Recent Advances on Stochastic and Noise Enhanced Methods in Error Correction Decoders

Chris Winstead, Tasnuva Tithi
ECE Department
Utah State University
Logan, UT 84322-4120
email: *chris.winstead@usu.edu*
tasnuvatithi@aggiemail.usu.edu

Emmanuel Boutillon
Lab-STICC, UMR 6285
Université de Bretagne Sud
56321 Lorient, France
email: *emmanuel.boutillon@univ-ubs.fr*

Fakhreddine Ghaffari
ETIS, UMR-8051
Université de Cergy-Pontoise, France
email: *fakhreddine.ghaffari@ensea.fr*

***Abstract*—This paper offers a review of recent developments in non-deterministic error correction decoding methods, which can be described in two broad classes. The first class uses stochastic computation to emulate the arithmetic operations of conventional decoding algorithms. The second class achieves noise enhancement by randomly perturbing the calculations of a standard decoder. Stochastic decoders inherit analysis techniques from the conventional algorithms they emulate, but the noise-enhanced algorithms are newer, more difficult to explain, and not yet fully understood. We describe a Markov chain analysis technique to both explain and optimize noise enhancement in these algorithms. Circuit implementation is also discussed, including both conventional hardware architectures and circuits based on memristor threshold logic, where memristor non-determinism can be exploited for noise enhancement.**

I. Introduction

Non-deterministic algorithms are well known in a variety of fields (optimization, imaging, neuromorphic computing, etc), in which the performance or efficiency is enhanced beyond what could be obtained by a deterministic method. For many years, there has been a corresponding interest in reliable or fault-tolerant computing with non-deterministic physical components. In this paper we survey a family of non-deterministic error correction algorithms and circuit implementations that merge these topics, providing a useful framework for noise enhancement at the physical layer of data communication, memory and storage. Most of the recent advances in this area focus on Low Density Parity Check (LDPC) codes, a well known class of error correction algorithms that are now included in numerous data communication standards. The survey will therefore be limited to the domain of LDPC codes, though the concepts are by no means restricted to that class of codes.

LDPC codes are used to encode data by inserting redundant parity bits in anticipation of corruption due to noisy communication channels or faulty memory cells. Error correction is usually done using Bayesian Belief Propagation (BP) or related algorithms. BP-based algorithms perform exceptionally well in that they achieve very low error rates after decoding retrieved data. An alternative class of Bit-Flipping (BF) algorithms is also widely studied. BF decoders are much simpler to implement in hardware, but achieve comparatively poor error rates. During the past two decades, a great deal of research has been invested into designing algorithms that optimize the tradeoff between BP's excellent performance and BF's simplicity.

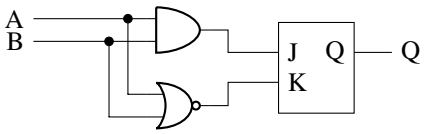


Fig. 1: Stochastic variable node implementation based on a Muller C-element with J/K flip-flop.

A. Review of stochastic decoding algorithms

LPDC decoders belong to a class of message-passing algorithms defined on factor graphs. A message passing decoder computes local probability or likelihood metrics at each node. Local calculations are iterated until all parity-check constraints are satisfied, indicating global convergence.

A typical BP-based decoder must perform a number of integer additions, lookup table operations, or comparisons to obtain a metric for every edge in the graph. In order to reduce the hardware cost of these decoders, beginning in 2003 researchers considered stochastic computing methods to emulate the required arithmetic [1]. The basic concept is to compute probability metrics by filtering pseudorandom bit sequences that model a Bernoulli process. A probability value is directly encoded in the frequency of 1's appearing in the sequence. The hardware is often simpler than BP-based decoders: adders are replaced by Muller C-element gates (similar in function to J/K flip-flops). An example C-element implementation is shown in Fig. 1. In the figure, the processor receives two input Bernoulli streams, A and B, with statistics p_A and p_B, respectively. It is well known that the output is a sequence with statistic

$$p_Q = \frac{p_A p_B}{p_A p_B + (1-p_A)(1-p_B)}$$

which corresponds to the BP symbol node operation.

