IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 19, 2022, accepted May 28, 2022, date of publication June 6, 2022, date of current version June 9, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3180368

Iterative Hard-Decision Decoding Algorithms for Binary Reed-Muller Codes

YONG-TING NI (倪詠庭), DUC NHAT NGUYEN (阮德日), FENG-KAI LIAO (廖烽凱), TZU-CHIEH KAO (高子傑), AND CHAO-YU CHEN (陳昭羽), (Member, IEEE)
Department of Engineering Science, National Cheng Kung University, Tainan 70101, Taiwan

Corresponding author: Chao-Yu Chen (super@mail.ncku.edu.tw)

This work was supported by the Ministry of Science and Technology, Taiwan, under Grant MOST 109–2628–E–006–008–MY3 and Grant MOST 111–2218–E–305–002.

ABSTRACT In this paper, novel hard-decision iterative decoding algorithms for binary Reed-Muller (RM) codes are presented. First, two algorithms are devised based on the majority-logic decoding algorithm with reliability measures of the received sequence. The bit-flipping (BF) and the normalized bit-flipping (NBF) decoding algorithms are hard-decision decoding algorithms. According to the updated hard reliability measures, the BF and NBF algorithms flip one bit of the received hard-decision sequence at a time in each iteration. The NBF decoding algorithm performs better than the BF decoding algorithm by normalizing the reliability measures of the information bits. Moreover, the BF and NBF algorithms are modified to flip multiple bits in one iteration to reduce the average number of iterations. The modified decoding algorithms are called the multiple-bits-flipping (MBF) algorithm and the normalized multiple-bits-flipping (NMBF) algorithm, respectively. The proposed algorithms have low computational complexities and can converge rapidly after a small number of iterations. The simulation results show that the proposed hard-decision decoding algorithms outperform the conventional decoding algorithm.

INDEX TERMS Reed-Muller code, bit-flipping, majority-logic decoding, iterative decoding.

I. INTRODUCTION

The Reed-Muller (RM) code is a class of multiple-error-correction codes. In the past, RM codes were used in wireless communications, especially in deep-space communications. RM codes were first discovered by Muller [1] and the conventional decoding scheme of RM code, which is the majority-logic decoding, was proposed by Reed in 1954 [2]. Ever since their discovery, a number of efficient decoding schemes have been constructed. In 1995, Schnabl and Bossert proposed a generation of Reed-Muller codes by multiple concatenations and provided a new decoding procedure using soft-decision information [3]. In 1999, a maximum-likelihood (ML) decoder which uses a distance-preserving map and multiple fast Hadamard transforms (FHTs) was presented by Jones and Wilkinson [4] whereas a maximum a posterior (MAP) decoding algorithm for the first-order RM codes was proposed in [5]. In 2000, a new soft-decision majority-logic decoding algorithm was presented by revising the conventional majority-logic decoding [6]. Besides, the recursive decoding algorithms were provided in [7]–[11]. In 2006, the recursive list decoding was modified by Dumer and Shabunov to approach the performance of ML decoding [9]. Recently, Ye and Abbe proposed the recursive projection-aggregation (RPA) decoding algorithm with lists to have comparable performance with the recursive list decoding [10]. In the same year, the recursive puncturing-aggregation (RXA) algorithm was proposed via replacing the projection by puncturing in RPA [11].

In recent years, the breakthrough of polar codes [12]–[15] has brought attention back to RM codes due to the similarity of these two codes. The performance comparison between RM codes and polar codes was demonstrated in [10], [15]–[17]. It has been shown that the RM codes with RPA decoding [10] can significantly outperform the polar codes under successive-cancellation list (SCL) decoding [18]–[20]. In addition, by exploiting the idea of decoding polar codes, [21], [22] presented permutation-based decoding methods for RM codes.

Although various algorithms for decoding RM codes have been designed, most of them are soft-decision decoding. In this paper, the hard-decision decoding is taken

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

into consideration with low complexity. The conventional majority-logic decoding has low complexity but the performance is not good enough. In contrast, the hard-decision ML decoding algorithm has better performance but the complexity of the ML decoding is extremely large. In this work, we aim for new algorithms outperforming the conventional majority-logic decoding with controllable complexity.

This paper presents novel iterative decoding algorithms for Reed-Muller codes by exploiting the idea of bit-flipping [23] and min-sum [24] decoding algorithms for low-density parity-check (LDPC) codes [25]. These proposed algorithms are devised based on the min-sum operation and the reliability measures of the received sequence. The first proposed algorithm is called the bit-flipping (BF) decoding algorithm because we flip one bit of the received hard-decision sequence at a time in each iteration. Through decoding iterations, the hard-decision sequence is updated.* Next, the second decoding algorithm, called the normalized bit-flipping (NBF) decoding algorithm, is proposed. The reliability measures of information bits are normalized depending on the degree to balance the number of votes. Therefore, the NBF decoding algorithm offers better performance compared with the BF decoding algorithm. To reduce the number of iterations and the complexity, a termination process is provided as well. These proposed iterative decoding algorithms can converge very rapidly. For the hard-decision BF and NBF decoding algorithms, we additionally provide a condition to allow flipping multiple bits in each iteration. The advantage of multiple-bits flipping is to reduce the required number of iterations and hence to have less computational complexity. We call them multiple-bits-flipping (MBF) decoding and normalized multiple-bits-flipping (NMBF) decoding, respectively.

The numerical experiment results show that the proposed hard-decision decoding algorithms perform better than the conventional majority-logic decoding algorithm over the additive white Gaussian noise (AWGN) channel.

The rest of this paper is organized as follows. Notations and definitions are provided in Section II. Section III introduces the conventional majority-logic decoding for RM codes. In Section IV, the steps and examples for the proposed algorithms are given. Section V shows the simulation results over the AWGN channel. Finally, the concluding remarks are given in Section VI.

II. PRELIMINARY AND NOTATIONS

We denote the binary rth-order Reed-Muller code of length 2^m over GF(2) by RM(r, m). Let the 2^m-tuple vectors

$$\boldsymbol{x}_i = (\underbrace{00\cdots0}_{2^{i-1}}\underbrace{11\cdots1}_{2^{i-1}}\underbrace{00\cdots0}_{2^{i-1}}\cdots\underbrace{11\cdots1}_{2^{i-1}}) \tag{1}$$

with 2^{m-i+1} segments for $i = 1, 2, \ldots, m$ and $\boldsymbol{x}_0 = (11\cdots1)$ which is the all-one vector. For vectors $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\boldsymbol{b} = (b_0, b_1, \ldots, b_{n-1})$, we denote $\boldsymbol{ab} = (a_0 \cdot b_0, a_1 \cdot b_1, \ldots, a_{n-1} \cdot b_{n-1})$ where "$\cdot$" represents the product and say that the product vector $\boldsymbol{x}_{i_1}\boldsymbol{x}_{i_2}\cdots\boldsymbol{x}_{i_\gamma}$ where $1 \le i_1 < i_2 < \cdots < i_\gamma \le m$ has degree γ. Then, RM(r, m) can be generated by the following vectors:

$$\{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m, \boldsymbol{x}_1\boldsymbol{x}_2, \ldots, \boldsymbol{x}_{m-1}\boldsymbol{x}_m, \ldots, \text{up to products of degree } r\}. \tag{2}$$

