FP7-ICT / FET-OPEN 309129 / i-RISC
D.4.2
Multi-Bit Flipping and Fast-Iterative Decoders

Editor:	Goran Đorđević
Deliverable nature:	Public
Due date:	January 31, 2015
Delivery date:	February 27, 2015
Version:	2.0
Date of current version:	April 15, 2016
Total number of pages:	71
Reviewed by:	i-RISC members
Keywords:	LDPC codes, TK memory, data-dependent errors, timing errors, one-step majority logic decoder, bit-flipping, probabilistic gradient descent bit-flipping, expander codes, robust interconnect

Abstract

This deliverable presents an overview of the activities carried out within the Work Package 4 (WP4) during the period Month 13 to Month 24 (M13-M24) of the project. These activities mainly include the analysis and design of low complexity LDPC decoders and corresponding memory architectures, robust to hardware unreliability.

A special attention is devoted to the decoder analysis under data-dependent logic gate failures. We propose memory architectures that build on the one-step majority logic decoder, and develop analytical tools to evaluate their robustness. We also investigate bit-flipping decodes built in part from unreliable components, and show they can tolerate a fixed fraction of error. An improved version of the probabilistic gradient descent bit flipping algorithm is also proposed, whose performance under unreliable hardware is shown to be superior compared to other known hard decision algorithms. In addition, we prove the existence of memory architectures that achieve an arbitrary small probability of failure under the data-dependent failure model, which represents the first such result for failure models other than the von Neumann model.

Moreover we started investigations on reliable data transport structures and this deliverable also presents preliminary steps towards the construction of energy effective reliable in-chip interconnects. To this end we targeted the Near/Sub-Threshold operating region and proposed a dual-rail interconnection strategy, which outperforms the single-rail counterpart in terms of energy consumption at the expense of some area overhead.

List of Authors

Participant	Author
ELFAK	Goran Đorđević (goran.t.djordjevic@elfak.ni.ac.rs)
	Bane Vasić (vasic@email.arizona.edu)
	Predrag Ivaniš (predrag.ivanis@etf.rs)
	Srđan Brkić (brka05@gmail.com)
	Omran Al Rasheed (omrano84@hotmail.com)
ENSEA	Elsa Dupraz (elsa.dupraz@ensea.fr)
	David Declercq (declercq@ensea.fr)
TUD	Joyan Chen (Joyan.Chen@tudelft.nl)
	Sorin Cotofana (S.D.Cotofana@tudelft.nl)

Contents

List of Dissemination Activities

[P1] S. Brkic, P. Ivanis, G. Djordjevic, B. Vasic, "Symbolic analysis of faulty logic circuits under correlated data-dependent gate failures," *Telfor Journal*, vol. 6, no. 1, pp. 2–6, 2014.

[P2] O. Al Rasheed, S. Brkic, P. Ivanis, B. Vasic, "Performance analysis of faulty Gallager-B decoding of QC-LDPC codes with applications," *Telfor Journal*, vol. 6, no. 1, pp. 7–11, 2014.

[P3] E. Dupraz, D. Declercq, B. Vasic, "Analysis of Taylor-Kuznetsov memory using one-step majority logic decoder," *in Information Theory and Applications Workshop*, San Diego, USA, Feb. 2015

[P4] P. Ivanis, O. Al Rasheed, and B. Vasic, "MUDRI: A fault-tolerant decoding algorithm," *IEEE International Conference on Communications (ICC 2015)*, London, UK, June 2015

[P5] S. Brkic, P. Ivanis, B. Vasic, "Reliable memories built of unreliable components: majority-logic decoding and analysis of data-dependent logic gate failures," *IEEE Transactions on Communications* (submitted)

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

BER	Bit Error Rate
FER	Frame Error Rate
LDPC	Low Density Parity Check
WP	Work Package
TK	Taylor-Kuznetsov
QC	Quasi-Cyclic
AG	Affine Geometry
PG	Projective Geometry
BSC	Binary Symmetric Chanel
LP	Linear Programming
BF	Bit-Flipping
BSC	Binary Symmetric Chanel
MUDRI	MUltiple Decoding attempts and Random re-Initializations
GDBF	Gradient Descent Bit-Flipping
PGDBF	Probabilistic Gradient Descent Bit-Flipping
FAID	Finite-Alphabet Iterative Decoder
SPA	Sum-Product Algorithm
QC	Quasi-Cyclic
PEG	progressive Edge Growth
LS	Latin Square
MIF	Modified Inverse Function
MAJ	MAJority logic
XOR	eXclusive OR
O-S-MAJ	One-Step MAJority logic
UPT	Universitatea Politehnica Timisoara
UCC	University College Cork
TUD	Technische Universiteit Delft
ELFAK	Elektronski Fakultet Nis

Introduction

This deliverable addresses various issues related to the potential on chip realization of data storage and transport facilities out of unreliable components. It summarizes the main research activities carried out during the second year of the i-RISC project, relating to the tasks of Work Package 4 (WP4) from the Description of Work document. The main objectives of WP4 include the analysis of state-of-the-art memory architectures under more realistic hardware failure modeling, the design of novel memory architectures able to tolerate data-dependent gate failures, and the design of new codes that ensure reliable and energy effective intra/inter-chip communication.

A Gantt chart of WP4 tasks and their time distribution, which indicates the tasks addressed and initiated during the period M13-M24, is presented in Figure 1 below. During this period, the main focus was on the investigation of hard decision LDPC decoders – suited for data storage and transport applications – and proposing techniques to increase their robustness when decoding operations are not perfectly reliable. Successful completion of all tasks in this WP will give us theoretical guidelines for building LDPC codes based low-complexity fault-tolerant memories and ensuring reliable data transport on unreliable hardware.

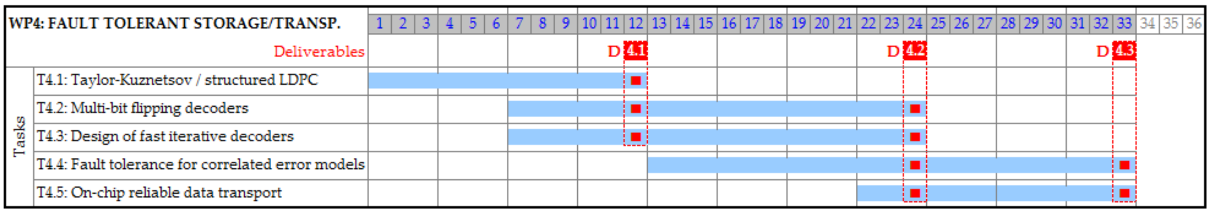


Figure 1: Gantt chart of WP4.

Executive Summary

We summarize below the main technical contributions reported in this deliverable. We discuss the most important results of our work and highlight their relevance to the overall i-RISC project strategy.

The following objectives have been addressed during the second year of the project:

Objective 4.1: Proposing novel memory architectures having linear complexity and high storage capacity that are based on structured LDPC codes.

Objective 4.2: Designing fault-tolerant memories with low redundancy rates.

Objective 4.3: Analysing the robustness of memories designed for tolerating both spatial and temporally correlated errors.

Objective 4.4: Designing the constrained codes for intra/inter-chip bus connections.

