MUDRI: A Fault-Tolerant Decoding Algorithm

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

Omran Al Rasheed
School of Electrical Engineering,
University of Belgrade
Email: omrano84@hotmail.com

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

***Abstract*—We propose an improved version of probabilistic gradient descent bit flipping algorithm for decoding low density parity check codes, based on MUltiple Decoding attempts and Random re-Initializations (MUDRI). The proposed algorithm significantly increases the probability of correcting error patterns uncorrectable by the existing variants of bit-flipping algorithm. The performance of the algorithm implemented in noisy hardware is analyzed for various code types and codeword lengths, and shown to be superior compared to other hard decision algorithms. The MUDRI decoder is mostly insensitive to the failures in registers and logic gates and therefore represents a desirable solution for implementation in unreliable hardware.**

I. Introduction

High integration factor of integrated circuits together with low power consumption requirements makes emerging semiconductor devices inherently unreliable [1]. Traditional von Neumann-type triple modular redundancy architectures that ensure fault tolerance are inefficient in handling such increased unreliability thus requiring solutions based on error control coding. In traditional models of computer and communications systems with error correction coding, it is assumed that the operation of a decoder is deterministic and the randomness (in the form of noise and/or errors) exists only in the communication/storage channel elements. While appropriate in systems where the reliability of registers and logic gates used in the decoder is many orders of magnitude higher than the reliability of the channel, this assumption is invalid if digital logic in the decoder is built of faulty components.

Recently there was a surge in research in fault-tolerant decoders. Vasic and Chilappagari [2] established and information theoretical framework for analysis and design of faulty decoders for low-density parity-check (LDPC) codes. They have also analyzed bit-flipping decoding [2] or one-step majority logic (MAJ) decoding [3], [4]. Methods for performance analysis of more complex decoders built from unreliable hardware based on the sum-product algorithm (SPA) [5] and its suboptimal (min-sum algorithm) version [6] have been also developed for transient failure model. In the similar context, finite-alphabet decoders (FAID) were analyzed by Huang and Dolecek in [7]. Density evolution analysis of the simplest massage-passing algorithm (Gallager-B) implemented in noisy hardware is given in [8] and [9].

The bit-flipping (BF) decoder is an attractive candidate for high speed applications when only hard decisions are available at the channel output, but since its performance is typically inferior when compared to the Gallager-B algorithm [9], numerous ingenious improvements of the BF algorithms have been proposed in the literature (see [10] and references therein) with the aim to close this gap. Recently, we proposed an modification of Gradient Descent Bit Flipping (GDBF) [11], appropriate for binary symmetric channel (BSC). The algorithm incorporates the idea of Probabilistic Bit Flipping (PBF) [12] in GDBF, with some additional improvements. The resulting algorithm, that we named Probabilistic Gradient Descent Bit Flipping (PGDBF) algorithm, was shown to be resilient to logic gate failures [13].

A probabilistic analysis of the PGDBF performed in this paper reveals that the PGDBF is not capable of correcting some low-weight error patterns. Therefore, we propose a modification of the algorithm, based on the principle of MUltiple Decoding attempts and Random re-Initializations (MUDRI) of decoders. A similar approach has resulted in improved performance of non-faulty (perfect) FAID [14]. The MUDRI decoder modification combined with new threshold adaptation method results in significant performance improvement and high level of immunity to the failures in registers and logic gates. We also demonstrate the algorithm's ability to control logic gate failures on the various code types - quasi-cyclic (QC), progressive edge growth (PEG) and Latin squares based (LS), with different column weights and codeword lengths.

The rest of the paper is organized as follows. Section II gives the necessary background. In Section III we present the MUDRI decoder. Section IV gives the performance analysis in the presence of hardware failures and comparison with the other decoding algorithms, and Section V concludes the paper.

II. Preliminaries

Let $\mathcal{C}$ denote an (N, K) binary LDPC code with rate $R = K/N$, defined by the null space of H, an $M \times N$ parity check matrix. Tanner graph representation of $\mathcal{C}$, denoted by G, consists of the set of variable nodes $V = \{v_1, v_2, ..., v_N\}$ and the set of check nodes $C = \{c_1, c_2, ..., c_M\}$. Two nodes are neighbors if there is an edge between them. A code represented by the graph G is said to be have a regular column-weight γ if all variable nodes in V have the same number of neighbors γ. The ρ-regular check regular code is defined analogously.

