List-GRAND: A Practical Way to Achieve Maximum Likelihood Decoding

Syed Mohsin Abbas, Marwan Jalaleddine, *Graduate Student Member, IEEE*, and Warren J. Gross, *Senior Member, IEEE*

***Abstract*— Guessing random additive noise decoding (GRAND) is a recently proposed universal maximum likelihood (ML) decoder for short-length and high-rate linear block codes. Soft-GRAND (SGRAND) is a prominent soft-input GRAND variant, outperforming the other GRAND variants in decoding performance; nevertheless, SGRAND is not suitable for parallel hardware implementation. Ordered Reliability Bits-GRAND (ORBGRAND) is another soft-input GRAND variant that is suitable for parallel hardware implementation; however, it has lower decoding performance than SGRAND. In this article, we propose List-GRAND (LGRAND), a technique for enhancing the decoding performance of ORBGRAND to match the ML decoding performance of SGRAND. Numerical simulation results show that LGRAND enhances ORBGRAND's decoding performance by 0.5–0.75 dB for channel codes of various classes at a target frame error rate (FER) of 10^{-7}. For linear block codes of length 127/128 and different code rates, LGRAND's VLSI implementation can achieve an average information throughput of 47.27–51.36 Gb/s. In comparison to ORBGRAND's VLSI implementation, the proposed LGRAND hardware has a 4.84% area overhead.**

***Index Terms*— Guessing random additive noise decoding (GRAND), maximum likelihood (ML) decoding, ordered reliability bits GRAND (ORBGRAND), soft GRAND (SGRAND), ultra reliable and low-latency communication (URLLC).**

I. Introduction

ULTRA reliable low-latency communication (URLLC) is considered an important use case of 5G and future communication networks because it enables applications that require high reliability and very low latency. Some of these emerging applications include augmented and virtual reality [1], intelligent transportation systems (ITS) [2], the Internet of Things (IoT) [3], [4], and machine-to-machine communication (M2M) [5]. These novel applications benefit from the use of short-length, high-rate error-correcting codes. Guessing random additive noise decoding (GRAND) [6] is a recently proposed universal maximum likelihood (ML) decoding technique for these short-length and high-rate linear block codes. GRAND is a noise-centric and code-agnostic decoder, which implies that, unlike traditional decoding techniques, GRAND attempts to guess the noise that corrupted the codeword during transmission through the communication channel. Therefore, GRAND can be used with both structured codes and unstructured codes, which are stored in a dictionary, provided that there exists a method to verify codebook membership of a given vector [7]. Furthermore, when used with random codebooks, GRAND achieves capacity [6].

GRAND and its variants work on the premise of guessing the channel-induced noise by first generating test error patterns (TEPs) ($\boldsymbol{e}$), then applying them to the received hard-demodulated vector of channel observation values ($\hat{\boldsymbol{y}}$), and finally querying the resulting vector ($\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$) for codebook membership. The order in which these TEPs are generated is the primary difference between the GRAND variants. GRAND with ABandonment (GRANDAB) [6], [8] is a hard decision input variant that generates TEPs in ascending Hamming weight order, up to the weight AB. Ordered reliability bits GRAND (ORBGRAND) [9] and soft GRAND (SGRAND) [10] are soft-input variants that efficiently leverage soft information [channel observation values ($\boldsymbol{y}$)], resulting in improved decoding performance compared to the decoding performance of the hard-input GRANDAB. In comparison to other code-agnostic channel code decoders such as brute-force ML decoding and ordered statistic decoding (OSD) [11], [12], GRAND offers a low-complexity decoding solution for short-length and high-rate channel codes. The scope of this work is restricted to short channel codes with high code rates because GRAND and its variants are proposed as ML decoders for short-length and high-rate channel codes.

Fig. 1(a) compares the decoding performance of different variants of GRAND with Berlekamp-Massey (B-M) [13], [14] decoder, OSD (Order $=$ 2), and ML decoding of Bose–Chaudhuri–Hocquenghem (BCH) code (127, 106). The ML decoding results are obtained from [15]. The numerical simulation results presented in this work are based on BPSK modulation over a additive white Gaussian noise (AWGN) channel. While both soft-input variants of GRAND (ORBGRAND and SGRAND) outperform the hard-input B-M decoder, SGRAND achieves ML performance similar to OSD. Fig. 1(b) compares the decoding performance of various GRAND variants for decoding 5G new radio (NR) cyclic redundancy check (CRC)-aided polar (CA-Polar) code (128, 105 + 11). Furthermore, the decoding performance of state-of-the-art soft-input decoders such as the CRC-aided successive cancellation list (CA-SCL) decoder [16], [17] and OSD (Order $=$ 2) is included for reference. The ORBGRAND and SGRAND outperform the hard-input

Manuscript received 6 May 2022; revised 6 August 2022 and 12 October 2022; accepted 13 November 2022. Date of publication 28 November 2022; date of current version 28 December 2022. *(Corresponding author: Syed Mohsin Abbas.)*

The authors are with the Department of Electrical and Computer Engineering, McGill University, Montréal, QC H3A 0G4, Canada (e-mail: syed.abbas@mail.mcgill.ca; marwan.jalaleddine@mail.mcgill.ca; warren.gross@mcgill.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2022.3223692.

Digital Object Identifier 10.1109/TVLSI.2022.3223692

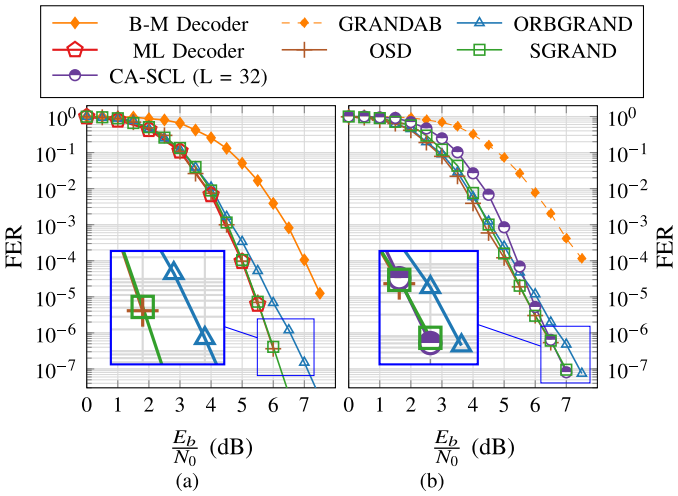

Fig. 1. Comparison of the decoding performance of different GRAND variants for (a) BCH Code (127, 106) and (b) 5G-NR Polar Code (128, 105 + 11).

GRANDAB (AB = 3) variant in decoding performance, and the SGRAND achieves ML decoding performance similar to OSD, as shown in Fig. 1(b).

As shown in Fig. 1, SGRAND outperforms the other GRAND variants in terms of decoding performance; however, SGRAND is not suitable for parallel hardware implementation. The TEPs generated by SGRAND are interdependent, and their query order varies with each received vector of channel observation values ($\boldsymbol{y}$) (explained in Section III-A). As a result, SGRAND does not lend itself to efficient parallel hardware implementation, and a sequential hardware implementation will result in high decoding latency, rendering it unsuitable for applications which require ultra low latency. The ORBGRAND, on the other hand, generates TEPs in a predetermined logistic weight (LW) order based on integer partitioning. The TEPs generated are mutually independent and can be generated in parallel. ORBGRAND is thus highly parallelizable and well suited to parallel hardware implementation. In [18], a VLSI architecture for ORBGRAND for $n = 128$ is presented, which can perform 1.16×10^5 codebook membership queries in 4226 clock cycles due to parallel generation of TEPs in hardware.

In this article, we propose List-GRAND (LGRAND), a technique for boosting the decoding performance of ORBGRAND in order to achieve ML decoding performance comparable to SGRAND. The idea behind the proposed LGRAND is to generate a list during the decoding process and choose the candidate with the highest likelihood to be the final one. The proposed LGRAND technique is not limited to the ORBGRAND TEP generation; rather, it can be used with any GRAND variant that uses a suboptimal TEPs generation scheme. However, since the ORBGRAND TEP generation is hardware friendly and the ORBGRAND archives good error decoding performance as a soft-input decoder, we use the ORBGRAND as a baseline to present our proposed LGRAND technique to achieve the decoding performance similar to an ML decoder such as SGRAND.

The proposed LGRAND introduces parameters that can be adjusted to match the target decoding performance and complexity budget of a specific application. For channel codes of different classes (BCH codes [19], [20], CRC codes [21] and CA-Polar codes [22]), the proposed LGRAND achieves decoding performance similar to SGRAND. LGRAND also achieves a 0.5–0.75 dB performance gain over ORBGRAND at a target frame error rate (FER) of 10^{-7}. Furthermore, because the proposed LGRAND algorithm is based on ORBGRAND, LGRAND lends itself well to parallel hardware implementation. The VLSI implementation results show that the proposed LGRAND can achieve an average information throughput of 47.27–51.36 Gb/s for linear block codes of length 127/128 and different code rates. In comparison to the ORBGRAND hardware, the proposed LGRAND hardware has a 4.84% area overhead. Furthermore, as long as the length and rate constraints are met, the proposed LGRAND hardware can be used to decode any code.

The rest of this work is structured as follows. Section II contains preliminary information on GRAND and ORBGRAND. Section III discusses the generation of TEPs as well as the computational complexity of GRAND and its variants. Section IV presents the proposed LGRAND technique, which is used to improve the decoding performance of ORBGRAND. The numerical simulation results are presented in Section V. Section VI describes the proposed LGRAND hardware architecture as well as the implementation results. Finally, in Section VII, concluding remarks are presented.

II. Preliminaries

A. Notations

Matrices are denoted by a bold upper-case letter ($\boldsymbol{M}$), while vectors are denoted with bold lower-case letters ($\boldsymbol{v}$). The transpose operator is represented by $^\top$. The number of k-combinations from a given set of n elements is noted by $\binom{n}{k}$. $\mathbb{1}_n$ is the indicator vector where all locations except the nth are 0 and the nth is 1. All the indices start at 1. For this work, all operations are restricted to the Galois field with two elements, noted $\mathbb{F}_2$. Furthermore, we restrict ourselves to (n, k) linear block codes, where n is the code length and k is the code dimension.

B. GRAND Decoding

For an (n, k) linear block code with codebook $\mathcal{C}$, a vector $\boldsymbol{u}$ of size k maps to a vector $\boldsymbol{c}$ of size n, and the ratio $R \triangleq (k/n)$ is known as the code rate. Furthermore, there exists a $k \times n$ matrix $\boldsymbol{G}$ called generator matrix ($\boldsymbol{c} \triangleq \boldsymbol{u} \cdot \boldsymbol{G}$) and an $(n-k) \times n$ matrix $\boldsymbol{H}$ called parity-check matrix.

