Journal of Signal Processing Systems (2022) 94:1047–1065
https://doi.org/10.1007/s11265-022-01775-2

Hardware Architecture for Guessing Random Additive Noise Decoding Markov Order (GRAND-MO)

Syed Mohsin Abbas[1] · **Marwan Jalaleddine**[1] · **Warren J. Gross**[1]

Received: 15 December 2021 / Revised: 13 April 2022 / Accepted: 1 May 2022 / Published online: 20 May 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Communication channels with memory are often sensitive to burst noise, which drastically reduces the decoding performance of standard channel code decoders, and this degradation worsens as channel memory increases. Hence, interleavers and de-interleavers are usually used to reduce the effects of burst noise at the expense of increased latency in the communication system. The delay imposed by interleavers/de-interleavers and the performance deterioration induced by channel memory are unacceptable in novel applications that require ultra-low latency and high decoding performance. Guessing Random Additive Noise Decoding (GRAND) is a universal Maximum Likelihood (ML) decoding technique for short-length and high-rate channel codes. GRAND Markov Order (GRAND-MO) is a hard-input variant of GRAND that has been specifically developed for communication channels with memory that are subject to burst noise. GRAND-MO can be used directly on hard demodulated channel signals, removing the requirement for extra interleavers/de-interleavers and considerably reducing overall latency in communication systems. This paper describes a high-throughput GRAND-MO VLSI architecture that can achieve an average throughput of up to 52 Gbps and 64 Gbps for code lengths of 128 and 79, respectively. Furthermore, we propose improvements to the GRAND-MO algorithm to simplify hardware implementation and reduce decoding complexity. When compared to GRANDAB, a hard-input variant of GRAND, the proposed improved GRAND-MO algorithm yields a decoding performance gain of $2 \sim 3$dB at a target FER of 10^{-5}. Similarly, as compared to the (79, 64) BCH code decoder, the proposed GRAND-MO decoder has a 33% reduced worst-case latency and a 2 dB gain in decoding performance at a target FER of 10^{-5}.

Keywords Guessing Random Additive Noise Decoding (GRAND) · Guessing Random Additive Noise Decoding Markov Order (GRAND-MO) · Maximum Likelihood Decoding (MLD) · Burst Errors · Low Latency · VLSI architecture

1 Introduction

Ultra-reliable low-latency communication (URLLC), a key use-case for 5G and beyond communication networks, enables applications that require high reliability and extremely low latency. Augmented and virtual reality [1, 2], Intelligent Transportation Systems (ITS) [3], the Internet of Things (IoT) [4, 5], and Machine-to-Machine (M2M) communication [6] are a few examples of these applications. These applications necessitate the use of short codes with high rates. Guessing Random Additive Noise Decoding (GRAND) [7] is a recently proposed universal decoding algorithm capable of achieving maximum likelihood (ML) decoding for short-length and high-rate codes. GRAND is code-agnostic and noise-centric, which implies that, unlike other decoding techniques that use the structure of the underlying code to decode, GRAND attempts to guess the noise that corrupted the codeword during transmission through the communication channel. As a result, GRAND can decode any code, whether structured or unstructured. GRAND also achieves capacity when used with random-codebooks [7].

GRAND has hard-input (GRAND with ABandonment (GRANDAB) [7]) and soft-input versions (Ordered Reliability Bits GRAND (ORBGRAND) [8], Symbol Reliability GRAND (SRGRAND) [9], Soft GRAND (SGRAND)

✉ Syed Mohsin Abbas
syed.abbas@mail.mcgill.ca

Marwan Jalaleddine
marwan.jalaleddine@mail.mcgill.ca

Warren J. Gross
warren.gross@mcgill.ca

[1] Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada

[10]) that are distinguished primarily by the order in which the channel-induced noise in the received signal is guessed. Recently, GRAND Markov Order (GRAND-MO) [11] has emerged as a hard-input GRAND variant capable of mitigating the effects of burst noise for communication channels with memory.

Generally, this burst noise degrades the decoding performance of typical channel code decoders. Hence, time-diversity techniques such as interleaving/de-interleaving have typically been used to mitigate the effect of burst noise and make the communication channel appear memory-less. The depth of interleavers/deinterleavers is crucial in dispersing burst noise over several codewords; the deeper the interleavers, the greater the overall noise dispersion and hence the lesser the degradation in decoding performance for a traditional channel code decoder. The depth of interleavers, on the other hand, introduces significant latency on both the transmitter and receiver ends of the communication system. We refer the reader to section IV (C) of [11] for more information on the required depth (hence the latency) of interleavers and its impact on the decoding performance of traditional channel code decoders with varying amounts of channel memory. In emerging applications like URLLC [1–6], where latency and reliability are critical, delays imposed by interleavers/de-interleavers or performance degradation caused by burst-noise are undesirable. GRAND-MO [11] eliminates the need for interleavers/deinterleavers for communication channels with memory, facilitating reliable communication even in the presence of burst noise.

GRAND and its variants generate test error patterns ($\boldsymbol{e}$) that are subsequently applied to the received vector of channel observation values (demodulated symbols) ($\boldsymbol{r}$), and the resulting vector ($\boldsymbol{r} \oplus \boldsymbol{e}$) is then tested for codebook membership using the underlying code's parity check matrix. If the codebook membership criteria is fulfilled, the generated test error pattern ($\boldsymbol{e}$) is declared as guessed noise, and the vector ($\hat{\boldsymbol{c}} \triangleq \boldsymbol{r} \oplus \boldsymbol{e}$) is returned as the estimated codeword.

For communication channels with memory that are prone to burst errors, the GRAND-MO mitigates the effects of noise bursts by exploiting noise correlations and adjusting its test error pattern generation. As a result, GRAND-MO [11] outperforms traditional channel code decoders in such communication channels with burst noise.

The computational complexity of GRAND-MO which refers to the maximum number of codebook membership queries performed, is directly proportional to the number of generated test error patterns. Furthermore, the GRAND-MO computational complexity influences both the decoding latency and the energy consumption of the GRAND-MO hardware. As a result, reducing computational complexity is critical for applications requiring ultra-low latency and low energy consumption.

In this paper, we examine the scheme which is used by GRAND-MO [11] to generate test error patterns ($\boldsymbol{e}$), and we propose a novel method for generating test error patterns which helps to reduce the complexity of GRAND-MO and results in a simpler hardware implementation. The proposed GRAND-MO test error pattern generation scheme introduces parameters that can be tweaked for different classes of channel codes to strike a balance between decoding performance requirements and the complexity/latency budget for a target application.

Furthermore, we propose a hardware architecture for the GRAND-MO algorithm. With a code length of 128 and a target Frame Error Rate (FER) of 10^{-5}, the proposed GRAND-MO hardware achieves an average throughput of up to 52 Gbps and outperforms the GRANDAB [12] decoder in error decoding performance by 3 dB with (128, 104) Random Linear Code (RLC). Compared to the (79, 64) BCH code decoder [13], the proposed VLSI architecture offers 33% lower worst case latency and a 2 dB gain in decoding performance for a target FER of 10^{-5}. GRAND is code-agnostic, therefore the GRAND-MO architecture can decode any error-correcting code as long as the length and rate criteria are met.

It should be noted that a part of this work was previously published in [14] in which the GRAND-MO VLSI architecture and the numerical simulation results for decoding Bose-Chaudhuri-Hocquenghem (BCH) codes with the GRAND-MO algorithm were presented. This paper extends the previous work [14] by presenting additional numerical simulation results for Cyclic Redundancy Check (CRC) codes, RLCs, and BCH codes of different lengths and code rates. Furthermore, a detailed parametric analysis of the proposed test-error pattern generation scheme for GRAND-MO as well as the supporting numerical simulation results are presented. The proposed GRAND-MO hardware is also modified to support additional parameters, and it is evaluated in terms of decoding throughput, hardware efficiency, energy efficiency, and error decoding performance gain. A comprehensive analysis of average latency and average throughput for the GRAND-MO hardware is also presented for different channel conditions.

The remainder of this paper is structured as follows: The GRAND-MO algorithm and the channel model under consideration are described in Sect. 2. Section 3 evaluates the error decoding performance of the GRAND-MO algorithm with various parameters. Section 4 discusses the generation of Test Error Pattern (TEP) for the GRAND-MO algorithm, whereby various modifications to the TEP generation are proposed to simplify the hardware implementation as well as reduce the complexity. Section 5 describes the proposed GRAND-MO hardware architecture in detail. Section 5 also compares the hardware implementation results for the proposed GRAND-MO architecture with GRANDAB [7]

decoder and a recently proposed BCH decoder [13]. Finally, concluding remarks are presented in Sect. 6.

Algorithm 1: GRAND Markov Order

Input: $\boldsymbol{H}$, $\boldsymbol{G}^{-1}$, $\boldsymbol{r}$, b, g, $\lfloor \frac{d}{2} \rfloor$
Output: $\hat{\boldsymbol{u}}$ OR $ABANDON$

1 $\boldsymbol{e} \leftarrow \boldsymbol{0}$
2 $\Delta l \leftarrow \lfloor \frac{\log\left(\frac{b}{g}\right)}{\log\left(\frac{1-g}{1-b}\right)} \rfloor$
3 **while** $\boldsymbol{H} \cdot (\boldsymbol{r} \oplus \boldsymbol{e})^{\top} \neq \boldsymbol{0}$ **do**
4 $[e, m, l] \leftarrow$ `generateNewMarkovErrorPattern`(Δl)
5 **if** $m = \lfloor \frac{d}{2} \rfloor$ *AND* $l = \lfloor \frac{d}{2} \rfloor$ **then**
6 **return** $ABANDON$
7 $\hat{\boldsymbol{u}} \leftarrow (\boldsymbol{r} \oplus \boldsymbol{e}) \cdot \boldsymbol{G}^{-1}$
8 **return** $\hat{\boldsymbol{u}}$

2 Preliminaries

2.1 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 n^{th} location are 0 and the n^{th} location is 1. All the indices start at 1.

2.2 Channel Model

The classic two-state Markov chain [15] is used in this work to model a binary communication channel with memory characterised by burst noise. When the channel is in the good state, G, it is noiseless; when the channel is in the bad state, B it becomes noisy and introduces errors. Furthermore, the channel has two transitional probabilities (g and b) associated with it. The probability of transitioning from state G to state B is b, and the probability of transitioning from state B to state G is g. Both b and g are supposed to be known and, in practice, can be estimated. A burst error is a sequence of consecutive errors introduced by the channel, with a length that follows a geometric distribution with mean $\frac{1}{g}$ and variance $\frac{1-g}{g^2}$. Similarly, the length of an error-free run, when the channel does not introduce any errors, has a mean of $\frac{1}{b}$. The Markov channel's stationary bit-flip probability p is $\frac{b}{b+g} = \mathrm{Q}(\sqrt{2R\frac{E_b}{N_0}})$ where R is the code rate. Please note that for $p = b$, the Markov channel is equivalent to a memoryless binary symmetric channel (BSC).

2.3 GRAND Decoding of Linear Block Codes

GRAND 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 test error patterns ($\boldsymbol{e}$) starting from the most likely up to the least likely pattern taking into account the channel model.

The primary difference between GRAND variants is the order in which these test error patterns are generated. GRAND with ABandonment (GRANDAB) [7] is a hard decision input variant of GRAND that produces test error patterns in ascending Hamming weight order up to weight AB. ORBGRAND [8] and SGRAND [10], which are soft-input variations of GRAND, outperform hard-input GRANDAB in decoding performance and generate test error patterns in logistic weight order and maximum likelihood order, respectively. We refer the reader to [7, 8, 10] for more information on the generation of test error patterns for various GRAND variations.

