Reliability of Memories Built from Unreliable Components under Data-Dependent Gate Failures

Srdan Brkic, *Student Member, IEEE,* Predrag Ivaniš, *Member, IEEE,* and Bane Vasić, *Fellow, IEEE*

***Abstract*—In this letter we investigate fault-tolerance of memories built from unreliable cells. In order to increase the memory reliability, information is encoded by a low-density parity-check (LDPC) code, and then stored. The memory content is updated periodically by the bit-flipping decoder, built also from unreliable logic gates, whose failures are transient and data-dependent. Based on the expander property of Tanner graph of LDPC codes, we prove that the proposed memory architecture can tolerate a fixed fraction of component failures and consequently preserve all the stored information, if code length tends to infinity.**

***Index Terms*—Data-dependence, faulty bit-flipping decoding, low-density parity-check codes, reliable memory architecture.**

I. Introduction

Recent research in the area of fault-tolerant memories based on decoders of low-density parity-check (LDPC) codes is mainly inspired by the studies presented in the late sixties and early seventies by Taylor [1] and Kuznetsov [2]. In their pioneering works, they proposed a memory architecture built entirely from unreliable components, which is capable of preserving stored information over arbitrary long time. The memory is composed of unreliable memory cells that are storing a codeword of an LDPC code, and are periodically updated using a faulty iterative decoder. Attractiveness of using LDPC codes lays in the theoretical guarantee that the decoding hardware overhead required to ensure reliable operation grows only linearly with the code length even when logic gates are faulty [1]. It was observed by Vasić *et al.* in [3] that an update cycle corresponds to one iteration of the Gallager-B decoder, built from unreliable logic gates. In the later work, Varshney [4] used density evolution analysis to prove that a memory based on the Gallager-A decoder is also capable of preserving information in asymptotic code length. Chilappagari and Vasić [5] used the expander arguments to show the existence of a reliable memory based on the bit-flipping algorithm. The reliability of the same architecture was recently studied by Dupraz *et al.* in [6].

Both Taylor and Kuznetsov as well as most of the related work modeled logic gate unreliability as transient independent failures, originally introduced by von Neumann [7]. Although the simplicity of this model makes it attractive for theoretical analysis, it is unrealistic. In practice, unreliability of logic gates is strongly data-dependent and correlated in time. One of the most dominant effects impacting reliability of logic gates built in energy-efficient subpowered CMOS technologies comes from the so-called timing violations, which depend on a gate's switching activity [8], [9]. Their effects to different hard decision decoders have been recently studied in [10]–[12].

In this letter we establish a bound on a number of correctable errors for a memory system which employ the bit-flipping decoder. Unlike in the prior research [5], [6], we evaluate the memory reliability in the presence of data-dependent gate failures. Following the recent work by Brkic *et al.* [12] on guaranteed error correction capability of faulty bit-flipping decoders, we prove that our memory can tolerate a fixed fraction of component failures. Consequently, we show that in the asymptotic case the memory can preserve all stored information, which presents the first proof of memory reliability under a failure model other then the von Neumann model. We refine the results presented in [5] by improving conditions required for the memory reliability. In addition to our analytical results, we present numerical results illustrating upper bounds on tolerable fractions of component failures.

II. System Model

A. The Memory Architecture

The information is stored in a memory as a codeword of a (γ, ρ)-regular LDPC code in n memory cells. Each memory cell stores one code bit. In order to preserve the stored codeword, the memory cells are periodically updated, at regular time instants $\tau, 2\tau, \dots, L\tau, L \in \mathbb{N}$, based on the error correction scheme, described as follows.

Consider a graphical representation of a (γ, ρ)-regular binary LDPC code given by Tanner bipartite graph $G = (V \cup C, E)$, where V is a set of variable nodes (variables), C is a set of check nodes, and E is a set of edges. An edge $e \in E$ is an unordered pair (v, c) which connects two nodes $v \in V$ and $c \in C$. Nodes v and c are called neighbors iff there is an edge between them. Let E_v (E_c) be a set of edges connected to a variable node v (check node c). Then, $|E_v| = \gamma$, $\forall v \in V$, and $|E_c| = \rho$, $\forall c \in C$, where $|\cdot|$ denotes the cardinality.

