Multi-Code Multi-Rate Universal Maximum Likelihood Decoder using GRAND

Arslan Riaz[1*], Vaibhav Bansal[1*], Amit Solomon[2+], Wei An[2+], Qijun Liu[1], Kevin Galligan[3], Ken R. Duffy[3], Muriel Medard[2], and Rabia Tugce Yazicigil[1]
[1]Department of Electrical and Computer Engineering, Boston University, Boston, MA, USA, [*]Equal Contribution
[2]Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA, [+]Equal Contribution
[3]Hamilton Institute, Maynooth University, Ireland

ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC) | 978-1-6654-3751-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ESSCIRC53450.2021.9567867

***Abstract*—We present the first fully-integrated universal Maximum Likelihood decoder in 40 nm CMOS using the Guessing Random Additive Noise Decoding (GRAND) algorithm for low-power applications. The 0.83 mm^2 multi-code multi-rate universal decoder can efficiently decode any code of length up to 128 bits with 1 µs latency at 68 MHz. Dynamic clock gating leveraging noise statistics reduces the average power dissipation to 3.75 mW at 1.1 V or 30.6 pJ/decoded bit with a throughput of 122.6 Mb/s. Universal decoding reduces hardware footprint, and the design allows seamless swapping between codebooks with no downtime, enabling use by multiple applications without switch-over.**

***Index Terms*—error correcting codes, maximum likelihood decoding, GRAND, hard-detection, universal decoder**

I. Introduction

All data is encoded prior to storage or transmission to protect it from potentially harmful corruption that can result in the received information differing from the original. By adding redundant bits, error detecting codes allow the data's integrity to be efficiently tested. Cyclic Redundancy Check (CRC) codes, for example, are ubiquitously deployed as error detection codes [1], [2]. Recent examples include Bluetooth Low Energy (BLE) packets, which have a 24-bit CRC appended, and all IEEE 802.15.4 IoT standard packets, which have a 16-bit CRC appended.

Error correcting codes allow errors to be rectified through the use of involved decoding algorithms. In some applications, such as data storage and optical communications, no additional information beyond the bits that are received is available to the decoder, which is called the hard-detection situation. In others, bits are received with a reliability measure that can be used to help guide error correction, which is called soft-detection.

Since the 1950s, by co-designing codes with particular structure and corresponding, distinct code-specific decoders, a wide variety of codes have been developed and deployed for error correction. Examples of codes with dedicated hard-detection decoders include Reed-Muller (RM) codes [3], [4] with Majority Logic Decoding, Reed-Solomon codes [5] and Berlekamp-Massey decoding [6], [7]. A recent example of an important code-structure that has a dedicated soft-detection decoder is CRC-Assisted Polar (CA-Polar) codes, which will be used for all control channel communications in 5G New Radio, with CRC-Assisted Successive Cancellation List (CA-SCL) decoding [8]–[10]. Efficient hardware realization for all of those decoders have been reported, e.g. [11]–[17], but, as the decoder is entirely coupled to the code structure, one needs a distinct implementation for each of them and often distinct pieces of hardware for different levels of redundancy. As a consequence, the standard co-design paradigm either leads to a proliferation of hardware to decode distinct codes or to restrictive code standardization to limit hardware footprint. As most codes only have a dedicated hard or soft-detection decoder, application design choices are further limited.

Guessing Random Additive Noise Decoding (GRAND) is a recently proposed class of universal decoding algorithms that challenges those limitations. Both hard and soft-detection variants [18]–[21] have been developed that, in theory and simulation, offer optimally precise Maximum Likelihood (ML) decoding for *any* moderate redundancy code. If GRAND algorithms can be translated into efficient hardware, they offer the possibility of decoding any code, regardless of whether it only has a hard or soft-detection decoder presently, in a single instantiation. Such a realization would significantly reduce existing hardware footprint while future-proofing a device so that it can decode any forthcoming code. Moreover GRAND would enable the automatic upgrade of omnipresent CRCs from error detection to accurate error correction [22].

Here we present the first fully-integrated universal hard-detection GRAND decoder shown in Fig. 1 for low-power, area-constrained, multi-code multi-rate applications. Our test chip demonstrates the multi-code and multi-rate efficient decoding up to 128 bits with a flexible rate R=0.66-1 that can correct up to 3-bit errors, at or above the guaranteed error-correction capability (GECC) of high-rate codebooks used in this work. In contrast to existing hardware that can only decode a handful of codes, the chip can accurately decode more than 2^{3696} distinct codes. Moreover, a single chip can support efficient, hard-detection decoding of any product code of up to 16,384 bits with rate R=0.43-1 and a higher GECC than its 128-bit component codes [23].

II. Guessing Random Additive Noise Decoding

Consider a binary codebook C_n which is a set of 2^k strings in $\{0,1\}^n$ where n is the code length, k is the number of information bits, and $R = k/n$ is the code rate. A transmitter sends a codeword X^n from the codebook C_n. Assume that a random noise effect N^n, which is independent of the channel input and also takes values in $\{0,1\}^n$, additively alters the

channel transmitted codeword X^n resulting in the received sequence:

$$Y^n = X^n + N^n. \quad (1)$$

Code-centric decoders attempt to identify X^n given Y^n by exploiting the structure of a specific codebook. GRAND's universality stems from the observation that if one can determine the effect of the noise, N^n in equation (1), one can deduce X^n, and doing so is more efficient for low or moderate redundancy codes [18]. To identify the noise effect N^n, GRAND generates a series of putative noise sequences E^n, subtracting them in turn from the received bits Y^n and checking if the resulting $Y^n - E^n$ is a member of the codebook. For a linear codebook, this can be achieved by calculating the product of $Y^n - E^n$ with the parity check matrix H for the given codebook. If the product is zero, the calculated sequence belongs to the codebook, and the membership check is satisfied. The first E^n that passes the membership check for $Y^n - E^n$ is used to identify X^n. If the transmitted codewords are all equally likely and the noise sequence E^n is generated from most likely to least likely according to the channel statistics, then the first resultant sequence $Y^n - E^n$ found in the codebook is an optimally accurate ML decoded output.

