Step-GRAND: A Low Latency Universal Soft-input Decoder

Syed Mohsin Abbas and Chi-Ying Tsui
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
Hong Kong, China
Emails: smabbas@connect.ust.hk, eetsui@ust.hk

Marwan Jalaleddine and Warren J. Gross
Department of Electrical and Computer Engineering
McGill University
Montréal, Québec, Canada
marwan.jalaleddine@mail.mcgill.ca, warren.gross@mcgill.ca

***Abstract*—GRAND features both soft-input and hard-input variants that are well suited to efficient hardware implementations that can be characterized with achievable average and worst-case decoding latency. This paper introduces step-GRAND, a soft-input variant of GRAND that, in addition to achieving appealing average decoding latency, also reduces the worst-case decoding latency of the corresponding hardware implementation. The hardware implementation results demonstrate that the proposed step-GRAND can decode CA-polar code $(128, 105+11)$ with an average information throughput of 47.7 Gbps at the target FER of $\leq 10^{-7}$. Furthermore, the proposed step-GRAND hardware is $10\times$ more area efficient than the previous soft-input ORBGRAND hardware implementation, and its worst-case latency is $\frac{1}{6.8}\times$ that of the previous ORBGRAND hardware.**

***Index Terms*—Guessing Random Additive Noise Decoding (GRAND), GRAND with ABandonment (GRANDAB), Ordered Reliability Bits GRAND (ORBGRAND), Maximum Likelihood (ML) decoding, Ultra-Reliable and Low Latency Communications (URLLC).**

I. Introduction

The emergence of novel applications that have stringent requirements for URLLC [1] has recently rekindled a great deal of interest in short channel codes and associated ML decoding approaches. A few examples of these applications include augmented and virtual reality [2], intelligent transportation systems [3], the internet of things [4] and machine-to-machine communication [5].

For short-length and high-rate channel codes, GRAND [6] has recently been proposed as a universal ML decoding technique. Since GRAND is noise-centric and code-agnostic, it attempts to guess the noise that corrupted the codeword during transmission through the communication channel rather than relying on the structure of the underlying code to decode a codeword. GRAND guesses the noise by generating Test Error Patterns (TEPs), and the order in which these TEPs ($\boldsymbol{e}$) are generated is the primary difference between different GRAND variants.

For both hard-input [6][7] and soft-input [8][9] GRAND variants, several high-throughput and energy-efficient hardware implementations have been reported in the literature [10][11][12][13][14]. In general, the decoding latency of GRAND-based hardware implementations can be categorized into average and worst-case decoding latency. While the average decoding latency of GRAND hardware is typically much lower than the worst-case decoding latency, the latter can still pose a significant barrier to the adoption of GRAND based hardware implementations for applications that require strict adherence to both average and worst-case decoding latency requirements, such as the URLLC application scenario [1][2][3][4][5].

In this work, we introduce step-GRAND, a soft-input variant of GRAND that not only offers a low average decoding latency but also reduces the worst-case decoding latency compared to soft-input ORBGRAND [12]. We propose a simplified TEP generation scheme and develop a high-throughput VLSI architecture for the proposed step-GRAND. Furthermore, the proposed step-GRAND introduces parameters that can be adjusted to meet the desired decoding performance and complexity/latency constraints for a target application.

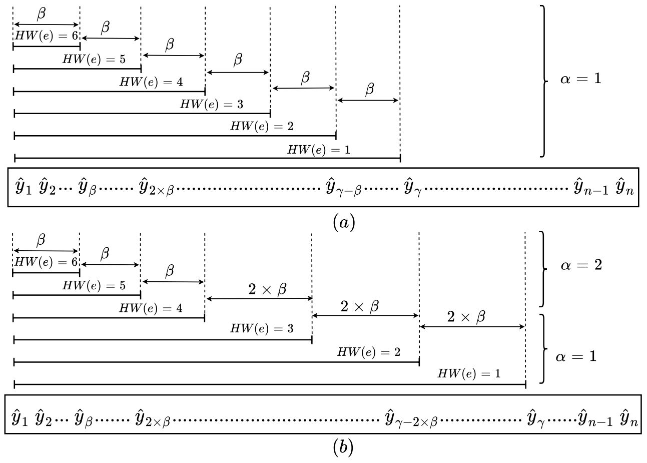
The VLSI implementation results show that for a linear block code with length 128 (n) and 105 information bits (k), the proposed step-GRAND hardware can achieve an average information throughput of 47.7 Gbps. Furthermore, in comparison to the previously proposed ORBGRAND hardware implementation [12], the proposed step-GRAND hardware is $10\times$ more area efficient and the worst-case latency of the step-GRAND hardware is $\frac{1}{6.8}\times$ the worst-case latency of previous ORBGRAND hardware [12].

The remainder of this paper is structured as follows: Preliminaries on GRAND are provided in Section II. Section III introduces the proposed step-GRAND. The numerical simulation results are presented in Section IV. The proposed step-GRAND VLSI architecture and its implementation results are presented in Section V. Finally, in Section VI, concluding remarks are made.

