Low-Complexity Decoding of a Class of Reed-Muller Subcodes for Low-Capacity Channels

Mohammad Vahid Jamali*, Mohammad Fereydounian†, Hessam Mahdavifar*, and Hamed Hassani†
*Department of Electrical Engineering and Computer Science, University of Michigan, {mvjamali,hessam}@umich.edu
†Department of Electrical and Systems Engineering, University of Pennsylvania, {mferey,hassani}@seas.upenn.edu

***Abstract*—We present a low-complexity and low-latency decoding algorithm for a class of Reed-Muller (RM) subcodes that are defined based on the product of smaller RM codes. More specifically, the input sequence is shaped as a multi-dimensional array, and the encoding over each dimension is done separately via a smaller RM encoder. Similarly, the decoding is performed over each dimension via a low-complexity decoder for smaller RM codes. The proposed construction is of particular interest to low-capacity channels that are relevant to emerging low-rate communication scenarios. We present an efficient soft-input soft-output (SISO) iterative decoding algorithm for the product of RM codes and demonstrate its superiority compared to hard decoding over RM code components. The proposed coding scheme has decoding (as well as encoding) complexity of $\mathcal{O}(n \log n)$ and latency of $\mathcal{O}(\log n)$ for blocklength n. This research renders a general framework toward efficiently decoding RM codes.**

I. Introduction

In recent years, there has been significant renewed interest in exploring Reed-Muller (RM) codes, which are one of the oldest families of error-correcting codes [1], [2]. RM codes are closely connected to polar codes [3] in the sense that the generator matrices of both codes are obtained by selecting rows from the same matrix, though by different selection rules. In contract to polar codes, which have channel-specific construction, RM codes have a universal encoding scheme. It is also conjectured that RM codes have similar characteristics to random codes in terms of weight enumeration [4] and scaling laws [5]. While it was proved earlier that RM codes achieve the Shannon capacity of binary erasure channels (BECs) [6], and that of binary symmetric channels (BSCs) at extreme rates converging to zero or one [7], Reeves and Pfister have shown very recently that that RM codes are also able to achieve the capacity of general binary-input memoryless symmetric (BMS) channels [8].

Although RM codes have shown excellent performance under maximum likelihood (ML) decoding, they still lack efficient decoding algorithms for general code parameters. To this end, Dumer's recursive list decoding algorithm [9] provides a complexity-performance trade-off by achieving close-to-ML decoding performance for large enough, e.g., exponential in blocklength, list sizes. Recently, a recursive projection-aggregation (RPA) algorithm was proposed in [10] for decoding RM codes. Despite its explicit structure and excellent decoding performance, the RPA algorithm (in its general form) requires a complexity of $\mathcal{O}(n^r \log n)$ for an RM code of length n and order r. Building upon the projection pruning idea in [10], there has been some recent attempts at reducing the complexity of the RPA algorithm [11], [12], and also applying it in other contexts than communication [13]. Moreover, building upon the computational tree of RM (and polar) codes, a class of neural encoders and decoders has been proposed in [14] via deep learning methods.

In this paper, our goal is to devise an efficient, low-complexity, and low-latency coding scheme for low-capacity channels [15]–[19], that are relevant to emerging low-rate communication scenarios, such as narrowband Internet-of-Things (NB-IoT) [20], deep-space communication, and covert (millimeter-wave) communication [21], among others. Users in these applications typically experience very low signal-to-noise ratios (SNRs). Consequently, reliable communication in such applications requires very large blocklengths, and challenges such as ensuring low latency/complexity and high reliability become more apparent. The current practical approaches for these scenarios are mainly based on large repetitions of a powerful moderate-rate code. While such a construction, i.e., concatenation of a repetition code and a moderate-rate code, results in low-latency codes, it has been shown in [15] that the error performance can be significantly degraded as a result of repetitions. Therefore, using more principled coding schemes to design low-rate codes can potentially lead to significantly more powerful codes. We will employ the recent advances in RM codes as well as product codes to design efficient coding schemes that achieve better performance while maintaining low complexity and low latency. Consequently, our proposed schemes are also of particular application to ultra-reliable and low-latency communications (URLLC).

We build upon product codes [22] to construct a larger RM code based on the product of smaller RM code components. It is well known that building larger codes upon product codes renders several advantages, such as low encoding and decoding complexity, large minimum distances, and a highly parallelized implementation [22]–[24], and it has very recently been shown that it also enables training neural encoders and decoders for relatively large channel codes [25].

While the framework in this paper is applicable to any RM code components, we particularly consider first-order RM codes as the components to take advantage of their ML performance with an $\mathcal{O}(n \log n)$ complexity, enabled by fast Hadamard transform (FHT) [10]. The resulting code will be a subcode of an order-Q RM code, when considering Q component codes in the product; thus, it can be a low-rate code depending on the blocklength of individual code components. We present an efficient soft-input soft-output (SISO) iterative decoding algorithm, enabled by our soft-FHT algorithm over code components.

We show that our decoder maintains a low complexity of $\mathcal{O}(n \log n)$ and a low latency of $\mathcal{O}(\log n)$, regardless of the value of Q. Moreover, our numerical results demonstrate the superiority of the proposed SISO decoder compared to hard decoding over RM code components as well as RPA-like decoding of RM subcodes [11]. We also demonstrate meaningful gains compared to conventional designs such as Turbo-repetition. Lastly, we remark that the proposed methods in this paper can lead to a general framework toward low-complexity decoding of RM codes.