For a hard-detection binary symmetric channel (BSC), GRAND generates the sequence of noise patterns according to the Hamming weight from low to high, i.e. $i = 0, 1, 2, 3, ..., n$ -bit errors. With one query for each noise sequence, the total number of queries required for a Hamming weight of i is $\binom{n}{i}$. It has been mathematically established that we do not need to query all the possible noise patterns for the GRAND decoding to be capacity achieving [18]. An upper bound or abandonment threshold can be set on the number of queries to be made after which the decoder reports failure to decode and either requests a re-transmission or drops the packet. We choose this abandonment threshold as 3 for the hard-detection GRAND decoder presented in this work. This threshold matches well with a realistic channel condition with a Bit Error Rate (BER) of 10^{-3} where the probability of having 4 or more bit errors is as low as 9.66×10^{-6}. This threshold also suits the guaranteed error-correction capability (GECC) of high-rate codebooks used in this work.

III. Universal Maximum Likelihood Decoder Architecture

Our universal decoder efficiently creates, on the fly, putative error sequences in decreasing order of likelihood and then forms the difference between the demodulated channel output, Y, and the next most likely noise effect sequence, E. Sequentially, the resultant bit string is checked for codebook membership by querying the codebook. The first member codeword (CW) found by successful noise guessing corresponds to the ML decoding. Conventional Syndrome Decoding uses large and onerous pre-computed code-specific syndrome look-up tables (LUTs), while GRAND computes only the minimum number of needed syndromes per decoding without requiring any LUT. Our decoder in Fig. 2 consists of a

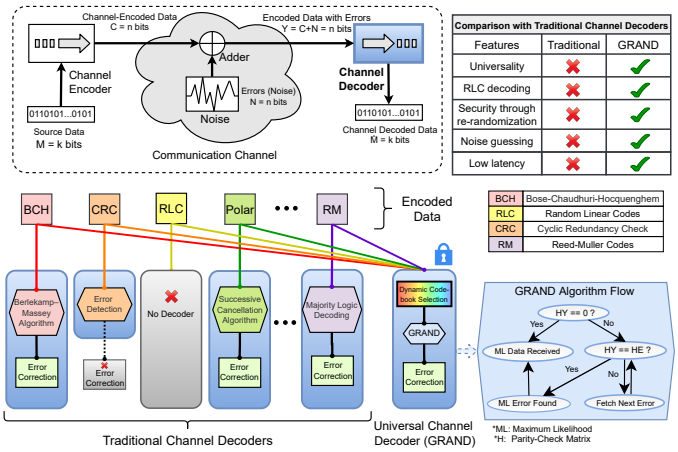
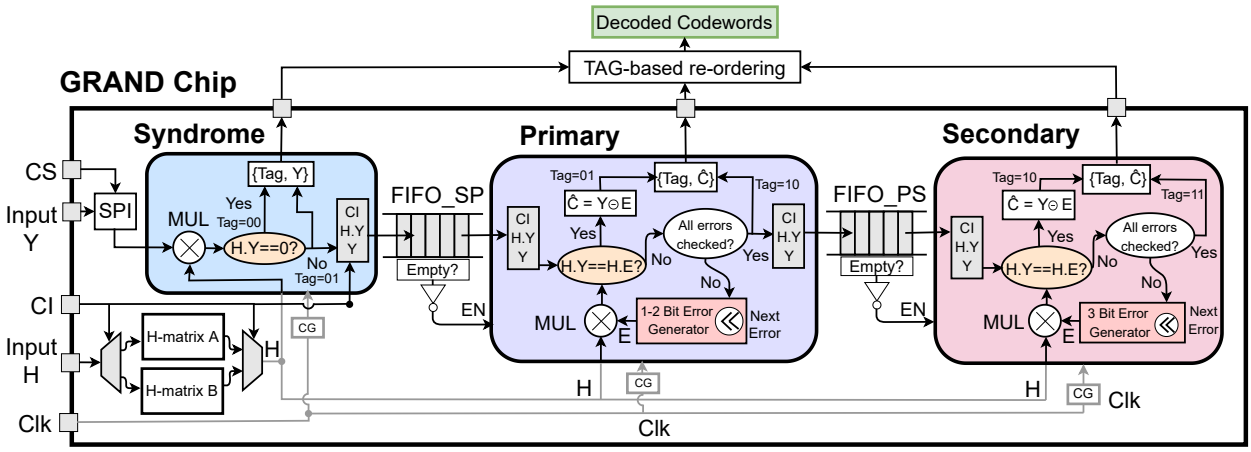


Fig. 1. Comparison of traditional code/rate-specific decoding with the universal noise-centric GRAND for low-power, low-latency, and area-constrained multi-code multi-rate wireless applications.

Syndrome calculator for checking codebook membership and an Error Generator (EG) for calculating rank-ordered binary symmetric channel noise patterns with 1 to 3-bit flips and checking for membership by querying the codebook. Since all the codewords have to go through the Syndrome computation first, the Syndrome block and EG are pipelined to avoid stalling the syndrome computation of the following codewords without any errors when the EG is searching through the noise sequence. The parity check matrix H is loaded into the chip for the corresponding codebook in a dual-port SRAM. The dimensions of H are $(n-k) \times n$. The chip supports $k \in [1, 128]$ and $n \in [k+1, \min(k+44, 128)]$. The channel output Y is a 128-bit vector that is used in the Syndrome block for checking the membership of the codebook by computing HY. The Syndrome block takes 64 cycles for the computation of HY. The HY matrix-vector multiplication is zero if and only if the received codeword is a member of the codebook. If HY is not equal to zero, the Syndrome block passes HY and Y to the EG. These EGs calculate rank-ordered BSC noise patterns (E_i) with 1 to 3-bit flips, and check for membership by querying the codebook.