The set of neighbors of a variable and check nodes is denoted as $\mathcal{N}_v$ and $\mathcal{N}_c$, respectively. Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ denote a codeword of $\mathcal{C}$, where x_v denotes the value of the bit associated with variable node v. The effect of the BSC with crossover probability α is modeled by an N-dimensional

binary random variable with independent coordinates E_v, such that $\Pr(E_v = 1) = \alpha, v = 1, 2, ..., N$ where e_v is realization of E_v. The vector received by a decoder is $\mathbf{y} = (y_1, y_2, \ldots, y_N)$, where $y_v = x_v \oplus e_v$, and $\oplus$ is the modulo-two sum. We shall refer to variable nodes initially in error as *erroneous nodes* and variable nodes initially correct as *correct nodes*.

We consider iterative decoders which at the l-th iteration ($l \in [0, L]$, where L is maximal number of iterations) produce the estimate $\hat{\mathbf{x}}^{(l)}$ as an output. The GDBF algorithm for all variable nodes v calculates the inverse function [11, Eqn. (6)]

$$\Delta_v^{(l)}(\chi, \eta) = \chi_v^{(l)}\eta_v + \sum_{c \in \mathcal{N}_v} \prod_{u \in \mathcal{N}_c} \chi_u^{(l)}, \tag{1}$$

where $\chi_v^{(l)} = 1 - 2\hat{x}_v^{(l)}$ and $\eta_v = 1 - 2y_v$ denote the "bipolar" versions of $\hat{x}_v^{(l)}$ and y_v. The estimate of a variable node v is initialized as $\chi_v^{(0)} = \eta_v$, and in the l-th iteration only the symbols with minimum value of the inverse functions are inverted to obtain $\chi_v^{(l+1)}$.

In [13], we shown that the GDBF algorithm can be adapted to the BSC. By using modulo-2 arithmetic, we have shown that the inverse function for the case of regular column-weight γ codes can be minimized by maximization of the following modified inverse function (MIF) [13]

$$\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = \hat{x}_v^{(l)} \oplus y_v + \sum_{c \in \mathcal{N}_v} \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(l)}. \tag{2}$$

In the PGDBF algorithm, the refreshed estimation in the l-th iteration is calculated as [13]

$$\hat{x}_v^{(l+1)} = \begin{cases} a_v \oplus \hat{x}_v^{(l)}, \Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = b^{(l)}, \\ \hat{x}_v^{(l)}, \Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) < b^{(l)}. \end{cases} \tag{3}$$

where $b^{(l)}$ denotes the largest value of the MIF at the l-th iteration, i.e., $b^{(l)} = \max_v(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$, and a_v denotes a realization of Bernoulli $B(1, p)$ random variable. The parameter p introduces a randomness in the flipping process, and if $p = 1$, PGDBF corresponds to deterministic GDBF for BSC channel.

In hardware, the calculation of $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})$ requires: (i) γ ρ-input exclusive or (XOR) gates which compute the parities in the neighboring check nodes, (ii) one two-input XOR gate to check if the v-th bit of the current estimate is the same as the bit initially estimated from the channel, and (iii) one $(\gamma + 1)$-input MAJ gate with adaptable threshold. As it is shown in [13], the calculation of the threshold $b^{(l)}$ can be realized without a global operation of integer maximizations. $b^{(l)}$ is initialized to the maximum possible value ($b^{(l),init} = \gamma + 1$), and decremented every time when all the MAJ gate outputs are zero. In this decrementation procedure, the first occurrence of a non-zero MAJ gate output indicates that the threshold reached the maximal MIF value. After that, a change of at least one MAJ gate output with respect to its previous value indicates that $\max_v{}_2(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$, the second MIF maximum is reached.

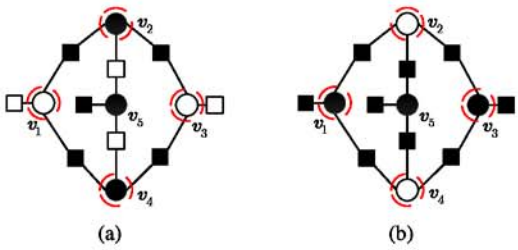

Fig. 1. a) A three-bit pattern, b) the pattern in the second iteration of GDBF.

III. The MUDRI Algorithm

In this section, we propose a modification of PGDBF algorithm presented in [13]. We begin with three illustrative examples, exhibiting a method used to analyze probabilistic algorithms and presenting the intuition behind our decoder.