For an acyclic graph, a stochastic decoder has performance equivalent to BP. Most code graphs are cyclic, and in this case the stochastic algorithm becomes trapped in deterministic

patterns, resulting in uncorrectable error patterns along the cyclic subgraphs. To resolve this problem, the variable node implementation (Fig. 2) is modified to have non-deterministic behavior [2]. The best performing modification is the family of "edge memory" methods, which restore performance close to BP [3]. An edge memory is a module that samples the statistics of an input bit stream, and emits a separate, independent stream with the same statistics. If an edge memory is inserted in every cycle within the decoder, then deterministic loops are eliminated while preserving the BP-based probability calculations.

B. Noise enhanced decoders

Arguably the first observation of non-deterministic enhancement was in 2001, in the context of analog decoders where device mismatch was observed to enhance performance in some cases [4]. This phenomenon was explained by the sub-optimality of BP on cyclic code graphs, which opened a possibility for accidental improvement in randomly altered designs. A more systematic approach to noise enhancement appeared with probabilistic bit flipping (PBF) in 2005, where Gallager-style bit-flipping algorithm was modified to flip bits only with probability p (i.e. the algorithm's normal bit flips are suppressed with probability $1-p$) [5]. In 2009, noise enhancement was observed for stochastic decoders implemented with a mixture of permanent defects and transient errors [6], where noise-induced upsets were found beneficial for compensating permanent faults.

In 2012, dithered belief propagation was described, marking the first explicit construction of a noise enhanced BP decoder where decoding operations are deliberately perturbed by artificial noise [7]. This was followed soon after by four developments in 2014. First, noise enhancement was observed for the BP-based Min-Sun (MS) algorithm on the binary symmetric channel (BSC) [8]. In the same year, Noisy Gradient Descent Bit Flipping (NGDBF) was described for the Gaussian channel [9] and the related Probabilistic Gradient Descent Bit Flipping (PGDBF) algorithm for the BSC [10]. These works led to a number of algorithm and hardware improvements for NGDBF [11]–[13] and PGDBF [14]–[17], along with potential specialized applications in solid-state storage [18]. A related approach, known as the Improved Differential Binary (IDB) decoder, was also proposed in 2014 [19]. The IDB decoder uses a combination of feedback (similar to the stochastic TFM) and perturbed threshold operations.

The GDBF algorithm is based on sum-and-threshold operations, as shown in Fig. 2. First, a hard decision syndrome vector is computed by the parity-check nodes. Then, for each variable node, an "energy function" is computed (also called a "reliability" or "flipping" function) by the expression

$$E_k^{(\text{GDBF})} = x_k y_k + \sum_{j \in \mathcal{M}_k} s_j, \tag{1}$$

where y_k is the channel information, $x_k \in \{+1, -1\}$ is the bipolar hypothesis decision, $\mathcal{M}_k$ is the parity-check

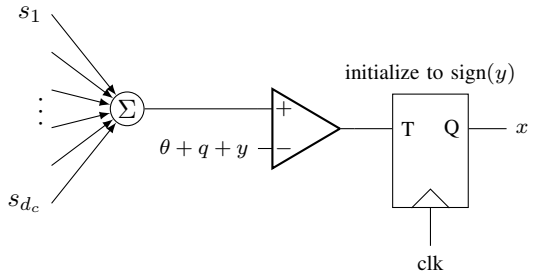


Fig. 2: Typical variable node implementation for the NGDBF algorithm based on a toggle flip-flop. Inputs are channel information y, threshold θ, random perturbation q, and adjacent parity-check messages s_j.

neighborhood of the k^{th} variable node, and the s_j are bipolar syndrome values (parity is satisfied when $s_j = +1$ and violated when $s_j = -1$). In each iteration, the decision x_k is flipped if $E_k < \theta$, where θ is a threshold that may vary from iteration to iteration or, in some algorithms, from node to node.