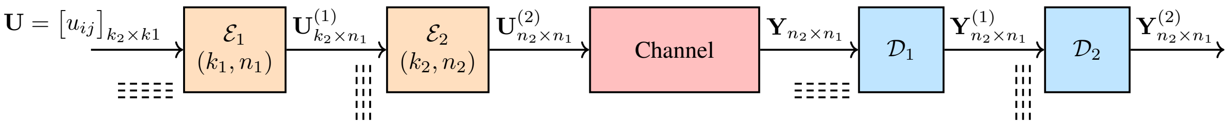
$\mathbf{U} = [u_{ij}]_{k_2 \times k1}$
$\mathcal{E}_1$
(k_1, n_1)
$\mathbf{U}^{(1)}_{k_2 \times n_1}$
$\mathcal{E}_2$
(k_2, n_2)
$\mathbf{U}^{(2)}_{n_2 \times n_1}$
Channel
$\mathbf{Y}_{n_2 \times n_1}$
$\mathcal{D}_1$
$\mathbf{Y}^{(1)}_{n_2 \times n_1}$
$\mathcal{D}_2$
$\mathbf{Y}^{(2)}_{n_2 \times n_1}$

Fig. 1. Demonstration of two-dimensional (2D) product codes. Each q-th encoder $\mathcal{E}_q$ and decoder $\mathcal{D}_q$, $q = 1, 2$, performs encodings and decodings over the q-th dimension of the 2D input arrays.

II. Preliminaries and Setting

A. RM Codes

An RM code is defined in terms of two parameters: (i) a positive integer m that defines the blocklength as $n = 2^m$; and (ii) a nonnegative integer $r \in \{0, 1, \cdots, m\}$, named the order of the RM code, that defines the code dimension k as $k = \sum_{i=0}^{r} \binom{m}{i}$. There are several ways, including the algebraic formulations in [10], to describe an RM code of length $n = 2^m$ and order r, denoted by $\mathcal{RM}(m, r)$. One simple description is through the so-called polarization matrix. Indeed, the generator matrix of an $\mathcal{RM}(m, r)$ code, denoted by $\mathbf{G}_{k \times n}$, can be obtained by choosing rows of the following matrix that have a Hamming weight of at least 2^{m-r}:

$$\mathbf{P}_{n \times n} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes m}, \tag{1}$$

where $\mathbf{F}^{\otimes m}$ is the m-th Kronecker power of a matrix $\mathbf{F}$. The resulting generator matrix $\mathbf{G}_{k \times n}$ can then be partitioned into sub-matrices as

$$\mathbf{G}_{k \times n} = \begin{bmatrix} \mathbf{G}_0 \\ \mathbf{G}_1 \\ \vdots \\ \mathbf{G}_r \end{bmatrix}, \tag{2}$$

where $\mathbf{G}_0$ is a length-n all-one row vector, and $\mathbf{G}_1$ is an $m \times n$ matrix that lists all the $n = 2^m$ unique length-m binary vectors $\{0, 1\}^m$ as the columns. Moreover, $\mathbf{G}_i$, for $1 \leqslant i \leqslant r$, is an $\binom{m}{i} \times n$ matrix whose each row is obtained by the element-wise product of a distinct set of i rows from $\mathbf{G}_1$ [26]. Accordingly, $\mathbf{G}_{k \times n}$ has exactly $\binom{m}{i}$ rows with the Hamming weight $n/2^i$, for $0 \leqslant i \leqslant r$.

B. Product Codes

Fig. 1 illustrates the encoding and decoding procedure for two-dimensional (2D) product codes. Assuming two code components $\mathcal{C}_1 : (k_1, n_1)$ and $\mathcal{C}_2 : (k_2, n_2)$, their product code is constructed by first forming the length-$k_1 k_2$ information sequence as a $k_2 \times k_1$ matrix, and then encoding each row using the first encoder $\mathcal{E}_1$ and each column using the second encoder $\mathcal{E}_2$. It can be shown that in the resulting encoded matrix of size $n_2 \times n_1$ (that can be reshaped to a length-$n_1 n_2$ vector as a codeword), each row is a codeword of $\mathcal{C}_1$ and each column is a codeword of $\mathcal{C}_2$. Therefore, after properly reshaping the noisy codewords at the receiver, the first decoder $\mathcal{D}_1$ decodes the rows of the received 2D array and the second decoder $\mathcal{D}_2$ decodes the columns of its input array. Note that the order of decoders as well as encoders can be interchanged given the symmetry of the problem.

In general, a Q-dimensional product code $\mathcal{C}$ can be constructed by iterating Q codes $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_Q$. More specifically, each q-th encoder, $q = 1, \cdots Q$, encodes the vectors in the q-th dimension of the Q-dimensional input array. Similarly, after properly reshaping the noisy codewords at the receiver, each q-th decoder decodes the noisy vectors on the q-th dimension of the incoming array. Then, assuming $\mathcal{C}_q : (k_q, n_q, d_q, R_q)$ with the generator matrix $\mathbf{G}^{(q)}$, where d and R stand for the minimum distance and rate, respectively, the parameters of the resulting product code $\mathcal{C}$ can be obtained as the product of the parameters of the component codes, i.e.,

$$p = \prod_{q=1}^{Q} p_q, \qquad p \in \{k, n, d, R\}, \tag{3}$$

$$\mathbf{G} = \mathbf{G}^{(1)} \otimes \mathbf{G}^{(2)} \otimes \cdots \otimes \mathbf{G}^{(Q)}. \tag{4}$$

It is known that applying a few decoding iterations (together with SISO decoding) usually improves the decoding performance of product codes [23]. Therefore, often a few, say I, iterations will be applied at the decoder of product codes.