GRAND [6] attempts to guess the noise that corrupted the transmitted codeword ($\boldsymbol{c}$) as it passed through the communication channel. To that end, GRAND first generates the TEPs ($\boldsymbol{e}$) starting from the most likely up to the least likely pattern taking into account the channel model. This is followed by combining the generated TEPs with the hard decided received vector of channel observation values (demodulated symbols) $\hat{\boldsymbol{y}}$, and evaluating if the resulting vector $\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$ is a member of the codebook ($\mathcal{C}$). If the resulting vector is a member of the codebook, the decoding is assumed to be successful, and

Algorithm 1 ORBGRAND Algorithm

Input: $\boldsymbol{y}$, $\boldsymbol{H}$, $\boldsymbol{G}^{-1}$, $LW_{\max}$, $HW_{\max}$
Output: $\hat{\boldsymbol{u}}$, $\boldsymbol{e}$, LW

1 **if** $\boldsymbol{H} \cdot \hat{\boldsymbol{y}}^{\top} == \boldsymbol{0}$ **then**
2 **return** $\hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{y}} \cdot \boldsymbol{G}^{-1}$
3 **else**
4 $\boldsymbol{ind} \leftarrow$ `sortChannelObservationValues(`$\boldsymbol{y}$`)` // $|\boldsymbol{y}_i| \leq |\boldsymbol{y}_j| \quad \forall i < j$
5 $\boldsymbol{e} \leftarrow \boldsymbol{0}$
6 **for** $i \leftarrow 1$ *to* $LW_{\max}$ **do**
7 $\boldsymbol{S} \leftarrow$ `generateAllIntPartitions(`i`)` // $(\lambda_1, \lambda_2, \ldots, \lambda_P) \vdash i \;\; , \;\; \forall P \in [1, HW_{\max}]$
8
9 **forall** $\boldsymbol{l}$ *in* $\boldsymbol{S}$ **do**
10 $\boldsymbol{e} \leftarrow$ `generateErrorPattern(`$\boldsymbol{l}$,$\boldsymbol{ind}$`)`
11 **if** $\boldsymbol{H} \cdot (\hat{\boldsymbol{y}} \oplus \boldsymbol{e})^{\top} == \boldsymbol{0}$ **then**
12 $\hat{\boldsymbol{u}} \leftarrow (\hat{\boldsymbol{y}} \oplus \boldsymbol{e}) \cdot \boldsymbol{G}^{-1}$
13 **return** $\hat{\boldsymbol{u}}$

$\boldsymbol{e}$ is declared as the guessed noise, whereas $\hat{\boldsymbol{c}} \triangleq \hat{\boldsymbol{y}} \oplus \boldsymbol{e}$ is outputted as the estimated codeword.

GRAND can be used with any codebook as long as there is a method for validating a vector's codebook membership. For any linear codebook ($\mathcal{C}$), the codebook membership of a vector can be verified using the underlying code's parity-check matrix $\boldsymbol{H}$, as follows:

$$\forall\, \boldsymbol{c} \in \mathcal{C}, \quad \boldsymbol{H} \cdot \boldsymbol{c}^{\top} = \boldsymbol{0}. \tag{1}$$

For other nonstructured codebooks, stored in a dictionary, the codebook membership of a vector can be checked with a dictionary lookup. For the rest of the discussion, we restrict ourselves to (n, k) linear block codes.

C. ORBGRAND Decoding

ORBGRAND [9] is centered around generating distinct integer partitions of a particular LW, and these integer partitions are then used to generate TEPs ($\boldsymbol{e}$). The LW corresponds to the sum of the indices of nonzero elements in the TEPs [9]. For example, $\boldsymbol{e} = [1, 1, 0, 0, 1, 0]$ has a Hamming weight of 3, whereas the LW is $1 + 2 + 5 = 8$.

An integer partition $\boldsymbol{\lambda}$ of a positive integer m, noted $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_P) \vdash m$ where $\lambda_1 > \lambda_2 > \cdots > \lambda_P$ is the multiset of positive integers λ_i ($\forall i \in [1, P]$) that sum to m. If all parts λ_i ($\forall i \in [1, P]$) of the integer partition are different, the partition is called distinct. Note that the Hamming weight of the generated TEP ($\boldsymbol{e}$) obtained from an integer partition $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_P)$ with P elements is P. ORBGRAND considers the maximum LW for an (n, k) linear block code to be $(n(n+1)/2)$ [$\text{LW}_{\max} = (n(n+1)/2)$]. Furthermore, the generated TEPs have a maximum Hamming weight of n ($\text{HW}_{\max} = n$). It should be noted that only distinct integer partitions are considered for generating TEPs, and all the parts (λ_i) of the integer partitions are less than or equal to n [$\lambda_i \leq n$ ($\forall i \in [1, P]$)] [9].

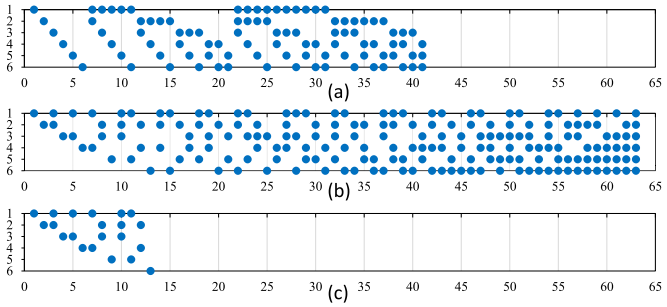

Fig. 2. TEP generation for GRAND for $n = 6$. (a) Top: TEP generation for GRANDAB (AB = 3). (b) Middle: TEP generation for ORBGRAND ($\text{LW}_{\max} = 21$). (c) Bottom: TEP generation for ORBGRAND ($\text{LW}_{\max} = 6$).

Algorithm 1 summarizes the steps of the ORBGRAND. The inputs to the algorithm are the vector of channel observation values [log-likelihood ratios (LLRs)] $\boldsymbol{y}$ of size n, an $(n-k) \times n$ parity-check matrix $\boldsymbol{H}$, an $n \times k$ matrix $\boldsymbol{G}^{-1}$, where $\boldsymbol{G}^{-1}$ refers to the inverse of the generator matrix $\boldsymbol{G}$ of the code such that $\boldsymbol{G} \cdot \boldsymbol{G}^{-1}$ is the $k \times k$ identity matrix, and the maximum Hamming weight $\text{HW}_{\max}$ as well as the maximum LW considered $\text{LW}_{\max}$.

The algorithm begins with evaluating the received vector's ($\hat{\boldsymbol{y}}$) codebook membership (line 1); if it is satisfied (1), the original message is retrieved (line 2); otherwise, $\boldsymbol{y}$ is sorted in ascending order according to the absolute values of the LLRs ($|\boldsymbol{y}_i| \leq |\boldsymbol{y}_j| \;\; \forall i < j$), and the relevant indices are recorded into a permutation vector $\boldsymbol{ind}$ (line 4). This is followed by generating all the integer partitions for each LW (line 7). The function *generateErrorPattern* generates a TEP ($\boldsymbol{e}$) using integer partition ($\boldsymbol{l}$), which is then ordered using the permutation vector $\boldsymbol{ind}$ (line 10). For instance, the generated error pattern, for $n = 6$ with $\boldsymbol{l} = (1, 2)$ and $\boldsymbol{ind} = (2, 6, 5, 4, 3, 1)$, will be $\boldsymbol{e} = (0, 1, 0, 0, 0, 1)$. The generated TEPs are then applied sequentially to the hard decision vector ($\hat{\boldsymbol{y}}$), which is obtained from $\boldsymbol{y}$. The resulting vector ($\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$) is then queried for codebook membership (line 11). If the codebook membership criterion (1) is met, then $\boldsymbol{e}$ is the guessed noise and $\hat{\boldsymbol{c}} \triangleq \hat{\boldsymbol{y}} \oplus \boldsymbol{e}$ is the estimated codeword. Otherwise, either the remaining error patterns for that LW or larger LWs are considered. Finally, using $\boldsymbol{G}^{-1}$ (line 12), the original message ($\hat{\boldsymbol{u}}$) is retrieved from the estimated codeword, and the decoding process is terminated.

III. GRAND: Analysis of TEP Generation and Computational Complexity

This section describes the TEP ($\boldsymbol{e}$) generating scheme and computational complexity analysis for GRAND and its variants.

A. TEP Generation for GRAND

Combining a TEP with the hard-demodulated received vector $\hat{\boldsymbol{y}}$ corresponds to flipping certain bits of that vector ($\hat{\boldsymbol{y}}$). GRANDAB [6] is a hard decision input version of GRAND that generates TEPs in increasing Hamming weight order up to a Hamming weight AB. The TEPs generated in Hamming weight order for $n = 6$ and AB = 3 are depicted in Fig. 2(a), where each column corresponds to a TEP and

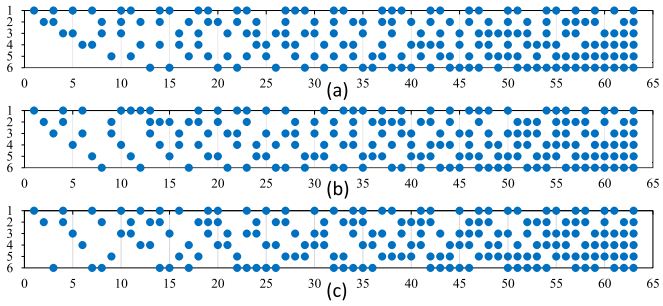

Fig. 3. TEP generation for SGRAND. (a) Top: ML order for $\boldsymbol{y_1} = [1.0, 2.1, 3.2, 4.3, 5.4, 6.5]$. (b) Middle: ML order for $\boldsymbol{y_2} = [1.3, 2.5, 3.6, 4.9, 5.8, 6.1]$. (c) Bottom: ML order for $\boldsymbol{y_3} = [1.8, 2.0, 3.9, 4.1, 5.6, 3.3]$.

