Approaching Maximum Likelihood Performance of LDPC Codes by Stochastic Resonance in Noisy Iterative Decoders

Bane Vasić
Department of ECE,
University of Arizona
Email: vasic@ece.arizona.edu

Predrag Ivaniš
School of Electrical Engineering,
University of Belgrade
Email: predrag.ivanis@etf.rs

David Declercq, Khoa LeTrung
ETIS lab, ENSEA
Université de Cergy-Pontoise/CNRS
Emails: declercq,letrung@ensea.fr

***Abstract*—In the 1960s–70s, Taylor and Kuznetsov obtained a remarkable result that information can be reliably retrieved from a noisy channel even if a decoder is made of noisy components. The results of Vasic and Chilappagari presented at the ITA Workshop ten years ago have revived the interest in decoders made of noisy hardware and since then a number of improvements of the iterative decoders have been made to bring their performance closer to that of their perfect counterparts. However, a common mantra has been that noisy decoders cannot be better than their perfect counterparts. In this talk we report an unexpected phenomenon we have recently discovered—noise can actually improve the error correction process by reducing the probability of decoding error, in some cases by more that two orders of magnitude. This new form of stochastic resonance enables us to use logic gate errors to correct channel errors. This novelty recognizes that the decoder—essentially an iterative minimization of the Bethe free energy on the code graph—can get trapped in local minima, and random perturbations help the decoder to escape from these minima and converge to a correct code-word. In the spirit of Marcus Tullius Cicero's "Clavus clavo eicitur," ("one nail drives out another") they operate on the principle: Error errore eicitur" - "one error drives out another." Crucially, such useful random perturbations require neither additional hardware nor energy, as they are built into the low-powered, noisy hardware itself.**

I. Introduction

The maximum likelihood (ML) decoding of a binary code is a Voronoi binning of points in the Hamming space with codewords as centers of the regions. Voronoi decomposition completely characterizes the performance of the decoding algorithm, but naturally, such an algorithm has exponential complexity and requires global knowledge of the entire space. On the other hand, iterative message passing decoders achieve low complexity by requiring only local knowledge - the value of a bit and its intermediate dependencies of other bits. The dependencies are described by a graph, and the algorithm operates on this graph. Hence, both performance and complexity of iterative decoders are governed by the nature of the local computation in the code graph as well as the choice of the graph. The suboptimality of the existing iterative decoders results from the fact that they ignore the topology of the bit neighborhood as they still operate under the assumption that the graph is a tree. However, the concept of message passing is rich enough to describe a large class of decoders - even ML decoders - by using messages which convey information regarding the local neighborhood of a node in the graph. An extreme example would be a decoder in which every message contains a description of the entire graph. Even more extreme would be a decoder in which each message conveys the description of the Voronoi tessellations. A natural question that arises in this context. What are the local message passing rules used in graph nodes to update the messages in a given iteration in order to adequately and efficiently use the information about the neighborhoods? At first sight, conveying the local structure of the Tanner graph by employing more bits in the decoder appears complicated and one might perhaps be tempted to conjecture that the message length and consequently the complexity would be prohibitively large. However as our results [1] demonstrate, small number of additional bits in the messages can implicitly carry the topological information and a variable node can infer such information on the basis of the incoming messages. In other words, the topology surrounding the variable node is compressed into a small number of bits in messages by using knowledge of what messages are possible for a given topology. The key point lies in the fact that all the computations in such Finite Alphabet Iterative Decoders (FAID) still remain local in nature [2], [3]. In addition to the fact that it lends itself to low complexity implementation, the above low message precision feature has another important implication. A number of different decoders exists that employ the same number of bits, and error patterns uncorrectable by one decoder may be correctable by another decoders [4]. As we have shown [3], it is beneficial to use a set of decoders and switch from one decoder to another to ensure success on a large set of error patterns.

In our work on ML approaching performance of FAID decoders under the diversity framework, for a given code we found a set of FAIDs is capable of correcting errors not correctable by individual decoders in the set [2], [3]. Moreover, we found that decoder diversity technique is ML approaching when the decoder set is large (9,236 distinct decoding rules in [3]). Moreover, the switching between decoders can be

accomplished by minor variations of the Boolean functions representing the message maps and therefore does not require elaborate changes to decoder architecture.

However, an interesting situation arises when a decoder itself is made of noisy components. For example, in low-powered submicron complementary metal–oxide–semiconductor (CMOS) chips, the supply voltage is kept low in order to reduce power consumption, thus making logic gates susceptible to noise and increasing the probability of incorrect logic gate output. Due to unreliability of its logic gates, the correction circuit - whose purpose is to correct errors - introduces errors in the process of correcting errors from the storage medium. Making logic gates reliable (for example by using larger supply voltages) appears as a logical solution. We show that the correction circuit can operate in the unreliable (low-power) gate - regime, and still be able to correct more errors than the noiseless correction circuit. Moreover, the random perturbations do not require any additional computational resources as they are built in the unreliable hardware itself. The main idea of this paper is that decoder diversity is inherently present in a single noisy decoder and that different decoders can be readily obtained simply, for example by changing supply voltage of some gates in the decoder, thus modifying the node update rule and consequently the decoding dynamics.

Fault-tolerant decoding and storage has attracted significant attention lately, and numerous approaches have been proposed which exploit the inherent redundancy of the existing decoders [5], [6], [7], [8]. It is not obvious that such noisy decoders can ensure reliable storage of information because more powerful decoders – capable of correcting more errors in the medium – may require more logic gates, thus introducing more computational errors. In light of that, it is even more surprising that the logic gate errors may help the decoder. We proceed to show exactly this. We present a class of noisy decoders that perform better than their noiseless counterparts.