Considering a code length of 128 bits with a single query per cycle, checking for all the 3-bit noise sequences will take 341,376 cycles which is 41× higher than the number of queries required for combined 1- and 2- bit noise pattern search. Therefore, the EG is further split into Primary and Secondary EGs. The Primary EG checks two error patterns in parallel for the 1- and 2- bit errors achieving 2× reduction in latency while the Secondary EG checks 16 error patterns in parallel for the 3-bit errors achieving 16× reduction in latency. Further, we implemented a three-stage pipelining for the Syndrome, Primary and Secondary EGs to increase the throughput by 1.7× for a channel BER of 10^{-3}.

The three-stage pipelined decoder architecture allows us to leverage the noise statistics to optimize and reduce the energy per decoded bit. To achieve this, we use dynamic clock gating



Universal Noise-centric Decoder Signals					
Y	Demodulated channel output	CS	Chip select signal	Tag	Two bit tag for re-ordering
H	Parity-check matrix for syndrome calculation	CG	Clock Gating	00	Tag showing no error in the codeword
E	Candidate error vector (Guessed noise sequence)	EN	Enable signal to the Error Generator	01	Tag showing 1 or 2 bit error
CI	Tag for code-interleaved architecture	H.E	Product of H-matrix and Error vector E	10	Tag showing 3 bit error
H.Y	Product of H-matrix and Y	Ĉ	Decoded codeword	11	Tag showing 4 or more bit error

Fig. 2. GRAND decoder system architecture and flow diagram.

of the EGs based on the probability of error likelihood. For example, for a code length n of 128 bits and BER of 10^{-3}, 87.98% of the codewords have no errors, 11.99% of them have 1- or 2-bit flips, and, only 0.03% of them have 3-bit flips. The Secondary EG remains active only for 0.03%, and the Primary is active for 11.99% of the decoding, reducing the power and enabling $8.6\times$ energy savings compared to the performance when EGs are continuously on.

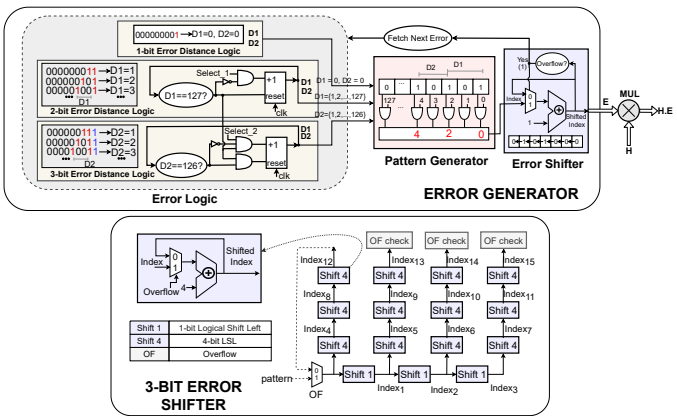


Fig. 3. Key blocks of the GRAND chip: Noise pattern (Error) Generator (top) and 3-bit Error Shifter (bottom).

A. Error Generator

The EG efficiently creates ordered noise patterns, i.e., error sequences, in parallel. It consists of three modules: the error logic, pattern generator, and error shifter as shown in Fig. 3. The error logic sequentially generates a distance pair (D1, D2) to indicate the distances between the active high bits (1s) in the guessed noise sequence (E_i). For example, the error vector with a size of 128×1, E_i = (00...0010011) has a distance logic pair of (D1=1, D2=3). The pattern generator constructs the bit sequence defined by (D1, D2), which is then used as the indices of the input-seed 1s for the error shifter. The error shifter creates in parallel 2 and 16 variations of the seed sequence through cyclical shifts for the Primary and Secondary EGs until there is an overflow. This overflow condition indicates the completion of error generation for the given distance pair. We check this condition to increment the distance D1 and then D2 to generate the next set of error sequences. For the 3-bit Error Shifter, the error vectors are generated in four parallel branches to reduce critical path delay via a combination of 1- and 4-bit logical-shift left. Error vectors are fed into the matrix-vector multiplication unit to calculate and check if HY, equals HE_i.

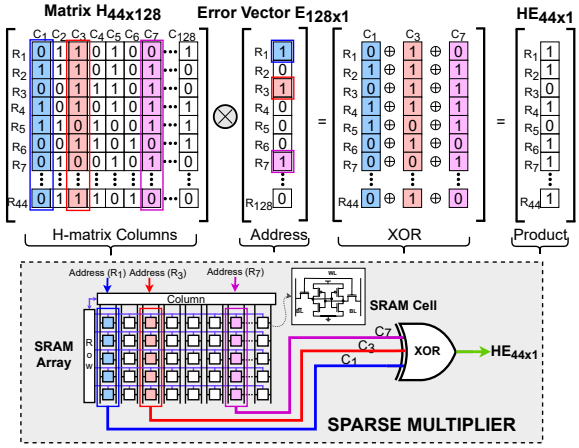


Fig. 4. Low-latency matrix-vector multiplication enabled by the sparsity of errors.

B. Sparse Low-Latency Multiplier

GRAND has heavy matrix-vector multiplication dependencies to check if the syndrome of the decoding HY equals HE_i. Traditional matrix-vector multiplication unit takes $(n-k)\times n$ multiply-add operations for an H parity-check matrix of size $(n-k)\times n$ and E_i of size $n\times 1$. However HE_i needs to be computed in one cycle to keep generating error sequences without stalling.

To achieve low-latency multiplication in a single cycle, we leverage the sparsity of error vectors (E_i) with very few nonzero entries. For instance, to achieve a BER of 10^{-3} for $n=128$ bits, the BSC noise statistics dictate the probability of having three-bit flips for only 0.03% of the codewords. Since the probability of receiving codewords with more than three-bit flips is negligible (0.00096%), we only check possible noise patterns up to three-bit flips, e.g., $e\in[0,3]$, and then decide to abandon. These bit flips are randomly located in the n-length codeword and, as $e << n$, the error vector E_i with a size of $n\times 1$ is sparse. Fig. 4 conceptually illustrates using the sparsity of the noise pattern in an SRAM-based multiplication unit for a codeword length of $n=128$ bits with a flexible rate support ($R\geq 0.66$). This technique uses active high bits of the error vector E_i as the column address to select the corresponding columns of the H matrix stored in a dual-port SRAM. These columns can then be XORed for the final product, HE_i.

