Improved Belief Propagation Polar Decoders with Bit-Flipping Algorithms

Yifei Shen, *Student Member, IEEE,* Wenqing Song, *Student Member, IEEE,* Houren Ji, Yuqing Ren, *Student Member, IEEE,* Chao Ji, Xiaohu You, *Fellow, IEEE,* Chuan Zhang, *Member, IEEE*

***Abstract*—Since the inherent serial nature of successive cancellation list (SCL) decoding results in a long latency, belief propagation (BP) decoding for polar codes has drawn attention for high-throughput applications. However, its error correction performance is inferior to that of SCL decoding. Therefore, the bit-flipping strategy has been recently applied to BP decoding, which can approach the SCL decoding performance through multiple additional decoding attempts. The original BP flip (BPF) decoding suffers from an inaccurate identification of erroneous bits by a fixed flip set (FS), which has been improved by the generalized BPF (GBPF) decoding. In this paper, the GBPF decoding is extended to support multiple bits being flipped in one decoding attempt. In addition, for two types of decoding errors: detected errors and undetected errors, we propose two novel methods to more effectively identify erroneous bits. For detected errors, the concept of loop sets is defined and a loop-based identification method is introduced based on the study of error patterns of BP decoding. On the other hand, a method to generate a more accurate fixed FS is proposed for undetected errors, which considers the bit error distribution under BP decoding. Combining the two methods, the GBPF with merged sets (GBPF-MS) decoding can achieve the SCL-8 performance and outperforms the state-of-the-art BPF, BP list, and SC flip (SCF) decoding, for polar codes with length 1024 and information rate $1/2$. Implemented by 40 nm CMOS technology, the proposed GBPF-MS decoder with ten flips exhibits an average throughput of 4.19 Gbps at 2.5 dB, which is $1.6\times$ and $1.72\times$ faster than the state-of-the-art SCL-4 and SCF decoders, respectively.**

***Index Terms*—Polar codes, belief propagation (BP) decoding, bit-flipping, error patterns, flip set.**

I. Introduction

POLAR codes were proposed by Arıkan [1], which are proven to achieve the Shannon limit over binary-input discrete memoryless channels. At the same time, Arıkan proposed two decoding algorithms for polar codes, namely successive cancellation (SC) decoding [1] and belief propagation (BP) decoding [2]. In 2016, polar codes were selected as the standard codes for the fifth-generation (5G) enhanced mobile broadband (eMBB) control channels [3].

SC decoding and its derived algorithms, such as SC list (SCL) [4] and SC flip (SCF) [5] decoding algorithms, have drawn more attention than other algorithms. When concatenating cyclic redundancy check (CRC) codes, polar codes decoded by these derivative algorithms exhibit superior error correction performance over low-density parity-check (LDPC) and Turbo codes [6]–[8]. In particular, the CRC-aided SCL (CA-SCL) decoding algorithm [6] was chosen as the baseline algorithm in the standardization process of 5G new radio [9].

In contrast to the serial SC-based decoding algorithms, BP decoding calculates intermediate soft information iteratively in parallel, achieving a higher throughput [10]. Besides, BP decoding is expected to support both 5G standard polar and LDPC codes with a set of hardware units in practical applications [11]. However, the performance of BP decoding is inferior to that of SCL. The original BP decoding [2] propagates messages over the generator matrix-based factor graph (FG). The authors of [12] pointed out that for a polar code of length N ($N = 2^n$), there exist a total of $n!$ representations of FG, each of which corresponds to a different decoding performance. To this end, a series of decoding schemes based on permuted FGs have been proposed [13]–[16], which are named as BP list (BPL) algorithms. When concatenating a CRC code and decoding it with the sum-product algorithm, the BPL decoding can achieve the CA-SCL performance at high signal-to-noise ratios (SNRs) [17].

Another approach to enhance BP performance is the BP flip (BPF) decoding. Compared with other BP-based algorithms, BPF decoding utilizes the information in the decoding procedure, e.g., log-likelihood ratio (LLR) values and parity-check state, so it is more targeted to the codeword being decoded. In 2010, the guessing algorithm proposed for LDPC codes [18] was applied to the BP decoding of polar codes [19], which guesses the values of error-prone variable nodes (VNs) and repeats the original BP decoding. Based on this algorithm, the authors of [20] put forward the idea of guessing the value of oscillating bits by assigning their a-priori LLRs with the maximum LLR magnitude. Such a scheme of multiple decoding based on bit-guessing has become an active research interest since 2019. At the start, BPF decoding was proposed in [21], which established a fixed critical set as the flip set (FS) to identify error-prone bits. By assigning both $\pm\infty$ as the initial LLR for each identified bit, the BPF decoding performs additional decoding attempts, approaching the SCL performance with the average complexity of BP decoding. However, it suffers from an inaccurate identification of bit errors by the critical set, and it guesses both '0' and '1' for each identified bit instead of performing the bit-flipping. In our latest work [22], a generalized BPF (GBPF) decoding algorithm was proposed, where flipping a bit is defined as

This work was supported in part by Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant KYCX20_0107, in part by NSFC under Grants 61871115 and 61501116, in part by the Jiangsu Provincial NSF for Excellent Young Scholars under Grant BK20180059, in part by the IEEE Circuits and Systems Society Pre-doctoral Grants, in part by the Six Talent Peak Program of Jiangsu Province under Grant 2018-DZXX-001, in part by the Distinguished Perfection Professorship of Southeast University, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Chuan Zhang.)*

Y. Shen, H. Ji, Y. Ren, C. Ji, X. You, and C. Zhang are with the LEADS of Southeast University, the National Mobile Communications Research Laboratory of Southeast University, and the Purple Mountain Laboratories, Nanjing 211189, China. (email: chzhang@seu.edu.cn).

W. Song is with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China. Y. Shen and W. Song contributed equally to this paper.

freezing it to the opposite value of its estimate output by the original BP decoding. The error-prone bits come from the unfrozen bits whose output LLR magnitudes are small, contributing to a better performance than [21]. Moreover, an enhanced BPF (EBPF) decoding [22] optimizes the GBPF by reducing the searching range of the unreliable bits. These two algorithms are similar to those in [5] and [23] for SCF decoding. For SCF decoding, there are numerous strategies to improve the accuracy of the FS, which can be found in [23]–[27]. Particularly, the dynamic SCF (DSCF) decoding [27] considers the decoding with higher bit-flipping order (i.e., the number of nested flipped bits in one decoding round) and is one of the best SCF decoding algorithms in terms of the frame error rate (FER) performance. In contrast, the method of generating FS for BPF decoding is worthy of being further studied, as well as the higher-order decoding with bit-flipping.

In this paper, a higher-order GBPF decoding algorithm is proposed, which is extended from the GBPF in [22]. With much less decoding attempts, the proposed algorithm outperforms the BPF decoding in [21] and does not suffer severe error floors at high SNRs. According to the parity-check state, the decoding errors can be divided into detected and undetected errors [28]. Based on these two error types, we propose two novel FS generation methods. When detected errors appear, there exist check nodes (CNs) that do not satisfy the parity-check, and these unsatisfied CNs (UCNs) can identify some erroneous VNs (EVNs). We propose a strategy to trace to the error-prone bits from the identified EVNs and generate the corresponding FS. For undetected errors, we discuss the bit error distribution under BP decoding and develop a fixed FS based on statistical LLR distribution. We employ these two sets onto GBPF decoding by merging them, thus presenting the algorithm of GBPF with merged sets (GBPF-MS). For polar codes of length 1024 and information rate $1/2$ with 11-bit CRC, the GBPF-MS decoding with 10 decoding attempts outperforms both BPL and DSCF decoding by 0.25 dB at FERs of 10^{-4}. The GBPF-MS decoder is implemented by TSMC 40 nm technology, which is $1.6\times$ and $2\times$ faster than SCL-4 and SCL-8 decoders [29] when it achieves their performance at 2.5 dB, respectively.

The contributions of this paper are summarized as follows.

1) The higher-order GBPF decoding algorithm is proposed, and its achievable performance bound is discussed, which is more accurate than [21].
2) For detected errors, the error patterns of polar BP decoding are summarized, and the concept of loop sets is proposed, inducing a loop-based identification method to generate the FS. For undetected errors, a fixed FS generation method is proposed based on the study of bit error distribution under SC and BP decoding.
3) The GBPF-MS decoding is developed by merging the two sets established by the above two methods, contributing to better performance compared to the state-of-the-art BPF, BPL, and SCF decoding. Also, it can achieve the CA-SCL performance with the complexity of BP decoding at high SNRs.
4) The architecture of GBPF-MS decoder is proposed, where the operations for generating the merged FS are hardware-friendly. The implementation results show its smaller area than adaptive SCL decoders and higher average throughput than GBPF, SCL, and SCF decoders.

The remainder of this paper is organized as follows. Section II introduces the preliminaries of polar codes, BP decoding, and existing bit-flipping strategies. In Section III, the higher-order GBPF decoding is proposed, together with its achievable performance bound. Section IV proposes two FS generation methods for detected and undetected errors and the GBPF-MS decoding. Numerical results are provided in Section V. Section VI discusses the hardware design and presents the implementation results. Section VII concludes this paper. Throughout this paper, boldface letters in italics and upper case denote vectors and matrices, respectively.

II. Preliminaries

A. Polar Codes

After channel combining and channel splitting, N independent copies of binary-input channels are converted to a set of bit channels, of which the capacities are extremally distributed as N increases. The most reliable $K+m$ bit channels are used for conveying information and CRC bits, and the remaining bit channels are chosen to transmit frozen bits, where K represents the effective information length and m denotes the CRC code length. Such a polar code is represented as an (N, K) polar code. The set of unfrozen bits is denoted as $\mathcal{A}$. The resulting bit vector $\boldsymbol{u}$ is encoded to the codeword $\boldsymbol{x}$ by $\boldsymbol{x} = \boldsymbol{u}\mathbf{G}$, where $\mathbf{G} = \mathbf{F}^{\otimes n}$ is the n-th Kronecker product of $\mathbf{F} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$. Throughout this paper, the codeword is modulated by binary-phase shift keying (BPSK) and transmitted over additive white Gaussian noise (AWGN) channels.

B. BP Decoding

Inspired by the realization of BP for Reed-Muller codes [30], Arıkan proposed a BP decoding algorithm for polar codes based on the generator matrix [2]. In addition, two uniform architectures of FG are proposed by exploiting the re-usability of processing elements (PEs). One of the architectures with $N = 8$ is shown in Fig. 1, in which the messages are updated through PEs according to Eq. (1) and undergo an S-shuffle operation [2] when crossing stages.