In the special case of RM component codes, the resulting product code is a subcode of a larger RM code, i.e., [26, Corollary 2]

$$\mathcal{RM}(m_1, r_1) \otimes \mathcal{RM}(m_2, r_2) \otimes \cdots \otimes \mathcal{RM}(m_Q, r_Q) \subseteq \mathcal{RM}\Big(\sum_{q=1}^{Q} m_q, \sum_{q=1}^{Q} r_q\Big). \tag{5}$$

Note, based on (3), that the resulting product code has a blocklength of $n_t := 2^{m_t}$, where $m_t := \sum_{q=1}^{Q} m_q$, that is the same as the blocklength of the larger code in the right-hand side (RHS) of (5). Also, given that an $\mathcal{RM}(m, r)$ code has a minimum distance of $d = 2^{m-r}$, one can observe that both the resulting product code and the code in the RHS of (5) have the same minimum distance $d_t := 2^{m_t - r_t}$, where $r_t := \sum_{q=1}^{Q} r_q$. However, the resulting product code has a smaller dimension than the larger RM code, i.e.,

$$\prod_{q=1}^{Q} \left[\sum_{i_l=0}^{r_q} \binom{m_q}{i_l} \right] \leqslant \sum_{i_t=0}^{r_t} \binom{m_t}{i_t}. \tag{6}$$

C. Problem Setting

In this paper, we consider binary phase-shift keying (BPSK) modulation and transmission over additive white Gaussian noise (AWGN) channels. More specifically, we first map each codeword $\mathbf{c}$ to $\tilde{\mathbf{c}} := 1 - 2\mathbf{c}$, before sending it through the channel. The received vector at the channel output is $\mathbf{y} = \tilde{\mathbf{c}} + \mathbf{n}$, where $\mathbf{n}$ is the noise vector whose elements are zero-mean Gaussian random variables with variance σ^2. In this case, the log-likelihood ratio (LLR) vector can be obtained from $\mathbf{y}$ as $\boldsymbol{l} = 2\mathbf{y}/\sigma^2$. Throughout the paper, we define the SNR as $\text{SNR} := 1/(2\sigma^2)$ and the energy-per-bit E_b to the noise ratio as $E_b/N_0 := \text{SNR}/R = n/(2k\sigma^2)$.

III. Proposed Scheme

A. Encoding Scheme

The general encoding procedure has been described in Section II-B. In this paper, we focus on first-order RM code components with two major motivations. First, using (5), the resulting

Algorithm 1 Decoding of Q-Dimensional Product Codes

```
Input: Noisy codeword $\mathbf{y}$, noise variance $\sigma^2$, number of decoding iterations $I$
Output: Decoded codeword $\hat{\mathbf{c}}$
1:  $\boldsymbol{l} \leftarrow 2\mathbf{y}/\sigma^2$        ▷ compute the LLR vector
2:  Properly reshape $\boldsymbol{l}$ to a $Q$-dimensional array $\mathbf{L}$
3:  for $i' = 1, 2, \cdots, I$ do
4:      for $q = 1, 2, \cdots, Q$ do
5:          $\mathbf{L} \leftarrow \mathcal{D}_q(\mathbf{L}, \dim = q)$    ▷ update the vectors on the $q$-th dimension of $\mathbf{L}$ after decoding them using $\mathcal{D}_q$
6:      end for
7:  end for
8:  Properly reshape $\mathbf{L}$ to a length-$n_t$ vector $\hat{\boldsymbol{l}}$
9:  $\hat{\mathbf{c}} \leftarrow 0.5(1 - \mathrm{sign}(\hat{\boldsymbol{l}}))$
10: return $\hat{\mathbf{c}}$
```

product code becomes a subcode of an $\mathcal{RM}(m_t, Q)$ code, and thus a low-rate code for large enough m_t's (compared to Q). Therefore, it aligns with the general objective of the paper, which is to design an efficient, low-complexity, and low-latency coding scheme for emerging low-capacity channels. Second, we can take advantage of the low-complexity FHT decoder for order-1 RM codes, that achieve the same performance as an ML decoder but with an $\mathcal{O}(n \log n)$ complexity instead of an $\mathcal{O}(n^2)$ complexity. In fact, we establish in Section III-D the possibility of decoding the product of any Q first-order RM codes with $\mathcal{O}(n \log n)$ complexity and $\mathcal{O}(\log n)$ latency.

B. Decoding Scheme

It is not hard to show that the vectors on each q-th dimension of the encoded Q-dimensional array, at the output of the product encoder, are codewords of the q-th component code $\mathcal{C}_q$, even if systematic encoders are not used. Therefore, the vectors on each q-th dimension of the received multi-dimensional array, after carefully reshaping the received signal, can be viewed as the noisy codewords of $\mathcal{C}_q$. Accordingly, the decoding procedure can be summarized as Algorithm 1. The reshaping of length-n_t vectors to Q-dimensional arrays and vice versa, performed in lines 2 and 8, respectively, need to be handled carefully with respect to the product encoder architecture (e.g., the parameter of the individual code components, order of the encoders, etc.). Additionally, in line 5, we considered a general decoder $\mathcal{D}_q$ for decoding the noisy codewords of $\mathcal{C}_q$ on the q-th dimension of the LLR array $\mathbf{L}$. In the case of order-1 RM codes, considered in this paper as the component codes, we apply a soft version of the FHT algorithm, developed in Section III-C, to enable an efficient SISO decoding for the underlying product code.

C. Soft-FHT Algorithm

Given $\boldsymbol{l} \in \mathbb{R}^n$ as the vector of channel LLRs, corresponding to the transmission of an (n, k) code with codebook $\mathcal{C}$ over a general binary-input memoryless channel, the ML decoder picks a codeword $\mathbf{c}^*$ according to the following rule [10]

$$\mathbf{c}^* = \underset{\mathbf{c} \in \mathcal{C}}{\mathrm{argmax}} \quad \langle \boldsymbol{l}, 1 - 2\mathbf{c} \rangle, \tag{7}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner-product of two vectors. A naive implementation of the ML decoder then requires an $\mathcal{O}(n 2^k)$ complexity to compute 2^k inner-products between length-n vectors. In particular, for first-order RM codes, $\mathcal{RM}(m, 1)$, that have $2^{m+1} = 2n$ codewords, this is equivalent to an $\mathcal{O}(n^2)$ complexity and an $\mathcal{O}(n)$ latency (when computing all the inner-products in parallel). However, one can do the ML decoding for order-1 codes in a more efficient way via the FHT algorithm. The high-level idea is that half of the $2n$ codewords of an $\mathcal{RM}(m, 1)$ code (in ± 1) are the columns of the standard $n \times n$ Hadamard matrix $\mathbf{H}$, and the other half are columns of $-\mathbf{H}$. Therefore, the ML decoder for order-1 RM codes boils down to the matrix multiplication of the LLR vector $\boldsymbol{l}$ and the Hadamard matrix $\mathbf{H}$, i.e., $\boldsymbol{l}_{\mathrm{WH}} := \boldsymbol{l}\mathbf{H}$, which can be performed in $\mathcal{O}(n \log n)$ complexity and $\mathcal{O}(\log n)$ via the FHT algorithm (see Lemma 3). Since $\boldsymbol{l}_{\mathrm{WH}}$ contains half of the $2n$ inner-products in (7), and the other half are just the elements of $-\boldsymbol{l}_{\mathrm{WH}}$, the FHT version of the ML decoder for first-order RM codes can be obtained as