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 i^{th} element of a vector $\boldsymbol{v}$ is denoted by v_i. 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. Similarly, $\mathbb{1}_{i,j\ldots,z} = \mathbb{1}_i \oplus \mathbb{1}_j \oplus \ldots \mathbb{1}_z$, with $i \neq j \ldots \neq k$. All the indices start at 1. For this work, all



β
$HW(e)=6$
$HW(e)=5$
$HW(e)=4$
$HW(e)=3$
$HW(e)=2$
$HW(e)=1$
$\alpha = 1$
$\hat{y}_1\ \hat{y}_2 \cdots \hat{y}_\beta \cdots \hat{y}_{2\times\beta} \cdots \hat{y}_{\gamma-\beta} \cdots \hat{y}_\gamma \cdots \hat{y}_{n-1}\ \hat{y}_n$
(a)
$2\times\beta$
$\alpha = 2$
$\hat{y}_1\ \hat{y}_2 \cdots \hat{y}_\beta \cdots \hat{y}_{2\times\beta} \cdots \hat{y}_{\gamma-2\times\beta} \cdots \hat{y}_\gamma \cdots \hat{y}_{n-1}\ \hat{y}_n$
(b)

Fig. 1. The subsets of $\hat{\boldsymbol{y}}$ for the proposed step-GRAND TEP generation. (a) $P = 6$ and $\alpha = 1$ (b) $P = 6$ and $\alpha = 2$

Algorithm 1: Step-GRAND

```
   Input: $\boldsymbol{y}$, $\boldsymbol{H}$, $\boldsymbol{G}^{-1}$, $P$, $\alpha$, $\beta$
   Output: $\hat{\boldsymbol{u}}$
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      $\boldsymbol{ind} \leftarrow$ sort($\boldsymbol{y}$)        // $|\boldsymbol{y}_i| \leq |\boldsymbol{y}_j| \quad \forall i < j$
5      $\boldsymbol{e} \leftarrow \boldsymbol{0}$; $HW \leftarrow 1$
6      for $i \leftarrow 1$ to $\alpha$ do
7          $\gamma \leftarrow \frac{(\alpha-i+1)\times(\alpha-i+2)}{2} \times \frac{P}{\alpha} \times \beta$
8          for $j \leftarrow 1$ to $\frac{P}{\alpha}$ do
9              for $k \leftarrow 1$ to $\binom{\gamma}{HW}$ do
10                 $\boldsymbol{e} \leftarrow$ generateNewTEP($HW$,$\gamma$,$\boldsymbol{ind}$)
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 $\hat{\boldsymbol{u}}$
14             $HW \leftarrow HW + 1$
15             $\gamma \leftarrow \gamma - (\alpha - i + 1) \times \beta$
```

operations are restricted to the Galois field with 2 elements, noted $\mathbb{F}_2$. Furthermore, we restrict ourselves to (n, k) linear block codes.

B. GRAND Decoding

GRAND is centered around generating TEPs ($\boldsymbol{e}$), applying them to the hard-demodulated received vector $\hat{\boldsymbol{y}}$ and querying the resultant vector $\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$ for codebook membership as follows:

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

where $\boldsymbol{H}$ is a $(n-k) \times n$ parity check matrix of the code. If this codebook membership constraint (1) is satisfied, $\boldsymbol{e}$ is the guessed noise and $\hat{\boldsymbol{c}} \triangleq \hat{\boldsymbol{y}} \oplus \boldsymbol{e}$ is the estimated codeword.

III. Proposed step-GRAND

The proposed step-GRAND generates TEPs ($\boldsymbol{e}$) in increasing Hamming weight order, with the highest Hamming weight of the generated TEPs ($\boldsymbol{e}$) being P. However, unlike GRANDAB [7], which generates all $\binom{n}{i}$ TEPs for each Hamming weight $i \ \forall\ i \in [1, P]$, the step-GRAND only generates a subset of the TEPs for each Hamming weight.

The step-GRAND begins by generating $\binom{\gamma}{1}$ TEPs with a Hamming weight of 1 ($\boldsymbol{e} = \mathbb{1}_i$, with $i \in [\![1\ ..\ \gamma]\!]$), where γ is the size of the subset of $\hat{\boldsymbol{y}}$. After that, for the subset of $\hat{\boldsymbol{y}}$ of size $\gamma - \beta$, $\binom{\gamma-\beta}{2}$ TEPs ($\boldsymbol{e}$) with a Hamming weight of 2 ($\boldsymbol{e} = \mathbb{1}_{i,j}$, with $i \in [\![1\ ..\ \gamma-\beta]\!]$, $j \in [\![1\ ..\ \gamma-\beta]\!]$ and $i \neq j$,) are generated. Please note that β, termed as *step size*, refers to the size difference between two successive subsets of $\hat{\boldsymbol{y}}$. Similarly, for each subsequent Hamming weight $HW \in [\![3\ ..\ P]\!]$, $\binom{\gamma-(HW-1)\times\beta}{HW}$ TEPs ($\boldsymbol{e}$) are generated for the subset of $\hat{\boldsymbol{y}}$ of size $\gamma - (HW-1) \times \beta$. Figure 1 (a) displays the $P = 6$ subsets of $\hat{\boldsymbol{y}}$ for the proposed step-GRAND TEP generation.