The generated test error patterns ($\boldsymbol{e}$) in GRAND are combined with the received vector from the communication channel ($\boldsymbol{r}$), and the resulting vector ($\boldsymbol{r} \oplus \boldsymbol{e}$) is evaluated to see if it 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 $\boldsymbol{e}$ is declared as the guessed noise, whereas $\hat{\boldsymbol{c}} \triangleq \boldsymbol{r} \oplus \boldsymbol{e}$ 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:

$$\boldsymbol{H} \cdot (\boldsymbol{r} \oplus \boldsymbol{e})^{\top} = \boldsymbol{0} \tag{1}$$

For the rest of the discussion, we restrict ourselves to (n, k) linear block codes, where n is the code length and k is the code dimension.

2.4 GRAND Markov Order

The pseudo-code for decoding a linear (n, k) block code with GRAND-MO[11] is presented in the algorithm 1. The algorithm's inputs are $\boldsymbol{r}$, b, g and $\lfloor \frac{d}{2} \rfloor$ where $\boldsymbol{r}$ is hard demodulated version of the received vector of size n, and d is the minimum distance of the code. Moreover, the algorithm also uses the $(n-k) \times n$ parity check matrix $\boldsymbol{H}$ and the $n \times k$ matrix $\boldsymbol{G}^{-1}$, with $\boldsymbol{G}$ being the generator matrix of the code ($\boldsymbol{G}^{-1} \cdot \boldsymbol{G} = \boldsymbol{I}$).

The error vector ($\boldsymbol{e}$) is initialized to $\boldsymbol{0}$ (line 1) in GRAND-MO, and Δl is computed using b and g (line 2). The function *generateNewMarkovErrorPattern* generates the test error patterns in the Markov query order, which is described in detail in [11] (Algorithm 1 in [11]). Without going into details, we summarize the test error pattern generating procedure for Markov query order; for additional information, please see [11]. The function

generateNewMarkovErrorPattern takes Δl as input and iteratively generates test error patterns with Hamming weight l and number of bursts m. (line 4). During each iteration, the generated test error patterns ($\boldsymbol{e}$) have either the same number of bursts (m) and a higher Hamming weight (l) than the previously generated error patterns, or they have the same Hamming weight (l) and a higher number of bursts (m) than the previously generated error patterns. The iterative procedure of generating error patterns is repeated until both, the number of bursts (m) and the Hamming weight (l) of the generated test error pattern equal $\lfloor \frac{d}{2} \rfloor$, where d is the code's minimum distance (line 5).

Finally, $\boldsymbol{r}$ is combined with the test error pattern ($\boldsymbol{e}$) that is being considered, and the resulting word $\boldsymbol{r} \oplus \boldsymbol{e}$ is queried for codebook membership using (1). If the resulting codeword belongs to the codebook, $\boldsymbol{e}$ is the guessed noise, $\hat{\boldsymbol{c}} \triangleq \boldsymbol{r} \oplus \boldsymbol{e}$ is the estimated codeword and the message ($\hat{\boldsymbol{u}}$) is recovered (line 7).

3 Decoding Performance Evaluation for GRAND-MO

In this section, we evaluate the decoding performance of GRAND-MO in terms of frame error rate (FER) performance. We use GRAND-MO alongside channel codes of distinct classes such as Bose–Chaudhuri–Hocquenghem (BCH) codes [16, 17], Random Linear Codes (RLCs) [18, 19] and Cyclic Redundancy Check (CRC) codes [20]) in Markov channels.

In Fig. 1, the frame error rate (FER) performance of GRAND-MO is compared to that of traditional Berlekamp-Massey (B-M) decoder [21, 22] with BCH code (127, 106) and BCH code (127, 113) in Markov channels. The demodulator provides hard decision values to the GRAND-MO decoder. The transitional probability (g) of the considered Markov channel is proportional to the memory of the channel. As channel memory increases (as indicated by a decreasing value of g), more burst noise can be observed on the channel. The traditional B-M decoder's decoding performance deteriorates significantly since interleavers/deinterleavers are not present to spread these noise bursts across multiple codewords. On the other hand, the performance of GRAND-MO improves with increasing channel memory due to GRANDMO's unique method of generating putative test error patterns.

Similar trends in decoding performance can be seen with other classes of channel codes, such as CRC codes and RLCs. CRC codes [20] are typically used to detect errors in communication systems and to assist list-based channel code decoders in selecting the final candidate codeword. Additionally, CRC codes can also be used for error correction with GRAND. The concept of using CRC codes for error correction with GRAND decoding was presented in [12] and expanded on in [23]. RLCs [18, 19] are linear block codes that are theoretically high-performing but have not been considered realistic in terms of decodability due to a lack of structure. Figures 2 and 3 plot the FER performance for GRAND-MO and GRANDAB decoding of RLCs and CRC codes of length $n = 128$ and different rates. We can see that as channel memory is increased (decreasing g), the GRAND-MO decoder outperforms the GRANDAB decoder in FER performance.

4 Test Error Pattern (TEP) Generation for GRAND-MO

Despite the absence of interleavers/deinterleavers to deal with burst noise, GRAND-MO outperforms traditional channel code decoders. This improvement in performance is credited to the unique error pattern generation scheme of GRAND-MO, which uses noise correlations to guess the noise bursts in the channel.

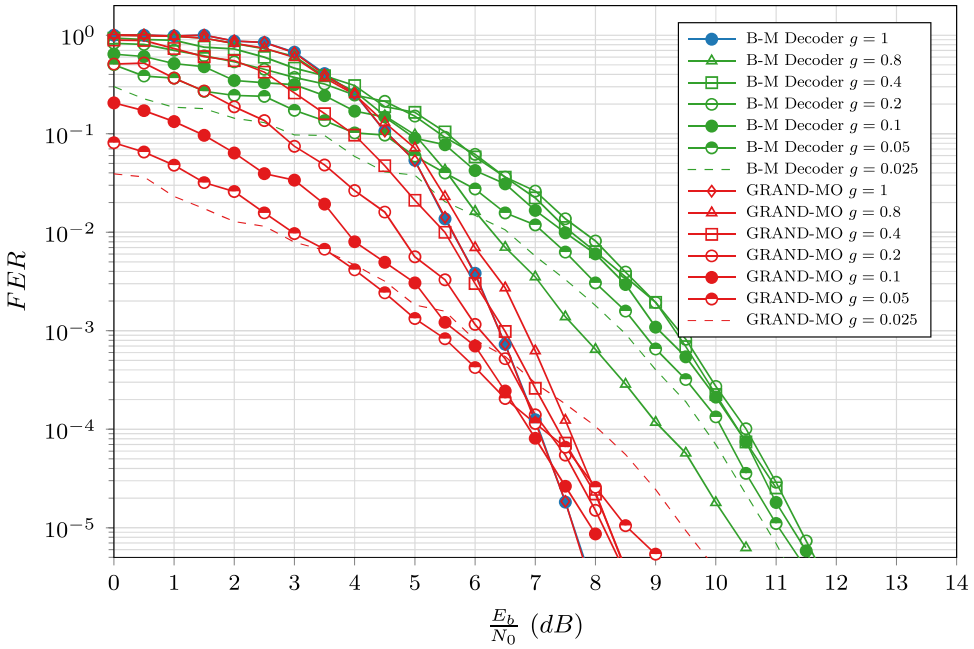
4.1 Markov Query Order for TEP Generation

The GRAND-MO decoder generates test error patterns in Markov query order. Combining a TEP with the the received vector $\boldsymbol{r}$ corresponds to flipping certain bits of vector $\boldsymbol{r}$. The Markov query order for code length $n = 6$ and $\Delta l = 2$ is presented in Fig. 4(a), where each column corresponds to a TEP and a dot corresponds to a bit-flip location in the received vector ($\boldsymbol{r}$). Please note that the GRAND-MO algorithm is proposed and explained in detail in [11] (Algorithm 1 in [11]). For further information on the Markov query order test error pattern generating technique, please see [11].

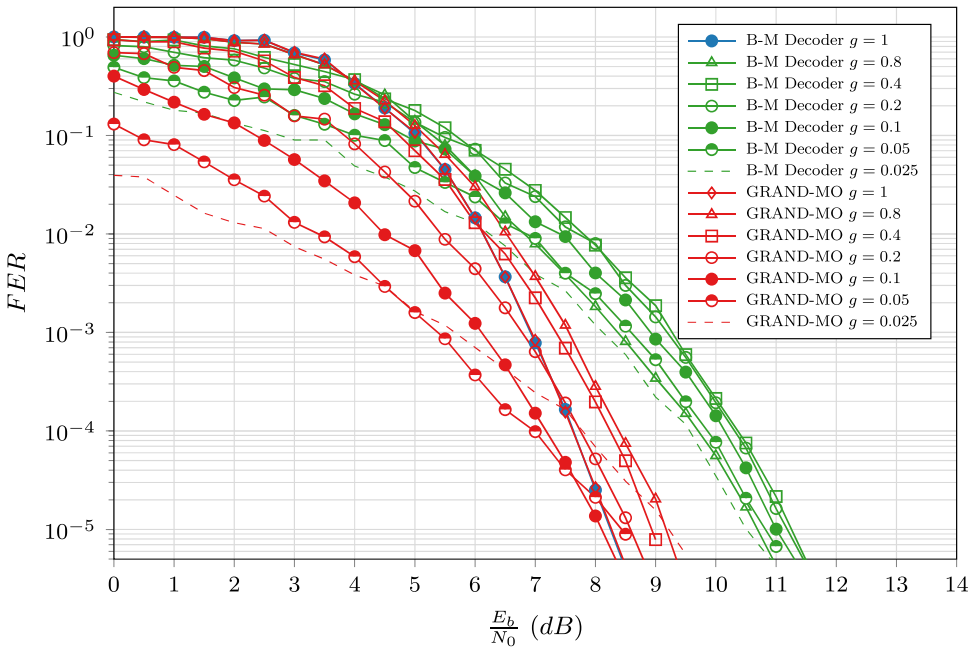
4.2 The Proposed TEP Generation Scheme for GRAND-MO

In this section, we propose simplifying the TEP generation for GRAND-MO. The worst-case computational complexity of GRAND and its variants is proportional to the maximum number of TEPs required. It should be noted that the average number of TEPs required for GRAND and its variants is significantly lower than the maximum number of TEPs required. The maximum number of TEPs (and thus worst-case complexity) for GRAND-MO decoding is determined by Δl, $\frac{1}{g}$, and $\frac{E_b}{N_0}$. Simplifying TEP generation will not only result in simpler hardware implementation, but will also help to reduce worst-case computational complexity.

