II. BACKGROUND

GRAND is a class of decoding algorithms for channel coding that concern themselves with the effects of noise. The core idea is to sequentially generate, from most to least likely, the binary noise effects that potentially corrupted the original message, based on knowledge of channel statistics or soft information. Each noise effect $z^n$ is removed from the demodulated channel output $y^n$, producing $\hat{x}^n = y^n \odot z^n$. If $\hat{x}^n$ is in the codebook, then the noise effect $z^n$ is the most likely one to have occurred, and $\hat{x}^n$ is returned as a maximum-likelihood decoding. GRAND is code-agnostic because this guessing process does not depend on code structure, and requires only a method of checking codebook membership, such as a syndrome computation for linear codes [18].

Both hard-input and soft-input variants of GRAND exist. Of interest here is ORBGRAND, which works as follows. Given rank-ordered reliability values $\{r_i\}$ for the demodulated bits $y^n$, where $r_i \geq 0$ and $r_i < r_j$ for $i < j$, the likelihood of a putative noise sequence $z^n$ is proportional to $f(z^n) = \sum_{i=1}^{n} r_i z_i^n$. The basic version of ORBGRAND approximates the reliability values with a line through the origin, $r_i = \beta_i$, $\beta \geq 0$, and based on this model it efficiently generates noise sequences in approximate maximum-likelihood order. The full version of the algorithm instead uses a multiline model. Here, we consider the 1-line model, which distinguishes itself from basic ORBGRAND by potentially having a non-zero intercept. 1-line ORBGRAND uses the approximation $r_i = \alpha + \beta i$, where $\alpha, \beta \geq 0$. Then $f(z^n) = \sum_{i=1}^{n} (\alpha + \beta i)z_i^n = \alpha \sum_{i=1}^{n} z_i^n + \beta \sum_{i=1}^{n} i z_i^n = \alpha w_H(z^n) + \beta w_L(z^n)$, where $w_H(z^n)$ is the Hamming weight and $w_L(z^n)$ is the logistic weight of $z^n$. We assume $c = \alpha/\beta$ to be a non-negative integer, and let $w_T(z^n) = f(z^n)/\beta = cw_H(z^n) + \beta w_L(z^n)$ be the total weight of a noise sequence, where $w_T(z^n) \in \{0, 1, 2, \ldots\}$.

The full ORBGRAND algorithm [7] details how to efficiently generate putative noise sequences in order of their total weight, which is an approximation of their maximum-likelihood order, without the need for dynamic memory. We detail a simple method for the 1-line version in section III, and also a method to find a suitable value for $c$.

A 2-dimensional product code [14] is a concatenation of two systematic component codes, $C_i$, for $1 \leq i \leq 2$. $C_i$ has parameters $[n_i, k_i, R_i, d_i]$, where $n_i$ is its code length, $k_i$ is its number of information symbols, $R_i = k_i/n_i$ is its code rate, and $d_i$ is its minimum Hamming distance. The input

---

Kevin Galligan  
Hamilton Institute  
Maynooth University, Ireland  
kevin.galligan.2020@muemail.ie

Muriel Médard  
Research Laboratory of Electronics  
Massachusetts Institute of Technology, USA  
medard@mit.edu

Ken R. Duffy  
Hamilton Institute  
Maynooth University, Ireland  
Ken.Duffy@mu.ie

---

Abstract—Guessing Random Additive Noise Decoding (GRAND) is a family of universal decoding algorithms suitable for decoding any moderate redundancy code of any length. We establish that, through the use of list decoding, soft-input variants of GRAND can replace the Chase algorithm as the component decoder in the turbo decoding of product codes. In addition to being able to decode arbitrary product codes, rather than just those with dedicated hard-input component code decoders, results show that ORBGRAND achieves a coding gain of up to 0.7dB over the Chase algorithm with same list size.

Index Terms—GRAND, Block Turbo Decoding, Product Codes, List Decoding

I. INTRODUCTION

GRAND is a recently introduced family of hard [1–3] and soft [4–7] decoding algorithms that can accurately decode arbitrary codes, even non-linear ones [8], with a moderate number of redundant bits. Their promise for practical, highly-parallelised decoding has led to the publication of several circuit designs [9–11] and chip implementations [12, 13].

