Dynamic Multi-Symbol Flipping Decoding of Non-Binary LDPC Codes

XIUWEN BAO\textsuperscript{1}, MING JIANG\textsuperscript{1,2} (Member, IEEE), DONGLI FAN\textsuperscript{1}, AND MINGYANG ZHU\textsuperscript{1} (Graduate Student Member, IEEE)

\textsuperscript{1}National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China
\textsuperscript{2}Purple Mountain Laboratories, Nanjing 211111, China

CORRESPONDING AUTHOR: M. JIANG (e-mail: jiang_ming@seu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1807200, and in part by the National Natural Science Foundation of China under Grant 61771133.

ABSTRACT A dynamic multi-symbol (DM) flipping scheme is proposed for various symbol flipping decodings of non-binary low-density parity-check (NB-LDPC) codes. This new approach divides the whole decoding process into multiple stages according to the number of iterations and allows a different maximum number of symbols to be flipped in each stage. Numerical analysis reveals that the proposed multi-symbol flipping scheme can yield a higher probability of correct flipping with the dynamic flipping threshold. The proposed DM scheme can be highly parallelized, extensively improving the decoder throughput over existing symbol flipping decoding algorithms based on prediction (SFDP). Numerical results show that the DM scheme can significantly enhance the error correction capability and achieve faster convergence speed with little increase in average computational complexity.

INDEX TERMS NB-LDPC codes, multi-symbol flipping decoding, hard reliability.

I. INTRODUCTION

LOW-DENSITY parity-check (LDPC) codes are now widely employed in broadcasting (DVB and ATSC) and wireless communications systems (5G NR and WiFi) because of their strong error correction capabilities. The soft-information-assisted algorithms for the binary LDPC codes such as the traditional belief propagation (BP) decoding algorithm and the min-sum (MS) decoding algorithm in [1] are widely applied to storage applications because of their excellent error-correcting performance. In [2], the bit-serial circuits are proposed to determine the minimum and sub-minimum values and the hardware cost is affordable for the bit-serial check-node architectures. To further reduce the complexity and fully utilize the decoder hardware, the MS-based decoding algorithms [3] adopt the simplified check-node unit to perform the row operation. Comprehensive studies have been conducted to remove computationally intensive hyperbolic computations from the traditional BP decoding algorithm in the hardware-efficient decoder [4]. In this context, the LDPC codes are hardware-friendly for high-throughput realization [5], [6].

Non-binary coding schemes have attracted considerable attention of many researchers because of a larger coding gain [7], [8]. Regular non-binary LDPC (NB-LDPC) codes show superior performance in the deeper error floor region and outperform irregular binary LDPC codes with short-to-moderate blocklength. They have been adopted in the BeiDou navigation satellite and positioning system applications (BDS) [9] and are further considered for the flash storage systems [10] and the optical fiber communications [11], [12]. NB-LDPC codes mainly have three classes of decoding algorithms: message-passing (MP) based algorithms [13], [14], [15], majority-logic decoding (MLgD) algorithms [16], [17] and symbol flipping decoding (SFD) algorithms [18], [19], [20]. An MP-based algorithm has a computational complexity dominated by the order of Galois field $GF(q)$ and the row weight of check-node, which is significantly higher as the $q$ value and code rate increase. The simplified MP decoding [14] based on the fast Fourier transform (FFT) reduces the computational complexity of check-node update from $O(q^2)$ to $O(q \cdot \log_2 q)$. The extended min-sum (EMS) algorithm [21] further reduces...
the computational burden through using only $n_m$ most reliable values for the check-node updating, and the complexity is successfully decreased to $O(n_m \log_2 n_m)$, with $n_m \ll q$. Although a larger degree of hardware parallelization to lower the decoding latency is considered in the trellis-EMS (T-EMS) algorithm [22], the large latency of soft output updating is unsuitable for high-speed applications, such as the read/write in NAND flash memory.

In the case of NB-LDPC codes, the SFD algorithms, particularly the hard-decision ones, are very attractive for flash storage systems employing the NB-LDPC codes since they can offer a reasonable trade-off between performance and complexity. Those symbol flipping algorithms can enjoy lower complexity than the bit-reliability based algorithms such as the weighted bit-reliability based algorithm [23] and the full bit-reliability based algorithm [24]. Recently, many SFD algorithms with the prediction mechanism (SFDP) have already shown excellent performance, e.g., the SFDP algorithms aided by plurality logic (P-SFDP) and the SFDP algorithms aided by binary Hamming distance (D-SFDP) [18], which are further improved with self-adjustment schedule (D-SA-SFDP) [19] and the randomly penalized flipping metrics (N-D-SFDP) [20]. All these SFDP algorithms flip only one symbol for each iteration with a serial schedule resulting in high decoding latency.

The multiple bit-flipping (BF) algorithms for binary codes usually flip all the bits if their reliabilities fall below a designated threshold. The multi-bit-noise-aided gradient-descent BF algorithm (M-NGDBF) in [25] flips every bit satisfying the inversion threshold. Similarly to the multi-flipping operation for the binary code, the M-P-SFDP algorithm for NB-LDPC codes in [26] flips all the symbols satisfying the predetermined threshold. The parallel flipping operations allow for faster convergence but suffer from severe performance degradation due to over-flipping. Two multi-symbol flipping decoding algorithms aiming at reducing the number of iterations are also proposed in [26]. The F-P-SFDP algorithm flips a fixed number of symbols and the A-P-SFDP algorithm adaptively flips all the symbols satisfying the fixed flipping threshold. The F-P-SFDP and A-P-SFDP algorithms accelerate the convergence speed at the expense of more comparison operations for sorting, which requires the global sort operation to determine which symbols to flip. Meanwhile, there is no significant improvement in error correction performance because the relaxed thresholds lead to performance degradation caused by over-flipping.

To better balance the decoding delay and error correction capability, we propose a dynamic multi-symbol (DM) flipping scheme applied for the existing x-SFDP (x: P, D, D-SA, N-D) algorithms, which dynamically flips multiple symbols during the iterative decoding. The DM scheme adopts the dynamic flipping threshold as well as the limitation on the maximum number of flipped symbols which significantly accelerates the convergence speed and lowers the decoding delay. In addition, in order to avoid excessive flipping, the maximum number of flipped symbols is limited, thereby greatly improving the BER performance under the same iterations. For instance, for a (96, 48) regular NB-LDPC code with column weight $\gamma = 2$ and row weight $\rho = 4$ over GF(64), the proposed DM-D-SFDP and DM-D-SA-SFDP algorithms can achieve the performance gain of approximately 0.5 dB and 0.6 dB, respectively, at a bit error rate (BER) of $10^{-7}$. The average number of iterations of the proposed DM scheme is decreased to less than half compared with the existing algorithm. Moreover, our proposed DM-P-SFDP algorithm outperforms the existing multi-symbol flipping algorithms in BER performance. For a (204, 102) regular NB-LDPC code over GF(16) at the BER of $10^{-6}$, the proposed DM-P-SFDP algorithm achieves about 0.3 dB and 0.25 dB gain over the A-P-SFDP and the F-P-SFDP algorithm, respectively.