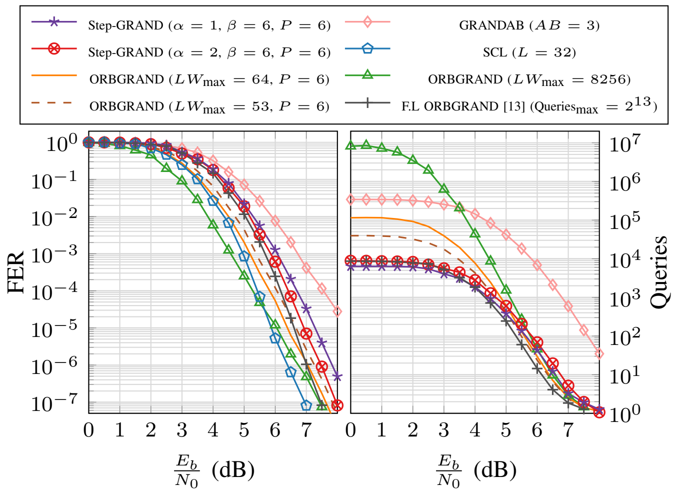
The P subsets of $\hat{\boldsymbol{y}}$ can also be divided into α segments, each of which has $\frac{P}{\alpha}$ subsets, as illustrated in Fig. 1 (b), where $P = 6$ and $\alpha = 2$. Furthermore, the *intra-segment step size* is a multiple of β such that the intra-segment step size of the i^{th} ($i \in [\![1\ ..\ \alpha]\!]$) segment is $(\alpha - i + 1) \times \beta$.

The step-GRAND decoding process is summarized in Algorithm 1. The inputs to the algorithm are the vector of channel observation values $\boldsymbol{y}$ of size n, an $(n-k) \times n$ matrix $\boldsymbol{H}$, an $n \times k$ matrix $\boldsymbol{G}^{-1}$ such that $\boldsymbol{G}^{-1} \cdot \boldsymbol{G}$ is the $n \times n$ identity matrix, with $\boldsymbol{G}$ a generator matrix of the code, the maximum Hamming weight P of TEPs ($\boldsymbol{e}$), the number of segments α and the step size β.

The step-GRAND sorts the received vector $\boldsymbol{y}$, Log-Likelihood Ratios (LLRs), in ascending order of absolute values of LLRs ($|\boldsymbol{y}_i| \leq |\boldsymbol{y}_j| \quad \forall i < j$), and the associated indices are recorded in a permutation vector denoted by $\boldsymbol{ind}$ (line 4). The input parameters α, β, and P are used to calculate the size (γ) of a subset of $\hat{\boldsymbol{y}}$ for segment i ($i \in [\![1\ ..\ \alpha]\!]$) (line 7). Following that, in segment i, all $\binom{\gamma}{HW}$ TEPs ($\boldsymbol{e}$) are generated for a particular Hamming weight HW corresponding to a subset j ($j \in [\![1\ ..\ \frac{P}{\alpha}]\!]$), with size γ (lines 8–10). The function *generateNewTEP* successively generates TEPs ($\boldsymbol{e}$), with Hamming weight HW, which are then ordered using the permutation vector $\boldsymbol{ind}$ (line 10). The generated TEPs ($\boldsymbol{e}$) are then applied sequentially to $\hat{\boldsymbol{y}}$ and the resulting vector ($\hat{\boldsymbol{y}} \oplus \boldsymbol{e}$) is then queried for codebook membership (line 11). If the codebook membership criterion (1) is satisfied, the original message ($\hat{\boldsymbol{u}}$) is retrieved and the decoding process is terminated (lines 12-13). Otherwise, γ, the size of the subset, is updated, and TEPs with Hamming weight $HW + 1$ are generated (lines 14-15).

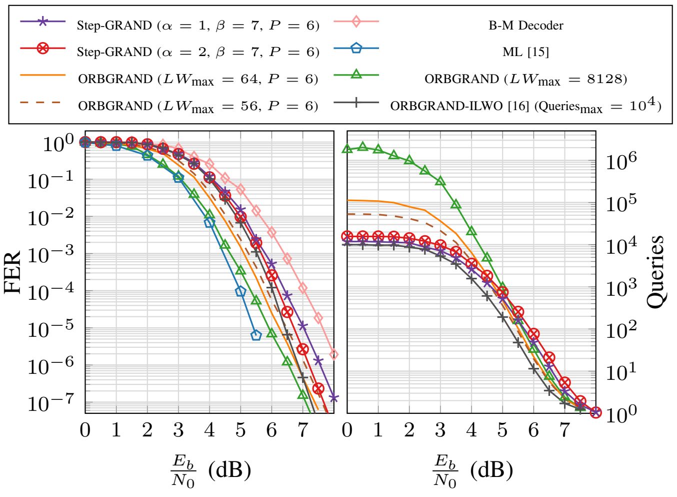
IV. Performance Evaluation

Figures 2 and 3 provide a comparison of the decoding performance as well as the computational complexity, expressed as the number of codebook membership queries required, of step-GRAND with other GRAND variants for the respective 5G NR CRC-Aided (CA)-polar code $(128, 105 + 11)$ and Bose-Chaudhuri-Hocquenghem (BCH) code $(127, 106)$ [17], [18]. Furthermore, the decoding performance of the Improved Logistics Weight Order (ILWO) ORBGRAND decoder [13],



(a) : FER (b) : Avg. Queries

Fig. 2. Comparison of decoding performance and average complexity of different GRAND variants for polar code $(128, 105+11)$.



(a) : FER (b) : Avg. Queries

Fig. 3. Comparison of decoding performance and average complexity of different GRAND variants for BCH $(127, 106)$ code.