The first trace of using randomness in the decoder can be found in Gallager's work where the random flips are used to resolve ties in the majority voting operation in the variable node, while the first iterative decoding algorithm that explicitly relies on randomness to correct errors is Miladinovic and Fossorier's Probabilistic Bit Flipping (PBF) [9]. A closely related technique of adding noise to messages in a BP decoder on the AWGN channel is by Leduc-Primeau *et al.* [10] for reducing error floor in the context of noiseless decoders. In these papers the randomness is deliberate. Recently it was shown by Sundararajan *et al.* [11] that random perturbations can be used to increase the performance of a gradient descent bit flipping decoder (GDBF), introduced by Wadayama *et al.* [12]. At the same time we observed that the randomness coming from computational noise even more improves the GDBF decoding performance. Based on that result we developed a probabilistic gradient-descent bit flipping (PGDBF) algorithm [13] for the Binary Symmetric Channel (BSC). Introducing a random perturbation is reminiscent of the operation of mutation in genetic algorithms [14] (the inverse or energy function in the PGDBF algorithm is reminiscent of the fitness function in genetic algorithms). Similar effect is observed in neural networks. For example, Karbasi *et al.* [15] showed than noise improves performance of the recall phase in associative memories.

For the scheme to work, not all logic gates can be allowed to be noisy. Small fraction of critical ones are made reliable. This can be practically done using larger transistor size, higher voltage supply or slower clock. Secondly, to avoid accumulation of errors and divergence from the true codeword, the decoder is periodically rewound - i.e., re-initialized and restarted.

II. Noisy Decoder

Prior to transmission over the Binary Symmetric Channel (BSC) with probability of error α_M, each k-bit user information is encoded by an (n, k) linear block code of length n and code rate $R = k/n$. Due to channel errors, the codeword $\mathbf{x}$ is received as $\mathbf{y}$. Thus, $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, the word at the output of the channel can be expressed as $\mathbf{y} = \mathbf{x} \oplus \mathbf{e}$, where $\oplus$ denotes XOR operation, and the elements of the vector $\mathbf{e}$ are independent Bernoulli random variables with parameter α_M. The vector $\mathbf{y}$ is input to the iterative decoding algorithm whose goal is to recover $\mathbf{x}$.

A noisy decoder can be defined as a circuit which performs computations on noisy logic gates that follow the Neumann failure mechanism. An obvious way to ensure robustness of a decoder is to employ the von Neumann multiplexing [16]. However, this comes with a price of very large redundancy. The first attempt to use a more advanced coding scheme from unreliable components is due to Taylor [17] and Kuznetsov [18]. The recent impressive developments on LDPC codes has renewed the interest for these early works.

Our approach is based on low-density parity check (LDPC) codes and iterative decoding [19]. It is applicable to any LDPC code type, but for the sake of clarity, we discuss a subclass of the (γ, ρ)-regular LDPC code ensemble [19]. The integer parameters γ and ρ determine code rate $R \geq 1 - \frac{\gamma}{\rho}$, and the structure of the decoder, namely the number of input arguments to Boolean functions used in the decoder.

The noisy iterative decoding algorithm [20] operates on a graphical representation of a code $G = (V \cup C, E)$. The code graph G is a bipartite graph whose edges in the set E connect the variable nodes in the set V with check nodes in C. The decoding algorithm consists of sending binary messages between variable nodes $v \in V$ corresponding to code bits and check nodes, $c \in C$ corresponding to parity check equations in which the variables (bits) are involved in. Various choices of algorithms are considered in literature. Our decoding algorithms are based on the modification of the Gallager-B [19] and gradient-descent bit-flipping (GDBF) [12] algorithms, for which the messages are represented by a single bit. For these algorithms the update functions require only two types of logic gates: majority logic (MAJ) and XOR gates. Note the logic gate errors can occur because of gate

unreliability, but can be also deliberately inserted with a goal to improve the performance of the decoder.

For these noisy decoders, we represent the effect of the noisy gates by errors on the messages exchanged between the two sets of nodes, in the following way. The messages $\tilde{\nu}^{(\ell)}$ from variable nodes and $\tilde{\mu}^{(\ell)}$ from check nodes during the ℓ-th iteration are expressed as

$$\begin{aligned}\tilde{\nu}^{(\ell)}_{v\to c} &= \nu^{(\ell)}_{v\to c} \oplus e^{(\ell)}_{MAJ,v\to c}\\ \tilde{\mu}^{(\ell)}_{c\to v} &= \mu^{(\ell)}_{c\to v} \oplus e^{(\ell)}_{\oplus,c\to v}.\end{aligned} \tag{1}$$

where $\nu^{(\ell)}$ and $\mu^{(\ell)}$ represent their counterparts calculated on reliable hardware. $e^{(\ell)}_{MAJ}$ and $e^{(\ell)}_{\oplus}$ in Eq.1 denote the errors in majority logic and XOR gates affecting computation of the messages $\nu^{(\ell)}_{v\to c}$ and $\mu^{(\ell)}_{c\to v}$, respectively. If the decoder was deterministic, then $e^{(\ell)}_{MAJ}$ and $e^{(\ell)}_{\oplus}$ in Eq.1 would be zero. For noisy decoders, they are independent Bernoulli random variables with parameters α_{MAJ} and $\alpha_{\oplus}$, respectively.

Let us now describe the specifics of the two algorithms used in this paper.

Gallager B decoder For the Gallager B we have $\nu^{(\ell)}_{v\to c} = \Phi_1(y_v, \mathbf{m}^{(\ell)})$, where $\mathbf{m}^{(\ell)} = \tilde{\mu}^{(\ell)}_{\mathcal{N}_v}$ denote the incoming messages to the variable node v from all its neighbors $\mathcal{N}_v$ excepts the check node c. The variable node update function $\Phi_1(\cdot)$ can be express as follows

$$\Phi_1(y_v, \mathbf{m}^{(\ell)}) = \begin{cases} s & \text{if } |\{c' \in \mathcal{N}_v \setminus c : \tilde{\mu}^{(\ell)}_{c'\to v} = s\}| > \lfloor \frac{\gamma}{2} \rfloor, \\ y_v & \text{otherwise.}\end{cases}$$

Similarly, $\mu^{(\ell)}_{c\to v} = \Psi_1(\mathbf{n}^{(\ell-1)})$, where $\mathbf{n}^{(\ell)} = \tilde{\nu}^{(\ell)}_{\mathcal{N}_c}$ denote all incoming messages to the check node c from its neighbors except the variable node v. For the Gallager B, the check node update function $\Psi_1(\cdot)$ is a simple XOR function

$$\Psi_1(\mathbf{n}^{(\ell-1)}) = \bigoplus_{v'\in\mathcal{N}_c\setminus\{v\}} \tilde{\nu}^{(\ell-1)}_{v',c}.$$

GDBF decoder The Gradient Descent Bit Flipping decoder [12] does not propagate extrinsic informations between the nodes. Instead, all messages sent from the variable node v in the ℓ-th iteration are the same and represent the current estimate of the bit $\nu^{(\ell)}_v$, i.e. $\nu^{(\ell)}_v = \nu^{(\ell)}_{v\to c}$, $\forall c \in \mathcal{N}_v$. Accordingly, all messages sent from the check node c are equal, and represent the value of the parity check i.e. $\mu^{(\ell)}_c = \mu^{(\ell)}_{c\to v}$, $\forall v \in \mathcal{N}_c$.