The remainder of the paper is organized as follows. In Section II, we give the concepts and notation of NB-LDPC codes, and then review the existing single symbol flipping algorithms in [18], [19], [20]. In Section III, we propose the dynamic multi-symbol flipping scheme and provide a detailed description of the parameter selection. In Section IV, the comparisons with the existing multi-symbol flipping algorithms in [26] are also presented to provide a benchmark for our proposed DM algorithms. In Section V, experimental evaluations and performance analysis are used to demonstrate the improvement of the error correction performance and the convergence rate of our proposed DM scheme. Finally, the concluding statements are given in Section VI.

II. PRELIMINARIES

Suppose a regular $(\gamma, \rho)$ NB-LDPC code over GF($q$), $q = 2^{d}$ is used for error correction in an additive white Gaussian noise channel with zero mean and variance $\sigma^2$. The $M \times N$ regular parity-check matrix of the NB-LDPC code with column weight $\gamma$ and row weight $\rho$ is $H = [h_{m,n}]_{M \times N}$, $h_{m,n} \in$ GF($q$). If $h_{m,n}$ is nonzero, the $m$-th check node (CN) and the $n$-th variable node (VN) are connected with each other in the Tanner graph. The hard-decision symbol vector $z^{(1)} = \{z_{1}^{(1)}, \ldots, z_{N}^{(1)}\}$ is obtained by the received signals, where each symbol has a binary image representation $z_{j}^{(1)} = (z_{j,1}^{(1)}, \ldots, z_{j,d}^{(1)})$, $z_{j,l} \in \{0, 1\}$, for $j = 1, \ldots, N$, $l = 1, \ldots, d$.

In the k-th iteration, the syndrome vector $s^{(k)} = z^{(k)} \ast H^{T}$ is firstly calculated to determine whether the current hard-decision vector $z^{(k)}$ satisfies all the parity-check constraints. If the parity check fails, the hard-decision symbol $z_{j}^{(k)}$ is passed from the j-th VN to its adjacent CNs. Then, the extrinsic information-sum (EXI) passed from the i-th CN to the j-th VN is denoted as

$$\sigma_{ij}^{(k)} = h_{ij}^{-1} \sum_{f \in \mathcal{N}_{i} \setminus j} h_{ij} f z_{j}^{(k)},$$

where $\mathcal{N}_{i} = \{j : 1 \leq j \leq N, h_{ij} \neq 0\}$. When the j-th symbol is selected to be flipped in this iteration, the $q - 1$ possible
flipped values can be represented by
\[ \Gamma_j^{(k)} = \left\{ \tilde{z}_j^{(k)} : z_j^{(k)} \in \mathrm{GF}(q), \tilde{z}_j^{(k)} \neq z_j^{(k)} \right\}. \] (2)

The P-SFDP algorithm combines the plurality logic to the flipping metric
\[ E_j^{(k)}(z_j^{(k)}, \tilde{z}_j^{(k)})_P = \tilde{z}_j^{(k)} \odot y_j + n(z_j^{(k)}) - z_j^{(k)} \odot y_j - n(z_j^{(k)}), \] (3)
where \( n(z_j^{(k)}) \) represents a fixed weighting factor and the number of times \( z_j^{(k)} \) occurs among the EXIs passed from all CNs connected to the \( j \)-th VN. \( \eta \) is the weighting factor.

The D-SFDP algorithm introduces the reliability of the binary Hamming distance \( d(z_j^{(k)}, \tilde{z}_j^{(k)}) \) into the flipping metric
\[ E_j^{(k)}(z_j^{(k)}, \tilde{z}_j^{(k)})_D = \tilde{z}_j^{(k)} \odot y_j + \sum_{i \in \mathcal{M}_j} \theta_d(z_j^{(k)}, \sigma_j^{(k)}) - z_j^{(k)} \odot y_j - \sum_{i \in \mathcal{M}_j} \theta_d(z_j^{(k)}, \sigma_j^{(k)}) \] (4)
where \( \mathcal{M}_j = \{ i : 1 \leq i \leq M, h_{ij} \neq 0 \} \), \( \theta = [\theta_0, \theta_1, \ldots, \theta_d] \) represents weighting factors.

However, the SFDP algorithms suffer from the problem that decoding may be trapped into a local maximum as a class of gradient descent algorithms. When the decoding process is trapped into a local maximum, the flipped VNs may drop into an oscillation. The cycle oscillation always leads to a decoding failure, where several VNs are changed circularly in subsequent iterations. In order to break the cycle oscillation in the P/D-SFDP algorithms, the flipping metric was modified with the self-adjustment strategy and the random penalty items introduced in [19], [20]. We use the D-SFDP algorithm to illustrate the improvements of the flipping metric in detail.

The D-SA-SFDP algorithm considers the statistics of the symbol flipping and develops a new flipping metric with the help of a self-adjustment schedule
\[ E_j^{(k)}(z_j^{(k)}, \tilde{z}_j^{(k)})_{D-SA} = E_j^{(k)}(z_j^{(k)}, \tilde{z}_j^{(k)})_D - \sigma_f \cdot t_j^{(k)}, \] (5)
where \( \sigma_f \) and \( t_j^{(k)} \) represent a fixed weighting factor and the number of times that the selected symbol is flipped to \( \tilde{z}_j^{(k)} \) during the first \( k-1 \) iterations, respectively.

Moreover, the N-D-SFDP algorithm uses the random penalty items to facilitate the D-SFDP algorithm escaping from the undesirable local optimum. The modified flipping metric with random penalty items can be calculated by
\[ E_j^{(k)}(z_j^{(k)}, \tilde{z}_j^{(k)})_{N-D} = \tilde{z}_j^{(k)} \odot y_j + \sum_{i \in \mathcal{M}_j} (g_{i,j}^{(k)} - s_{i,j}^{(k)}) + \sum_{i \in \mathcal{M}_j} \theta_d(z_j^{(k)}, \sigma_j^{(k)}), \] (6)
where \( s_{i,j}^{(k)} \) and \( g_{i,j}^{(k)} \) denote the random variables to penalize the weighted hard reliabilities of \( \tilde{z}_j^{(k)} \) and \( z_j^{(k)} \), respectively.