Let $x_v(t)$ be a value of a memory cell associated to a variable node v at time t. Let $\overrightarrow{m}_\ell(e)$ ($\overleftarrow{m}_\ell(e)$) be messages passed on an edge e from/to variable node to/from check node during an update cycle $\ell\tau$, $\ell > 0$, respectively. The ℓ-th update cycle can be summarized as follows.

- The content of a memory cell v, $v \in V$, at time $\ell\tau - \delta_0$, where δ_0 denotes an infinitesimal duration of time, is

This work was supported by the Seventh Framework Programme of the European Union, under Grant Agreement number 309129 (i-RISC project), Serbian Ministry of Science under project TR32028, NSF under Grants CCF-0963726, CCF-1314147, ECCS-1500170 and the Fulbright Scholar Program.

S. Brkic, and P. Ivaniš are with the University of Belgrade, Serbia, School of Electrical Engineering, (e-mails: srdjan.brkic@ic.etf.rs, predrag.ivanis@etf.rs), B. Vasić is with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721 USA (e-mail: vasic@ece.arizona.edu).

passed to the neighboring check nodes, i.e., $\overrightarrow{m}_\ell(e) = x_v(\ell\tau - \delta_0)$, $\forall e \in E_v$.

- Each check node $c \in C$ calculates ρ XOR operations

$$\overleftarrow{m}_\ell(e) = \bigoplus_{e' \in E(c)\setminus\{e'\}} \overrightarrow{m}_\ell(e'),\ \forall e \in E_c.$$

- The content of a memory cell $v \in V$ is updated by using a γ-input majority logic (MAJ) gate as follows

$$x_v(\ell\tau+\delta_0) = \begin{cases} s, \text{ if } |\{e' \in E_v : \overleftarrow{m}_\ell(e') = s\}| \geq \lceil \frac{\gamma}{2} \rceil, \\ x_v(\ell\tau - \delta_0), \quad \text{otherwise}, \end{cases}$$

where $s \in \{0, 1\}$ and $\lceil \gamma/2 \rceil$ denotes the smallest integer greater than $\gamma/2$.

Note that during an update cycle a check node calculates estimates of the neighboring variable nodes, rather than the parity check equation, since the value of the bit that is estimated is not used in the calculation. However, the update cycle is functionally equivalent to one iteration of the parallel bit-flipping decoder [13].

Hardware unreliability of the correction scheme comes from unreliable computation of messages $\overrightarrow{m}_\ell(e)$ and $\overleftarrow{m}_\ell(e)$, as logic gates performing these functions are prone to data-dependent failures, which are described in the following subsection.

B. Failure Models

Two types of hardware components failures are considered in this letter: memory cell failures and logic gate failures. We assume that failures in the memory cells are a consequence of supply voltage variations. These errors are transient and manifest as random flips that corrupt values stored in memory cells without damaging the cells [14]. Hence, we can assume that between two update cycles memory content is transmitted through the binary symmetric channel for which $\Pr\{x_v(\ell\tau + \delta_0) \neq x_v((\ell+1)\tau - \delta_0)\} = p_m$, $\forall v \in V$ and $\ell > 0$.

On the other hand, failures of subpowered CMOS logic gates are data-dependent and correlated in time and cannot be represented in the same manner as memory failures [8], [9]. In this letter we consider the *probabilistic gate-output switching model* (GOS), recently proposed by Amaricai *et al.* [9], which assumes that a gate failure depends on switching activity of the gate.

Let $z(\ell\tau)$ be the correct output of a logic gate at time $\ell\tau$. Due to unreliability of the gate, the actual output is $z(\ell\tau) \oplus \xi(\ell\tau)$, where $\xi(\ell\tau) \in \{0, 1\}$ is the error at time $\ell\tau$. In the GOS error model, when the correct gate output is unchanged during two consecutive time instants, the actual gate output is always correctly computed, i.e., $\Pr\{\xi(\ell\tau) = 1 | z(\ell\tau) = z((\ell-1)\tau)\} = 0$. On the other hand, the gate fails to switch with probability $\Pr\{\xi(\ell\tau) = 1 | z(\ell\tau) \neq z((\ell-1)\tau)\} = p_g$, $p_g > 0$, $g \in \{\oplus, \gamma\}$, where with slight abuse of notation γ signifies a γ-input MAJ gate. In general, the failure probability of XOR gates $p_\oplus$ can be different from the failure probability of MAJ gates p_γ. Although the GOS model does not capture all the effects that lead to failures of subpowered CMOS circuits, it was shown that loss of accuracy by using this modeling approach is relatively small [9]. The GOS model has been studied recently in a number of papers [10]–[12].

