Fault-Resilient Decoders and Memories made of Unreliable Components

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

Predrag Ivanis, Srdan Brkic
School of Electrical Engineering,
University of Belgrade
Emails: predrag.ivanis@etf.rs, srdjan.brkic@ic.etf.rs

Vida Ravanmehr
Department of ECE,
University of Arizona
Email: vravanmehr@ece.arizona.edu

***Abstract*—In this paper we present our recent results on iterative Gallager B decoder made of unreliable logic gates. We show evidence that probabilistic behavior of a decoder due to unreliable components can be exploited to our advantage and lead to an improved performance and reduced hardware redundancy. We provide examples of such decoder behavior and give an explanation of this phenomenon using iterative decoding dynamics. Iterative decoding can be viewed as a recursive procedure for Bethe free energy function minimization, and the randomness in a message update may help the decoder to escape from local minima. The decoder operates in a stochastic fashion, but the random perturbations do not require any additional hardware as they are built-in the faulty hardware itself.**

I. Introduction

Due to the increase in density integration, lower supply voltages, and variations in technological process, complementary metal-oxide-semiconductor (CMOS) and emerging nanoelectronic devices are inherently unreliable. Moreover, the demands for energy efficiency require that in current CMOS design the energy consumption must be reduced by several orders of magnitude, which can be done only by an aggressive scaling of supply voltage. Consequently, the signal levels are much lower and closer to the noise level, which drastically reduces the component noise immunity and leads to unreliable behavior. It is widely accepted that future generations of circuits and systems must be designed to deal with such unreliable components.

The main feature of the existing work on fault-tolerant decoders is a focus on the analysis of the existing decoder types and demonstrating their robustness to unreliability of logic gates [1]–[6]. This was also an underlying idea of our prior work [7], [8]. On the contrary, the idea of the this paper is to allow or deliberately introduce randomness in a decoder in order to improve convergence in the spirit of stochastic approximation method [9], [10].

The first trace of this idea 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) [11]. A closely related technique of adding noise to messages in a BP decoder on the AWGN channel is by Leduc-Primeau *et al.* [12] for reducing error floor in the context of perfect decoders. Other schemes such as stochastic decoding also uses randomness but in a very different way - by representing messages as random sequences [13]–[16]. Randomness in message update schedule [17], is also observed to yield improved convergence.

Recently it was shown by Sundararajan *et al.* [18] that random perturbations can be used to increase the performance of a gradient descent bit flipping decoder (GDBF), introduced by Wadayama *et al.* [19]. 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 [20] for the Binary Symmetric Channel (BSC). Our decoder incorporates the idea of GDBF with Miladinovic and Fossorier's probabilistic approach [11], but it contains some critical novelties which consists in modifying the inverse function [19, Eqn. (6)]. We showed that random perturbations applied to variable nodes lead to escaping from a local maximum of the GDBF objective function. In addition, the perturbations make the decoder inherently tolerant to hardware unreliability. Our most recent work contains the further improvement of the PGDBF algorithm based on multiple decoding attempts and random re-initializations (MUDRI) of decoders [21].

In this paper we consider the Gallager B decoder in which operations are subjected to processing noise. We show how the hardware unreliability can be used to increase error-correction capability of certain quasi-cyclic low-density parity-check (QC-LDPC) codes, in the error-floor region. Our conclusion relies on the fact that logic gates failures can break small trapping sets, which are the main cause of the decoder failures in that region. Vasic and Chilappagari [22] observed that the faulty Gallager B decoder is equivalent to the Taylor memory architecture [23], which indicates that our results can be directly applied to the reliability analysis of memories built from unreliable components.

The rest of the paper is organized as follows. In Section II we give a brief description of the faulty Gallager B decoder. In Section III the idea of breaking trapping sets is discussed. Section IV is dedicated to the numerical results. Finally, some concluding remarks and future research directions are given in Section V.

II. The Faulty Gallager B Decoder

Consider a (γ,ρ)-regular binary LDPC code, denoted by (N,K), with code rate $R = K/N \geq 1-\gamma/\rho$ and parity check matrix $\mathbf{H}$. The parity check matrix is the bi-adjacency matrix of a bipartite (Tanner) graph $G=(V\cup C,E)$, where V represents the set of N variable nodes, C is the set of $N\gamma/\rho$ check nodes, and E is the set of $N\gamma$ edges. Each matrix element $\mathbf{H}_{c,v}=1$ indicates that there is an edge between nodes $c\in C$ and $v\in V$, which are referred as *neighbors*. Let $\mathcal{N}_v$ ($\mathcal{N}_c$) be the set of neighbors of the variable node v (check node c). Then, $|\mathcal{N}_v|=\gamma$, $\forall v\in V$ and $|\mathcal{N}_c|=\rho$, $\forall c\in C$, where $|\cdot|$ denotes cardinality.