A. Example 1

Due to their very nature, probabilistic BF algorithms render inapplicable the trapping set analysis method developed for their deterministic counterparts [10]. In order to analyze the correctability of low-weight error patterns, let us consider a a three-bit error pattern shown in Fig. 1(a), where white (black) circles represent the correct (erroneous) variable nodes, and the (black) white squares denote (un)satisfied checks. If the GDBF algorithm is applied (for which $p = 1$), the largrest MIF value $b^{(1)} = 2$ is associated with variable nodes v_1, v_2, v_3 and v_4 in the first iteration. These variable nodes are dashed-circled. In the next iteration, the MIF value $b^{(1)} = 4$ is associated with the same variable nodes, as presented in Fig. 1(b). Note that these nodes have different value compared to the initial values. As it results in the fixed set [15], this error pattern cannot be corrected by the GDBF algorithm.

On the other hand, if the PGDBF is applied, the four bits with the largest MIF are not flipped automatically but are only the *candidates for flipping*. It can be shown that there is only one *flipping sequence* that results in a successful decoding after exactly two iterations. Denote by s_l the probability that a given error pattern is successfully decoded in the l-th iteration. In our example, probability of the flipping sequence $f = (f_1, f_2) = ((0, 1, 0, 1), (1))$ is $s_2 = p^3(1 - p)^2$.

It is clear that $s_1 = 0$, as this pattern cannot be corrected in the first iteration. Note that other flipping choices resulting in different flipping sequences might lead to the successful decoding but, possibly, in a larger number of iterations. We refer to such flipping sequences as suboptimal. In our case this number of iterations is $l > 2$. As there may be many suboptimal flipping sequences, the closed form expression for s_l is complicated. However, its numerical value can be easily estimated by using Monte Carlo simulation. The probability of unsuccessful decoding at the L iteration is obtained as

$$p_{PGDBF}(L) = 1 - \sum_{l=1}^{L} s_l. \tag{4}$$

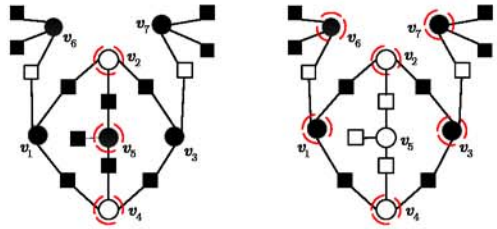

Fig. 2. a) A five-bit error pattern uncorrectable by using GDBF, b) The second iteration of PGDBF, after the first iteration with the optimal choice.

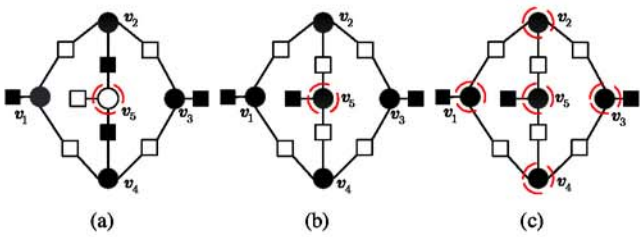

Fig. 3. a) A four-bit pattern critical in PGDBF, b) the second iteration if $b^{(l)} = \max_v(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$, c) the second iteration if $b^{(l)} = \max_v(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})) - 1$.

B. Example 2

As the PGDBF algorithm is probabilistic in nature, successful decoding of certain types of errors cannot be guaranteed as it is possible for deterministic algorithms (e.g. algorithm in [10] correct all triple errors for some codes). By using Eq. (4) we are able to estimate $p_{PGDBF}(L)$ for any error pattern, and the general conclusion is that it can be reduced by increasing parameter L. However, there are some error patterns which have high values of $p_{PGDBF}(L)$ even for high values of L, and one such error pattern is shown in Figure 2(a).

In the first iteration, $b^{(1)} = 3$ is associated with the variable nodes v_2, v_4 and v_5. The PGDBF update rule allows an independent flipping of all these variables (2^3 possible choices), but only some of them are actually flipped. If only v_5 is flipped (with the probability $p(1-p)^2$), the error pattern at the beginning of the second iteration looks like the one shown in Fig. 3(b). In this case, $b^{(2)} = 2$ and six bits are considered for flipping, with 2^6 possibilities for the flipping choices in this step. If only the bits that are incorrectly received are chosen for flipping (v_1, v_3, v_6 and v_7), with the probability $p^4(1-p)^2$, the decoding process is successfully completed. As only one flipping sequence results in decoding after two iterations, the corresponding probability is obtained by multiplying the probabilities in two successive steps as $s_2 = p^5(1-p)^4$.