We briefly explore TEP generation for the GRANDAB [7] decoder before describing our proposed simplified



(a) BCH code $(127, 106)$

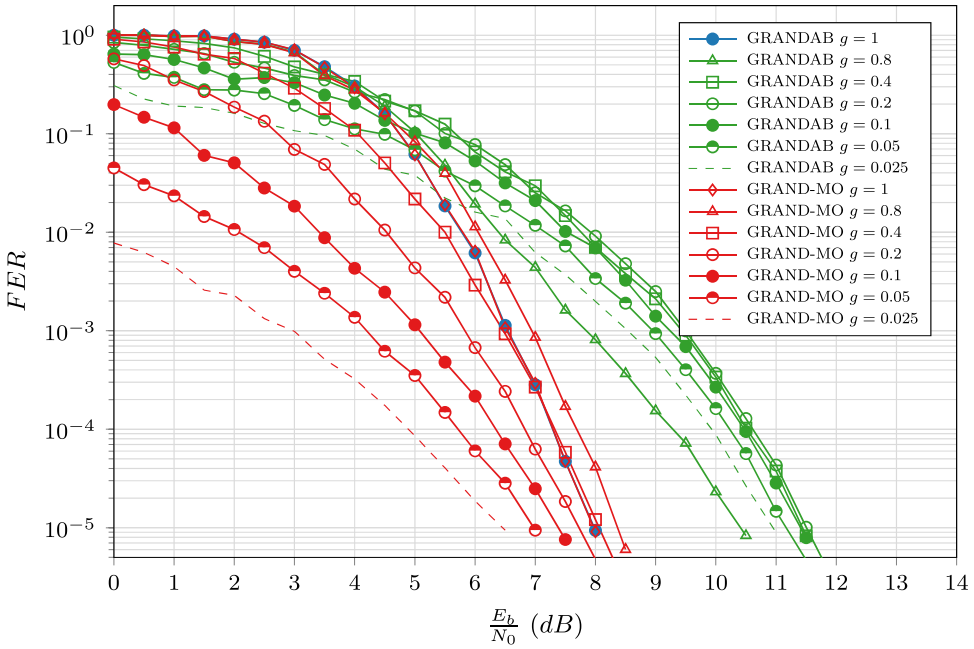


(b) BCH code $(127, 113)$

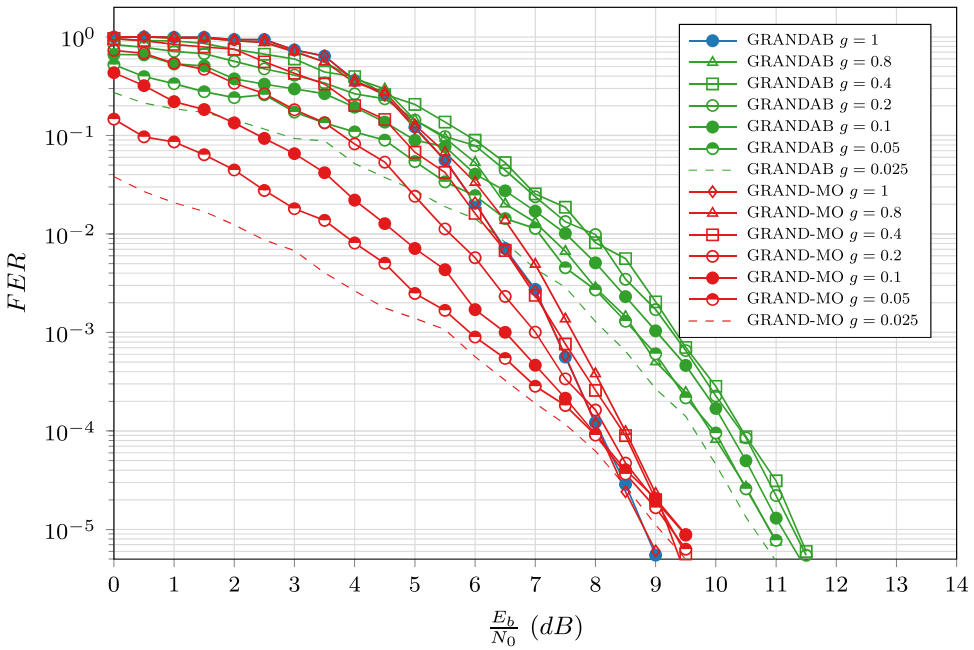
Figure 1 Comparison of the GRAND-MO and BCH Berlekamp-Massey (B-M) decoding performance for BCH code in Markov channels.

TEP generating technique for GRAND-MO. GRANDAB is a hard decision input version of GRAND that generates TEPs by increasing Hamming weight order up to a Hamming weight AB. The TEPs generated in increasing Hamming weight order for $n = 6$ and $AB = 3$ are shown in Fig. 5. TEPs with Hamming weight 1 are generated first, followed by TEPs with Hamming weight 2 and 3, as illustrated in Fig. 5. This simplified Hamming-weight order TEP generation scheme was used in [12] to develop a high-throughput hardware decoder.

Similarly to GRANDAB [7], where TEPs are generated with increasing hamming weight, we proposed to generate TEPs with an increasing number of bursts as well as increasing burst size. Figure 4(b) depicts our proposed TEP



(a) RLC $(128, 104)$

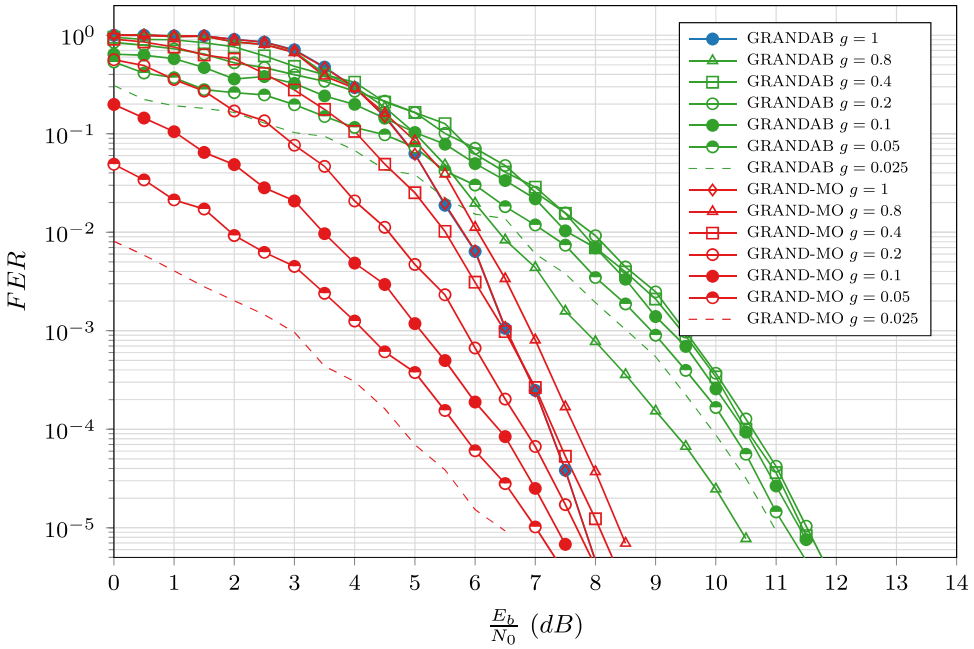


(b) RLC $(128, 112)$

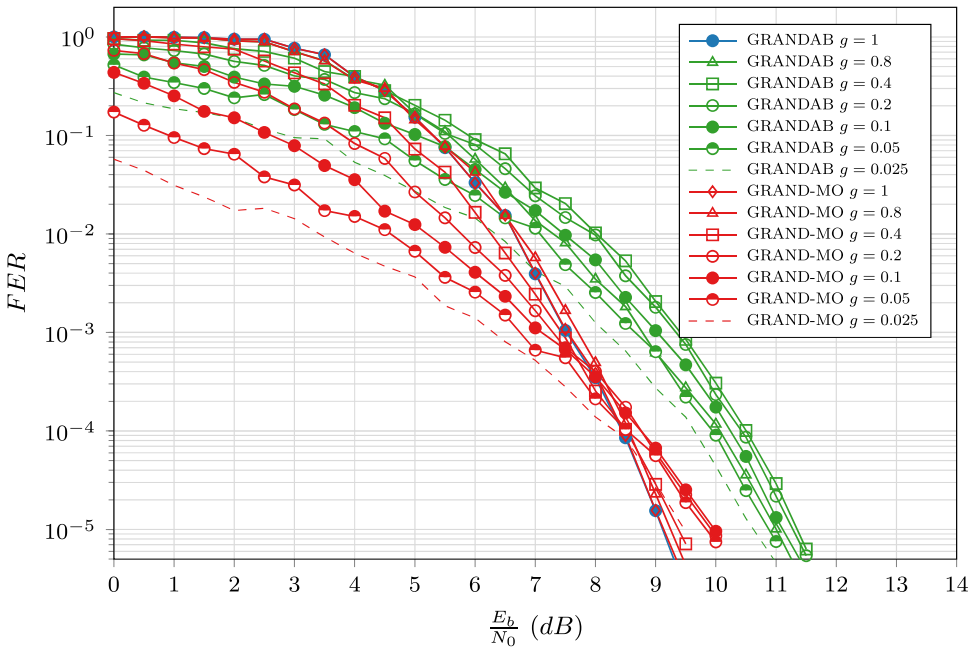
Figure 2 Comparison of the decoding performance of GRAND-MO and GRANDAB with RLC in Markov channels. GRANDAB ($AB = 3$) is used for RLC with $k = 104$, and GRANDAB ($AB = 2$) is used for RLC with $k = 112$.

generation scheme, in which all TEPs corresponding to a single noise burst (TEPs 1 - 21 in Fig. 4(b)) are generated, followed by TEPs with 2 bursts (TEPs 22 - 55 in Fig. 4(b)) and 3 bursts (TEPs 56 - 60 in Fig. 4(b)).

Furthermore, we propose restricting the number of bursts m ($\forall m \in [1, m_{max}]$) and burst sizes l_m in the generated test error patterns to reduce worst-case complexity. Please note that l_m is the maximum burst size when the number of bursts is m. The TEPs of our proposed query order with varying m and l_m are presented in Fig. 4(c) and (d). The TEPs corresponding to a single noise burst ($m = 1$) are displayed in Fig. 4(c) (TEPs 1 - 21 in Fig. 4(c)), where the maximum burst size is restricted to 6 ($l_1 = 6$). Similarly, the TEPs corresponding to two bursts ($m = 2$) are presented in Fig. 4(d), where the burst size is restricted to 4 ($l_1 = 4$) and 3 ($l_2 = 3$), respectively.



(a) CRC $(128, 104)$



(b) CRC $(128, 112)$

Figure 3 Comparison of the decoding performance of GRAND-MO and GRANDAB with CRC codes in Markov channels. GRANDAB ($AB = 3$) is used for CRC with $k = 104$, and GRANDAB ($AB = 2$) is used for CRC with $k = 112$.

Compared to the Markov query order (Fig. 4(a)), the proposed query order with parameters $m = 1$ and $l_1 = 6$ reduces the worst-case complexity from 60 to 21 TEPs (Fig. 4(c)). Similarly, the proposed query order with parameters $m = 2, l_1 = 4$ and $l_2 = 3$ reduces the worst-case complexity to 51 TEPs (Fig. 4(d)).

4.3 Parametric Analysis for the Proposed TEP Generation

In this section, we investigate the effect of varying the parameters m and l_m for the proposed TEP query order for GRAND-MO with different classes of channel codes.