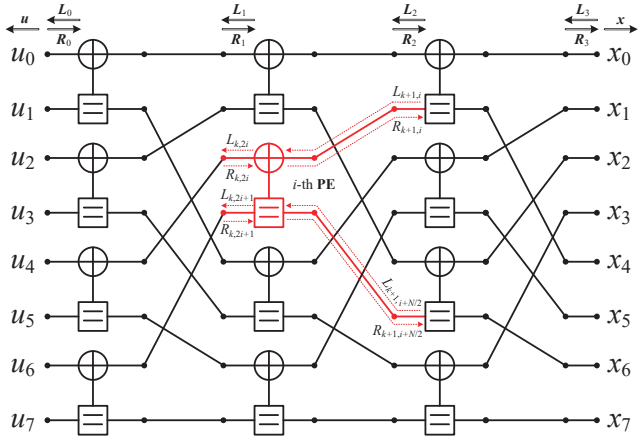

Fig. 1. A uniform BP decoding architecture for $N = 8$, and one PE is marked in red.

$$\begin{aligned} L_{k,2i} &= g(L_{k+1,i}, R_{k,2i+1} + L_{k+1,i+N/2}, \beta_L), \\ L_{k,2i+1} &= g(L_{k+1,i}, R_{k,2i}, \beta_L) + L_{k+1,i+N/2}, \\ R_{k+1,i} &= g(R_{k,2i}, R_{k,2i+1} + L_{k+1,i+N/2}, \beta_R), \\ R_{k+1,i+N/2} &= g(R_{k,2i}, L_{k+1,i}, \beta_R) + R_{k,2i+1}, \\ g(x,y,\beta) &= \mathrm{sgn}(x)\cdot\mathrm{sgn}(y)\cdot\max(\min(|x|,|y|)-\beta, 0). \end{aligned} \tag{1}$$

In Eq. (1), the symbols L and R, represented in the LLR domain, denote the right-to-left and left-to-right messages calculated by the i-th PE at stage k, respectively. The function $g(x, y, \beta)$ employs the offset min-sum approximation [31], and the offset factors for calculating L and R are denoted as β_L and β_R, respectively. The L-messages at the rightmost stage are the LLRs received by the decoder. The R-messages at stage 0 carry a-priori information of each bit, which is $+\infty$ for a frozen bit and 0 for an unfrozen bit. The value of other messages is initially set to 0. The propagation starts from the rightmost stage and terminates when the number of iterations reaches the predefined threshold $I_{\max}$, or the check of $\hat{\boldsymbol{x}} = \hat{\boldsymbol{u}}\mathbf{G}$ is satisfied.

C. Bit-Flipping

For polar codes, the strategy of bit-flipping was first applied to the SC decoding [5] with a single bit flipped, which is called SCF. After the original SC decoding, the SCF decoder creates an FS with size T and re-decodes the codeword up to T times. The bits in the FS are flipped in turn, only when the flipped bit is the first erroneous bit of the original SC decoding, the flipping is valid. How to improve the hit rate for the first wrong bit is studied in [5], [24]–[26]. In [27], the DSCF decoding with higher bit-flipping order is proposed.

The BPF decoding is proposed in [21]. Let BPF-Ω denote a BPF decoding procedure whose *maximum bit-flipping order* is Ω. In [21], each element of the size-T FS includes Ω bit indices. When $\Omega = 1$, the FS, which is a fixed critical set for a specific polar code, is created in the same way as in [25] and is composed of the first bit index of each rate-1 node. The construction method of the critical set for $\Omega \geq 2$ can be found in [21]. For a flipped bit, its a-priori LLR is assigned to $+\infty$ and $-\infty$ in turn, namely both '0' and '1' are tried when guessing this bit value. Therefore, the BPF-Ω decoding in [21] involves up to $2^{\Omega}\times T$ attempts of additional BP decoding. CRC detection is performed after each attempt as an *inter-BP* early termination.

In our latest work [22], a generalized form of BPF decoding was proposed and implemented for 5G polar codes. Based on the output $\boldsymbol{L}_0$ of the original BP decoding, the FS of GBPF decoding consists of T unfrozen indices with the smallest LLR magnitude. Compared with [21], the flipping strategy is modified as assigning $(2\hat{u}_i - 1) \times \infty$ for the flipped bit i. Moreover, an enhanced version was developed to reduce the searching range of unreliable bits, resulting in a lower area consumption than the GBPF [22].

III. Higher-Order GBPF Decoding

In this section, we propose a general form of BPF decoding with a higher bit-flipping order, which is an extended algorithm of the GBPF in [22]. We name the proposed algorithm GBPF-Ω when the maximum bit-flipping order is Ω, to distinguish it from the scheme of always flipping Ω bits simultaneously in [21]. Moreover, compared with [21], a more accurate performance bound of the BP decoding with bit-flipping is presented.

A. Redefinition of Higher-Order Bit-Flipping

The inaccuracy of the fixed critical set in [21] has been pointed out in [22], which comes from the fact that the fixed critical set covers statistically unreliable bits under SC decoding. In other words, it neither considers the statistical bit error distribution under BP decoding nor utilizes information output from the original decoding. The resulting error correction performance improvement becomes insignificant when $\Omega \geq 2$, even with nearly a thousand attempts, and obvious error floors can be observed at high SNRs.

The proposed GBPF-Ω decoding is similar to the improved SCF-Ω decoding in [24], which performs the additional decoding attempts with the incrementing bit-flipping order, until the bit-flipping order exceeds Ω or any estimate passes CRC detection. In other words, the procedure of GBPF-1 is the same as the decoding procedure proposed in [22], and inductively, the GBPF-Ω decoding includes the GBPF-$(\Omega-1)$ decoding. Fig. 2 illustrates the procedure of GBPF-Ω decoding. During the decoding, each intermediate bit-flipping order $\omega (1 \leq \omega \leq \Omega)$ corresponds to a size-T_ω sub-FS $\mathcal{S}_\omega$, in which each element contains ω nested flipped bits. The set $\mathcal{S}_\omega$ is established based on the preorder decoding result and the set $\mathcal{S}_{\omega-1}$. If all estimates of the decoding with order ω do not pass the CRC, then $T_{\omega,\omega-1}$ elements are extracted from $\mathcal{S}_{\omega-1}$, each of which identifies $T_{\omega,\omega}$ error-prone bits as the ω-th flipped bit. The identification method follows [22], which selects the bit indices with the smallest LLR magnitude based on the output of the preorder decoding. One identified ω-th flipped bit, together with its preorder $\omega-1$ bits in $\mathcal{S}_{\omega-1}$, constitutes an element in $\mathcal{S}_\omega$. Therefore, the maximum number of order-ω decoding attempts is $T_\omega = T_{\omega,\omega-1} \times T_{\omega,\omega}$, and the maximum additional attempt number for GBPF-Ω decoding is $\sum_{\omega=1}^{\Omega} T_\omega$. The overall FS $\mathcal{S} = \bigcup_{\omega=1}^{\Omega} \mathcal{S}_\omega$.

It is intuitive that the size of the FS and the number of nested flipped bits both increase with Ω, which brings challenges to the memory and computational resources. Thus, in this paper, only the decoding of GBPF-1 and GBPF-2 is studied, and the detailed procedure of the GBPF-2 decoding is listed in Algorithm 1. If all T_1 candidate estimates fail CRC during the order-1 decoding, then the first $T_{2,1}$ candidates of them are selected for the order-2 decoding. The set of the nested flipped bits for current decoding attempt is denoted as $\mathcal{F} = \{i_1, i_2\}$, which is an element of $\mathcal{S}_2$. The bit i_1 is selected from the $T_{2,1}$ flipped bits of $\mathcal{S}_1$, and the bit i_2 is selected from the $T_{2,2}$ flipped bits that are identified based on the decoding result with i_1 flipped. In lines 7, 18, and 19 of Algorithm 1, the bit-flipping is executed by modifying the a-priori information of flipped bits. In [21], [22], the value of τ is ∞. Instead, we can choose an appropriate finite value in real applications, which could improve the performance further. Once bit i is wrongly identified, flipping it by setting $\pm\infty$ leads to an absolute decoding error, because $\hat{u}_i$ is the hard decision of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3017656, IEEE Transactions on Communications

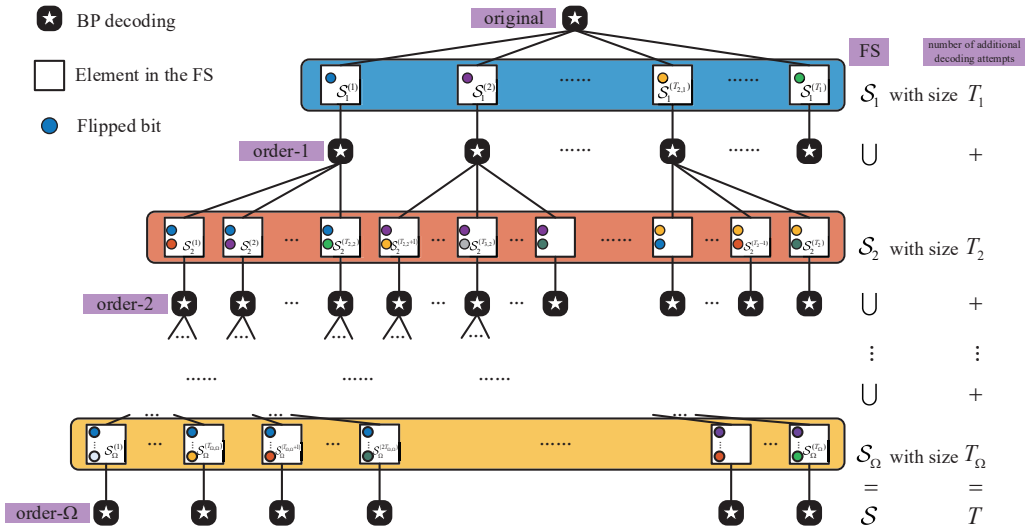

Fig. 2. The procedure of GBPF-Ω decoding, where $\mathcal{S}_\omega^{(i)}$ denotes the i-th element in $\mathcal{S}_\omega$.

Algorithm 1: GBPF-2 Decoding

$\hat{\boldsymbol{u}}_0 \leftarrow$ BP decoding;
if CRC detection succeeds **then**
  return $\hat{\boldsymbol{u}}_0$;
else
  // decoding with bit-flipping order 1
  $\mathcal{S}_1 \leftarrow i \in \mathcal{A}$ of T_1 smallest $|\boldsymbol{L}_0|$; // generate $\mathcal{S}_1$
  for $t = 1$ **to** T_1 **do**
    $\mathcal{F} = \{i_1\} \leftarrow \mathcal{S}_1^{(t)}$;
    $R_{0,i_1} \leftarrow (2\hat{u}_0[i_1] - 1) \times \tau$;
    $\hat{\boldsymbol{u}}_1^{(t)} \leftarrow$ BP decoding;
    if CRC detection succeeds **then**
      return $\hat{\boldsymbol{u}}_1^{(t)}$;
  // decoding with bit-flipping order 2
  for $t_1 = 1$ **to** $T_{2,1}$ **do**
    $\mathcal{S}' \leftarrow i \in \mathcal{A}$ of $T_{2,2}$ smallest $|\boldsymbol{L}_{1_0}^{(t_1)}|$;
    // generate a temporary size-$T_{2,2}$ set $\mathcal{S}'$
    $\left\{\mathcal{S}_2^{((t_1-1)\cdot T_{2,2}+1)}, \cdots, \mathcal{S}_2^{(t_1 \cdot T_{2,2})}\right\} \leftarrow \{\mathcal{S}_1^{(t_1)} \cup \mathcal{S}'^{(1)}, \cdots, \mathcal{S}_1^{(t_1)} \cup \mathcal{S}'^{(T_{2,2})}\}$; // generate $\mathcal{S}_2$
  for $t_1 = 1$ **to** $T_{2,1}$ **do**
    for $t_2 = 1$ **to** $T_{2,2}$ **do**
      $\mathcal{F} = \{i_1, i_2\} \leftarrow \mathcal{S}_2^{((t_1-1)\cdot T_{2,2}+t_2)}$;
      $R_{0,i_1} \leftarrow (2\hat{u}_0[i_1] - 1) \times \tau$;
      $R_{0,i_2} \leftarrow (2\hat{u}_1^{(t_1)}[i_2] - 1) \times \tau$;
      if CRC detection succeeds **then**
        return $\hat{\boldsymbol{u}}_2^{((t_1-1)\cdot T_{2,2}+t_2)}$;
  return $\hat{\boldsymbol{u}}_2^{(T_2)}$;

$(L_{0,i} + R_{0,i})$. In contrast, setting a finite value (e.g., $\tau = 8$) can mitigate the overestimation, which could avoid some error estimates caused by wrong bit-flipping.