Therefore, the generator matrix of RM(r, m) can also be expressed as

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_m \\ \boldsymbol{x}_1\boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{m-1}\boldsymbol{x}_m \\ \vdots \\ \text{up to products of degree } r \end{bmatrix}. \tag{3}$$

Assume that $\boldsymbol{u} = (u_0, u_1, \ldots, u_{K-1})$ is the information vector where K is the number of information bits. Then, the encoded codeword $\boldsymbol{c} = (c_0, \ldots, c_{2^m-1})$ can be obtained by

$$\boldsymbol{c} = \boldsymbol{u} \cdot \boldsymbol{G} = \sum_{j=0}^{K-1} u_j \boldsymbol{g}_j \tag{4}$$

where $\boldsymbol{g}_j$ is the jth row of the generator matrix $\boldsymbol{G}$. For $1 \le j \le K-1$, if u_j is the coefficient of the product vector $\boldsymbol{x}_{i_1}\boldsymbol{x}_{i_2}\cdots\boldsymbol{x}_{i_\gamma}$ where $1 \le i_1 < i_2 < \cdots < i_\gamma \le m$, we call u_j the information bit corresponding to a product vector of degree γ. Also, u_0 is called the information bit corresponding to the all-one vector because it is the coefficient of $\boldsymbol{x}_0$. Take RM(2, 3) for example, if the information vector is $\boldsymbol{u} = (u_0, u_1, u_2, u_3, u_4, u_5, u_6)$, then the encoded codeword is expressed as

$$\boldsymbol{c} = u_0\boldsymbol{x}_0 + u_1\boldsymbol{x}_1 + u_2\boldsymbol{x}_2 + u_3\boldsymbol{x}_3 + u_4\boldsymbol{x}_1\boldsymbol{x}_2 + u_5\boldsymbol{x}_1\boldsymbol{x}_3 + u_6\boldsymbol{x}_2\boldsymbol{x}_3$$

where u_0 is the information bit corresponding to all-one vector; u_1, u_2, and u_3 are the information bits corresponding to the product vectors of degree 1; u_4, u_5, u_6 are the information bits corresponding to the product vectors of degree 2.

III. REVIEW OF THE CONVENTIONAL MAJORITY-LOGIC DECODING

For the hard-decision decoding, the conventional decoding algorithm is the majority-logic decoding algorithm which was first proposed by Reed in [2]. The concept of the conventional majority-logic decoding is simple and the majority-logic decoding has low complexity. However, the performance is limited. In contrast, the hard-decision ML decoding algorithm has better performance but the complexity is extremely large. The comparison of complexities for different hard-decision decoding algorithms is given in Table 1. In our work, the major contribution is to modify the conventional

*The BF algorithm was briefly described in the conference paper [26].

majority-logic decoding with better performance and acceptable complexity. In the following, we will introduce the procedure of the conventional majority-logic decoding.

During the decoding process, there are $2^{m-\gamma}$ votes that can be used to estimate each information bit corresponding to a product vector of degree γ. We denote k the index of vote where $0 \le k < 2^{m-\gamma}$. Let u_j be the coefficient of the product vector $\boldsymbol{x}_{i_1}\boldsymbol{x}_{i_2}\cdots\boldsymbol{x}_{i_\gamma}$ of degree γ where $1 \le i_1 < i_2 < \cdots < i_\gamma \le m$. We first define the set A as

$$A = \Big\{a_1 2^{i_1-1} + a_2 2^{i_2-1} + \cdots + a_\gamma 2^{i_\gamma-1} : a_t \in \{0,1\}, \text{ for } t = 1,2,\ldots,\gamma\Big\}. \quad (5)$$

Then, the set E is defined as

$$E = \{0,1,\ldots,m-1\}\backslash\{i_1-1, i_2-1, \ldots, i_\gamma-1\} = \{j_1, j_2, \ldots, j_{m-\gamma}\} \quad (6)$$

and the set B is defined as

$$B = \Big\{b_1 2^{j_1} + b_2 2^{j_2} + \cdots + b_{m-\gamma} 2^{j_{m-\gamma}} : b_t \in \{0,1\}, \text{ for } t = 1,2,\ldots,\gamma\Big\}. \quad (7)$$

For each $q_k \in B$, we can construct the following set:

$$S_k = q_k + A = \{q_k + p : p \in A\} \quad (8)$$

consisting of bit indices of the received sequence.

The steps of the conventional majority-logic decoding for RM(r, m) can be described as follows:

Initialization: Let $\gamma = r$ and $n = 2^m$.

Step 1. For $0 \le l < n$, make a hard-decision on each received value y_l from noisy channel by

$$v_l = \begin{cases} 0, & \text{if } y_l > 0; \\ 1, & \text{otherwise.} \end{cases} \quad (9)$$

Step 2. For any information bit u_j which is the coefficient of a product vector of degree γ, compute each $\hat{u}_{j,k}$ for u_j by

$$\hat{u}_{j,k} = \sum_{l \in S_k} v_l \quad (10)$$

for $k = 0, 1, \ldots, 2^{m-\gamma}-1$ where S_k is given in (8).

Step 3. Then, determine the estimation of u_j according to the majority voting of $2^{m-\gamma}$ equations from (10). If $\gamma = 0$, go to Step 5.

Step 4. Remove the estimated information of u_j's corresponding to vectors of degree γ from the received sequence and modify $v_l \to v'_l$ for $0 \le l < n$ by

$$\boldsymbol{v}' = \boldsymbol{v} - \sum_{1 \le i_1 < i_2 < \cdots < i_\gamma \le m} u_j \boldsymbol{x}_{i_1}\boldsymbol{x}_{i_2}\cdots\boldsymbol{x}_{i_\gamma} \quad (11)$$

where $\boldsymbol{v} = (v_0, v_1, \ldots, v_{n-1})$ and $\boldsymbol{v}' = (v'_0, v'_1, \ldots, v'_{n-1})$. Set $\gamma = \gamma - 1$, go to Step 2.

Step 5. Stop decoding and obtain all decoded information bits u_j.

In Step 2, it can be found that there are $2^{m-\gamma}$ estimates for one information bit u_j and the estimated u_j is determined based on these $2^{m-\gamma}$ votes in Step 3. The estimation of u_j is the value ($\in \{0,1\}$) with majority votes. That is why it is called the majority-logic decoding. Then, Step 4 decreases the degree by setting $\gamma = \gamma - 1$ and repeats the decoding process until $\gamma = 0$.

IV. PROPOSED ITERATIVE DECODING ALGORITHMS

In this section, we first demonstrate the hard-decision BF and NBF decoding algorithms. Then, two modified decoding algorithms, i.e., MBF and NMBF algorithms, which consider multiple-bits flipping are introduced.

A. BF DECODING ALGORITHM

We follow the same notations given in Section III. Besides, let $I_{\max}$ be the maximum number of iterations in the decoding process. For $1 \le i \le I_{\max}$, let $\boldsymbol{v}^{(i)} = (v_0^{(i)}, v_1^{(i)}, \ldots, v_{n-1}^{(i)})$ denote the hard-decision codeword generated in the ith decoding iteration. For $1 \le i \le I_{\max}$ and $0 \le j < K$, $R_j^{(i)}$ denotes the reliability measure of the jth information bit u_j in the ith decoding iteration. Furthermore, M_l represents the set of index j such that $G_{j,l} = 1$ which is the element in the jth row and lth column of the generator matrix $\boldsymbol{G}$ in (3). Giving an additional condition on γ, M_l^γ represents the collection of j's such that $G_{j,l} = 1$ and the jth row vector of $\boldsymbol{G}$ is a product vector of degree γ. Moreover, D_l denotes the difference between the estimated codeword bit $v_l^{(i)}$ and the reliability measure in the lth position of the received sequence. Based on the notations defined above, the block diagram of the decoding scheme is illustrated in Fig. 1.