$$\mathbf{c}^* = \frac{1}{2}[1 - \mathrm{sign}(\boldsymbol{l}_{\mathrm{WH}}(i^*))\mathbf{h}_{i^*}] \quad \text{s.t.} \quad i^* = \underset{i=1,2,\cdots n}{\mathrm{argmax}} \, |\boldsymbol{l}_{\mathrm{WH}}(i)|, \tag{8}$$

where $\boldsymbol{l}_{\mathrm{WH}}(i)$ is the i-th element of the vector $\boldsymbol{l}_{\mathrm{WH}}$, and $\mathbf{h}_i$ is the i-th column of the matrix $\mathbf{H}$.

It will be shown in Section IV that soft decoding of the RM product codes results in a much better performance than their hard decoding. To enable a SISO decoder for RM product codes under consideration, we derive the soft version of the FHT algorithm, referred to as soft-FHT in this paper, for first-order RM code components. We do this in two steps, i.e., first calculating the LLRs of the information bits and then calculating the LLRs of the encoded bits, which will be discussed in the following.

For the AWGN channel model $\mathbf{y} = \tilde{\mathbf{c}} + \mathbf{n}$ and any (n, k) binary linear code $\mathcal{C}$, the LLR $\boldsymbol{l}_{\mathrm{inf}}(i)$ of each i-th information bit u_i, $i = 1, 2, \cdots, k$, can be obtained form the channel LLRs vector $\boldsymbol{l}$, using the *max-log* approximation, as [11]

$$\boldsymbol{l}_{\mathrm{inf}}(i) \approx \max_{\mathbf{c} \in \mathcal{C}_i^0} \, \langle \boldsymbol{l}, 1 - 2\mathbf{c} \rangle \; - \; \max_{\mathbf{c} \in \mathcal{C}_i^1} \, \langle \boldsymbol{l}, 1 - 2\mathbf{c} \rangle, \tag{9}$$

where $\mathcal{C}_i^0$ and $\mathcal{C}_i^1$ denote the subsets of codewords that have $u_i = 0$ and $u_i = 1$, respectively. In the particular case of order-1 codes, one can compute $\boldsymbol{l}_{\mathrm{inf}}$ more efficiently by invoking the FHT algorithm.

The generator matrix $\mathbf{G}_{k \times n}$ of a first-order RM code has one row of Hamming weight n and m rows of weight $n/2$. Assuming that the first row is the all-one row, the calculation of $\boldsymbol{l}_{\mathrm{inf}}$ for u_1 should be carried out differently from the other u_i's. Let $\mathbf{U}_{2^k \times k}$ be a matrix listing all binary vectors of length k as the rows such that the j-th row, $j = 1, 2, \cdots 2^k$, is the binary representation of the number $j - 1$ in k bits with the most significant bit being at the left. The matrix multiplication $\mathbf{C}_{2^k \times n} := \mathbf{U}\mathbf{G}$ (over the binary field) then lists all the codewords in a way that the upper half (the first n rows) of $\tilde{\mathbf{C}} := 1 - 2\mathbf{C}$ is equal to $\mathbf{H}$ and the lower half is equal to $-\mathbf{H}$. Therefore, given that u_1 is equal to zero for the first half of the codewords and equal to one for the second half, we have using (9)

$$\boldsymbol{l}_{\mathrm{inf}}(1) \approx \max_{i'=1,2,\cdots n} \, \boldsymbol{l}_{\mathrm{WH}}(i') \; - \; \max_{i'=1,2,\cdots n} \, -\boldsymbol{l}_{\mathrm{WH}}(i'). \tag{10}$$

To compute the LLRs $\boldsymbol{l}_{\mathrm{inf}}(i)$ for $i = 2, \cdots k$, we only need to find the set of indices of the first half of the codewords that have $u_i = 0$ and $u_i = 1$, denoted by the sets $\mathcal{I}_{0,i} \subset \{1, 2, \cdots n\}$

Algorithm 2 Soft-FHT Algorithm for $\mathcal{RM}(m,1)$ Codes

Input: The channel LLR vector $\boldsymbol{l}$; RM code parameter m, the sets of indices $\mathcal{I}_{0,i}$ and $\mathcal{I}_{1,i}$ for each i-th bit, $i = 2, \cdots m+1$
Output: Soft decisions (i.e., the updated LLR vector) $\hat{\boldsymbol{l}}$

```
$\boldsymbol{l}_{\rm WH} \leftarrow \boldsymbol{l}\mathbf{H}$    ▷ apply FHT algorithm to $\boldsymbol{l}$
Initialize $\boldsymbol{l}_{\rm inf}$ as an all-zero vector of length $m+1$
$\boldsymbol{l}_{\rm inf}(1) \leftarrow$ Eq. (10)    ▷ calculate $\boldsymbol{l}_{\rm inf}(1)$ using (10)
for $i = 2, \cdots, m+1$ do
    $\boldsymbol{l}_{\rm inf}(i) \leftarrow$ Eq. (11)    ▷ calculate $\boldsymbol{l}_{\rm inf}(i)$ using (11)
end for
Initialize $\boldsymbol{l}_{\rm enc}$ as an all-zero vector of length $n := 2^m$
$\mathbf{R} \leftarrow \texttt{repeat}(\boldsymbol{l}_{\rm inf}^T, 1, n)$    ▷ concatenate $n$ copies of $\boldsymbol{l}_{\rm inf}^T$
$\mathbf{V} \leftarrow \mathbf{R} \odot \mathbf{G}$    ▷ element-wise matrix multiplication
for $j = 1, 2, \cdots, n$ do
    $\mathbf{v} \leftarrow$ nonzero elements in the $j$-th column of $\mathbf{V}$
    $\boldsymbol{l}_{\rm enc}(j) \leftarrow \prod_{j'} {\rm sign}(\mathbf{v}(j')) \times \min_{j'} |\mathbf{v}(j')|$    ▷ using (12)
end for
$\hat{\boldsymbol{l}} \leftarrow \boldsymbol{l}_{\rm enc}$
return $\hat{\boldsymbol{l}}$
```