B. Performance Bound of GBPF Decoding

In [5], [24], an oracle-assisted SC (OA-SC) decoder is designed to analyze the performance bound of the SCF, where the first erroneous bit is assigned with the correct value known by an oracle. In the SC decoding, incorrect estimates are caused by the channel noise or the forward error propagation, and the first bit error is definitely caused by the noise [5]. Therefore, when $\Omega = 1$, only flipping the first erroneous bit makes it possible to correct the error while flipping the following error bits does not help the performance improvement. Similarly, a BP-Genie decoder is presented in [21] to show the achievable performance of the BPF decoding, which also flips the first erroneous bit (i.e., the erroneous bit with the smallest bit index) when the error information is known by a "genie". However, the BP decoding is a parallel algorithm, after which all bits are estimated simultaneously. Flipping other bits also contributes to the error-correcting due to the flooding property. Moreover, the noise-caused error can be propagated to both forward and backward directions and might be reinforced by cycles in the FG. The first wrong bit in BP decoding is not always caused by the channel noise, and flipping it may not work. Hence, it is not reasonable to analyze only the first index when discussing the performance bound.

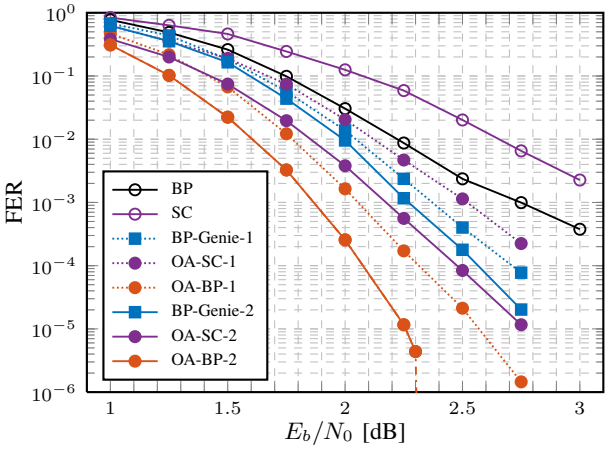

Fig. 3. FER performance of the BP-Genie, OA-SC, and OA-BP decoders with $\Omega = 1$ and 2. Code parameter: $(1024, 512)$ polar codes with 11-bit CRC, $I_{\max} = 100$, $\beta_L = 0$, $\beta_R = 0.25$, and the design SNR of constructions is 2.5 dB.

Herein, we modify the design of BP-Genie decoder and name the new one as oracle-assisted BP (OA-BP) decoder to differentiate them. The OA-BP decoder can predict the optimal performance that the BPF/GBPF decoding could achieve. For

$\Omega = 1$, the OA-BP decoder knows which bit estimates make mistakes after the original decoding and re-decodes the codeword in turn by making one erroneous bit frozen to the correct value. When $\Omega = 2$, if freezing one bit still results in errors, then the bits from the new erroneous bit sequence are frozen in turn.

Fig. 3 shows the FER performance of the OA-BP decoder for $(1024, 512)$ polar codes concatenated with 11-bit CRC. The performance of the BP-Genie [21] and OA-SC [5], [24] decoders is also provided as comparisons. To achieve satisfactory performance of BP decoding, we employ the construction using a genetic algorithm (GenAlg) [32] to allocate bit channels for BP decoding throughout this paper. For a fair comparison, for SC-based decoding, the nearly optimal Tal-Vardy method [33] is adopted to construct polar codes. It can be observed that the OA-SC shows a significant improvement compared with SC, achieving 0.7 dB and 1 dB gains for $\Omega = 1$ and 2 at FER$= 10^{-3}$, respectively. At the same FER, the OA-BP obtains gains from 0.7 dB for $\Omega = 1$ to 0.9 dB for $\Omega = 2$ than BP decoder. Due to the excellent performance of BP under the GenAlg construction, OA-BP shows the optimal performance under the same bit-flipping order, and an ultra-low error probability is reached after 2.3 dB. By contrast, the BP-Genie decoder only considers the first erroneous bit while neglects the flooding property of BP, resulting in only a 0.4 dB gain for $\Omega = 2$ compared with BP when FER$= 10^{-3}$. As discussed above, flipping the erroneous bit with the smallest index does not necessarily correct the error. On the other hand, in some cases, flipping one bit can even correct multiple noise-caused erroneous bits at the same time, leading to the significant performance improvement of the OA-BP.

IV. GBPF Decoding with Merged Sets

The performance of GBPF decoding depends on the accuracy of the FS. If the FS is able to hit the error-correctable bits, then the GBPF decoding could achieve the performance of OA-BP as Fig. 3 presents. The LLR magnitude-based FS adopted in the GBPF decoding shows flexibility and improved accuracy compared with the fixed critical set in [21]. However, the LLR sorter occupies most of the area in the GBPF decoder. In addition, this method requires 7-bit quantization, because a lower quantization bit number makes it difficult to distinguish LLRs of different bits, introducing the performance degradation. To this end, we propose the GBPF-MS decoding algorithm. According to the two error types: detected errors and undetected errors, we propose two targeted methods to generate their corresponding FSs. The GBPF-MS decoding employs a merged FS by combining the above generated two sets to an FS with a fixed size.

A. Error Patterns

Before introducing the proposed algorithm, the error patterns of the BP decoding need to be well understood. Unlike the SC decoding, errors in BP decoding are initially caused by external noises and local cycles in the FG. Then, these errors are bidirectionally propagated to more VNs over the FG. From the view of the parity-check state, errors can be divided into detected and undetected errors. Fig. 4 is a variant of Fig. 1, which is composed of VNs and CNs. In the FG, the shortest cycle has a girth of 12 and contains 6 VNs and 6 CNs. All VNs and CNs form a set $\mathcal{G}$, and VNs at intermediate stages represent partial sums. Denote a column of VNs at stage k as $\boldsymbol{s}_k$, thus $\boldsymbol{s}_0 = \boldsymbol{u}$ and $\boldsymbol{s}_n = \boldsymbol{x}$. Each PE includes a degree-3 CN and a degree-2 CN, with the constraints of $s_{k,2i} + s_{k,2i+1} + s_{k+1,i} = 0$ and $s_{k,2i+1} + s_{k+1,i+N/2} = 0$, respectively.

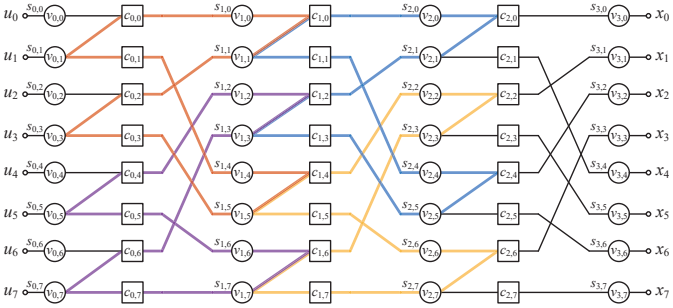

Fig. 4. CNs and VNs (denoted by $c_{k,i}$ and $v_{k,i}$) in the FG with $N = 8$, and four shortest cycles are painted with colors.

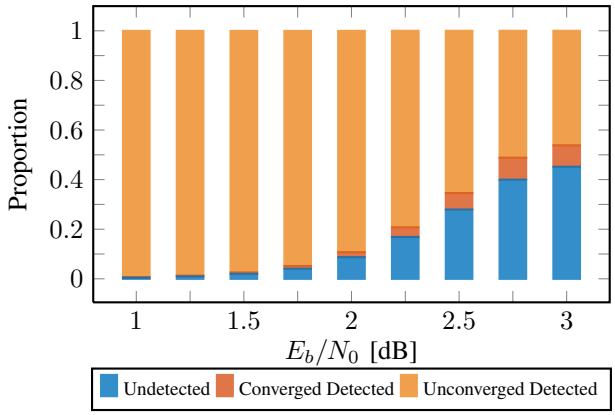

Fig. 5. Error distribution for BP decoding with the same code parameters as Fig. 3. There are cases that an erroneous codeword passes the CRC, but such cases only appear at low SNRs, with extremely low proportion (nearly 0.03% before 2 dB).

Fig. 5 presents the distribution of undetected and detected errors. Undetected errors are all converged, while detected errors can be further divided into unconverged, oscillation, and converged types [34]. In this figure, the oscillation detected errors are considered as uncoverged, because some of them have changing period and are hard to distinguish. It can be seen that detected errors are dominant for given SNRs, and the frequency of undetected errors rises gradually as SNR increases.

B. FS for Detected Errors

For detected errors, there exist a few UCNs, which can be obtained by examining the parity-check constraints. Denote the sets of UCNs and EVNs as $\mathcal{C}_u$ and $\mathcal{V}_e$, respectively. The nodes of stage 0 in the set $\mathcal{V}_e$ are our focus but are unknown to the decoder. However, the set $\mathcal{C}_u$ can help identify those erroneous bit nodes. Let $\Gamma(\mathcal{C}_u)$ be the set of neighboring VNs of CNs in $\mathcal{C}_u$, then $\Gamma(\mathcal{C}_u) \cap \mathcal{V}_e \neq \varnothing$. In other words, the VNs connected to a UCN must cover at least one EVN. Based on the set of $\Gamma(\mathcal{C}_u) \cap \mathcal{V}_e$, we can identify more EVNs that contribute to

mis-satisfied CNs. When it comes to stage 0, error-prone bits can be located. Therefore, the structure of EVNs needs to be studied first, which bridges the set $\Gamma(\mathcal{C}_u)$ and erroneous bits.