The list of objectives indicates that Task 4.1 – Taylor-Kuznetsov memory architectures based on structured LDPC codes, Task 4.3 – Design of iterative decoders with the fastest convergence, Task 4.4 – Fault tolerance for correlated error models and Task 4.5 – On-chip reliable data transport were mostly investigated during M13-M24. Note that Task 4.2 – Multi-bit flipping decoders was also scheduled to be investigated in M13-M24 time period, but most of the work related to this task was performed during the first year of the project and was included in Deliverable D4.1. Our main technical contributions relating to these tasks can be summarized below.

Analysis of Taylor-Kuznetsov Memory using One-Step Majority Logic Decoder (Task 4.1) In Chapter 1 we investigate the reliability of the Taylor-Kuznetsov memory which employs a faulty one-step majority logic decoder. Contrary to the recent research in which only the faulty one-step majority logic decoder was considered, we here examine the reliability of the whole memory construction. Based on a sequence of output errors probabilities at successive time instants we define a threshold that predicts the noise level which can be tolerated for the memory to stay reliable. Our analysis is restricted to the independent gate failure model.

Reliable Memories Built of Unreliable Components: Majority-Logic Decoding and Analysis of Data-Dependent Logic Gate Failures (Task 4.4). The analysis presented in Chapter 2 completes the work done during the first year of project related to one-step majority logic (O-S-MAJ) decoding. Based on the results presented in Deliverable 4.1, we investigate the performance of O-S-MAJ decoders in the presence of timing gate failures. We derive the analytical expressions for the upper and lower bounds on bit error rate, and use it to evaluate the performance of finite geometry LDPC codes. As a convenient measure of performance variation caused by timing gate failures we also define the data-dependence factor. In addition, we propose the low complexity memory architecture based on O-S-MAJ decoder, which proved to be robust to data-dependent gate failures.

Expander Graph Arguments for Bit-Flipping Decoders Made of Unreliable Gates (Task 4.4). In Chapter 3 we examine the simple bit-flipping (BF) decoder whose check node

operations are prone to data-dependent gate failures. We show that expander graph arguments can be successfully used to establish lower bounds on guaranteed error correction capability of faulty BF decoders. More precisely, we prove that if underlying Tanner graph has a sufficient expansion the number of errors that can be tolerated increases linearly with the code length. Similarly, we prove that the error correction of a noisy BF decoder can also be guaranteed when Tanner graph of a code has sufficient girth. In addition, we show that memory architecture based on BF decoder can tolerate a fixed faction of component failures. Based on that conclusion, we use Chernof bound to prove that in the asymptotic code length our memory architecture achieves arbitrary small probability of failure under data-dependent failure model. The work presented in Chapter 3 is of the theoretical importance to overall i-RISC project strategy, since represent the first result on guaranteed error correction of faulty decoders.

MUDRI: A Fault-Tolerant Decoding Algorithm (Task 4.3). Chapter 4 is dedicated to the design of a novel bit-flipping algorithm resistant to hardware unreliability. This algorithm represents an improved version of probabilistic gradient descent bit flipping (PGDBF) decoder, presented in Deliverable 3.2. We show that PGDBF decoder can be improved by random re-initializations of decoding process after certain number of iterations. This modifications not only ensures correction of some error patterns that are uncorrectable by PGDBF, but also increases the immunity of the decoder to the failures in registers and logic gates. Presented algorithm outperforms all known BF decoders and represents one of the most important results of WP 4.

Robust Interconnect for Near/Sub-Threshold Region (Task 4.5). In Chapter 5 we present our initial results towards the construction of energy effective reliable data transport structures. Due to its intrinsic low power consumption we targeted the Near/Sub-Threshold operating region and proposed a dual-rail interconnection strategy, which outperforms the single-rail counterpart in terms of energy consumption at the expense of some area overhead. One remarkable feature of our proposal is the fact that it almost completely eliminates the overshot, which is very detrimental for signal integrity, thus the dual-rail interconnect we propose exhibits a built-in level of fault tolerance. Given that preliminary results are promising we plan to continue this research avenue and investigate the impact of various coding schemes, e.g., constrained coding, on the energy consumption and reliability of the proposed interconnect scheme when utilized in larger digital circuits.

Chapter 1

Analysis of Taylor-Kuznetsov Memory using One-Step Majority Logic Decoder

Abstract: *This chapter addresses the problem of constructing reliable memories from unreliable components. We consider the memory construction proposed by Taylor in which a codeword stored in a faulty memory is regularly updated by an LDPC decoder to overcome the memory degradation. We assume that the LDPC decoder used in the system is a faulty one-step majority logic decoder. Compared to [1, 2] which analyze only the faulty one-step majority logic decoder, we analyze here the reliability of the whole memory construction. We introduce a sequence of output errors probabilities at successive time instants and determine the properties and the fixed points of the sequence. From the fixed-point analysis, we define a threshold that predicts the noise level which can be tolerated for the memory to stay reliable. We finally represent the reliability regions of the Taylor-Kuznetsov memory with respect to the decoder noise parameters and validate the theoretical results with Monte-Carlo simulations.*

The work reported in this Chapter has been published in: E. Dupraz, D. Declercq, B. Vasic, "Analysis of Taylor-Kuznetsov memory using one-step majority logic decoder", *Information Theory and Applications Workshop*, San Diego, CA, USA, February 2015 [P3]

1.1 Introduction

Over the past few years, important electronic chip size reductions coupled with huge increase in integration factors have made electronic devices much more sensitive to noise. The hardware noise may introduce errors during elementary computation operations and may also affect the memory units. As a consequence, there is a need to address the issue of constructing reliable memories running on faulty hardware.

Taylor [3] and Kuznetsov [4] were the first to address the issue of constructing reliable memories built from unreliable components. In the memory architecture proposed in [3, 4], the information is stored as a codeword obtained from a Low Density Parity Check (LDPC) code. The codeword is regularly passed through an LDPC decoder in order to correct the errors introduced by the faulty hardware. As the LDPC decoders runs on the same faulty hardware as the memory, it is assumed faulty as well.

More recently, Chilappagari et al. [1, 2] and Brkic et al. [5] considered the use of a faulty One-Step Majority Logic (OS-MAJ) LDPC decoder in the memory architecture proposed by Taylor and Kuznetsov. The authors of [1, 2, 5] analyzed the Bit Error Rate (BER) performance of the OS-MAJ decoder alone, but they did not evaluate the reliability of the whole memory architecture. Vasić and Chilappagari [6] identified an equivalence between a faulty Gallager B decoder and the memory architecture proposed by Taylor and Kuznetsov. However, they evaluated the reliability of the memory only through finite-length simulations. Chilappagari et al. [7] evaluated the reliability of the memory architecture by providing an analytic expression of the maximum fraction of errors that can be corrected by the faulty LDPC decoder at successive time instants. The result in [7] was aimed at providing rigorous conditions for memory reliability, based on graph expansion arguments. However, computing expansion of large graphs is a hard problem and as a consequence, the results of [7] are not convenient for the design of a reliable memory architecture.