In each decoding iteration, it allows flipping only one bit from the updated codeword at a time.

Initialization: Set $i = 1$ and $\gamma = r$. Let $I_{\max}$ denote the maximum number of iterations and let $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ be the received sequence over AWGN channel.

Step 1. For $0 \le l < n$, determine $v_l^{(1)}$ for each received bit y_l according to the hard-decision rule:

$$v_l^{(1)} = \begin{cases} 1, & \text{if } y_l > 0; \\ -1, & \text{otherwise.} \end{cases} \quad (12)$$

Also, let $\boldsymbol{\phi}^{(1)} = \boldsymbol{v}^{(1)}$.

Step 2. For $0 \le k < 2^{m-\gamma}$, compute each vote information for the information bit u_j corresponding to a vector of degree γ:

$$\hat{u}_{j,k} = \prod_{l \in S_k} \phi_l^{(i)} \quad (13)$$

where S_k is the set of indices in the received sequence which are related to u_j as given in (8).

Step 3. Obtain the reliability measure of the information bit u_j by summing all $2^{m-\gamma}$ votes from Step 2:

$$R_j^{(i)} = \sum_{k=0}^{2^{m-\gamma}-1} \hat{u}_{j,k}. \quad (14)$$

If $\gamma = 0$, go to Step5.

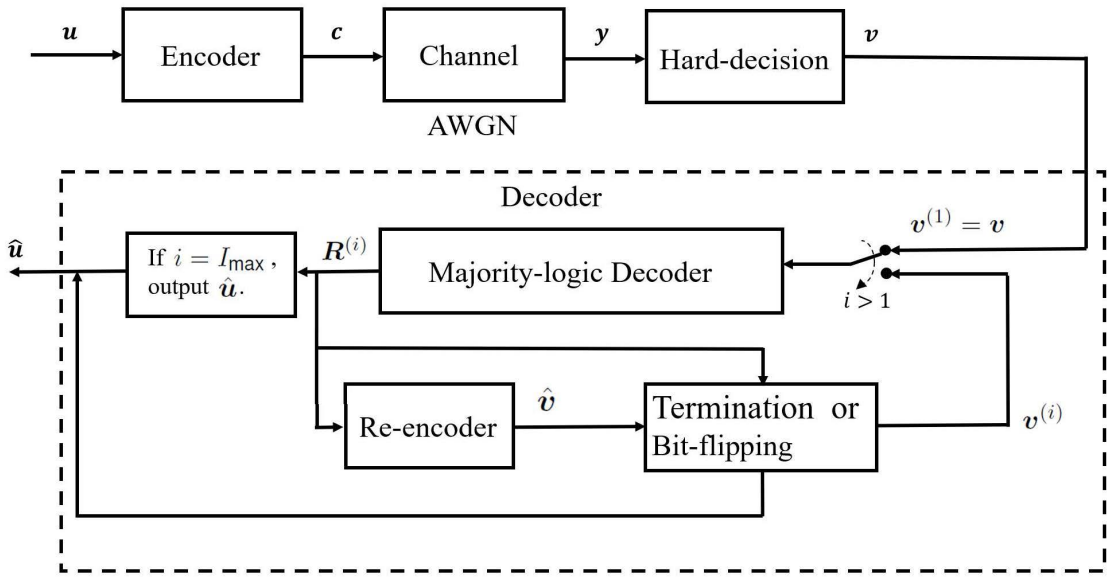

FIGURE 1. The block diagram of the hard-decision bit-flipping decoding algorithm.

Step 4. For $l = 0, 1, \ldots, n-1$, if $\phi_l^{(i)}$ does not involve the information of any information bit of degree γ, then $\left(\phi_l^{(i)}\right)' = \phi_l^{(i)}$; otherwise,

$$\left(\phi_l^{(i)}\right)' = \phi_l^{(i)} \prod_{j \in M_l^{\gamma}} \text{sgn}(R_j^{(i)}). \tag{15}$$

Set $\gamma = \gamma - 1$ and $\boldsymbol{\phi}^{(i)} = \left(\boldsymbol{\phi}^{(i)}\right)'$. Go to Step 2.

Step 5. If $i = I_{\max}$, make the following hard-decision: 1) $u_j = 1$, if $R_j^{(i)} \leq 0$; 2) $u_j = 0$, if $R_j^{(i)} > 0$ and stop decoding. Otherwise, go to Step 6.

Step 6. *(Re-encoding)* Let $\hat{\boldsymbol{v}} = (\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_{n-1})$ where

$$\hat{v}_l = \prod_{j \in M_l} \text{sgn}(R_j^{(i)}) \tag{16}$$

and $0 \leq l < n$. Go to Step 7.

Step 7. (*Termination*) If $v_l^{(i)} = \hat{v}_l$ for all l, make the following hard-decision: 1) $u_j = 1$, if $R_j^{(i)} \leq 0$; 2) $u_j = 0$, if $R_j^{(i)} > 0$ and stop decoding. Otherwise, go to Step 8.

Step 8. For $l = 0, 1, \ldots, n-1$, if

$$v_l^{(i)} \neq \hat{v}_l, \tag{17}$$

calculate

$$D_l = \left| v_l^{(i)} - \hat{v}_l \cdot \min_{j \in M_l} |R_j^{(i)}| \right|;$$

otherwise, $D_l = 0$. Go to Step 9.

Step 9. Find l' with the maximum value of D_l, i.e.,

$$l' = \underset{0 \leq l < n}{\text{argmax}}\, D_l.$$

Set $i = i + 1$. Then, for $l = 0, 1, \ldots, n-1$, update the estimated codeword bit as follows:

$$v_l^{(i)} = \begin{cases} -v_l^{(i-1)}, & \text{if } l = l'; \\ v_l^{(i-1)}, & \text{otherwise.} \end{cases}$$

Set $\gamma = r$ and $\boldsymbol{\phi}^{(i)} = \boldsymbol{v}^{(i)}$. Go to Step 2.

To reduce the complexity, a termination process is developed in Step 7. The decoding process is terminated when the updated sequences are identical within two consecutive iterations.

Note that Step 2, Step 4, and Step 8 of the BF algorithm in [26] are modified in this manuscript to have simpler expressions.