Product codes are a class of long, high redundancy codes that are constructed by concatenating shorter component codes. Iterative GRAND (IGRAND) [3] adapts GRAND for accurate hard-input iterative decoding of product codes [14]. Here we establish how GRAND can decode product codes with the aid of soft information. Block turbo decoding achieves near-optimal soft-input decoding of product codes. Block turbo decoding uses a soft-input component decoder, customarily the Chase algorithm [15], to generate a selection of candidate decodings for a row or column of the product code, which are used to update the reliability of each bit in that row or column.

We reconsider soft-input GRAND as a list decoding algorithm and use it to replace the Chase algorithm. The decoding list produced by GRAND can be used to update the bit reliabilities as before, with the benefit that GRAND’s code-agnosticism allows it to turbo decode any product code. We present results for block turbo decoding with Ordered Reliability Bits GRAND (ORBGRAND) [6, 7, 16, 17], a soft-input variant of GRAND that is particularly suited to hardware implementation [10, 13]. The results are based on an additive white Gaussian noise (AWGN) channel model with binary phase shift keying (BPSK) modulation. We also provide analytical support for list decoding with GRAND algorithms and consider the standalone list decoding performance of ORBGRAND.

978-1-6654-5181-9/23/$31.00 ©2023 IEEE
symbols are arranged as a $k_2 \times k_1$ array. The rows of this array are extended by encoding them with $C_1$, then the columns are extended by encoding them with $C_2$. The result is an $n_2 \times n_1$ array, all rows and columns of which are codewords of $C_1$ and $C_2$. This is a codeword of the product code, with parameters $[n_1, n_2, k_1, k_2, R_1, R_2, d_1, d_2]$. We use $C(n, k, d)^2$ to denote a product code with row and column code $C$, where $C$ has parameters $[n, k, k/n, d]$. Pyndiah [19] extended the turbo decoding technique from convolutional codes to product codes, where row decoding output informs the soft input of the column decoding, and vice versa, as we describe later. Hard-input GRAND algorithms have been applied to product codes and related coding structures [3, 20, 21].

III. TURBO AND LIST DECODING WITH GRAND

List decoding [22] is a decoding procedure in which the decoder outputs a list of $L$ codewords. The original GRAND algorithm stops the guessing process once it finds a codeword, since that codeword is a maximum-likelihood decoding. In a form of list decoding, Abbas et al. [16] recently proposed an extension to basic ORBGRAND in which codewords are accumulated within some distance of the first codeword that is found and the resulting list is used to improve hard-output accuracy. Here, we provide analytical support for list decoding with any GRAND algorithm, based on theorems from [1]. GRAND can list decode by continuing its guessing process until it has accumulated $L$ codewords, rather than stopping after the first one. This is described in Algorithm 1.

Algorithm 1: GRAND list decoding of hard channel output $y$, possibly with soft output $r$ informing the likelihood of noise effects. Given a codebook $C$, code length $n$ and a target list size $L$.

1. $\mathcal{L} \leftarrow \{\}$
2. $z^* \leftarrow 0^n$; // all-zero is most likely
3. while $|\mathcal{L}| < L$ do
4.   $c^* \leftarrow y \oplus z^*$; // undo noise effect
5.   if $c^* \in C$ then
6.     add $c^*$ to $\mathcal{L}$;
7.   end if
8. end while
9. return $\mathcal{L}$;

Underlying GRAND is a race between two processes: the number of guesses to find the true channel noise effect $Z^n : \Omega \to \{0,1\}^n$, which recovers the correct codeword, and the number of guesses $U : \Omega \to \{1,\ldots,2^n\}$ before GRAND identifies an incorrect codeword. GRAND’s guessing order is defined by a bijective function $G : \{0,1\}^n \to \{1,\ldots,2^n\}$ that maps each noise effect to its position in the guessing order. The number of guesses to identify the true channel noise effect is $G(Z^n)$. GRAND identifies the correct decoding when $G(Z^n) < U$, the asymptotic probability of which as $n$ tends to infinity is derived in [1] for uniform random codebooks.