Let $\mathbf{x}=(x_1,x_2,\ldots,x_N)$ denote a codeword of an LDPC code, where x_v represents the binary value associated with the variable node v. During the transmission over a Binary Symmetric Channel (BSC), the code bits are flipped with probability α and received as $\mathbf{y}=(y_1,y_2,\ldots,y_N)$.

The Gallager B decoder works by sending binary messages over the edges of the graph. The messages are calculated based on the *nodes update functions*, following the rule that a message sent over an edge is obtained based on all received messages except the one arriving over that edge [24]. The check node update function corresponds to the $(\rho-1)$-input XOR logic gate, and $(\gamma-1)$-input majority logic (MAJ) gate is used for the variable node update function implementation.

Due to hardware unreliability the results of the update functions are not always correctly computed. We adopted the "wire" model described in [1], where an edge is modeled as a BSC with a known crossover probability. Let $\nu_{v,c}^{(\ell)}$ be the message sent by the variable node v to its neighbor $c\in\mathcal{N}_v$, at the iteration ℓ, and let $\hat{\nu}_{v,c}^{(\ell)}$ be the message that is actually received by the node c. Then, we define the following relation

$$\hat{\nu}_{v,c}^{(\ell)} = \begin{cases} \nu_{v,c}^{(\ell)} & \text{with probability } 1-\varepsilon_{MAJ}, \\ \nu_{v,c}^{(\ell)}\oplus 1 & \text{with probability } \varepsilon_{MAJ}, \end{cases} \tag{1}$$

where ε_{MAJ} represents the probability of failure of MAJ gates. Similarly, let $\nu_{c,v}^{(\ell)}$ be the message sent by the variable node c to its neighbor $v\in\mathcal{N}_c$, at the iteration ℓ, and let $\hat{\nu}_{c,v}^{(\ell)}$ be the message that is actually received by the node v. Then, we have

$$\hat{\nu}_{c,v}^{(\ell)} = \begin{cases} \nu_{c,v}^{(\ell)} & \text{with probability } 1-\varepsilon_{\oplus}, \\ \nu_{c,v}^{(\ell)}\oplus 1 & \text{with probability } \varepsilon_{\oplus}, \end{cases} \tag{2}$$

where $\varepsilon_{\oplus}$ represents the probability of failure of XOR gates. We next summarize the faulty Gallager B decoder.

- *Variable to check node update*: For each variable node $v\in V$:
  At iteration $\ell=0$: $\nu_{v,c}^{(0)}=y_v$, $\forall c\in\mathcal{N}_v$.
  At iteration $l>0$:

$$\nu_{v,c}^{(\ell)} = \begin{cases} s & \text{if } |\{c'\in\mathcal{N}_v\setminus c : \hat{\nu}_{c',v}^{(\ell-1)}=s\}| > \lfloor\gamma/2\rfloor, \\ y_v & \text{otherwise.} \end{cases} \tag{3}$$

- *Check to variable node update*. For each check node $c\in C$ and $\forall v\in\mathcal{N}_c$, at iteration $l\geq 0$:

$$\nu_{c,v}^{(\ell)} = \bigoplus_{v'\in\mathcal{N}_c\setminus\{v\}} \hat{\nu}_{v',c}^{(\ell)}. \tag{4}$$

The decoding is terminated when all parity-check equations are satisfied or the maximum number of iterations (denoted by L) is reached.

Note that in addition to logic gates needed to calculate messages that are passed on the edges of the bipartite graph the decoder also requires logic gates for the final bit estimations and parity-checks calculation. If we allow these gates to be unreliable, the performance of the decoder would be determined by the failure probabilities of these gates, not by the error control scheme. Thus, it is reasonable to assume that these gates are perfect. Similar assumption was also used in other relevant literature [22], [25], [26].

III. Eliminating The Trapping Sets by Gate Failures

It is well known that the failures of Gallager B decoding caused by the low-weight error patterns are mainly due to the existence of the harmful structures in the Tanner graph of LDPC codes called "trapping sets" [27]. A set of variable nodes T is called an (a,b) trapping set if it contains a variable nodes and the subgraph induced by these variable nodes has b odd degree check nodes. In [28], the most harmful structures of column weight 3 LDPC codes using the Gallager B decoder over the BSC are given.

A faulty (5,3) trapping set in a column weight 3 LDPC code of girth $g=8$ is shown in Fig. 1. In this figure, the circles denote the variable nodes and the squares denote the check nodes. The faulty XOR gates are at the check nodes c_2 and c_3 and the faulty MAJ gates are at the variable nodes v_2, v_4 and v_5 which are shown with black arrows.