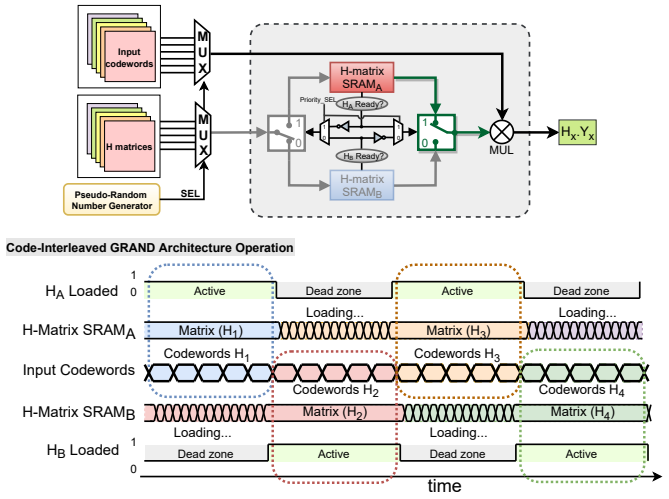


Fig. 5. Code-interleaved GRAND architecture to dynamically re-randomize codebooks without dead zones.

C. Code-Interleaved GRAND Architecture

A core feature of a hardware implementation of GRAND is its ability to decode all distinct codes, limited only by length and maximum redundancy, in a single chip. Thus we can use our chip to support multiple codes, say for different applications. To switch codes on the fly, avoiding expensive coordination for switch-over, which would lead to decoding downtime, we implemented a Code-Interleaved (CI) architecture shown in Fig. 5. This code interleaving architecture enables seamless loading of a new H matrix and simultaneous decoding of the received codeword using the previously loaded H matrix. We use a single bit CI tag to indicate which of the two SRAMs contains the corresponding H matrix for a received codeword.

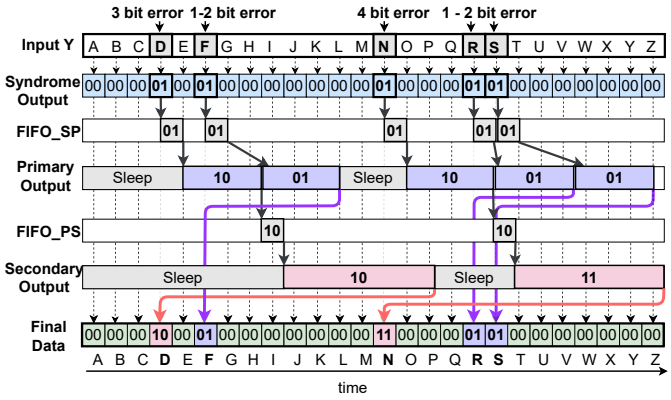


Fig. 6. Tag-based reordering of the decoded codeword for the parallelized and pipelined operation.

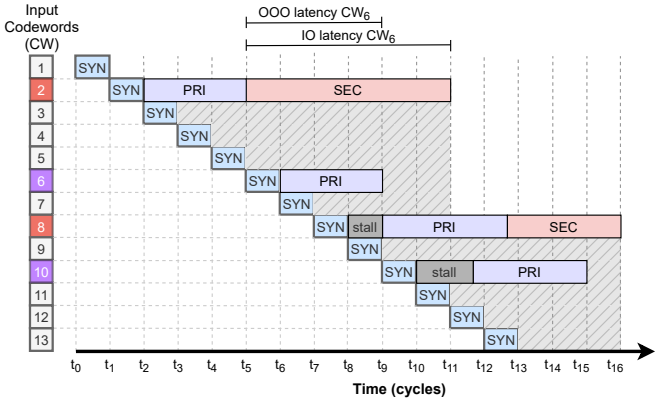


Fig. 7. In-order and Out-of-order latency for the three stage pipelined architecture.

D. Tag-Based Re-Ordering

The GRAND architecture can process multiple received codewords simultaneously and output the decoded results at different time instances due to its pipelined operation. Stage one of the three-stage pipeline operation is the Syndrome calculator which performs the HY syndrome computation and codebook membership check. This block takes 71 cycles in the decoding process. In the event of detecting a bit error in the decoding process, the second stage, the Primary EG, is used for 1- and 2-bit error detection and correction. A single cycle is required in the event of a 1-bit error in the most significant bit of a codeword Y, otherwise a maximum of 4167 cycles is required for a worst-case 2-bit error pattern. The last block in the pipeline, the Secondary EG, is used for detecting and correcting three-bit errors, and requires between one and at most 25,262 cycles, depending on the positions of the three bit flips within the codeword Y.