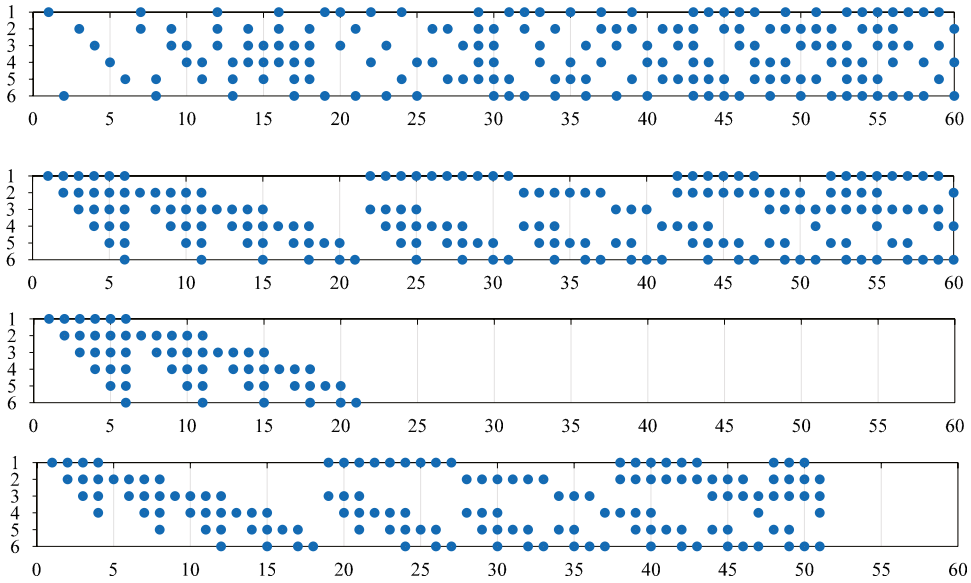


Figure 4 Test error pattern generation for GRAND-MO for $n = 6$ and $\Delta l = 2$ **a** First (Upmost): Markov query order **b** Second: Proposed re-arranged query order **c** Third: Proposed query order with parameters ($m = 1, l_1 = 6$). **d** Fourth (Bottom): Proposed query order with parameters ($m = 2, l_1 = 4$ and $l_2 = 3$).

These parameters have an effect on both, the decoding performance and the complexity of the GRAND-MO decoder. Figure 6 depicts the FER performance for GRAND-MO decoding, with proposed query order (m, l_1, l_2), of BCH code (127, 106). The FER performance of the GRAND-MO decoder using the proposed query order with different parameters (m, l_m) is compared to the performance of the baseline GRAND-MO decoder that use the original Markov query order. The proposed query order with parameters ($m = 2$, $l_1 = 32$, $l_2 = 8$) results in a 0.4 dB degradation in FER at 10^{-5} with BCH code (127, 106); however, the maximum number of codebook membership queries is reduced from 3530504 queries required by Markov order to 487818 queries at $\frac{E_b}{N_0} = 8$ dB. Similarly, the proposed query order with parameters ($m = 2$, $l_1 = 32$, $l_2 = 16$) reduces the maximum number of codebook membership queries to 1788530 queries. To conclude, the parameters of the proposed query order (m, l_m) can be tweaked for a specific application's target decoding performance and complexity budget.

For a specific channel code, the parameters of the proposed query order (m, l_m) can be appropriately chosen by evaluating the decoding performance of GRAND-MO corresponding to a target FER, and selecting the set of parameters that provides the best decoding performance. Figure 7 depicts the decoding performance of GRAND-MO on BCH (127, 106) with various values of g and using the proposed query order ($m = 2$, l_1, l_2) at $\frac{E_b}{N_0} = 8$ dB. The parameters that achieve the best decoding performance are the parameters capable of minimizing the FER at a constant $\frac{E_b}{N_0}$.

As shown in Fig. 7(a), for $g = 0.2$, the GRAND-MO decoder with proposed query order parameters ($m = 2$, $l_1 = 32$, $l_2 = 16$) outperforms all other sets of proposed query order parameters. Similarly, for $g = 0.4$ and $g = 0.8$, the proposed query orders with parameters ($m = 2$, $l_1 = 32$, $l_2 = 10$) and ($m = 2$, $l_1 = 16$ and $l_2 = 4$) outperform other parameters in GRAND-MO decoding performance, as shown in Fig. 7(b) and (c).

The proposed query order's parameters m and l_m can be set to match the FER performance of the baseline GRAND-MO with Markov query order for channel codes of different classes (BCH, RLC, and CRC). This is applicable for codes of different lengths and rates, and for channels of various average burst lengths (g) as shown in Figs. 8 and 9. Furthermore, these parameters (m, l_i $\forall\, i \in [1, m]$) can be tuned for a specific application's target decoding performance and complexity budget.

5 VLSI Architecture for GRAND-MO

This section describes the proposed VLSI architecture for the GRAND-MO decoder.

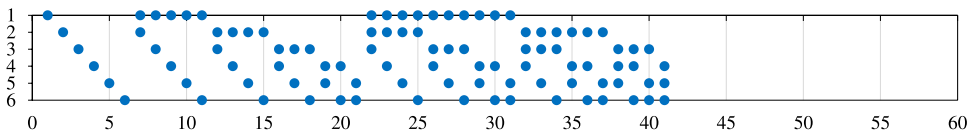
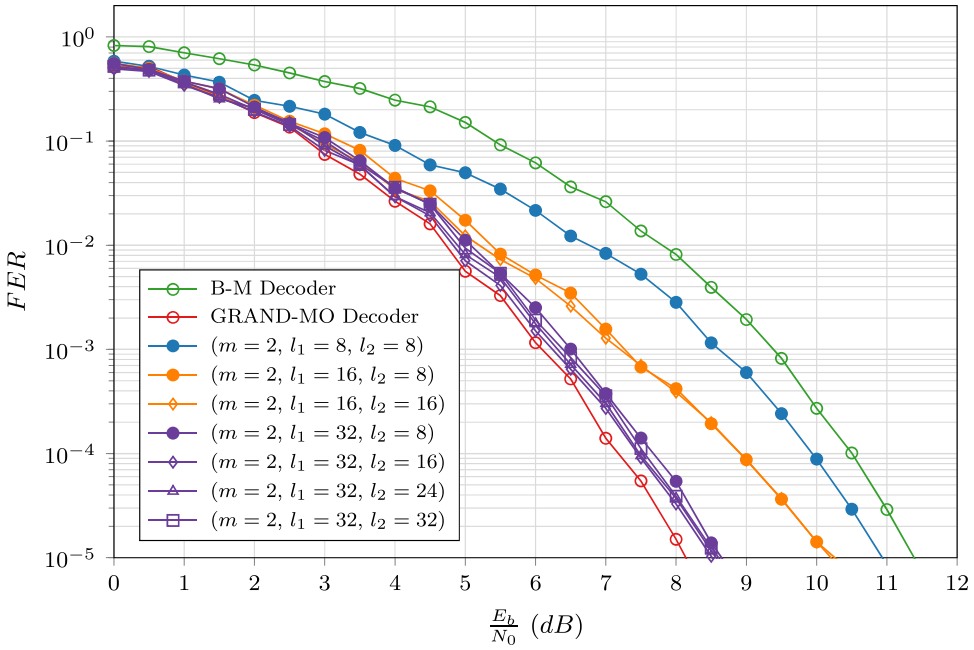
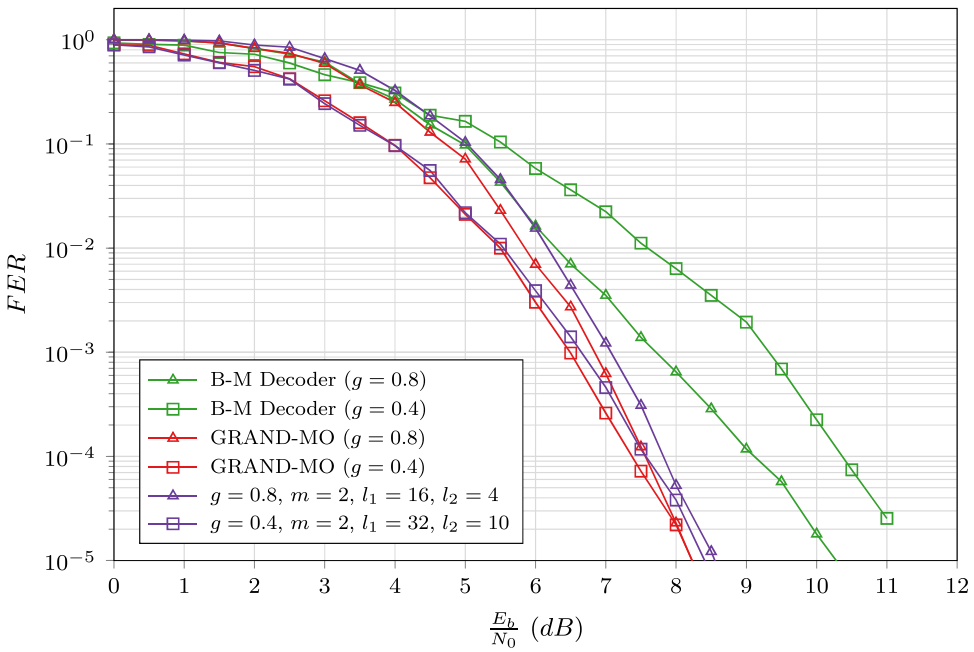


Figure 5 Test error pattern generation for GRANDAB [7].



(a) GRAND-MO $(g = 0.2)$



(b) GRAND-MO $(g = 0.4$ and $g = 0.8)$

Figure 6 Comparison of the GRAND-MO decoding performance using BCH code (127, 106) with Markov query order and proposed query order with parameters (m, l_1, l_2).

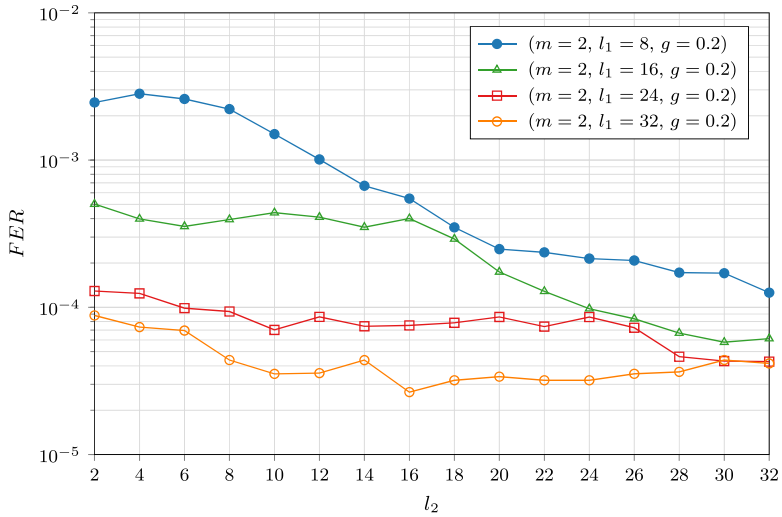
5.1 GRAND VLSI Architecture

In [12], a VLSI architecture for the hard-input GRAND variant, GRANDAB (AB=3) decoder, was proposed for a (n, k) linear block code. Given the parity check matrix $(\boldsymbol{H})$ for a linear block code, the vector $\boldsymbol{r} \oplus \boldsymbol{e}$ is a valid codeword if the codebook membership criterion (1) is satisfied, where $\boldsymbol{e}$ represents the generated test error pattern.