In this chapter, we consider the memory architecture of [1,5] with a faulty OS-MAJ decoder. In this memory architecture, discrete time instants $t = 0, \dots, T$, are considered, and a codeword obtained from an LDPC code is stored in memory at initial time instant $t = 0$. Between two time instants t and $t + 1$, the faulty hardware induces a degradation in the memory, which is represented by a memory degradation parameter α. In order to overcome the memory degradation, the codeword stored in the memory is also passed through a OS-MAJ LDPC decoder between t and $t + 1$. As the LDPC decoder runs on the same hardware as the memory, the faulty hardware introduces some noise inside the decoder. At time instant T, the information is extracted from the memory. The codeword stored in the memory at time instant T is passed through a Gallager B LDPC decoder in order to recover the information that was initially stored at time instant $t = 0$. The performance of the memory architecture can be evaluated in terms of redundancy and reliability.

The redundancy was defined in [3] as the number of noisy elements required to construct the memory architecture divided by the memory capability, that is the number of information bits stored by the memory. It is required that the redundancy does not depend on k, which induces that the complexity of the memory architecture is linear witk k, so that the memory architecture can be constructed even for large values of k. The memory is said reliable if at time instant T, the Gallager B decoder can perfectly recover the information that was stored at time instant $t = 0$.

In this chapter, we propose an analytical method to analyze the reliability of the memory architecture as a function of the memory degradation level α. We first express the error probabilities in the memory at successive time instants. Then we analyse the convergence properties of the sequence of error probabilities and introduce a threshold definition that indicates the maximum degradation level that the memory can tolerate to stay reliable. This definition enables us to represent reliability regions as a set of degradation levels and decoder noise parameters for which a reliable storage of information is possible. We also provide finite-length simulation results which validate the theoretical analysis.

The outline of the chapter is as follows. Section 1.2 presents the memory architecture and the faulty OS-MAJ decoder used in the architecture. Section 1.3 derives the sequence of error probabilities in the memory and analyzes the growing and convergence properties of the sequence. Section 1.4 introduces the reliability definition and the memory threshold definition. Section 1.4 also provides the reliability regions. Section 1.5 presents the finite-length simulation results.

1.2 Fault-Tolerant Memories

In this section, we present the memory architecture and the error model describing the memory degradation induced by the faulty hardware. The memory architecture and the error model were originally introduced in [3, 4] and latter considered in [1, 2, 5–7]. We explain how, as initially proposed by [3], LDPC codes can be used to overcome the memory degradation induced by the faulty hardware. We then describe the faulty OS-MAJ LDPC decoder used as a correction circuit.

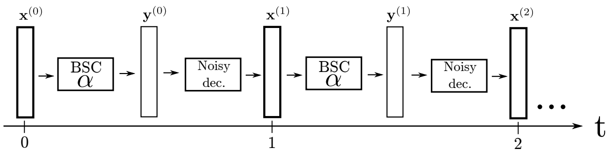


Figure 1.1: Information stored in the memory at successive time instants

1.2.1 Memory Degradation

Consider a memory with a storage capability of k bits, and consider the discrete time instants $t = 0, \ldots, T$. Denote by $\mathbf{x}^{(0)}$ the binary information vector of length k initially stored in memory at time instant $t = 0$, and denote by $\mathbf{x}^{(t)}$ the binary information vector of length k that is in the memory at time instant t. Let $x_v^{(t)}$ be the v-th component of the vector $\mathbf{x}^{(t)}$. The memory degradation between two successive time instants t and $t+1$ is modeled by a Binary Symmetric Channel (BSC) of parameter α, which is denoted BSC(α). The BSC gives a symmetric and memoryless error model. Although such a model may not take into account all the errors induced by the faulty hardware in a realistic memory, we consider it here as a first step for the analysis.

Unfortunately, because of the memory degradation, the number of errors in $\mathbf{x}^{(t)}$ with respect to $\mathbf{x}^{(0)}$ increases with t. For large enough t, $\mathbf{x}^{(t)}$ will contain too many errors, and it will not be possible to recover the initial $\mathbf{x}^{(0)}$ from $\mathbf{x}^{(t)}$ anymore. In order to overcome this effect, the information vector is encoded by an LDPC code, as described in the following.

1.2.2 Taylor-Kuznetzov Memory Architecture

Le $\mathbf{x}^{(0)}$ be a codeword obtained from an LDPC code of dimension k defined by a parity check matrix H of size $m \times n$, with $k \leq m - n$. The vector $\mathbf{x}^{(0)}$ is stored in the memory at time instant $t = 0$, and the memory has a storage capability of k information bits. The Tanner graph of the code is composed of n Variable Nodes (VN) $v \in \{1, \ldots, n\}$ and m Check Nodes (CN) $c \in \{1, \ldots, m\}$. The degree of the VN v is denoted as d_v and the degree of the CN c is denoted as d_c. Here, the code is assumed to be regular, *i.e*, d_v does not depend on v, and d_c does not depend on c. Denote by $\mathcal{N}(v)$ the set of CNs connected to the VN v, and denote by $\mathcal{N}(c)$ the set of VNs connected to the CN c.

Between two successive time instants t and $t+1$, the vector $\mathbf{x}^{(t)}$ stored in the memory undergoes two operations, as depicted in Figure 1.1. First, $\mathbf{x}^{(t)}$ is passed through BSC(α), which gives the degraded vector $\mathbf{y}^{(t)}$. Second, $\mathbf{y}^{(t)}$ is passed through an LDPC decoder called the refresh decoder. The refresh decoder reads the memory content $\mathbf{y}^{(t)}$ at time t and outputs the vector $\mathbf{x}^{(t+1)}$ which is written back in the memory and stored in memory at instant $t+1$. The refresh decoder has to correct most of the errors introduced by the BSC between two time instant t and $t+1$, so that $\mathbf{x}^{(t)}$ is never too far away from $\mathbf{x}^{(0)}$. Here, the refresh decoder is a OS-MAJ decoder. We choose a very low complexity OS-MAJ decoder in order to limit the redundancy of the memory architecture. In addition, as the refresh decoder runs on the same faulty hardware as the memory, it will be assumed faulty as well.

Finally, when the information has to be extracted from the memory at time instant T, we allow the use of a second LDPC decoder called the final decoder. The final decoder has to reconstruct almost perfectly $\mathbf{x}^{(0)}$ from $\mathbf{x}^{(T)}$. Here, the final decoder is a noiseless Gallager B decoder. Indeed, as the final decoder is used only once at the end of the storage, we allow it to be both noiseless and more complex compared to the refresh decoder.