In the NGDBF algorithm, the threshold operation is perturbed by injected noise q:

$$E_k^{(\text{NGDBF})} = x_k y_k + \sum_{j \in \mathcal{M}_k} s_j + q, \tag{2}$$

In the PGDBF algorithm, the sum-and-threshold operation is carried out using the GDBF rule (1), and if $E_k < \theta$ then the bit is flipped only with probability p_{flip}. In the NGDBF and PGDBF algorithms, some kind of random number generator (RNG) is needed to generate the non-deterministic behavior. For NGDBF, Gaussian perturbations have generally been used, but other distributions have also been used with a small performance penalty [9]. There are now numerous variations on the NGDBF and PGDBF algorithms, which differ mainly in the method and scheduling of how θ is determined.

Besides bit-flipping and BP-based decoders, other types of noise-enhanced decoders have been reported, such as a recent stochastic BCH decoder [20] and a stochastic Chase algorithm [21]. These are often described as stochastic decoders, however they do not utilize the hardware techniques traditionally labeled as "stochastic computing" and are arguably more similar to noise-enhanced decoders than to the BP-based stochastic decoders described in Sec. I-A, since they perform a discrete codeword search aided by non-determinism. Research on noisy BP-based decoders has also continued to advance, as with the recent Noise-Against-Noise Decoder (NAND) algorithm [22].

II. Theory

The development of noise-enhanced decoders was experimentally driven, and significant performance enhancing effects were discovered and optimized empirically prior to developing a sound theoretical footing. The empirical approach nevertheless realized a high degree of performance gain. One example is shown in Fig. 3, which compares performance of stochastic and NGDBF algorithms with ideal BP decoding on the AWGN

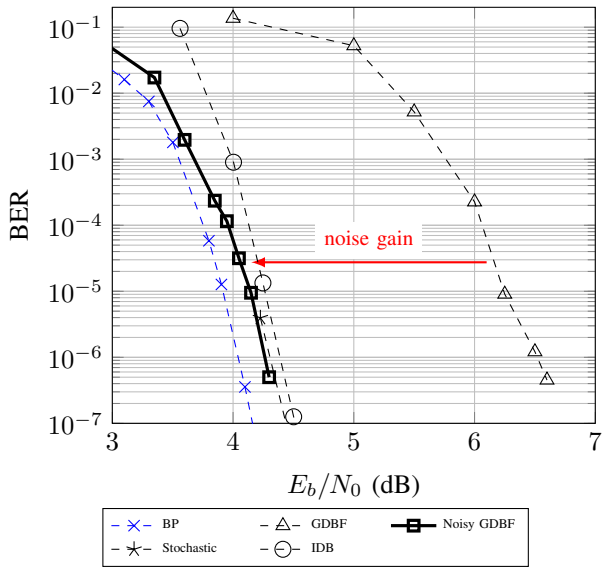


Fig. 3: Example of noise-enhanced performance for stochastic, IDB and NGDBF algorithms, applied to the IEEE 802.3an 10GBase-T standard. NGDBF allows a maximum of 600 iterations to decode a frame.

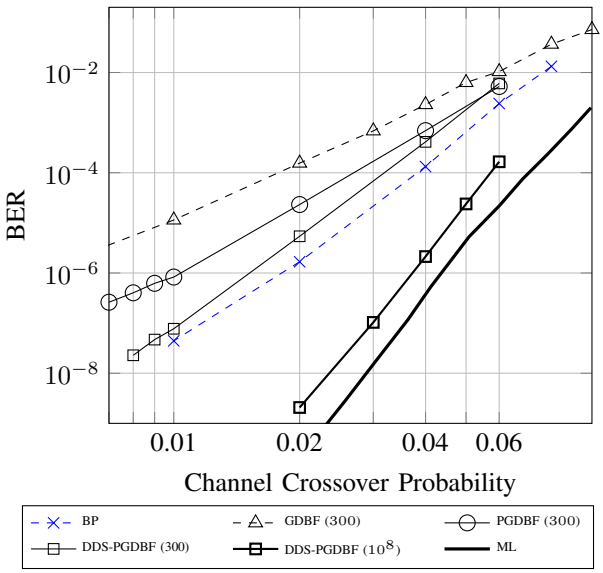


Fig. 4: Example of ML-approaching performance for the PGDBF algorithm on the Tanner code with $N = 155$. For each algorihtm, the maximum allowed iterations per frame are indicated in parentheses.