CA-Successive Cancellation List (CA-SCL) decoder [19] and ML decoder [15] are presented for reference. Please note that, a BPSK modulation over an AWGN channel with variance σ^2 is considered for the numerical simulation results presented in this section. As demonstrated in Fig. 2 and 3, the proposed step-GRAND decoder outperforms the hard-input GRANDAB decoder [6] and the Berlekamp-Massey (B-M) [20][21] decoder. Furthermore, the decoding performance of step-GRAND approaches that of ORBGRAND [8] with improved channel conditions and various parametric settings.

The parameters maximum logistic weight ($LW_{\max}$) and P impact both the decoding performance and computational complexity of the baseline ORBGRAND [8][12]. However, the decoding performance and computational complexity of the proposed step-GRAND are influenced by the parameters α, β and P (Alg. 1). The worst-case complexity for the proposed step-GRAND is $\sum_{HW=1}^{P} \binom{\gamma}{HW}$, where γ is the size

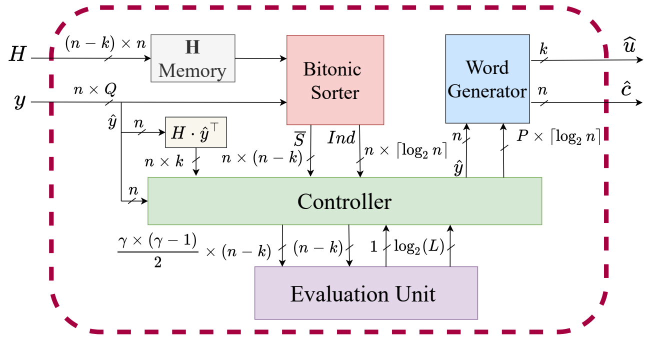


Fig. 4. Proposed VLSI architecture for step-GRAND

of the subset of $\hat{\boldsymbol{y}}$ for which TEPs with a Hamming weight HW ($HW \in [1, P]$) are evaluated.

At a target FER of $\leq 10^{-7}$, the proposed step-GRAND ($\alpha = 2$, $\beta = 6$, $P = 6$) achieves similar decoding performance to ORBGRAND ($LW_{\max} = 53$, $P = 6$), but experiences a degradation in decoding performance of 0.3 dB when compared to ORBGRAND ($LW_{\max} = 64$, $P = 6$), as shown in Fig. 2 (a). The Worst-Case (W.C) complexity for ORBGRAND ($LW_{\max} = 53$, $P = 6$) is 3.92×10^4 queries [22], whereas the W.C complexity for step-GRAND ($\alpha = 2$, $\beta = 6$, $P = 6$) is $8\,828$ queries[1]. Therefore, the W.C complexity of the proposed step-GRAND is $\frac{1}{4}\times$ that of ORBGRAND ($LW_{\max} = 53$, $P = 6$). Similarly, when decoding BCH code $(127, 106)$ at a target FER of $\leq 10^{-7}$, the step-GRAND ($\alpha = 2$, $\beta = 7$, $P = 6$) achieves similar decoding performance to ORBGRAND ($LW_{\max} = 56$, $P = 6$) but experiences a performance degradation of 0.2 dB when compared to ORBGRAND ($LW_{\max} = 64$, $P = 6$), as shown in Fig. 3 (a). However, step-GRAND ($\alpha = 2$, $\beta = 7$, $P = 6$) has a W.C complexity of $15\,778$ queries while ORBGRAND ($LW_{\max} = 56$, $P = 6$) has a W.C complexity of 5.37×10^4 queries [22]; as a result, the W.C complexity of step-GRAND ($\alpha = 2$, $\beta = 7$, $P = 6$) is $\frac{1}{3}\times$ that of ORBGRAND ($LW_{\max} = 56$, $P = 6$).

To summarize, the parameters of the proposed step-GRAND (α, β, P) can be adjusted for various classes of channel codes in order to achieve a balance between decoding performance requirements and the complexity/latency budget for a target application.

V. VLSI Architecture for step-GRAND

In this section, we present the VLSI architecture for step-GRAND, which is designed for universal decoding of (n, k) linear block codes. The proposed hardware architecture builds upon the previously proposed hard-input GRANDAB hardware [11] and soft-input ORBGRAND hardware [12], both of which use shift registers to store the $(n-k)$-bit syndromes associated with TEPs with a Hamming weight of 1 (denoted as $\boldsymbol{s}_i = \boldsymbol{H} \cdot \mathbb{1}_i^\top,\ i \in [\![1\,..\,n]\!]$). Furthermore, we leverage the linearity property of the underlying code to combine l TEP syndromes, corresponding to error patterns with Hamming

[1]With parameters ($\alpha = 2$, $\beta = 6$, $P = 6$), the (γ, HW) values are $(54,1)$,$(42,2)$,$(30,3)$,$(18,4)$,$(12,5)$ and $(6,6)$. (Sec. III)