At iteration ℓ, the bit estimate is the result of the update function $\Phi_2(y_v, \nu^{(\ell-1)}_v, \tilde{\mathbf{m}}^{(\ell)})$, where $\tilde{\mathbf{m}}^{(\ell)}$ represents the set of all incoming noisy messages to the variable node v.

$$\Phi_2(y_v, \nu^{(\ell-1)}_v, \tilde{\mathbf{m}}^{(\ell)}) = \begin{cases} \nu^{(\ell-1)}_v \oplus 1 & \text{if } \Lambda^{(\ell)}_v = b^{(\ell)}, \\ \nu^{(\ell-1)}_v & \text{otherwise,}\end{cases}$$

where the energy function [12] $\Lambda^{(\ell)}_v$ is calculated as

$$\Lambda^{(\ell)}_v = \nu^{(\ell-1)}_v \oplus y_v + \sum_{c\in\mathcal{N}_v} \tilde{\mu}^{(\ell)}_c,$$

the threshold value $b^{(\ell)}$ represents the maximal value of energy function at the current iteration, i.e., $b^{(\ell)} = \max_v(\Lambda^{(l)}_v)$. The procedure of dynamical tuning of the threshold $b^{(\ell)}$ is considered in [21], where we shown that the performance of GDBF can be improved for some codes if the threshold value changes with the iteration number. In this paper we assume that the circuit that performs the calculation of the threshold is made of noiseless gates.

In check nodes, only parity check equations are calculated and passed to all neighboring variable nodes

$$\Psi_2(\tilde{\mathbf{n}}^{(\ell-1)}) = \bigoplus_{v'\in\mathcal{N}_c} \tilde{\nu}^{(\ell-1)}_{v',c}.$$

For both type of decoders described above, the iterative procedure is halted when all parity checks are satisfied or the predefined maximum number of iterations L, is reached. The decoding is called successful if a codeword (either correct or wrong) is found. Otherwise, the decoding is said to have failed. The event of producing a codeword estimate which is a wrong codeword is called an undetected error.

A. Rewinding

To allow the decoder to benefit from the noise present in Eq. 1, large number of iterations is needed. However, too many logic gate errors can overwhelm the decoder, and lead to decoder failure or undetected errors. In addition to the update rules given in the previous section, our decoders are equipped with a key feature which prevents the accumulation of errors in the messages when the number of iterations is large.

The decoder is initialized by the channel values, and run for L_0 iterations, after which the bit estimates, $\hat{x}_v$ are stored in a reference vector $\mathbf{u} = (u_v)_{1\le v\le n}$, where $u_v = \Omega(\hat{x}^{(L_0)}_v)$, and $\Omega : \{\pm 1\} \to \mathcal{Y}$ (for the BSC, $\Omega(a) = \text{sgn}(a)$). L_0 is a predefined iteration number (determined for example as iteration in which the noiseless decoder is expected to correct most channel errors).

If a codeword is not found in L_0 iterations, the decoding algorithm is re-initialized with the reference vector $\mathbf{u}$ rather than with the word received from the channel and run for another $r-1$ rounds where the round i has L_i iterations, $\sum_{0\le i\le r-1} L_i = L$. At the beginning of each round, the decoder is rewind and initialized by the reference vector $\mathbf{u}$.

$$u^{(\ell)}_v = \begin{cases} y_v & \ell = 0 \\ \Omega(\hat{x}^{(L_0)}_v) & \text{otherwise}\end{cases}$$

Thus instead of running the whole L iterations, the decoder instead runs r shorter rounds.

$$\nu^{(\ell)}_{v\to c} = \begin{cases} u_v & \ell \in \mathcal{I} \\ \Upsilon(\cdot) & \text{otherwise}\end{cases}$$

where

$$\Upsilon(\cdot) = \begin{cases} \Phi_1(u^{(\ell)}_v, \mathbf{m}^{(\ell-1)}) & \text{for Gallager B} \\ \Phi_2(y_v, \hat{x}^{(\ell-1)}_v, \tilde{\mathbf{m}}^{(\ell-1)}) & \text{for GDBF}\end{cases}$$

$\mathcal{I} = \{I_i\}_{0 \le i < r-1}$ and the iteration numbers at the beginning of the rounds are $I_0 = 0$ and $I_i = I_{i-1} + L_{i-1}$.

A decoder with such rewinding schedule is referred to as the rewind-decoder and denoted by $\mathcal{F}^{\circlearrowleft^r}(\mathbf{L})$, where $\mathbf{L} = (L_0, L_1, \ldots, L_{r-1})$. To simplify explanations, we will assume that $L_i = L_R$, $0 < i < r$.

We write this as

$$\mathcal{F}^{\circlearrowleft^r}(\mathbf{L}) = \underbrace{\mathcal{F}(L_0) \diamond \mathcal{F}(L_1) \diamond \cdots \diamond \mathcal{F}(L_{r-1})}_{r}. \qquad (2)$$

where $\diamond$ denotes the rewinding schedule. Clearly, the plain noisy Gallager-B decoder with no rewinding, denoted by $\mathcal{F}(L)$, is a special case of the rewinding decoder, $\mathcal{F}(L) = \mathcal{F}^{\circlearrowleft^1}(L)$.

B. Critical Gates Must be Noiseless

Note that the logic gates gates used to extract the bits from any decoder must be noiseless, otherwise the error rate would be bounded by the reliability of these external gates [20]. Thus, the majority-logic gates in the decoder's decision logic (the function $\hat{\Phi}$) are made of noiseless gates. We also assume that the encoder is noiseless. The effect of errors in the encoder are incorporated in the value of the channel error probability α_M. The register inside the decoder which temporarily stores the word read from a memory medium (the channel values) is also reliable. This is necessary because otherwise the codeword estimate would drift away from the true codeword in the course of decoding, as iterations progress. Registers for storing the intermediate results of computations of Φ and Ψ are unreliable, and their unreliability is accounted for in α_{MAJ} and $\alpha_{\oplus}$.