1) Error Structures

From the perspective of the cause, we can classify the EVNs into three types: intrinsic EVNs, extrinsic EVNs, and propagative EVNs. They are caused by cycles, directly caused by the noise, and propagated from extrinsic or intrinsic EVNs, respectively. It is widely known that a small cycle can degrade the decoding performance, and only if an FG is cycle-free, the BP decoding can reach the performance of the maximum likelihood decoding [28]. If cycles lead to an error, the error will propagate across cycles, making EVNs form *loop structures*. Since the FG is multi-stage based on the generator matrix, an originally intrinsic or extrinsic EVN can spread the error through the PE and split into a series of EVNs in *tree structures*. The property of tree structures for binary erasure channels (BECs) has been thoroughly studied in [19]. For AWGN channels, the splitting manner of errors onto each PE is determined by the noise realization and local cycles. Therefore, the specific splitting branches of the tree are much more difficult to analyze, and sometimes there appears more than one error chain.

When intrinsic EVNs spread errors out of the cycles and forms the error branches, the trapping set is a union of the tree-structure EVNs and the loop-structure EVNs. In LDPC decoding, trapping sets have been well studied to overcome the error floor, which are formed by a combination of multiple cycles in the bipartite LDPC FG [35]. Similar to LDPC, the intrinsic errors in polar BP decoding attribute to one or more cycles, causing EVNs in these cycles. But the difference is that these intrinsic errors will diffuse out of the cycles in the multi-stage FG, making the cardinality of the EVN set larger. Therefore, it is inappropriate to analyze the intrinsic errors using the definition of trapping sets in LDPC. Herein, a new concept, loop set, is proposed with the following definition.

Loop Set: A set $\mathcal{L} \subset \mathcal{G}$ is an $(a, b|e)$ loop set, in which a VNs form a *closed* loop with b neighboring CNs, and e denotes the number of UCNs among the b CNs.

The loop set includes only intrinsic EVNs. The value of e determines the error-correcting effect of CNs in the loop set. During message propagation, satisfied CNs attempt to reinforce the current state of associated PEs, but the remaining UCNs try to correct the detected EVNs. If e is large, the UCNs can provide forceful messages to local PEs, thus possibly correcting errors. If e is small or even equal to 0, the decoder may trap into the loop set, converging to a false codeword or oscillating among different states periodically. To better illustrate the EVN structures of converged and unconverged detected errors, two cases are provided as follows.

Fig. 6 shows a $(6, 4|1)$ loop set, which is formed by the shortest cycle adding 4 neighboring CNs. Three of them satisfy the check constraints due to their associated propagative EVNs. Messages coming from the only one UCN are too weak to overcome messages from the other 9 satisfied CNs, causing the decoder to converge to a codeword other than the transmitted one.

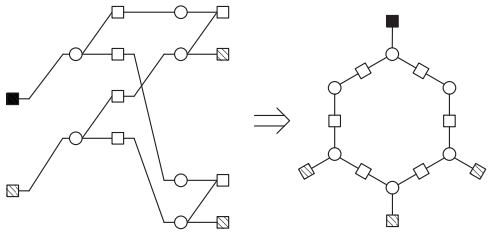

Fig. 6. A $(6, 4|1)$ loop set, and the right sub-figure is in an unrolled form.

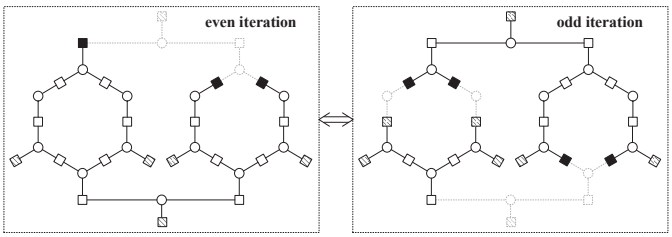

Fig. 7. A loop set that behaves as a $(12, 8|3)$ loop set and a $(10, 10|4)$ loop set periodically.

Fig. 7 depicts an oscillation case, in which two shortest cycles at the same stage connect at the previous stage and form a stable loop set. With CNs being contending for the check-satisfying, this loop set behaves as $(12, 8|3)$ and $(10, 10|4)$ periodically. In even iterations, one cycle is broken, and three UCNs try to correct errors, thus presenting a new structure in odd iterations. The resulting four UCNs will lead to the loop set of even iterations, forming oscillation.

2) FS Generation Method

The generation of FS for detected errors is based on UCNs. The UCNs in stage k are obtained by performing parity-checks of stage k after calculating $\boldsymbol{R}_{k+1}$. There must be an EVN directly associated with a UCN, which will connect other EVNs in tree or loop structures, but the specific structure is unknown to us during the decoding. Therefore, we consider both structures and propose a two-step method to bridge UCNs and erroneous bits. The first step is to locate more error-prone VNs along loops based on the UCNs, and the second step is to trace to the leftmost end and identify error-prone bits.

For LDPC codes, it is widely known that the BP performance depends on the length of the shortest cycle. Also, it is widely observed that small trapping sets are dominant [28], [36], which are formed by short cycles. Polar BP decoding shows similar behavior, where the loop sets with one or two shortest cycles are most frequent according to extensive simulations. Therefore, we only focus on the shortest cycle. According to Fig. 6, the shortest cycle forms the minimum loop set including six VNs, where two of the VNs are on the left and the remaining four VNs are at the right stage. If the left-hand two VNs are located at stage k, we denote such a minimum loop set as $\mathcal{L}_k$ [1]. The procedure of the FS generation method is listed below.

- **Step 1:** For a CN $c \in \mathcal{C}_u$ at stage k, identify the VNs in the following three loop sets, $\mathcal{L}_{k-1}, \mathcal{L}_k$, and $\mathcal{L}_{k+1}$, where the node c is located or connected. All VNs identified by

[1]One stage has $N/4$ loop sets, so a row index is needed to refer to a specific loop set. For ease of description, we omit the row index in the following.

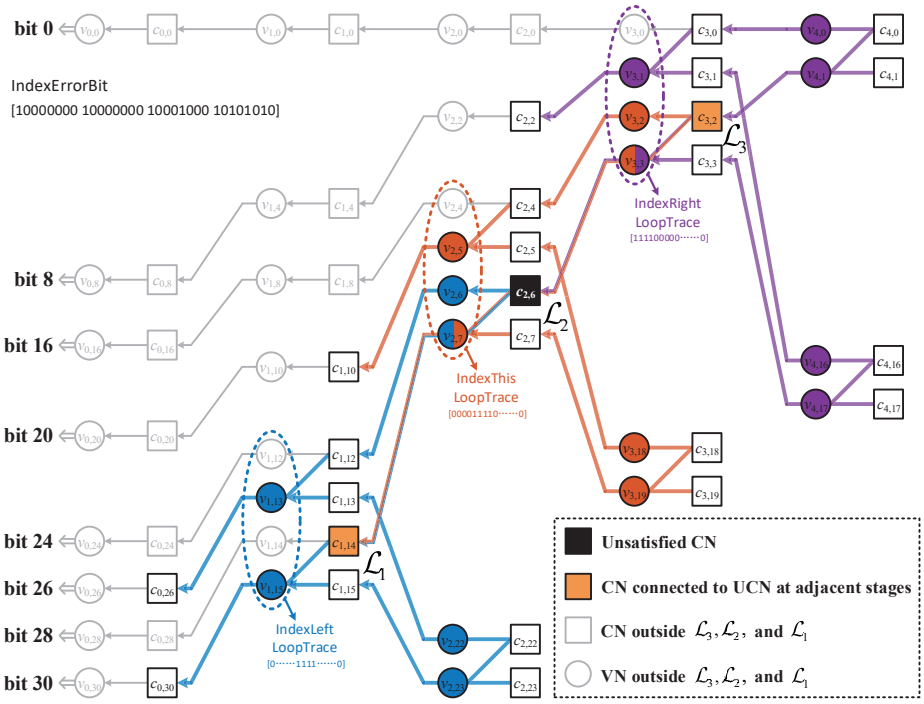

Fig. 8. A case of identifying the error-prone bits given the UCN of $c_{2,6}$ for $N = 32$, where the three loop sets $\mathcal{L}_1, \mathcal{L}_2$, and $\mathcal{L}_3$ (with their covers VNs) are marked by blue, red, and purple, respectively.

UCNs in this way are considered as error-prone VNs.

- **Step 2:** According to the shuffled architecture shown in Fig. 1, get the index in the original order for each identified VN by tracing it to the bit side. Then, merge the same indices, and select indices belonging to $\mathcal{A}$ to build the FS.

Let us start with an example with one UCN. Under such condition, other CNs are all satisfied. Fig. 8 depicts the proposed identification method with one UCN of $c_{2,6}$, and its associated three loop sets are marked by blue, red, and purple, respectively. Since every two adjacent loop sets share one VN, a total of 16 VNs are identified by one UCN. The next step is to trace to error-prone bits based on these identified VNs. If an identified VN is erroneous, it will propagate the error to the left until the error comes to the bit side. This connection path from an identified VN to a bit is called a tracing path, which also includes EVNs and CNs at intermediate stages. If the error passes a degree-2 CN on the tracing path, it will directly propagate from this CN to the left. For a degree-3 CN, the error only propagates to one direction, either to the left-hand VN or to the lower left VN, otherwise, it will not match the assumption of satisfied CNs. To simplify operations, we select the left-hand VN with the same index as the degree-3 CN, which is proved to be feasible for accurate identification of bit errors through extensive simulations. To this end, there is only one tracing path from an EVN, ending at one bit.

According to the above tracing criteria, 16 identified EVNs generate a total of 8 tracing paths, because the paths of some nodes are overlapped. It is intuitive that for each loop set, the left-hand two VNs and their connected VNs on the right share the same tracing paths. For example, in Fig. 8, nodes $c_{2,5}$ and $c_{3,18}$ have the same tracing path ending at bit 20. Hence, a loop set $\mathcal{L}_k$ has four tracing paths, on which the four indices of VNs at stage k are consecutive. For a UCN $c_{k,i}$, the four indices of its related VNs at stage k are $4\times\lfloor i/4\rfloor$ to $4\times\lfloor i/4\rfloor+3$. As shown in Fig. 8, based on $c_{2,6}$, four tracing paths from $\mathcal{L}_2$ cover nodes $v_{2,4}$ to $v_{2,7}$ at stage 1, ending at bits $16, 20, 24$, and 28. The row indices of two adjacent CNs on the left and right of $c_{k,i}$ can be obtained by shuffling i, so as to locate VN indices at stage $k-1$ and $k+1$ on the tracing paths from $\mathcal{L}_{k-1}$ and $\mathcal{L}_{k+1}$, respectively. For example, in Fig. 8, the CN connected to $c_{2,6}$ at stage 3 is $c_{3,2}$, and this node $c_{3,2}$ can index the tracing paths from $\mathcal{L}_3$. Thus, the possible errors of bits identified by the tracing paths from $\mathcal{L}_3$ can be regarded as coming from nodes $v_{3,0}$ to $v_{3,3}$. For a loop set, two of the tracing paths are overlapped with tracing paths of a neighboring loop set. Thus, the tracing paths from 16 VNs will converge on 8 bits at the bit side, as shown in Fig. 8. The located 8 bits based on one UCN are considered as error-prone, and all the bits that belong to $\mathcal{A}$ and are located by UCNs are included in the generated FS.