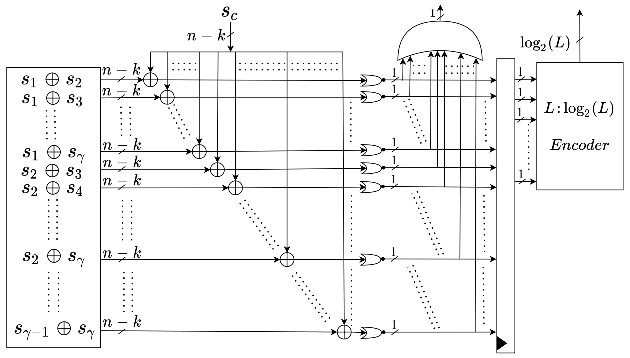


Fig. 5. Proposed *evaluation unit* for step-GRAND

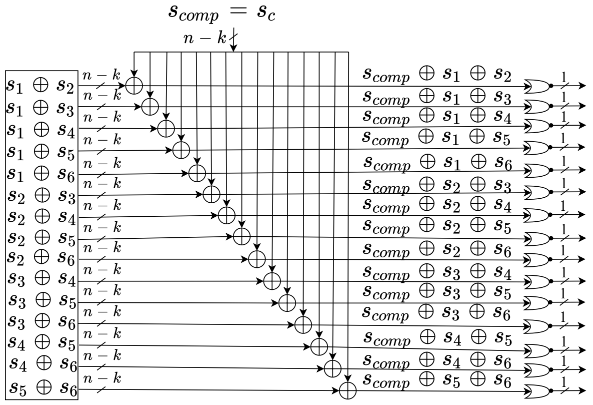


Fig. 6. Evaluating TEPs with a Hamming weight of 2 for $\gamma = 6$.

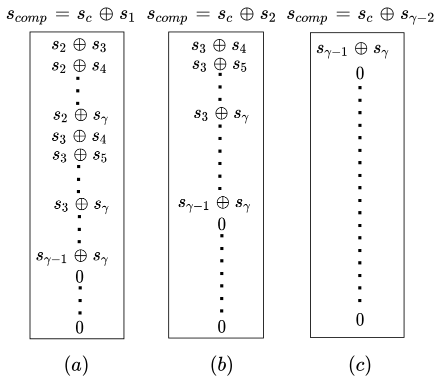


Fig. 7. Evaluating TEPs with a Hamming weight of 3

weight of 1 ($\boldsymbol{s}_i$), to generate syndromes corresponding to an error pattern with a Hamming weight of l ($\boldsymbol{s}_{1,2\dots,l} = \boldsymbol{H} \cdot \mathbb{1}_1^\top \oplus \boldsymbol{H} \cdot \mathbb{1}_2^\top \dots \oplus \boldsymbol{H} \cdot \mathbb{1}_l^\top$). We refer the reader to [11] and [12] for more details.

The proposed VLSI architecture for the step-GRAND, which can be used to decode any linear block code with a length of n and a coding rate of $0.75 \le R \le 1$, is shown in Fig. 4. Any parity check matrix (H) can be loaded into $(n-k) \times n$-bit H *memory* to support various classes of channel codes. The proposed step-GRAND hardware receives soft channel observations values (LLRs) $\boldsymbol{y}$ from the communication channel as an input and then applies the codebook membership verification (1) to the hard-decided vector $\hat{\boldsymbol{y}}$. The decoding is terminated if the codebook membership criterion

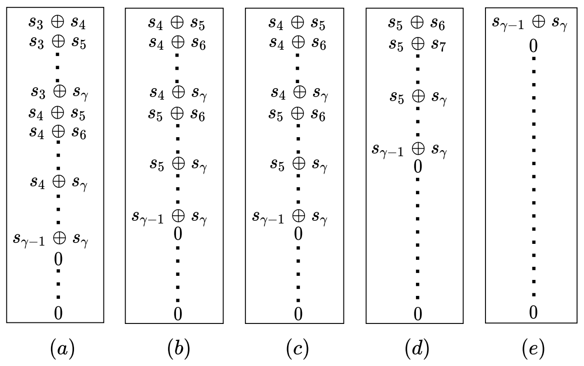


Fig. 8. Evaluating TEPs with a Hamming weight of 4 (a) $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_1} \oplus \boldsymbol{s_2}$ (b) $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_1} \oplus \boldsymbol{s_3}$ (c) $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_2} \oplus \boldsymbol{s_3}$ (d) $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_3} \oplus \boldsymbol{s_4}$ (e) $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s}_{\gamma-3}\boldsymbol{s}_{\gamma-2}$

(1) is satisfied for $\hat{\boldsymbol{y}}$ ($\boldsymbol{s_c} = \boldsymbol{H} \cdot \hat{\boldsymbol{y}}^\top = \boldsymbol{0}$). Otherwise, the *bitonic sorter* [23] is employed to sort the LLRs $(\boldsymbol{y})$ in ascending order of their absolute values ($|\boldsymbol{y}_i| \le |\boldsymbol{y}_j| \quad \forall i < j$). The bitonic sorter is pipelined to $\log_2(n)$ stages, and thus, sorting the received LLRs $(\boldsymbol{y})$ takes $\log_2(n)$ clock cycles. Following that, in a single time step, the codebook membership of all TEPs ($\boldsymbol{e} = \mathbb{1}_i$) with a Hamming weight of 1 is evaluated ($\boldsymbol{s_c} \oplus \boldsymbol{s}_i = \boldsymbol{0}, \forall\, i \in [1, \gamma]$).

The TEPs with Hamming weight $HW > 1$ ($\forall\ HW \in [2, P]$) are evaluated for codebook membership by the *controller* module in conjunction with the *evaluation unit*. If any of the evaluated TEPs ($\boldsymbol{e} = \mathbb{1}_{1,2\dots,HW}$) satisfy the codebook membership constraint ($\boldsymbol{s_c} \oplus \boldsymbol{s}_{1,2\dots,HW} = \boldsymbol{0}$), a 2D *priority encoder* module is used in conjunction with the controller module to pass the corresponding indices to the *word generator* module, which maps the sorted index values ($\boldsymbol{ind}$) to the appropriate bit flip locations in $\hat{\boldsymbol{c}}$.