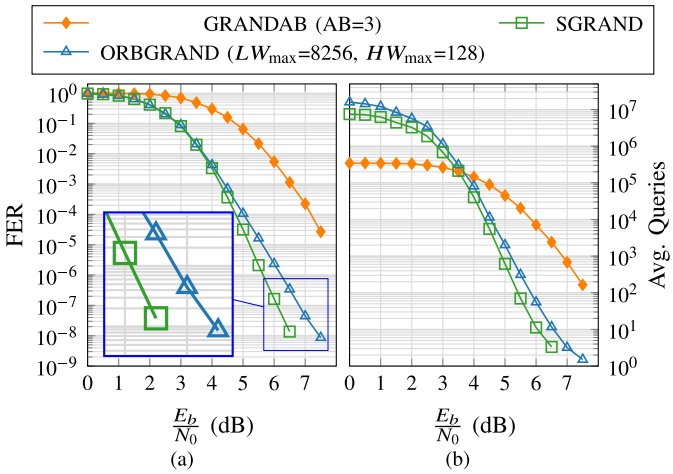

Fig. 4. Comparison of decoding performance and average complexity GRANDAB, ORBGRAND, and SGRAND ($\text{Queries}_{\max} = 5 \times 10^7$) decoding of CRC Code (128, 104). (a) FER. (b) Avg. queries.

a dot corresponds to a flipped bit location of the received hard-demodulated vector ($\hat{\boldsymbol{y}}$).

On the other hand, ORBGRAND is a soft-input GRAND variant that uses the LW order to generate TEPs. The TEPs generated by ORBGRAND with $\text{LW}_{\max} = 21$ are shown in Fig. 2(b). The integer partitions of an integer m ($\forall m \in [1, 21]$) are generated sequentially and these integer partitions are then used to generate TEPs. For $n = 6$ and $\text{LW}_{\max} = 21$, 63 TEPs are generated with the maximum Hamming weight ($\text{HW}_{\max}$) of the generated TEPs being 6. However, when $\text{LW}_{\max}$ is reduced from 21 to 6 the number of TEPs is reduced to 13, as shown in Fig. 2(c). As a result, the parameter $\text{LW}_{\max}$ can be adjusted to limit the maximum number of TEPs.

SGRAND [10] incorporates all soft information into the decoder to generate the ML order of TEPs, and an efficient method for generating the ML order can be found in [23]. The ML order for generating TEPs for $n = 6$ is shown in Fig. 3. The ML order for TEP generation is dependent on $\boldsymbol{y}$ and changes with each new vector received (we refer the reader to [10, Algorithm 2] for further details about ML order TEP generation). Let $\boldsymbol{y_1} = [1.0, 2.1, 3.2, 4.3, 5.4, 6.5]$ be the received vector of channel observation values at time instant 1, and the ML order corresponding to $\boldsymbol{y_1}$ is shown in Fig. 3(a). At the second time step, the received vector from the channel is $\boldsymbol{y_2} = [1.3, 2.5, 3.6, 4.9, 5.8, 6.1]$ and the corresponding ML order for TEP generation is depicted in Fig. 3(b). Unlike ORBGRAND, even though the order of the absolute value of LLRs is the same for $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$, the TEPs are generated in a different order for $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$. Similarly, as shown in Fig. 3(c), the ML order changes at a third time instant when $\boldsymbol{y_3} = [1.8, 2.0, 3.9, 4.1, 5.6, 3.3]$ changes.

As a result of the changing TEP query order with each received vector from the channel ($\boldsymbol{y}$) and the TEP interdependence [10], SGRAND does not lend itself to efficient parallel hardware implementation. Alternatively, developing a sequential hardware implementation for SGRAND will result in a high decoding latency, which is unsuitable for applications requiring ultralow latency. ORBGRAND, on the other hand, generates TEPs in the predetermined LW order. Therefore, ORBGRAND is far better suited to parallel hardware implementation than SGRAND.

B. Computational Complexity of GRAND

The computational complexity of GRAND and its variants can be expressed in terms of the number of codebook membership queries required. In GRAND and its variants, a codebook membership query consists of simple operations such as bit-flips and a syndrome check [codebook membership verification (1)]. Furthermore, the complexity can be divided into two categories: worst case complexity, which corresponds to the maximum number of codebook membership queries required; and average complexity, which corresponds to the average number of codebook membership queries required. For a code length of $n = 128$, the worst case number of queries for GRANDAB (AB = 3) decoder is 349 632 queries ($\sum_{i=1}^{\text{AB}} \binom{n}{i}$ [6]). The worst case number of queries for the ORBGRAND decoder depends on the value of the parameter $\text{LW}_{\max}$; for example, with $\text{LW}_{\max} = 96$ and $n = 128$, the worst case complexity is 3.69×10^6 queries [24]. For SGRAND [10], the parameter $\text{Queries}_{\max}$ ($\text{Queries}_{\max} = 5 \times 10^7$; see Fig. 4), which represents the maximum number of queries allowed, determines the worst case complexity.

Fig. 4 compares the FER performance and average complexity for different GRAND variants for decoding CRC Code (128, 104). As seen in Fig. 4(b), as channel conditions improve, the average complexity of GRAND and its variants decreases sharply because transmissions subject to light noise are decoded quickly [6], [9], [10]. In terms of error decoding performance, SGRAND outperforms other GRAND variants by generating TEPs in ML order [10], [23]. As a result, SGRAND achieves ML decoding performance while requiring the fewest average number of codebook membership queries, as shown in Fig. 4. However, as explained previously, the SGRAND is not suited for parallel hardware implementation.

IV. Enhancing the Error Decoding Performance of ORBGRAND

In this section, we will look at techniques for boosting ORBGRAND's decoding performance to match the ML decoding performance of SGRAND. We begin by analyzing the effect of ORBGRAND's parameters on the decoding performance, and then we propose a list-based technique to improve ORBGRAND's decoding performance. The proposed LGRAND algorithm introduces parameters that can be

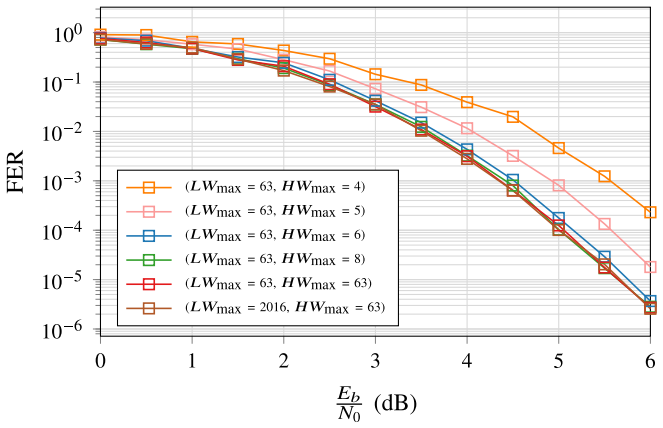

Fig. 5. Comparison of decoding performance of ORBGRAND (LW_{max}, HW_{max}) decoding of BCH Code (63, 45).

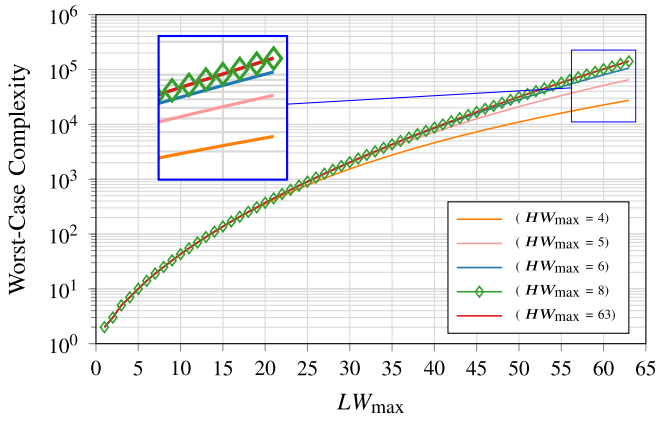

Fig. 6. Maximum number of queries (worst case complexity) comparison for ORBGRAND decoding of BCH Code (63, 45).

tweaked to match the ML decoding performance of SGRAND as well as the target decoding performance and complexity budget of a specific application.

A. Parametric Analysis of ORBGRAND

LW_{max} and HW_{max} are two important ORBGRAND parameters that impact both decoding performance and the maximum number of codebook membership queries required (the worst case complexity) by ORBGRAND. The impact of parameters (LW_{max}, HW_{max}) on the decoding performance and the worst case complexity of ORBGRAND for decoding BCH code (63,45) with BPSK modulation over an AWGN channel is depicted in Figs. 5 and 6, respectively. The performance of ORBGRAND decoding is improved by increasing the values of the parameters LW_{max} and HW_{max}; however, as shown in Fig. 5, the worst case complexity also increases.

B. Proposed LGRAND

Algorithm 2 describes the proposed LGRAND decoding approach. The inputs of LGRAND are identical to those of ORBGRAND, with the exception of an extra parameter δ (threshold for LW). Unlike ORBGRAND, which terminates decoding as soon as any vector ($\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$) fulfills the codebook membership criterion (1), LGRAND generates a list ($\mathcal{L}$) of

Algorithm 2 LGRAND Algorithm

Input: $\boldsymbol{y}$, $\boldsymbol{H}$, $\boldsymbol{G}^{-1}$, LW_{max}, HW_{max}, δ
Output: $\hat{\boldsymbol{u}}$
if $\boldsymbol{H} \cdot \hat{\boldsymbol{y}}^{\top} == \boldsymbol{0}$ **then**
  return $\hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{y}} \cdot \boldsymbol{G}^{-1}$
else
  $\boldsymbol{ind} \leftarrow$ `sortChannelObservationValues(`$\boldsymbol{y}$`)`
  `//` $|\boldsymbol{y}_i| \leq |\boldsymbol{y}_j| \quad \forall i < j$
  $\boldsymbol{e} \leftarrow \boldsymbol{0}$; $\Lambda \leftarrow LW_{max}$; $\Delta \leftarrow HW_{max}$;
  $\mathcal{L} \leftarrow \emptyset$;
  for $i \leftarrow 1$ *to* Λ **do**
    $\boldsymbol{S} \leftarrow$ `generateAllIntPartitions(`i`)`
    `//` $(\lambda_1, \lambda_2, \ldots, \lambda_P) \vdash i$
    forall l *in* $\boldsymbol{S}$ **do**
      $\boldsymbol{e} \leftarrow$ `genErrorPatternMaxHammingWt(`$\boldsymbol{l}$,$\boldsymbol{ind}$, Δ`)` `// HammingWeight(`$\boldsymbol{e}$`)` $\leq \Delta$
      if $\boldsymbol{H} \cdot (\hat{\boldsymbol{y}} \oplus \boldsymbol{e})^{\top} == \boldsymbol{0}$ **then**
        $\hat{\boldsymbol{c}} \leftarrow \hat{\boldsymbol{y}} \oplus \boldsymbol{e}$
        `addToList(`$\mathcal{L}$,$\hat{\boldsymbol{c}}$`)`
        if $\Lambda == LW_{max}$ **then**
          $\Lambda \leftarrow min(i + \delta, LW_{max})$
          $\Delta \leftarrow$ `HammingWeight(`$\boldsymbol{e}$`)`
  $\hat{\boldsymbol{c}}_{final} \leftarrow \arg\max_{\hat{\boldsymbol{c}} \in \mathcal{L}} p(\boldsymbol{y}|\hat{\boldsymbol{c}})$
  $\hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{c}}_{final} \cdot \boldsymbol{G}^{-1}$
  return $\hat{\boldsymbol{u}}$