As syndrome checker is used as a decoding halting criterion, it must be also made noiseless. To reduce power consumption in a decoder, we may want to perform syndrome checking not in every iteration but according to a predefined schedule. The optimal schedule is beyond scope of this paper, and we will simply adopt a schedule in which the decoder runs on noisy hardware for maximum of L iterations, during which the (noiseless) syndrome checker is used only in the a few first iterations and in the last Z iterations.

The decoding is halted in the first of these Z iterations in which a codeword is found. The Z/L ratio determines the decoding efficiency, thus is kept low. In the rewinding schedule, the syndrome checker is used in all L_R iterations in the first decoding round, and after the first rewinding it is used in the last Z iterations in each round.

III. Annihilation of Trapping Sets by Logic Gate Errors

If a decoding graph of the LDPC code has sufficient expansion [22], the noiseless iterative decoder guarantees correction of number of errors linear in codeword length, n. We proved that a noisy iterative decoder also tolerates a constant fraction of failures in all the components [23]. However, since all the gates were assumed to be noisy, the number of correctable errors was smaller then that of the noiseless decoder. For finite length codes, even a noiseless iterative decoder fails to correct some low-weight error patterns due its sub-optimality. The decoding orbit $\{\boldsymbol{\mu}^{(\ell)}\}_{\ell \ge 0}$ corresponding to such error pattern exhibits periodic or fixed point behavior. Subgraphs induced by variable nodes corresponding to the support of these error patterns are known as trapping sets [24]. Now we show that memory element errors inducing trapping sets can be corrected by logic gate errors.

It is known that for small α_M, dominant contribution to the FER comes from small trapping sets [24]. Trapping sets of Gallager-B decoding are completely characterized for the $\gamma = 3$ case [25]. For our discussion it suffices to define an (a, b) trapping set as a bipartite subgraph induced by a variable nodes, which contains b odd degree check nodes.

The above trapping set phenomenon can be explained by viewing iterative decoding as a recursive procedure for Bethe free energy function minimization. The relation of a class of iterative decoders, called Belief Propagation (BP) and the Bethe free energy (BFE) minimization has been established in [26] and studied thoroughly in recent years [27]. The fixed points of BP correspond to the stationary points of the BFE. While in a tree-like graph the BFE is a concave function, in a loopy graph, local minima are present leading to oscillation and multiple fixed points. When the BP is used to compute the marginals of codeword symbols on (loopy) code graphs, these local minima are responsible for decoding failures and correspond to trapping sets [28]. The same holds for iterative decoders that can be derived from BP, such as the Gallager-B algorithm used in our scheme. The reason the trapping sets are annihilated is that the randomness in message updates helps the decoder to escape from these local minima.

Good Deeds of Logic Gate Errors

Now we provide experimental evidence that the errors in logic gates improve the decoding performance. We first show that a decoder with noisy gates is capable of correcting $(5, 3)$ trapping sets that are uncorrectable by the noiseless Gallager-B decoder. We consider the $(3, 5)$-regular $(n, k) = (155, 64)$ LDPC code constructed by Tanner [29], which is widely used in literature. The minimum Hamming distance between codewords is $d_{min} = 20$, thus the noiseless maximum likelihood (ML) decoder would be able to correct any nine-error pattern. However, the noiseless Gallager-B decoder fails on some three-error patterns [30]. In the $(155, 64)$ Tanner code, every uncorrectable three-error pattern induces a (5,3) trapping set of a unique topology. Now, we show that these three-error patterns can be corrected by our noisy decoder.

We consider two different scenarios: (i) when check node processing is noiseless and only MAJ gates are noisy, and (ii) when variable node processing is noiseless and only XOR gates are noisy. The logic gate errors are von Neumann's.

The conditional FER of the Tanner (155,64) code for the most critical three-bit error patterns (denoted by $FER_{\mathbf{e}}$) is presented in (Fig. 1(a) and (b)). The particular low-weight error pattern cannot be decoded by noiseless hardware, but it can be decoded with non-zero probability for a wide range of gate error probabilities α_{MAJ} and $\alpha_{\oplus}$. After sufficient number

of iterations, the minimum FER is not achieved for noiseless gates but for some nonzero value of the gate error rates α_{MAJ} and $\alpha_{\oplus}$. For a broad range of gate error rates, our decoder actually benefits from logic gate errors.

Increasing the maximum number of iterations, L, reduces the probability that the error pattern remains uncorrected. The impact of L is especially noticeable for high reliability gates (i.e., low α_{MAJ} and $\alpha_{\oplus}$), as it can be seen From Fig. 1(a) and Fig. 1(b). In this case, hardware errors cannot help much in annihilating trapping sets because the state transition probabilities in $\mathcal{W}$ are small for most transitions other than those that already exist in the noiseless decoder. Consequently, the convergence to a codeword takes longer (it also grows with n). On the other hand, increasing Z is has stronger effect when gates are very noisy. Hence, prolonging the second stage of the decoding algorithm greatly improves the performance of a decoder made of the less reliable hardware. For high gate error rates, L has smaller impact. If some FER degradation can be tolerated, the low ratio Z/L can be used in order to improve the decoder energy efficiency in this case. Also, it can be noticed that a noisy XOR gate has approximately $(\rho-1)$ stronger effect on the FER performance than a noisy MAJ gate. In other words $\alpha_{\oplus} \approx (\rho-1)\alpha_{MAJ}$ would result in approximately the same performance, as shown earlier. Therefore, the impact of the both types of failures can be represented by using a single parameter denoted as α_G.

It is important to notice that the results in (Fig. 1) are given for an error pattern that is uncorrectable by the noiseless decoder. Therefore, for this particular error pattern, the noisy decoder works better than the noiseless one for any gate failure rates in the range $\alpha_{MAJ} \leq 0.02$ and $\alpha_{\oplus} \leq 0.1$, and for any L and Z. Optimization of these parameters can lead to further improvements in decoder performance and energy efficiency.