However, if a wrong choices are made in a few iterations at the beginning of decoding, it does not have to be completed successfully even for very large value of L. Therefore, we propose the modification of the algorithm. If the syndrome has non-zero value after L_1 iterations, the decoding is stopped and repeated $\lfloor L/L_1 \rfloor$ times starting from the received word for the other flipping random choices. If the random sequences are independent, the probability that the decoding fails is

$$p_{MUDRI}(L, L_1) = \left(1 - \sum_{l=1}^{L_1} s_l\right)^{\lfloor L/L_1 \rfloor}. \quad (5)$$

In the special case when $L_1 = L$, we have a single attempt with L iterations, and the above expression reduces to Eq. (4). The probability of unsuccessful decoding can be minimized with the proper choice of this parameter L_1.

C. Example 3

Finally, we show that the four bit error pattern presented in Fig. 3(a) is uncorrectable by the PGDBF, and propose the appropriate modification. In the first iteration, only v_5 has two unsatisfied checks and it has to be flipped. In the next step (Fig. 3(b)) there are three variable nodes with one unsatisfied check, but only v_5 has the value different from the value initially received from the channel. As the same bit has the maximal MIF value in two successive iterations, and as failing to flip cannot help when there are only one candidate, we conclude that $p_{PGDBF}(L) = 1$ for any L.

In such a situation we propose decrementing the threshold in variable nodes until it reaches the second largest value, i.e. $b_{mod}^{(l)} = \max_2(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$. In our example $b_{mod}^{(2)} = 1$, the nodes with $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})) \geq 1$ are flipped and the decoding is successful after the second iteration (Fig. 3(c)).

The above modifications are combined with the PGDBF algorithm [13] to obtain the MUDRI decoding algorithm, formally given in Algorithm 1. The modification related to the threshold adaptation (explained in Example 3) is implemented as a separate function FUN, where $in^{(l)}$ denotes the number of variable nodes that should be flipped in GDBF in the l-th iteration. The modification is applied under the condition that $in^{(l)} = in^{(l-1)} = 1$ and that in two successive iterations the maximal MIF value corresponds to the same bit in the codeword (denoted by $v_f^{(l)}$).

IV. Analysis and Numerical Results

In this section, the impact of the parameters in the MUDRI is considered, and the corresponding numerical results are presented. Then, the performance of the algorithm implemented in the faulty hardware is presented to illustrate its robustness to the logic gate failures.

A. Analysis of the MUDRI algorithm

To evaluate the algorithm performance, we first consider the decoding of the error patterns presented in the motivating examples, illustrated in Figures 1(a) and 2(a), for the case when these patterns appear in the Tanner (155,64) code. The probability of successful decoding at exactly l iterations is estimated by using Monte Carlo simulation, and the corresponding probability distributions are presented in Fig. 4.

As expected, the probability that a three-bit error pattern is not successfully decoded steadily decreases with the increase of the parameter L, and we obtain $p_{PGDBF}(100) = 3 \times 10^{-5}$ for the standard PGDBF algorithm. On the contrary, the simulation results show that the five-bit error pattern from Fig.

Algorithm 1 MUDRI decoder

Input: $\mathbf{y}$
$\forall v \in V:\ \hat{x}_v^{(0)} \leftarrow y_v$
$\mathbf{s}^{(0)} \leftarrow \hat{\mathbf{x}}^{(0)} H^T\ (\forall c \in C:\ s_c^{(0)} \leftarrow \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(0)})$
$n = 0, l = 0$
while $\mathbf{s}^{(l)} \neq \mathbf{0}$ and $n \leq \lfloor L/L_1 \rfloor$ **do**
  $l = 0, in^{(0)} = 0, v_f^{(0)} = 0, \forall v \in V:\ \hat{x}_v^{(0)} \leftarrow y_v$
  $\mathbf{s}^{(0)} \leftarrow \hat{\mathbf{x}}^{(0)} H^T\ (\forall c \in C:\ s_c^{(0)} \leftarrow \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(0)})$
  while $\mathbf{s}^{(l)} \neq \mathbf{0}$ and $l \leq L_1$ **do**
    $\forall v \in V$: Compute $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$
    $b^{(l)}, in^{(l)}, v_f^{(l)} \leftarrow FUN(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}), in^{(l-1)}, v_f^{(l-1)})$
    for $\forall v \in V$ **do**
      if $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) \geq b^{(l)}$ **then**
        $\hat{x}_v^{(l+1)} \leftarrow a_v \oplus \hat{x}_v^{(l)}$
      else
        $\hat{x}_v^{(l+1)} \leftarrow \hat{x}_v^{(l)}$
      end if
    end for
    $\mathbf{s}^{(l+1)} \leftarrow \hat{\mathbf{x}}^{(l+1)} H^T$
    $l \leftarrow l + 1$
  end while
  $n \leftarrow n + 1$
end while
Output: $\hat{\mathbf{x}}^{(l)}$