We will first prove that the memory architecture can tolerate a fixed fraction of errors in all components. Then we will use Chernoff bounds to extend our results to the presented probabilistic error models. Namely, in the first part of our proof we assume that a component failure follows the statistics described above, but we allow only a fraction of failures during the interval $((\ell-1)\tau, \ell\tau)$, $\ell > 0$. This means that the number of memory cell failures between two update cycles is bounded by $\alpha_m n$. Similarly, we allow $\alpha_\gamma n$ MAJ logic gates to be faulty, while the rest of $(1 - \alpha_\gamma)n$ MAJ gates operate reliably. Note that we do not require reliable XOR gates, i.e., according to the GOS model failure of every XOR gate used in the correction scheme can occur when the gate output changes.

C. Error correction of bit-flipping decoders

Our proof that the memory can tolerate a fixed fraction of errors in all components relies on expanders, and here we give the necessary lemmas established by Sipser and Spielman [13] and Brkic *et al.* [12] regarding the error correction capability of bit-flipping decoders. Lemma 1 pertains to decoders made of reliable components, while Lemma 2 gives the correction capability of a faulty decoder whose gates fail as described in the previous subsection.

Definition 1. *[13] A Tanner graph G of a (γ, ρ)-regular LDPC code is a $(\gamma, \rho, \alpha, \delta)$ expander if for every subset S of at most αn variable nodes, at least $\delta|S|$ check nodes are incident to S.*

Let V_ℓ be a set of corrupt (erroneous) variables at the beginning of the ℓ-th decoding iteration. The following lemmas depict the error correction capabilities of bit-flipping decoders when the underlying Tanner graph is $(\gamma, \rho, \alpha, (7/8 + \epsilon)\gamma)$, $\epsilon > 0$ expander.

Lemma 1. *The parallel bit-flipping decoder built from reliable components can correct any fraction of $\alpha_r < (3 + 8\epsilon)\alpha/4$ errors after at most $\log_{2/(1-8\epsilon)} \alpha_r n$ iterations. Also, for every $|V_1| \leq \alpha_r n$ and $\ell > 1$ holds $|V_{\ell+1}| \leq (1 - 8\epsilon)|V_\ell|/2$.*

Proof: See [13]. ■

Lemma 2. *The parallel bit-flipping decoder built from unreliable check nodes can correct any fraction of $\alpha_u < 3(3 + 8\epsilon)\alpha/32$ errors. Also, for every $|V_1| \leq \alpha_u n$ and $\ell > 1$ holds*

$$(1 - 8\epsilon)|V_\ell| \geq 2|V_{\ell+1}| - (1 - 8\epsilon)|V_{\ell-1}|. \tag{1}$$

Proof: See [12]. ■

III. Reliability of the Memory Architecture

When the memory is built entirely from unreliable components, the bits read from the memory at some time instant, in the most of the cases will not be the same as in the originally stored codeword. Thus, if we want to recover the information, the final step of codeword extraction must be performed by reliable logic gates. In this letter we follow the system setup proposed by Taylor [1], which states that *memory failure* is declared only if the sequence read from the memory cannot be successfully decoded by the noiseless version of the same decoder in a finite number of iterations. We show that our memory architecture under certain conditions achieves arbitrary low memory failure probability.

Note that logic gate failures at the time of the first update cycle depend on the instant before the codeword is stored in the memory cells. There is a practical approach to resolve this issue by slowing down the clock in the first update cycle and letting the signal level stabilize [11]. This leads to fully reliable logic gate operations in the first update cycle, which is assumed in our analysis.

We first investigate what fraction of memory failures α_m can be tolerated by our memory if we allow all gates, used in the correction scheme, to be faulty. This is given in the following lemma.

Lemma 3. *The proposed memory architecture built on a (γ, ρ)-regular LDPC code free of four cycles can tolerate a fraction of memory failures if* $\alpha_m \leq \left\lfloor \gamma/6 \right\rfloor /n$.