The above method could identify most EVNs either in splitting trees or loop sets. If the error propagates in a minimum loop set, this method can effectively identify all EVNs in this shortest cycle. For a UCN $c_{k,i}$, if the index i mod 4 equals to 0 or 1, the minimum loop sets where $c_{k,i}$ is located include $\mathcal{L}_{k-1}$ and $\mathcal{L}_k$; otherwise, the minimum loop sets where $c_{k,i}$ is located include $\mathcal{L}_{k-1}$, $\mathcal{L}_k$, and $\mathcal{L}_{k+1}$. When the decoder traps into a loop set and converges into a stable state, the node $c_{k,i}$ could identify the EVNs in $\mathcal{L}_{k-1}$ or $\mathcal{L}_{k+1}$. When the decoder oscillates, the UCN may occur in the temporally broken loop and identify all EVNs in this loop. If errors propagate in tree structures, an EVN could propagate the error to any unfrozen VNs (with different probabilities). The specific propagation is difficult to analyze, which depends on the decoding algorithm, quantization scheme, and noise

realization. However, the proposed method can enlarge the identification range of EVNs, compared with identifying the VNs directly connected to UCNs, thereby enhancing the hit rate for erroneous bits.

Algorithm 2: FS generation of detected errors

IndexUCN ← **new** binary array of size N;
IndexThisLoopTrace, IndexLeftLoopTrace, IndexRightLoopTrace ← **new** binary array of size N;
// record VN indices at stage $k, k-1$ and $k+1$ on tracing paths from $\mathcal{L}_k, \mathcal{L}_{k-1}$ and $\mathcal{L}_{k+1}$, respectively
IndexErrorBit ← **new** binary array of size N; // records the identified unfrozen bits
for $k = 0$ **to** $n-1$ **do**
  for $i = 0$ **to** $N/2-1$ **do**
    IndexUCN$[2i] \leftarrow s_{k,2i} \oplus s_{k,2i+1} \oplus s_{k+1,i}$;
    IndexUCN$[2i+1] \leftarrow s_{k,2i+1} \oplus s_{k+1,i+N/2}$;
  // CNIndexToFourVNIndex expands each CN index to four VN indices
  IndexThisLoopTrace ← CNIndexToFourVNIndex(IndexUCN);
  IndexLeftLoopTrace ← CNIndexToFourVNIndex(R2L_Shuffle(IndexUCN));
  IndexRightLoopTrace ← CNIndexToFourVNIndex(L2R_Shuffle(IndexUCN));
  switch k **do**
    case 0
      ShuffleToOriginal(IndexThisLoopTrace, k);
      ShuffleToOriginal(IndexRightLoopTrace, $k+1$);
      IndexErrorBit ← IndexErrorBit | IndexThisLoopTrace | IndexRightLoopTrace;
    case $n-1$
      ShuffleToOriginal(IndexLeft, $k-1$);
      IndexErrorBit ← IndexErrorBit | IndexLeftLoopTrace;
    case $n-2$
      ShuffleToOriginal(IndexLeftLoopTrace, $k-1$);
      ShuffleToOriginal(IndexThisLoopTrace, k);
      IndexErrorBit ← IndexErrorBit | IndexLeftLoopTrace | IndexThisLoopTrace;
    otherwise
      ShuffleToOriginal(IndexLeftLoopTrace, $k-1$);
      ShuffleToOriginal(IndexThisLoopTrace, k);
      ShuffleToOriginal(IndexRightLoopTrace, $k+1$);
      IndexErrorBit ← IndexErrorBit | IndexLeftLoopTrace | IndexThisLoopTrace | IndexRightLoopTrace;
IndexErrorBit ← IndexErrorBit & IndexUnfrozen;

The procedure to identify error-prone bits and generate the FS is summarized in Algorithm 2. All operations are performed in the binary field, which is quite friendly to hardware implementations. The indices of UCNs at current stage are recorded in a binary array IndexUCN, and whether a bit is included in the FS is recorded in array IndexErrorBit. The function ShuffleToOriginal(array, k) shuffles the array from stage k to the original order. It should be noted that both stage 0 and n keep the original order of indices. Thus, during implementations, the shuffling operations can be carried out to either stage 0 or stage n.

C. Fixed FS for Undetected Errors

As shown in Fig. 5, undetected errors become more frequent as the SNR increases. If the errors are not detected by parity-check, EVNs form a so-called stopping set [19], where all CNs connected to EVNs are mis-satisfied. Nevertheless, the undetected errors can be exposed by CRC test. With unknown information about the EVNs, the output LLR magnitude can help identify error-prone bits. Besides, an alternative is to establish a fixed FS according to statistical error distribution, which has been widely studied in SCF decoding [26].

Fig. 9 presents the average LLR magnitude under SC and BP decoding, where the simulation configuration is consistent with Fig. 3. The mean LLR value of each unfrozen bit under SC is predicted by the Gaussian approximation (GA) method [37], and those belonging to the critical set in [21] are marked in yellow. Density evolution or GA can also be applied to predict the LLR distribution for BP decoding, but resulting in lower accuracy due to the existence of extensive cycles. Herein, the Monte Carlo method is employed to compute the mean LLR magnitude for BP decoding. It can be seen that bits in the critical set are prone to errors under SC decoding, because the bit channels where they are located are not reliable. However, since most of them are adjacent to the frozen bits carrying infinity a-priori messages, they exhibit extremely high reliability under BP decoding. Hence, the critical set in [21] only considers the bit channel quality but ignores the bit correlation in the BP decoding.

Intuitively, we should select the most unreliable T bits under BP decoding as flipped bits when $\Omega = 1$, which is feasible when the initial value τ of flipped bits is ∞. Fig. 10 compares the prediction accuracy of different fixed FSs for $(1024, 512)$ codes, which is defined as the ratio of corrected error frames to sampled error frames. When $\tau = \infty$, the fixed FS generated by unreliable bits under BP shows a similar accuracy to the FS based on the output LLR magnitudes. By comparison, using the critical set in [21] requires 10 decoding attempts to achieve the performance of $T = 1$ in [22]. When $\tau = 8$, the accuracy of FS based on BP error distribution is improved by 20%, because some overestimation of $\tau = \infty$ is avoided. Exceptionally, the critical set in [21] exhibits a more significant performance improvement. The unfrozen bits in that critical set are reliable under BP

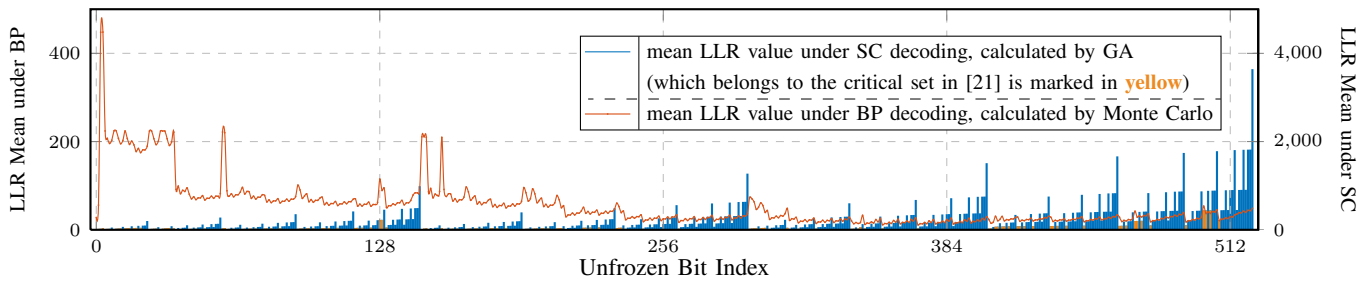

Fig. 9. The distribution of LLR mean under SC and BP decoding at 2.5 dB for $(1024, 512)$ polar codes with 11-bit CRC.

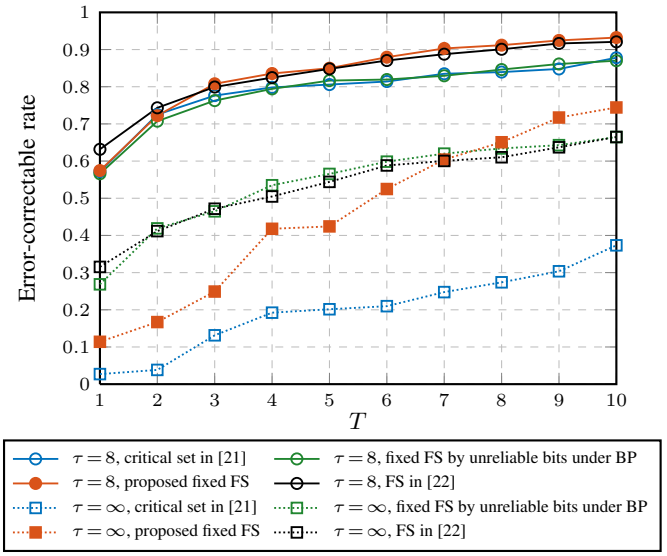

Fig. 10. Prediction accuracy of different fixed FSs with the same code parameters as Fig. 9.

decoding, usually presenting extremely large magnitudes of output LLRs. Therefore, if bit i is flipped, it is likely to be wrongly flipped, e.g., $\hat{u}_i = u_i = 0$ but $R_{0,u_i} = -\tau$ for the new decoding attempt. When $\tau = \infty$, the wrong flipping of u_i will definitely cause an error. However, if the value of τ is finite and appropriate, e.g., $\tau = 8$, the large value of L_{0,u_i} possibly eliminates the effect of the inaccurate R_{0,u_i}. Meanwhile, such a R_{0,u_i} message can be regarded as an external noise for the decoder, which can help the trapping or oscillating state escape from the loop set.

Thus, to achieve satisfactory error-correcting performance, a fixed FS should consider both LLR distribution under SC and BP decoding. On the one hand, flipping an unreliable bit under BP decoding can greatly correct bit errors. On the other hand, flipping an unreliable bit under SC decoding can introduce artificial noise to correct some intrinsic EVNs. To this end, we propose a novel method to generate the fixed FS based on a hybrid LLR mean, where the hybrid LLR mean is the average of the LLR mean under SC and BP decoding. For $\Omega = 1$, the fixed FS consists of T indices with the smallest hybrid LLR mean. For $\Omega = 2$, when determining the second flipped bit i_2 based on the first flipped bit i_1, the LLR distribution of BP is simulated with bit i_1 being frozen as the correct value. It should be noted that the fixed FS can be generated off-line and pre-stored before online decoding.

As shown in Fig. 10, the proposed fixed FS outperforms the set based on LLR mean under either SC or BP decoding for $\tau = 8$, saving 2 decoding attempts at an accuracy of 90%. Also, it exhibits slight accuracy improvement than the output LLR magnitude-based FS.

D. Merged FS

After analyzing the error patterns of detected errors and error distributions of undetected errors, we have proposed two FS generation methods for two error types. The FS for detected errors is generated case by case. Its cardinality is not fixed, which is usually considerably large for unconverged errors and generally small for converged errors.