Those aforementioned \( x \)-SFDP algorithms prevent flipping the same symbol in two consecutive iterations to avoid oscillation in a close region of the local maximum. The maximum value of the \( q-1 \) flipping metrics of the \( j \)-th symbol is defined as the flipping function
\[ E_j^{(k)}(z_j^{(k)}, \tilde{z}_j^{(k)}) \]
and its related flipping value is
\[ \mu_j^{(k)} = \arg \max_{\tilde{z}_j^{(k)} \in I_j^{(k)}} E_j^{(k)}(\tilde{z}_j^{(k)}, z_j^{(k)}). \] (8)

### III. THE PROPOSED DM SCHEME

Our proposed DM scheme is superior to existing multi-symbol flipping algorithms. Different from the fixed flipping threshold in the A/F-P-SFDP algorithms in [26], our proposed DM scheme adopts the dynamic flipping threshold and effectively limits the probability of a relaxed threshold that brings about over-flipping. We dynamically adjust the maximal number of flipped symbols under explicit numerical limits to accelerate the convergence speed.

In what follows, \( \Delta, E^{(k)} = \{ E_1^{(k)}, \ldots, E_N^{(k)} \}, \mu^{(k)} = \{ \mu_1^{(k)}, \ldots, \mu_N^{(k)} \} \) and \( f^{(k)} = \{ f_1^{(k)}, \ldots, f_N^{(k)} \} \) denote the threshold adjustment factor, the vector of flipping functions, the vector of symbol flipping values and the indicator vector of the flipped positions to record the flipped positions in the previous iteration, respectively. For the avoidance of cycle oscillation, we prevent the symbols from flipping in two consecutive iterations. Specifically, \( f_j^{(k)} = 1, j = 1, \ldots, N \) indicates the \( j \)-th symbol that has already been flipped in the \((k-1)\)-th iteration, which is unable to be flipped in the \( k \)-th iteration. We set the maximum number of iterations to \( k_T \) and divide the entire decoding process into \( T \) stages. In the \( t \)-th, \( 1 \leq t \leq T \) stage, up to \( w_t \) symbols can be flipped during the iteration period \( \{k_t, k_t\} \). The relation between the division of the decoding stages and the maximum numbers of flipped symbols can be represented as
\[ \{ (0, k_1), (k_1, k_2), \ldots, (k_{T-1}, k_T) \} \iff \{ w_1, w_2, \ldots, w_T \}. \] (9)

Here, we have to determine the endpoints of \( T \) iteration stages \( k_1, k_2, \ldots, k_T \) and the maximum numbers of the symbols to be flipped in each stage \( w_1, w_2, \ldots, w_T \). We find that the DM scheme with \( T = 3 \) stages can provide a noticeable performance improvement over the original single symbol flipping algorithms and a more complex division of decoding stages is unnecessary. More specifically, \( w_1 \) is related to the code length and is usually set to a relatively large value (e.g., 6, 10, 15). \( w_2 \) and \( w_3 \) are limited to small values to avoid over-flipping, which are usually set to 2 and 1, respectively. \( k_T \) equals the maximum number of iterations and we also take \( T = 3 \) stages as an example. We firstly set \( k_3 = 100 \) then optimize \( k_1 \) and \( k_2 \) with experimental test. We provide some empirical reference values of \( k_1, k_2, k_3 \) and
w_1, w_2, w_3 in Section V for different codes which are finally determined through simulation.

The dynamic threshold value plays a key role in accelerating the convergence speed and reducing the decoding delay. The predetermined threshold adjustment factor $\Delta$ can only be found by simulations and is directly correlated to the signal-to-noise ratio, the degree distribution, and the decoding algorithm used. If we apply the threshold adjustment factor with the second largest flipping function value, the parameter $\Delta$ can be set to a constant number which is concluded empirically. Different optimized values of $\Delta$ should be set (e.g., 1, 2, 3) when we apply it to the threshold $E_\alpha - \Delta$, where $E_\alpha$ represents the maximum value of $E$. The $\Delta$ is typically set to 1 in $E_\beta - \Delta$ that is empirically shown to provide good performance for typical codes thus the dynamic threshold $E_\beta - \Delta$ exhibits more robustness, where $E_\beta$ represents the second maximum value of $E$. The threshold adjustment factor applied to the largest or the second largest flipping functions has similar performance in terms of decoding simulations. The only difference is the value of the predefined threshold adjustment factor. In summary, the threshold adjustment factor is applied to the second largest flipping function $E_\beta$ other than the largest one $E_\alpha$ in the following simulations for its easy implementation.

In the $k$-th iteration, we first determine the decoding stage satisfying $k \in (k_{-1}, k_1]$. Then, we select two maximum flipping functions $E_\alpha^{(k)}$ and $E_\beta^{(k)}$ and the corresponding candidate flipping values $\mu_\alpha^{(k)}$ and $\mu_\beta^{(k)}$ for $\alpha, \beta \in [1, N]$, respectively. After that, we try to flip the $\alpha$-th symbol with the maximum value of the flipping function $E_\alpha^{(k)}$. If this symbol was unflipped in the last iteration and $z_\alpha^{(k)} \neq \mu_\alpha^{(k)}$, we flip this symbol to $z_\alpha^{(k+1)} = \mu_\alpha^{(k)}$, update the $j_{\alpha_{k+1}} = 1$ and accumulate the number of flipped symbols $w_t$. In the same way, if the predefined number $w_t$ is larger than $w$, $f_\beta^{(k)} = 0$ and $z_\beta^{(k)} \neq \mu_\beta^{(k)}$, we will flip the second candidate symbol $z_\beta^{(k+1)} = \mu_\beta^{(k)}$ and update $f_\beta^{(k+1)} = 1$. Lastly, all symbols satisfying the flipping criterion $E_j^{(k)} \geq E_j^{(k)} - \Delta, j \in [1, N], j \neq \alpha, \beta$ are also flipped until at most $w_t$ symbols are flipped. We update the symbol sequence $z^{(k+1)}$ and then set $f_\beta^{(k+1)} = 0$ if the $i$-th symbol is unflipped.

Multiple symbols are selected to be flipped in the DM-x-SFDP algorithms according to the flipping functions calculated by (7). We first set the iteration counter to $k = 1$, initialize the indicator $f^{(1)} = 0_N$ and calculate all $M$ syndromes with $s^{(k)} = z^{(k)} \ast H^T$. If $s^{(k)} = 0_M$, we terminate the decoding process and output the current hard-decision sequence $z^{(k)}$. Otherwise, we calculate $\gamma$ EXIs and the $q - 1$ flipping metrics of each symbol by (1) and the objection function. Then, we get the flipping function values $E_\alpha^{(k+1)}, \ldots, E_N^{(k+1)}$ and the corresponding symbol flipping values $\mu_\alpha^{(k+1)}, \ldots, \mu_N^{(k+1)}$ of $N$ symbols by (7) and (8), respectively. Next, we flip multiple symbols with the DM scheme elaborated in Algorithm 1. The inputs are the iteration counter $k$, the vector of flipping functions vector of symbol flipping values $\mu^{(k)}$ and the indicator vector $f^{(k)}$. Finally, we update the hard-decision symbol vector $z^{(k)}$, accumulate the iteration counter and repeat the above operations until the iteration counter reaches $k_T$. The proposed DM scheme can be adapted to all existing x-SFDP algorithms in Section II.