and $\mathcal{I}_{1,i} \subset \{1, 2, \cdots n\}$, respectively[1]. In fact, for any codeword in the first half that has $u_i = 0$ or $u_i = 1$, we have exactly the negative of that codeword in the second half, corresponding to the same realization of the bits $(u_2, u_, \cdots, u_k)$ but with $u_i = 1$ instead of $u_1 = 0$ (recall that the first row of $\mathbf{G}$ is all-one). Therefore, using (9), we have

$$\begin{aligned}\boldsymbol{l}_{\rm inf}(i \neq 1) &\approx \max_{i' \in \mathcal{I}_{0,i}} \pm \boldsymbol{l}_{\rm WH}(i') - \max_{i' \in \mathcal{I}_{1,i}} \pm \boldsymbol{l}_{\rm WH}(i') \\ &= \max_{i' \in \mathcal{I}_{0,i}} |\boldsymbol{l}_{\rm WH}(i')| - \max_{i' \in \mathcal{I}_{1,i}} |\boldsymbol{l}_{\rm WH}(i')|. \quad (11)\end{aligned}$$

Once we have the LLRs of the information bits, we can use them to calculate the LLRs of the encoded bits, denoted by $\boldsymbol{l}_{\rm enc}$. Note that the j-th encoded bit c_j, $j = 1, \cdots, n$, is obtained using the j-th column of $\mathbf{G}$ as $c_j = \sum_{i=1}^{m+1} u_i g_{i,j}$. Therefore, the LLR $\boldsymbol{l}_{\rm enc}(j)$ of the j-th encoded bit can be obtained using the well-known *min-sum* approximation as

$$\boldsymbol{l}_{\rm enc}(j) = \prod_{i \in \Lambda_j} {\rm sign}(\boldsymbol{l}_{\rm inf}(i)) \times \min_{i \in \Lambda_j} |\boldsymbol{l}_{\rm inf}(i)|, \qquad (12)$$

where Λ_j is the set of indices corresponding to the nonzero elements in the j-th column of $\mathbf{G}$. The soft-FHT algorithm is summarized in Algorithm 2.

D. Complexity and Latency Analysis

The following two lemmas establish sufficient conditions for decoding *any* Q-dimensional product code with an $\mathcal{O}(n \log n)$ complexity and an $\mathcal{O}(\log n)$ latency.

Lemma 1. *Any Q-dimensional product code can be decoded with an $\mathcal{O}(n \log n)$ complexity if the component codes can be decoded with an $\mathcal{O}(n \log n)$ complexity.*

Proof: Let $\mathcal{N}(n_q, k_q)$ denote the decoding complexity of the q-th decoder, $q = 1, 2, \cdots, Q$. At each iteration, the decoder needs to perform n_t/n_q decodings over length-n_q vectors, each incurring an $\mathcal{N}(n_q, k_q)$ complexity. Given that there are Q decoders at each iteration, the overall decoding complexity $\mathcal{N}_t$ will be

$$\begin{aligned}\mathcal{N}_t &= I \sum_{q=1}^{Q} \frac{n_t}{n_q} \mathcal{N}(n_q, k_q) \\ &\overset{(a)}{=} I n_t \sum_{q=1}^{Q} \mathcal{O}(\log n_q) \\ &\overset{(b)}{=} I n_t \mathcal{O}(\log n_t), \qquad (13)\end{aligned}$$

where step (a) is by the assumption that the q-th decoder requires $\mathcal{N}(n_q, k_q) = \mathcal{O}(n_q \log n_q)$ complexity, and step (b) follows by $\sum_{q=1}^{Q} \log n_q = \log \prod_{q=1}^{Q} n_q = \log n_t$. As we numerically verify in Section IV, I is a small number (usually less than 5) and does not impact the complexity and latency. ■

Lemma 2. *Any Q-dimensional product code can be decoded with an $\mathcal{O}(\log n)$ latency if the component codes can be decoded with an $\mathcal{O}(\log n)$ latency.*

Proof: Given that all n_t/n_q decodings at each q-th dimension can be executed in parallel, the overall latency is $I \sum_{q=1}^{Q} \mathcal{O}(\log n_q) = I\mathcal{O}(\log n_t)$. ■

Lemma 3. *Besides having an $\mathcal{O}(n \log n)$ complexity, the FHT algorithm performs the ML decoding in $\mathcal{O}(\log n)$ latency for first-order RM codes of blocklength n.*

Proof: The core idea behind the implementation of the FHT algorithm is that the $2^m \times 2^m$ matrix $\mathbf{H}$ can be written as the product of m matrices of size $2^m \times 2^m$, say $\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_m$, each having only two non-zero elements per column [27, page 421]. Therefore,

$$\boldsymbol{l}_{\rm WH} := \boldsymbol{l}\mathbf{H} = \boldsymbol{l}\mathbf{M}_1\mathbf{M}_2 \cdots \mathbf{M}_m \qquad (14)$$

boils down to m matrix multiplications of the form $\mathbf{f}_s := \mathbf{f}_{s-1}\mathbf{M}_s$, $s = 1, 2, \cdots, m$, with $\mathbf{f}_0 := \boldsymbol{l}$. Given that each matrix $\mathbf{M}_s$ has two non-zero elements per column, we only need a single addition/subtraction to compute each of 2^m elements of each vector $\mathbf{f}_s$. Therefore, each $\mathbf{f}_s$ can be computed with $\mathcal{O}(2^m)$ complexity and $\mathcal{O}(1)$ latency (when computing all 2^m elements of $\mathbf{f}_s$ in parallel). Finally, since each of m vectors $\mathbf{f}_s$'s should be computed serially, to get $\boldsymbol{l}_{\rm WH}$, we need $\mathcal{O}(m2^m)$ complexity and $\mathcal{O}(m)$ latency in total. ■