In this subsection, the GBPF-MS decoding is proposed by combining the two strategies for undetected and detected errors. The merged FS in the GBPF-MS has a fixed size to avoid excessive memory requirements when the size of the generated set for detected errors is too large. Denote the FS for detected errors as $\mathcal{S}_{\text{det}}$. The procedure is listed as follows.

If $|\mathcal{S}_{\text{det}}|$ is larger than a predefined value T' (i.e, T_1 for $\omega = 1$ and $T_{2,2}$ for $\omega = 2$), the set $\mathcal{S}_{\text{det}}$ is pruned to a size of T'. Namely, the first T' elements of $\mathcal{S}_{\text{det}}$ are identified as error-prone bits. Otherwise, the set $\mathcal{S}_{\text{det}}$ is supplemented by the proposed fixed FS, and then the first T' elements of the enlarged set are identified. Except for the set generation, the other operations of GBPF-MS decoding are the same as the GBPF decoding.

V. Numerical Results

In this section, the results of the error correction performance are provided. The offset factors, β_L and β_R, are set to 0 and 0.25 according to [22], respectively. Unless stated otherwise, the initial R_0 value of flipped bits is 8, and $I_{\max} = 100$. The SCL, DSCF, and BPL decoding algorithms are simulated as comparisons, and the design SNR of all constructions is 2.5 dB.

A. Comparison with BPF Decoding

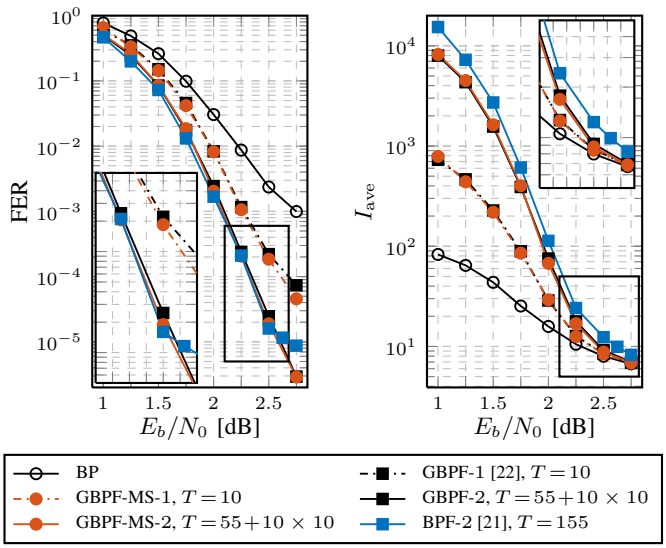

Fig. 11. FER performance and average iteration comparisons of BPF, GBPF, and GBPF-MS decoding for $(1024, 512)$ polar codes with 11-bit CRC.

Fig. 11 compares the FER performance of the BPF [21], GBPF, and GBPF-MS decoding for $(1024, 512)$ polar codes with 11-bit CRC. When $T = 10$, the GBPF-MS-1 presents a 0.5 dB gain at FER$= 10^{-3}$ compared with the BP decoding, and it gradually outperforms the GBPF decoding with the increasing of E_b/N_0.

Under GenAlg construction at design SNR of 2.5 dB, the value of T in the BPF decoding [21] is 155. Hence, the maximum numbers of additional decoding attempts for BPF-1 and BPF-2 are 310 and 620, respectively. For BPF-2, a severe error floor appears after 2.5 dB, which results from the inaccurate order-2 critical set proposed in [21]. In this set, the second flipped bit is serially determined, which is the least

reliable bit under SC decoding with a larger index than the first flipped bit. In contrast, the GBPF-2 and GBPF-MS-2 decoding exhibit nearly the same performance with only 155 decoding attempts. Their performance gains at high SNRs are attributed to the error-prone bit identifications based on the result of the preorder decoding.

Regarding the average iteration number I_{ave}, the GBPF and GBPF-MS are also superior to the BPF, because the BPF decoding requires two attempts for each flipped bit. The GBPF-MS-2 reduces the average iterations by 28.9% at 2.5 dB compared with BPF. Also, it can be seen that all procedures with bit-flipping endure a high iteration number at low SNRs but gradually approach the complexity of BP decoding as E_b/N_0 increases.

B. Comparison with SC-Based and BPL Decoding

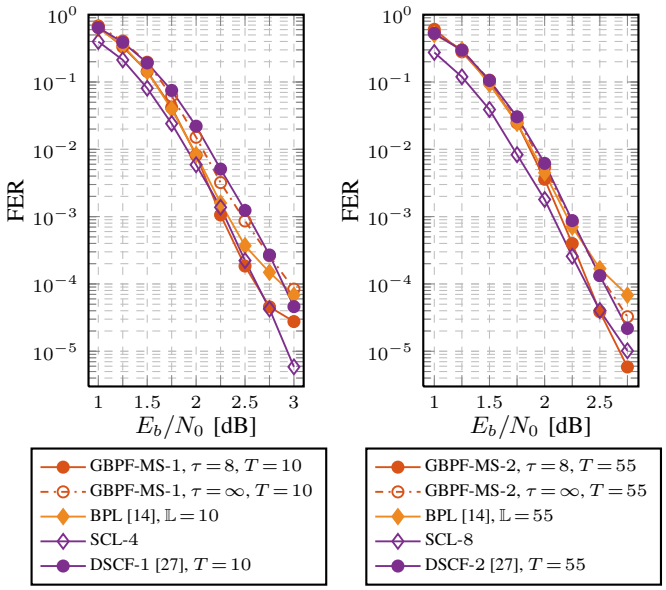

Fig. 12. FER performance of GBPF-MS decoding against BPL, SCL, and DSCF decoding for $(1024, 512)$ polar codes with 11-bit CRC, where $\mathbb{L}$ is the list size and the parameter α in [27] is set to 0.4.

Fig. 12 provides the performance of the GBPF-MS decoding against the BPL, SCL, and DSCF decoding. Also, it depicts the performance improvement of GBPF-MS decoding by reducing the τ value from ∞ to 8, achieving gains of 0.35 dB and 0.15 dB at FERs of 10^{-4} for the shown two cases, respectively. The curves of BPL decoding enter a region in which performance flattens earlier, resulting in 0.35 dB gaps compared with the GBPF-MS for both cases.

In contrast to SCL decoding, the GBPF-MS can achieve the performance of $\mathbb{L} = 4$ and 8 at 2.5 dB with $\Omega = 1, T = 10$ and $\Omega = 2, T = 55$, requiring average 8.38 and 8.68 iterations, respectively. With the same bit-flipping parameters, the GBPF-MS outperforms the DSCF [27] by nearly 0.2 dB at FER$=10^{-4}$. This is because the DSCF decoding can correct up to Ω bit errors, but the proposed BPF decoding can correct more than Ω erroneous bits as discussed at the end of Section III-B. Nevertheless, with $T = 55$, the DSCF-2 has approached the performance of OA-SC decoding while the GBPF-MS-2 is 0.3 dB away from the OA-BP. This is due to that the metric in [27] can reflect the probability of whether a bit is the first erroneous bit. The study of the metric for evaluating the probability that a bit is error-correctable under BP decoding is left as an open research point.

C. Error Correction Performance with Quantization

The above comparisons are under floating-point representations, where the GBPF-MS exhibits slight performance improvement compared to GBPF. Besides, if we directly use the generated fixed FS to identify error-prone bits, the resulting performance has negligible degradation (0.05 dB) as shown in Fig. 13. By contrast, the performance gain employing the merged sets becomes remarkably explicit under low-bit quantization.

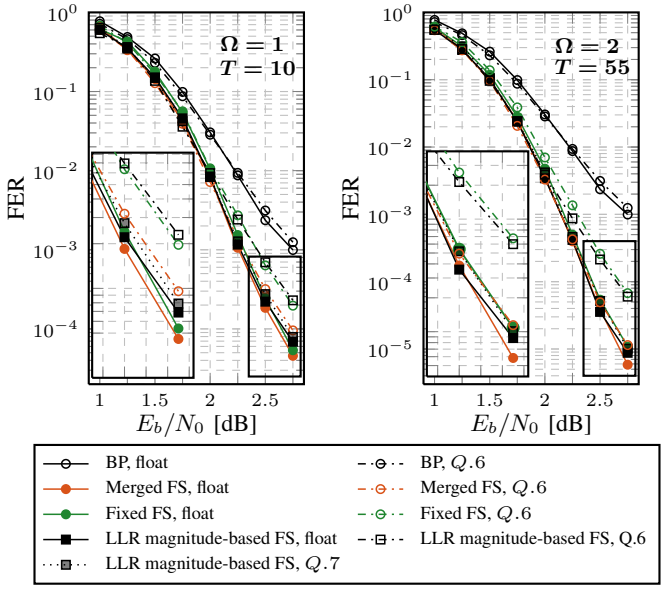

Fig. 13. FER performance of GBPF decoding with LLR-based FS, fixed FS, and merged FS under different quantization schemes for $(1024, 512)$ polar codes with 11-bit CRC, where $Q.x$ represents x-bit quantization. For $Q.6$, the value of $\tau = 6$.

As mentioned in [22] and shown in Fig. 13, the GBPF decoding requires 7-bit quantization to keep the floating-point performance. A lower bit quantization makes more LLR values the same, being not be distinguished. For the decoding with the LLR magnitude-based FS and the fixed FS, 6-bit quantization introduces 0.25 dB performance loss at FERs of 10^{-4}, where the last two bits represent the fractional part. Instead, the GBPF-MS decoding utilizes the accurate parity-check information to generate the FS, making its 6-bit quantized performance consistent with 7-bit quantized GBPF decoding performance. Therefore, the GBPF-MS decoder not only avoids the sorting operations, but also requires less area consumption for BP decoding module. In the next section, the hardware architecture will be designed and implemented to illustrate the advantages of GBPF-MS in terms of area efficiency and throughput.

VI. Hardware Architectures and Implementation Results

In this section, the hardware architecture for the GBPF-MS decoder is introduced with latency analysis, and ASIC implementation results are presented based on 40 nm CMOS technology.

A. Hardware Architecture

Fig. 14 shows the top-level architecture of the proposed GBPF-MS decoder. The BP decoder employs an architecture similar to that shown in Fig. 3-a) of [38], so that the critical path includes one MUX, one PE, and one routing module. The size of the internal LLR memory is $(n-1)\times N\times Q$. On the basis of the BP decoder, the GBPF-MS decoder includes additional termination unit and FS generation & bit-flipping unit. The termination unit performs CRC computation as inter-BP termination and parity-check as intra-BP termination. Different from [39], the hard decision is executed for all stages to obtain the UCNs. If the number of UCNs is 0, the intra-BP termination is activated.