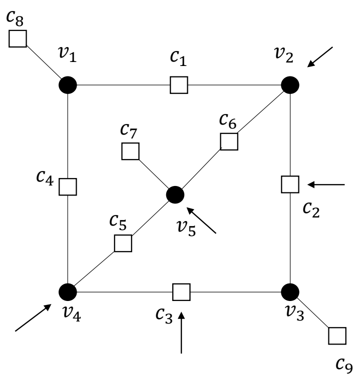

Fig. 1: A faulty (5,3) trapping set.

Logic gate failures as inherent sources of randomness in the existing literature, are usually seen as undesirable. This belief comes from the fact that the capacity of LDPC codes cannot be achieved under unreliable decoding operations [1], [5], [29]. However, the presence of randomness can be beneficial for the finite length codes, as can eliminate small trapping sets.

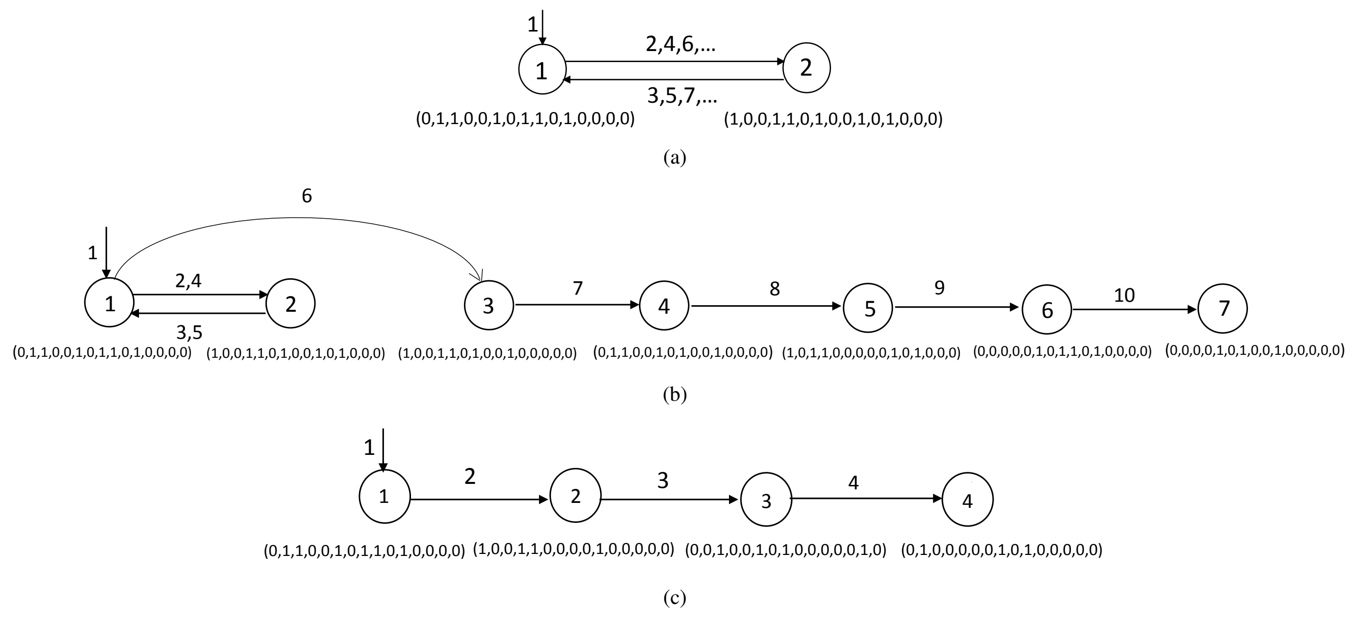

Fig. 2: The state diagram of the Markov chain model on a faulty (5,3) trapping set. (a) $\varepsilon_{\oplus} = 0.001$ and the decoder never converges to a codeword, (b) $\varepsilon_{\oplus} = 0.01$ and the decoder converges to the zero codeword after 10 iterations, (c) $\varepsilon_{\oplus} = 0.1$ and the decoder converges to the zero codeword after 4 iterations.

To illustrate this, we study the dynamic of messages passed from check nodes to variable nodes on a trapping set in each iteration of the faulty Gallager B decoder. For this purpose, we introduce a Markov chain model, which captures the dynamic of the messages.

Let C_T be the set of check nodes in the subgraph induced by the trapping set T. Let define the Markov source S with $2^{|T|\gamma}$ states, where the state i, $1 \leq i \leq 2^{|T|\gamma}$, corresponds to one possible binary sequence of length $|T|\gamma$. Clearly, a sequence $\{\nu_{c,v}{}^{(\ell)}\}_{c \in C_T, v \in T}$, where $\nu_{c,v}^{(\ell)}$ is the message passed from the check node c to the variable node v at time ℓ, maps to one state of the source S. A transition from a state i at time ℓ, to state j at time $\ell + 1$ is obtained by the variable and check node operations, as well as the level of unreliability of logic gates parameters $\varepsilon_{\oplus}$ and ε_{MAJ}.