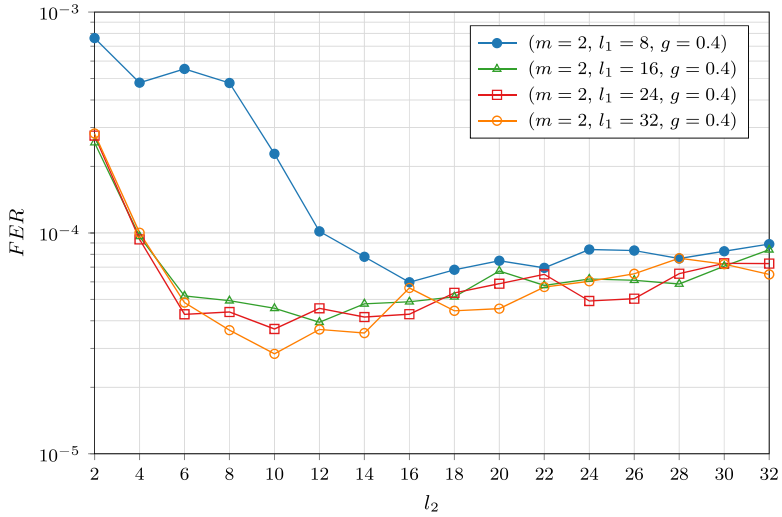
For the error patterns with Hamming weight of 1 ($\boldsymbol{e} = \mathbb{1}_i$, with $i \in [\![1 \,..\, n]\!]$) using the distributive rule, the code membership of the vector $\boldsymbol{r} \oplus \mathbb{1}_i$ can be checked as

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

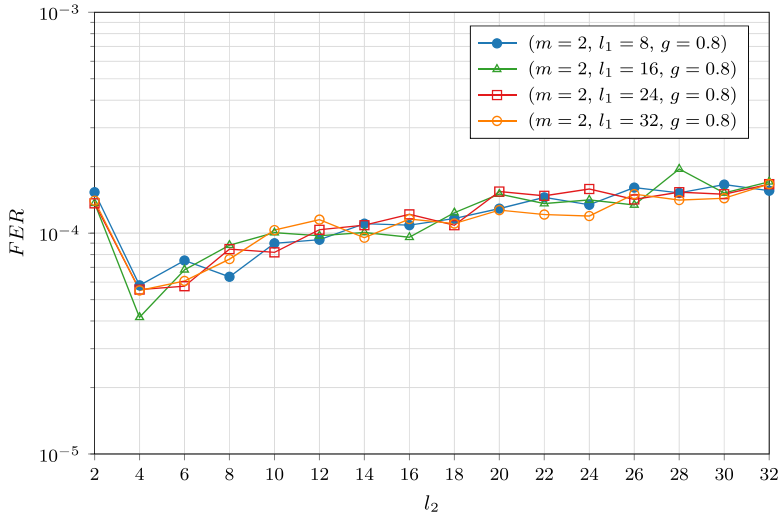
where $\boldsymbol{H} \cdot \boldsymbol{r}^\top$ (denoted as $\boldsymbol{s}_c$) is the $(n-k)$-bits syndrome associated with the received vector $\boldsymbol{r}$ and $\boldsymbol{H} \cdot \mathbb{1}_i^\top$ (denoted



(a) $g = 0.2$



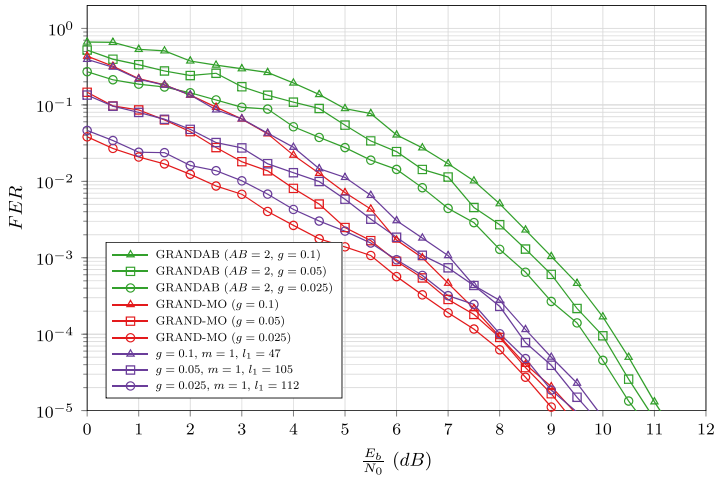
(b) $g = 0.4$



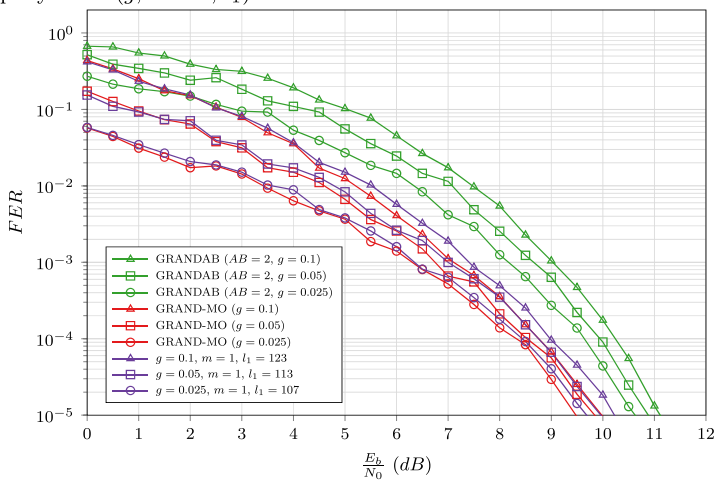
(c) $g = 0.8$

Figure 7 GRAND-MO decoding performance using BCH code (127, 106), at $\frac{E_b}{N_0} = 8$ dB, with proposed query order with parameters (m, l_1, l_2).

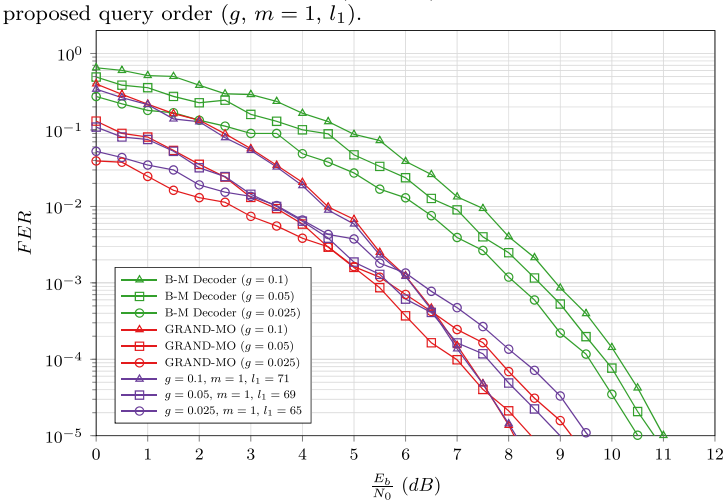
Figure 8 GRAND-MO decoding performance with proposed query order with parameters $(m = 1, l_1)$.



(a) Comparison of the GRANDAB ($AB = 2$) and GRAND-MO decoding performance using RLC (128, 112) with Markov query order and proposed query order (g, $m = 1$, l_1).



(b) Comparison of the GRANDAB ($AB = 2$) and GRAND-MO decoding performance using CRC code (128, 112) with Markov query order and proposed query order (g, $m = 1$, l_1).



(c) Comparison of the B-M Decoder and GRAND-MO decoding performance using BCH code (127, 113) with Markov query order and proposed query order (g, $m = 1$, l_1).

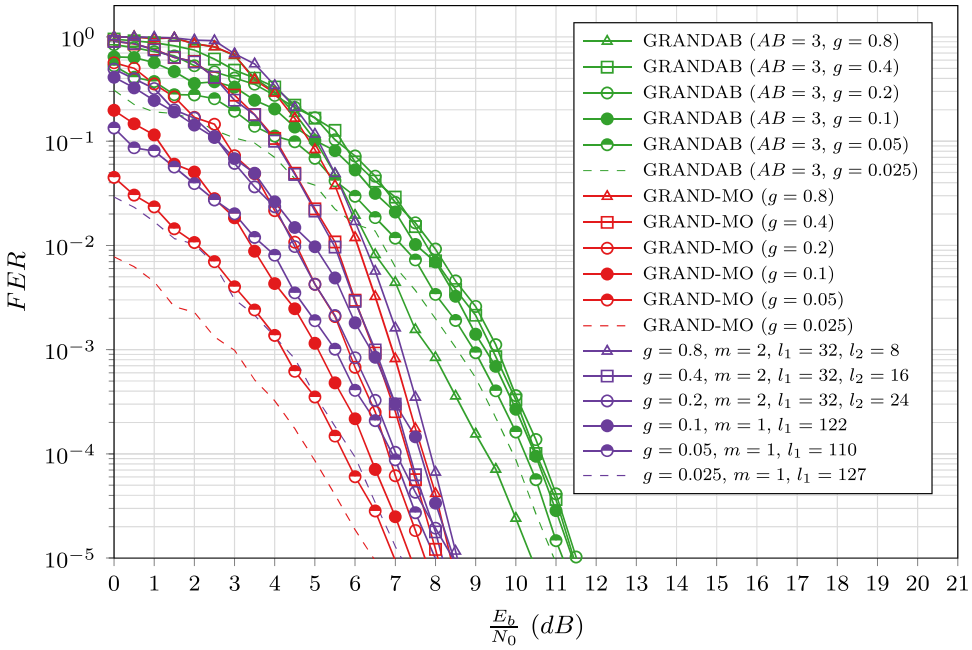


Figure 9 Comparison of the GRANDAB ($AB = 3$) and GRAND-MO decoding performance using RLC code (128, 104) with Markov query order and proposed query order (g, m, $l_i \, \forall \, i \in [1, m]$).

as $\boldsymbol{s}_i$) is the $(n-k)$-bits syndrome associated with the error pattern with Hamming weight of 1 ($\mathbb{1}_i$).

The GRANDAB VLSI architecture [12] uses a $n \times (n-k)$-bit shift register to store all the n syndromes associated with the error patterns with Hamming weight of 1 ($\boldsymbol{s}_i$). To check all the error patterns with Hamming weight of 1, each row of the shift register is combined with the syndrome of the received vector ($\boldsymbol{s}_c$) to compute (2) in one time step. Each of the n test syndromes is NOR-reduced, to feed an n-to-$\log_2 n$ priority encoder as depicted in Fig. 10. The output of each NOR-reduce is 1 if and only if all the bits of the syndrome computed by (2) are 0.

Similarly, the error patterns with Hamming weight of 2 ($\boldsymbol{e} = \mathbb{1}_{i,j}$, with $i \in [\![1 \, .. \, n]\!]$, $j \in [\![1 \, .. \, n]\!]$ and $i \neq j$) can be expressed as $\boldsymbol{H} \cdot (\boldsymbol{r} \oplus \mathbb{1}_{i,j})^\top = \boldsymbol{H} \cdot \boldsymbol{r}^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_i^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_j^\top$. Moreover, the error patterns with Hamming weight of 3 ($\boldsymbol{e} = \mathbb{1}_{i,j,k}, i \neq j \neq k$) can be written as $\boldsymbol{H} \cdot (\boldsymbol{r} \oplus \mathbb{1}_{i,j,k})^\top = \boldsymbol{H} \cdot \boldsymbol{r}^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_i^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_j^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_k^\top$.

In summary, the GRANDAB decoder [12] employs $n \times (n-k)$-bit shift registers to store error pattern syndromes with Hamming weight of 1 ($\boldsymbol{s}_i$). The rows of the shift register ($\boldsymbol{s}_i$) are combined with the syndrome of the received vector ($\boldsymbol{s}_c$) using XOR gates to test these error patterns. Furthermore, the GRANDAB decoder [12] uses the linearity property of the underlying code to combine l 1-bit flip error syndrome to generate an error pattern with the Hamming weight of l ($\boldsymbol{s}_{1,2\ldots,l} = \boldsymbol{H} \cdot \mathbb{1}_1^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_2^\top \ldots \oplus \boldsymbol{H} \cdot \mathbb{1}_l^\top$).