IV. Average FER Performance

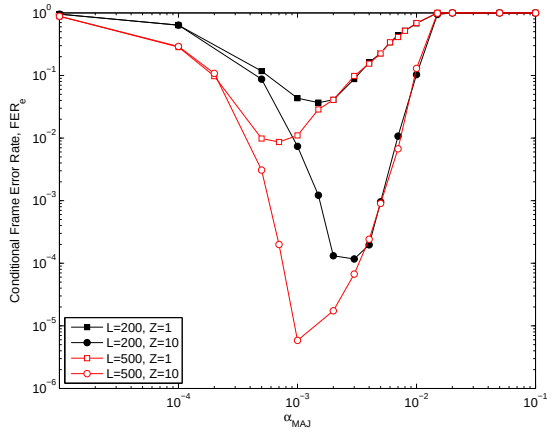
In Section III we have shown the performance of Gallager-B decoder for a critical error pattern. By averaging over all error patterns, we obtain the average FER as

$$FER(\mathcal{D}) = \sum_{\mathbf{e}\in\{0,1\}^n} \Pr\{\mathbf{e}\} \times FER_{\mathbf{e}}(\mathcal{D}). \tag{3}$$

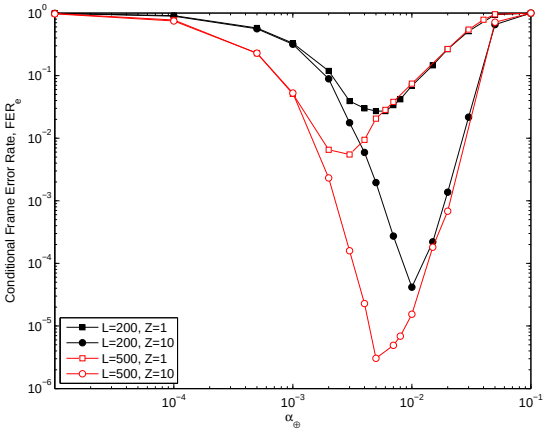
The probability of the error pattern $\mathbf{e}$, $\Pr\{\mathbf{e}\}$, depends on its Hamming weight $w(\mathbf{e})$. In the case of transient i.i.d. memory errors occurring with probability α_M, the probability $\mathbf{e}$, $\Pr\{\mathbf{e}\}$ can be expressed as

$$\Pr\{\mathbf{e}\} = \alpha_M^{w(\mathbf{e})}(1-\alpha_M)^{n-w(\mathbf{e})}. \tag{4}$$

A noiseless decoder for which $\alpha_{MAJ} = 0$ and $\alpha_{\oplus} = 0$, converges to either one of the codeword - which are the fixed points - or to a trapping sets - which can be either fixed points or cycle attractors. Noiseless decoder may oscillate between these states, and fail to converges to a codeword. On the other hand, in a noisy decoder - where α_{MAJ} and/or $\alpha_{\oplus}$ are nonzero - every output message can be produced with a nonzero probability. Thus, the noisy decoder will eventually converge to a codeword - either correct or incorrect one.



(a)



(b)

Fig. 1. The frame error rate performance of noisy Gallager-B decoder for Tanner (155,64) code as a function of error rates in the logic gates, α_{MAJ} and $\alpha_{\oplus}$. The FER curves are estimated by using Monte Carlo simulations. The storage medium introduces the worst case three errors-pattern which induces the $(5,3)$ trapping set. The maximum number of iterations is $L = 200$ and $L = 500$, and the noiseless syndrome checker was used in the last $Z \leq L$ iterations only. Gate errors affect (a) MAJ gates only, (b) XOR gates only.

In the case when the decoding algorithm have small probability of miscorrection in the first decoding iterations, it is better to use the rewinding decoder with $r = L/L_R$ rounds, where the restarts and re-initializations are performed after L_R iterations, $L_R << L$. We denote this decoder by $\mathcal{D}^{\circlearrowleft} = \mathcal{F}^{\circlearrowleft^r}(L_R)$. From Eq. 2, it follows that the rewinding decoder is a composition of r rounds of the non-rewinding decoder $\mathcal{D} = \mathcal{F}(L_R)$ which runs for L_R iterations, with Z iterations of syndrome checking, i.e., $\mathcal{D}^{\circlearrowleft} = \mathcal{D}\diamond\mathcal{D}\diamond\cdots\diamond\mathcal{D}$. This fact allows us to to obtain the miscorrection probabilities for every particular error pattern. The restart is performed only in the case of decoding failure, i.e. when the syndrome is not equal to zero. The miscorrection probability of this decoder is

$$MER_{\mathbf{e}}(\mathcal{D}^{\circlearrowleft}) = \sum_{\ell=1}^{r} MER_{\mathbf{e}}(\mathcal{D})\left(FER_{\mathbf{e}}(\mathcal{D}) - MER_{\mathbf{e}}(\mathcal{D})\right)^{\ell-1}, \tag{5}$$

and the frame error rate after r rewinds includes the cases of miss-corrections in all r rounds rounds and the case when the

syndrome is not zero in all iterations. Therefore,

$$FER_{\mathbf{e}}(\mathcal{D}^{\circlearrowleft}) = MER_{\mathbf{e}}(\mathcal{D}^{\circlearrowleft}) + \left(FER_{\mathbf{e}}(\mathcal{D}) - MER_{\mathbf{e}}(\mathcal{D})\right)^{r}. \tag{6}$$

Finally, the average FER is obtained as

$$FER(\mathcal{D}^{\circlearrowleft}) = \sum_{\mathbf{e}} \Pr\{\mathbf{e}\} FER_{\mathbf{e}}(\mathcal{D}^{\circlearrowleft}). \tag{7}$$

Note that in the above discussion it is assumed that the noiseless syndrome checker was used in every iteration. However, due to power consumption reasons, the noiseless checker might be used in only last Z out of L iterations.

While the above expressions for FER and MER completely determine the decoder's performance, the analysis on an entire code graph is numerically inefficient and can be done only for very short codes. However, the theory of noiseless iterative decoders, gives the code graph topologies - known as trapping sets - responsible for decoding failures. The exact characterization of the relationship between the FER of an LDPC code and its trapping sets is known to be a formidable task. Thus, instead of considering the entire code graph, the decoder performance can be estimated by its ability to escape from Markov chain states corresponding to dominant trapping sets, as we explain next.