Proof: See Appendix A. ■

Note that increasing n forces α_m to reduce, and the number of tolerable memory failures $\alpha_m n$ cannot exceed $\lfloor \gamma/6 \rfloor$. This means that under this conditions the arbitrary small memory failure probability can be achieved only if γ tends to infinity. The main reason for a such behavior lies in the fact that, under the GOS model, failures of MAJ gates can cancel out error correction gain achieved during the update cycle. The only way to prevent this is to allow a number of MAJ gates to operate fully reliable. In other words, we bound a fraction of faulty MAJ gates to α_γ, but we do not put any restrictions on the number of faults in check nodes. They all remain prone to data-dependent failures.

Theorem 1. *The proposed memory architecture based on a* $(\gamma, \rho, \alpha, (7/8 + \epsilon)\gamma)$ *expander code can preserve all stored bits for an arbitrary long time period if*

$$\alpha_m + \alpha_\gamma < 3\epsilon(3 + 8\epsilon)\alpha/4. \tag{2}$$

Proof: At t=0 a codeword of our expander code is written into the memory. The memory cells are updated at time instants $\ell\tau$, $\ell > 0$, by performing one iteration of the bit-flipping algorithm. Let $V(t)$ be a set of corrupt variables (memory cells) at time t. The number of corrupt variables before the first update $|V(\tau - \delta_0)|$ is bounded by

$$|V(\tau - \delta_0)| \leq n\alpha_m.$$

After the update cycle we have

$$|V(\tau + \delta_0)| \leq \beta\alpha_m n + \alpha_\gamma n,$$

where, according to Lemma 1, $\beta = (1 - 8\epsilon)/2$. In the time interval $(\tau, 2\tau)$ there can be at most $\alpha_m n$ memory cells failures, which in the worst case will lead to $\alpha_m n$ additional corrupt variables. Then,

$$|V(2\tau - \delta_0)| \leq \beta\alpha_m n + \alpha_m n + \alpha_\gamma n.$$

Based on Eq. (1) and the previous discussion for all $\ell > 1$ we obtain

$$|V((\ell+1)\tau - \delta_0)| \leq \beta\Big(|V(\ell\tau - \delta_0)| + |V((\ell-1)\tau - \delta_0)|\Big) + \alpha_\gamma n + \alpha_m n.$$

From the previous inequality follows that the number of corrupt memory cells can be upper bounded, which is formally presented in the following lemma.

Lemma 4. *The number of corrupt memory cells before the ℓ-th update cycle $|V(\ell\tau - \delta_0)|$, for all $\ell > 0$, satisfies*

$$|V(\ell\tau - \delta_0)| \leq (\alpha_m n + \alpha_\gamma n)/(8\epsilon).$$

Proof: See Appendix B. ■

Since by Eq. (2)

$$(\alpha_m n + \alpha_\gamma n)/(8\epsilon) < (3 + 8\epsilon)\alpha n/4,$$

from Lemma 4 follows that the number of corrupt memory cells at any time instant does not exceed the error correction capability of the bit-flipping decoder, given by Lemma 1, and the memory content is preserved. This proves the theorem. ■

It is important to note that, in order to prove the memory reliability bound presented in [5], the number of faulty XOR gates had to be bounded. Here we do not need this condition for the data-dependent failure model and the larger number of faulty components can be used in the memory.

Now we utilize Theorem 1 to bound the memory performance under the probabilistic failure model. Let $\Delta_m > 0$ and $\Delta_\gamma > 0$ be such that $p_m + \Delta_m = \alpha_m$ and $p_\gamma + \Delta_\gamma = \alpha_\gamma$. When condition given by Eq. (2) is satisfied, the following lemma can be formulated.

Lemma 5. *The probability that memory failure occurs after L update cycles, $P(L)$, is bounded by*

$$P(L) \leq L(e^{-2\Delta_m^2 n} + e^{-2\Delta_\gamma^2 n}).$$

Proof: The proof follows from the fact that by Chernoff bounds the probability of failure of more than a fixed fraction of components at time interval τ is bounded. ■

The previous lemma describes a weak bound on the memory performance and its main goal is to show that probability of failure $P(L)$ decreases exponentially when the code length increases. It proves the existence of a memory that can preserve all stored bits in asymptotic code length under the data-dependent gate failure model.