We now examine the case of list decoding with GRAND, and its approximate complexity. Consider a random binary code of length $n$ with $2^k$ codewords. For list size $L = 2^h$, denote the position of the $i$-th incorrect codeword in GRAND’s guessing order by the random variable $U_i : \Omega \to \{1,\ldots,2^n\}$, where $1 \leq i \leq 2^k - 1$. As the codebook is uniformly at random, the $\{U_i\}$ appear uniformly in the guesswork order $\{1,\ldots,2^n\}$. Let the codewords be ordered such that $U_1 < U_2 < \ldots < U_{2^k-1}$. Given hard channel output $Y^n : \Omega \to \{0,1\}^n$ and the $i$-th codeword $C_i \in C$, $U_i = G(Y^n \oplus C_i)$, since $C_i = Y^n \oplus (Y^n \oplus C_i)$. The total number of guesses to accumulate codewords is $\sum_{i=2^k-1}^{n+k-1} U_i$.

The expected total number of guesses $\mathbb{E}[Y_L]$ is derived as follows. Since $\mathbb{E}[U_1] = \sum_u \mathbb{E}[U_1|U_2 = u]P(U_2 = u) = (1/2)\sum_u uP(U_2 = u) = \mathbb{E}[U_2]/2$, the expected guesses from the first codeword to the second is $\mathbb{E}[U_2 - U_1] = \mathbb{E}[U_2] - \mathbb{E}[U_1] = \mathbb{E}[U_2]/2 = \mathbb{E}[U_1]$. A similar argument proves $\mathbb{E}[U_i - U_{i-1}] = \mathbb{E}[U_1]$ for all $i$, and $\mathbb{E}[2^k - U_{2^k-1}] = \mathbb{E}[U_1]$, which is the expected number of guesses from the final codeword to the last binary sequence that GRAND can guess; thus, $\mathbb{E}[Y_L] = LE[U_1]$. The expected number of guesses to cover all $2^n$ possible noise effects is $\mathbb{E}[\sum_{i=1}^{2^n} U_i - U_{i-1}] = 2^k \mathbb{E}[U_1] = 2^n$, where $U_0 = 0$ and $U_{2^n} = 2^n$. Hence $\mathbb{E}[U_1] = 2^{n-k}$ and $\mathbb{E}[Y_L] = LE[U_1] = 2^{n-k+1}$.

The above argument informs the choice of list size and code rate in coding scheme design. As $n$ becomes large, $G(Z^n) \leq 2^{nH}$ with high likelihood, where $H$ is the Shannon entropy of the channel noise [23]. The correct codeword ends up on the decoding list with high likelihood when $G(Z^n) < \mathbb{E}[Y_L]$, which is true when $2^{nH} < \mathbb{E}[Y_L] = 2^{n-k+1} = 2^{n(1-H+\theta)}$. Letting $l = n\theta$ for $\theta > 0$, the requirement becomes $2^{nH} < 2^{n(1-R+\theta)}$, or $H < 1 - R + \theta$. Stated in a form closer to the noisy-channel coding theorem [24], $R < 1 - H + \theta$. This tells us that by increasing the list size we can perform effective channel coding at higher code rates, as asserted in [22].

Regarding complexity, $2^{n-k+1}$ is an upper bound on the expected number of GRAND queries for random codebooks. From this arises a design trade-off between list size and the number of parity bits. To keep the bound constant, a parity bit must be removed if the list size is doubled. The bound corresponds to the expected number of queries to identify $L$ incorrect codewords, although in practice the correct codeword will typically be added to the list after a small number of queries and fewer overall queries will be required as a result.