1.2.3 Refresh Decoder: Faulty OS-MAJ Decoder

We first describe the noiseless version of the OS-MAJ decoder. At time instant t, the refresh decoder receives the degraded vector $\mathbf{y}^{(t)}$. As a first step of the decoding, each CN c computes a message $\gamma_{c\to v}$ for all the VNs $v \in \mathcal{N}(c)$ connected to it. Let $A = \{a_1, \dots, a_d\}$ be a multiset of d binary digits and define the function realizing the XOR sum of the d terms of A as

$$f_{\oplus}(A) = a_1 \oplus \dots \oplus a_d. \tag{1.1}$$

The CN messages are calculated from the function $f_\oplus$ as $\forall c \in \{1, \dots, m\}$, $\forall v \in \mathcal{N}(c)$,

$$\gamma_{c\to v} = f_{\oplus}\left(\{y_{v'}^{(t)}\}_{v'\in\mathcal{N}(c)\setminus v}\right). \tag{1.2}$$

From the CN messages it receives, each VN $v \in \{1, \dots, n\}$ computes a decision value η_v. Let $B = \{b_1, \dots, b_d\}$ be a binary multiset of size d, and let a be a binary digit. We define the majority voting function as

$$f_{\text{maj}}(B, a) = \begin{cases} 1 & \text{if } |\text{supp}(B)| > \frac{d_v}{2} \\ 0 & \text{if } |\text{supp}(\bar{B})| > \frac{d_v}{2} \\ a & \text{otherwise} \end{cases} \tag{1.3}$$

where supp(B) is the support of B, $\bar{B}$ is the complement of B, and $|.|$ is the cardinality of a set. The decision values are calculated from the function f_{maj} as $\forall v \in \{1, \dots, n\}$,

$$\eta_v = f_{\text{maj}}\left(\{\gamma_{c\to v}\}_{c\in\mathcal{N}(v)}, y_v^{(t)}\right). \tag{1.4}$$

At the end of the decoding, each VN is set as $x_v^{(t+1)} = \eta_v$. The resulting vector $\mathbf{x}^{(t+1)}$ corresponds to the vector stored in memory at time instant $t+1$.

We now describe the faulty version of the OS-MAJ decoder. Denote by $\tilde{\gamma}_{c\to v}$ the noisy versions of the CN messages $\gamma_{c\to v}$ in (1.2), and by $\tilde{\eta}_v$ the noisy versions of the decision values η_v. In order to obtain a faulty version of the OS-MAJ decoder from the above noiseless description, we replace the noiseless functions $f_\oplus$ in (1.1) and f_{maj} in (1.3) by their noisy versions $\tilde{f}_\oplus$ and $\tilde{f}_{\text{maj}}$. in (1.4).

The noisy XOR sum function $\tilde{f}_\oplus$ is defined as

$$\tilde{f}_{\oplus}(A) = f_{\oplus}(a_1, \dots, a_d) \oplus e_{\oplus} \tag{1.5}$$

where $e_\oplus$ is a random variable distributed according to the Bernoulli distribution with parameter $p_\oplus$. The parameter $p_\oplus$ represents the error probability of the function. As for the memory degradation effect, the error model for the XOR sum function is assumed memoryless and symmetric, as a first step of the analysis. With the definition of the function $\tilde{f}_\oplus$ in (1.5), we assume that the noise applies only at the end of the whole XOR sum computation. Another model would be to assume that the CN computation is realized from $(d-1)$ 2-inputs faulty XOR gates, each with error probability $p_{\text{xor},2}$. However, the two models are equivalent as we have the relation $p_\oplus = P(\tilde{f}_\oplus(a_1, \dots, a_d) \neq f_\oplus(a_1, \dots, a_d)) = \frac{1}{2} - \frac{1}{2}(1 - p_{\text{xor},2})^{(d-1)}$. It thus suffices to calculate the parameter $p_\oplus$ from $p_{\text{xor},2}$ to obtain the error model defined by the function$\tilde{f}_\oplus$ in (1.5).

The noisy majority voting function $\tilde{f}_{\text{maj}}$ is defined as

$$\tilde{f}_{\text{maj}}(B, a) = f_{\text{maj}}(b_1, \dots, b_d, a) \oplus e_{\text{maj}}, \tag{1.6}$$

where e_{maj} is a random variable distributed according to the Bernoulli distribution with parameter p_{maj}. The parameter p_{maj} represents the error probability of the majority voting function.

As for the XOR sum computation, the noise is assumed to be only at the end of function computation. While this model may not capture all the noise effects that could appear inside the majority voting function, it does not require knowledge of a particular hardware implementation of the function. However, it appears sufficient for the first step of the analysis and more accurate models will be considered in future works. Note that in the faulty OS-MAJ decoder considered in [1,2,5] the majority voting functions are noiseless and only the XOR gates are assumed faulty.

At the end, because of the refresh decoder, the memory constructed from faulty components requires more components than the memory built from reliable units. In order to evaluate the amount of induced additional complexity, the redundancy of the memory architecture can be evaluated as follows.

1.2.4 Redundancy of the Memory Architecture

The redundancy of the memory is defined in [3] as the number of noisy components used in the memory architecture divided by the memory capability k. For the memory architecture considered in the paper, the redundancy of the memory is expressed as [7]

$$\mathrm{R_{ed}} = \frac{1 + D + d_v(d_c - 2)}{1 - \frac{d_v}{d_c}}. \tag{1.7}$$

where D is the complexity of the majority voting unit and D depends only on d_v. In (1.7), the final decoder is no taken into account as it is used only once at the end of the storage. The redundancy depends only on the code parameters d_v, d_c, but does not depend on the memory capability k. As a result, the complexity of the memory architecture built from unreliable components is only linear with the memory capability.

Now, we would like to determine whether and for which parameters α, $p_{\oplus}$, p_{maj}, the considered memory architecture is reliable. Thus for a fixed code and a given redundancy, we now analyze the reliability of the considered memory architecture with respect to the memory degradation parameter α and to the decoder noise parameters $p_{\oplus}$, p_{maj}.

1.3 Error Probability Evaluation

In this section, we analyze the reliability of the memory by expressing the bit error probabilities in the successive vectors $\mathbf{x}^{(t)}$ stored in the memory at time instants $t = 0, \dots, T$. We first express analytically the error probability of the faulty OS-MAJ refresh decoder. We then use this probability to derive the successive error probabilities in the $\mathbf{x}^{(t)}$.

1.3.1 Error Probability of the Faulty OS-MAJ Decoder

Here, we find the error probability of the OS-MAJ decoder as a function of the memory degradation level at the input of the decoder. The error probability of the faulty OS-MAJ decoder was given in [8,9] for different decoder noise error models. We restate it here for the error model we consider in the paper. For now, assume that the input degradation level is α, which corresponds to the degradation level in the first time interval between $t = 0$ and $t = 1$.

The BSC representing the memory degradation and the faulty functions $\tilde{f}_{\oplus}$ and $\tilde{f}_{\mathrm{maj}}$ used in the decoder are symmetric in the sense of [10]. Thus, from [10], we can assume that the all-zero codeword was initially stored in memory, which greatly simplifies the analysis. From the all-zero codeword assumption, we obtain the error probability of the Majority Logic decoder by expressing the probabilities of the messages exchanged during the decoding.

Denote by $\tilde{p}_{\gamma} = P(\tilde{\gamma}_{c \to v} = 1)$ the probability of a noisy CN message $\tilde{\gamma}_{c \to v}$. Denote by $p_{\eta} = P(\eta_v = 1)$ the probability of a noiseless decision value η_v, and denote by $\tilde{p}_{\eta} = P(\eta_v = 1)$

the probability of its noisy version. The probability $\tilde{p}_\gamma$ of the noisy CN message $\tilde{\gamma}_{c\to v}$ can be written as

$$\tilde{p}_\gamma = \frac{1}{2} - \frac{1}{2}(1-2p_\oplus)(1-\alpha)^{(d_c-1)}. \tag{1.8}$$

It corresponds to the probability of an occurence of odd number of ones among the (d_c-1) inputs of the CN, or of an error in the XOR sum computation. The probability p_η of the noiseless decision value η_v is calculated depending on the parity of the VN degree d_v. If d_v is odd, p_η can be written as

$$p_\eta = \sum_{j=\lceil \frac{d_v}{2} \rceil}^{d_v} \binom{d_v}{j} \tilde{p}_\gamma^j (1-\tilde{p}_\gamma)^{(d_v-j)} \tag{1.9}$$

If d_v is even, we get

$$\begin{aligned} p_\eta = &\sum_{j=\frac{d_v}{2}}^{d_v} \binom{d_v}{j} \tilde{p}_\gamma^j (1-\tilde{p}_\gamma)^{(d_v-j)} \\ &- (1-\alpha)\binom{d_v}{\frac{d_v}{2}} \tilde{p}_\gamma^{\left(\frac{d_v}{2}\right)} (1-\tilde{p}_\gamma)^{\left(\frac{d_v}{2}\right)}. \end{aligned} \tag{1.10}$$

In both cases, p_η corresponds to the probability of a majority of 1 digits at the input of the noiseless majority voting unit.