Theorem 4. *Any RM subcode that is obtained as the product of order-1 RM codes can be decoded in $\mathcal{O}(n \log n)$ complexity and $\mathcal{O}(\log n)$ latency via soft-FHT algorithm over component codes.*

Proof: This follows immediately from Lemmas 1 and 2, and noting that the proposed soft-FHT algorithm, similar to the FHT algorithm, requires $\mathcal{O}(n \log n)$ complexity and $\mathcal{O}(\log n)$ latency to decode order-1 RM codes. ■

Theorem 5. *The proposed coding scheme has the encoding complexity of $\mathcal{O}(n \log n)$ and encoding latency of $\mathcal{O}(\log n)$.*

Proof: Note, based on the general encoding procedure of binary linear codes $\mathbf{c} = \mathbf{uG}$, that the encoding complexity and latency are $\mathcal{O}(kn)$ and $\mathcal{O}(k)$, respectively. For order-1 RM code components we have $k = m + 1 = 1 + \log n$, which results in the encoding complexity and latency of $\mathcal{O}(n \log n)$ and $\mathcal{O}(\log n)$, respectively, for the code components. Following

[1] Note that these sets of indices are fixed across the decoding and can be computed before hand to reduce the decoding complexity and latency.

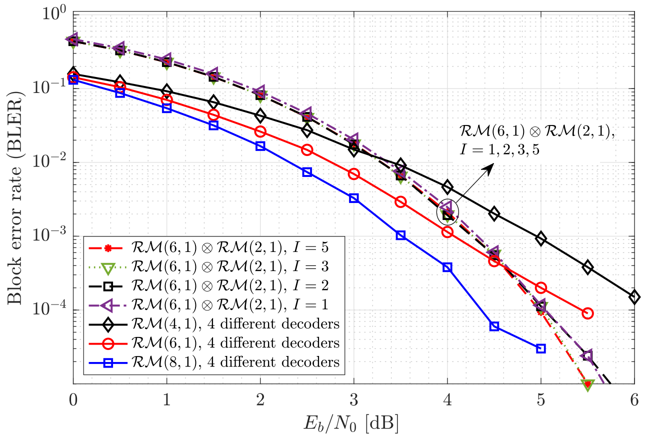

Fig. 2. Accordance of the performance of 4 different decoders, namely FHT, soft-FHT, MAP, and soft-MAP [11], for first-order RM codes. The impact of the number of iterations I is also illustrated for $\mathcal{RM}(6,1)\otimes\mathcal{RM}(2,1)$.

similar procedures to Lemmas 1 and 2, one can show that the overall encoding complexity and latency of any Q-dimensional product code are also $\mathcal{O}(n\log n)$ and $\mathcal{O}(\log n)$, respectively, if the underlying code components have that encoding complexity and latency. ■

IV. Numerical Results

In this section, we present extensive numerical results to study the performance of the proposed coding scheme in various aspects, while focusing on 2D product codes. We first verify the accuracy of the soft-FHT decoder in Fig. 2. As seen, all decoders, namely FHT, soft-FHT, MAP, and soft-MAP [11], match for order-1 RM codes. Fig. 2 also shows the impact of the number of iterations I on the performance of a sample product code, i.e., $\mathcal{RM}(6,1)\otimes\mathcal{RM}(2,1)$. It is observed that not many iterations are required for our proposed decoder.

Note that, as shown in Fig. 1, we first do the decoding over $\mathcal{C}_1$ and then over $\mathcal{C}_2$. As such, decoder $\mathcal{D}_1$ is expected to decode noisier codewords than $\mathcal{D}_2$. Therefore, one needs to use a stronger code (e.g., larger blocklength and/or lower rate) for $\mathcal{C}_1$ compared to $\mathcal{C}_2$. In the context of the product of order-1 RM codes, considered here, this is equivalent to having $m_1 > m_2$. This is confirmed in Figs. 3 and 4 for subcodes of $\mathcal{RM}(13,2)$ and $\mathcal{RM}(8,2)$, obtained as the product of $\mathcal{RM}(m_1,1)\otimes\mathcal{RM}(m_2,1)$ such that $m_1+m_2=13$ and $m_1+m_2=8$, respectively. It is observed that the system performance improves[2] as we increase m_1-m_2.

Fig. 4 also compares the performance of hard decoding with soft decoding for various subcodes of $\mathcal{RM}(8,2)$. The results for hard decoding are obtained by applying the FHT algorithm to the component codes to return hard decisions of the noisy codewords over each dimension. The hard decisions $\hat{\mathbf{y}}_i\in\{0,1\}^{n_i}$, $i=1,2$, are then mapped to $1-2\hat{\mathbf{y}}_i$ before feeding the next FHT decoder. As seen, our SISO decoder significantly outperforms hard decoding. Additionally, the same trend is observed for hard decoding as we increase m_1-m_2.