The proposed GRAND-MO architecture is based on the architecture of the GRANDAB decoder. To understand the details of the method used to check error patterns with Hamming weight ≥ 1, we refer the reader to [12]. Because the proposed GRANDAB architecture [12] can only generate test error patterns with Hamming weights $l \leq 3$, many improvements are required to support the generation of error patterns with burst lengths $l \geq 3$.

5.2 Proposed GRAND-MO VLSI Architecture

To permit the generation of error patterns with burst lengths $l \geq 3$, we modify the hardware architecture used for realizing GRANDAB [12].

The contents of the $n \times (n-k)$-bit shift register and the associated peripheral circuitry are shown in Fig. 11. This structure is used to generate the test error patterns corresponding to a noise burst of length l ($1 \leq l \leq n$). Each row of the shift register stores a syndrome associated with a noise burst, with the l^{th} row storing the syndrome associated with a noise burst of length l ($\boldsymbol{s}_{1,2\ldots,l} = \boldsymbol{H} \cdot \mathbb{1}_1^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_2^\top \ldots \oplus \boldsymbol{H} \cdot \mathbb{1}_l^\top$).

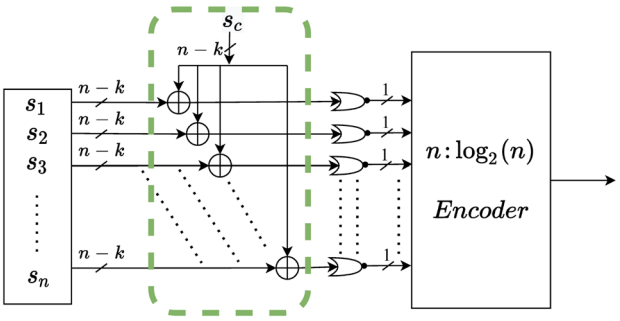


Figure 10 VLSI architecture for checking error patterns with Hamming weight of 1 ($s_i = \boldsymbol{H} \cdot \quad ^\top_i, i \in [\![1 \, .. \, n]\!]$).

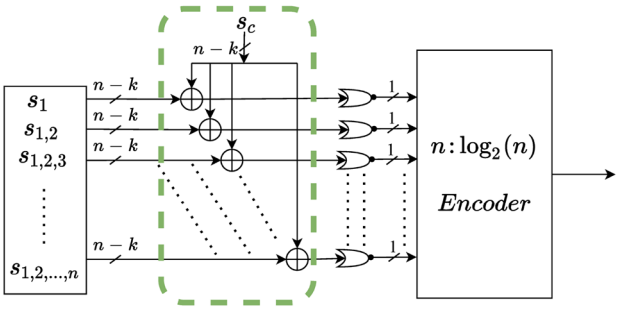


Figure 11 Checking test error patterns corresponding to a noise burst of length $1 \leq l \leq n$, where $s_{1,2,\ldots,l} = \boldsymbol{H} \cdot {}^{\top}_{1} \oplus \boldsymbol{H} \cdot {}^{\top}_{2} \ldots \oplus \boldsymbol{H} \cdot {}^{\top}_{l}$.

Using $(n-k)$-bit-wide XOR gates, each row of the shift register is combined with the syndrome of the received vector $\boldsymbol{s_c}$ ($\boldsymbol{s_c} = \boldsymbol{H} \cdot \boldsymbol{r}^{\top}$), and the syndrome of the test error patterns corresponding to a noise burst of length l is computed. Each of the n test syndromes is NOR-reduced before being fed into a n-to-$\log_2 n$ priority encoder. Each NOR-reduce generates a 1 if and only if all of the bits of the syndrome computed by (1) are 0. In that case, the corresponding TEP is chosen as the guessed error pattern.

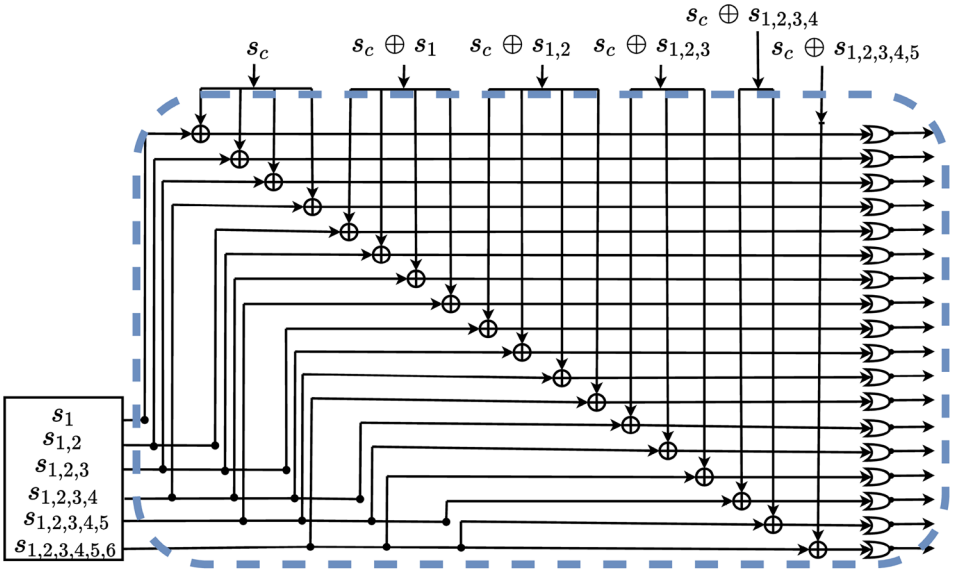
5.3 Scheduling and Details

In this section, we explain the VLSI architecture for the GRAND-MO decoder used to generate all the proposed TEPs shown in Fig. 4(d). This architecture can be expanded to support any linear block code.

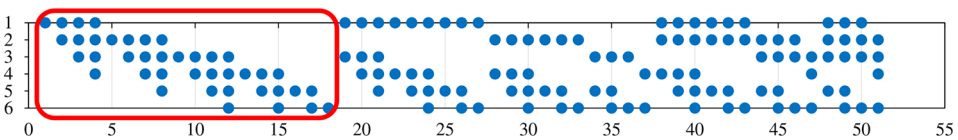
Figure 12(a) shows the contents of the shift register and the arrangement of XOR gates needed to produce the highlighted patterns in Fig. 12(b). This structure is used to generate test error patterns corresponding to the proposed query order with parameters $n = 6$, $m = 2$, $l_1 = 4$ and $l_2 = 3$. The priority encoder and its associated signals are omitted in the figure for clarity. Because of the specific arrangement of the shift register and the XOR gates, all error patterns corresponding to a single noise burst ($m = 1$) of size $l \leq 4$ (test error patterns from 1 to 18 in Fig. 12) are checked in a single time step.

A controller is used in conjunction with the shift register to generate the test error patterns corresponding to $m > 1$. The syndrome that the controller outputs, which is denoted as $\boldsymbol{s}_{comp}$, and the contents of the shift register used to generate test error patterns for $m = 2$ and $|l_1| = 1$ are depicted in Fig. 13. Please note that $|l_i|$ is the size of

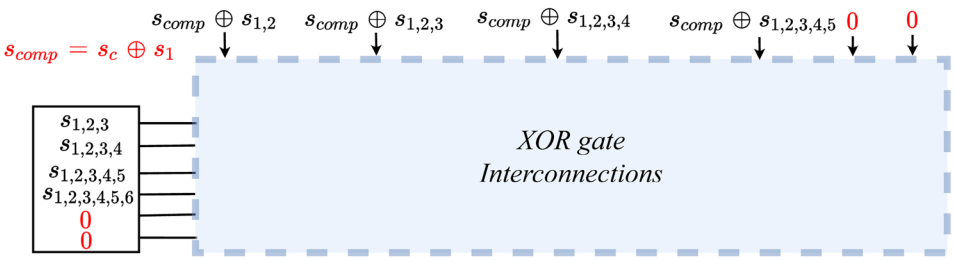
Figure 12 Checking all test error patterns for GRAND-MO decoding using the proposed query order with parameters n =6, $m = 2$, $l_1 = 4$ and $l_2 = 3$.



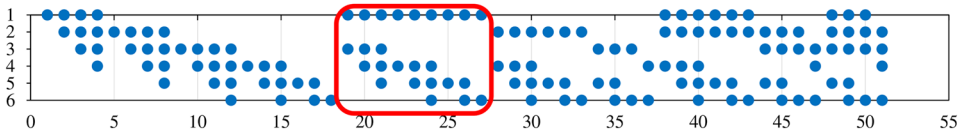
(a) Checking all test error patterns corresponding to a single noise burst of size $l \leq 4$ in one time-step



(b) Error patterns corresponding to a single noise burst of size $l \leq 4$ (red-rectangle).



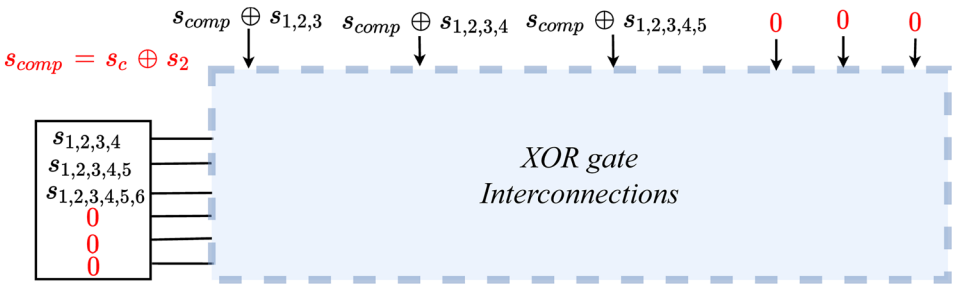
(a) Checking all test error patterns corresponding to $m = 2$ and $|l_1| = 1$ with $s_{comp} = s_c \oplus s_1$.



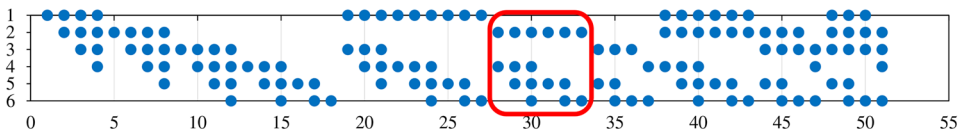
(b) Error patterns corresponding to $m = 2$ and $|l_1| = 1$ with $s_{comp} = s_c \oplus s_1$ and $n = 6$ (red-rectangle).

i^{th} burst ($\forall |l_i| \in [1, l_m]$). The shift register is shifted up by two positions, and the controller outputs $s_{comp} = s_c \oplus s_1$. As a result, all test error patterns with $s_{comp} = s_c \oplus s_1$ are checked in a single time step. Similarly, at the next time step, the controller module outputs $s_{comp} = s_c \oplus s_2$ and the shift register is shifted up by one position, as illustrated in Fig. 14. This specific arrangement enables us to generate all of the test error patterns by using $s_{comp} = s_c \oplus s_2$. Following a similar discussion, we can deduce that with a code length of n, it takes $n - 2$ time steps to generate all test error patterns (test error patterns from 19 to 37 in Fig. 14) corresponding to $m = 2$ and $|l_1| = 1$, where the shift register is shifted up by one position at each time step.