We now turn to the block turbo decoding algorithm of Pyndiah [19]. The decoding of a single component (row or column) of the product code, with accompanying per-bit soft information $R \in \mathbb{R}^n$, works as follows:

1) Apply Chase decoding to produce a list $\mathcal{L} \subseteq C$, where $1 \leq |\mathcal{L}| \leq 2^v$ for some small positive integer $p$.
2) Select $D = \text{argmin}_{C \in \mathcal{L}}|R - C|^2$ as the new value of the component, where $|R - C|^2$ is the Euclidean distance between $R$ and the modulated form of $C$.
3) Individually update the soft information of each bit in the component. If there is a codeword that disagrees with $D$ on the value of the $i$-th bit, $C^* \in \text{argmin}_{C \in \mathcal{L}, \not D_i} |R - C|^2$, then the soft output for that bit is given by $r_i = \ldots$

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
(2D_n - 1)/(|R - C_n|^2 - |R - D_n|^2)/4.
4) If \( \arg \min_{C \in \mathcal{Z}, C \neq D_n} |R - C|^2 \) is empty, then instead use \( r_n = \beta \) for some constant \( \beta \geq 0 \).

A soft-input list decoding variant of GRAND can replace the Chase algorithm in step (1) of the block turbo decoding algorithm, with the remainder of the algorithm untouched. Indeed, a variant of ORBGRAND has independently been proposed [17] for use in Pyniah-style soft-input soft-output iterative decoding of a concatenated code from [25]. Iterative soft-input soft-output GRAND decoding has also recently been considered in [26].

We have described how \( 2^{n-k+l} \) is an upper bound on the expected number of GRAND queries for list decoding of a random \( n \)-bit code with list size \( L = 2^l \). This leads to a bound on the average decoding complexity of block turbo decoding with GRAND. Given a product code whose row and column codes are random codes of length \( n \) with \( k \) information bits, and given a maximum of \( N \) decoding iterations where \( 2n \) component codes are decoded in each iteration, an upper bound on the average number of GRAND codebook queries during block turbo decoding is \( nN 2^{n-k+l+1} \).

An advantage of GRAND as a component decoder is that it can decode any component code, and thus can turbo decode any product code. The Chase algorithm requires that a specialised hard-input decoder exists for the component codes, which for example is not the case for Random Linear Codes (RLCs) and CRC codes. GRAND also distinguishes itself by populating its list with codewords in maximum-likelihood order, assuming its query order is correct, while Chase makes no such guarantees and may output duplicate codewords.

ORBGRAND is a practical component decoder for turbo decoding, given its accuracy and efficiency. As turbo decoding converges, however, the distribution of reliability values shifts upwards, as in Fig. 1, which tends to make basic ORBGRAND’s linear approximation a poorer fit. We thus propose to turbo decode with the full ORBGRAND algorithm, parameterised to use a single line. This allows ORBGRAND’s model to have a non-zero intercept and so better capture the reliability distribution during later iterations of turbo decoding.

In Algorithm 2, we present a simple method to construct noise sequences for 1-line ORBGRAND that has similar implementation complexity as basic ORBGRAND. Noise sequences are generated in order of their total weight, \( w_T \). For each \( w_T \), we iterate over all pairs of non-negative integers \((w_H, w_L)\) such that \( w_T = cw_H + w_L \), and the landslide algorithm [7] generates all noise sequences for each pair. Parity constraints [27] may be used to halve the number of codebook queries performed by this algorithm. For even-parity codes, such as eBCH codes, the parity of the noise effect \( \mathbf{e}^n \) can be inferred from \( \mathbf{y}^n \), since, given codeword \( \mathbf{c}^n \), \( \sum_{i=0}^n y^n_i = \sum_{i=0}^n (c^n_i + z^n_i) = \sum_{i=0}^n c^n_i + \sum_{i=0}^n z^n_i = \sum_{i=0}^n s^n_i \) (mod 2). Incorrect-parity Hamming weights can then be skipped.

A simple and effective method to pick \( c \) is to fit a line through the points \((1, r_1)\) and \((n/2, r_{n/2})\), where \( n \) is the code length. Then \( \beta = (r_{n/2} - r_1)/(n/2 - 1) \) and \( c = \max(0, [(r_1 - \beta)/\beta]) \), where \( \lfloor \cdot \rfloor \) rounds to the nearest integer.