channel. Compared to noiseless GDBF, the leap in performance due to solely to noise is particularly striking. A second example is shown in Fig. 4, which shows performance of a "decoder-dynamic shift" (DDS) PGDBF algorithm [23] approaching maximum-likelihood performance on the BSC when a large iteration limit is allowed. The noise gains depicted in Figures 3 and 4 were discovered essentially by accident, and theoretical explanations came later. To date, the most productive explanation is that noise tends to disrupt trapping sets in the GDBF decoder. Trapping set analysis aids in parametric optimization for noisy decoders, but does not support ensemble optimizations like the ones obtained with density evolution for BP-based decoders.

LDPC codes are represented by compact graphs with many cycles, which break the formal correctness of BP and related algorithms. Certain error patterns can be difficult or impossible to correct on cyclic subgraphs known as trapping sets. In the case of BP-based algorithms, trapping sets are well known as the cause of error floors. In the domain of bit-flipping algorithms, trapping sets have proven useful for understanding and optimizing noise enhancement effects.

A Markov process approach to trapping sets was previously described [23], and is here illustrated by a simple example. We consider a trapping set involving m variable nodes. Then there are 2^m possible binary states for the variable nodes involved in the trapping set. Only one of these states is correct, and without loss of generality we assume it is state 0. Since the decoding algorithm is non-deterministic, in each iteration it may undergo a transition from its present state s_i to any other allowed state s_j with probability p_{ij}, hence the decoding iterations constitute a homogeneous Markov chain characterized by a square transition matrix Γ of dimension 2^m with entries p_{ij}.

For an initial erroneous state $\vec{s}_0$, the probability that the error is corrected after L iterations is the zeroth element of $\vec{s}_0\Gamma^L$. Since Γ and s_0 depend on the received channel information, we use a Markov Chain Monte Carlo procedure to evaluate the trapping set's contribution to the error probability over many frames. This procedure reveals that noise is essential for BF algorithms to correct trapping set errors. Furthermore, since Γ is dependent on various algorithms parameters (e.g. the noise statistics and distribution, weighting or clipping of channel samples, etc), these parameters can be adjusted and optimized at design time [24].

III. Implementation

A. Conventional ASIC and FPGA architectures

ASIC implementations of stochastic and noise enhanced BF decoders usually are fully parallel architectures that precisely map the code's factor graph topology. The circuit implementations have few degrees of freedom for the variable and check nodes, hence most design effort tends to focus on generating and distributing random numbers or on the design of efficient edge memories. Recent stochastic decoders proved competitive with BP-based decoders in both performance and silicon area [25]. More recent BF implementations require $2\times$ to $3\times$ less area than stochastic decoders, with similar or better decoding performance. The area reduction is explained by the data flow architectures shown in Fig. 5. In every iteration, stochastic decoders must compute two messages for every edge in the factor graph, whereas the BF decoders need only compute one message per node.

The normalized power consumption of BF decoders, reported in Joules per bit, is the lowest among decoders implemented in comparable technology. Due to their non-deterministic convergence, stochastic and noisy BF decoders cannot guarantee the same latency for all frames, and may require buffering frames to accommodate occasional delays. The average throughput is nevertheless high enough for major data communication standards, such as IEEE 802.3an 10GBase-T [11], [25].