V. Numerical Results

So far we considered the effects of the worst case error pattern. If the storage medium is modeled as BSC, any error pattern can occur at the decoder input with certain probability. Therefore, we estimate the performance of noisy Gallager-B decoder is this case as well. For the case when both XOR and MAJ gates are noisy with $\alpha_{\oplus}$=α_{MAJ}, and the numerical results are presented in Fig. 2(a) for the case when $\alpha_M = 2 \times 10^{-3}$. Since the previously considered low-weight pattern is most critical, the main effects are same as in Fig. 1. For the noiseless decoder, the average FER is approximately equal to the probability of appearance of the dominant trapping set at the decoder input. In a noisy decoder, the lowest FER is achieved for the gate error rates that maximize successful correction of most critical trapping sets.

The noisy Gallager-B decoder is more efficient than its noiseless counterpart for any values of the failure rates in the logic gates less than 10^{-2}. Even more importantly, when $L = 1000$ and the gate error rates have near-optimal values, the noisy hard-decision algorithm has better performance than the much stronger and more complex soft-decision min-sum algorithm realized in noiseless hardware. For low gate failure rates, L has the dominant effect on the FER, especially for the lower values of the failure rates as shown in Fig. 2(a). The second stage of the algorithm is crucial, because for small Z even very large L does not improve the performance (see e.g., the $L = 1000$, $Z = 1$ curve). This is the direct consequence of the fact that in this case the absorbing states are reached in only a small number of iterations when the syndrome checker is turned on, while in most of the iterations only transient states are visited.

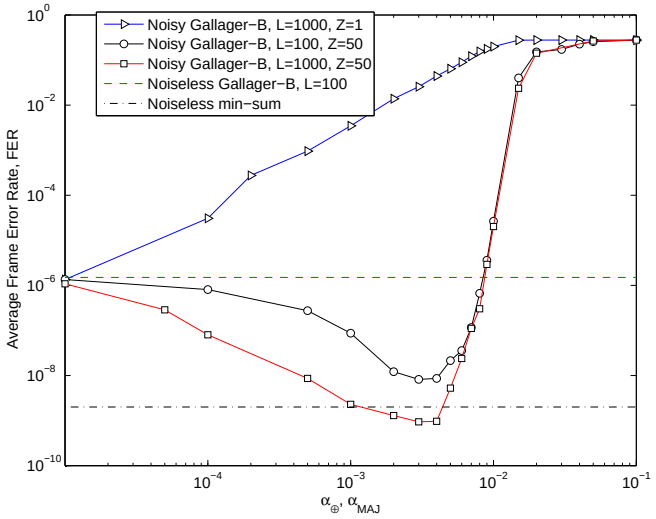
In Fig. 2(b), the average FER as a function of number of iterations is presented for $\alpha_M = 5 \times 10^{-3}$. If the rewinding is not applied, larger values of the failure rates result in performance improvement when compared to the noiseless decoder, but the increase of maximum number of iterations does not result in further decrease of FER. On the other hand, for smaller failure rates the significant performance improvement is noticeable for large values of parameter L. However, if the maximum number of decoding iterations is limited, the performance can be inferior when compared to the case of higher failure rates. If the rewinding is applied, the positive effect of the logic gate failures is exploited several times for the various reinitializations. It is reasonable to chose the rewinding period L_R to be equal to the number of iterations where the performance improvement saturates in the case without rewinding.

A comparison of different decoding strategies suitable for logic gates with high or low reliability is shown in Fig. 3. The FER curves for decoding with no rewinding are shown in Fig. 3(a). For all L, the decoder $\mathcal{F}$ outperforms the ideal decoder $\overline{\mathcal{F}}$, and for large L its performance approaches the ideal decoder capable of correcting any combination of nine errors. The positive effect of the rewinding is shown in Fig. 3(b) for various choices of $L = r \times L_R$ combinations and various maximal number of iterations L. For $\alpha_G = 10^{-2}$, the rewind decoder $\mathcal{F}^{\circlearrowleft}$ performs beyond the $\lfloor \frac{d_{min}-1}{2} \rfloor$ bound. The total number of iterations $L = r \times L_R$ is kept the same as for the corresponding decoder $\mathcal{F}$.

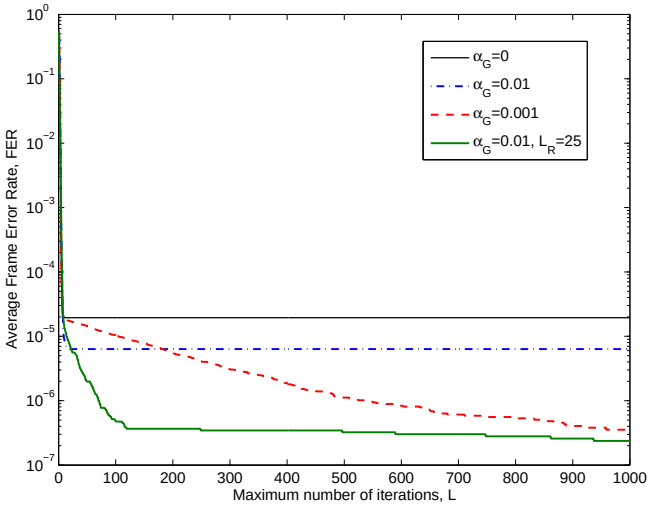
To conclude, the above two decoding strategies cover two gate error rate regimes. For more reliable logic gates, large number of iterations is needed before the decoder start benefiting from the positive effects of logic gate errors. In this case, we apply syndrome checker in the first twenty iterations, as the average FER rapidly decreases only at the beginning of the decoding. Then, we turn-off both the final bit-estimation circuit as well as the syndrome checker, and allow for sufficient number of iterations before we again activate the noiseless syndrome checking. Clearly, this strategy results in energy saving as the noiseless gates are used in a reduced number of iterations. When gate failures rate are high, errors correctable by the noiseless decoder may turn uncorrectable, or lead to miscorrections as they may lead to large deviations from the trajectory of the noiseless decoder. A solution for this case is to rewind the decoder. The higher the gate error rate, the lower optimal rewind period L_R.