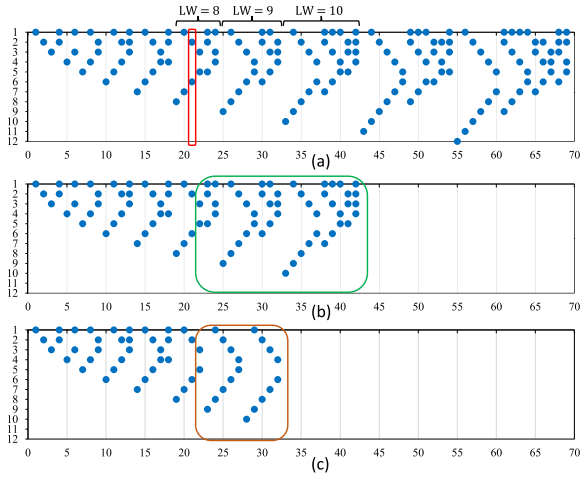

Fig. 7. TEP generation for LGRAND for (n = 12, LW_{max} = 12, HW_{max} = 4). (a) Top: Codebook membership criterion (1) satisfied by 21st TEP ($\boldsymbol{e}$) with LW = 8 and HW = 2 (red rectangle). (b) Middle: Checking additional TEPs for LGRAND. (δ = 2) (green rectangle). (c) Bottom: Restricting HW of additional TEPs to $\leq$2 [δ = 2 and Δ = HammingWeight ($\boldsymbol{e}$)] (brown rectangle).

estimated codewords ($\hat{\boldsymbol{c}}$) and selects the most likely one [$\arg\max_{\hat{\boldsymbol{c}} \in \mathcal{L}} p(\boldsymbol{y}|\hat{\boldsymbol{c}})$] as the final estimated codeword $\hat{\boldsymbol{c}}_{\text{final}}$.

LGRAND proceeds similar to ORBGRAND by sorting $\boldsymbol{y}$ in ascending order of absolute value ($|\boldsymbol{y}_i| \leq |\boldsymbol{y}_j| \; \forall i < j$), and

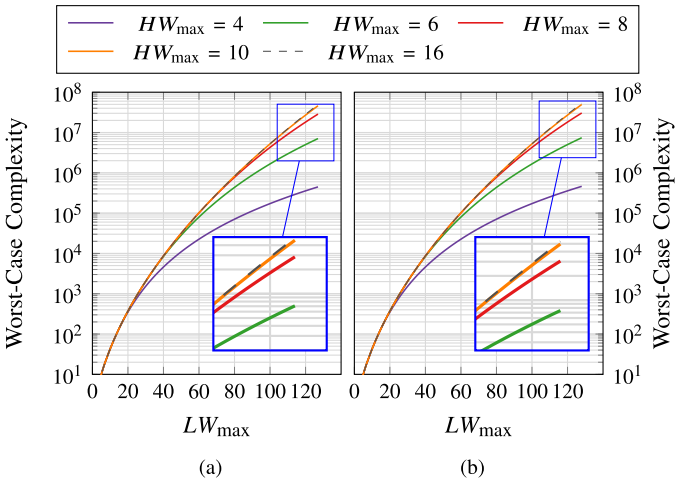

Fig. 8. Worst case complexity for ORBGRAND and LGRAND decoding of linear block codes of length n. (a) $n = 127$. (b) $n = 128$.

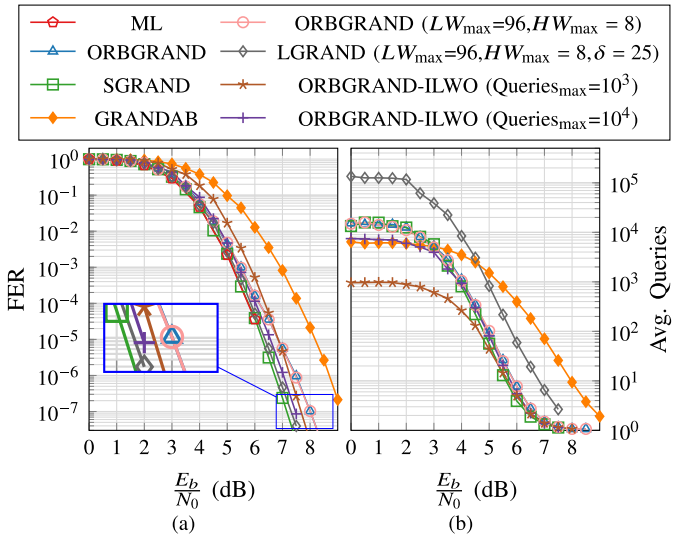

Fig. 9. Comparison of decoding performance and average complexity of different GRAND variants for BCH code (127, 113). (a) FER. (b) Avg. queries.

the corresponding indices are recorded in a permutation vector denoted by $\boldsymbol{ind}$ (line 4). Following this, integer partitions of an LW i ($\forall i \in [0, \Lambda]$, where $\Lambda = \text{LW}_{\text{max}}$) are generated. Then, LGRAND generates TEPs ($\boldsymbol{e}$) by using the generated integer partitions and the TEPs are ordered using the permutation vector $\boldsymbol{ind}$ (line 10). Note that the Hamming weight of the generated TEPs is restricted to $\leq \Delta$ [Δ is initialized to HW_{max} (line 5)]. These TEPs are then applied to $\hat{\boldsymbol{y}}$ to check for codebook membership criterion (1). Whenever a vector $\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$ meets the codebook membership criterion (1), LGRAND adds the vector $\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$ to the list $\mathcal{L}$ (line 13).

Fig. 7(a) depicts the ORBGRAND TEPs for parameters $n = 12$, $\text{LW}_{\text{max}} = 12$, $\text{HW}_{\text{max}} = 4$, and $\delta = 2$. Suppose that the 21st TEP, which corresponds to an integer partition of 8 (LW = 8) and has a Hamming weight of 2, fulfills the codebook membership criterion (1). Rather than stopping the decoding process, LGRAND checks additional TEPs corresponding to LW = 9 and $LW = 10$ ($\delta = 2$), as illustrated in Fig. 7(b). If any of these additional TEPs meet codebook membership constraint (1), they are added to the List $\mathcal{L}$, and the most likely codeword is chosen as the final codeword $\hat{\boldsymbol{c}}_{\text{final}}$ (line 18).

To reduce the number of generated additional TEPs, the maximum Hamming weight of the additional TEPs is restricted to the Hamming weight of the first TEP [$\Delta =$ HammingWeight ($\boldsymbol{e}$)] that satisfied the codebook membership criterion (1) when combined with $\hat{\boldsymbol{y}}$ (line 16). Limiting the Hamming weight of additional TEPs implies that only TEPs ($\boldsymbol{e}$) with Hamming weights $\leq \Delta$ will be generated, as shown in Fig. 7(c).

C. Parametric Analysis of LGRAND

The LGRAND technique introduces the parameter δ that influences both the decoding performance and the average complexity [average number of codebook membership queries (TEPs)] of the algorithm. It should be noted that the worst case complexity of LGRAND is the same as that of ORBGRAND, because the worst case complexity of ORBGRAND and LGRAND is dependent on the parameters LW_{max} and HW_{max}, which are same for both algorithms (see Section IV-A). The worst case complexity for ORBGRAND and LGRAND decoding of linear block codes with lengths $n = 127$ and $n = 128$ is shown in Fig. 8.

Fig. 9 compares the decoding performance and average complexity of different GRAND variants for decoding BCH code (127, 113). Furthermore, the ML decoding performance results [15] are included for reference. The parameter AB = 2 is chosen for the GRANDAB hard-input decoder. As shown in Fig. 9(a), SGRAND ($\text{Queries}_{\text{max}} = 10^6$) outperforms ORBGRAND in decoding performance by 0.8 dB at the target FER of 10^{-7}. However, with the appropriate choice of parameter δ, the proposed LGRAND technique can bridge the decoding performance gap between SGRAND and ORBGRAND as shown in Fig. 9(a).

The effect of changing the value of δ on both the decoding performance and the average computational complexity for LGRAND decoding of BCH code $(127, 113)$ at $E_b/N_0 =$ 6.5 dB is depicted in Fig. 10. As shown in Fig. 10(a), increasing the value of parameter δ improves decoding performance [FER at $E_b/N_0 = 6.5$ dB], but it also increases average computational complexity, as shown in Fig. 10(b). As a consequence, the appropriate value of parameter δ can be chosen to strike a balance between decoding performance and average complexity. At a target FER of 10^{-7}, LGRAND with parameters ($\text{LW}_{\text{max}} = 96$, $\text{HW}_{\text{max}} = 8$, $\delta = 25$) achieves error decoding performance comparable to SGRAND and outperforms ORBGRAND by 0.75 dB, as shown in Fig 9(a).

1) Analyzing Average List Size ($|\mathcal{L}|_{avg}$): The effect of parameter δ on the average list size ($|\mathcal{L}|_{\text{avg}}$) for LGRAND ($\text{LW}_{\text{max}} = 96$, $\text{HW}_{\text{max}} = 8$, $\delta = 25$) decoding for BCH code $(127, 113)$ at different E_b/N_0 values is depicted in Fig. 11. Note that at least 100 errors are collected for each value of parameter δ, and the average list size is plotted in Fig. 11 for various E_b/N_0 values. As seen in Fig. 11, the average list size increases as the value of δ increases for all E_b/N_0 values. However, as shown in Fig. 10, increasing the value of

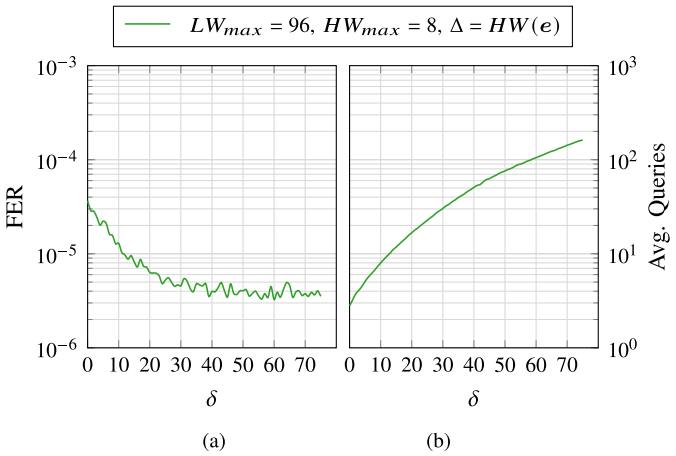

Fig. 10. Parametric analysis of LGRAND (LW_{max}, HW_{max}, δ) for BCH code (127,113) code (at $E_b/N_0 = 6.5$ dB). (a) FER. (b) Avg. queries.