We now express the error probability of the decoder. Denote $\nu = (p_\oplus, p_{\text{maj}})$ the decoder noise parameter pairs. The error probability $P_{e,\nu}(\alpha) = P(x_v^{(1)} = 1 | x_v^{(0)} = 0)$ of the faulty OS-MAJ decoder can be calculated from p_η as

$$P_{e,\nu}(\alpha) = p_\eta(1-p_{\text{maj}}) + (1-p_\eta)p_{\text{maj}}. \tag{1.11}$$

The error probability $P_{e,\nu}(\alpha)$ of the decoder is equal to the probability $\tilde{p}_\eta = P(\eta_v = 1)$ of the noisy decision value η_v. The error probability $P_{e,\nu}(\alpha)$ depends on the input noise level α, and on the decoder noise parameters $p_\oplus, p_{\text{maj}}$.

The expression of the error probability $P_{e,\nu}(\alpha)$ in (1.11) can now be used to evaluate the successive error probabilities in the memory.

1.3.2 Successive Error Probabilities in the Memory

In this section, we want to derive the expressions of the error probabilities in the $\mathbf{x}^{(t)}$ at successive time instants $t = 0, \ldots, T$. In order to do this, we also give the expressions of the degradation levels in the $\mathbf{y}^{(t)}$ at successive time instants $t = 0, \ldots, T$. As before, from the symmetry of the functions and error models considered in the memory architecture, we can assume that the all-zero codeword was initially stored in memory. According to the memory architecture defined in Section 1.2, the successive degradation levels in the $\mathbf{y}^{(t)}$ and the successive error probabilities in the $\mathbf{x}^{(t)}$ are given in the following proposition.

Proposition 1. *Denote $\beta_\nu^{(t)}(\alpha)$ the degradation level in $\mathbf{y}^{(t)}$ with respect to $\mathbf{x}^{(0)}$,* i.e., *$\beta_\nu^{(t)}(\alpha) = P(y_v^{(t)} = 1 | x_v^{(t)} = 0)$. The successive degradation levels $\beta_\nu^{(t)}(\alpha)$ can be expressed recursively as*

$$\beta_\nu^{(1)}(\alpha) = \alpha, \tag{1.12}$$

and $\forall t > 1$,

$$\beta_\nu^{(t)}(\alpha) = (1-\alpha)P_{e,\nu}\left(\beta_\nu^{(t-1)}(\alpha)\right) + \alpha\left(1 - P_{e,\nu}\left(\beta_\nu^{(t-1)}(\alpha)\right)\right). \tag{1.13}$$

The error probabilities in the successive $\mathbf{x}^{(t)}$ *are given by* $\delta_\nu^{(t)}(\alpha) = P(x_v^{(t)} = 1 | x_v^{(0)} = 0)$, *and*

$$\delta_\nu^{(t)}(\alpha) = P_{e,\nu}\left(\beta_\nu^{(t)}(\alpha)\right), \ \forall t \geq 1. \tag{1.14}$$

The initial $\beta_\nu^{(1)}(\alpha)$ is given by the fact that $\mathbf{y}^{(1)}$ is the output of BSC(α). At time instant $(t-1)$, the stored vector $\mathbf{x}^{(t-1)}$ has error probability $P_{e,\nu}\left(\beta_\nu^{(t-1)}(\alpha)\right)$ and is passed through BSC(α). The resulting $\mathbf{y}^{(t-1)}$ can be seen as the output of the concatenation of BSC(α) and of BSC$\left(P_{e,\nu}\left(\beta_\nu^{(t-1)}(\alpha)\right)\right)$, respectively, which gives the expression of $\beta_\nu^{(t)}(\alpha)$ in (1.13). The faulty decoder then produces $\mathbf{x}^{(t)}$ from its input $\mathbf{y}^{(t)}$, and as a result, the error probabilities in the successive $\mathbf{x}^{(t)}$ are given by the $\delta_\nu^{(t)}(\alpha)$ in (1.14).

Proposition 1 gives the recursive expression of the sequences $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ and $\{\delta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ of error probabilities in the memory. In the following, we analyze the sequence of $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ instead of the sequence of $\{\delta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$. Indeed, we are more interested in the degradation levels that the memory can tolerate than in the successive error probabilities. This is in compliance with the conventional analysis of LDPC decoders in which we define a threshold on the channel parameter.

The memory will be reliable if the successive degradation levels are small enough so that at any time instant, we can guarantee that $\mathbf{x}^{(t)}$ is in a close proximity of $\mathbf{x}^{(0)}$ and can be recovered by a perfect Gallager B decoder. In order to be able to check this condition for various values of α, $p_\oplus$, p_{maj}, and for different choices of LDPC codes, we first analyze the increasing and convergence properties of the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$.

1.3.3 Sequence Properties

In this section, we first analyze the increasing properties of the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$. The properties of the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ are stated in the following Proposition.

Proposition 2. *Consider the sequence* $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ *given in Proposition 1.*

1. *Fix* $p_\oplus$ *and* p_{maj}. *If the function* $\alpha \to P_{e,\nu}(\alpha)$ *is increasing with* α, *then*

$$\forall \alpha < 1/2, \quad \beta_\nu^{(t)}(\alpha) \leq \beta_\nu^{(t+1)}(\alpha). \tag{1.15}$$

2. *Fix* $p_\oplus$ *and* p_{maj}. *If the function* $\alpha \to P_{e,\nu}(\alpha)$ *is increasing with* α, *then*

$$\forall t > 0, \quad \alpha_1 \leq \alpha_2 \ \Rightarrow \ \beta_\nu^{(t)}(\alpha_1) \leq \beta_\nu^{(t)}(\alpha_2). \tag{1.16}$$

3. *Fix* α, p_{maj}, *and denote* $\nu_1 = (p_{xor,1}, p_{maj})$, $\nu_2 = (p_{xor,2}, p_{maj})$. *If the function* $p_\oplus \to P_{e,\nu}(\alpha)$ *is increasing with* $p_\oplus$, *then*

$$\forall t > 0, \quad p_{xor,1} \leq p_{xor,2} \ \Rightarrow \ \beta_{\nu_1}^{(t)}(\alpha) \leq \beta_{\nu_2}^{(t)}(\alpha). \tag{1.17}$$

4. *Fix* α, $p_\oplus$, *and denote* $\nu_1 = (p_\oplus, p_{maj,1})$, $\nu_2 = (p_\oplus, p_{maj,2})$. *If the function* $p_{maj} \to P_{e,\nu}(\alpha)$ *is increasing with* p_{maj}, *then*

$$\forall t > 0, \quad p_{maj,1} \leq p_{maj,2} \ \Rightarrow \ \beta_{\nu_1}^{(t)}(\alpha) \leq \beta_{\nu_2}^{(t)}(\alpha). \tag{1.18}$$

Proof.