For a perfect decoder in which $\varepsilon_{MAJ} = 0$ and $\varepsilon_{\oplus} = 0$, the attractor basin has a relatively small number of states. In the state diagram of this model, the states oscillate and the decoder never converges to a valid codeword. While in a faulty decoder where ε_{MAJ} and/or $\varepsilon_{\oplus}$ are sufficiently greater than 0, the number of states is much larger than that of a perfect decoder. That is because that in the faulty decoder, there is a positive probability for each of the possible $2^{|T|\gamma}$ messages to appear as a state in the Markov chain model. Moreover, there is a positive probability of leaving the states of the attractor basin of a perfect decoder. These features of the faulty decoder may cause the decoder to eventually correct the errors and converge to the zero codeword.

To show how the faulty gates can break a trapping set and correct the errors, we consider the (5,3) trapping set and keep track of the messages from the check nodes to variable nodes in each iteration of the faulty Gallager B decoder. For simplicity, we assume that $\varepsilon_{MAJ} = 0$. Thus, the faulty gates are only the XOR gates with the probability of failure $\varepsilon_{\oplus} > 0$. Therefore, the messages from check nodes $\{c_1, c_2, ..., c_9\}$ to variable nodes $\{v_1, v_2, ..., v_5\}$ are flipped with probability $\varepsilon_{\oplus}$. We know that the critical number of the (5,3) trapping set is 3 and the 3 variable nodes that need to be in error are those that are connected to degree-1 check nodes. Recall that the critical number of a trapping set is the minimum number of the erroneous variable nodes that cause failure of decoder [30]. Thus, assume that v_1, v_3 and v_5 are in error; i.e. $v_1 = v_3 = v_5 = 1$. Suppose the decoder stops if the decoder finds a codeword or reaches the maximum number of iterations. The state transitions corresponding to the messages from check nodes to variable nodes for $\varepsilon_{\oplus} = 0.001$, $\varepsilon_{\oplus} = 0.01$ and $\varepsilon_{\oplus} = 0.1$ are shown in Fig. 2.

As can be seen in Fig. 2(a), for $\varepsilon_{\oplus} = 0.001$, the messages oscillate between states 1 and 2. In this case, the decoder never converges to a codeword. We note that when $\varepsilon_{\oplus} = 0$, the dynamic of messages is the same as the dynamic given in Fig. 2(a). Fig. 2(b) shows that the decoder corrects all errors in 10 iterations when $\varepsilon_{\oplus} = 0.01$. As shown in Fig. 2(b), in the first 5 iterations, the messages oscillate between states 1 and 2. However, after the 6th iteration, the states change from the state 3 to 7 in which the decoder stops and corrects all errors. Finally, in Fig. 2(c), the dynamic of messages is shown for $\varepsilon_{\oplus} = 0.1$ that depicts the convergence of the decoder to the zero codeword in 4 iterations.

In the above discussion, we investigated the dynamic of messages from the check nodes to variable nodes in an isolated (5,3) trapping set. To see how the faulty Gallager B decoder performs on a (5,3) trapping set in the (155,64) Tanner code, we put 3 errors on the variable nodes connecting to degree-1

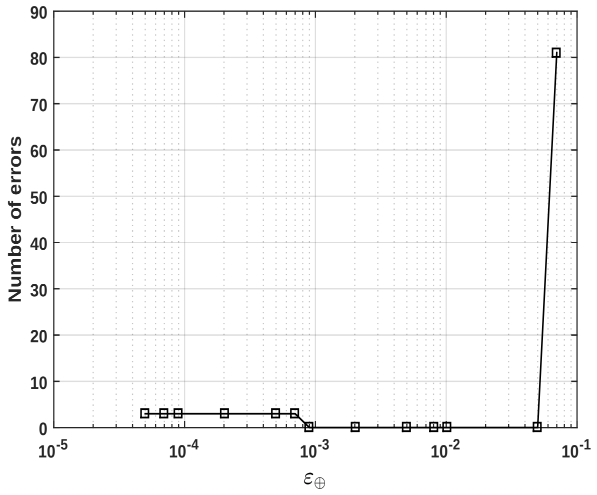

Fig. 3: Number of errors versus $\varepsilon_\oplus$ after running the faulty Gallager B decoder for 100 iterations on the (155,64) Tanner code. The input of the decoder has 3 errors located on the variable nodes that are connected to degree-1 check nodes in a (5,3) trapping set.