Example 1: To demonstrate the decoding process, let us consider RM(1, 3) with the generator matrix given by

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Assume the information vector is $\boldsymbol{u} = (u_0, u_1, u_2, u_3) = (0, 0, 1, 0)$ where u_0 is the information bit corresponding to the all-one vector and u_1, u_2 and u_3 are the information bits corresponding to vectors of degree 1. Hence, we have the encoded codeword $\boldsymbol{c} = (c_0, c_1, \ldots, c_7) = (0, 0, 1, 1, 0, 0, 1, 1)$. Assume the received sequence is

$$\begin{aligned} \boldsymbol{y} &= (y_0, y_1, \ldots, y_7) \\ &= (1.5, 0.4, -2.3, 1.5, 0.3, 2.4, -1.2, -0.7) \end{aligned}$$

over the AWGN channel. First of all, we make the hard-decision of $\boldsymbol{y}$ and obtain the initial information $\boldsymbol{\phi}^{(1)} =$

$(\phi_0^{(1)}, \phi_1^{(1)}, \ldots, \phi_7^{(1)}) = (1, 1, -1, 1, 1, 1, -1, -1)$ according to Step 1. From (13), we have the following four equations

$$\begin{aligned}\hat{u}_{1,0} &= \phi_0^{(1)}\phi_1^{(1)} = 1;\\ \hat{u}_{1,1} &= \phi_2^{(1)}\phi_3^{(1)} = -1;\\ \hat{u}_{1,2} &= \phi_4^{(1)}\phi_5^{(1)} = 1;\\ \hat{u}_{1,3} &= \phi_6^{(1)}\phi_7^{(1)} = 1\end{aligned}$$

for $j = 1$. Then, the reliability measure of u_1 in the first iteration is

$$R_1^{(1)} = \hat{u}_{1,0} + \hat{u}_{1,1} + \hat{u}_{1,2} + \hat{u}_{1,3} = 1 + (-1) + 1 + 1 = 2$$

according to (14). Similarly, we can also obtain $R_2^{(1)} = -2$ and $R_3^{(1)} = 2$ which are the reliability measures of u_2 and u_3, respectively. Next, we need to modify $\boldsymbol{\phi}^{(1)}$ by removing the contributions from u_1, u_2, and u_3. Then, we obtain the estimations of u_0 as follows:

$$\begin{aligned}\hat{u}_{0,0} &= \phi_0^{(1)} = 1;\\ \hat{u}_{0,1} &= \phi_1^{(1)}\mathrm{sgn}(R_1^{(1)}) = 1 \cdot \mathrm{sgn}(2) = 1;\\ \hat{u}_{0,2} &= \phi_2^{(1)}\mathrm{sgn}(R_2^{(1)}) = -1 \cdot \mathrm{sgn}(-2) = 1;\\ \hat{u}_{0,3} &= \phi_3^{(1)}\mathrm{sgn}(R_1^{(1)})\mathrm{sgn}(R_2^{(1)}) = 1 \cdot \mathrm{sgn}(2)\mathrm{sgn}(-2) = -1;\\ \hat{u}_{0,4} &= \phi_4^{(1)}\mathrm{sgn}(R_3^{(1)}) = 1 \cdot \mathrm{sgn}(2) = 1;\\ \hat{u}_{0,5} &= \phi_5^{(1)}\mathrm{sgn}(R_1^{(1)})\mathrm{sgn}(R_3^{(1)}) = 1 \cdot \mathrm{sgn}(2)\mathrm{sgn}(2) = 1;\\ \hat{u}_{0,6} &= \phi_6^{(1)}\mathrm{sgn}(R_2^{(1)})\mathrm{sgn}(R_3^{(1)}) = -1 \cdot \mathrm{sgn}(-2)\mathrm{sgn}(2) = 1;\\ \hat{u}_{0,7} &= \phi_7^{(1)}\mathrm{sgn}(R_1^{(1)})\mathrm{sgn}(R_2^{(1)})\mathrm{sgn}(R_3^{(1)})\\ &= -1 \cdot \mathrm{sgn}(2)\mathrm{sgn}(-2)\mathrm{sgn}(2) = 1.\end{aligned}$$

Therefore, the reliability measure of u_0 is

$$\begin{aligned}R_0^{(1)} &= \hat{u}_{0,0} + \hat{u}_{0,1} + \hat{u}_{0,2} + \hat{u}_{0,3}\\ &\quad + \hat{u}_{0,4} + \hat{u}_{0,5} + \hat{u}_{0,6} + \hat{u}_{0,7}\\ &= 1 + 1 + 1 + (-1)\\ &\quad + 1 + 1 + 1 + 1 = 6.\end{aligned}$$

In Step 6, we can re-encode the estimated codeword based on the reliability measures of information bits:

$$\begin{aligned}\hat{v}_0 &= \mathrm{sgn}(R_0^{(1)}) = \mathrm{sgn}(6) = 1;\\ \hat{v}_1 &= \mathrm{sgn}(R_0^{(1)})\mathrm{sgn}(R_1^{(1)}) = \mathrm{sgn}(6)\mathrm{sgn}(2) = 1;\\ \hat{v}_2 &= \mathrm{sgn}(R_0^{(1)})\mathrm{sgn}(R_2^{(1)}) = \mathrm{sgn}(6)\mathrm{sgn}(-2) = -1;\\ \hat{v}_3 &= \mathrm{sgn}(R_0^{(1)})\mathrm{sgn}(R_1^{(1)})\mathrm{sgn}(R_2^{(1)})\\ &= \mathrm{sgn}(6)\mathrm{sgn}(2)\mathrm{sgn}(-2) = -1;\\ \hat{v}_4 &= \mathrm{sgn}(R_0^{(1)})\mathrm{sgn}(R_3^{(1)}) = \mathrm{sgn}(6)\mathrm{sgn}(2) = 1;\\ \hat{v}_5 &= \mathrm{sgn}(R_0^{(1)})\mathrm{sgn}(R_1^{(1)})\mathrm{sgn}(R_3^{(1)})\\ &= \mathrm{sgn}(6)\mathrm{sgn}(2)\mathrm{sgn}(2) = 1;\\ \hat{v}_6 &= \mathrm{sgn}(R_0^{(1)})\mathrm{sgn}(R_2^{(1)})\mathrm{sgn}(R_3^{(1)})\\ &= \mathrm{sgn}(6)\mathrm{sgn}(-2)\mathrm{sgn}(2) = -1;\\ \hat{v}_7 &= \mathrm{sgn}(R_0^{(1)})\mathrm{sgn}(R_1^{(1)})\mathrm{sgn}(R_2^{(1)})\mathrm{sgn}(R_3^{(1)})\\ &= \mathrm{sgn}(6)\mathrm{sgn}(2)\mathrm{sgn}(-2)\mathrm{sgn}(2) = -1.\end{aligned}$$

In this example, we find $v_3^{(1)} \neq \hat{v}_3$. Therefore, we have

$$D_3 = \left|v_3^{(1)} - \hat{v}_3 \cdot \min\{|6|, |2|, |-2|\}\right| = |1 - (-2)| = 3$$

and

$$\boldsymbol{D} = (D_0, D_1, \ldots, D_7) = (0, 0, 0, 3, 0, 0, 0, 0)$$

where there is only one position of $\boldsymbol{v}^{(1)}$ distinct from the re-encoded codeword. The largest value is $D_3 = 3$, so we obtain $l' = 3$. Thus, we flip one bit of the initial sequence $\boldsymbol{v}^{(1)}$ by

$$v_l^{(2)} = \begin{cases} -v_l^{(1)}, & \text{if } l = 3;\\ v_l^{(1)}, & \text{otherwise.}\end{cases}$$

Then, $\boldsymbol{\phi}^{(2)} = \boldsymbol{v}^{(2)} = (1, 1, -1, -1, 1, 1, -1, -1)$ is the updated sequence for the next decoding iteration.