This gives the best estimate of \((1, r_1)\), the most important point, and accurately approximates the remaining points if they follow a line-like distribution as in Fig. 1.

### Algorithm 2: Noise effect generation algorithm for 1-line ORBGRAND, given integer parameter \( c \geq 0 \) and code length \( n \).

1. \( \text{yield } 0^n; // \text{ all-zero is most likely} \)
2. \( w_T \leftarrow c + 1; // \text{ minimum possible weight} \)
3. while \( w_T \leq cn + \frac{n(n+1)}{2} \) do
   4. \( w_H \leftarrow \max(1, \lfloor \frac{1 + 2(n+c) - \sqrt{(1 + 2(n+c))^2 - 8w_T}}{2} \rfloor); \)
   5. while \( w_H \leq n \) do
      6. \( w_L \leftarrow w_T - cw_H; \)
      7. if \( w_L < \frac{w_H(w_H+1)}{2} \) then
         8. \( \text{break }; // \text{ invalid pair} \)
   9. \( \text{yield noise effects generated by Landslide}(w_H, w_L, n); \)
10. \( w_H \leftarrow w_H + 1; \)
11. \( w_T \leftarrow w_T + 1; \)

IV. PERFORMANCE EVALUATION

We begin with a study of ORBGRAND’s list decoding performance, since this is critical to its performance as a turbo component decoder. We run simulations in an AWGN channel with BPSK modulation. Fig. 2 shows the list decoding block error rate (BLER) of basic ORBGRAND versus that of Chase decoding for an extended BCH code [18], eBCH(32, 26, 4), which is one of the component codes from [19]. A list decoding block error occurs when, given transmitted codeword \( \mathbf{C} \) and output decoding list \( \mathcal{L}, \mathbf{C} \notin \mathcal{L} \). The list size \( L \) ranges from 4 to 16 and \( L = 2^l \) corresponds to a Chase parameter of \( \rho = l \). At a BLER of \( 10^{-5} \) and \( L = 16 \), basic ORBGRAND provides a coding gain of 1 dB over Chase. That ORBGRAND...
outperforms Chase as a list decoder is a promising indicator that it will be a good turbo component decoder.

Fig. 3 shows the basic ORBGRAND list decoding BLER of a BCH code, BCH(31, 21, 5), versus a random linear code, RLC(31, 21, 4). The RLC was purposely constructed to have a lower minimum distance. Its decoding is only enabled by ORBGRAND’s code-agnosticism. As the list size increases, the performance of the two codes converges, which is consistent with Elias’s claim that a larger list size should compensate for structural weakness in a code [22]. This suggests that powerful product codes can be constructed from imperfect component codes, but later results will demonstrate this is not true.

Fig. 4 shows the list decoding accuracy of a further selection of codes with basic ORBGRAND: eBCH(32, 26, 4), RLC(32, 26, 3) and CRC(32, 26, 3). The CRC polynomial is 0x33 in Koopman notation [28]. The list size ranges from 4 to 16. Performance is essentially equivalent at these list sizes, despite the RLC and CRC having lower minimum distance.

Fig. 5 and Fig. 6 show the bit error rate (BER) of eBCH(32, 26, 4) and eBCH(64, 57, 4) product codes with block turbo decoding. 1-line ORBGRAND and Chase are used as component decoders. These codes were tested in [19], and, as in that paper, we run turbo decoding for 4 iterations. The list sizes are 4, 8 and 16. Decoding is halted if the rows or columns are error-free at the beginning of any iteration.

Fig. 5 concerns an eBCH(32, 26, 4) code (n = 1024, k = 676, d = 79) and, as component decoders. These codes were tested in [19], and, as in that paper, we run turbo decoding for 4 iterations. The list sizes are 4, 8 and 16. Decoding is halted if the rows or columns are error-free at the beginning of any iteration.

Fig. 6 shows results for an eBCH(64, 57, 4) code (n = 4096, k = 3249, R = 0.79). Again, when L = 16, 1-line ORBGRAND provides a coding gain of approximately 0.2dB at a BER of 10^{-5}. Even with L = 4, 1-line ORBGRAND achieves nearly the same performance, whereas the performance of Chase degrades rapidly as the list size decreases.