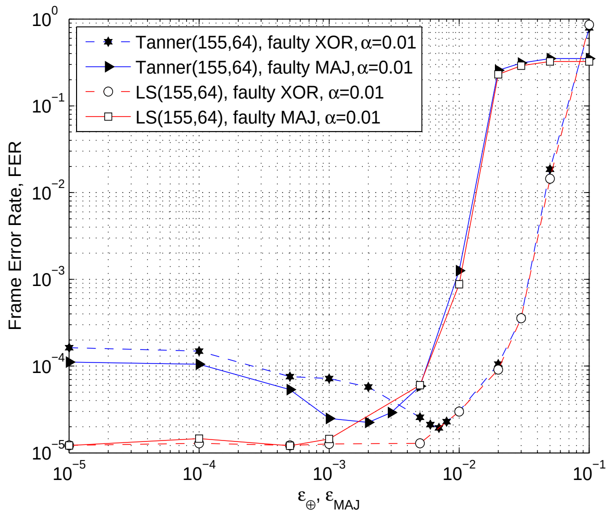

Fig. 4: FER as a function of probability of the gate failure, $\alpha = 0.01$, $L = 100$.

check nodes in a subgraph corresponding to the (5,3) trapping set. Then, for different values of $\varepsilon_\oplus$, we ran the faulty Gallager B decoder for at most 100 iterations and stored the number of variable nodes that are eventually in error. The result is shown in Fig. 3. As we expected, for sufficiently small $\varepsilon_\oplus$, the decoder cannot correct errors located on the trapping set and for large enough values of $\varepsilon_\oplus$, the decoder incorrectly decodes some other variable nodes that were originally correct, to 1. However, there are some values of $\varepsilon_\oplus$ for which the decoder corrects all errors.

Thus, as shown both in Fig. 2 and Fig. 3, for some values of $\varepsilon_\oplus$, the trapping sets responsible for the failure of the Gallager B decoder are broken which in turn may lead to improve the performance of the decoder.

IV. Numerical Results

In this section we further elaborate the idea that hardware unreliability can lead to the performance improvement. We evaluate the frame error rate (FER) and the bit error rate (BER) of various LDPC codes, under i.i.d. failures in logic gates, explained in Section II.

We mostly analyze quasi-cyclic (QC) codes, based on circulant matrices [31] and Margulis codes [32], for which are known to perform poorly in the error floors, as their bipartite graphs contain small trapping sets [33]. We also consider Latin square (LS) based codes that are designed to be free of small trapping sets [34], as a reference for the improvement obtained by breaking the trapping sets in faulty gates. We only investigate (3,5)-regular LDPC codes, with girth $g = 8$, as they allow low decoding complexity, but we expect that similar conclusions can be derived for other regular LDPC codes.

First, we consider how the probability of random i.i.d. failures in XOR/MAJ gates of Gallager B decoder affects the FER of the two codes with the same construction parameters, but different error floor behavior. In Fig. 4 the two different scenarios are illustrated: (i) when check node processing is reliable and only MAJ gates are faulty, and (ii) when variable node processing is reliable and only XOR gates are faulty. The performance of LS(155,64) code (also denoted as C_1 code in [34]), marked with red lines, are approximately constant, up to the certain probability of logic gate failures. It can be noticed that MAJ gates failures are more damaging, as the performance degradation rapidly increase when $\varepsilon_{MAJ} > 10^{-3}$, while, in the case of faulty XOR gates, the similar effect is visible only for $\varepsilon_\oplus > 5 \times 10^{-3}$. On the other hand, the performance of $(155, 64)$ Tanner code, presented with blue lines, is not necessarily degraded with the increase of logic gates unreliability. In fact, there exist $\varepsilon_{MAJ} > 0$, and $\varepsilon_\oplus > 0$, which results in the highest error-correction capability. For that optimal case the performance of the $(155, 64)$ Tanner code approaches closely to the performance of LS(155,64), which are a magnitude higher when decoding is done by the perfect decoder.

This surprising effect is related to the positive impact of hardware failures on breaking of the trapping sets, and it is similar to the effect reported for the gradient descent bit flipping decoder [20]. However, we noticed that the benefit of using unreliable logic gates is related also to the mutual relations of communication channel crossover probability α and $\varepsilon_\oplus$ and ε_{MAJ}, respectively. When channel errors are dominant, $\alpha \gg \varepsilon_\oplus, \varepsilon_{MAJ}$, the influence of hardware unreliability on the decoder performance is limited, and FER stays mostly determined by the smallest trapping sets. On the other hand, if the logic gate failures probabilities are much higher than the channel crossover probability ($\alpha \ll \varepsilon_\oplus, \varepsilon_{MAJ}$), gate failures are no more useful. In such a case, errors added during the decoding prevent the correction of channel errors.