1) The proof is made recursively.

First compute $\beta_\nu^{(2)}(\alpha) = \alpha(1-2P_{e,\nu}(\alpha)) + P_{e,\nu}(\alpha) \geq \alpha$. The inequality come from $(1-2P_{e,\nu}(\alpha)) > 0$. As $\beta_\nu^{(1)}(\alpha) = \alpha$, we get $\beta_\nu^{(2)}(\alpha) \geq \beta_\nu^{(1)}(\alpha)$.

Then assume that $\beta_\nu^{(t)}(\alpha) \geq \beta_\nu^{(t-1)}(\alpha)$ and compute

$$\begin{aligned}
&\beta_\nu^{(t+1)}(\alpha) - \beta_\nu^{(t)}(\alpha) \\
&= (1-2\alpha)\left(P_{e,\nu}\left(\beta_\nu^{(t)}(\alpha)\right) - P_{e,\nu}\left(\beta_\nu^{(t-1)}(\alpha)\right)\right) \\
&\geq\ 0.
\end{aligned}$$

The inequality come from the above assumption. At the end, we get $\beta_\nu^{(t+1)}(\alpha) \geq \beta_\nu^{(t)}(\alpha)$, which proves (1.15).

2) Compute

$$\begin{aligned}
&\beta_\nu^{(t+1)}(\alpha_2) - \beta_\nu^{(t+1)}(\alpha_1) \\
&= (1-2\alpha_1)P_{e,\nu}\left(\beta_\nu^{(t)}(\alpha_1)\right) + (1-2\alpha_2)P_{e,\nu}\left(\beta_\nu^{(t)}(\alpha_2)\right) \\
&\quad + (\alpha_1 - \alpha_2) \\
&\geq 0,
\end{aligned}$$

which proves (1.16)

3) The proof is made recursively. We first show that $P_{e,\nu}\left(\beta_{\nu_1}^{(1)}(\alpha)\right) \leq P_{e,\nu}\left(\beta_{\nu_2}^{(1)}(\alpha)\right)$ and that $\beta_{\nu_1}^{(2)}(\alpha_2) \leq \beta_{\nu_2}^{(2)}(\alpha)$. We then assume that at step t, $P_{e,\nu}\left(\beta_{\nu_1}^{(t)}(\alpha)\right) \leq P_{e,\nu}\left(\beta_{\nu_2}^{(t)}(\alpha)\right)$ and $\beta_{\nu_1}^{(t)}(\alpha_2) \leq \beta_{\nu_2}^{(t)}(\alpha)$, and we show that this is also true at step $t+1$.

The proof for 4) is the same as the proof for 3).

□

Proposition 2 assumes that the function $P_{e,\nu}(\alpha)$ is increasing with α and with the decoder noise parameters. Although it is reasonable to assume that the error probability of the faulty decoder increases with the BSC parameter and with the decoder noise parameters,the results of [11, 12] show that the second assumption is not always true. For example, for the discrete Min-Sum decoder with 7 quantization levels for the messages, the authors of [11] observe that the noise in the decoder can sometimes improve the decoder performance compared to the noiseless case. The same effect is observed for the Probabilistic Gradient Descent Bit-Flipping decoders introduced in [12]. As a consequence, Proposition 2 does not hold for such decoders. On the other hand, for the faulty OS-MAJ decoder, we can show that the function $P_{e,\nu}(\alpha)$ is increasing with α and with the decoder noise parameters. As a consequence, for the memory architecture we consider in the chapter, higher values of α, $p_\oplus$, p_{maj}, will lead to increased degradation levels $\beta_\nu^{(t)}(\alpha)$.

Proposition 2 also shows that the successive degradation levels $\beta_\nu^{(t)}(\alpha)$ are increasing with t. As a result, even with the refresh decoder, the faulty memory keeps degrading the stored information. However, we hope that the sequence of degradation levels $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ converges to a fixed point which is no too high, so that the initial $\mathbf{x}^{(0)}$ can always be recovered from $\mathbf{x}^{(t)}$, even for large values of t. In order to verify this condition, we now analyze the convergence behavior of $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$.

1.3.4 Fixed-point analysis

Here, we analyze the asymptotic behavior of $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ by determining the fixed points of the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$. The fixed points of the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ are the values β satisfying

$\beta = (1-\alpha)P_{e,\nu}(\beta) + \alpha(1-P_{e,\nu}(\beta))$, or equivalently if $\alpha \neq 1/2$,

$$P_{e,\nu}(\beta) = \frac{\beta - \alpha}{1 - 2\alpha}. \tag{1.19}$$

From the condition (1.19), the fixed points of the sequence of $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ correspond to the intersection of the curve representing $P_{e,\nu}(\beta)$ and the straight line $y = \frac{\beta-\alpha}{1-2\alpha}$. This gives a very simple condition to determine the fixed points of $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$. Note that if $P_{e,\nu}(1/2) = 1/2$ (which is always satisfied), then $\beta = 1/2$ is always a fixed point of $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$, whatever the value of α is.

The fixed points correspond to the possible limits of the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$. We know that $\beta = 1/2$ is always a fixed point, but it is a bad one for which we cannot recover the original $\mathbf{x}^{(0)}$ from $\mathbf{x}^{(t)}$. Thus, we hope that the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ has other fixed points which correspond to degradation levels that can be handled by the final decoder. In the following, we propose a definition of the memory reliability that accounts for this condition.

1.4 Reliability Conditions

In this section, we give a definition of the reliability of the memory. The reliability definition relies on the above asymptotic analysis on the sequence of degradation levels $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$. From the reliability conditions, we propose a threshold definition that determines the set of degradation parameters α that lead to a reliable storage of information.

1.4.1 Reliability Conditions

The reliability conditions we give here are based on the reliability conditions originally introduced in [3, Section 2.2]. The following definition adapts the reliability conditions of [3] to our analysis of the convergence properties of the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$.

Definition 1. *Consider the memory architecture of Section 1.2 and fix the parameters α, $p_\oplus$, p_{maj}. Consider the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ given in Proposition 1. Denote $\mathcal{B}$ the set of fixed-points of $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ excluding $1/2$, and denote by $\beta^\star$ the threshold of the noiseless Gallager B decoder.*

A memory is said to be reliable *for the parameters α, $p_\oplus$, p_{maj}, if the following three conditions are verified*

1. Bounded redundancy*: The redundancy R_{ed} of the memory does not depend on the memory capability k,*
2. Stability*: The set $\mathcal{B}$ is nonempty,*
3. Admissibility*: The set $\mathcal{B}$ is such that* $\max \mathcal{B} \leq \beta^\star$.