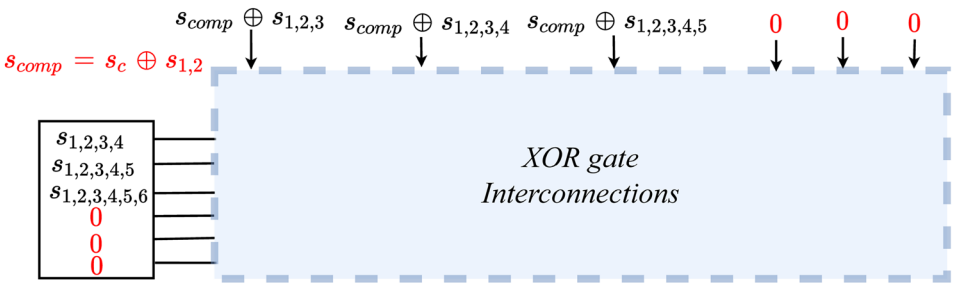
At the following time step, the shift register is reset and shifted up by 3 positions to generate the test error patterns corresponding to parameters $m = 2$ and $|l_1| = 2$, and the controller outputs $s_{comp} = s_c \oplus s_1 \oplus s_2$, as shown in Fig. 15. To generate all test error patterns corresponding to $m = 2$ and $|l_1| = 2$ (test error patterns from 38 to 47 in Fig. 14) , the shift register is shifted up by 1 in each time step, and a total of $n - 3$ time steps are required. Similarly, $n - 4$ time steps are required to generate the test error patterns corresponding to $m = 2$ and $|l_1| = 3$ (test error patterns 49 to 51 in Fig. 14).



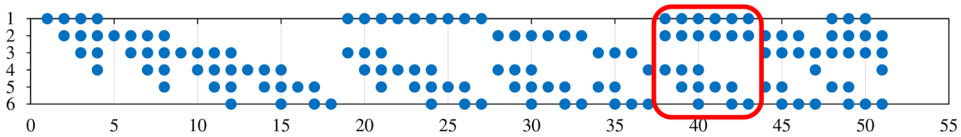
(a) Checking all test error patterns corresponding corresponding to $m = 2$ and $|l_1| = 1$ with $s_{comp} = s_c \oplus s_2$.



(b) Error patterns corresponding to $m = 2$ and $|l_1| = 1$ with $s_{comp} = s_c \oplus s_2$ and $n = 6$ (red-rectangle).



(a) Checking all test error patterns corresponding to $m = 2$ and $|l_1| = 2$ with $s_{comp} = s_c \oplus s_{1,2}$.



(b) Error patterns corresponding to $m = 2$ and $|l_1| = 2$ with $s_{comp} = s_c \oplus s_{1,2}$ and $n = 6$ (red-rectangle).

Figure 15 Checking test error patterns corresponding to proposed query order for GRAND-MO at time step 6.

Based on the preceding discussion, the number of time steps required to generate test error patterns corresponding to parameters $m = 2$ and $L = min(l_1, l_2)$ with the proposed GRAND-MO hardware is $(n-2) + (n-3) + \ldots + (n-(L+1))$, which can be written as $L \times n - (2 + 3 \ldots (L+1))$. Moreover, with some algebraic manipulations, the equation can be further simplified to $L \times n - \frac{(L+1)\times(L+2)}{2} + 1$[1, 2].

It should be noted that the proposed VLSI architecture is highly parallelized and only requires 2 time-steps to check all of the error patterns corresponding to a single noise burst $(n, m = 1, l_1)$. The first time step is used to compute s_c and the second time step is used to query all of the error patterns associated with a single noise burst.

In summary, for the proposed VLSI architecture, the number of required time steps to check all the error patterns corresponding to the proposed query order with parameters $(n, m = 2, L = min(l_1, l_2))$ is given as:

$$L \times (\frac{2 \times n - L - 3}{2}) + 2. \tag{3}$$

The proposed GRAND-MO VLSI architecture is shown in Fig. 16 and can be used to decode any linear block code of length n. The control and clock signals are omitted for clarity. The proposed architecture takes $\boldsymbol{r}$ as an input and outputs the estimated word $\hat{\boldsymbol{u}}$. Any $\boldsymbol{H}$ matrix can be loaded into the $(n-k) \times n$-bit *H memory* at any time to support different codes and rates.

In the first phase of decoding, the input vector $\boldsymbol{r}$ is subjected to a syndrome check (1). If the syndrome is verified, decoding is assumed to be successful. Otherwise, the the *decoding core* generates the test error patterns $\boldsymbol{e}$ using the proposed query order and applies them to the $\boldsymbol{r}$. Finally, the resulting vector $\boldsymbol{r} \oplus \boldsymbol{e}$ is checked for codebook membership.

If any of the generated test error patterns satisfy the codebook membership constraint (1), the *controller* module forwards the corresponding indices to the *word generator* module, which translates these index values to their correct bit flip locations.

5.4 Implementation Results

The proposed GRAND-MO architecture with parameters ($n = 128$, $m = 2$, $l_1 \leq 32$ and $l_2 \leq 32$) and ($n = 128$, $m = 1$, $l_1 \leq 128$), has been implemented in Verilog HDL and synthesized using Synopsys Design Compiler with general-purpose TSMC 65 nm CMOS technology. The design has been verified using test benches generated via the bit-true C model of the proposed hardware.

Table 1 presents the synthesis results for the proposed decoder with $n = 128$ and the proposed architecture can support code rates between 0.75 and 1. To ensure accuracy in power measurements, switching activities from real test vectors are extracted for all of the VLSI architectures presented in Table 1. GRAND-MO implementations with parameters ($n = 128$, $m = 2$, $l_1 \leq 32$ and $l_2 \leq 32$) and ($n = 128$, $m = 1$, $l_1 \leq 128$) can support maximum frequencies of 500 MHz and 400 MHz, respectively. Because no pipelining strategy is used, one clock cycle corresponds to one time-step.

[1] $1 + 2 \ldots n = \frac{n\times(n+1)}{2}$

[2] Can be simplified to $L \times (\frac{2\times n - L - 3}{2})$

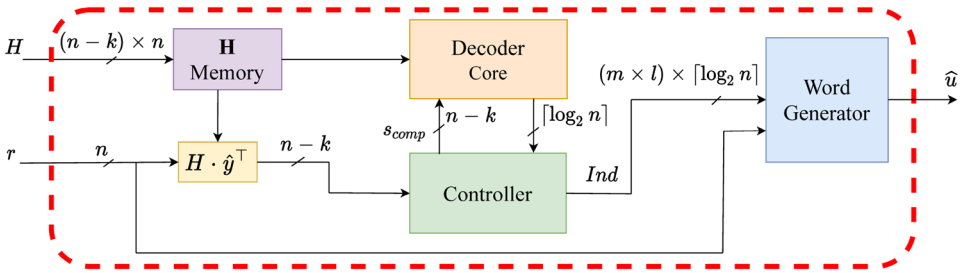


Figure 16 The Proposed VLSI Architecture for GRAND-MO.

For $n = 128$ and parameters ($m = 2, l_1 \leq 32$ and $l_2 \leq 32$) 3538 cycles (3) are required in the worst-case (W.C.) scenario, resulting in a W.C. latency of 7.0 μs. Figure 18(a) depicts the average latency of the proposed GRAND-MO architecture for RLC (128, 104) with different parameters and compares it to the average latency of GRANDAB [12] VLSI architecture. This average latency is computed using the bit-true C model, for the proposed GRAND-MO and GRANDAB, after considering at least 100 frames in error for each $\frac{E_b}{N_0}$ point. Regardless of the parameters, we can see that the average latency decreases as the channel condition improves, eventually reaching 1 clock cycle per decoded codeword. The counterpart of the average latency, the average throughput, is given in Fig. 18(b). It should be noted that the average information throughput increases with $\frac{E_b}{N_0}$, reaching values of up to 52 Gbps for the (128,104) RLC code as shown in Fig. 18(b).

In comparison to the GRANDAB decoder (AB = 3) [12], the proposed GRAND-MO decoder ($m = 2$, $l_1 \leq 32$ and $l_2 \leq 32$) has a 2.8× area overhead and is 2.7× less energy efficient than the GRANDAB decoder [12]. The average decoding throughput of the proposed GRAND-MO and GRANDAB decoders are of comparable magnitude. The proposed GRAND-MO decoder, on the other hand, has 13.6% percent lower W.C. latency, resulting in 15.8% percent higher W.C. decoding throughput. Moreover, the proposed GRAND-MO with parameters ($g = 0.4$, $m = 2$, $l_1 = 32$, and $l_2 = 16$) outperforms the GRANDAB decoder in decoding performance by at least 3 dB for target FERs $\leq 10^{-5}$ on RLC (128, 104), as shown in Fig. 9.

The GRAND-MO architecture with parameters ($n = 128$, $m = 1, l_1 \leq 128$) can support a single noise burst of length l ($\forall\, l \in [1, 128]$) and has a worst-case latency of 5 ns (requires 2 cycles), resulting in a worst-case throughput of 20.8 Gbps. When proposed GRAND-MO ($n = 128$, $m = 1$, $l_1 \leq 128$) is compared to GRANDAB decoder (AB = 3) [12], the GRANDAB is 6.5× more area-efficient, 2× more energy-efficient, and has a 25% higher average throughput. However, when compared to GRANDAB, the GRAND-MO ($n = 128$, $m = 1, l_1 \leq 128$) achieves 1640× higher worst-case throughput. Furthermore, for RLC (128, 104), the GRAND-MO with parameters ($g = 0.05$, $m = 1, l_1 = 110$) outperforms the

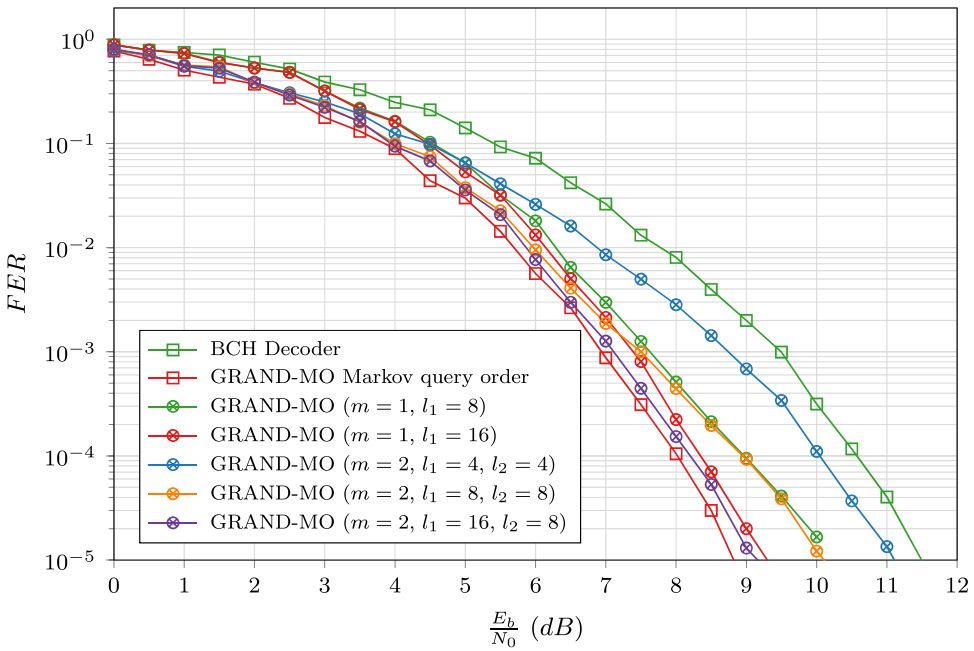


Figure 17 A Comparison of GRAND-MO and PGZ decoding performance for BCH code (79, 64) in Markov channels ($g = 0.4$).