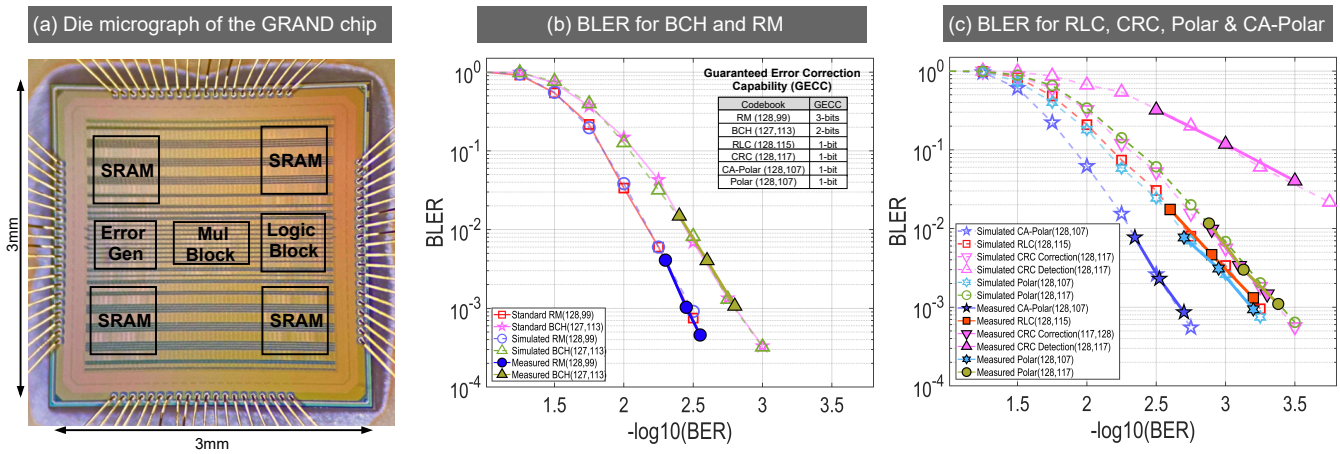


Fig. 8. (a) Die micrograph of the universal GRAND decoder implemented in 40 nm CMOS. Universality of chip via measured BLER vs. BER (b) for BCH and RM and comparison with the standard decoding, (c) for CA-Polar, RLC, and CRC for both error detection and correction.

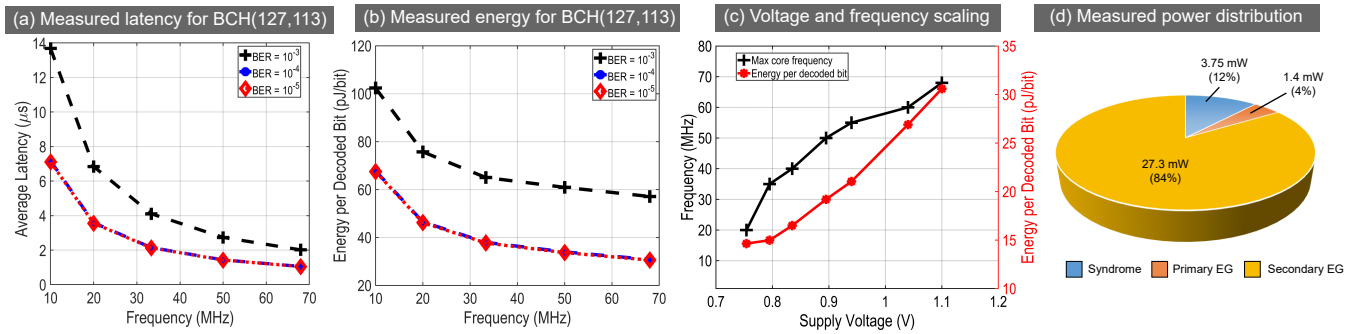


Fig. 9. (a) Measured latency vs. frequency; (b) Measured energy per decoded bit using a 1.1V supply voltage for a BER of 10^{-3} to 10^{-5} for BCH (127,113); (c) Scaling of maximum core frequency and minimum supply voltage for a BER of 10^{-5}; (d) Measured power distribution of individual blocks including the leakage power from a 1.1V supply voltage at 68 MHz of clock frequency.

Given these varying processing times within the Syndrome, Primary EG, and Secondary EG, the list of decoded results may be out of order. To re-establish the given input ordering of the received codewords, we use a simple two-bit tag strategy for successful information recovery as shown in Fig. 6. The tag value is initialized to 00 and is incremented in the event that a block in the pipeline process unsuccessfully recovers the received codeword. Thus, a tag value of 00 indicates 0-bit flips in the received codeword, 01 indicates 1- or 2- bit flips, and 10 indicates 3 bit flips, all indicating successful decoding with codeword re-ordering. Lastly, a tag value of 11 indicates decoding failure.

As the pipeline process may result in out-of-order decoded codewords, we use two latency metrics to analyze the system performance for this architecture. First, we define in-order (IO) latency as the processing time of a codeword such that the processed output list corresponds to the received input list ordering. Thus, the IO latency of a given codeword has a dependency on the processing time of the immediately preceding codeword. Second, we define the out-of-order (OOO) latency as the minimum time required for a given codeword to be processed, independently of the codeword list order. Thus, the OOO latency will be less than or equal to the IO latency, as illustrated in Fig. 7 by the sixth codeword.

IV. Measurement Results

The GRAND chip shown in Fig. 8(a) is implemented in 40 nm CMOS technology occupying 0.83 mm^2 of active core area. The chip supports all binary linear codes of length up to 128 bits and rates from 0.66 to 1. The performance of the GRAND chip is measured for different codebooks and multiple rates at frequencies ranging from 10 MHz to 68 MHz at a supply voltage of 1.1 V. Further, the chip supports voltage scaling from 1.1 V down to 0.75 V.

A. Block Error Rate (BLER) Performance

The BLER performance of the GRAND decoder is measured for different codebooks with different rates to demonstrate the universality of the chip. Figure 8(b) shows the BLER performance of the GRAND chip for Reed Muller (RM) and Bose-Chaudhuri-Hocquenghem (BCH) with a code rate of 0.77 and 0.89, and a code length of 128 and 127 bits, respectively. The BLER performance of this decoder is compared with the standard decoders for RM [4] and BCH [24] which show that GRAND has competitive BLER performance. The BLER performance of our GRAND chip is also measured

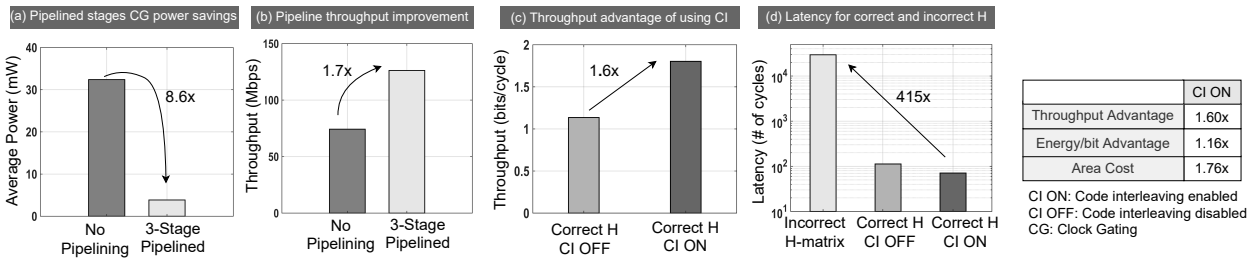


Fig. 10. (a) Average power savings due to dynamic clock gating of pipelined stages according to channel noise statistics; (b) Throughput gain using the three-stage pipelined architecture; (c) Throughput gain with CI ON versus CI OFF; and (d) Latency loss when incorrect H matrix is used.