δ improves decoding performance at the expense of average computational complexity.

2) Suboptimality Analysis: The ORBGRAND applies the TEPs in a predetermined LW order as discussed in Section III-A, and it is obvious from the performance difference between the ORBGRAND and SGRAND decoder—which applies the TEPs in an optimal (ML) order—that this LW order is not the optimal schedule. By generating a list of potential candidates during the decoding process and choosing the most likely candidate, the proposed LGRAND improves the decoding performance of ORBGRAND.

Suboptimality count refers to the number of instances where the most likely candidate is not the first on the list ($\mathcal{L}$). Note that if the selected candidate is the first on the list ($\mathcal{L}$), the LGRAND will perform similar to the ORBGRAND. Fig. 12 illustrates the suboptimality count for LGRAND ($\text{LW}_{\text{max}} = 96$, $\text{HW}_{\text{max}} = 8$, $\delta = 25$) decoding of BCH code $(127, 113)$. It should be noted that at least 100 errors were captured at each E_b/N_0 point. As observed in Fig. 12, the suboptimality count increases with higher E_b/N_0 values, revealing the ORBGRAND's suboptimality and providing an explanation for the performance improvement achieved by the proposed LGRAND.

D. Comparison With Enhanced ORBGRAND TEP Scheduling Schemes

Recently, an improved ORBGRAND TEP scheduling technique was presented in [25]; this approach generates TEPs using improved LW order (ILWO) as opposed to the conventional LW order [9]. The TEPs with lower Hamming weights are given precedence in the ILWO, whereas the TEPs with higher Hamming weights are penalized. The parameter $\text{Queries}_{\text{max}}$, which indicates the maximum number of queries allowed, also influences the computational complexity of the ILWO-ORBGRAND as well as the decoding performance [25]. We refer the reader to [25] for more details on the proposed ILWO TEP schedule and performance/complexity tradeoffs. Fig. 9(a) illustrates the ORBGRAND decoder with the ILWO TEP schedule [25], which outperforms the traditional ORBGRAND decoder [9] by 0.3–0.55 dB at the target FER of 10^{-7}. To strike a balance between the decoding performance and computational complexity, appropriate values of $\text{Queries}_{\text{max}}$ can be selected.

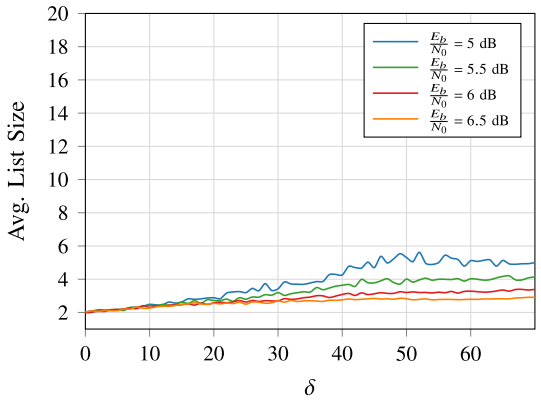

Fig. 11. Average list size ($|\mathcal{L}|_{\text{avg}}$) for LGRAND [$\text{LW}_{\text{max}} = 96$, $\text{HW}_{\text{max}} = 8$, $\Delta = \text{HW}(\boldsymbol{e})$] decoding for BCH code $(127, 113)$.

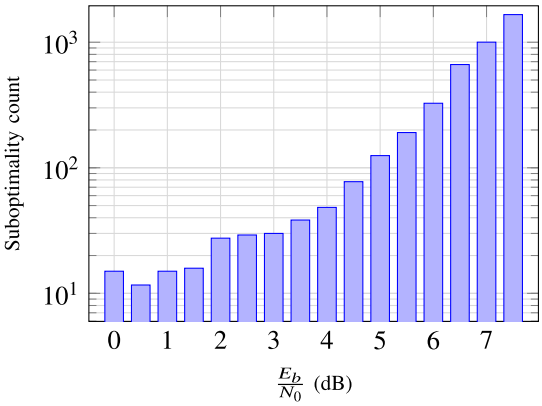

Fig. 12. Suboptimality count for LGRAND ($\text{LW}_{\text{max}} = 96$, $\text{HW}_{\text{max}} = 8$, $\delta = 25$) decoding of BCH code $(127, 113)$.

Note that baseline ORBGRAND and LGRAND TEP schedules allow parallel online computation of TEPs using simple $n \times (n-k)$-bit shift registers and a network of XOR gates, as illustrated in Section VI-B. Thus, the proposed LGRAND presents itself as a viable option for a hardware-friendly solution to achieve ML decoding performance. In a similar way, the ORBGRAND-ILWO [25] can also be implemented in hardware leveraging a network of XOR gates and shift registers with a few minor modifications.

V. Performance Evaluation

In this section, we evaluate the proposed LGRAND in terms of decoding performance and computational complexity for distinct classes of channel codes (BCH, CA-Polar, and CRC). Fig. 13(a) compares the FER performance of LGRAND with different variants of GRAND for decoding BCH code $(127, 106)$. In addition, the ML decoding [15] results are included for reference. Note that the maximum number of queries (worst case complexity) for the LGRAND and ORBGRAND decoders is 4.93×10^7 with codes of length 127, which correspond to the parameters $LW_{\text{max}} = 127$ and $\text{HW}_{\text{max}} = 16$ (see Fig. 8). Furthermore, for the numerical simulation results shown in Fig. 13, the parameter

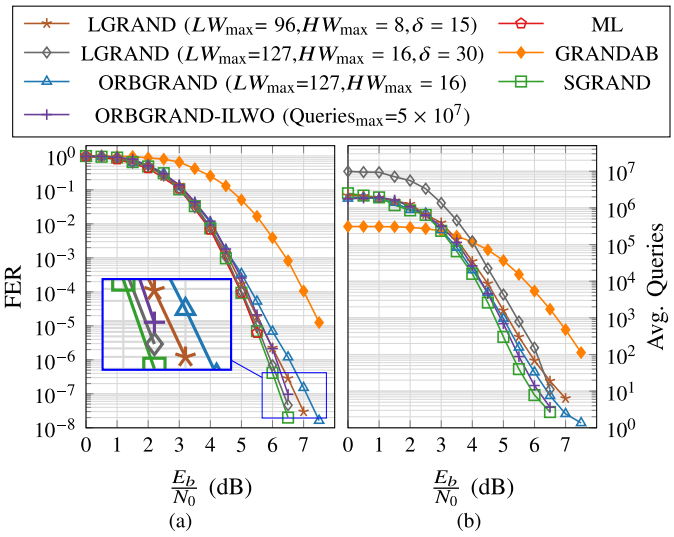

Fig. 13. Comparison of decoding performance and average complexity of different GRAND variants for BCH code (127, 106). (a) FER. (b) Avg. queries.

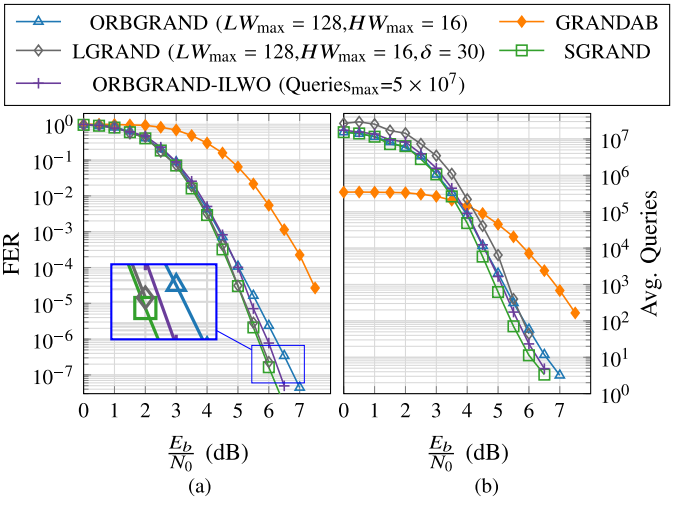

Fig. 14. Comparison of decoding performance and average complexity of different GRAND variants for CRC code (128, 104). (a) FER. (b) Avg. queries.

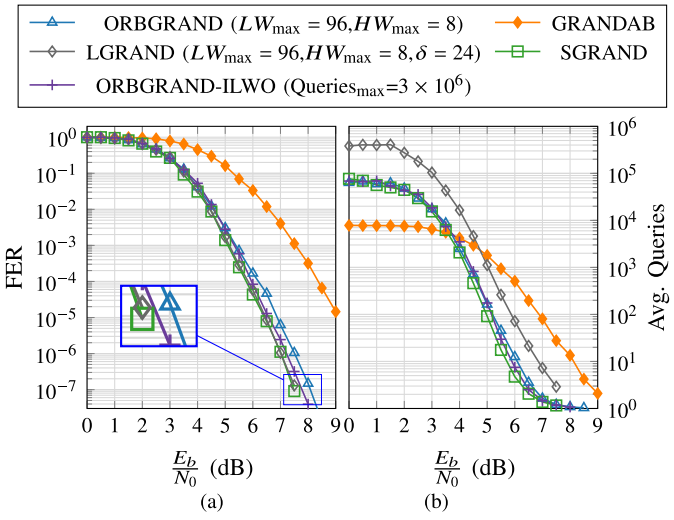

Fig. 15. Comparison of decoding performance and average complexity of different GRAND variants for CRC code (128, 112). (a) FER. (b) Avg. queries.

Queries$_{\max}$ = 5 × 10^7 is used for SGRAND decoder and AB = 3 for GRANDAB decoder.

As demonstrated in Fig. 13(a), while both soft-input variants of GRAND (ORBGRAND and SGRAND) outperform the hard-input GRANDAB, SGRAND achieves the ML performance. The proposed LGRAND (with different parameter settings) outperforms ORBGRAND in decoding performance by 0.25–0.7 dB at a target FER of 10^{-7}. Furthermore, as explained in Section IV-D, these parameters can be tweaked to match SGRAND's ML decoding performance. For the BCH code (127, 106), LGRAND with parameters LW$_{\max}$ = 127, HW$_{\max}$ = 16, and δ = 30 results in a decoding performance gain of 0.7 dB over ORBGRAND at a target FER of 10^{-7} as depicted in Fig. 13(a). Additionally, as demonstrated in Fig. 13(a), the improved TEP schedule ORBGRAND-ILWO (Queries$_{\max}$ = 5 × 10^7) [25] outperforms the baseline ORBGRAND by 0.6 dB at a target FER of 10^{-7}. However, as stated in Section IV, the proposed LGRAND, which is based on the ORBGRAND TEP schedule, is a suitable choice for parallel hardware implementation since it supports online parallel TEP generation using shift registers and a network of XOR gates.