As it can be observed from Fig. 5, the improvement caused by XOR logic gates unreliability is notable for a wide range

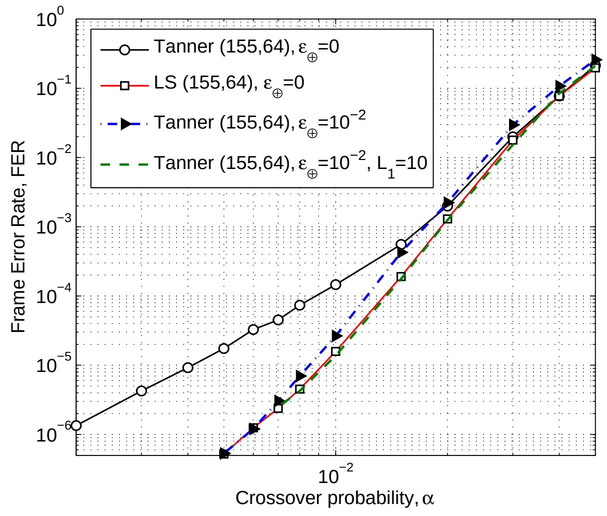

Fig. 5: FER as a function of crossover probability, for the two $(155, 64)$ codes

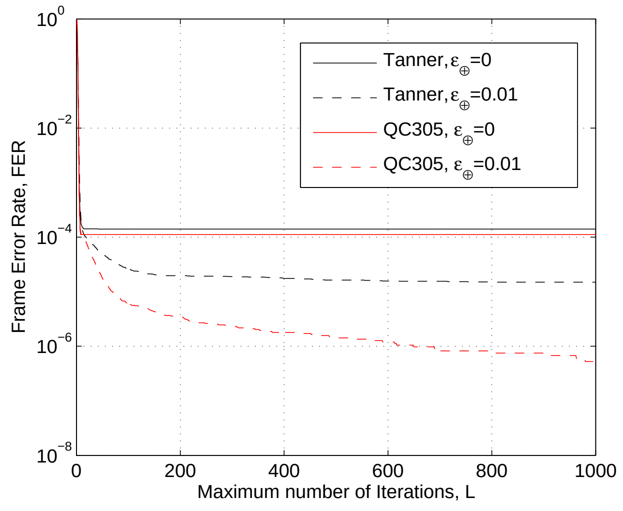

Fig. 6: FER as a function of maximum number of iterations, impact of gate failures, $\alpha = 0.01$.

of failure probabilities, and it is mostly pronounced in the error-floor region, when channel induced errors are rare. For that case the $(155, 64)$ Tanner code performs the same as LS(155,64), when decoded by the perfectly reliable decoder. However, a small degradation is observed for large values of α.

This results indicates that the performance of the Tanner code $(155, 64)$ can be improved even when perfectly reliable decoder is used, if the random binary sequence with $Pr(1) \approx 0.01$ is added in the check nodes, or the corresponding sequence with $Pr(1) \approx 0.002$ is added in variable nodes. For this purpose, stochastic XOR/MAJ gates can be used [35]. The performance of Tanner code can be further improved if no failures are added at the beginning of the decoding process, for instance during the first L_1 decoding iterations. In this case the Gallager B decoder is given time to correct the error pattern, and only if decoding fails after L_1 iterations the probabilism is used. The corresponding numerical results, presented in Fig. 5, reveals that, when L_1 adequately chosen ($L_1 = 10$), the perfectly reliable Gallager B decoder is outperformed for all considered crossover probabilities α. The use of probabilism in order to improve the Gallager B decoding is included in our future research.

We next compare the performance of the $(155, 64)$ Tanner code with the code from the same (3,5)-regular ensemble and approximately same code rate, but with the codeword length $N = 305$ (constructed over field GF(61)) [31]. The FER values of these two codes are illustrated in Fig. 6 as a function of maximal number of iterations L. The results are presented for the crossover probability $\alpha = 0.01$, with and without failures in XOR gates. Although the longer code have better correction capability when Belief-Propagation decoding algorithm is applied [31], decoding by the Gallager B algorithm results in the minor performance improvement, when compared to the $(155, 64)$ Tanner code. For both codes, the decoding saturates after 15 iterations and the performance is not further improved with the increase of parameter L. If hardware failures are present, the performance continues to improve with increase of L, up to the number of iterations that is comparable with the codeword length. If we allow L to be sufficiently large, FER of the code with $N = 155$ can be reduced for one order of magnitude and FER of the code with $N = 305$ can be reduced for the two orders of magnitude.

In Fig. 7, we illustrate XOR gate failures influence to the performance of QC codes with $N = 305$ and $N = 755$ [31]. Similarly, as observed for the $(155, 64)$ Tanner code, the performance of both codes are improved, if the number of decoding iterations is large. The code with $N = 755$ is of a special interest due to its atypical behavior. It is constructed over GF(151) and has the bad distance profile, and therefore has an inferior performance in the error floor region, when compared to the randomly constructed code with the same codeword length and code rate [31]. By using the faulty Gallager B decoder we implicitly randomize decoding process and the performance is improved as the impact of the small trapping sets is minimized. It should be noted that, for the code with $N = 755$, the probability of converging to a codeword different from the transmitted codeword is not negligible. We observed that, for example, for $\alpha = 0.01$ and $L = 1000$, in average half of the decoder failures are the result of that phenomenon.