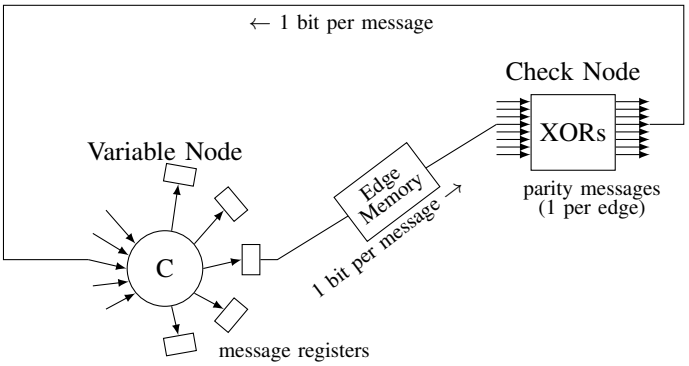
Non-deterministic BF decoders have been reported to perform well with a variety of non-ideal random number sources. Where the earliest designs typically employed linear feedback shift registers (LFSRs) to produce random numbers, it would be prohibitive to include an LFSR in every variable node. This motivated random sample reuse, where a large shift register is used to distribute samples from a single RNG [9]. Simpler approaches use no RNG at all, as random samples can be pre-generated as initialization values for a circular shift register [11]. Another LFSR-free approach uses an "intrinsic valued random generator" to extract random numbers from switching activity within the decoder [26].

For quasi-cyclic LDPC (QC-LDPC) codes, the recently described Variable Node Shift Architecture (VNSA) implements PGDBF without requiring any RNG [17]. In the VNSA, a unique pre-generated noise sample is hard-coded into each variable node. The code's structure permits cyclically shifting the variable node states after each iteration. With each iteration, the variable nodes' logical positions are shifted to new physical locations, where they obtain an updated noise sample from the local hardware instantiation. The VNSA decoder has hardware area almost as low as GDBF, making it one of the most area efficient decoder architectures.

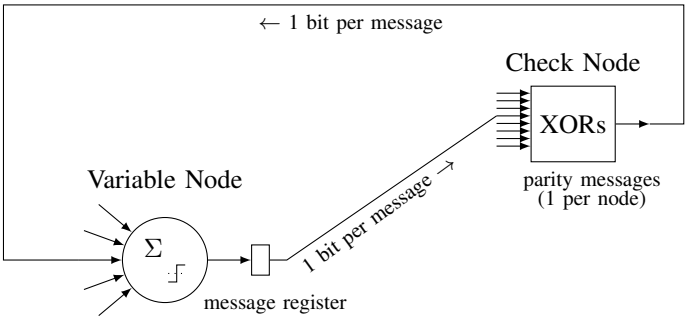
B. Unconventional emerging devices: memristors

In the development of stochastic and noise-enhanced decoders, two central points emerged: first, deterministic logic is fatal for stochastic decoders with cycles; second, non-determinism is essential for performance gain in noise enhanced decoders. In hardware implementation, non-determinism is usually provided using RNG circuits that incur substantial overhead in area and power. There is now an interest in exploiting nano-scale "native non-determinism" as a computational resource to mitigate the hardware cost of RNGs.

Recent research points to resistance-switching devices, commonly identified as a subclass of memristors, as having non-deterministic switching statistics that can be modeled with some precision, and can be adjusted by manipulating pulse timing or amplitude in memristive logic circuits [27]. A memristor of this type can be induced to switch between high-resistance and low-resistance states by applying forward and reverse voltages (respectively) in excess of the device's threshold voltage. When a lesser sub-threshold voltage is applied, the device's switching is a stochastic process. This phenomenon already attracted interest from the neuromorphic circuits community, where



(a) Single-bit message passing decoder.



(b) True bit-flipping decoder.

Fig. 5: Comparison of iterative data flow in stochastic vs BF architectures.

stochastic switching provides a model for neural networks based on spike timing dependent plasticity (STDP) [27], [28]. Memristor circuits were previously considered for LDPC decoders [29], and for stochastic computation [30].