Performance of the GDBF decoder with noisy XOR gates is given in Fig. 4(a). It can be observed that noisy decoder have better performance than its noiseless counterpart for a certain range of failure rates in XOR gates $\alpha_{\oplus}$.

In Fig. 4(b) we present the performance of the probabilistic GDBF decoder, where the randomness in inserted intentionally at the output of MAJ gates. The numerical results are given for r rewind decoding rounds, per L_R iterations each. Additionally, we assume that an additional decrementation of the threshold in MAJ gate is applied periodically after n iterations. It can be observed that PGDBF have the higher slope of FER

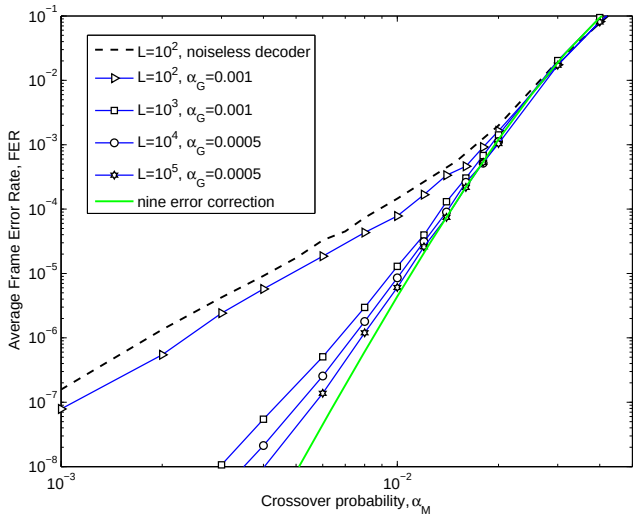


(a)

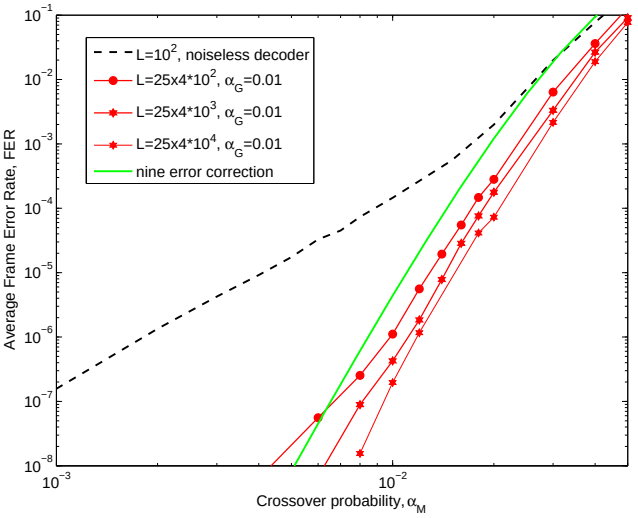


(b)

Fig. 2. (a) Performance of the noisy Gallager-B decoder for Tanner (155,64) code as a function of failure rates (bot XOR and MAJ gates are noisy with the same failure rate); comparison with the noiseless Gallager-B decoder and noiseless min-sum decoder, errors in memory are i.i.d. with $\alpha_M = 2\times10^{-3}$. (b) The FER performance of the noisy Gallager-B decoder on the Tanner (155,64) code as a function of number of iterations. The decoders with and without rewinding are considered and the effect of the per-round number of iterations L_R is illustrated for $\alpha_M = 2 \times 10^{-3}$.



(a)



(b)

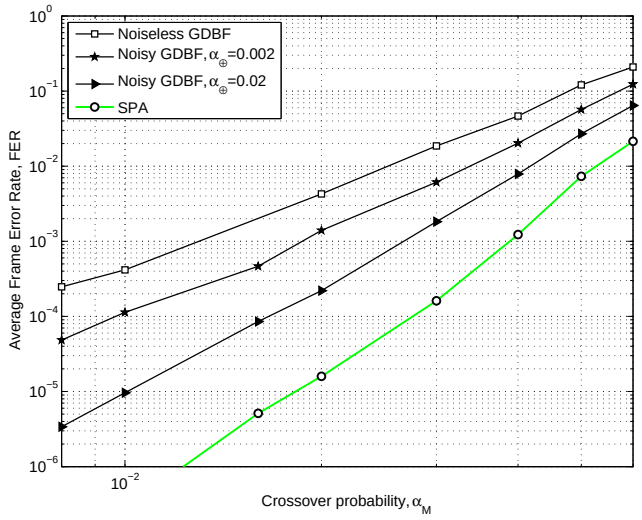
Fig. 3. Performance of the noisy Gallager-B decoder for Tanner (155,64) code as a function of the BSC crossover probability α_M for various decoding strategies (no rewind decoding) (a) and rewind decoding (b)), effective for low and high failure rate ranges.

curve in the error floor region when compared to SPA decoder, both without rewinding and for the same maximum number of iterations ($L = 100$). If the number of iterations is increased, the decoder performs better. When $n = 5000$ and $L_R = 4000$ the ML bound is reached, and during the simulations only correct decoding or misscorrections appear in this case.

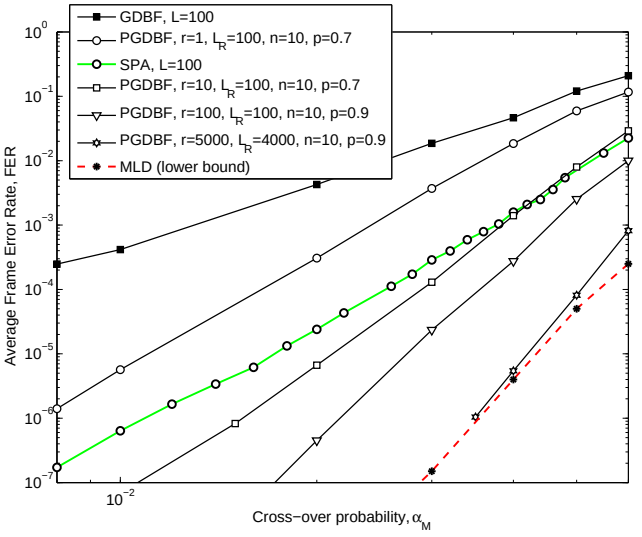
Discussion

The noisy decoder proposed in this paper is a rare example of a system built from a mixture of noisy and noiseless components that works better than a noiseless system of the same or even higher complexity. The exact energy consumption analysis would require hardware implementation and is beyond the scope of this paper, but the fact that our decoders use the noiseless syndrome computations Z/L fraction of time, implies overall energy savings compared to completely noiseless decoders.