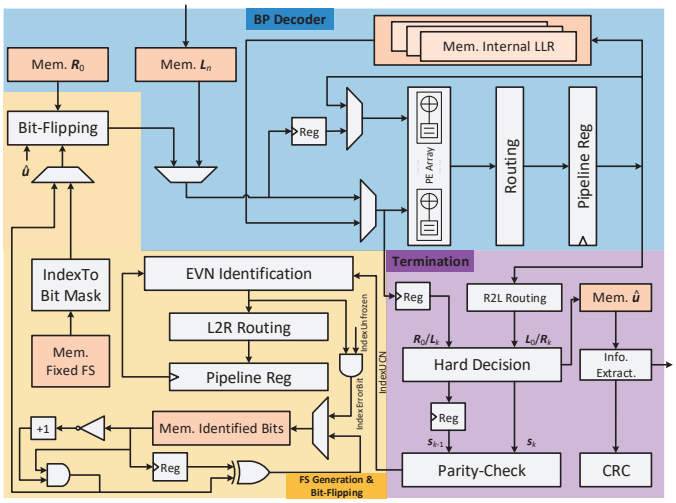

Fig. 14. Overall hardware architecture of proposed GBPF-MS decoder.

After parity-check at stage k, the UCNs of this stage are carried in an N-length binary array IndexUCN and fed into the EVN Identification module. Fig. 15 maps Algorithm 2 to hardware architecture. After the operations of CNIndexTo-FourVNIndex, the labeled three arrays record the indices of VNs at stage $k-1$, k, and $k+1$ on the tracing paths of $\mathcal{L}_{k-1}$, $\mathcal{L}_k$, and $\mathcal{L}_{k+1}$, respectively. We choose to trace these indices to stage n to get their original orders, so that the tracing operations can be pipelined following the calculation of R-messages. First, all VN indices at stage $k+1$ of all tracing paths are obtained by shuffling from left to right and bitwise OR. Then, these indices will shuffle one stage to the right and merge with the indices acquired at the next clock cycle. Therefore, all VN indices are finally traced to the rightmost stage with original orders. Such pipelined design avoids a longer critical path.

The resulting N-length IndexErrorBit array is stored in the memory, containing which bits are error-prone as the w-th flipped bits, based on the corresponding decoding result with order $\omega-1$. Before calculating the R-messages in the next decoding, the flipped bits are sent to the bit-flipping module in the form of a bitmask. In other words, each element in the FS $\mathcal{S}$ generates an N-length binary array, where the indices of flipped bits are labeled as '1'. When $\Omega=1$, the bitmask contains '1' only in the first identified bit. For detected error cases, the generation of the bitmask requires a complement and a bitwise AND. For example, if $N=8$ and the memory of identified bits output an array "10100100" that identifies bits 7, 5, and 2, the complement of this array is "01011100". The result of the bitwise AND is "00000100", which is a one-hot array indexing only bit 2. Then, the bitwise XOR is performed to toggle this bit in the memory, so as the bitmask for the next decoding attempt is "00100000". If all parity-checks are satisfied or all elements in the FS for detected errors have been flipped, the flipped bits are selected from the pre-stored fixed FS. The bitmask generated by each flipped bit during the order-1 decoding is saved, to generate the bitmask of order-2 decoding that contains two '1's.

B. Latency Analysis

Our BP decoder requires $2(n+1)$ clock cycles for each iteration, in which each message-passing direction requires n clock cycles for PEs and an additional clock cycle for accessing $\boldsymbol{L}_1/\boldsymbol{R}_{n-1}$. The termination unit is activated when R-messages are calculated. When the pipelined register outputs $\boldsymbol{R}_k$, the partial sum $\boldsymbol{s}_k$ can be obtained at the same time, and the UCNs at stage $k-1$ are obtained as well. After the pipelined tracing operations, the identified bits are stored in the memory at the last clock cycle of the BP iteration.

In contrast, the LLR sorter in the GBPF decoder can only be activated after the BP decoding is terminated, which occupies additional $7\sim10$ clock cycles for $N=1024$ before the decoding with one bit flipped. For the proposed GBPF-MS decoder, the order-1 decoding can be immediately performed after the original decoding. This is due to that the first $n-1$ clocks do not require $\boldsymbol{R}_0$ message. Namely, the update of $\boldsymbol{R}_0$ by bit-flipping can continue even when the L-messages of the next decoding attempt are being calculated. Once the decoder passes CRC, the BP decoder needs to re-access the received LLR, so there exists an interval of two clock cycles between two frames. Thus, the average latency of the GBPF-MS decoder is $2(n+1)\times I_{\text{ave}}+2$.

C. Implementation Results

Table I shows the synthesis results of the GBPF-MS decoder using TSMC 40 nm technology, where $(1024, 512)$ polar codes are decoded with 11-bit CRC aided, $\Omega=1$ and $T=10, 64$. As comparisons, the results of GBPF [22], fast simplified SCL (Fast-SSCL) [29], fast simplified SCF (Fast-SSCF) [40], [41], and adaptive SCL [42] decoders are provided.

Avoiding LLR sorter, the proposed GBPF-MS decoder is $1.4\times$ faster and $1.7\times$ smaller than the GBPF decoder in [22], thus presenting $2.4\times$ area efficiency. Although the area consumption of GBPF decoder can be reduced by 26.8% as the search range is halved [22], the resulting EBPF decoder [22] is still 42% less area-efficient than the proposed GBPF-MS decoder. Compared to the Fast-SSCL decoder with four lists, the GBPF-MS decoder with $T=10$ shows a similar performance and a 60% throughput improvement at 2.5 dB. The Fast-SSCF decoder in [40] enjoys nearly the same area and average throughput as the SC decoder when $T=20$, which exhibits the highest efficiency but nearly a half throughput and worse error correction performance compared to the GBPF-MS with $T=10$. With a reasonable timing schedule,

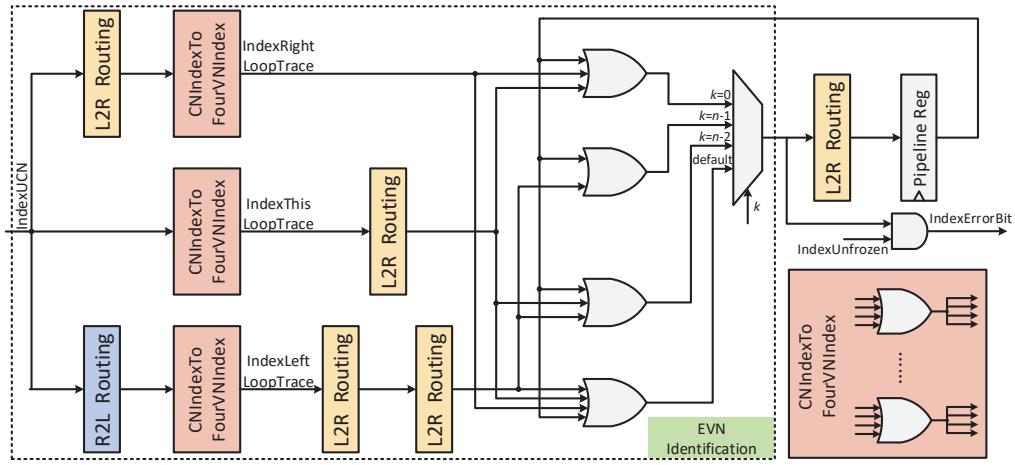

Fig. 15. The module of generating the FS for detected errors.

TABLE I
SYNTHESIS RESULTS OF PROPOSED GBPF-MS DECODER AGAINST THE STATE-OF-THE-ART BP, SCL, ADAPTIVE SCL, AND SCF DECODERS FOR $(1024, 512)$ POLAR CODES.

	This work			[TCAS-II'20] [22]	[TCAS-I'19] [43]	[TSP'17] [29]		[arXiv'19] [42]	[TCAS-I'20] [40]	[ISVLSI'19] [41]
Decoder	**BP**	**GBPF-MS**		**GBPF**	**BP**$^{\diamond}$	**Fast-SSCL**		**TA-SCL**‡	**Fast-SSCF**	**Fast-SSCF**
Methodology	Synthesis			Synthesis	Silicon	Synthesis		Synthesis	Synthesis	Synthesis
Technology [nm]	40	40		65	40	65		90	65	28
LLR Quantization [bit]	6	6		7	—	6		6, 7	6	6
Parameter	—	$T=10$	$T=64$	$T=10$	—	$\mathbb{L}=4$	$\mathbb{L}=8$	$\mathbb{L}_s=2, \mathbb{L}_l=8$	$T=20$	$T=8$
E_b/N_0**@FER=**$\mathbf{10^{-4}}$[dB]	3.6	2.74	2.4	2.71	3.8	2.65	2.4	2.4	2.96	2.86
Ave. Iter./Attempt*	5.06	7.17	11.01	7.21	7.47	—		1	1.01	1.007
Area [mm^2]	0.923	0.946		4.25	0.704	1.822	3.975	7.67	0.56	0.08
Max. Freq. [MHz]	806	**806**		319	500	840	722	595	—	704
Coded T/P†[Gbps]	4.68	4.19	3.97	1.85	4.93	1.61	1.12	3.00	1.50	0.48
Normalized to 40nm *										
Norm. Area [mm^2]	0.923	**0.946**		1.61	0.704	0.69	1.51	1.52	0.212	0.163
Norm. Coded T/P† [Gbps]	4.68	**4.19**	**3.97**	3.00	4.93	2.61	1.95	6.75	2.43	0.34
Area Eff.† [Gbps/mm^2]	5.07	**4.43**	**4.19**	1.87	7.00	3.79	1.29	4.46	11.47	2.06

$^{\diamond}$ The BP decoder in [43] employs the bidirectional double-column architecture.
‡ TA-SCL is the two-stage adaptive SCL, where $\mathbb{L}_s$ is the basic list size and $\mathbb{L}_l$ is the Max. list size.
† Average value at 2.5 dB. * Average iteration number for BP and average decoding attempt for SC.
* Normalized to 40 nm technology based on: area $\propto 1/z^2$ and frequency $\propto z$, where z is the scaling factor to 40 nm.

the TA-SCL decoder in [42] achieves the throughput of SCL-2 (6.75 Gbps) and the performance of SCL-8, at the cost of the sum of both SCL-2 and SCL-8 areas. By comparison, with $\Omega = 1$ and $T = 64$, the GBPF-MS decoder achieves the SCL-8 performance with a throughput of 3.97 Gbps and the same area efficiency. Besides, it is $3.25\times$ more area-efficient than the Fast-SSCL-8 decoder [29].

VII. CONCLUSION

In this paper, the higher-order GBPF decoding algorithm is proposed with a generalized strategy of bit-flipping. Moreover, we propose the GBPF-MS decoding that employs both FSs generated for detected errors and undetected errors. The FS built for detected errors utilizes the parity-check information and identify error-prone bits based on UCNs. The fixed FS for undetected errors considers statistical unreliable bits under SC and BP decoding. The proposed GBPF-MS decoder exhibits better FER performance and higher average throughput than the state-of-the-art BPF and SCF decoders. With the same performance at 2.5 dB, the GBPF-MS decoder is twice as fast as the Fast-SSCL decoder with list size 8, and it shows nearly the same area efficiency as the adaptive SCL decoder.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Seyyed Ali Hashemi and Dr. Ahmed Elkelesh for helpful discussions and the anonymous reviewers for their helpful comments.

