parallel Process. Lett.*, vol. 2, nos. 2–3, pp. 205–211, 1992.

[28] 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.

[29] C. Xiong, J. Lin, and Z. Yan, "Symbol-based successive cancellation list decoder for polar codes," in *Proc. IEEE SiPS*, Oct. 2014, pp. 1–6.

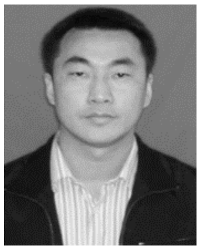


XIAOJUN ZHANG was born in 1980. He received the B.S. and M.S. degrees in computer science from the Shandong University of Science and Technology in 2003 and 2007, respectively, and the Ph.D. degree from East China Normal University in 2011. He is currently a Lecturer with the Shandong University Science and Technology. His recent research interests include VLSI design, signal processing, and channel coding.



RONGQUAN SUI received the B.S. degree in electronic information science and technology from the Shandong University of Science and Technology, Qingdao, China, in 2015, where he is currently pursuing the M.S. degree. His research interests include VLSI design and algorithms optimization for polar codes.

JIANMING CUI was born in 1969. He received the B.S. and M.S. degrees in automation from the Shandong University of Science and Technology in 1992 and 1999, respectively, and the Ph.D. degree from East China Normal University in 2013. He is currently an Associate Professor with Shandong University Science and Technology. His recent research interests include authentication protocol, signal processing, and channel coding.



DEXUE ZHANG received the B.S. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2000 and 2006, respectively. He is currently an Associate Professor with the Shandong University of Science and Technology, Qingdao, China. His research interests include SoC design and polar decoding.

QINGTIAN ZENG received the Ph.D. degree in computer software and theory from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. In 2008, he was a Visiting Professor with the City University of Hong Kong, Hong Kong. He is currently a Professor with the Shandong University of Science and Technology, Qingdao, China. His current research interests include signal processing and wireless communication.