A. Scheduling and Details

Figure 5 depicts the microarchitecture of the *evaluation unit* of the proposed step-GRAND, which employs a $\binom{\gamma}{2} \times (n-k)$-bit shift register to store $\binom{\gamma}{2}$ syndromes of TEPs with a Hamming weight of 2 ($\boldsymbol{s}_{i,j}, \forall\ i \in [\![1 \mathinner{..} \gamma-1]\!],\ j \in [\![i+1 \mathinner{..} \gamma]\!]$). The generated test syndromes ($\boldsymbol{s_c} \oplus \boldsymbol{s}_{i,j}$) are NOR reduced to evaluate all the $\binom{\gamma}{2}$ TEPs for codebook membership in parallel, as shown in Fig. 6 for $\gamma = 6$ and $HW = 2$. With the proposed evaluation unit, it only requires one time-step to evaluate, for codebook membership, all $\binom{\gamma}{2}$ TEPs with a Hamming weight of 2. The L-to-$\log_2 L$ priority encoder (shown in Fig. 5) is enabled to output the corresponding indices to the controller module, in $\frac{\gamma \times (\gamma-1)}{2 \times L}$ time-steps, if and only if the tested syndromes satisfy the codebook membership criteria ($\boldsymbol{s_c} \oplus \boldsymbol{s}_{i,j} = \boldsymbol{0}$).

To evaluate all TEPs corresponding to Hamming weight $3 \le HW \le P$, the controller module generates the composite syndrome $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_{1,2\dots,P-2}}$, which is combined with the syndromes stored in the shift register. Figure 7(a) displays the contents of the shift register and the composite syndrome $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_1}$ generated by the controller to evaluate the first set of $\frac{(\gamma-1)\times(\gamma-2)}{2}$ TEPs with a Hamming weight of 3. The shift register is shifted-up by $\gamma - 2$ at the next time step, and the controller generates $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_2}$ to evaluate the subsequent $\frac{(\gamma-2)\times(\gamma-3)}{2}$ TEPs as shown in Fig. 7(b). This procedure is repeated until the controller generates $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_{\gamma-2}}$ and the final TEP with Hamming weight of 3 ($\boldsymbol{s_c} \oplus \boldsymbol{s}_{\gamma-2} \oplus \boldsymbol{s}_{\gamma-1} \oplus \boldsymbol{s}_{\gamma}$) is evaluated, as illustrated in Fig. 7 (c). Therefore, evaluating all $\binom{\gamma}{3}$ TEPs with a Hamming weight of 3 requires $\binom{\gamma-2}{1}$ time steps, where the controller outputs $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_i}, \forall\, i \in [\![1 \mathinner{..} \gamma-2]\!]$ at each time step and the shift register is shifted up by $\gamma - i - 1, \forall\, i \in [\![1 \mathinner{..} \gamma-2]\!]$.

To evaluate the TEPs with a Hamming weight of 4, the controller generates $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_1} \oplus \boldsymbol{s_2}$ in the following time step. Figure 8 (a) shows the content of the shift register required to evaluate $\frac{(\gamma-2)\times(\gamma-3)}{2}$ TEPs with $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_1} \oplus \boldsymbol{s_2}$ generated by the controller. The shift register is shifted up by $\gamma - 3$ in the subsequent time step, as shown in Fig. 8 (b), and the controller generates $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_1} \oplus \boldsymbol{s_3}$ to evaluate the next $\frac{(\gamma-3)\times(\gamma-4)}{2}$ TEPs. In $\gamma - 3$ time steps, all TEPs with a Hamming weight of 4 and $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_1} \oplus \boldsymbol{s_i}, \forall\, i \in [\![2 \mathinner{..} \gamma-3]\!]$ are evaluated.

The controller generates $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_2} \oplus \boldsymbol{s_3}$ in the next time step, and the contents of the shift registers are shown in Fig. 8 (c). This configuration evaluates the set of $\frac{(\gamma-4)\times(\gamma-5)}{2}$ TEPs, with a Hamming weight of 4 and $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_2} \oplus \boldsymbol{s_3}$ output by the controller. The shift register is shifted up $\gamma - 4$ in the following time step and controller generates $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_2} \oplus \boldsymbol{s_3}$ to evaluate the next $\frac{(\gamma-5)\times(\gamma-6)}{2}$ TEPs as shown in Fig. 8 (d). This process is repeated, and after $\gamma - 4$ time-steps, all TEPs with $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s_2} \oplus \boldsymbol{s_i}, \forall\, i \in [\![3 \mathinner{..} \gamma-4]\!]$ are evaluated. This procedure continues, and the controller module generates $\boldsymbol{s_{comp}} = \boldsymbol{s_c} \oplus \boldsymbol{s}_{\gamma-3} \boldsymbol{s}_{\gamma-2}$ as shown in Fig. 8 (e) in order to evaluate the final TEP with a Hamming weight of 4. We can infer from the prior explanation that it takes $\binom{\gamma-2}{2}$ time steps to evaluate all $\binom{\gamma}{4}$ TEPs with a Hamming weight of 4.