REFERENCES

[1] E. Arıkan, "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] ——, "Polar codes: A pipelined implementation," in *Proc. Int. Symp. Broad. Commun. (ISBC)*, 2010, pp. 11–14.
[3] *Chairmans notes of agenda item 7.1.5 Channel coding and modulation*, 3GPP TSG RAN WG1 meeting #87, R1-1613710, Nov. 2016.
[4] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.
[5] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *Proc. IEEE Asilomar Conf. on Signal, Syst. and Comput. (Asilomar)*, 2014, pp. 2116–2120.
[6] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, 2012.

[7] 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, Nov. 2016.

[8] A. Balatsoukas-Stimming, P. Giard, and A. Burg, "Comparison of polar decoders with existing LDPC and Turbo decoders," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2017, pp. 1–6.

[9] *Final Report of 3GPP TSG RAN WG1 #AH1_NR v1.0.0*, 3GPP TSG RAN WG1 meeting #88, R1-1701553, Feb. 2017.

[10] S. M. Abbas, Y. Fan, J. Chen, and C.-Y. Tsui, "High-throughput and energy-efficient belief propagation polar code decoder," *IEEE Trans. VLSI Syst.*, vol. 25, no. 3, pp. 1098–1111, 2016.

[11] N. Yang, S. Jing, A. Yu, X. Liang, Z. Zhang, X. You, and C. Zhang, "Reconfigurable decoder for LDPC and polar codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–5.

[12] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2009, pp. 1488–1492.

[13] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation decoding of polar codes on permuted factor graphs," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2018, pp. 1–6.

[14] N. Doan, S. A. Hashemi, M. Mondelli, and W. J. Gross, "On the decoding of polar codes on permuted factor graphs," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.

[15] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1536–1539, 2018.

[16] Y. Ren, Y. Shen, Z. Zhang, X. You, and C. Zhang, "Efficient belief propagation polar decoder with loop simplification based factor graphs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5657–5660, 2020.

[17] G. Marvin, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "CRC-aided belief propagation list decoding of polar codes," *arXiv preprint:2001.05303*, Jan. 2020.

[18] H. Pishro-Nik and F. Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 50, no. 3, pp. 439–454, 2004.

[19] A. Eslami and H. Pishro-Nik, "On bit error rate performance of polar codes in finite regime," in *Proc. IEEE Annual Allerton conf. commun. contr. comput. (Allerton)*, 2010, pp. 188–194.

[20] A. Elkelesh, S. Cammerer, M. Ebada, and S. ten Brink, "Mitigating clipping effects on error floors under belief propagation decoding of polar codes," in *Proc. IEEE Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2017, pp. 384–389.

[21] Y. Yu, Z. Pan, N. Liu, and X. You, "Belief propagation bit-flip decoder for polar codes," *IEEE Access*, vol. 7, pp. 10 937–10 946, 2019.

[22] Y. Shen, W. Song, Y. Ren, H. Ji, X. You, and C. Zhang, "Enhanced belief propagation decoder for 5G polar codes with bit-flipping," *IEEE Trans. Circuits Syst. II*, vol. 67, no. 5, pp. 901–905, 2020.

[23] C. Condo, F. Ercan, and W. J. Gross, "Improved successive cancellation flip decoding of polar codes based on error distribution," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2018, pp. 19–24.

[24] L. Chandesris, V. Savin, and D. Declercq, "An improved SCFlip decoder for polar codes," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2016, pp. 1–6.

[25] Z. Zhang, K. Qin, L. Zhang, H. Zhang, and G. T. Chen, "Progressive bit-flipping decoding of polar codes over layered critical sets," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2017, pp. 1–6.

[26] F. Ercan, C. Condo, and W. J. Gross, "Improved bit-flipping algorithm for successive cancellation decoding of polar codes," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 61–72, 2019.

[27] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-SCFlip decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2333–2345, 2018.

[28] E. Cavus and B. Daneshrad, "A performance improvement and error floor avoidance technique for belief propagation decoding of LDPC codes," in *Proc. IEEE Int. Symp. Personal Indoor Mobile Radio Commun. (PIMRC)*, vol. 4, 2005, pp. 2386–2390.

[29] 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.

[30] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 520–548, 2001.

[31] J. Chen and M. P. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, 2002.

[32] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Decoder-tailored polar code design using the genetic algorithm," *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 4521–4534, 2019.

[33] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[34] S. Sun, S.-G. Cho, and Z. Zhang, "Error patterns in belief propagation decoding of polar codes and their mitigation methods," in *Proc. IEEE Asilomar Conf. on Signal, Syst. and Comput. (Asilomar)*, 2016, pp. 1199–1203.

[35] T. Richardson, "Error floors of LDPC codes," in *Proc. annual Allerton conf. commun. control and computing*, vol. 41, no. 3, 2003, pp. 1426–1435.

[36] S. Landner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. IEEE Int. Conf. Wireless Netw. Commun. Mobile Comput. (WiCOM)*, vol. 1, 2005, pp. 630–635.

[37] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221–3227, Nov. 2012.

[38] S. Sun and Z. Zhang, "Architecture and optimization of high-throughput belief propagation decoding of polar codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2016, pp. 165–168.

[39] B. Yuan and K. K. Parhi, "Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders," *IEEE Trans. Signal Process.*, vol. 62, no. 24, pp. 6496–6506, 2014.

[40] F. Ercan, T. Tonnellier, and W. J. Gross, "Energy-efficient hardware architectures for fast polar decoders," *IEEE Trans. Circuits Syst. I*, vol. 67, no. 1, pp. 322–335, 2020.

[41] J. Zeng, Y. Zhou, J. Lin, and Z. Wang, "Hardware implementation of improved Fast-SSC-flip decoder for polar codes," in *Proc. IEEE Computer Society Annual Symp. VLSI (ISVLSI)*, 2019, pp. 580–585.

[42] C. Xia, Y. Fan, and C. Tsui, "High throughput polar decoding using two-staged adaptive successive cancellation list decoding," *arXiv preprint arXiv:1905.09120*, May. 2019.

[43] Y. Chen, W. Sun, C. Cheng, T. Tsai, Y. Ueng, and C. Yang, "An integrated message-passing detector and decoder for polar-coded massive MU-MIMO systems," *IEEE Trans. Circuits Syst. I*, vol. 66, no. 3, pp. 1205–1218, 2019.

Yifei Shen (S'16) was born in 1997. He received the B.S. degree from the Chien-Shiung Wu College (Honors College) and the M.S. degree from the School of Information Science and Engineering, Southeast University, Nanjing, China, in 2016 and 2018, respectively. He is now pursuing the Ph.D. degree in Southeast University. His research interests include error-correction codes, VLSI design for digital signal processing, and synthetic biology.

He was the recipient of the Best Student Paper Award at the 2016 IEEE International Conference on Digital Signal Processing and the 2020 IEEE Circuits and Systems Society Pre-Doctoral Scholarship Award.

Wenqing Song (S'16) was born in 2000. She received the B.S. degree from the Chien-Shiung Wu College (Honors College), Southeast University (SEU), Nanjing, China, in 2017. She is currently working toward the Ph.D. degree in the School of Electronic Science and Engineering at Nanjing University (NJU), Nanjing. Her current research interests include error-correction codes, reconfigurable computing and efficient VLSI design.

Houren Ji was born in 1995. He received the B.S. degree from the College of Electronics, Communication and Physics, Shandong University of Science and Technology, Qingdao, China, in 2018. He is studying for the M.S. degree at the School of Information Science and Engineering at Southeast University (SEU). His research interests include decoding algorithm for error-correction codes and VLSI design for digital signal processing.

Yuqing Ren (S'19) was born in 1996. He received the B.S. degree from the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China, in 2018. He is now pursing the M.E. degree in electronics and communication engineering at the School of Information Science and Engineering, Southeast University (SEU), Nanjing, China. His research interests include error-correcting codes and efficient hardware implementation.

Chao Ji received the B.S. degree from the School of Information Science and Engineering, Southeast University, Nanjing, China, in 2018, where he is currently working toward the Ph.D. degree in information and communication engineering at the National Mobile Communications Research Laboratory. His current research interest includes very large scale integration (VLSI) design for digital signal processing and digital communication.

Xiaohu You (F'12) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Nanjing Institute of Technology, Nanjing, China, in 1982, 1985, and 1989, respectively. From 1987 to 1989, he was a Lecturer with the Nanjing Institute of Technology. Since 1990, he has been with Southeast University, first as an Associate Professor and then as a Professor. His research interests include mobile communications, adaptive signal processing, and artificial neural networks with applications to communications and biomedical engineering. He contributed over 40 IEEE journal papers and two books in the areas of adaptive signal processing and neural networks and their applications to communication systems. He was the Premier Foundation Investigator of the China National Science Foundation. From 1999 to 2002, he was the Principal Expert of the C3G Project, responsible for organizing China's 3G mobile communications research and development activities. From 2001 to 2006, he was the Principal Expert of the National 863 FuTURE Project. He received the Excellent Paper Award from the China Institute of Communications in 1987 and the Elite Outstanding Young Teacher Award from Southeast University in 1990, 1991, and 1993. He is currently the Chairman of the IEEE Nanjing Section. He was selected as IEEE Fellow in 2012 for his contributions to the development of mobile communications in China.

Chuan Zhang (S'07-M'13) received the B.E. degree in microelectronics and the M.E. degree in very-large scale integration (VLSI) design from Nanjing University, Nanjing, China, in 2006 and 2009, respectively, and the M.S.E.E. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities (UMN), USA, in 2012.

He is currently the Excellence Professor and the Purple Mountain Professor with Southeast University. He is also with the LEADS, National Mobile Communications Research Laboratory, Quantum Information Center of Southeast University, and the Purple Mountain Laboratories, Nanjing, China. His current research interests include low-power high-speed VLSI design for digital signal processing and digital communication, bio-chemical computation and neuromorphic engineering, and quantum communication.

Dr. Zhang serves as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS. He serves as a Corresponding Guest Editor for the IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS twice. He is also the Secretary-Elect of the Circuits and Systems for Communications TC of the IEEE Circuits and Systems Society. He is also a member of the Seasonal School of Signal Processing and Design and Implementation of Signal Processing Systems TC of the IEEE Signal Processing Society, and Circuits and Systems for Communications TC, VLSI Systems and Applications TC, and Digital Signal Processing TC of the IEEE Circuits and Systems Society. He received the Best Contribution Award of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS) in 2018, the Best Paper Award in 2016, the Best (Student) Paper Award of the IEEE International Conference on DSP in 2016, three Best (Student) Paper Awards of the IEEE International Conference on ASIC in 2015, 2017, and 2019, the Best Paper Award Nomination of the IEEE Workshop on Signal Processing Systems in 2015, three Excellent Paper Awards and two Excellent Poster Presentation Awards of the International Collaboration Symposium on Information Production and Systems from 2016 to 2018, the Outstanding Achievement Award of the Intel Collaborative Research Institute in 2018, and the Merit (Student) Paper Award of the IEEE APCCAS in 2008. He also received the Three-Year University-Wide Graduate School Fellowship of UMN and the Doctoral Dissertation Fellowship of UMN.