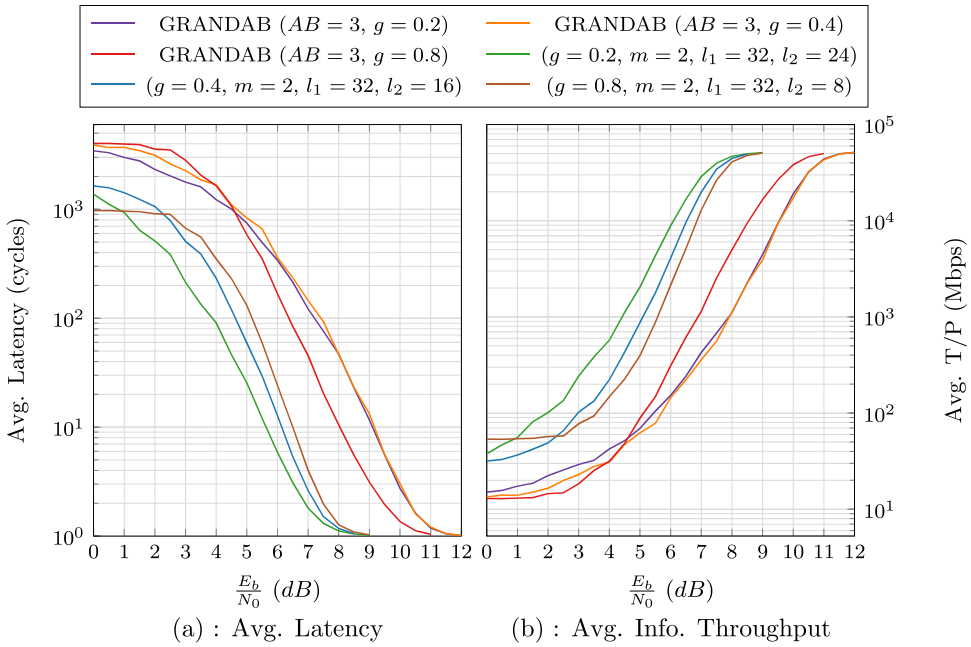


Figure 18 Comparison of average latency and average information throughput for the GRANDAB ($AB = 3$) [12] architecture and the proposed GRAND-MO architecture with parameters (g, $m = 2, l_1, l_2$) for the (128,104) RLC code.

GRANDAB decoder in decoding performance by at least 3 dB for target FERs $\leq 10^{-5}$, as shown in Fig. 9. Please note that traditional channel code decoders will require interleavers/deinterleavers to scatter burst noise across multiple codewords, which will introduce latency into the communication system. We refer the reader to [11] for a detailed comparison of the number of interleavers/deinterleavers required for traditional channel code decoders with varying amounts of channel memory. GRAND-MO does not need these interleavers/deinterleavers, allowing for reliable communication even in the presence of burst noise.

A high throughput VLSI architecture based on the PGZ algorithm for a (79,64) BCH code decoder [13] was recently proposed. Table 2 compares the synthesis results for GRAND-MO ($n = 79, m = 1, l_1 \leq 16$), which can support a single noise burst of length l ($\forall\ l \in [1, 16]$), and the BCH decoder in [13]. In the W.C. scenario, the GRAND-MO decoder with parameters ($m = 1, l_1 = 16$) requires 2 cycles. The BCH decoder in [13] is 63× more area efficient and 28× more energy efficient than the GRAND-MO decoder. However, in the proposed GRAND-MO architecture, the W.C. latency is reduced by 33%, resulting in a W.C throughput of 32 Gbps. The proposed GRAND-MO ($m = 1, l_1 = 16$) decoder achieves an average information throughput of up to 64 Gbps for a target FER of $\leq 10^{-5}$, while the BCH decoder is limited to 58 Gbps. Furthermore, the proposed GRAND-MO architecture can decode

Table 1 TSMC 65 nm CMOS Synthesis Comparison for GRANDAB with GRAND-MO for $n = 128$.

	GRANDAB [12]	GRAND-MO	GRAND-MO
Parameters	$AB = 3$	$m = 2, l_1 \leq 32$ and $l_2 \leq 32$	$m = 1, l_1 \leq 128$
Technology (nm)	65	65	65
Supply (V)	0.9	0.9	0.9
Max. Frequency (MHz)	500	500	400
Area (mm^2)	0.25	0.71	1.3
W.C. Latency (ns)	8196	7076	5
Avg. Latency (ns)	2	2	2.5
W.C. T/P (Mbps)[a]	12.68	14.69	20800
Avg. T/P (Gbps)[a]	52	52	41.6
Power (mW)	41	113	65.97
Energy per Bit (pJ/bit)	0.788	2.17	1.58
Area Efficiency (Gbps/mm^2)	208	73.24	32
Code compatible	Yes	Yes	Yes
Rate compatible	Yes	Yes	Yes

[a] For $k = 104$

Table 2 TSMC 65 nm CMOS Synthesis Comparison for BCH decoder with GRAND-MO ($m = 1, l_1 \leq 16$) for $n = 79$.

	GRAND-MO	(79,64) BCH decoder [13]
Technology (nm)	65	65
Supply (V)	1.1	1.2
Max. Frequency (GHz)	1	N/A
Area (μm^2)	225964	3264
W.C. Latency (ns)	2	3
Avg. Latency (ns)	1	1.1
W.C. T/P (Gbps)[a]	32	21.3
Avg. T/P (Gbps)[a]	64	58.2
Power (mW)	39.9	1.29
Energy per Bit (pJ/bit)	0.62	0.022
Area Efficiency (Gbps/mm^2)	283	17830
Code compatible	Yes	No
Rate compatible	Yes	No

[a] For $k = 64$

any code with $n = 79$ and $R \geq 0.75$, whereas the BCH decoder [13] can only decode the (79,64) BCH code. Additionally, at a target FER of $\leq 10^{-5}$, the GRAND-MO decoder with parameters ($m = 1, l_1 = 16$) provides a 2 dB gain in decoding performance over the BCH decoder [13], as shown in Fig. 17.

6 Conclusion

We propose a highly parallelized VLSI architecture for the code-agnostic GRAND-MO algorithm in this paper. GRAND-MO is a member of the GRAND family of universal channel code decoding algorithms. GRAND algorithms guess the noise introduced by the communication channel to decode the received channel signal. Using GRAND-MO in a communication channel with memory eliminates the need for interleavers/deinterleavers in a communication system, which reduces the decoding latency and enables low latency communication. We propose modifications to the test error pattern generation for GRAND-MO, which results in a simpler hardware implementation. Furthermore, the proposed test error pattern generation scheme is highly parametrizable and can be modified to meet the target decoding performance and complexity budget of a specific application. ASIC synthesis results show that for a code length of 128 and a target FER of 10^{-5}, an average decoding throughput of 52 Gbps can be achieved. The proposed VLSI architecture outperforms the GRANDAB, a hard-input variant of GRAND, in decoding performance by at least 3 dB at target FER of 10^{-5} with RLC (128, 104). When compared to the BCH decoder designed for a (79, 64) code, the proposed VLSI implementation achieves 33% lower worst case latency while also providing a 2 dB gain in decoding performance for a target FER of 10^{-5}. Moreover, for the same parameters, the average throughput can reach 64 Gbps. Because GRAND and its variants are code agnostic, the proposed GRANDMO VLSI architecture can be used to decode any linear block code of length $n = 128$ and code-rates between 0.75 and 1. Finally, the proposed architecture paves the way for future GRAND-MO algorithm implementations that use soft information as input.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Durisi, G., Koch, T., & Popovski, P. (2016). Toward massive, ultrareliable, and low-latency wireless communication with short packets. *Proceedings of the IEEE, 104*(9), 1711–1726.
2. 3GPP. (2018). Study on physical layer latency enhancements for NR ultrareliable and low case (URLLC). http://www.3gpp.org/DynaReport/38-series.htm, Tech. Rep. TR 38.824, 2018, Rel. 16.
3. Parvez, I., Rahmati, A., Guvenc, I., Sarwat, A. I., & Dai, H. (2018). A survey on low latency towards 5g: Ran, core network and caching solutions. *IEEE Communications Surveys Tutorials, 20*(4), 3098–3130.
4. Ma, Z., Xiao, M., Xiao, Y., Pang, Z., Poor, H. V., & Vucetic, B. (2019). High-reliability and low-latency wireless communication for internet of things: Challenges, fundamentals, and enabling technologies. *IEEE Internet of Things Journal, 6*(5), 7946–7970.
5. Zhan, M., Pang, Z., Dzung, D., & Xiao, M. (2018). Channel coding for high performance wireless control in critical applications: Survey and analysis. *IEEE Access,* vol. 6, pp. 29 648–29 664.
6. Chen, H., Abbas, R., Cheng, P., Shirvanimoghaddam, M., Hardjawana, W., Bao, W., et al. (2018). Ultra-reliable low latency cellular networks: Use cases, challenges and approaches. *IEEE Communications Magazine, 56*(12), 119–125.
7. Duffy, K. R., Li, J., & Médard, M. (2019). Capacity-achieving guessing random additive noise decoding. *IEEE Transactions on Information Theory,* vol. 65, no. 7, pp. 4023–4040.
8. Duffy, K. R. (2021). Ordered reliability bits guessing random additive noise decoding. In ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8268–8272.
9. Duffy, K. R., & Médard, M. (2019). Guessing random additive noise decoding with soft detection symbol reliability information - sgrand. In 2019 IEEE International Symposium on Information Theory (ISIT), pp. 480–484.
10. Solomon, A., Duffy, K. R., & Médard, M. (2020). Soft maximum likelihood decoding using grand. In ICC 2020–2020 IEEE International Conference on Communications (ICC), pp. 1–6.
11. An, W., Médard, M., & Duffy, K. R. (2020). Keep the bursts and ditch the interleavers. In GLOBECOM 2020–2020 IEEE Global Communications Conference, pp. 1–6.

12. Abbas, S. M., Tonnellier, T., Ercan, F., & Gross, W. J. (2020). High-throughput VLSI architecture for GRAND. In *IEEE Workshop on Signal Processing Systems (SiPS),* 1–6.
13. Choi, S., Ahn, H. K., Song, B. K., Kim, J. P., Kang, S. H., & Jung, S. (2019). A decoder for short BCH codes with high decoding efficiency and low power for emerging memories. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 2, pp. 387–397.
14. Abbas, S. M., Jalaleddine, M., & Gross, W. J. (2021). High-Throughput VLSI Architecture for GRAND Markov Order. In *IEEE Workshop on Signal Processing Systems (SiPS),* 158–163.
15. Gilbert, E. N. (1960). Capacity of a burst-noise channel. *The Bell System Technical Journal, 39*(5), 1253–1265.
16. Bose, R. C., & Ray-Chaudhuri, D. K. (1960). On a class of error correcting binary group codes. *Information and Control*, vol. 3, no. 1, pp. 68–79, 1960.
17. Hocquenghem, A. (1959). Codes correcteurs d'erreurs, Chiffres, 1959.
18. Gallager, R. G. (1968). Information theory and reliable communication.
19. Coffey, J. T., & Goodman, R. M. (1990). Any code of which we cannot think is good. *IEEE Transactions on Information Theory, 36*(6), 1453–1461.
20. Peterson, W. W., & Brown, D. T. (1961). Cyclic codes for error detection. *Proceedings of the IRE, 49*(1), 228–235.
21. Berlekamp, E. (1968). Nonbinary BCH decoding (abstr.). *IEEE Transactions on Information Theory, 14*(2), 242.
22. Massey, J. (1969). Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory, 15*(1), 122–127.
23. An, W., Médard, M., & Duffy, K. R. (2021). CRC codes as error correcting codes. arXiv preprint arXiv:2104.13663.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.