The average computational complexity for different GRAND variants is shown in Fig. 13(b). Despite the fact that SGRAND requires the fewest queries of any GRAND variant, as explained in Section IV, it is not suitable for parallel hardware implementation. As a result, comparing the number of queries required by the proposed LGRAND to the number of queries required by ORBGRAND is reasonable because both are equally suitable for parallel hardware implementation.

Figs. 14 and 15 compare LGRAND decoding performance, as well as average computational complexity, with other GRAND variants for CRC codes [21]. CRC codes are typically used to detect errors in communication systems and to assist list-based channel code decoders in selecting the final candidate codeword. On the other hand, CRC codes can also be used for error correction using the GRAND algorithm. The concept of using CRC codes for error correction with GRAND decoding was presented in [26] and expanded on in [27]. For CRC code (128, 104) and CRC code (128, 112), the generator polynomials are `0xB2B117` and `0x1021`, respectively.

The worst case complexity of both LGRAND and ORBGRAND decoders, corresponding to parameters (LW$_{\max}$ = 128, HW$_{\max}$ = 16) and (LW$_{\max}$ = 96 and HW$_{\max}$ = 8), with codes of length 128 is 5.33 × 10^7 and 3.10 × 10^6 (see Fig. 8), respectively. Furthermore, Queries$_{\max}$ = 5 × 10^7 is employed for SGRAND and AB = 3 is selected for GRANDAB in the numerical simulation results displayed in Fig. 14. Similarly, for the simulation results shown in Fig. 15, Queries$_{\max}$ = 3 × 10^6 and AB = 2 for the SGRAND and GRANDAB decoders, respectively.

At the target FER of 10^{-7}, LGRAND (LW$_{\max}$ = 128, HW$_{\max}$ = 16, δ = 30) achieves similar decoding performance to SGRAND (Queries$_{\max}$ = 5 × 10^7) for the CRC code (128, 104) shown in Fig. 14. Similarly, with the CRC code (128, 112) shown in Fig. 15, LGRAND (LW$_{\max}$ = 96, HW$_{\max}$ = 8, δ = 24) achieves SGRAND (Queries$_{\max}$ = 3 × 10^6) decoding performance (0.5 dB gain over ORBGRAND

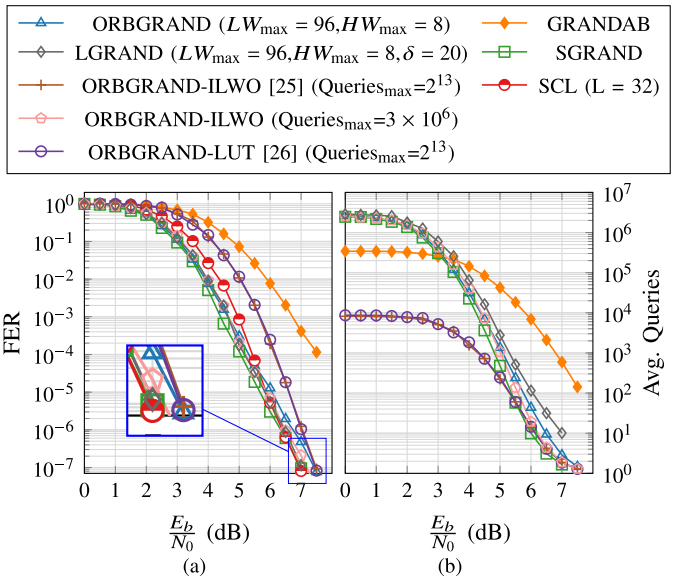

Fig. 16. Comparison of decoding performance and average complexity of different GRAND variants for Polar code (128, 105 + 11). (a) FER. (b) Avg. queries.

at the target FER of 10^{-7}). Furthermore, as illustrated in Figs. 14 and 15, the proposed LGRAND is compared with ORBGRAND-ILWO. At the target FER of 10^{-7}, as shown in Figs. 14 and 15, the proposed LGRAND performs better than the ORBGRAND-ILWO [25] by 0.1–0.2 dB.

Fig. 16 compares the proposed LGRAND's decoding performance, as well as the required average number of queries, with different variants of GRAND for decoding 5G NR CA-polar code (128, 105 + 11). Furthermore, the decoding performance of state-of-the-art soft-input decoder such as the CA-SCL decoder [16], [17] is included for reference. Note that for LGRAND decoding of CA-polar code (128, 105 + 11), the CRC bits are not used to select the most likely candidate from the list ($\mathcal{L}$). Instead, we select the most likely candidate $[\arg\max_{\hat{c}\in\mathcal{L}} p(\boldsymbol{y}|\hat{\boldsymbol{c}})]$ from the list using the ML criterion (see Section IV-B).

The worst case complexity of the ORBGRAND and LGRAND decoder, which corresponds to parameters $\text{LW}_{\text{max}} = 96$ and $\text{HW}_{\text{max}} = 8$, is 3.10×10^6 (see Fig. 8) for the numerical simulation results depicted in Fig. 16. Furthermore, SGRAND employs $\text{Queries}_{\text{max}} = 3 \times 10^6$ and the GRANDAB decoder employs AB = 3.

The LGRAND decoder is also compared with enhanced ORGBGRAND TEP schedules, ORBGRAND-ILWO [25] and lookup table (LUT) assisted fixed latency ORBGRAND (F.L. ORBGRAND) decoder [28], using the same (128, 105) polar code shown in Fig. 16. At a target FER of 10^{-7}, LGRAND ($\text{LW}_{\text{max}} = 96$, $\text{HW}_{\text{max}} = 8$, $\delta = 20$) achieves a decoding performance similar to SGRAND and outperforms traditional ORBGRAND [9] as well as enhanced TEP schedule ORBGRAND ($\text{Queries}_{\text{max}} = 2^{13}$) [25], [28] by ~ 0.5 dB as shown in Fig. 16.

To conclude, LGRAND's parameters (LW_{max}, HW_{max}, δ) can be appropriately chosen for channel codes of different classes (BCH, CA-Polar, and CRC) to achieve ML decoding performance. Furthermore, the complexity overhead for different LGRAND parameter choices can be explored further in order to strike a balance between decoding performance requirements and the complexity/latency budget for a target application.

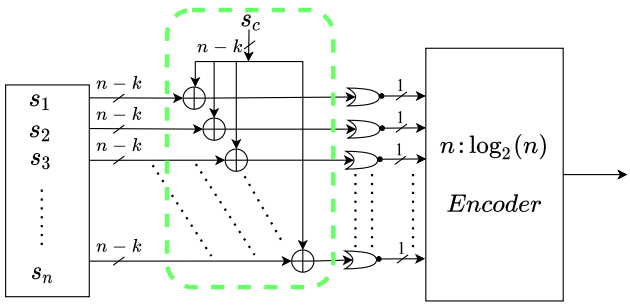

Fig. 17. VLSI architecture for checking error patterns with Hamming weight of 1 ($\boldsymbol{s}_i = \boldsymbol{H} \cdot \mathbb{1}_i^\top$, $i \in [\![1 \,..\, n]\!]$).

VI. VLSI Architecture for LGRAND

This section describes the proposed VLSI architecture for LGRAND. For (n, k) linear block codes, VLSI architectures for GRANDAB and ORBGRAND were proposed in [26] and [18]. Without going into details, we will briefly explain the techniques used in [18] and [26] to generate TEPs with Hamming weights $\geq$1. By using the parity-check matrix ($\boldsymbol{H}$) and the received vector $\hat{\boldsymbol{y}}$ from the channel, the TEP with Hamming weight of 1 ($\boldsymbol{e} = \mathbb{1}_i$, $i \in [\![1 \,..\, n]\!]$) can be checked for codebook membership as

$$\boldsymbol{H} \cdot (\hat{\boldsymbol{y}} \oplus \mathbb{1}_i)^\top = \boldsymbol{H} \cdot \hat{\boldsymbol{y}}^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_i^\top \tag{2}$$

where $\boldsymbol{H} \cdot \hat{\boldsymbol{y}}^\top$ (denoted as $\boldsymbol{s}_c$) is the $(n-k)$-bits syndrome associated with the received vector $\hat{\boldsymbol{y}}$, and $\boldsymbol{H} \cdot \mathbb{1}_i^\top$ (denoted as $\boldsymbol{s}_i$) is the $(n-k)$-bits syndrome associated with the error pattern with Hamming weight of 1 ($\mathbb{1}_i$).

Shift registers are used in [18] and [26] to store syndrome of error patterns with a Hamming weight of 1 ($\boldsymbol{s}_i$) as shown in Fig. 17. To test these error patterns, all of the rows of the shift register ($\boldsymbol{s}_i$) are combined with the syndrome of the received vector ($\boldsymbol{s}_c$) using a network of XOR gates. Following that, each of the n syndromes obtained ($\boldsymbol{s}_i \oplus \boldsymbol{s}_c$) is NOR reduced and fed to a priority encoder, which chooses the TEP that meets the codebook membership criteria (2). Each NOR-reduce output is 1 if and only if all of the bits of the syndromes computed by (1) are 0.

Furthermore, the proposed GRANDAB [26] and ORBGRAND [18] decoders use the linearity property of the underlying code to combine l syndromes of error patterns with a Hamming weight of 1 ($\boldsymbol{s}_i$) to generate syndromes corresponding to an error pattern with a Hamming weight of l ($\boldsymbol{s}_{1,2,\ldots,l} = \boldsymbol{H} \cdot \mathbb{1}_1^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_2^\top \cdots \oplus \boldsymbol{H} \cdot \mathbb{1}_l^\top$). To understand the details of the VLSI implementation that is used to check error patterns with Hamming weight $\geq$ 1, we refer the reader to [18] and [26].