After performing Steps 1 to 6 in the second iteration, we can obtain that $v_l^{(2)} = \hat{v}_l$ for all $l = 0, 1, 2, \ldots, n$. That is,

$$\begin{aligned}v_0^{(2)} &= \hat{v}_0 = \mathrm{sgn}(R_0^{(2)}) = 1;\\ v_1^{(2)} &= \hat{v}_1 = \mathrm{sgn}(R_0^{(2)})\mathrm{sgn}(R_1^{(2)}) = 1;\\ v_2^{(2)} &= \hat{v}_2 = \mathrm{sgn}(R_0^{(2)})\mathrm{sgn}(R_2^{(2)}) = -1;\\ v_3^{(2)} &= \hat{v}_3 = \mathrm{sgn}(R_0^{(2)})\mathrm{sgn}(R_1^{(2)})\mathrm{sgn}(R_2^{(2)}) = -1;\\ v_4^{(2)} &= \hat{v}_4 = \mathrm{sgn}(R_0^{(2)})\mathrm{sgn}(R_3^{(2)}) = 1;\\ v_5^{(2)} &= \hat{v}_5 = \mathrm{sgn}(R_0^{(2)})\mathrm{sgn}(R_1^{(2)})\mathrm{sgn}(R_3^{(2)}) = 1;\\ v_6^{(2)} &= \hat{v}_6 = \mathrm{sgn}(R_0^{(2)})\mathrm{sgn}(R_2^{(2)})\mathrm{sgn}(R_3^{(2)}) = -1;\\ v_7^{(2)} &= \hat{v}_7 = \mathrm{sgn}(R_0^{(2)})\mathrm{sgn}(R_1^{(2)})\mathrm{sgn}(R_2^{(2)})\mathrm{sgn}(R_3^{(2)}) = -1.\end{aligned}$$

It means that the re-encoded codeword $\hat{\boldsymbol{v}}$ is identical to the updated sequence $\boldsymbol{v}^{(2)}$ in the previous iteration. Hence, there is no need to flip any bit. The updated hard-decision sequence $\boldsymbol{v}^{(2)} = (v_0^{(2)}, v_1^{(2)}, \ldots, v_7^{(2)}) = (1, 1, -1, -1, 1, 1, -1, -1)$ is a valid codeword, so we can terminate the decoding process. Then, we output the decoded information vector $(\hat{u}_0, \hat{u}_1, \hat{u}_2, \hat{u}_3) = (0, 0, 1, 0)$ based on the hard-decision of the reliability measures $R_0^{(2)}$, $R_1^{(2)}$, $R_2^{(2)}$, and $R_3^{(2)}$ in Step 7.

B. NBF DECODING ALGORITHM

For the BF algorithm, the reliability measure is updated only based on the signum of $R_j^{(i)}$'s in (15). In the NBF algorithm, we want to introduce the magnitudes of reliability measures when $\phi_l^{(i)}$'s are updated. In addition, the concept of the min-sum algorithm is exploited and then (13) and (15) are modified with the min-sum operations. Besides, the reliability measures $R_j^{(i)}$'s in (14) have to be normalized to balance the number of votes for different degrees. Therefore, the modified algorithm is called the normalized BF algorithm. By letting μ_γ be the normalized factor used for degree γ, Steps 2 to 4 of the BF decoding are modified and described as follows.

Step 2. For $0 \leq k < 2^{m-\gamma}$, compute each vote information for the information bit u_j corresponding to a vector

of degree γ by the min-sum operation:

$$\hat{u}_{j,k} = \left(\prod_{l \in S_k} \operatorname{sgn}(\phi_l^{(i)}) \right) \times \min_{l \in S_k} |\phi_l^{(i)}| \tag{18}$$

where S_k is the set of indices in the received sequence which are related to u_j as given in (8).

Step 3. Obtain the reliability measure of the information bit u_j by summing all $2^{m-\gamma}$ votes from Step 2:

$$R_j^{(i)} = \frac{1}{\mu_\gamma} \sum_{k=0}^{2^{m-\gamma}-1} \hat{u}_{j,k}. \tag{19}$$

where μ_γ is a positive integer. If $\gamma = 0$, go to Step 5.

Step 4. For $l = 0, 1, \ldots, n-1$, if $\phi_l^{(i)}$ does not involve the information of any information bit of degree γ, then $\left(\phi_l^{(i)}\right)' = \phi_l^{(i)}$; otherwise,

$$\left(\phi_l^{(i)}\right)' = \left(\operatorname{sgn}(\phi_l^{(i)}) \prod_{j \in M_l^\gamma} \operatorname{sgn}(R_j^{(i)}) \right) \times \min \left\{ |\phi_l^{(i)}|, \min_{j \in M_l^\gamma} |R_j^{(i)}| \right\}. \tag{20}$$

Set $\gamma = \gamma - 1$ and $\boldsymbol{\phi}^{(i)} = \left(\boldsymbol{\phi}^{(i)}\right)'$. Go to Step 2.

Remark 1: For the BF decoding algorithm, because $\phi_l^{(i)} \in \{1, 0, -1\}$ and $R_j^{(i)} \in \mathbb{N}$, (18) and (20) can be reduced to

$$\begin{aligned} \hat{u}_{j,k} &= \left(\prod_{l \in S_k} \operatorname{sgn}(\phi_l^{(i)}) \right) \times \min_{l \in S_k} |\phi_l^{(i)}| \\ &= \prod_{l \in S_k} \phi_l^{(i)} \end{aligned}$$

and

$$\begin{aligned} \left(\phi_l^{(i)}\right)' &= \left(\operatorname{sgn}(\phi_l^{(i)}) \prod_{j \in M_l^\gamma} \operatorname{sgn}(R_j^{(i)}) \right) \times \min \left\{ |\phi_l^{(i)}|, \min_{j \in M_l^\gamma} |R_j^{(i)}| \right\} \\ &= \phi_l^{(i)} \prod_{j \in M_l^\gamma} \operatorname{sgn}(R_j^{(i)}) \end{aligned}$$

which are identical to (13) and (15), respectively.

Remark 2: To obtain better performance, μ_γ can be adjusted for different RM(r, m). We will discuss its impact on the error performance in Section V.

C. MBF AND NMBF DECODING ALGORITHMS

Except for the notations given in Section IV-A, we define the new parameter D_{Th} which is a pre-defined threshold. If the distances calculated in Step 8 of the BF algorithm are larger than D_{Th}, then the corresponding bits are flipped. Therefore, multiple bits can be flipped at a time in each iteration. The MBF algorithm can be obtained by modifying Step 9 of the BF algorithm.

Initialization: Set $i = 1$ and $\gamma = r$. Assign a pre-defined value to D_{Th}. Let the maximum number of iterations be $I_{\max}$.

Step 9. Find the index l' with the maximum value of D_l, i.e.,

$$l' = \underset{0 \le l < n}{\operatorname{argmax}} D_l.$$

Set $i = i + 1$. Then, update the estimated codeword by flipping not only the codeword bit with the largest distance but also the codeword bits whose distances are larger than D_{Th}. That is, for $0 \le l < n$,

$$v_l^{(i)} = \begin{cases} -v_l^{(i-1)}, & \text{if } l = l'; \\ -v_l^{(i-1)}, & \text{if } l \ne l' \text{ and } D_l > D_{\text{Th}}; \\ v_l^{(i-1)}, & \text{if } l \ne l' \text{ and } D_l \le D_{\text{Th}}. \end{cases}$$

Set $\gamma = r$ and $\boldsymbol{\phi}^{(i)} = \boldsymbol{v}^{(i)}$. Go to Step 2.