Algorithm 2 FUN: Adaptation of threshold in MAJ gates

Input: $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}), in^{(l-1)}, v_f^{(l-1)}$
$b^{(l)} \leftarrow \max_v(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})))$
$in^{(l)} = 0, v_f^{(l)} = 0$
for $\forall v \in V$ **do**
  if $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = b^{(l)}$ **then**
    $in^{(l)} = in^{(l)} + 1$
    $v_f^{(l)} \leftarrow v$
  end if
end for
if $l > 1$ and $in^{(l)} = in^{(l-1)} = 1$ and $v_f^{(l)} = v_f^{(l-1)}$ **then**
  $b^{(l)} \leftarrow \max_{2,v}(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})))$
end if

2(a) is either corrected in 14 or less iterations, or it cannot be corrected at all ($s_l \approx 0$ for $l > 14$). In this case, the probability of decoding failure is estimated as $p_{PGDBF}(14) = 0.8768$. The increase of L cannot help by itself, but combined with the proposed modification with multiple attempts, it results in lowering probability of unsuccessful decoding. Further optimization of the parameter L_1 also results in lowering p_{MUDRI}, as presented in Fig. 5. It can be noticed that the best results are obtained for approximately $L_1 = 6$ decoding iterations per attempt.

In Fig. 6, the frame error rate (FER) as a function of number of iterations is presented for $\alpha = 0.01$. Although

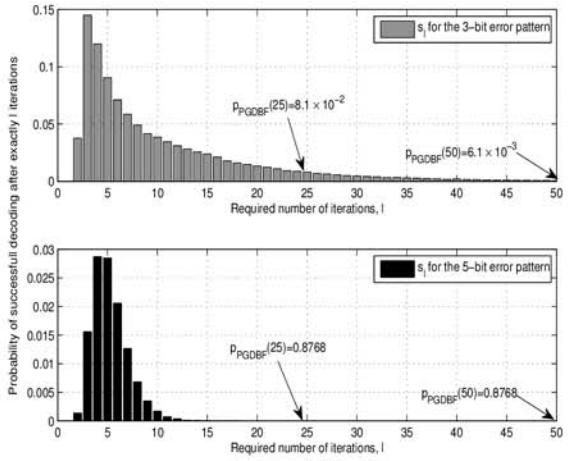

Fig. 4. Probability distribution of the successful decoding in the l-th iteration of PGDBF, three-bit and five-bit error pattern, Tanner (155,64) code, $p = 0.7$.

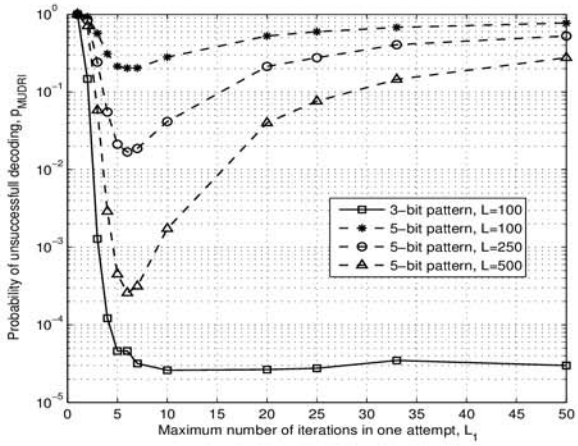

Fig. 5. Probability of unsuccessful decoding for the three-bit and five-bit error pattern, MUDRI with $\lfloor L/L_1 \rfloor$ attempts per L_1 iterations each, $p = 0.7$.