using Random Linear Code (RLC), Cyclic Redundancy Check (CRC), Polar and CRC-Aided Polar (CA-Polar) with code length of 128 bits and a code rate of 0.90, 0.91, 0.84, and 0.84, respectively. In Fig. 8(c), we demonstrate: the first ML decoding of RLCs; the first hard detection ML decoding of CA-Polar codes; and the first universal error correction of up to 3 bit flips for CRC codebooks. It has been already shown that soft information decoding performance of Polar codes is poor as compared to CA-Polar codes [9]. Here we find the same effect in hard detection decoding, that the BLER of Polar codes is significantly higher than the CA-Polar codes. For a fixed BLER of 10^{-3}, there is a performance improvement of approximately 1 dB in E_b/N_0 for CA-Polar (128,107) as compared to Polar (128,107). Similarly, CRC performs slightly better than the Polar code of the same length and rate.

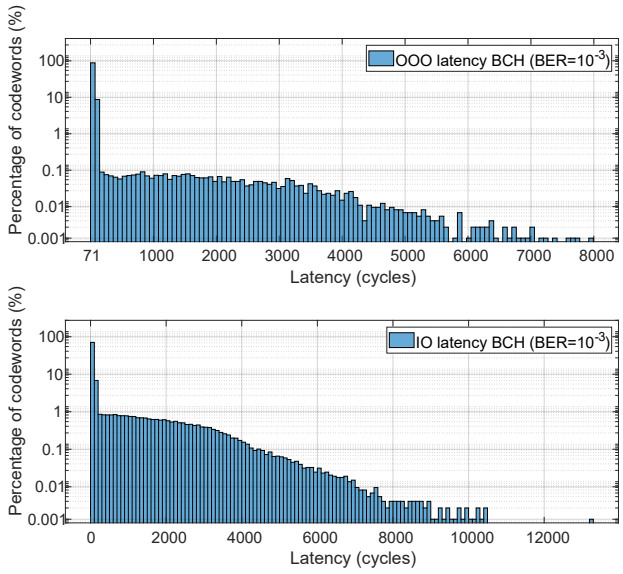


Fig. 11. Measured in-order and out-of-order latency for a BCH code with k=113, n=127 at BER of 10^{-3}. The histogram shows the percentage of codewords vs. measured latency performance.

B. Energy, Latency, and Power Consumption

The decoder achieves 1.04 μs average latency with a throughput of 122.6 Mb/s and consumes 30.6 pJ/decoded bit from a 1.1V supply for a BER of 10^{-5} as shown in Fig. 9(a) and (b). For the same BER, the measured average power

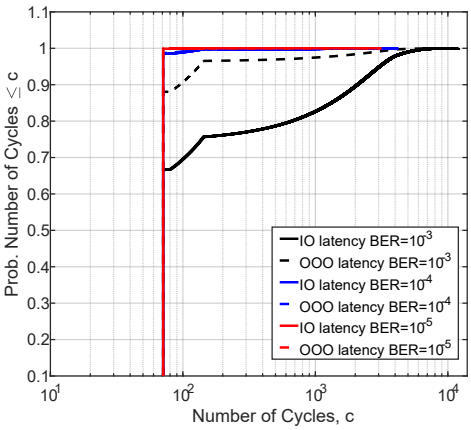


Fig. 12. Measured in-order and out-of-order latency in cycles vs. probability that the number of cycles for a given codeword is less than c.

consumption is 3.75 mW from a 1.1 V supply at 68 MHz of clock frequency. Figure 9(c) illustrates the performance of the GRAND decoder as a function of the supply voltage, demonstrating a maximum frequency of 68 MHz at 1.1 V and 20 MHz at 0.75 V. The distribution of the decoder power consumption including the leakage power is shown in Fig. 9(d) for its nominal operating conditions at 68 MHz and 1.1 V supply. As shown in Fig. 10(a), by leveraging dynamic clock gating of the pipelined stages based on channel noise statistics, the average power and energy consumption of the system can be significantly lowered since the secondary EG, which operates at higher power than the primary one, is only active for $3.4 \times 10^{-8}\%$ of the decoding process for a BER of 10^{-5}. The three pipelined stages also provide improvement in the throughput, which is $1.7\times$ higher than a system with no pipelining for a BER of 10^{-3} as shown in Fig. 10(b).

C. Code Interleaving

We assessed the CI architecture by dynamically re-randomizing RLC codebooks. When the CI is enabled, both SRAMs are used in a time-interleaved fashion to increase the throughput by $1.6\times$, while consuming 14% lower energy per decoded bit in contrast to when the CI is disabled as shown by the measurement results in Fig. 10(c). However, this code

interleaving feature comes at the expense of an increased area by a factor of 1.76.

The latency measurement results in Fig. 10(d) explore the performance degradation when an incorrect H matrix is used for RLC decoding. Using the incorrect matrix for decoding leads to severe failures to decode. As a corollary, a non-authentic decoder makes many guesses, yet is unsuccessful. For a non-authentic decoder, these unsuccessful guesses degrade latency by a factor of 415, while increasing energy per bit by a factor of 2920. Thus non-authentic decoder expends energy and time, while still producing incorrect decodings. This feature points to the possible use of RLCs to provide an additional security layer via dynamic randomization of the codebook, as any non-authentic decoder will encounter performance degradation in latency and energy consumption along with severe failures to decode.