Fig. 7 shows the turbo decoding accuracy of product codes whose component codes are the same as in Fig. 4, with 1-line ORBGRAND decoding. Despite having equivalent list decoding performance, the RLC and CRC fare worse than the
eBCH code as component codes, and so product codes appear to compound structural weakness in their component codes.

Fig. 8 shows the average number of codebook queries, a standard measure of GRAND complexity [7, 16], performed by 1-line ORBGRAND with parity constraints [27] during turbo decoding of eBCH(32, 26, 4)$^2$ and eBCH(64, 57, 4)$^2$. Fig. 9 shows the same data but as a ratio over the average codebook queries when $L = 4$.

V. CONCLUSIONS

GRAND algorithms can list decode accurately, and soft-input list decoding GRAND algorithms are a viable replacement for Chase as the component decoder in block turbo decoding. We have presented a code for which basic ORBGRAND list decoding gains as much as 1dB over Chase at a BLER of $10^{-5}$. For turbo decoding, the distribution of rank-ordered soft information shifts so that the full version of ORBGRAND is required for effective component decoding. Turbo decoding simulations show that 1-line ORBGRAND gains up to 0.7dB over Chase at a BER of $10^{-5}$ for two different product codes, an improvement in accuracy that may be explained by the maximum-likelihood guarantees on GRAND’s list decoding output. We have characterised the list and turbo decoding complexity of GRAND, and with the recent low-energy synthesised ORBGRAND chip [13], it will be possible to estimate the energy consumption of block turbo decoding with ORBGRAND. As with any component decoder, the choice of list size is a trade-off between accuracy and complexity. Another way to control complexity is to terminate GRAND’s guessing procedure early and increase the reliability of all bits if it identifies a codeword within a small number of guesses, based on [29] and [17]. GRAND’s universality allows list and turbo decoding to be applied to codes without bespoke decoders, such as CRCs and RLCs, as well as product codes that are concatenations of those codes. This leads to the possibility of new channel coding applications.

Acknowledgment: This publication has emanated from research conducted with the financial support of Science Foundation Ireland under grant number 18/CRT/6049. The opinions, findings and conclusions or recommendations expressed
in this material are those of the author(s) and do not necessarily reflect the views of the Science Foundation Ireland. This work was partially supported by Defense Advanced Research Projects Agency contract number HR00112120008.

REFERENCES

[1] K. R. Duffy, J. Li, and M. Medard, “Capacity-achieving guessing random additive noise decoding,” IEEE Trans. Inf. Theory, vol. 65, no. 7, pp. 4023–4040, 2019.

[2] W. An, M. Mérédard, and K. R. Duffy, “Keep the bursts and ditch the interleavers,” in IEEE GLOBECOM, 2020.

[3] K. Galligan, A. Solomon, A. Riaz, M. Mérédard, R. T. Yazicigil, and K. R. Duffy, “IGRAND: decode any product code,” in IEEE GLOBECOM, 2021.

[4] K. R. Duffy, M. Mérédard, and W. An, “Guessing random additive noise decoding with symbol reliability information (SRGRAND),” in IEEE Trans. Commun., vol. 70, no. 1, 2022, pp. 3–18.

[5] A. Solomon, K. R. Duffy, and M. Mérédard, “Soft maximum likelihood decoding using GRAND,” in IEEE Int. Commun. Conf., 2020.

[6] K. R. Duffy, “Ordered reliability bits guessing random additive noise decoding,” in IEEE ICASSP, 2021.

[7] K. R. Duffy, W. An, and M. Medard, “Ordered reliability bits guessing random additive noise decoding,” in IEEE Trans. Sig. Proc., vol. 70, 2022, pp. 4528 – 4542.

[8] A. Cohen, R. G. L. D’Oliveira, K. R. Duffy, J. Woo, and M. Mérédard, “AES as Error Correction,” 2022, arXiv:2203.12047.