The detailed decoding process of the proposed DM-x-SFDP algorithm is shown in Algorithm 2.

We consider two flipping rules. The first flipping rule prevents the variable nodes flipped in the previous iteration from being flipped in the current iteration, while the second flipping rule does not take into account whether the variable nodes have been flipped in the previous iteration. Specifically, we consider a (204, 102) NB-LDPC code $C_1$ over GF(16). The values of $E_\alpha$ versus the number of iterations with those two flipping rules under the D-SFDP algorithm and our proposed DM-D-SFDP algorithm for $C_1$ at $E_b/N_0 = 4.5$ dB are exhibited in Fig. 1, where the maximum number
Algorithm 2: The Proposed DM-x-SFDP Algorithms

Parameter
- $T$: the division of the iterative decoding interval;
- $k$: the iteration counter;
- $k_T$: the maximum number of iterations;

Input
- $k_1, k_2, \ldots, k_T$: the iteration interval;
- $w_1, w_2, \ldots, w_T$: symbols to flip in each interval;
- $\mathbf{z}^{(1)} = [z_1^{(1)}, \ldots, z_N^{(1)}]$: the hard-decision vector;
- $\mathbf{f}^{(k)} = 0_N$: the indicator vector of the flipped positions;

for $(k = 1, \ldots, k_T)$ do
- compute $\mathbf{s}^{(k)} = \mathbf{z}^{(k)} \ast \mathbf{H}^T$
- if $\mathbf{s}^{(k)} = 0_H$ break
- update $\sigma_i^{(k)}$ and $E_j^{(k)}(z_j^{(k)}, \bar{z}_j^{(k)})$ by (1) - (6)
- compute $\mathbf{E}^{(k)}$ by (7) and $\mathbf{\mu}^{(k)}$ by (8)
- $\mathbf{z}^{(k)} = \text{Multi-Symbol Flip} \left( k, \mathbf{E}^{(k)}, \mathbf{\mu}^{(k)}, \mathbf{f}^{(k)} \right)$
- set $k = k + 1$

output $\mathbf{z}^{(k)}$

IV. COMPARISON WITH EXISTING MULTI-SYMBOL FLIPPING ALGORITHMS

A. PERFORMANCE COMPARISONS

We compare the P-SFDP algorithm applied our proposed DM scheme (i.e., the DM-P-SFDP algorithm) with existing multi-symbol flipping algorithms in [26]. The DM scheme can also be applied to other x-SFDP ($x$: P/D, D-SA, D-N) algorithms and the detailed simulation is provided in the following section.

The BER performance comparisons of the original P-SFDP algorithm, our proposed DM-P-SFDP algorithm and the existing M/F/A-P-SFDP algorithm are provided in Fig. 2. It can be seen that the DM scheme greatly improves the BER performance over the existing multi-symbol flipping algorithms under the maximum number of iterations of 100. The average number of iterations are presented in Fig. 3. Although the A-P-SFDP algorithm performs well in terms of convergence speed but the error performance is not
TABLE 1. The comparisons for different multi-symbol flipping algorithms.

| Algorithm       | * DM-P-SFDP | M-P-SFDP | F-P-SFDP<sub>τ</sub> | A-P-SFDP |
|-----------------|-------------|----------|---------------------|----------|
| Threshold       | Dynamic: <sup>E</sup>j<sup>(k)</sup> > ξ<sub>β</sub> – Δ | Fixed: <sup>E</sup>j<sup>(k)</sup> > 0 | Fixed: <sup>E</sup>j<sup>(k)</sup> > 0 | Fixed: <sup>E</sup>j<sup>(k)</sup> > 0 |
| Flipping rule   | (0, k<sub>1</sub>) ⇐⇒ Less than w<sub>1</sub> | Fixed: M | (0, k<sub>1</sub>) ⇐⇒ Less than T | (0, k<sub>1</sub>) ⇐⇒ No limit |
|                 | (k<sub>1</sub>, k<sub>2</sub>) ⇐⇒ Less than w<sub>2</sub> |                 | (k<sub>1</sub>, k<sub>τ</sub>) ⇐⇒ 1 |                      |
|                 | ...         |          |                     |                      |
|                 | (k<sub>τ</sub>−1, k<sub>τ</sub>) ⇐⇒ Less than w<sub>k</sub> |          |                     |                      |
| Description     | Dynamic threshold | Over-flipping | The relaxed threshold | The relaxed threshold |
| Converge speed  | Faster convergence | Fastest convergence | Slightly faster convergence | Faster convergence |
| BER performance | Significant improvement | Inferior to the P-SFDP | Almost remain the same | Slightly better |

* Our proposed DM-P-SFDP algorithm

TABLE 2. Parameter settings for the multiple symbol flipping algorithm of C<sub>1</sub>.

| Algorithm       | * DM-P-SFDP | M-P-SFDP | F-P-SFDP<sub>τ</sub> | A-P-SFDP |
|-----------------|-------------|----------|---------------------|----------|
| Flipping rule   | (0, 30) ⇐⇒ Less than 15 | Fixed: 2 | (0, 100) ⇐⇒ Fixed 2 | (0, 10) ⇐⇒ Dynamic |
|                 | (30, 40) ⇐⇒ Less than 2 |          | (10, 100) ⇐⇒ 1 | (10, 100) ⇐⇒ 1 |
|                 | (40, 100) ⇐⇒ 1 |          |                     |                      |
| Threshold       | Dynamic: ξ<sub>β</sub> – Δ | Fixed: <sup>E</sup>j<sup>(k)</sup> > 0 | Fixed: <sup>E</sup>j<sup>(k)</sup> > 0 | Fixed: <sup>E</sup>j<sup>(k)</sup> > 0 |

* Our proposed DM-P-SFDP algorithm

competitive when compared to our proposed DM-P-SFDP algorithm. The DM-P-SFDP algorithm achieves about 0.3 dB gain over the A-P-SFDP algorithm at the BER of 10<sup>−5</sup>. Our proposed DM-P-SFDP algorithm achieves more noticeable performance gain than the F-P-SFDP and A-P-SFDP algorithms compared with the original single symbol flipping algorithms.