D. In-Order and Out-Of-Order Latency Performance

The in-order (IO) and out-of-order (OOO) latency is measured using a BCH (127,113) code of rate 0.89. We demonstrated the IO and OOO latency measurements as the histogram plots in Fig. 11 for the BER of 10^{-3}. As shown in Fig. 11, for a noisy channel, the IO latency is higher for a higher percentage of codewords. The IO and OOO latency is also measured across different channel conditions, BER, and shown as empirical CDFs in Fig. 12. This measurement demonstrates that as the channel condition improves, the IO latency approaches OOO latency and the probability that any given codeword will take only 71 cycles to decode approaches 1. This distinctive feature of GRAND's noise-centric approach means that as the channel improves, the decoder's latency performance automatically improves. As a result, the GRAND decoder immediately and inherently gets the benefit of improved channel conditions without the need for signalled adaptation while providing the flexibility of switching to a new rate or a new codebook in real-time due to the CI architecture.

V. Performance Summary and Comparison

GRAND is the first universal decoding algorithm to be implemented successfully in hardware, thus uniquely enables this architecture to be compared with any hard decoder. The performance comparison of our GRAND chip is shown in Fig. 13 versus other fully integrated and post-layout code-specific BCH hard decoders [13]–[15], and to a synthesized-only alternative GRAND decoder [25]. Comparing [13]–[15], our decoder has 1.57× lower, 4.37× higher and 1.02× lower area, respectively. Given our singular architecture and footprint, our chip can successfully decode BCH and any other binary linear code, contrasting code-specific decoders. Further, our GRAND chip consumes 1.68× lower power and 2.08× lower energy per bit relative to [14], and 4.42× lower power but 6.22× higher energy per bit as compared to [15] under different designed operating frequency conditions. An alternative GRAND architecture proposed in [25] is targeted towards high throughput data storage systems achieving one cycle best-case latency at 500 MHz frequency. Only synthesized results have been reported without considering the performance changes after place and route. The power consumption of this design has been estimated by parallel register-file computation of syndrome product in a single cycle at 500 MHz frequency. Our GRAND decoder targeting low-power applications demonstrates 200× lower measured power under latency trade-off compared to [25]. Unlike [13]–[15], [25], our decoder supports multiple communication standards simultaneously with no dead zones for instantaneous switching, providing seamless communication while reducing the footprint and the energy costs associated with using separate code-specific decoders.

VI. Conclusions

We present the first integrated code-agnostic decoder implemented in 40 nm CMOS technology that supports any code family with a code length up to 128 bits and a flexible code rate (0.66-1). The three staged pipelined architecture of the decoder enables dynamic clock gating of stages according to channel noise characteristics resulting in 8.6× energy savings and 1.7× higher throughput. The 0.83 mm^2 GRAND chip is measured using BCH, RM, CRC, RLC, Polar, and CA-Polar codebooks with rates of 0.89, 0.77, 0.91, 0.90, 0.91, and 0.84, respectively, and provides a good error correction performance. The chip operates at 1.1 V supply voltage consuming 3.75 mW of power and 30.6 pJ/bit of energy at a frequency of 68 MHz providing 1.04 μs of delay at BER of 10^{-5} with a throughput of 122.6 Mb/s. This universal ML decoder also enables switching codebooks in real time without dead zones using CI architecture to provide simultaneous decoding of multiple communication standards.

While the results presented here demonstrate GRAND's hardware performance on short, high-rate codes, such as CRCs, it can also be used for precise, low-latency universal decoding of product codes, which are a class of long, powerful codes that are used in storage and optical network applications. Product codes are constructed by concatenating short components codes and have a higher error correction capability than the associated component codes [26]. They inherently interleave, providing additional protection against bursty noise. A single GRAND chip can efficiently decode 16,384-bit product code having a code rate between 0.43 and 1, composed of 128-bit component codes, with 56 pJ/b of energy consumption. By using 16 GRAND chips in parallel, at no extra cost of energy per bit, 8.4 μs delay with 1.95 Gb/s throughput can be achieved while consuming 60 mW of power. The number of parallel decoder branches can be chosen according to the application requirements and the system can adapt to have a better performance by turning parallel branches on or off if the requirement changes. This makes it particularly suitable for low latency and high throughput hard-detection applications.

VII. Acknowledgments

This work was partially supported by the Battelle grant Low-Probability-of-Detect/Intercept Communications Employing Peaky Frequency-Shift-Key Modulation (PO

References	**This Work**	ISSCC '06 [13]	JSSC '10 [14]	TVLSI Systems '14 [15]	SiPS '20 [25]
Technology (nm)	40	90	90	130	65
Implementation	Integrated Design	Integrated Design	Post Layout	Integrated Design	**Synthesized Only**
Voltage (V)	1.10	N.R.	1.00	1.20	0.90
Decoding Algorithm	Universal Noise-Centric GRAND	Berlekamp-Massey (BM) + Chien Search	Inversion-less Berlekamp-Massey(iBM)+Chien	Overlapped Berlekamp-Massey (oBM)	Universal Noise-Centric GRAND
Supported Code Families	BCH, RM, CRC, RLC, Polar, CA-Polar (Universal)	BCH	BCH	BCH	CRC (Universal)[a]
Parallelism	16	8	N.A.	32	128
Pipelining	Three-Stage Pipeline	N.R.	Single-Stage Pipeline	N.R.	No Pipelining
Code Length (n)	127, 128[b]	16460[c], 32767[d]	32400	8640	128
Code Rate (R)	0.656 - 1.000[e]	0.997[d]	0.994[f]	0.948[g]	0.75
Average power (mW)	3.75[h]	N.R.	6.32	16.6	751[k]
Energy per Decoded Bit (pJ/bit)	30.6[h]	N.R.	63.60	4.92	N.R
Throughput (Mbps)	122.6[h]	N.R.	99.3	6400	9000
Frequency (MHz)	10-68	25	200	200	500
Average Latency (µs)	1.04[i]	34.6	324.12[j]	2.565	N.R
Latency (# of cycles)	Best case: 71 Average case: 71.1[i] Worst case: 29500	Best case: 150 Average case: 865 Worst case: 6250	Best case: N.R. Average case: N.R. Worst case: 64824	Best case: N.R. Average case: 513 Worst case: N.R.	Best case: 1.0 Average case: N.R Worst case: 4098
Security	Yes (Rapidly-Changing RLC Decoding)	No	No	No	No
Area (mm^2)	0.83[l]	1.30[m]	0.19[n]	0.85[o]	0.25[p]

N.R. = Not reported; N.A. = Not applicable.