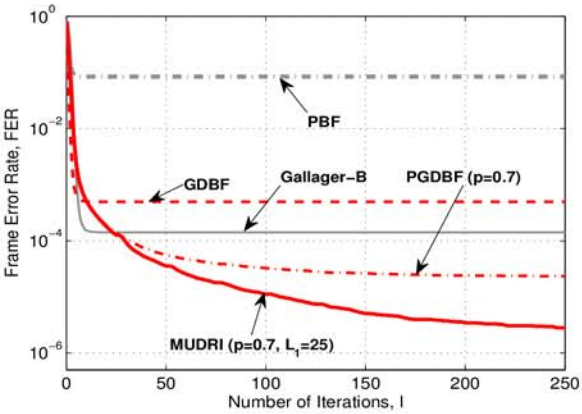

Fig. 6. FER as a function of number of iterations l, Tanner $(155, 64)$ code, $\alpha = 0.01$, various decoding algorithms.

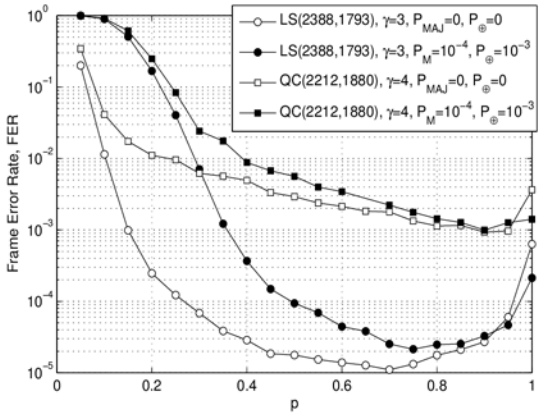

Fig. 7. FER as a function of parameter p, LDPC codes with $\gamma = 3$ and $\gamma = 4$, $\alpha = 0.004$.

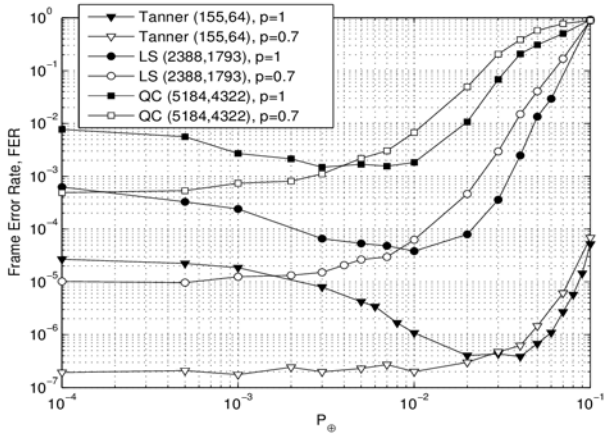

Fig. 8. FER as a function of probability of error in XOR gates, $\alpha = 0.004$, $P_R = 0$, LDPC codes with $\gamma = 3$ and girth-8, with various code rates and codeword lengths.

it is not convenient to adapt parameter L_1 for every error pattern, the simulations indicates that the minimal value of FER (i.e. $p_{MUDRI}(L, L_1)$ averaged over all received error patterns) is achieved for $L_1 \approx 25$ for Tanner (155,64) code and this parameter is somewhat larger for longer codes.

It is interesting to notice that while the PBF, GDBF and Gallager-B decoders need not more than 30 iterations to converge, after which their FER performance has reached the lowest possible value, the PGDBF continues to improve its FER performance up to 100 iterations and results in significant gain compared to the GDBF. The MUDRI, with ten attempts per each of $L_1 = 25$ iterations, results in an order of magnitude lower FER when compared to the PGDBF. The algorithm performance further improves with the increase of parameter L, to approximately FER=6×10^{-7} when $L = 2000$.

B. Performance in the faulty hardware implementation

With an aim of demonstrating the robustness of the algorithm to the hardware failures, we consider the canonical transient von-Neumann logic gate failure mechanism in which the failures in different gates and in different time instants are independent and identically distributed. The failures manifest themselves as random bit flips at the gate outputs. All XOR gates have probability of failure $P_{\oplus}$, and failures in the register where $\hat{\mathbf{x}}^{(l)}$ is stored occur with probability P_R. We also assume that MAJ gates are reliable, i.e. $P_{MAJ} \approx 0$. Although optimistic, this can be readily realized by using, for example, larger transistors in MAJ gates. Now we present the numerical results of Monte Carlo simulations for $L = 100$ and $L_1 = 50$.