To demonstrates the efficiency of the proposed SISO decoder, we compare its performance with the sub-RPA algorithm [11], that achieves close-to-ML performance though with full-projection decoding incurring $\mathcal{O}(n^r\log n)$ complexity for a sub-

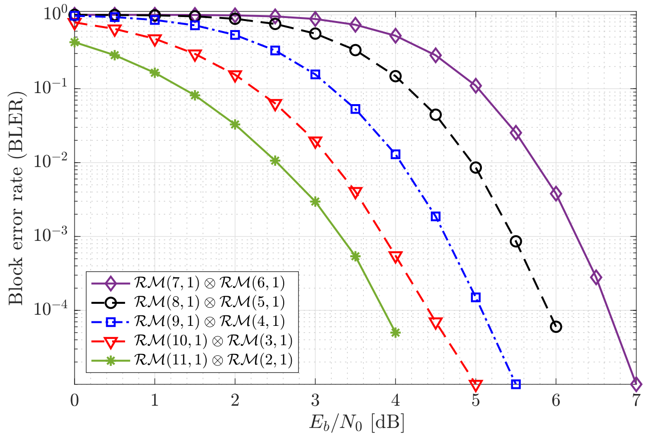

Fig. 3. Impact of code component parameters on the performance of various subcodes of $\mathcal{RM}(13,2)$.

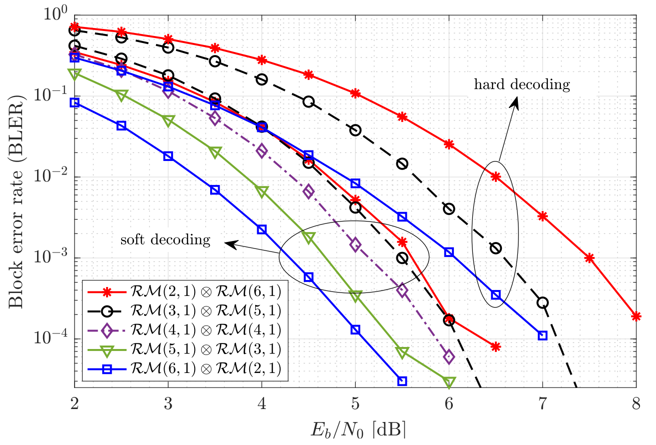

Fig. 4. Impact of code component parameters on the performance of various subcodes of $\mathcal{RM}(8,2)$. The comparison between hard decoding and soft decoding is also included.

code of $\mathcal{RM}(m,r)$. Fig. 5 shows that the full-projection sub-RPA decoding outperforms our low-complexity and low-latency decoder by almost 0.5 dB at the BLER of 10^{-3}, for a subcode of $\mathcal{RM}(8,2)$ obtained as the product of $\mathcal{RM}(6,1)\otimes\mathcal{RM}(2,1)$. However, a more fair comparison is to limit the number of projections in the sub-RPA decoder to a level with a comparable complexity to our SISO decoder. Indeed, the full-projection sub-RPA decoder applies $n-1=255$ projections resulting in $\mathcal{O}(n^2\log n)$ overall complexity. If we apply 5 random projections for the sub-RPA decoder (we tried 8 different random selections of 5 subspaces from 255 possible subspaces), the performance is then inferior to our SISO decoder by a large margin. Also, the sub-RPA algorithm cannot beat our low-complexity decoder even with 16 projections (that is still much more complex than our decoder). Our additional simulations with 32 projections for the sub-RPA decoder show that there are a few (2 out of 8) random trials of the selection of projections that get close to our decoder, while most of the random trials with 64 projections get slightly better than our decoder.

Finally, Fig. 6 compares the performance of the proposed coding scheme with Turbo-repetition and polar codes. the Turbo-repetition is obtained by repeating a $(120,40)$ Turbo code 68 times to obtain a $(40,8160)$ code. It is observed that the equivalent RM product codes have sharper slopes and achieve much better performances over moderate to low BLER regimes,

[2] Note that the channel capacity is approximately linear in SNR over low-capacity regimes. Therefore, based on the definition of E_b/N_0, it is logical to compare the performance of different low-rate codes in terms of E_b/N_0.

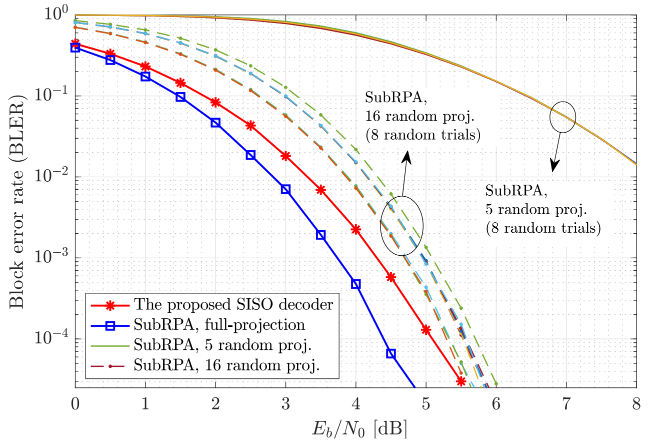

Fig. 5. Comparison of the proposed SISO decoder with the sub-RPA algorithm [11] with full-projection as well as 5 and 16 random projections. Product code $\mathcal{RM}(6,1) \otimes \mathcal{RM}(2,1)$ is considered.

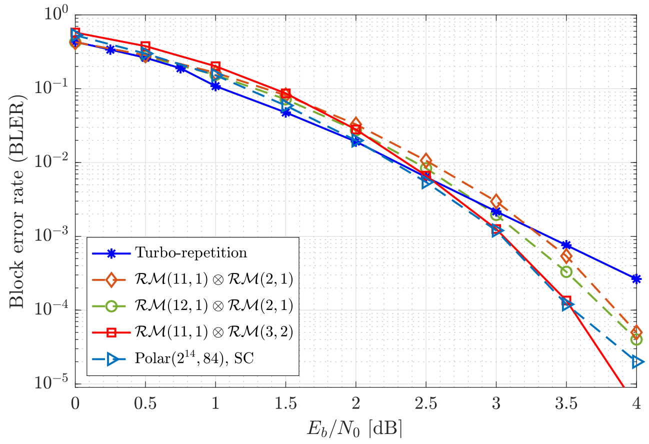

Fig. 6. Comparison of the proposed coding scheme with Turbo-repetition and polar under successive cancellation (SC) decoding.