The general conclusion that for the large codeword length we need larger maximal number of iterations to obtain the full gain of the gates unreliability, especially in the waterfall region, is additionally strengthened by the case of Margulis code $(2640, 1320)$. It can be easily observed from Fig. 8 that decoding convergence continue for more than $L = 1000$ iterations for both analyzed values of the crossover probability ($\alpha = 0.01$ and $\alpha = 0.015$). As the code has good distance properties probability of miss-correction is negligible even for very large values of L. It is clear that the increase of parameter

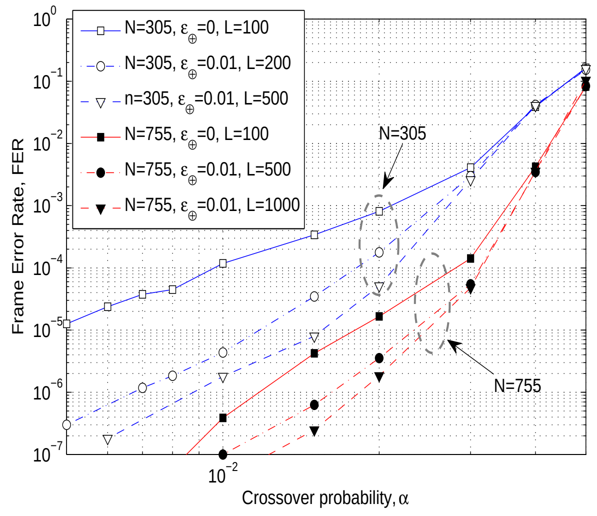

Fig. 7: FER as a function of crossover probability, two codes with $\gamma = 3$, $\rho = 5$.

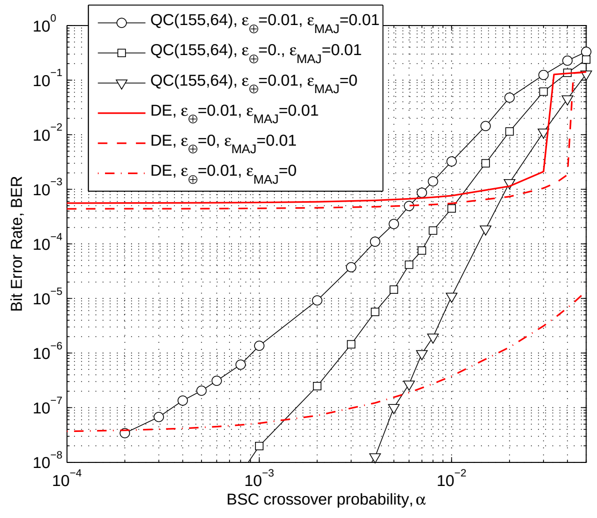

Fig. 9: BER as a function of crossover probability, Monte Carlo simulation for a finite length code and numerical results obtained by using density evolution technique.

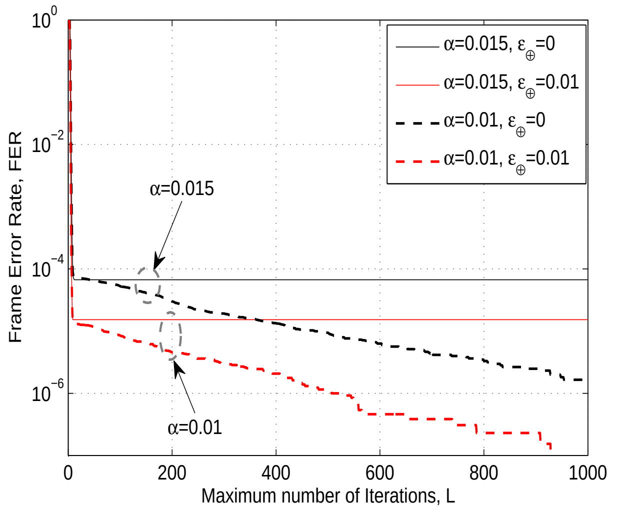

Fig. 8: FER as a function of maximum number of iterations for the $(2640, 1320)$ Margulis code.

L increase the maximum latency of the decoding process and this can be critical for certain applications.

The past work related to the faulty Gallager B decoder was mostly dedicated to the infinite code length analysis by the density evolution (DE) technique [3], [4]. The results obtained using the DE tool are rather pessimistic, as they show that the BER performance converges to a rather high value, especially when decoder is built from faulty MAJ gates. However, in Fig. 9 we show different behavior of finite length codes. For example, it can be observed that the $(155, 64)$ Tanner code, when $L = 100$, outperforms the infinite length code from the same ensemble, in the error floor region. This additionally illustrates the significance of trapping set analysis, for the decoders built from unreliable components.