This concept can be also used in a different scenario when errors are deliberately inserted in order to improve performance. The errors can be in a form of random bit flips added to logic gate outputs. Generating random bits using noisy hardware is a known concept in the very large scale integration (VLSI) community. The so called "true" random number generators (TRNG) of negligible complexity compared to pseudo random counterparts based on linear feedback shift registers, may be realized using variety of fundamental noise mechanisms in electronics circuits [31].



(a)



(b)

Fig. 4. Performance of two types of the GDBF decoders for Tanner (155,64) code as a function of the BSC crossover probability α_M: Noisy GDBF decoder for $L = 100$ iterations (no rewind decoding) (a) and Probabilistic GDBF decoder for various maximum number of iterations (with rewinding) (b)).

ACKNOWLEDGEMENT

This work was supported by the Seventh Framework Program of the European Union, under Grant Agreement number 309129 (i-RISC project), French ANR project NAND under grant agreement ANR-15CE25-0006-01, and in part by the NSF under Grants CCF-0963726, CCF-1314147 and ECCS-1500170. Bane Vasic acknowledges generous support of the Fulbright Scholar Program. Long version of this paper is submitted for publication to IEEE Transactions on Communications, and part of the results will be submitted to 2016 IEEE International Symposium on Information Theory.

REFERENCES

[1] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasić, "Finite alphabet iterative decoders, Part I: Decoding beyond belief propagation on the binary symmetric channel," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4033–4045, Nov. 2013.

[2] D. Declercq, E. Li, B. Vasić, and S. Planjery, "Approaching maximum likelihood decoding of finite length LDPC codes via FAID diversity," in *IEEE Information Theory Workshop*, Lausanne, Switzerland, Sep. 3–7 2012, pp. 487–491.

[3] D. Declercq, B. Vasić, S. K. Planjery, and E. Li, "Finite alphabet iterative decoders approaching maximum likelihood performance on the binary symmetric channel," in *Proc. Inform. Theory and Applications Workshop*, San Diego, CA, USA, Feb. 5–10 2012, pp. 23–32.

[4] ——, "Finite alphabet iterative decoders, Part II: Improved guaranteed error correction of LDPC codes via iterative decoder diversity," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4046–4057, Nov. 2013.

[5] B. Vasić and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.

[6] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," *IEEE Communications Letters*, vol. PP, no. 99, pp. 1–4, 2014.

[7] T. Yazdi, S. M. Sadegh, H. Cho, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, no. 5, pp. 1660–1673, May 2013.

[8] S. Brkic, P. Ivanis, and B. Vasić, "Analysis of one-step majority logic decoding under correlated data-dependent gate failures," in *in Proc. IEEE Int. Symp. Inf. Theory (ISIT 2014)*, Honolulu, Hawaii, June 29–July 4 2014, pp. 2599–2603.

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

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

[11] 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.

[12] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding LDPC codes," *IEEE Transactions on Communications*, vol. 58, no. 6, pp. 1610–1614, June 2010.

[13] O.-A. Rasheed, P. Ivanis, and B. Vasić, "Fault-tolerant probabilistic gradient-descent bit flipping decoders," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487 – 1490, September 2014.

[14] L. D. Davis and M. Mitchell, Eds., *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[15] A. Karbasi, A. H. Salavati, A. Shokrollahi, and L. R. Varshney, "Noise-enhanced associative memories," in *Proc. Neural Information Processing Systems (NIPS)*, 2013.

[16] J. V. Neumann, *Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components*, ser. Automata Studies. Princeton: Princeton University Press, 1956, pp. 43–98.

[17] M. Taylor, "Reliable information storage in memories designed from unreliable components," *Bell System Technical Journal*, vol. 47, pp. 2299–2337, 1968.

[18] A. V. Kuznetsov, "Information storage in a memory assembled from unreliable components," *Problems of Information Transmission*, vol. 9, no. 3, pp. 254–264, 1973.

[19] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: M.I.T. Press, 1963.

[20] B. Vasić, S. K. Chilappagari, S. Sankaranarayanan, and R. Radhakrishnan, "Failures of the Gallager B decoder: analysis and applications," in *Proc. 2nd Information Theory and Applications Workshop*. University of California at San Diego, Feb. 2006. [Online]. Available:

[21] P. Ivanis, O.-A. Rasheed, and B. Vasić, "Mudri: A fault-tolerant decoding algorithm," in *IEEE International Conference on Communications (ICC 2015)*, London, UK, June 8–12 2015, pp. 4291– 4296.

[22] D. Burshtein and G. Miller, "Expander graph arguments for message-passing algorithms," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 782–790, Feb. 2001.

[23] S. K. Chilappagari and B. Vasić, "Fault tolerant memories based on expander graphs," in *Proc. IEEE Information Theory Workshop (ITW '07)*, Lake Tahoe, CA, Sept. 2007, pp. 126–131.

[24] B. Vasić, D. Nguyen, and S. K. Chilappagari, *Chapter 6 - Failures and*

Error Floors of Iterative Decoders. Oxford: Academic Press, 2014, pp. 299 – 341.

[25] B. Vasić, S. Chilappagari, D. Nguyen, and S. Planjery, "Trapping set ontology," in *Proc. 47th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, Sep. 30–Oct. 2 2009, pp. 1–7.

[26] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, pp. 2282–2312, July 2005.

[27] J. M. Mooij and H. J. Kappen, "Sufficient conditions for convergence of the sum–product algorithm," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4422–4437, Dec. 2007.

[28] V. Chernyak, M. Chertkov, M. G. Stepanov, and B. Vasić, "Error correction on a tree: an instanton approach," *Physics Review Letter*, vol. 93, no. 19, pp. 198 702–198 705, Nov. 2004.

[29] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. ISTA*, 2001.

[30] S. Chilappagari and B. Vasić, "Error-correction capability of column-weight-three LDPC codes," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2055–2061, May 2009.

[31] C. S. Petrie and J. A. Connelly, "A noise-based ic random number generator for applications in cryptography," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 615–621, May 2000.