The condition 1 requires bounded redundancy. From (1.7), The condition 1 is always fulfilled. The conditions 2 and 3 are related to the asymptotic analysis of the successive degradation levels in the memory. The condition 2 requires that the sequence $\{\beta_\nu^{(t)}(\alpha)\}_{t=1}^{+\infty}$ has fixed points other than $\beta = 1/2$. It guarantees that the memory is stable in the sense that the degradation levels converge to a fixed point. The condition 3 ensures that the fixed point corresponds to a degradation level that can be handled by the final decoder.

The validity of Conditions 2 and 3 depend on the value of the memory degradation parameter α, and on the decoder noise parameters $p_\oplus$ and p_{maj}. In order to identify the set of parameters α, $p_\oplus$, p_{maj}, that lead to a reliable memory, we introduce a threshold definition as follows.

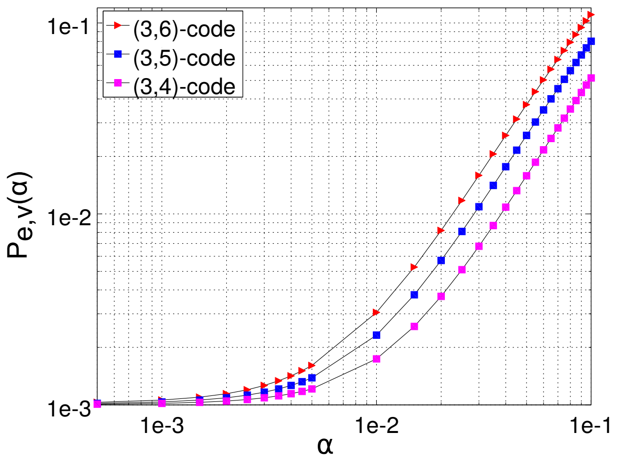


Figure 1.2: Error probabilities w.r.t. α, for $p_{\oplus} = 10^{-3}$, $p_{\mathrm{maj}} = 10^{-3}$ and Codes with $d_v = 3$

1.4.2 Threshold Definition

The following threshold definitions were introduced for LDPC codes in channel coding. The noiseless threshold in [13] was defined as the maximum channel parameter α such that $P_{e,\nu}(\alpha) = 0$. This condition cannot be applied here because of the noise introduced by the faulty hardware, which prevents the decoder from reaching an error probability 0. This is why several other threshold definitions were introduced for noisy decoders: the useful threshold [14], the target-BER threshold [14, 15], and the functional threshold [16]. However, these threshold definitions cannot be used in our context, because they characterize the behavior of the faulty decoder alone. Here, we introduce a new threshold definition that takes into account the dynamic of the whole memory architecture.

Definition 2. *Consider the memory architecture of Section 1.2 and fix the decoder noise parameters $p_{\oplus}$, p_{maj}. The* degradation threshold *is defined as*

$$\overline{\alpha} = \arg\max_{\alpha} \{\textit{The memory is reliable in the sense of Definition 1}\}. \tag{1.20}$$

The degradation threshold is defined as the maximum parameter α for which the memory is reliable. In the above definition, the decoder noise parameters $p_{\oplus}$ and p_{maj} are fixed, and the threshold is only on α. Indeed, α is the parameter of the memory, while $p_{\oplus}$ and p_{maj} are the parameters of the correction circuit, and we want to express the threshold in terms of reliability of the memory elements.

At the end, the degradation threshold enables to characterize the set of parameters α, $p_{\oplus}$, p_{maj}, that lead to a reliable memory. These parameters can be represented in the form of reliability regions, as described in the following.

1.4.3 Reliability Regions

In this section, we provide reliability regions as the set of parameters α, $p_{\oplus}$, p_{maj}, that lead to a reliable memory. We consider regular LDPC codes of VN degree $d_v = 3$ and CN degrees $d_c = 4$, $d_c = 5$, $d_c = 6$, respectively. For the faulty OS-MAJ decoder, we set $p_{\oplus} = 10^{-3}$, and $p_{\mathrm{maj}} = 10^{-3}$. In order to verify the reliability of the memory for given parameters α, $p_{\oplus}$, p_{maj}, we need the expression $P_{e,\nu}(\alpha)$ of the error probability of the faulty OS-MAJ decoder. Thus we first discuss the curves representing $P_{e,\nu}(\alpha)$ as a function of α.

The error probabilities $P_{e,\nu}(\alpha)$ are calculated from (1.11). Figure 1.2 shows $P_{e,\nu}(\alpha)$ as a function of α for the codes of VN degree $d_v = 3$. We see that the error probability increases with α but does it very slowly for small α. However, there is no distinguishable threshold value on α that would separate the low and the high error probability regions. However, from the analysis

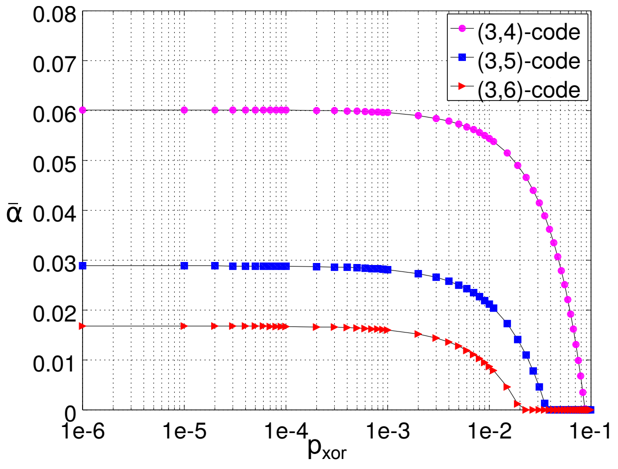


Figure 1.3: Reliability regions w.r.t $p_{\oplus}$, for $p_{\mathrm{maj}} = 10^{-3}$

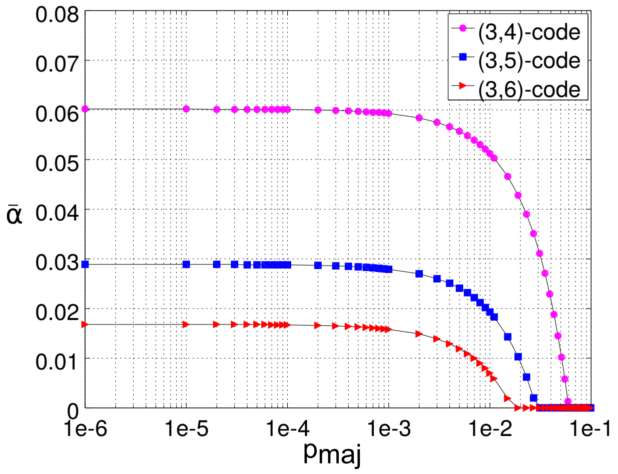


Figure 1.4: Reliability regions w.r.t p_{maj}, for $p_{\oplus} = 10^{-3}$

carried in the chapter, we can identify a memory threshold by analyzing the properties of the sequence $\{\beta_{\nu}^{(t)}(\alpha)\}_{t=1}^{+\infty}$, as illustrated in the following.

For the codes with $d_v = 3$, Figure 1.3 represents the threshold values $\bar{\alpha}$ obtained from Definition 2 with respect to $p_{\oplus}$ Figure 1.3 thus gives the reliability regions with respect to $p_{\oplus}$. As expected, the reliability regions shrink with the code rate increase. The reliability regions are convex, even for large values of $p_{\oplus}$. When the decoder noise parameter $p_{\oplus}$ becomes too large, the threshold value $\bar{\alpha}$ becomes 0, which means that the decoder noise is too high to enable the memory to be reliable . Figure 1.4 represents the reliability regions with respect to p_{maj}. We get the same conclusions as before.