The preceding discussion can be summarized as, by leveraging the proposed hardware, TEPs with Hamming weights $3 \leq HW \leq P$ can be evaluated in $\sum_{HW=3}^{P} \binom{\gamma-2}{HW-2}$ time steps. Hence, the worst-case latency of the proposed step-GRAND hardware is given by:

$$3 + \log_2(n) + \sum_{HW=3}^{P} \binom{\gamma-2}{HW-2}. \tag{2}$$

where $\log_2(n)$ is the latency of the sorter module.

B. Implementation Results

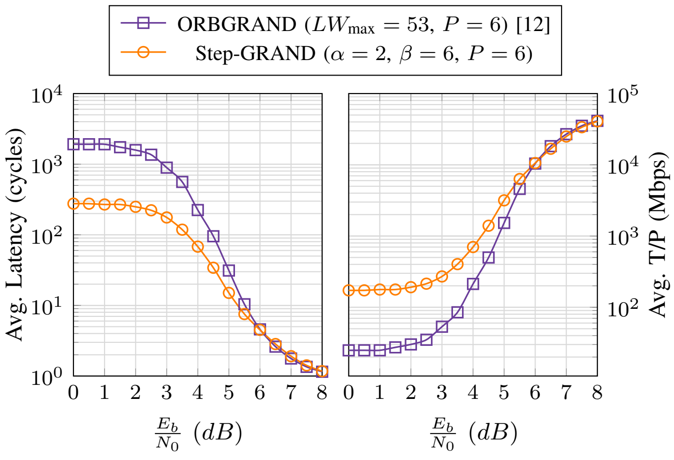
The proposed step-GRAND VLSI architecture, with parameters ($\alpha = 2$, $\beta = 6$, $P = 6$), has been implemented in Verilog HDL and synthesized with Synopsys Design Compiler using general-purpose TSMC 65 nm CMOS technology. Table I presents the synthesis results for the proposed step-GRAND implementation, with $n = 128$, and code-rate $0.75 \leq R \leq 1$, and compares them with state-of-the-art ORBGRAND hardware implementations[12][13][14]. The input channel LLRs ($\hat{\boldsymbol{y}}$) are quantized on 5 bits ($Q = 5$), with 1 sign bit and 3 bits for the fractional part. Please note that the step-GRAND VLSI architecture has been verified using test benches generated with the bit-true C model of the proposed hardware.

The proposed step-GRAND hardware can support a maximum frequency of 454 MHz, and since no pipelining is employed, one time step equals one clock cycle. In the worst-case (W.C.) scenario, the proposed step-GRAND hardware requires 279 cycles[2] (Eq. (2)), which translates to a W.C. latency of 614.5 ns and a W.C. throughput of 170.8 Mbps, as shown in Table I. Figure 9 illustrates the average latency as well as the average information throughput for the proposed hardware. Please note that the bit-true C model is used to compute the average latency, for the proposed hardware, after taking into account at least 100 frames in error for each $\frac{E_b}{N_0}$ point. As the channel conditions improve, the average latency

TABLE I
TSMC 65 NM CMOS SYNTHESIS COMPARISON RESULTS FOR STEP-GRAND.

	This Work	[12]	[13]	[14]
Parameters	$\alpha \leq 2$ $\beta \leq 6$ $P \leq 6$	LW≤64 $P \leq 6$	$Q_{\max} = 2^{13}$ $Q_{\text{LUT}} = 512$ $Q_{\text{S}} = 256$ $T = 34$	LW≤104 P≤13
Implementation		Synthesis		Fabricated
Technology (nm)	65	65	7	40
Supply (V)	0.9	0.9	0.5	1.0
Quantization (bits)	5	5	N.R	6
Code length (n)	128	128	128	256
Max. Frequency (MHz)	454	454	701	90
Area (mm^2)	1.18	1.82	3.70	0.4
W.C. Latency (ns)	614.5	4224.6[a]	58.49	N.R
Avg. Latency (ns)	2.2[b]	2.2[b]	58.49	40[c]
W.C. T/P (Mbps)	170.8[e]	24.85[e]	73610[e]	N.R[d]
Avg. T/P (Gbps)	47.7[e]	47.7[e]	73.61[e]	6.5[f]
Code compatible	Yes	Yes	Yes	Yes
Rate compatible	Yes	Yes	Yes	Yes

[a] Corresponding to parameters $LW = 53$ and $P = 6$ (Fig. 2 (a))
[b] For $\frac{E_b}{N_0} \geq 8$dB (Fig. 9), [c] At the target FER of 10^{-7}, [d]N.R: Not Reported
[e] For CA-Polar Code (128,105), [f] For CA-Polar Code (256,240).



(a) : Avg. Latency (b) : Avg. Info. Throughput

Fig. 9. Comparison of average latency and average information throughput for the ORBGRAND[12] VLSI architecture and the proposed step-GRAND architecture for Polar code (128,105+11).

[2]With parameters ($\alpha = 2$, $\beta = 6$, $P = 6$), the (γ, HW) values are $(30, 3)$,$(18, 4)$,$(12, 5)$ and $(6, 6)$. (Sec. III)

decreases until it only takes 1 cycle to decode a codeword, enabling up to 47.7 Gbps of information throughput.