A detailed comparison between the existing multi-symbol flipping algorithms and our proposed DM-P-SFDP algorithm is summarized in Table 1. The corresponding parameters for those algorithms coefficients of C<sub>1</sub> are provided in Table 2. We provide a rational explanation of why our proposed DM-P-SFDP algorithm is superior to existing multi-symbol flipping algorithms in the next two sections.

B. DYNAMIC FLIPPING THRESHOLD

Table 1 shows that the M-P-SFDP, A-P-SFDP and F-P-SFDP algorithms in [26] apply the fixed flipping rule <sup>E</sup>j<sup>(k)</sup> > 0 which is a loose limit for flipping symbols. Our proposed DM-P-SFDP algorithm applies the dynamic flipping threshold to avoid excessive flipping while the M-P-SFDP, A-P-SFDP and F-P-SFDP algorithms adopt the fixed flipping threshold <sup>E</sup>j<sup>(k)</sup> > 0 which is too relaxed to guarantee the flipping reliability. Although the convergence speed can be accelerated, they suffer a severe BER performance degradation because of excessive flipping in the first few iterations. The relaxed flipping threshold and no limitation on number of flipped symbols will further lower the reliability of
flipping in the iterative decoding thus the A-P-SFDP algorithm suffers from BER performance degradation caused by over-flipping. It can be seen in Fig. 2 that the A-P-SFDP and the F-P-SFDP as multi-symbol flipping algorithms have almost no BER improvement compared with the original single symbol flipping algorithm and the M-P-SFDP algorithm even suffers from BER performance deterioration due to excessive flipping. By contrast, the DM-P-SFDP algorithm achieves more noticeable performance gain than the F-P-SFDP and A-P-SFDP algorithms.

C. LIMITATION ON THE NUMBER OF FLIPPED SYMBOLS
We provide statistical results to show that the proposed DM scheme can be advantageous in terms of BER performance. The proportion of the number of symbols actually flipped in one iteration during the decoding process for $C_1$ is provided. The comparisons between the existing multi-symbol flipping algorithms (F/A-P-SFDP) and our proposed DM-P-SFDP algorithm are shown in Fig. 4. The logarithmic coordinates are adopted in the Y-axis for a fair comparison. The $w$ in the legend of Fig. 4 represents the number of actually flipped symbols in one iteration during the decoding process. The M-P-SFDP algorithm is not considered below due to the significant performance degradation compared with the original single flipping algorithm.

Fig. 4(a) clearly shows that our proposed DM-P-SFDP algorithm applies explicit numerical limitation on the maximum number of flipped symbols in each iteration to avoid excessive flipping. The DM-P-SFDP algorithm has a better convergence speed than the F-P-SFDP algorithm in Fig. 4(b) because the DM-P-SFDP algorithm can effectively flip multiple symbols with an even distribution of the actual number of flipped symbols as depicted in Fig. 4(a). The vertical axis of the Fig. 4(b) adopts the logarithmic coordinates and only one symbol is flipped in most of the cases, which accounts for more than 70% of all iterations. At the same time, the number of symbols of dynamic flipping in our proposed DM scheme is not allowed to exceed the predefined value (e.g., 15), so the occurrence of over-flipping is suppressed. The A-P-SFDP algorithm in [26] flips all the symbols satisfying $E^{(k)} > 0$ without limitation. The number of symbols flipped during a single iteration is even more than 50 which leads to serious over-flipping as seen in Fig. 4(c). The shaded area in Fig. 4(c) highlights the cases where the number of actually flipped symbols is larger than 20, which can lead to severe over-flipping. As the authors clarify [26], the A-P-SFDP algorithm provides effective tradeoffs between convergence speed and computational complexity but appears no improvement in BER performance. The probabilistic results demonstrate that the dynamic flipping threshold combined with the number restriction on flipped symbols in our proposed DM scheme limits the probability of occurrences of over-flipping.

D. COMPUTATIONAL COMPLEXITY COMPARISON
The detailed computational complexity comparisons of the aforementioned multi-symbol flipping algorithms are depicted in Table 3. The parameters for NB-LDPC codes in SFDP algorithms are defined in Table 4. Our proposed DM-P-SFDP algorithm has the same Galois field additions/multiplications (GA/GM) and the integer/real additions (IA/RA) computational complexity as that of the F-P-SFDP and A-P-SFDP algorithms and has $2n - 4$ more integer/real comparisons (IC/RC) operations than that of the A-P-SFDP algorithm. Although the comparison complexity of our proposed DM-P-SFDP algorithm is slightly increased, we effectively avoid over-flipping and significantly improve the BER performance. The F-P-SFDP algorithm in [26] defines
TABLE 3. The computational complexity per iteration of various SFDP algorithms.

| Algorithms          | GM            | GA            | IM/RM         | IA/RA         | IC/RC         |
|---------------------|---------------|---------------|---------------|---------------|---------------|
| P-SFDP in [12]      | $A$           | $2\gamma(\rho - 1)$ | $-\gamma d$  | $(\gamma + d - 1)\tau$ | $(\gamma + d - 1)\tau + 2(n - 1)$ |
| *DM-P-SFDP          | $w_k A$       | $w_k 2\gamma(\rho - 1)$ | $-w_k(\gamma + d + 3\gamma + 1)\tau$ | $w_k(\gamma + d - 1)\tau + 3n - 4$ |
| F-P-SFDP            | $w_k A$       | $w_k 2\gamma(\rho - 1)$ | $-w_k(\gamma + d + 3\gamma + 1)\tau$ | $w_k(\gamma + d - 1)\tau + \Theta_T$ |
| A-P-SFDP            | $w_k A$       | $w_k 2\gamma(\rho - 1)$ | $-w_k(\gamma + d + 3\gamma + 1)\tau$ | $w_k(\gamma + d - 1)\tau + n$ |

* Our proposed DM-P-SFDP algorithm

GA/GM: Galois Field Addition/Multiplication
RA/RC/RM: Real Addition/Comparison/Multiplication
IA/I/IC: Integer Addition/Comparison/Multiplication

$\Lambda = \gamma (2\rho - 1)$, $\tau = (\gamma \rho - \gamma + 1)$

When $|S^{(k)}| \leq T$, $\Theta_T = n + 1$.
When $|S^{(k)}| > T$, $\Theta_T = \left(\mid S^{(k)} \mid - 1 \right) + \left(\mid S^{(k)} \mid - 2 \right) + \cdots + \left(\mid S^{(k)} \mid - T \right) + n + 1$.

FIGURE 5. The FER (dashed)/ BER (solid) performance of different decodings for $C_1$ (left), $C_2$ (middle) and $C_3$ (right).