[a] Universal GRAND synthesized results shown only for CRC.
[b] BCH codeword length of 127 bits and RLC codeword length of 128 bits are used for the measurements.
[c] Latency (# of cycles) for [13] is reported for a code length of 16460 in [15].
[d] Code rate is reported for a code length of 32767 in [13]. Fixed rate.
[e] Measured energy, latency, and throughput is reported for R = 0.88.
[f] Rates available: 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, and 9/10.
[g] Fixed rate.
[h] Measured performance at BER = 10^{-5} and 68MHz for a BCH code.
[i] Measured in-order latency at BER = 10^{-5} and 68MHz for a BCH code.
[j] Only worst-case latency is reported in [14].
[k] Estimated for parallel register file computation of syndrome in one cycle at 500 MHz.
[l] Active area for code-interleaving (CI) OFF GRAND
[m] ECC area overhead. Chip area is 140mm^2
[n] Core area.
[o] BCH decoder area.
[p] Reported area estimate from synthesized design.

Fig. 13. Performance comparison with prior hard decoders, note that code/rate-specific decoders' total area increases proportionally with the number of code-rate pairs supported, unlike our multi-code multi-rate decoder.

US0011-0000743557). This publication has emanated from research supported in part by a grant from 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. We thank Alex Ji and Utsav Banerjee for tool-related discussions.

REFERENCES

[1] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *IEEE/IFIP DSN-C*, 2004, pp. 145–154.
[2] P. Koopman, K. Driscoll, and B. Hall, "Selection of cyclic redundancy code and checksum algorithms to ensure critical data integrity," *DOT/FAA/TC-14/49*, 2015.
[3] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IEEE Trans. Inf. Theory*, vol. 4, no. 4, pp. 38–49, 1954.
[4] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection,," *Trans. I.R.E. Prof. Group Elec. Comp.*, vol. 3, no. 3, pp. 6–12, 1954.
[5] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960.
[6] E. Berlekamp, *Algebraic Coding Theory*. World Scientific, 1968.
[7] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf Theory*, vol. 15, no. 1, pp. 122–127, 1969.
[8] K. Niu and K. Chen, "CRC-aided decoding of Polar codes," *IEEE Commun. Letters*, vol. 16, no. 10, pp. 1668–1671, 2012.
[9] I. Tal and A. Vardy, "List decoding of Polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.
[10] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of Polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, 2015.
[11] S. Clerc, F. Abouzeid, V. Heinrich, A. Jain, A. M. Veggetti, D. Crippa, P. Roche, and G. Sicard, "A 40nm CMOS, 1.27nJ, 330mV, 600kHz, Bose Chaudhuri Hocquenghem 252 bits frame decoder," in *IEEE ICICDT*, 2010, pp. 78–81.
[12] S. Scholl and N. Wehn, "Hardware implementation of a Reed-Solomon soft decoder based on information set decoding," in *DATE*, 2014.
[13] R. Micheloni, R. Ravasio, A. Marelli, E. Alice, V. Altieri, A. Bovino, L. Crippa, E. Di Martino, L. D'Onofrio, A. Gambardella, E. Grillea, G. Guerra, D. Kim, C. Missiroli, I. Motta, A. Prisco, G. Ragone, M. Romano, M. Sangalli, P. Sauro, M. Scotti, and S. Won, "A 4Gb 2b/cell NAND flash memory with embedded 5b BCH ECC for 36MB/s system read throughput," in *IEEE ISSCC*, 2006, pp. 497–506.
[14] Y.-M. Lin, C.-L. Chen, H.-C. Chang, and C.-Y. Lee, "A 26.9 K 314.5 Mb/s soft (32400,32208) BCH decoder chip for DVB-S2 system," *IEEE J Solid-State Circuits*, vol. 45, no. 11, pp. 2330–2340, 2010.
[15] Y. Lee, H. Yoo, I. Yoo, and I.-C. Park, "High-throughput and low-complexity BCH decoding architecture for solid-state drives," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 5, pp. 1183–1187, 2014.
[16] C.-F. Teng, C.-H. Chen, and A.-Y. Wu, "An ultra-low latency 7.8–13.6 pJ/b reconfigurable neural network-assisted polar decoder with multi-code length support," in *IEEE VLSIC*, 2020, pp. 1–2.
[17] Y. Tao, S.-G. Cho, and Z. Zhang, "A configurable successive-cancellation list Polar decoder using split-tree architecture," *IEEE J Solid-State Circuits*, vol. 56, no. 2, pp. 612–623, 2021.
[18] K. R. Duffy, J. Li, and M. Médard, "Capacity-achieving guessing random additive noise decoding," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4023–4040, 2019.
[19] K. R. Duffy and M. Médard, "Guessing random additive noise decoding with soft detection symbol reliability information," in *IEEE ISIT*, 2019.
[20] A. Solomon, K. R. Duffy, and M. Médard, "Soft maximum likelihood decoding using GRAND," in *IEEE ICC*, 2020.
[21] K. R. Duffy, "Ordered reliability bits guessing random additive noise decoding," in *IEEE ICASSP*, 2021.
[22] W. An, M. Médard, and K. R. Duffy, "CRC codes as error correcting codes," in *IEEE ICC*, 2021.
[23] C. Häger and H. D. Pfister, "Approaching miscorrection-free performance of product codes With anchor decoding," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 2797–2808, 2018.
[24] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control.*, vol. 3, no. 1, pp. 68–79, 1960.
[25] S. M. Abbas, T. Tonnellier, F. Ercan, and W. J. Gross, "High-throughput VLSI architecture for GRAND," in *IEEE SiPS*, 2020.
[26] P. Elias, "Error-free coding," *IEEE Trans. Inf. Theory*, vol. 4, no. 4, pp. 29–37, 1954.