The major difference between the BF and the MBF algorithms is that the MBF algorithm allows flipping multiple bits in each iteration. Therefore, the required iterations of the MBF algorithm is reduced. In addition, the bit error rate (BER) performance is not affected if an appropriate value of D_{Th} is selected.

Next, by utilizing the concept of normalization in NBF, we can devise the normalized MBF decoding algorithm, i.e., the NMBF decoding algorithm. The NMBF algorithm can perform close to the NBF algorithm with fewer iterations, which will be demonstrated in the next section.

V. SIMULATION RESULTS AND COMPLEXITY ANALYSIS

In this section, we compare the performance and complexity among various decoding algorithms over the AWGN channel. First, we will discuss the influence of the normalization factor μ_γ on BER performance. In the NBF decoding algorithm, $R_j^{(i)}$ is normalized by μ_γ since the number of votes could be very large. Furthermore, in order to balance the number of votes for different degrees, we set $\mu_\gamma = 2\mu_{\gamma+1}$ for $\gamma = 0, 1, \ldots, r-1$. For the ease of presentation, we use "NX" to denote the normalization factor $\mu_0 = X$. Also, "IX" represents that the maximum number of iterations is X. E.g., in Fig. 2, "N4" and "I30" represent that we use $\mu_0 = 4$ and $I_{\max} = 30$, respectively, for the NBF decoding.

Fig. 2 demonstrates the BER performances of the NBF decoding algorithm with different μ_0 for RM(2, 7). From Fig. 2, we can observe that the NBF decoding algorithm with $\mu_0 = 32$ (N32) has the best BER performance in RM(2, 7).

To evaluate the number of iterations for the BF and NBF decoding algorithms to reach the performance of convergence, we provide the BER performances of RM(2, 7) decoded with these two algorithms in Fig. 3 and Fig. 4, respectively. Fig. 3 shows that the performances of RM(2, 7) decoded with the BF decoding algorithm converge at 10 iterations. Similarly, the NBF decoding algorithm requires

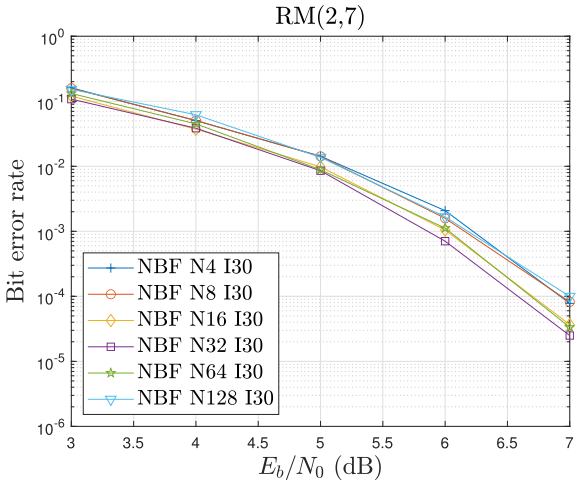

FIGURE 2. BER performances of RM(2, 7) decoded with the NBF algorithm for different μ_0.

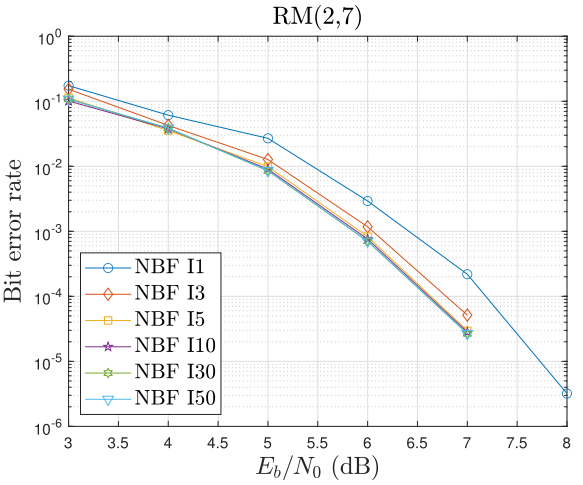

FIGURE 4. BER performances of RM(2, 7) decoded with the NBF algorithm for different iterations.

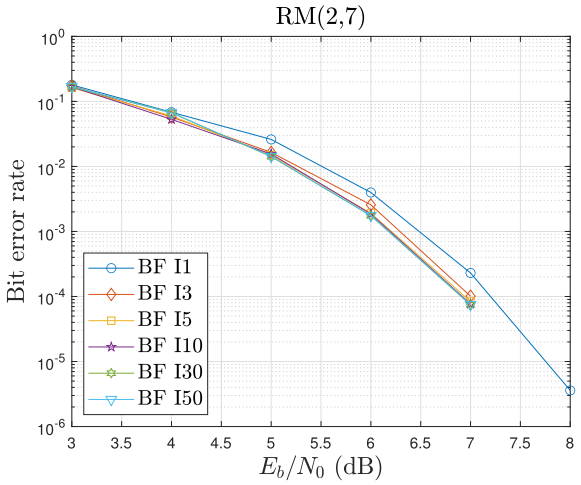

FIGURE 3. BER performances of RM(2, 7) decoded with the BF algorithm for different iterations.

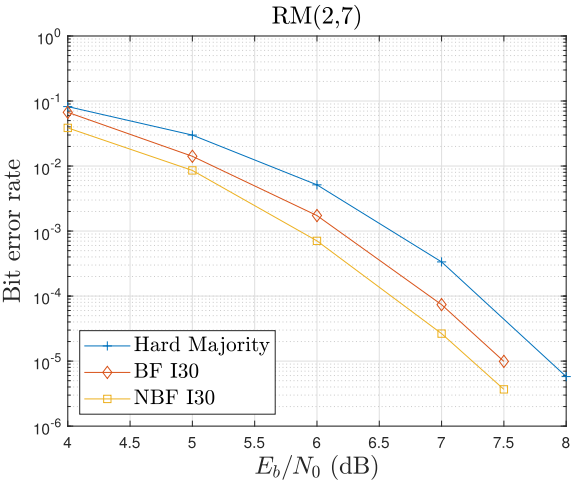

FIGURE 5. BER performances of RM(2, 7) decoded with various hard-decision algorithms.

10 iterations to converge to the best performance according to Fig. 4. Numerical experiments show that a small number of iterations is needed for performance convergence. In the sequel, we set $I_{\max} = 30$ for the BF and NBF decoding algorithms.

Moreover, the BER performances of different hard-decision decoding algorithms are shown in Fig. 5 and Fig. 6. We compare the performances among the proposed BF decoding algorithm, NBF decoding algorithm, and the conventional majority-logic decoding algorithm (It is labeled by the hard majority.) in Figs. 5 and 6. We can see that the BF decoding algorithm outperforms the conventional majority-logic decoding algorithm. At the BER of 10^{-5}, the proposed BF decoding algorithm has at least 0.3 dB gain over the conventional majority-logic decoding algorithm. The NBF decoding algorithm performs better than the BF decoding algorithm. As shown in Fig. 6, the NBF algorithm can have 0.5 dB gain compared to the BF algorithm. Moreover, there are 0.55 dB and 0.81 dB gain in comparison with the conventional majority-logic decoding algorithm for RM(2, 7) and RM(3, 8), respectively.