TABLE 4. Parameters of the NB-LDPC codes for the proposed DM scheme.

| Parameter | Description |
|-----------|-------------|
| $\gamma$  | row weight  |
| $\rho$    | column weight |
| $n$       | The code length |
| $k$       | The information code length |
| $d$       | The vector space of the Galois field |
| $q = 2^d$ | The Galois field size |
| $w_k$     | The actual number of flipped symbols in the k-th iteration |

V. SIMULATION RESULTS AND COMPLEXITY ANALYSIS

The proposed DM scheme can be applied to existing single symbol flipping algorithms and we compare the decoding performance and the convergence rate of different flipping algorithms with or without DM scheme. Three regular NB-LDPC codes are selected for testing, denoted by $C_1$, $C_2$ and $C_3$. a candidate set of flipping symbols with positive flipping metrics in the k-th iteration as $S^{(k)} = \{ j : j \in \mathcal{N}, E_j^{(k)} > 0 \}$. The F-P-SFDP$_T$ algorithm involves too many comparison operations when picking $T$ symbols. If $|S^{(k)}| > T$, the complexity of the total sorting algorithm is as high as $\Theta_T = (|S^{(k)}| - 1) + (|S^{(k)}| - 2) + \cdots + (|S^{(k)}| - T) + n + 1$. Our proposed DM scheme is a low-complexity decoding scheme that avoids the comparison operations of the full sorting for selecting the largest $T$ flipping function values compared with the F-P-SFDP algorithm.

The A-P-SFDP algorithm suffers from over-flipping without number limitation on flipped symbols. Although the F-P-SFDP algorithm imposes a limitation on the number of flipped symbols at the initial stage of the decoding, the high complexity increment of full sorting still limits the decoding delay in hardware implementation. Our proposed DM scheme avoids the high complexity increment and achieves a faster convergence speed as well as significant performance gain with the help of a dynamic flipping threshold and upper limitation of flipped numbers.
threshold adjustment factor $\Delta_1$.

excessive flipping, which is optimized by experimental trials.

functions of the proposed DM scheme in this section are provided in Table 5, where $T = 3$ stages are set for total 100 iterations. The threshold adjustment factor $\Delta$ is designed in consideration of improving the parallelism of flipping as well as avoiding excessive flipping, which is optimized by experimental trials.

A. ERROR CORRECTION PERFORMANCE

Fig. 5 shows the frame error rate (FER) and BER performances of the proposed DM-$x$-SFDP (x: D, D-SA, N-D) algorithms and the original $x$-SFDP algorithms. The FER and BER performance scale similarly for the given NB-LDPC code $C_1$, $C_2$ and $C_3$. It can be seen that the proposed DM-$x$-SFDP (x: D, D-SA, N-D) algorithms perform much better than their corresponding $x$-SFDP algorithms. For example, at a BER of $10^{-6}$, the performance gains of the DM-D-SFDP algorithm for the $C_1$, $C_2$ and $C_3$ are 0.4 dB, 0.15 dB and 0.4 dB, respectively, when compared with the original D-SFDP algorithm. For $C_1$ and $C_2$ under maximum number of iterations $k_3 = 100$, the proposed DM-N-D-SFDP algorithm both can achieve about $0.1 \sim 0.15$ dB gain over the N-D-SFDP algorithm at an FER of $10^{-4}$. While, the DM scheme brings about 0.4 dB gain compared with the D-SFDP and D-SA-SFDP algorithms for $C_3$ at the same FER of $10^{-4}$.

B. CONVERGENCE SPEED

The DM-N-D-SFDP algorithm, which almost always has the best performance, also achieves significantly faster convergence rate for three codes. We define the average number of iterations as $\bar{k}$ and give the statistic results of $\bar{k}$ for the DM-$x$-SFDP (x: D, D-SA, N-D) algorithms in Fig. 6, which drops to less than half of the original $x$-SFDP algorithms, significantly reducing the decoding delay.

The comprehensive simulation results reveal that the application of the DM scheme allows for better FER performance and faster convergence speed. The proposed DM-$x$-SFDP algorithms retain high reliability by dynamically adjusting the number of flipped symbols to improve the decoding reliability and reducing the processing latency.

C. COMPUTATIONAL COMPLEXITY ANALYSIS

Table 6 presents the computational complexity per iteration of various SFDP algorithms. The operation numbers of the Galois field additions/multiplications $\phi_g$, the integer/real additions $\phi_a$ and the integer/real comparisons $\phi_c$, in the $x$-SFDP algorithms are constants, since only one symbol will be flipped. Thus, the average computational complexity of different operations $\Phi_g$, $\Phi_a$ and $\Phi_c$ in the $x$-SFDP algorithms can be directly calculated by multiplying the average iteration number (e.g., $\Phi_g = \bar{k}\phi_g$). The corresponding operation numbers $\hat{\Phi}_g$, $\hat{\Phi}_a$ and $\hat{\Phi}_c$ in the DM-$x$-SFDP algorithms can be represented as the functions of the actual number $w_k$ of flipped symbols in the $k$-th iteration, where $w_k \leq w_1$ in the $t$-th stage, $k \in \{k_{t-1}, k_t\}, t = 1, \ldots, T$. Here, the operation numbers of GA/GM and IA/RA with DM scheme follow the relation, $\hat{\Phi}_g = w_k\phi_g$ and $\hat{\Phi}_a = w_k\phi_a$, where $\phi_g = (\gamma^2 + 2\gamma d + 2\gamma + 4d)(\gamma\rho - \gamma + 1), \phi_a = \gamma(4\rho - 3) + \gamma(\gamma\rho - \gamma + 1)(d + \gamma)$. The IC/RC operations of the $x$-SFDP and the DM-$x$-SFDP algorithms in each iteration are $\phi_c = 2(N - 1) + \delta_c$ and $\hat{\Phi}_c = 3N - 4 + w_k\delta_c$, respectively, where $\delta_c = (\gamma + d - 1)(\gamma\rho + 1)$. We statistic a total of $F$ correctly decoded frames and define the actual iteration number of the DM-$x$-SFDP algorithms of each frame as $k^i, i = 1, \ldots, F$. The average computational complexity of the DM-$x$-SFDP algorithms is statistically analyzed

$$\hat{\Phi}_g + \hat{\Phi}_a + \hat{\Phi}_c = \sum_{i=1}^{F} \sum_{k=1}^{k^i} (w_k(\phi_g + \phi_a + \delta_c) + 3N - 4)/F \quad (10)$$
**TABLE 6.** The computational complexity per iteration of various SFDP algorithms.