IV. Numerical Results

We next show how the right side of the Eq. (2), denoted by $\alpha_{total}(\alpha, \epsilon) = 3\epsilon(3 + 8\epsilon)\alpha/4$, can be upper bounded. For that purpose the following lemma is used.

Lemma 6. *Let assume the existence of a* $(\gamma, \rho, \alpha, (7/8+\epsilon)\gamma)$, $\epsilon > 0$ *expander. Then, when code length goes to infinity, α and ϵ must satisfy* $\epsilon \leq (1 - (1 - \alpha)^\rho)/(\alpha\rho) - 7/8$.

Proof: See [13, Theorem 25] and [12, Lemma 7]. ■

We can numerically express upper bounds on α_{total}, which satisfy the condition given by Lemma 6, for fixed values of ρ. The bounding values are divided between α_m and α_γ, which creates the tolerable error regions presented in Fig. 1. It can be observed that by increasing ρ, under fixed γ, the number of neighbors of a set with αn variable nodes reduces. On the other hand, we require that the set with αn variable nodes have

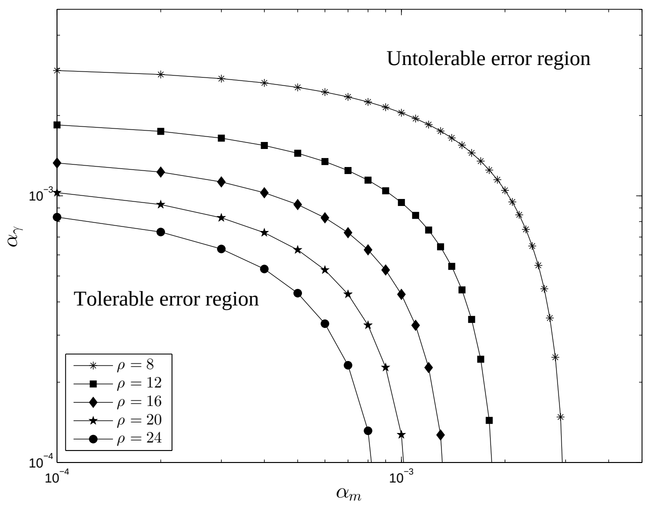

Fig. 1: Upper bounds on tolerable error fractions.

the expansion of more than $7\gamma/8$, which can be only satisfied by reducing α. Consequently, α_{total} is inversely proportional to ρ. For example, when $\rho = 8$ $\alpha_{total} = 0.003$, while for $\rho = 24$ the memory cannot tolerate the fraction of more then $\alpha_{total} = 0.0009$ errors.

V. Conclusion

In this letter we proved the existence of a memory architecture that achieves arbitrary small failure probability under the data-dependent gate failure model, which presents the first such result under failure models other than the von Neumann model. In addition, we provided upper bounds on fractions of component failures that can be tolerated by our memory.

Appendix A (Proof of Lemma 3)

When perfectly reliable correction scheme is used, for every LDPC code free of four cycles, during an update cycle up to $\lfloor \gamma/2 \rfloor$ erroneous cells can be corrected. In other words, each bit will be decoded correctly if the number of incorrect estimates at inputs of the MAJ gate is no greater then $\lfloor \gamma/2 \rfloor$. In the faulty scheme an incorrect estimate can also appear due to unreliability of a XOR gate. In the worst case N_m memory errors can produce N_m incorrect estimates and, similarly, $N_\oplus$ XOR failures can lead to $N_\oplus$ incorrect estimates. Thus, each bit will be correctly decoded if $N_m + N_\oplus \leq \lfloor \gamma/2 \rfloor$. If the total number of errors in the time interval τ is bounded by the previous condition, after every correcting cycle in the memory cells only correct values are written, and there is no failures of MAJ gates.

Let $\mathcal{F}_\ell$ be a set of all erroneous memory cells between $(\ell-1)$-th and ℓ-th update cycle. It follows that $|\mathcal{F}_\ell| = N_m \leq \alpha_m n$, for any $\ell > 0$. It is clear that the maximal number of faulty XOR gates correspond to the case when $\mathcal{F}_{\ell-1} \cap \mathcal{F}_\ell = \emptyset$. Then, the total number of faulty XOR gates used for decoding a particular bit is bounded by $N_\oplus \leq 2\alpha_m n$.