thus demonstrating potential applications to URLLC. Fig. 6 also shows that it is useful to increase the rate of the second component when the first component is a strong enough code to support such a high rate. For example, $\mathcal{RM}(11,1) \otimes \mathcal{RM}(3,2)$ (via soft-MAP [11] over $\mathcal{RM}(3,2)$) achieves almost 0.3 dB gain over $\mathcal{RM}(12,1) \otimes \mathcal{RM}(2,1)$ and 0.9 dB over Turbo-repetition at the BLER of 10^{-4} (note that the performance of Turbo-repetition does not change in E_b/N_0 by doubling the number of repetitions as the SNR will increase by the same factor of two that the rate is decreased). Moreover, our $\mathcal{RM}(11,1) \otimes \mathcal{RM}(3,2)$ code achieves the same performance as the equivalent polar code of parameters $(2^{14}, 84)$, under successive cancellation (SC) decoding, despite its much lower latency. List decoding of the proposed RM product codes to further improve their performance is a subject of future research.

V. Conclusions

In this paper, we presented a low-complexity and low-latency coding scheme, based on the product of smaller (particularly, first-order) RM code components, with particular applications to emerging low-capacity scenarios. We proposed an iterative SISO decoder enabled by soft-FHT decoding of code components. It was shown that the proposed coding scheme requires $\mathcal{O}(n \log n)$ complexity and $\mathcal{O}(\log n)$ latency for both encoding and decoding. Through extensive numerical results, we studied the performance and efficiency of the proposed coding scheme in various aspects. Given the recent breakthrough result [8] proving the capacity-achievability of RM codes over any BMS channel, the design of efficient decoders for RM codes becomes even more substantial than ever. And, based on the fact that any RM code can be written as the union of RM subcodes, defined as the product of smaller RM codes [26], we believe that the research in this paper opens a new framework toward efficient decoding of RM codes.

References

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

[2] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *IRE Trans. Inf. Theory*, no. 3, pp. 6–12, 1954.

[3] 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, 2009.

[4] T. Kaufman, S. Lovett, and E. Porat, "Weight distribution and list-decoding size of Reed-Muller codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 2689–2696, 2012.

[5] H. Hassani, S. Kudekar, O. Ordentlich, Y. Polyanskiy, and R. Urbanke, "Almost optimal scaling of Reed-Muller codes on BEC and BSC channels," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 311–315.

[6] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Şaşoğlu, and R. L. Urbanke, "Reed-Muller codes achieve capacity on erasure channels," *IEEE Trans. Inf Theory*, vol. 63, no. 7, pp. 4298–4316, 2017.

[7] E. Abbe, A. Shpilka, and A. Wigderson, "Reed-Muller codes for random erasures and errors," *IEEE Trans. Inf. Theory*, vol. 61, no. 10, pp. 5229–5252, 2015.

[8] G. Reeves and H. D. Pfister, "Reed-Muller codes achieve capacity on BMS channels," *arXiv preprint arXiv:2110.14631*, 2021.

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

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

[11] M. V. Jamali, X. Liu, A. V. Makkuva, H. Mahdavifar, S. Oh, and P. Viswanath, "Reed-Muller subcodes: Machine learning-aided design of efficient soft recursive decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 1088–1093.

[12] D. Fathollahi, N. Farsad, S. A. Hashemi, and M. Mondelli, "Sparse multi-decoder recursive projection aggregation for Reed-Muller codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 1082–1087.

[13] M. Soleymani, M. V. Jamali, and H. Mahdavifar, "Coded computing via binary linear codes: Designs and performance limits," *IEEE J. Sel. Areas Inf. Theory,*, vol. 2, no. 3, pp. 879–892, 2021.

[14] A. V. Makkuva, X. Liu, M. V. Jamali, H. Mahdavifar, S. Oh, and P. Viswanath, "KO codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*. PMLR, 2021, pp. 7368–7378.

[15] M. Fereydounian, M. V. Jamali, H. Hassani, and H. Mahdavifar, "Channel coding at low capacity," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2019, pp. 1–5.

[16] M. V. Jamali and H. Mahdavifar, "Massive coded-NOMA for low-capacity channels: A low-complexity recursive approach," *IEEE Trans. Commun.*, vol. 69, no. 6, pp. 3664–3681, 2021.

[17] M. V. Jamali and H. Mahdavifar, "A low-complexity recursive approach toward code-domain NOMA for massive communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.

[18] I. Dumer and N. Gharavi, "Codes for high-noise memoryless channels," in *Proc. IEEE Int. Symp. Inf. Theory Appl. (ISITA)*, 2020, pp. 101–105.

[19] ——, "Codes approaching the Shannon limit with polynomial complexity per information bit," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 238–243.

[20] R. Ratasuk, N. Mangalvedhe, Y. Zhang, M. Robert, and J.-P. Koskinen, "Overview of narrowband IoT in LTE Rel-13," in *Proc. IEEE Conf. Standard Commun. Netw. (CSCN)*. IEEE, 2016, pp. 1–7.

[21] M. V. Jamali and H. Mahdavifar, "Covert millimeter-wave communication: Design strategies and performance analysis," *IEEE Trans. Wireless Commun.*, Oct. 2021.

[22] P. Elias, "Error-free coding," *Research Laboratory of Electronics, Massachusetts Institute of Technology*, 1954.

[23] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003–1010, 1998.

[24] H. Mukhtar, A. Al-Dweik, and A. Shami, "Turbo product codes: Applications, challenges, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 3052–3069, 2016.

[25] M. V. Jamali, H. Saber, H. Hatami, and J. H. Bae, “ProductAE: Towards training larger channel codes based on neural product codes,” *arXiv preprint arXiv:2110.04466*, 2021.

[26] A. J. Salomon and O. Amrani, “Augmented product codes and lattices: Reed-Muller codes and Barnes-Wall lattices,” *IEEE Trans. Inf. Theory*, vol. 51, no. 11, pp. 3918–3930, 2005.

[27] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. Amsterdam, The Netherlands: North-Holland, 1977.