| Algorithms         | GM       | GA      | IM/RM | IA/RA    | IC/RC   |
|--------------------|----------|---------|-------|----------|---------|
| D-SFDP in [12]     | $A$      | $2\gamma(p-1)$ + $\tau\gamma(d+\gamma)$ | $\gamma^2 + 2\gamma d + 2\gamma + 4d\tau$ | $\gamma + d - 1\tau + 2(n-1)$ |
| *DM-D-SFDP         | $w_k A$  | $w_k 2\gamma(p-1)$ + $w_k \tau\gamma(d+\gamma)$ | $w_k (\gamma^2 + 2\gamma d + 2\gamma + 4d\tau)$ | $w_k (\gamma + d - 1\tau + 3n - 4)$ |
| D-SA-SFDP in [13]  | $A$      | $2\gamma(p-1)$ + $\tau\gamma(d+\gamma)$ | $\min(k, nq)\gamma^2 + 2\gamma d + 2\gamma + 4d\tau$ + $\min(k, nq) + 1\gamma + d - 1\tau + 2(n-1)$ |
| *DM-D-SA-SFDP      | $w_k A$  | $w_k 2\gamma(p-1)$ + $w_k \tau\gamma(d+\gamma)$ | $w_k \min(k, nq)\gamma^2 + 2\gamma d + 2\gamma + 4d\tau$ + $w_k \min(k, nq) + w_k \gamma + d - 1\tau + 3n - 4$ |
| N-D-SFDP in [14]   | $A$      | $2\gamma(p-1)$ + $\tau\gamma(d+\gamma)$ | $\gamma^2 + 4d + 3\gamma d + 3\gamma\tau$ | $\gamma + d - 1\tau + 2(n-1)$ |
| *DM-N-D-SFDP       | $w_k A$  | $w_k 2\gamma(p-1)$ + $w_k \tau\gamma(d+\gamma)$ | $w_k (\gamma^2 + 4d + 3\gamma d + 3\gamma\tau)$ | $w_k (\gamma + d - 1\tau + 3n - 4)$ |

* New algorithms applied the DM scheme proposed in this paper

IA/IC/IM: Integer Addition/Comparison/Multiplication
RA/RC/RM: Real Addition/Comparison/Multiplication

\[ \Lambda = \gamma (2\rho - 1) \quad \tau = (\gamma \rho - \gamma + 1) \]

**FIGURE 7.** The computational complexity for $C_2$.

**FIGURE 8.** The computational complexity for $C_3$.

**FIGURE 9.** The iteration distribution histogram of the D-SFDP (above) and DM-D-SFDP (below) algorithms and the computational complexity for $C_3$ at $E_b/N_0 = 6dB$.

Fig. 7 and Fig. 8 present a comparison of the average computational complexity of the D-SFDP algorithm ($\Phi_\gamma$, $\Phi_\tau$, $\Phi_\rho$) and our proposed DM-D-SFDP algorithm ($\hat{\Phi}_\gamma$, $\hat{\Phi}_\tau$, $\hat{\Phi}_\rho$) for $C_2$ and $C_3$, respectively. Since the number of flipped symbols $w_k$ in the $k$-th iteration is much less than $w_i$ during the $t$-th stage, $k \in (k_{t-1}, k_t)$, the actual computational complexity of each iteration will rapidly decrease as the iteration number grows. At the same time, the average number of iterations $\bar{k}$ is almost halved with the help of the highly parallelized DM scheme. The average computational complexity of DM-$x$-SFDP algorithms is close to the $x$-SFDP algorithms. The average complexity with IC/RC operations for the DM-D-SFDP algorithm ($\hat{\Phi}_\rho$) is even less than that of the the D-SFDP algorithm ($\Phi_\rho$) as depicted in Fig. 7 and Fig. 8.

As a typical case, we consider the NB-LDPC code over GF(64) in the BeiDou standard and use the iteration distribution histogram to visualize the proportion of the number of actual iterations in Fig. 9. We record the actual
number of iterations of the D-SFDP algorithm with or without the DM scheme by $F = 10^2$ correctly decoded frames. The computational complexity of the D-SFDP algorithm and the DM-D-SFDP algorithm is also presented in the table of Fig. 9. The DM scheme significantly accelerates the convergence and reduces $\kappa$ to 47%. This phenomenon explains why our proposed DM scheme can significantly lower the decoding delay by reducing the average iteration number to less than half. The average computational complexity is statistically analyzed over $10^2$ successfully decoded frames with (10), where $\Phi_g$ and $\Phi_d$ in the DM-D-SFDP algorithm increase at most 15% while $\Phi_c$ decreases 29% compared with those of the D-SFDP algorithm for $C_3$ at $E_b/N_0 = 6$ dB.

The computational complexity $\Phi_c$ is closely affected by the parameter $N$, which also decreases 10% to 40% in $C_2$ with the DM-D-SFDP decoding. Although the computational complexity $\Phi_g$, $\Phi_d$ per iteration of DM-x-SFDP algorithms is proportional to the coefficient of $w_k$, which is relatively high at the initial decoding period, the number of iterations decreases to less than half as the DM scheme speeds up the convergence. The actual execution time of the decoding is significantly reduced, as the number of iterations drops to less than half, while the increase in computational complexity of each iteration is minimal.

VI. CONCLUSION
This paper proposes a DM scheme for the SFDP decoding algorithms of regular NB-LDPC codes, which dynamically flips multiple symbols in parallel to accelerate the decoding process. The proposed DM scheme can be highly parallelized in hardware implementations with dynamic modification of the number of flipped symbols in each iteration, which extensively lowers the decoding latency and improves the throughput over existing SFDP algorithms. The resulting DM scheme is better than the single symbol flipping algorithms because it significantly enhances the error correction capability and achieves faster convergence speed with little increase in average computational complexity. Numerical results reveal that the proposed DM-x-SFDP algorithm exhibits better BER performance than existing multi-symbol flipping algorithms and decreases using the global search or sort operations. The implementation of our proposed DM scheme in SFDP algorithms can achieve better trade-offs between the computation complexity and error correction performance, providing a promising new approach for the practical applications of NB-LDPC codes. It has been well demonstrated through extensive simulations that better BER and FER performance can be achieved with a slight increase in average computational complexity. Moreover, our proposed DM scheme has ultra-fast convergence speed and can be highly parallelized for practical applications in hardware implementations.