A possible memristor-based randomness source is shown in Fig. 6. The memristor is initialized in a high-resistance state by applying a reverse pulse V_{rst} which exceeds the memristor's switching threshold. A train of sub-threshold pulses are then applied at V_p. A sub-threshold pulse can cause the memristor to switch to a low-resistance state with probabiilty p_{sw}. The switching statistics are a function of the pulse amplitude and width [27]. The memristor's resistance state is detected by monitoring the voltage in the resistor divider circuit formed by R. Each time the memristor switches to a low-resistance state, a reset cycle is required to restore the memristor to its high-resistance condition. The digital output Q is a true random Bernoulli sequence. Memristor-based random generators therefore have the potential to produce random perturbations with superior statistical features (i.e. true random instead of pseudo-random sequences), with less circuit area than traditional RNGs.

IV. Conclusion

This paper gave an abbreviated review of recent progress in stochastic and noise-enhanced decoding, emphasizing advances during the past five years. Progress in this topic appears to be accelerating and there are now applications for commercial

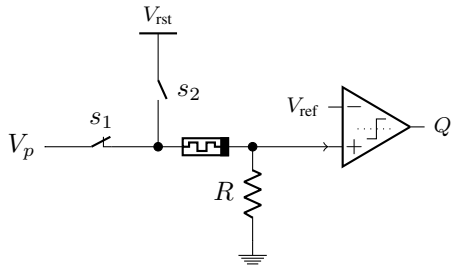


Fig. 6: A memristor-based randomness source.

standards. Better theoretical tools are still needed to understand convergence behavior and optimize throughput, to establish performance bounds, and to determine optimal code constructions for use with these algorithms. With the introduction of memristor circuits, new territory is now opened for continued research and discovery bridging the physical and algorithmic foundations of communication and memory.

ACKNOWLEDGEMENT

This work was partly supported by the US National Science Foundation under award number 1410000, by the State of Utah USTAR initiative under the UTAG program, and by the french ANR under grant number ANR-15-CE25-0006-01 (NAND project).

REFERENCES

[1] V. C. Gaudet and A. C. Rapley, "Iterative decoding using stochastic computation," *Electronics Letters*, vol. 39, no. 3, pp. 299–301, 2003.

[2] C. Winstead, V. C. Gaudet, A. Rapley, and C. Schlegel, "Stochastic iterative decoders," in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*. IEEE, 2005, pp. 1116–1120.

[3] S. S. Tehrani, W. J. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," *IEEE Communications Letters*, vol. 10, no. 10, pp. 716–718, Oct 2006.

[4] F. Lustenberger and H.-A. Loeliger, "On mismatch errors in analog-VLSI error correcting decoders," in *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, vol. 4. IEEE, 2001, pp. 198–201.

[5] N. Miladinovic and M. P. Fossorier, "Improved bit-flipping decoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1594–1606, 2005.

[6] C. Winstead and S. Howard, "A probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 6, pp. 484–488, 2009.

[7] F. Leduc-Primeau, S. Hemati, S. Mannor, and W. J. Gross, "Dithered belief propagation decoding," *IEEE Transactions on Communications*, vol. 60, no. 8, pp. 2042–2047, 2012.

[8] C. L. K. Ngassa, V. Savin, and D. Declercq, "Unconventional behavior of the noisy min-sum decoder over the binary symmetric channel," in *2014 Information Theory and Applications Workshop (ITA)*, Feb 2014, pp. 1–10.

[9] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for LDPC codes," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3385–3400, Oct 2014.

[10] O. Al Rasheed, P. Ivanis, and B. Vasic, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *Communications Letters, IEEE*, vol. 18, no. 9, pp. 1487–1490, Sept 2014.

[11] G. Sundararajan and C. Winstead, "ASIC design of a noisy gradient descent bit flip decoder for 10GBASE-T ethernet standard," arXiv:1608.06272, 2016.

[12] J. Asatani, Y. Kondo, K. Katayama, E. Kulla, and H. Tokushige, "An improved noisy gradient descent bit-flipping decoding algorithm for LDPC codes," in *2016 International Symposium on Information Theory and Its Applications (ISITA)*, Oct 2016, pp. 591–595.