TABLE 1. The comparison of computational complexities for different hard-decision decoding algorithms.

Decoding Algorithm	Complexity Order	
	Addition	Multiplication
Hard majority	$\sum_{i=0}^{r} \binom{m}{i} \cdot 2^{m-i}$	
HDML	2^{m+K}*	
BF/MBF	$I_{\max} \times (\sum_{i=0}^{r} \binom{m}{i} \cdot 2^{m-i})$	
NBF/NMBF	$I_{\max} \times (\sum_{i=0}^{r} \binom{m}{i} \cdot 2^{m-i})$	$I_{\max} \times K$

The general complexity comparison for hard-decision algorithms is provided in Table 1. From Table 1, we find

*Note that, for the first-order RM codes, the complexity order of the hard-decision maximum likelihood (HDML) algorithm is $O(n \log n)$ when FHT decoding is used [27], [28]. Besides, K denotes the code dimension and 2^K is the number of codewords.

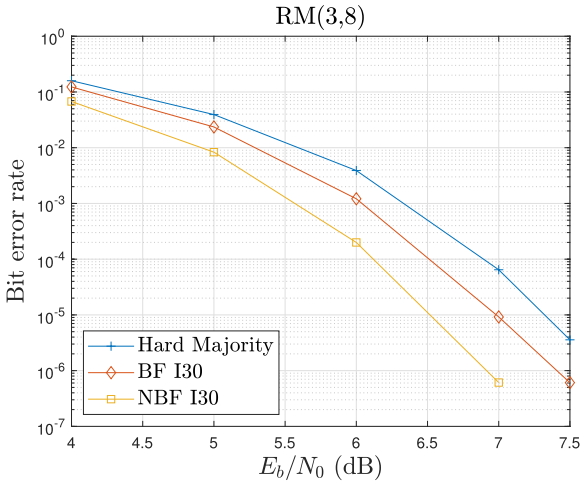

FIGURE 6. BER performances of RM(3, 8) decoded with various hard-decision algorithms.

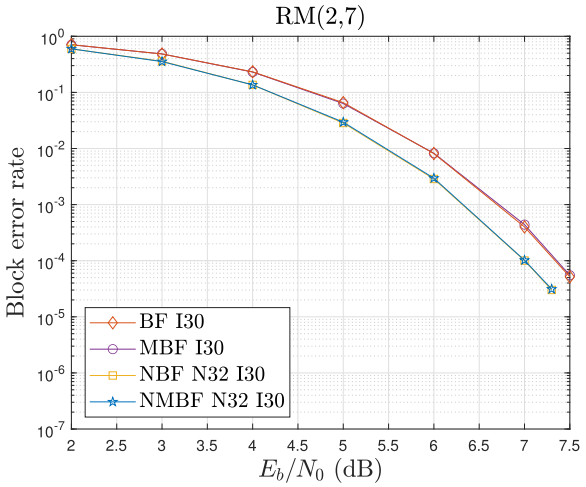

FIGURE 8. BLER performances of RM(2, 7) decoded with various hard-decision algorithms.

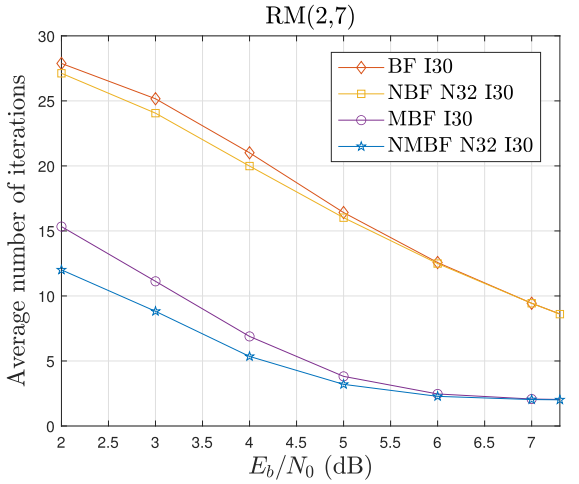

FIGURE 7. The average number of iterations for RM(2, 7) decoded with the BF and NBF algorithms.

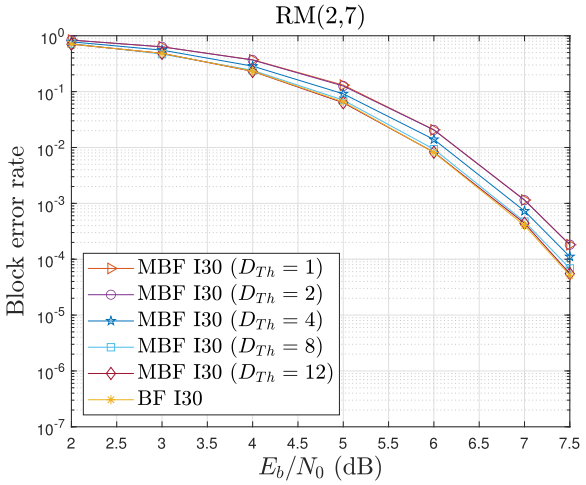

FIGURE 9. BLER performances of RM(2, 7) decoded with the MBF algorithm for different D_{Th}.

that our proposed BF and NBF algorithms have the same complexities of additions. The NBF algorithm outperforms the BF algorithm in BER, but it requires additional multiplications. However, we can choose the normalization factor to be a power of two. Hence, multiplications can be avoided for fixed-point implementation.

The comparison of the average number of iterations for the proposed algorithms is shown in Fig. 7. We set $I_{\max} = 30$. That is, the decoding process will be terminated before or at the 30th iteration. Since the BF and NBF decoding algorithms flip only one bit in each iteration, they require more iterations to successfully decode the erroneous codewords. The MBF and NMBF decoding algorithms require fewer iterations compared with the BF and NBF algorithms, respectively. The number of iterations can be reduced by 40% to 80%. The major complexities of these iterative algorithms are the computations in each iteration. For each iteration, the complexity order of required operations is $O\left(\sum_{i=0}^{r}\binom{m}{i}\cdot 2^{m-i}\right)$. If we can terminate the decoding process with fewer iterations, the overall complexity can be reduced significantly.

Fig. 8 shows the block error rate (BLER) performances of the BF, MBF, NBF, and NMBF algorithms. The NBF and NMBF algorithms have the same BLER performances. The MBF algorithm performs close to the BF algorithm. There is only a 0.02 dB difference at BLER of 10^{-4}. However, at SNR of 7 dB, the average number of iterations for the BF algorithm is 9.44 while that for the MBF algorithm is only 2.07. The reduction ratio is 78.07% ((9.44 − 2.07)/9.44).

Fig. 9 demonstrates the performances of the MBF algorithm with different values of D_{Th}. We observe that if an appropriate threshold D_{Th} is selected, the MBF algorithm can perform close to the BF algorithm.

VI. CONCLUSION

In this paper, we have proposed several hard-decision decoding algorithms for binary RM codes. They are iterative

decoding algorithms that have not been proposed in the literature. The performance can be improved by updating the reliability measures and flipping codeword bits. These algorithms do not need large computational complexity since they can converge very rapidly in a small number of iterations. For hard-decision decoding, our proposed NBF decoding algorithm outperforms the conventional majority-logic decoding algorithm with 0.55 dB to 0.8 dB gain. These proposed algorithms have the same complexity orders. Specifically, the BF and MBF decoding algorithms only require integer additions.