[9] S. M. Abbas, T. Tonnellier, F. Ercan, and W. J. Gross, “High-Throughput VLSI Architecture for GRAND,” in IEEE Workshop on Sig. Proc. Sys., 2020, pp. 681–693.

[10] S. M. Abbas, T. Tonnellier, F. Ercan, M. Jalaleddine, and W. J. Gross, “High-Throughput and Energy-Efficient VLSI Architecture for Ordered Reliability Bits GRAND,” IEEE Trans. on VLSI Sys., vol. 30, no. 6, 2022.

[11] S. M. Abbas, M. Jalaleddine, and W. J. Gross, “High-Throughput VLSI Architecture for GRAND Markov Order,” in IEEE Workshop Sig. Proc. Sys., 2021.

[12] A. Riaz, V. Bansal, A. Solomon, W. An, Q. Liu, K. Galligan, K. R. Duffy, M. Medard, and R. T. Yazicigil, “Multi-Code Multi-Rate Universal Maximum Likelihood Decoder using GRAND,” in ESSCIRC, 2021.

[13] A. Riaz, A. Yasar, F. Ercan, W. An, J. Ngo, K. Galligan, M. Mérédard, K. R. Duffy, and R. T. Yazicigil, “A Sub-0.8 pJ/bit, 16.3 Gbps/mm2 Universal Soft-detection Decoder using ORBGRAND in 40 nm CMOS,” in IISCC, in press.

[14] P. Elias, “Error-free Coding,” Trans. IRE Prof. Group Inf. Theory, vol. 4, no. 4, pp. 29–37, 1954.

[15] D. Chase, “Class of algorithms for decoding block codes with channel measurement information,” IEEE Trans. Inf. Theory, vol. 18, no. 1, pp. 170–182, 1972.

[16] S. M. Abbas, M. Jalaleddine, and W. J. Gross, “List-GRAND: A Practical Way to Achieve Maximum Likelihood Decoding,” IEEE Trans. on VLSI Sys., vol. 31, pp. 43–54, 2023.

[17] C. Condo, “Iterative soft-input soft-output decoding with ordered reliability bits GRAND,” 2022, arXiv:2207.06691.

[18] S. Lin and D. J. Costello, Error control coding: fundamentals and applications. Pearson/Prentice Hall, 2004.

[19] R. Pyndiah, “Near-optimum decoding of product codes: block turbo codes,” IEEE Trans. Commun., vol. 46, no. 8, pp. 1003–1010, 1998.

[20] I. Chatzigeorgiou, “Transversal GRAND for Network Coded Data,” in IEEE Int. Symp. on Inf. Theory, 2022.

[21] S. Allahkaram, F. A. Monteiro, and I. Chatzigeorgiou, “URLLC with Coded Massive MIMO via Random Linear Codes and GRAND,” 2022, arXiv:2208.00086.

[22] P. Elias, “List decoding for noisy channels,” in IRE WESCON Convention Record, 1957.

[23] M. M. Christiansen and K. R. Duffy, “Guesswork, Large Deviations, and Shannon Entropy,” IEEE Trans. Inf. Theory, vol. 59, no. 2, pp. 796–802, 2013.

[24] C. E. Shannon, “A mathematical theory of communication,” The Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, 1948.

[25] “OpenROADM MSA 5.0 W-Port Digital Specification,” 2021.

[26] H. Sarieddeen, M. Mérédard, and K. R. Duffy. “Soft-Input, Soft-Output Joint Detection and GRAND,” in IEEE GLOBECOM, 2022.

[27] M. Rowsan and J. Yuan, “Constrained Error Pattern Generation for GRAND,” in IEEE Int. Symp. on Inf. Theory, 2022.

[28] P. Koopman and T. Chakravarty, “Cyclic redundancy code (CRC) polynomial selection for embedded networks,” in Int. Conf. on Dep. Sys. and Net., 2004.

[29] S. Kerouedan, P. Adde, and R. Pyndiah, “How do we implement block turbo codes?” in Turbo Codes, Error-correcting codes of widening application, ser. Innovative technology, Inf. sys. and networks. Hermes Penton Science, 2002, pp. 127 – 141.