A. VLSI Architecture for Baseline ORBGRAND

Fig. 18 depicts the top-level ORBGRAND VLSI architecture [18], which can decode any linear block code with a

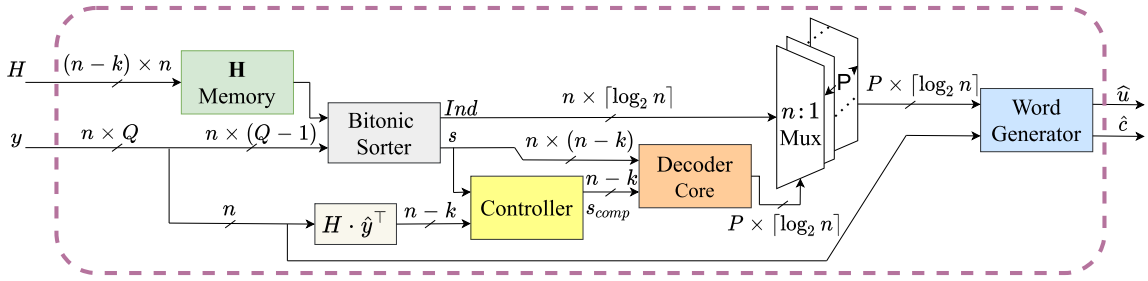

Fig. 18. VLSI architecture for ORBGRAND [18].

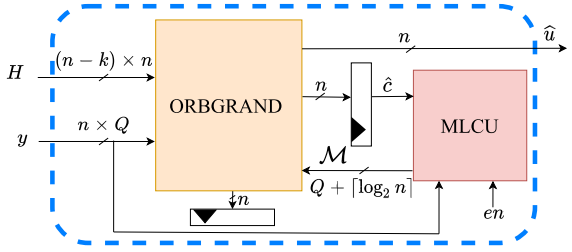

Fig. 19. Proposed LGRAND VLSI architecture.

length of n and code rate $R \geq 0.75$. The proposed architecture takes a vector of channel observation values ($\boldsymbol{y}$) as input and returns the estimated word $\hat{\boldsymbol{u}}$ as output. Any matrix can be loaded into $(n-k) \times n$-bit *H memory* at any time to support various codes and rates. The hard-demodulated vector $\hat{\boldsymbol{y}}$ is subjected to a syndrome check (1) in the first phase of decoding. If the syndrome ($\boldsymbol{s}_c$) is verified ($\boldsymbol{s}_c = \boldsymbol{0}$), decoding is presumed to be successful. Otherwise, the *decoding core* generates the TEPs ($\boldsymbol{e}$) in the LW order and applies them to $\hat{\boldsymbol{y}}$, after which the resulting vector $\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$ is checked for codebook membership (1). If any of the tested syndrome combinations satisfy the parity-check constraint (1), the 2-D priority encoder is used in conjunction with the *controller* module to forward the respective indices to the word generator module, where P multiplexers are used to convert the sorted index values to their appropriate bit-flip locations.

B. Proposed LGRAND VLSI Architecture

Fig. 19 depicts the proposed VLSI architecture for LGRAND, which builds up on the VLSI architecture for ORBGRAND [18] and adds module *ML computation unit* (MLCU). As described in Section IV-B, the LGRAND selects the most likely codeword from a list ($\mathcal{L}$) of candidates. In the proposed LGRAND VLSI architecture, the ORBGRAND decoder works with the *MLCU* to select the most likely codeword.

The proposed *MLCU*'s microarchitecture is depicted in Fig. 20. The *MLCU* takes two inputs, $\hat{\boldsymbol{c}}$ and $n \times Q$-bit $\boldsymbol{y}$, where Q is the quantization width, and outputs the $Q + \lceil \log_2 n \rceil$-bit value $\mathcal{M}$. An adder tree with $\log_2 n$ stages is used to add the elements ($\boldsymbol{y}_i, \forall i \in [1, n]$) of the $\boldsymbol{y}$ vector to compute the likelihood $\mathcal{M} \leftarrow \sum_{i=1}^{n} [(-1)^{\hat{c}_i} \boldsymbol{y}_i]$ [29]. Furthermore, the components *SMto2C* and *2CtoSM* are used to convert from sign-magnitude to 2's complement form to facilitate signed addition and from 2's complement to sign-magnitude representation to facilitate comparison.

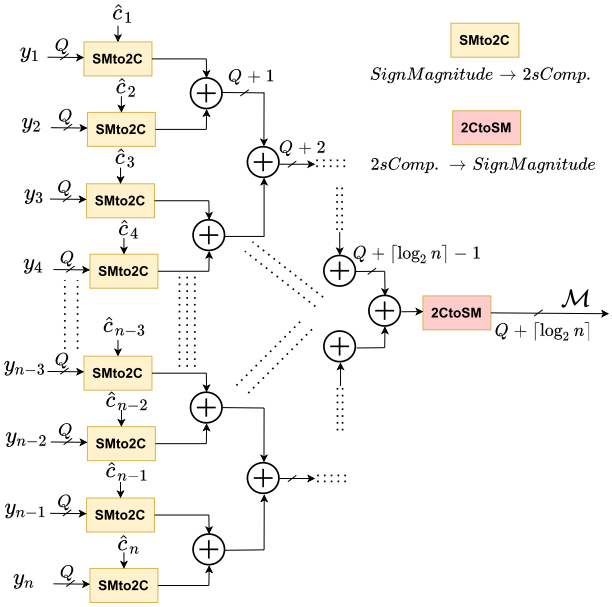

Fig. 20. Microarchitecture for MLCU ($\mathcal{M} \leftarrow \sum_{i=1}^{n} [(-1)^{\hat{c}_i} \boldsymbol{y}_i]$).

In the proposed LGRAND VLSI architecture shown in Fig. 19, the ORBGRAND decoder delivers the estimated codeword ($\hat{\boldsymbol{c}}$) to the *MLCU*, which computes the likelihood value $\mathcal{M}$ for the estimated codeword ($\hat{\boldsymbol{c}}$), then the *MLCU* returns $\mathcal{M}$ to the ORBGRAND decoder. Note that the list of estimated candidate codewords ($\mathcal{L}$) is not stored in a separate memory in the proposed LGRAND hardware; instead, as soon as an estimated codeword ($\hat{\boldsymbol{c}}$) is available at the ORBGRAND decoder, it is passed to the *MLCU*.

The decoding core compares the likelihood values of the currently estimated codeword ($\mathcal{M}_{\text{curr}}$) and the previously estimated codeword ($\mathcal{M}_{\text{prev}}$). If and only if the $\mathcal{M}_{\text{curr}}$ value is higher than the $\mathcal{M}_{\text{prev}}$ value, the previous estimated codeword is replaced with the new one. If not, the decoding core retains the previous estimated codeword. The decoding core always maintains the estimated codeword with the highest likelihood by repeating this process for each successive estimated codeword. Finally, the original message ($\hat{\boldsymbol{u}}$) is retrieved from the estimated codeword with the highest likelihood ($\hat{\boldsymbol{c}}_{\text{final}} \leftarrow \arg\max_{\hat{\boldsymbol{c}} \in \mathcal{L}} \sum_{i=1}^{n} [(-1)^{\hat{c}_i} \boldsymbol{y}_i]$ [29]) using $\boldsymbol{G}^{-1}$, and the decoding process is completed.

C. Implementation Results

The proposed LGRAND with parameters (LW $\leq$ 96, HW $\leq$ 8, $\delta \leq$ 30) has been implemented in Verilog HDL and

TABLE I

TSMC 65-nm CMOS SYNTHESIS COMPARISON FOR LGRAND (LW $\leq$ 96, HW $\leq$ 8, $\delta \leq 30$) WITH ORBGRAND (LW $\leq$ 96, HW $\leq$ 8) FOR $n = 128/127$ AND $0.75 \leq R \leq 1$

		LGRAND	ORBGRAND[18]
Parameters		(LW$\leq$96, HW$\leq$8, $\delta \leq 30$)	(LW$\leq$96, HW$\leq$8)
Technology (nm)		65	65
Supply (V)		0.9	0.9
Max. Frequency (MHz)		454	454
Area (mm^2)		2.38	2.27
W.C. Latency (μs)		205.76	205.76
Avg. Latency (ns)		2.2[a]	2.2[a]
W.C. T/P (Mbps)	$k = 104$[b]	0.505	0.505
	$k = 105$[c]	0.510	0.510
	$k = 106$[d]	0.515	0.515
	$k = 112$[e]	0.544	0.544
	$k = 113$[f]	0.549	0.549
Avg. T/P (Gbps)	$k = 104$[b]	47.27	47.27
	$k = 105$[c]	47.72	47.72
	$k = 106$[d]	48.18	48.18
	$k = 112$[e]	50.90	50.90
	$k = 113$[f]	51.36	51.36
Power (mW)		146.27	134.43
Energy per Bit (pJ/bit)	$k = 104$[b]	3.09	2.84
	$k = 105$[c]	3.06	2.81
	$k = 106$[d]	3.03	2.79
	$k = 112$[e]	2.87	2.64
	$k = 113$[f]	2.84	2.62
Area Efficiency (Gbps/mm^2)	$k = 104$[b]	19.86	20.82
	$k = 105$[c]	20.05	21.02
	$k = 106$[d]	20.24	21.22
	$k = 112$[e]	21.38	22.42
	$k = 113$[f]	21.58	22.62
Code compatible		Yes	Yes
Rate compatible		Yes	Yes

[a] For $\frac{E_b}{N_0} \geq 8.5$dB (Fig. 21), [b] CRC Code (128,104), [c] Polar code (128,105+11)
[d] BCH Code (127,106), [e] CRC Code (128,112), [f] BCH Code (127,113)
Information Throughput (Gbps) = $\frac{k}{\text{Decoding Latency (ns)}}$
Energy per Bit (pJ/bit) = $\frac{\text{Power (mW)}}{\text{Avg. T/P (Gbps)}}$, Area Efficiency (Gbps/mm^2) = $\frac{\text{Avg. T/P (Gbps)}}{\text{Area (mm}^2\text{)}}$

TABLE II

TSMC 65-nm CMOS SYNTHESIS COMPARISON FOR LGRAND (LW $\leq$ 96, HW $\leq$ 8, $\delta \leq 30$) WITH F.L. ORBGRAND [28] DECODER FOR 5G NR CA-POLAR CODE (128,105 + 11)

	LGRAND	F.L. ORBGRAND[a] [26]
Parameters	(LW$\leq$96, HW$\leq$8, $\delta \leq 30$)	$Q_{\max} = 2^{13}$
Technology (nm)	65	7
Supply (V)	0.9	0.5
Max. Frequency (MHz)	454	701
Area (mm^2)	2.38	3.70
W.C. Latency (ns)	205764.3	58.49
Avg. Latency (ns)	2.2[b]	58.49
W.C. T/P (Mbps)	0.51	73610
Avg. T/P (Gbps)	47.7	73.61
Power (mW)	146.27	170.84
Energy per Bit (pJ/bit)	3.06	2.32
Area Efficiency (Gbps/mm^2)	20.04	19.89
Code compatible	Yes	Yes
Rate compatible	Yes	Yes