As shown in Table I, the proposed step-GRAND hardware implementation requires 35% less area than the previously proposed ORBGRAND hardware [12]. While the average latency for both ORBGRAND and step-GRAND hardware can be as low as 1 clock cycle (Fig. 9), the W.C. latency for ORBGRAND hardware is 1 918 clock cycles [12][3], corresponding to parameters $LW = 53$ and $P = 6$ (Fig. 2 (a)), whereas step-GRAND requires only 279 clock cycles. Therefore, the W.C. latency of step-GRAND hardware $\frac{1}{6.8}\times$ the W.C. latency of ORBGRAND [12]. Furthermore, the step-GRAND is $10\times$ more area-efficient[4][5] than ORBGRAND hardware [12].

The proposed step-GRAND is also compared with Fixed-Latency (F.L.) ORBGRAND decoder [13] as well as with recent ORBGRAND hardware implementation [14] that employs a sequential sorter, and the comparison results are shown in Table I. Please note that scaling is not employed due to the vast disparity in technology nodes and implementation methods (Synthesis vs. Integrated Design). The F.L. ORBGRAND [13] deploys T decoder pipeline stages and stores the TEPs ($\boldsymbol{e}$) in $T - 2$ $Q_s \times n$-bit *pattern memories*, where Q_{max} denotes the maximum number of TEPs applied. The proposed step-GRAND reduces the average latency to 2.2 ns, whereas the F.L. ORBGRAND [13] offers a fixed latency of 58.49 ns for decoding polar code (128,105+11). The sequential sorter based ORBGRAND implementation [14] can achieve an average latency of 40 ns; the W.C. latency, however, is Not Reported (N.R). The proposed step-GRAND employs a parallel bitonic sorter and, owing to parallel evaluation of TEPs (discussed in section V), can achieve a W.C. latency of 614.5 ns while reducing the average latency to 2.2 ns. As a result, step-GRAND is suitable for mission-critical applications that have stringent requirements for both average and worst-case latency.

VI. Conclusion

In this work, we introduce step-GRAND, a soft-input variant of GRAND, along with corresponding hardware architecture. The proposed hardware can decode any linear block code with length $n = 128$ and code-rates between 0.75 and 1. Furthermore, the step-GRAND introduces parameters that can be tuned to match the desired decoding performance and complexity/latency budget of a target application. The ASIC synthesis results demonstrate that for a code length of 128 and a target FER of 10^{-7}, an average information throughput of 47.7 Gbps can be achieved. Furthermore, compared to the previously proposed ORBGRAND hardware implementation, the proposed step-GRAND hardware is $10\times$ more area efficient and its worst-case latency is $\frac{1}{6.8}\times$ that of the previous ORBGRAND hardware.

[3]Please refer to Fig. 12 of [12] for the W.C. latency of ORBGRAND hardware for various LW and P.

[4]Area Efficiency (Mbps/mm^2) $= \frac{\text{W.C. Throughput (Mbps)}}{\text{Area (mm}^2)}$

[5]The area efficiency for step-GRAND and ORBGRAND is 144.7 Mbps/mm^2 and 13.65 Mbps/mm^2, respectively.

References

[1] 3GPP, "Study on physical layer enhancements for NR ultrareliable and low latency case (URLLC)," http://www.3gpp.org/DynaReport/38-series.htm, Tech. Rep. TR 38.824, 2018, Rel. 16.

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

[3] 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 Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3098–3130, 2018.

[4] 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 of Things Journal*, vol. 6, no. 5, pp. 7946–7970, 2019.

[5] H. Chen, R. Abbas, P. Cheng, M. Shirvanimoghaddam, W. Hardjawana, W. Bao, Y. Li, and B. Vucetic, "Ultra-reliable low latency cellular networks: Use cases, challenges and approaches," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 119–125, 2018.

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

[7] K. R. Duffy, A. Solomon, K. M. Konwar, and M. Médard, "5G NR CA-Polar maximum likelihood decoding by GRAND," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2020, pp. 1–5.

[8] K. R. Duffy, W. An, and M. Médard, "Ordered reliability bits guessing random additive noise decoding," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4528–4542, 2022.

[9] A. Solomon, K. R. Duffy, and M. Médard, "Soft maximum likelihood decoding using GRAND," in *2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[10] 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 *IEEE 47th European Solid State Circuits Conference (ESSCIRC)*, 2021, pp. 239–246.

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

[12] 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 Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–13, 2022.

[13] C. Condo, "A Fixed Latency ORBGRAND Decoder Architecture With LUT-Aided Error-Pattern Scheduling," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 5, pp. 2203–2211, 2022.

[14] A. Riaz, A. Yasar, F. Ercan, W. An, J. Ngo, K. Galligan, M. Medard, K. R. Duffy, and R. T. Yazicigil, "A sub-0.8pJ/b 16.3Gbps/mm2 universal soft-detection decoder using ORBGRAND in 40nm CMOS," in *IEEE International Solid- State Circuits Conference (ISSCC)*, 2023, pp. 432–434.

[15] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn, "Database of Channel Codes and ML Simulation Results," www.uni-kl.de/channel-codes, 2019.

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

[17] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, 1959.

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

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

[20] E. Berlekamp, "Nonbinary BCH decoding (abstr.)," *IEEE Transactions on Information Theory*, vol. 14, no. 2, pp. 242–242, 1968.

[21] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, 1969.

[22] S. M. Abbas, M. Jalaleddine, and W. J. Gross, "List-grand: A practical way to achieve maximum likelihood decoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 1, pp. 43–54, 2023.

[23] K. E. Batcher, "Sorting networks and their applications," in *Spring Joint Computer Conference*. Association for Computing Machinery, 1968, p. 307–314.