V. Conclusion

In this paper we showed that uncertainty of the logic gate operations is not always undesirable in the message-passing decoding of LDPC codes. In fact, we observed that the faulty gates may significantly improve the Gallager B decoder performance. Random failures of logic gates result in correction of some error patterns in a faulty decoder that are uncorrectable by the decoder made of reliable components. By analyzing the dynamic of the faulty Gallager B decoder, we found that the improvement is mostly notable with codes that contain small trapping sets. Accordingly, their performance in the error-floor region is highly improved.

Our next step is to try to exploit the observed effects in order to create more powerful low-complexity hard decision decoders. We believe that "controlled" failure injection can lead to obtain even higher performance gain. Our future work also focuses on design of fault-tolerant LDPC code-based memories made from unreliable components.

Acknowledgement

This work was supported by the Seventh Framework Program of the European Union, under Grant Agreement number 309129 (i-RISC project), and in part by the NSF under Grants CCF-0963726 and CCF-1314147. Bane Vasic acknowledges generous support of The United States Department of State Bureau of Educational and Cultural Affairs through the Fulbright Scholar Program.

References

[1] L. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Trans. Inf. Theory*, vol. 57, no. 7, pp. 4427–4444, July 2011.

[2] C.-H. Huang and L. Dolecek, "Analysis of finite-alphabet iterative decoders under processing errors," in *Proc. 2013 IEEE Int. Conf. Acoustics, Speech, Sig. Proc.*, May 2013, pp. 5085–5089.

[3] F. Leduc-Primeau and W. Gross, "Faulty Gallager-B decoding with optimal message repetition," in *Proc. 50th Annual Allerton Conference on Communication, Control, and Computing*, Oct 2012, pp. 549–556.

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

[5] 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.

[6] C. Ngassa, V. Savin, and D. Declercq, "Min-Sum-based decoders running on noisy hardware," in *IEEE Global Communications Conference (GLOBECOM 2013)*, Dec. 2013, pp. 1879–1884.

[7] S. K. Chilappagari, M. Ivkovic, and B. Vasić, "Analysis of one-step majority logic decoders constructed from faulty gates," in *Proceedings of IEEE International Symposium on Information Theory (ISIT '06)*, Seattle, WA, July 2006, pp. 469–473.

[8] 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.

[9] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 09 1951.

[10] W. Gardner, "Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique," *Signal Processing*, vol. 6, no. 2, pp. 113 – 133, 1984.

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

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

[13] V. C. Gaudet and A. C. Rapley, "Iterative decoding using stochastic computation," *IEE Electronic Letters*, vol. 39, no. 3, pp. 299–301, February 6 2003.

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

[15] S. S. Tehrani, S. Mannor, and W. Gross, "Fully parallel stochastic LDPC decoders," *IEEE Transactions on Signal Processing*, vol. 56, no. 11, pp. 5692–5703, November 2008.

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

[17] Y. Mao and A. Banihashemi, "Decoding low-density parity-check codes with probabilistic scheduling," *IEEE. Commun. Letters*, vol. 5, no. 10, pp. 414–416, Oct 2001.

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

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

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

[21] P. Ivanis, O. Al Rasheed, and B. Vasic, "MUDRI: A fault-tolerant decoding algorithm," in *in IEEE Int. Conf. on Commun. (ICC 2015)*, London, June. 2015, p. (paper acccepted).

[22] B. Vasic 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 Trans. on Circuits and Syst. I, Reg. Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.

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

[24] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA, USA: MIT Press, 1963.

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

[26] E. Dupraz, D. Declercq, B. Vasic, and V. Savin, "Finite alphabet iterative decoders robust to faulty hardware: Analysis and selection," in *Proceedings of 8th International Symposioum on Turbo Codes and Iterative Information Processing (ISTC)*, Bremen, Germany, Aug. 2014, pp. 1–10.

[27] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conference on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2003, pp. 1426–1435.

[28] S. Chilappagari, D. Nguyen, and B. Vasić, "Trapping set ontology." [Online]. Available: http://www2.engr.arizona.edu/~vasiclab/Projects/CodingTheory/TrappingSetOntology.html

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

[30] 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.

[31] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. on Inf. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.

[32] G. A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, pp. 71–78, 1982.

[33] D. MacKay and M. Postol, "Weaknesses of margulis and ramanujan-margulis low-density parity-check codes," in *Proc. MFCSIT*, Galway, 2002.

[34] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasic, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012.

[35] K. Mansinghka, E. M. Jonas, and J. B. Tenenbaum, "Stochastic digital circuits for probabilistic inference," *MIT Computer Science and Artificial Intelligence Laboratory, Technical Report MIT-CSAIL-TR-2008-069*, 2008.