First, we present the FER performance of two codes with similar codeword lengths but different column weights. The performance of $(2388, 1793)$ code (code $\mathcal{C}_1$) with girth-8 and $\gamma = 3$ based on Latin Squares [15] and $(2212, 1880)$ code (code $\mathcal{C}_2$) with girth-6 and $\gamma = 4$ are determined as a function of parameter p and presented in Fig. 7. When $p = 1$, the algorithm realized in faulty hardware has lower FER than the algorithm implemented in perfect hardware, and the performance can be further improved by reducing

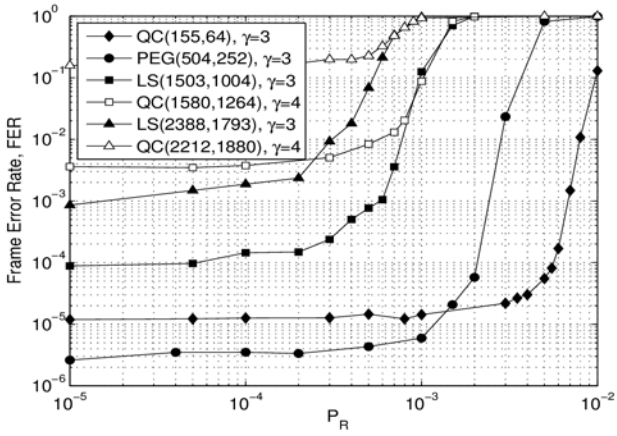

Fig. 9. FER as a function of probability of error in registers, $\alpha = 0.008$, $P_{\oplus} = 10^{-3}$, LDPC codes with $\gamma = 3$ and girth-8, various codeword lengths.

the parameter p for both codes. For $\mathcal{C}_1$ the best performance is obtained for $p \approx 0.7$ in the non faulty case, while the lowest FER is obtained for $p \approx 0.8$ when $P_{\oplus} = 10^{-3}$, and $P_R = 10^{-4}$. This corresponds to the previously published results for short quasi-cyclic codes with girth-8 and $\gamma = 3$ [13]. For $\mathcal{C}_2$ (with $\gamma = 4$), the best performance are obtained if $p \approx 0.9$ for the non-faulty case, while for the faulty implementation the optimum value of p is slightly larger.

The FER performance of QC and LS codes with various code rates are presented in Fig. 8, as a function of the parameter $P_{\oplus}$. If $p = 1$, the best performance is achieved for the non-zero value of $P_{\oplus}$. On the other hand, if $p = 0.7$ the FER is significantly reduced for small values of $P_{\oplus}$, when compared to the $p = 1$ case. More importantly, when $p = 0.7$ the FER is almost insensitive to $P_{\oplus}$ in a wide range of $P_{\oplus}$ values, up to a certain threshold, and is dominantly determined by the codeword length. The threshold can be estimated as $P_{\oplus,th} = 5/N$ for the codes with $\gamma = 3$ and girth-8.

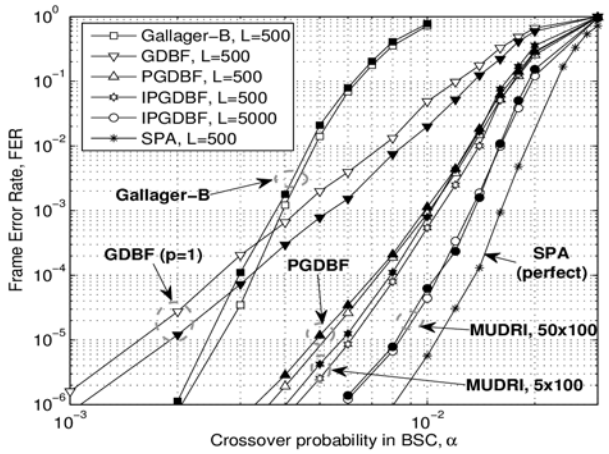

Fig. 10. FER as a function of crossover probability in BSC channel. The code is LS$(2388, 1793)$ $(\mathcal{C}_1)$, the empty markers corresponds to perfect hardware and full markers to faulty hardware with $P_{\oplus} = 10^{-3}$, $P_R = 10^{-4}$, $p = 0.7$ in the PGDBF and MUDRI and other decoding algorithms.