[a] For $Q_{\text{LUT}} = 512$, $Q_S = 256$, $T = 34$
[b] For LGRAND with parameters $LW_{\max} = 96$, $HW_{\max} = 8$ and $\delta = 20$

synthesized with Synopsys Design Compiler using general-purpose TSMC 65-nm CMOS technology. Furthermore, the proposed LGRAND VLSI implementation is compared to the ORBGRAND [18] VLSI implementation with parameters (LW $\leq$ 96, HW $\leq$ 8), and the synthesis results for $n = 128/127$ and code rate R ($0.75 \leq R \leq 1$) are shown in Table I. Both designs, as shown in Table I, are validated using test benches generated by the proposed hardware's bit-true C model. The input channel LLRs are quantized on 5 bits, with 1 sign bit and 3 bits for the fractional part. To ensure accuracy in power measurements,

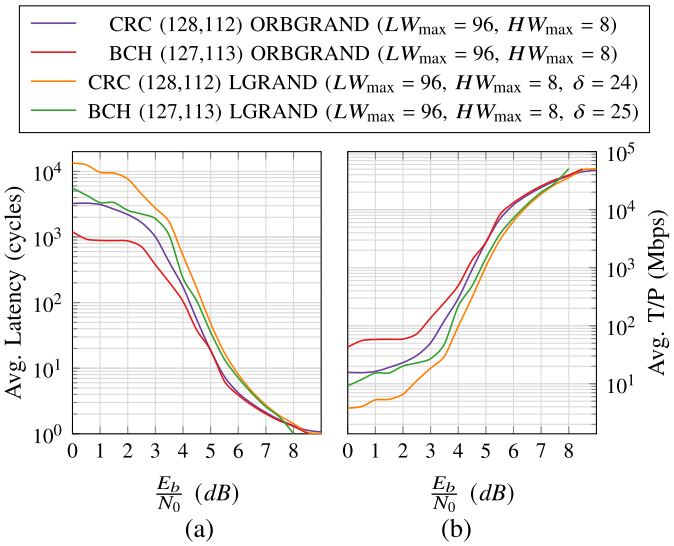

Fig. 21. Comparison of average latency and average information throughput for the ORBGRAND [18] VLSI architecture and the proposed LGRAND VLSI architecture for CRC code (128, 112) and BCH code (127, 113). (a) Avg. latency. (b) Avg. info. throughput.

switching activities from real test vectors are extracted for both hardware architectures shown in Table I.

The proposed LGRAND (LW $\leq$ 96, HW $\leq$ 8, $\delta \leq 30$) has a 4.84% area overhead over ORBGRAND (LW $\leq$ 96, HW $\leq$ 8), resulting in $\sim$4.6% less area efficiency than ORBGRAND. Furthermore, the proposed LGRAND is 7.7%–8.2% less energy efficient than the ORBGRAND (LW $\leq$ 96, HW $\leq$ 8). However, for decoding CRC code (128, 112), the proposed LGRAND with parameters ($\text{LW}_{\max} = 96$, $\text{HW}_{\max} = 8$, $\delta = 24$) outperforms ORBGRAND by 0.5 dB at the target FER of 10^{-7}, as illustrated in Fig. 15(a). Similarly, as shown in Fig 9(a), the proposed LGRAND with parameters $\text{LW}_{\max} = 96$, $\text{HW}_{\max} = 8$, and $\delta = 25$ outperforms ORBGRAND by 0.75 dB at a target FER of 10^{-7} for decoding BCH code (127, 113).

The proposed LGRAND implementation can support a maximum frequency of 454 MHz. One clock cycle corresponds to one time step because we do not consider any pipelining technique for the ORBGRAND decoder core. The proposed LGRAND architecture achieves a worst case information throughput (W.C. T/P) of 0.5–0.549 Mb/s for different classes of channel codes as shown in Table I. The average latency, on the other hand, is significantly smaller than the worst case latency, especially at the higher E_b/N_0 region. The average latency is computed using the bit-true C model, of the proposed hardware, after taking into account at least 100 frames in error for each E_b/N_0 point. As channel conditions improve, the average latency for both ORBGRAND and LGRAND decreases until it reaches only 1 cycle per decoded codeword, as illustrated in Fig. 21(a), resulting in an average latency of 2.2 ns (corresponding to a maximum clock frequency of 454 MHz). The average information throughput, which is the inverse of average latency, is depicted in Fig. 21(b). It should be noted that the average information throughput increases with E_b/N_0, reaching values of 47.27–51.36 Gb/s.

The proposed LGRAND hardware is also compared to a recently proposed state-of-the-art F.L. ORBGRAND decoder [28], and the comparison results are presented in Table II. The F.L. ORBGRAND decoder deploys T decoder pipeline stages and stores the TEPs for the ORBGRAND decoding process in $T-2$ $Q_s \times n$-bit *pattern memories*. When decoding 5G NR CA-Polar code (128, 105 + 11), the F.L. ORBGRAND decoder can provide a maximum information throughput of 73.61 Gb/s with a fixed latency of 58.49 ns owing to the highly pipelined VLSI architecture. The proposed LGRAND hardware, however, can achieve an average information throughput of 47.7 Gb/s for the same polar code (128, 105). At the target FER of 10^{-7}, the proposed LGRAND with parameters ($\text{LW}_{\max} = 96$, $\text{HW}_{\max} = 8$, $\delta = 20$) outperforms both the ORBGRAND and F.L. ORBGRAND decoder [28] by $\sim$0.5 dB and can perform similar to SGRAND, as depicted in Fig. 16. Note that scaling is not used to compare LGRAND and F.L. ORBGRAND [28] due to the vast disparity in the technology nodes employed (65 nm versus 7 nm). The F.L. ORBGRAND [28] and LGRAND decoders are both code and rate compatible and can decode any code.

VII. Conclusion

SGRAND and ORBGRAND are soft-input variants of GRAND, a universal decoder for short-length and high-rate codes. SGRAND delivers ML decoding performance but is not suitable for parallel hardware implementation. ORBGRAND is suitable for parallel hardware implementation, however its decoding performance is inferior to SGRAND. In this article, we introduced LGRAND, a technique for improving the decoding performance of ORBGRAND. The proposed LGRAND includes parameters that can be tweaked to match the decoding performance and complexity budget of a target application. Furthermore, with the appropriate choice of parameters, LGRAND achieves decoding performance comparable to SGRAND. Numerical simulation results show that the proposed LGRAND achieves a 0.5–0.75 dB performance gain over ORBGRAND for channel codes of different classes (BCH, CA-Polar, and CRC) at a target FER of 10^{-7}. LGRAND, like ORBGRAND, lends itself to parallel hardware implementation. According to the VLSI implementation results, the proposed LGRAND has a 4.84% area overhead over the ORGRAND hardware implementation. Furthermore, the proposed LGRAND VLSI architecture can achieve an average information throughput of 47.27–51.36 Gb/s for linear block codes of length 127/128 and different code rates.

References

[1] G. Durisi, T. Koch, and P. Popovski, "Toward massive, ultrareliable, and low-latency wireless communication with short packets," *Proc. IEEE*, vol. 104, no. 9, pp. 1711–1726, Sep. 2016.

[2] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3098–3130, 4th Quart., 2018.

[3] Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor, and B. Vucetic, "High-reliability and low-latency wireless communication for Internet of Things: Challenges, fundamentals, and enabling technologies," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7946–7970, Mar. 2019.

[4] M. Zhan, Z. Pang, D. Dzung, and M. Xiao, "Channel coding for high performance wireless control in critical applications: Survey and analysis," *IEEE Access*, vol. 6, pp. 29648–29664, 2018.

[5] H. Chen et al., "Ultra-reliable low latency cellular networks: Use cases, challenges and approaches," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 119–125, Dec. 2018.

[6] K. R. Duffy, J. Li, and M. Médard, "Capacity-achieving guessing random additive noise decoding," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4023–4040, Jul. 2019.

[7] K. R. Duffy, M. Médard, and W. An, "Guessing random additive noise decoding with symbol reliability information (SRGRAND)," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 3–18, Jan. 2022.

[8] K. R. Duffy, A. Solomon, K. M. Konwar, and M. Medard, "5G NR CA-polar maximum likelihood decoding by GRAND," in *Proc. 54th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2020, pp. 1–5.

[9] K. R. Duffy, "Ordered reliability bits guessing random additive noise decoding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 8268–8272.

[10] A. Solomon, K. R. Duffy, and M. Médard, "Soft maximum likelihood decoding using GRAND," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[11] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1379–1396, Sep. 1995.

[12] J. Van Wonterghem, A. Alloum, J. J. Boutros, and M. Moeneclaey, "On short-length error-correcting codes for 5G-NR," *Ad Hoc Netw.*, vol. 79, pp. 53–62, Oct. 2018.

[13] E. Berlekamp, "Nonbinary BCH decoding (abstr.)," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 2, p. 242, Mar. 1968.

[14] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory*, vol. IT-15, no. 1, pp. 122–127, Jan. 1969.

[15] M. Helmling et al. (2019). *Database of Channel Codes and ML Simulation Results*. [Online]. Available: https://www.uni-kl.de/channel-codes

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

[17] 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.

[18] S. M. Abbas, T. Tonnellier, F. Ercan, M. Jalaleddine, and J. W. Gross, "High-throughput and energy-efficient VLSI architecture for ordered reliability bits GRAND," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 6, pp. 1–13, Jun. 2022.

[19] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, pp. 147–156, Sep. 1959.

[20] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, no. 1, pp. 68–79, Mar. 1960.

[21] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proc. IRE*, vol. 49, no. 1, pp. 228–235, Jan. 1961.

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

[23] A. Valembois and M. Fossorier, "An improved method to compute lists of binary vectors that optimize a given weight function with application to soft-decision decoding," *IEEE Commun. Lett.*, vol. 5, no. 11, pp. 456–458, Nov. 2001.

[24] S. M. Abbas, T. Tonnellier, F. Ercan, M. Jalaleddine, and W. J. Gross, "High-throughput VLSI architecture for soft-decision decoding with ORBGRAND," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 8288–8292.

[25] C. Condo, V. Bioglio, and I. Land, "High-performance low-complexity error pattern generation for ORBGRAND decoding," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2021, pp. 1–6.

[26] S. M. Abbas, T. Tonnellier, F. Ercan, and W. J. Gross, "High-throughput VLSI architecture for GRAND," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2020, pp. 1–6.

[27] W. An, M. Medard, and K. R. Duffy, "CRC codes as error correction codes," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[28] C. Condo, "A fixed latency ORBGRAND decoder architecture with LUT-aided error-pattern scheduling," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 5, pp. 2203–2211, May 2022.

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