Appendix B (Proof of Lemma 4)

The number of corrupt variables before the ℓ-th update cycle satisfies

$$|V(\ell\tau - \delta_0)| \leq A_1\lambda_1^{-\ell} - A_2\lambda_2^{-\ell} + K, \tag{3}$$

where $\lambda_1 = -0.5(1 + \sqrt{1 + 4/\beta})$, $\lambda_2 = 0.5(\sqrt{1+4/\beta} - 1)$, A_1 and A_2 are constants of the complementary solution of the difference equation $x_\ell - \beta x_{\ell-1} - \beta x_{\ell-2} = \alpha_m n + \alpha_\gamma n$, whose particular solution K satisfies initial conditions $x_1 = \alpha_m$ and $x_2 = (\beta+1)\alpha_m n + \alpha_\gamma n$. It is not difficult to show that

$$A_j = \frac{\beta(2+\lambda_j)\alpha_m n + (1+\beta\lambda_j)\alpha_\gamma n}{\beta(\lambda_2 - \lambda_1)(1-2\beta)}, \quad j = 1, 2,$$

$$K = (\alpha_m + \alpha_\gamma)n/(1-2\beta) = (\alpha_m + \alpha_\gamma)n/(8\epsilon).$$

Since $|A_1/A_2| < 1$ and $2 > |\lambda_1/\lambda_2| > 1$, the right hand side of Eq. (3) monotonically increases with ℓ and we have

$$\lim_{\ell\to\infty}\left[A_1\lambda_1^{-\ell} - A_2\lambda_2^{-\ell} + K\right] = K.$$

References

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

[2] A. Kuznetsov, "Information storage in a memory assembled from unreliable components," *Problems of Information Transmission*, vol. 9, pp. 254–264, 1973.

[3] 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 Transactions on Circuits and Systems I, Regular Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.

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

[5] S. Chilappagari and B. Vasic, "Fault tolerant memories based on expander graphs," in *Proceedings of IEEE Information Theory Workshop*, Tahoe City, CA, USA, 2–7 Sep. 2007, pp. 126–131.

[6] E. Dupraz, Declercq, and B. Vasic, "Analysis of Taylor-Kuznetsov memory using one-step majority logic decoder," in *Proceedings of 10th Information Theory and Applications Workshop (ITA 2015)*, San Diego, CA, Feb. 2015, paper 273, [Online Available:] http://ita.ucsd.edu/workshop/15/files/paper/paper 3446.pdf.

[7] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies*, C.E. Shannon and J. McCarty, eds., Princeton Univ. Press, July 1956, pp. 43–98.

[8] S. Zaynoun, M. S. Khairy, A. M. Eltawil, F. J. Kurdahi, and A. Khajeh, "Fast error aware model for arithmetic and logic circuits," in *Proceedings of 30th IEEE International Conference on Computer Design (ICCD)*, Montreal, QC, Sep.–Oct. 2012, pp. 322–328.

[9] A. Amaricai, S. Nimara, O. Boncalo, J. Chen, and E. Popovici, "Probabilistic gate level fault modeling for near and sub-threshold CMOS circuits," in *Proc. 17th Euromicro Conf. on Digital Syst. Design (DSD)*, Verona, Avg. 2014, pp. 473–479.

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

[11] S. Brkic, O. Al Rasheed, P. Ivanis, and B. Vasic, "On fault tolerance of the Gallager B decoder under data-dependent gate failures," *IEEE Communications Letters*, vol. 19, no. 8, pp. 1299–1302, Aug. 2015.

[12] S. Brkic, P. Ivanis, and B. Vasic, "Majority logic decoding under data-dependent logic gate failures," submitted for publication, http://arxiv.org/abs/1507.07155.

[13] M. Sipser and D. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.

[14] A. Khajeh, K. Amiri, M. Khairy, A. M. Eltawil, and F. Kurdahi, "A unified hardware and channel noise model for communication systems," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 10)*, Miami, Florida, USA, 6–10 Dec. 2010, pp. 1–5.