REFERENCES
[1] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, “On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes,” IEEE Trans. Commun., vol. 53, no. 4, pp. 549–554, Apr. 2005.
[2] A. Darabiha, A. C. Carusone, and F. R. Kschischang, “A bit-serial approximate min-sum LDPC decoder and FPGA implementation,” in Proc. IEEE Int. Symp. Circuits Syst., 2006, p. 4.
[3] C.-W. Sham, X. Chen, W. M. Tam, Y. Zhao, and F. C. M. Lau, “A layered QC-LDPC decoder architecture for high speed communication system,” in Proc. IEEE Asia-Pacific Conf. Circuits Syst. (APCCAS), 2012, pp. 475–478.
[4] Q. Lu, Z. Shen, C.-W. Sham, and F. C. M. Lau, “A parallel-routing network for reliability inferences of single-parity-check decoder,” in Proc. Int. Conf. Adv. Technol. Commun. (ATC), 2015, pp. 127–132.
[5] X. Chen, J. Kang, S. Lin, and V. Akella, “Memory system optimization for FPGA-based implementation of quasi-cyclic LDPC codes decoders,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 1, pp. 98–111, Jan. 2011.
[6] Q. Lu, J. Fan, C.-W. Sham, W. M. Tam, and F. C. M. Lau, “A 3.0 Gb/s throughput hardware-efficient decoder for cyclically-coupled QC-LDPC codes,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 63, no. 1, pp. 134–145, Jan. 2016.
[7] S. Nowak, G. Smietanka, and R. Kays, “High efficiency broadband transmission with LDPC codes over $G(F^2)$,” in Proc. IEEE BMSB, 2011, pp. 1–6.
[8] X. Yang, M. Jiang, Z. Li, C. Wang, and C. Zhao, “Efficient decoding approach for NB-LDPC codes with short blocking,” IET Commun., vol. 13, pp. 2141–2145, Aug. 2019.
[9] Y. Tan, X. G. Xi, H. Yang, and R. Jiang, “An improved low-complexity decoding algorithms for non-binary LDPC codes in BeiDou navigation system,” in Proc. IOP Conf. Ser. Mater. Sci. Eng., 2018, Art. no. 12043.
[10] C. Wang, J. Li, L. Kong, F. Shu, and F. C.-M. Lau, “Adaptive 2-D scheduling-based nonbinary majority-logic decoding for NAND flash memory,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 67, no. 7, pp. 1349–1353, Jul. 2020.
[11] I. B. Djordjevic and B. V. Vasic, “Nonbinary LDPC codes for optical communication systems,” IEEE Photon. Technol. Lett., vol. 17, no. 10, pp. 2224–2226, Oct. 2005.
[12] L. Changyu, I. B. Djordjevic, W. M. Weiming, and C. Yi, “Nonbinary LDPC BICM for next-generation high-speed optical transmission systems,” ZTE Commun., vol. 15, no. 1, pp. 61–66, 2019.
[13] M. C. Davey and D. MacKay, “Low-density parity check codes over $G(F)$,” IEEE Commun. Lett., vol. 2, no. 6, pp. 165–167, Jun. 1998.
[14] L. Barnault and D. Declercq, “Fast decoding algorithm for LDPC over $G(F^2)$,” in Proc. IEEE Inf. Theory Workshop, 2003, pp. 70–73.
[15] V. Savin, “Min–max decoding for non binary LDPC codes,” in Proc. IEEE Int. Symp. Inf. Theory, 2008, pp. 960–964.
[16] X. Li, T. Qin, H. Q. Chen, Y. Sun, Q. Liang, and L. Luo, “Hard-information bit-reliability based decoding algorithm for majority-logic decodable nonbinary LDPC codes,” IEEE Commun. Lett., vol. 20, no. 5, pp. 866–869, May 2016.
[17] S. Yeo and J.-C. Park, “Improved hard-reliability based majority-logic decoding for non-binary LDPC codes,” IEEE Commun. Lett., vol. 21, no. 2, pp. 230–233, Feb. 2017.
[18] S. Wang, Q. Huang, and Z. Wang, “Symbol flipping decoding algorithms based on prediction for non-binary LDPC codes,” IEEE Trans. Commun., vol. 65, no. 5, pp. 1913–1924, May 2017.
[19] B. Dai, R. Liu, C. Gao, and Z. Mei, “Symbol flipping algorithm with self-adjustment strategy for LDPC codes over $G(F)$,” IEEE Trans. Veh. Technol., vol. 68, no. 7, pp. 7189–7193, Jul. 2019.
[20] Z. Zhao, X. Jiao, J. Mu, and Y.-C. He, “Randomly Penalized symbolic flipping decoding of non-binary LDPC codes,” IEEE Trans. Veh. Technol., vol. 70, no. 1, pp. 639–654, Jan. 2021.
[21] A. Voicila, D. Declercq, F. Verdier, M. P. C. Fossorier, and P. Urard, “Low-complexity decoding for non-binary LDPC codes in high order fields,” IEEE Trans. Commun., vol. 58, no. 5, pp. 1365–1375, May 2010.
[22] E. Li, D. Declercq, and K. K. Gunnam, “Trellis-based extended min-sum algorithm for non-binary LDPC codes and its hardware structure,” IEEE Commun. Lett., vol. 61, no. 7, pp. 2606–2611, Jul. 2013.
[23] Q. Huang, M. Zhang, Z. Wang, and L. Wang, “Bit-reliability based low-complexity decoding algorithms for non-binary LDPC codes,” IEEE Trans. Commun., vol. 62, no. 12, pp. 4230–4240, Dec. 2014.
[24] Q. Huang and S. Yuan, “Bit-reliability-based decoders for non-binary LDPC codes,” IEEE Trans. Commun., vol. 64, no. 1, pp. 38–48, Jan. 2016.
[25] G. Sundararajan, C. Winstead, and E. Boutillon, “Noisy gradient descent bit-flip decoding for LDPC codes,” IEEE Trans. Commun., vol. 62, no. 10, pp. 3385–3400, Oct. 2014.

[26] Z. Zhao, X. Jiao, J. Mu, Y.-C. He, and J. Guo, “Fast-converging flipping rules for symbol flipping decoding of non-binary LDPC codes,” IEICE Trans. Fundam. Electron. Commun. Comput. Sci., vol. 102-A, no. 7, pp. 930–933, 2019.

XIUWEN BAO received the B.Sc. degree in communication and information engineering from Soochow University, Suzhou, China, in 2020. She is currently pursuing the M.S. degree in information and communication engineering with Southeast University, Nanjing, China. Her research interests include joint source-channel coding and channel coding.

MING JIANG (Member, IEEE) received the B.Sc., M.S., and Ph.D. degrees in communication and information engineering from Southeast University, Nanjing, China, in 1998, 2003, and 2007, respectively. He is currently an Associate Professor with the National Mobile Communications Research Laboratory, Southeast University and the Purple Mountain Laboratories, Nanjing. His research interests include coding and modulation technology.

DONGLI FAN received the B.Sc. degree in communication and information engineering, from Xidian University, Xi’an, China, in 2019, and the M.S. degree in communication and information engineering from Southeast University, Nanjing, China, in 2022. Her research interests include coding and modulation technology.

MINYANG ZHU (Graduate Student Member, IEEE) received the B.E. degree in information engineering from Southeast University, Nanjing, China, in 2018, where he is currently pursuing the Ph.D. degree in information and communication engineering. His research interests include mobile communications, optical communications, and error control coding.