Decoding the RM codes with our iterative decoding algorithms converges very fast with a small number of iterations. Moreover, the proposed MBF/NMBF algorithms can decrease the number of iterations such that the overall complexity can be reduced. The average number of iterations can be reduced by 40% to 80% compared to the BF/NBF algorithms.

Possible future work includes the theoretical analysis of the core parameters of the proposed algorithms, e.g., μ_0 and D_{Th}. Furthermore, the generalization of our proposed algorithms to decode non-binary RM codes is suggested as a future research topic.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to an Associate Editor Dr. Mingjun Dai and an anonymous reviewer for their valuable comments that have significantly improved the quality of this paper.

REFERENCES

[1] D. E. Müller, "Application of Boolean algebra to switching circuit design and to error detection," *Trans. IRE Prof. Group Electron. Comput.*, vol. EC-3, no. 3, pp. 6–12, Sep. 1954.

[2] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IRE Prof. Group Inf. Theory*, vol. 4, no. 4, pp. 38–49, Sep. 1954.

[3] G. Schnabl and M. Bossert, "Soft-decision decoding of Reed–Müller codes as generalized multiple concatenated codes," *IEEE Trans. Inf. Theory*, vol. 41, no. 1, pp. 304–308, Jan. 1995.

[4] A. E. Jones and T. A. Wilkinson, "Performance of Reed–Müller codes and a maximum-likelihood decoding algorithm for OFDM," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 949–952, Jul. 1999.

[5] A. Ashikhmin and S. Litsyn, "Simple MAP decoding of first-order Reed–Müller and Hamming codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1812–1818, Aug. 2004.

[6] I. Dumer and R. Krichevskiy, "Soft-decision majority decoding of Reed–Müller codes," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 258–264, Jan. 2000.

[7] I. Dumer, "Recursive decoding and its performance for low-rate Reed–Müller codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 811–823, May 2004.

[8] I. Dumer, "Soft-decision decoding of Reed–Müller codes: A simplified algorithm," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 954–963, Mar. 2006.

[9] I. Dumer and K. Shabunov, "Soft-decision decoding of Reed–Müller codes: Recursive lists," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1260–1266, Mar. 2006.

[10] M. Ye and E. Abbe, "Recursive projection-aggregation decoding of Reed–Müller codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4948–4965, Aug. 2020.

[11] M. Lian, C. Häger, and H. D. Pfister, "Decoding Reed–Müller codes using redundant code constraints," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 42–47.

[12] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jan. 2009.

[13] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, Oct. 2012.

[14] K. Niu, K. Chen, J. Lin, and Q. T. Zhang, "Polar codes: Primary concepts and practical decoding algorithms," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 192–203, Jul. 2014.

[15] S. A. Hashemi, N. Doan, M. Mondelli, and W. J. Gross, "Decoding Reed–Müller and polar codes by successive factor graph permutations," in *Proc. IEEE 10th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Dec. 2018, pp. 1–5.

[16] E. Arıkan, "A performance comparison of polar codes and Reed–Müller codes," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 447–449, Jun. 2008.

[17] M. Mondelli, S. H. Hassani, and R. L. Urbanke, "From polar to Reed–Müller codes: A technique to improve the finite-length performance," *IEEE Trans. Commun.*, vol. 62, no. 9, pp. 3084–3091, Sep. 2014.

[18] K. Chen, K. Niu, and J. R. Lin, "List successive cancellation decoding of polar codes," *Electron. Lett.*, vol. 48, no. 9, pp. 500–501, Apr. 2012.

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

[20] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.

[21] M. Kamenev, Y. Kameneva, O. Kurmaev, and A. Maevskiy, "A new permutation decoding method for Reed–Müller codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 26–30.

[22] K. Ivanov and R. Urbanke, "Permutation-based decoding of Reed–Müller codes in binary erasure channel," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 21–25.

[23] J. Zhang and M. P. C. Fossorier, "A modified weighted bit-flipping decoding of low-density parity-check codes," *IEEE Commun. Lett.*, vol. 8, no. 3, pp. 165–167, Mar. 2004.

[24] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[25] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.

[26] Y.-T. Ni, C.-Y. Pai, and C.-Y. Chen, "An iterative bit-flipping decoding algorithm for binary Reed–Müller codes," in *Proc. Int. Symp. Inf. Theory Appl. (ISITA)*, Kapolei, HI, USA, Oct. 2020, pp. 1–5.

[27] R. R. Green, "A serial orthogonal decoder," *JPL Space Programs Summary*, vol. 4, nos. 37–39, pp. 247–253, 1966.

[28] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: Elsevier, 1977.

YONG-TING NI received the B.S. and M.S. degrees in engineering science from the National Cheng Kung University (NCKU), Tainan, Taiwan, in 2018 and 2020, respectively. Currently, he is an Engineer with Phison Electronics Corporation, Taiwan.

DUC NHAT NGUYEN received the B.S. and M.S. degrees in engineering science from the National Cheng Kung University (NCKU), Tainan, Taiwan, in 2019 and 2021, respectively. Currently, he is an Engineer with Wistron NeWeb Corporation, Taiwan.

FENG-KAI LIAO received the B.S. degree in mechanical and electro-mechanical engineering from the National Sun Yat-sen University (NSYSU), Kaohsiung, in 2020, and the M.S. degree in engineering science from the National Cheng Kung University (NCKU), Tainan, Taiwan in 2022.

TZU-CHIEH KAO received the B.S. degree in engineering science from the National Cheng Kung University (NCKU), Tainan, in 2021, where he is currently pursuing the Ph.D. degree in engineering science. His research interests include sequence design and error-correcting codes.

CHAO-YU CHEN (Member, IEEE) received the B.S. degree in electrical engineering and the M.S. and Ph.D. degrees in communications engineering from the National Tsing Hua University (NTHU), Hsinchu, Taiwan, in 2000, 2002, and 2009, respectively, under the supervision of Prof. Chi-Chao Chao.

From 2008 to 2009, he was a Visiting Ph.D. Student with the University of California, Davis (with Prof. Shu Lin). From 2009 to 2016, he was the Technical Manager with the Communication System Design Division, Mediatek Inc., Hsinchu. From July 2018 to August 2018, he was with the University of California, Davis, as a Visiting Scholar (with Prof. Shu Lin). Since February 2016, he has been a Faculty Member with the National Cheng Kung University (NCKU), Tainan, Taiwan, where he is currently an Associate Professor with the Department of Engineering Science. His current research interests include sequence design, error-correcting codes, digital communications, and wireless networks.

Dr. Chen was a recipient of the 15th and 20th Y. Z. Hsu Science Paper Award Administered by Far Eastern Y. Z. Hsu Science and Technology Memorial Foundation, Taiwan, in 2017 and 2022, and the Best Paper Award for Young Scholars by the IEEE Information Theory Society Taipei/Tainan Chapter and the IEEE Communications Society Taipei/Tainan Chapter, in 2018. Since January 2019, he has been serving as the Vice Chair for the IEEE Information Theory Society Tainan Chapter.