In Fig. 9, we present the FER performance for five LDPC codes with various code constructions (QC, PEG, LS), column weights and codeword lengths (available in [16]), and for the case when $\alpha = 0.008$, $P_{\oplus} = 10^{-3}$ and $p = 0.7$. It is clear that the MUDRI decoder has approximately same performance, up to a certain threshold of P_R. The value of P_R where FER doubles with respect to the non-faulty case is dominantly determined by the codeword length. For the codes with $\gamma = 3$ and girth-8, this threshold is estimated to be $P_{R,th} = 1/(2N)$. Although the codes with $\gamma = 4$ and girth-6 have lower error correction capability, they are somewhat less sensitive to the logic gate failures.

The FER performance of $\mathcal{C}_1$ for various decoders is presented in Fig. 10. It can be noticed that the Gallager-B outperforms the GDBF for lower values of crossover probability in BSC channel and the GDBF is more effective in the waterfall region. In the presence of gate failures, the performance is degraded for the Gallager-B decoder, but is improved for the GDBF $(p = 1)$. The performance of MUDRI with $p = 0.7$ outperforms all hard decision algorithms for the analyzed crossover probability, and the increase of the parameter L results in additional performance improvement. In addition, the MUDRI is less sensitive to hardware failures when compared to the Gallager-B and PGDBF.

V. Conclusion

By analyzing the characteristics trapping sets, we have improved the PGDBF algorithm. The new algorithm is capable of correcting error patterns uncorrectable by the GDBF algorithm and its probabilistic variant. The modification results in a significant performance improvement, especially in the case when large maximum number of iterations is permitted. It has been shown that the corresponding decoder is robust to the logic gate failures. We have shown that the critical probability of failure in XOR logic gates and registers is rather insensitive to the code construction method and rate, and it is mostly determined by the codeword length.

Our future research focuses on identifying the flipping sequences resulting in minimum number of iterations required for successful decoding of critical error patterns. As a result, we expect to design a low complexity deterministic modification of GDBF algorithm with fast convergence.

Acknowledgement

This work was supported by the Seventh Framework Programme 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.

References

[1] S. Ghosh and K. Roy, "Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1718–1751, Oct. 2010.

[2] 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. Circuits Syst. I, Reg. Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.

[3] S. Chilappagari, M. Ivkovic, and B. Vasic, "Analysis of one step majority logic decoders constructed from faulty gates," in *Proc. 2006 IEEE International Symposium on Information Theory*, July, pp. 469–473.

[4] S. Brkic, P. Ivanis, and B. Vasic, "Analysis of one-step majority logic decoding under correlated data-dependent gate failures," in *Proc. 2014 IEEE International Symposium on Information Theory (ISIT)*, June, pp. 2599–2603.

[5] L. Varshney, "Performance of LDPC Codes Under Faulty Iterative Decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, July 2011.

[6] C. Kameni Ngassa, V. Savin, and D. Declercq, "Min-Sum-based decoders running on noisy hardware," in *Proc. 2013 IEEE GLOBECOM*, Dec.

[7] C.-H. Huang and L. Dolecek, "Analysis of finite-alphabet iterative decoders under processing errors," in *Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May, pp. 5085–5089.

[8] 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.

[9] 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.

[10] D. V. Nguyen and B. Vasic, "Two-Bit Bit Flipping Algorithms for LDPC Codes and Collective Error Correction," *IEEE Trans. Comm.*, vol. 62, no. 4, pp. 1153–1163, April 2014.

[11] 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.

[12] 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.

[13] O. A. Rasheed, P. Ivanis, and B. Vasic, "Fault-Tolerant Probabilistic Gradient-Descent Bit Flipping Decoder," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487–1490, Sept 2014.

[14] D. Declercq, E. Li, B. Vasic, and S. Planjery, "Approaching maximum likelihood decoding of finite length LDPC codes via FAID diversity," in *Proc. 2012 IEEE Information Theory Workshop (ITW)*, Sept 2012, pp. 487–491.

[15] D. Nguyen, S. Chilappagari, M. Marcellin, and B. Vasic, "On the Construction of Structured LDPC Codes Free of Small Trapping Sets," *IEEE Trans. Inform. Theory*, vol. 58, no. 4, pp. 2280–2302, April 2012.

[16] L. Danjean and S. K. Planjery, http://www2.engr.arizona.edu/~vasiclab/tools/LDPC_Code_List.