[13] B. Dai, R. Liu, C. Gao, and Z. Mei, "Noisy gradient descent bit-flipping decoder based on adjustment factor for LDPC codes," *IEEE Communications Letters*, vol. 22, no. 6, pp. 1152–1155, 2018.

[14] K. Le, F. Ghaffari, D. Declercq, and B. Vasic, "Hardware optimization of the perturbation for probabilistic gradient descent bit flipping decoders," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[15] K. Le, F. Ghaffari, D. Declercq, and B. Vasić, "Efficient hardware implementation of probabilistic gradient descent bit-flipping," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 4, pp. 906–917, 2017.

[16] B. Unal, A. Akoglu, F. Ghaffari, and B. Vasić, "Hardware implementation and performance analysis of resource efficient probabilistic hard decision LDPC decoders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, no. 99, pp. 1–11, 2018.

[17] K. Le, D. Declercq, F. Ghaffari, L. Kessal, O. Boncalo, and V. Savin, "Variable-node-shift based architecture for probabilistic gradient descent bit flipping on QC-LDPC codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 7, pp. 2183–2195, July 2018.

[18] P. Ivaniš and B. Vasić, "Error errore eicitur: A stochastic resonance paradigm for reliable storage of information on unreliable media," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3596–3608, Sept 2016.

[19] K. Cushon, S. Hemati, C. Leroux, S. Mannor, and W. J. Gross, "High-throughput energy-efficient LDPC decoders using differential binary message passing," *IEEE Transactions on Signal Processing*, vol. 62, no. 3, pp. 619–631, Feb 2014.

[20] K. Han, J. Wang, and W. Gross, "A hardware efficient soft-decision BCH decoder for WBAN systems using stochastic computing," in *2018 IEEE Midwest Symposium on Circuits and Systems (MWSCAS)*, August 2018.

[21] C. Leroux, S. Hemati, S. Mannor, and W. J. Gross, "Stochastic Chase decoding of Reed-Solomon codes," *IEEE Communications Letters*, vol. 14, no. 9, pp. 863–865, September 2010.

[22] F. Cochachin, D. Declercq, E. Boutillon, and L. Kessal, "Density evolution thresholds for noise-against-noise min-sum decoders," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct 2017, pp. 1–7.

[23] D. Declercq, C. Winstead, B. Vasic, F. Ghaffari, P. Ivanis, and E. Boutillon, "Noise-aided gradient descent bit-flipping decoders approaching maximum likelihood decoding," in *International Symp. on Turbo Codes and Iterative Information Processing (ISTC)*. IEEE, 2016, pp. 300–304.

[24] K. Le Trung, "New direction on low complexity implementation of probabilistic gradient descent bit-flipping decoder," Ph.D. dissertation, université de cergy-pontoise, 2017.

[25] S. S. Tehrani, A. Naderi, G. A. Kamendje, S. Hemati, S. Mannor, and W. J. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4883–4896, Sept 2010.

[26] K. Le, F. Ghaffari, D. Declercq, and B. Vasic, "Efficient hardware implementation of probabilistic gradient descent bit-flipping," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 4, pp. 906–917, 4 2017.

[27] M. Al-Shedivat, R. Naous, G. Cauwenberghs, and K. Salama, "Memristors empower spiking neurons with stochasticity," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 5, no. 2, pp. 242–253, June 2015.

[28] R. Naous, M. Al-Shedivat, and K. N. Salama, "Stochasticity modeling in memristors," *IEEE Transactions on Nanotechnology*, vol. 15, no. 1, pp. 15–28, 2016.

[29] J. H. Poikonen, E. Lehtonen, M. Laiho, and J. K. Poikonen, "Memristive circuits for LDPC decoding," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 4, pp. 412–426, Dec 2014.

[30] P. Knag, W. Lu, and Z. Zhang, "A native stochastic computing architecture enabled by memristors," *IEEE Transactions on Nanotechnology*, vol. 13, no. 2, pp. 283–293, March 2014.