We conclude this section by a remark that the analysis of the chapter and the threshold definition enable characterization of the set of parameters α, $p_{\oplus}$, p_{maj}, for which the memory is reliable. To verify the accuracy of the characterization, we now give finite-length simulation results.

1.5 Finite-length Simulations

In this section, we evaluate at finite length the reliability of the memory architecture we consider in the chapter. We consider regular LDPC codes of dimension $k = 400$ with VN degree $d_v = 3$ and CN degrees $d_c = 4$, $d_c = 5$, $d_c = 6$, respectively.

Figure 1.5 represents the BER of the Faulty OS-MAJ Decoder for the three considered codes

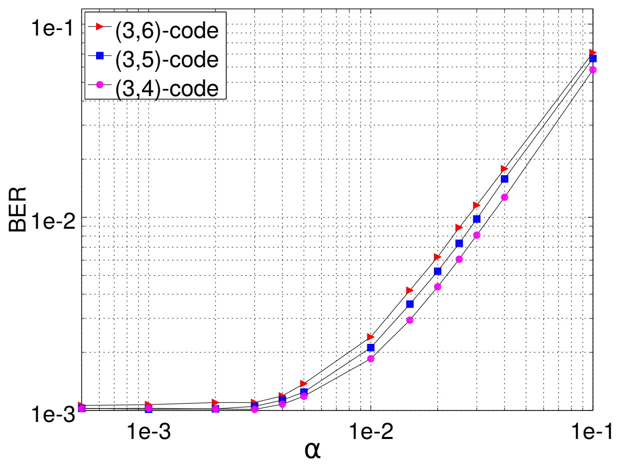


Figure 1.5: BER w.r.t. α of the faulty OS-MAJ Decoder with $p_{\oplus} = 10^{-3}$ and $p_{\text{maj}} = 10^{-3}$

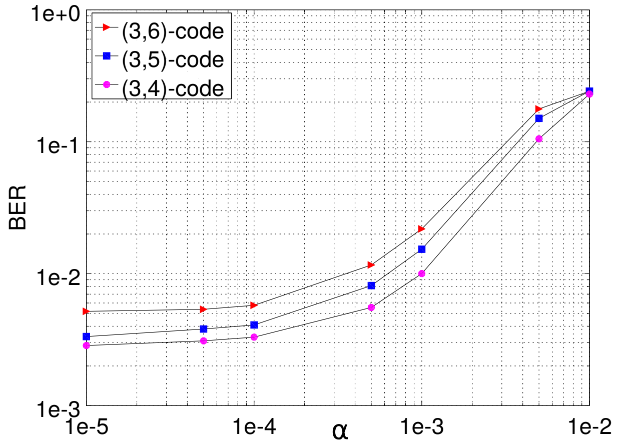


Figure 1.6: BER w.r.t. α of the memory architecture with $p_{\oplus} = 10^{-3}$ and $p_{\text{maj}} = 10^{-3}$ and $T = 200$

with $p_{\oplus} = 10^{-3}$ and $p_{\text{maj}} = 10^{-3}$. The curves are very similar to the curves of Figure 1.2 that represent the error probabilities with respect to α. The only difference is that the curves of Figure 1.5 are closer to each other than the curves of Figure 1.2.

Figure 1.6 represents the BER for the whole memory architecture after $T = 200$ time instants for the three codes with $p_{\oplus} = 10^{-3}$ and $p_{\text{maj}} = 10^{-3}$. As expected, when the rate of the code increases, the BER increases. We see that when α becomes too large, the BER after $T = 200$ is too high and the memory is not reliable anymore. The value α for which the memory is not reliable anymore should correspond to the threshold value $\bar{\alpha}$. However, here, for the three considered codes, this value α is smaller than the threshold. Indeed, the analysis carried in the chapter is an asymptotic analysis and as a consequence, the finite-length results are more pessimistic. Increasing the size of the memory may reduce the gap between the asymptotic results and the finite-length results.

Figure 1.7 represents the BER for the whole memory architecture after $T = 200$ time instants for the code with $d_c = 6$, for several values of $p_{\oplus}$ and p_{maj}. As expected, the memory is less reliable when the decoder noise increases. In particular, for $p_{\oplus} = p_{\text{maj}} = 5 \times 10^{-2}$ and $p_{\oplus} = p_{\text{maj}} = 10^{-2}$, the memory is not reliable, whatever the value of α.

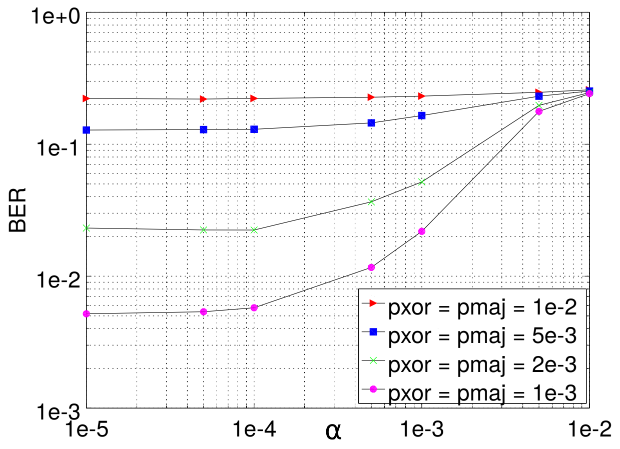


Figure 1.7: BER w.r.t. α of the memory architecture with $p_{\oplus} = 10^{-3}$ and $p_{\text{maj}} = 10^{-3}$ and $T = 200$

1.6 Conclusion

In this chapter, we provided an analysis of the reliability of the memory architecture proposed by Taylor [17] and Kuznetsov [18]. We expressed the successive error probabilities in the memory and we introduced a threshold definition to characterize the set of memory degradation parameters and decoder noise parameters that lead to a reliable memory. The results of the paper can be extended to irregular codes and to other refresh decoders, such as faulty Gallager A or B decoders with a small number of iterations.

Chapter 2

Reliable Memories Built of Unreliable Components: Majority-Logic Decoding and Analysis of Data-Dependent Logic Gate Failures

Abstract: *In this chapter we propose a memory architecture build from unreliable components. The data are stored as codewords of an error correction code and corrected by an one-step majority logic decoder between periodic updates of memory registers. The decoder is also made of unreliable logic gates whose failures are transient and data-dependent. A closed-form expression for the average residual bit error rate after one memory update cycle derived in [5] is used for finding bounds on the one-step majority logic decoder performance under the gate timing errors. In addition, we investigate the ability of the proposed memory to store information over time. We proved that the proposed memory architecture is reliable under the adversarial failure model. We illustrate the results by numerical examples of memory architectures which employ finite geometry codes.*

The work presented in this chapter has been submitted for publication in: S. Brkic, P. Ivanis, B. Vasic, "Reliable memories built of unreliable components: majority-logic decoding and analysis of data-dependent logic gate failures", *IEEE Transactions on Communications* (submitted) [P5]

2.1 Introduction

Increased integration factor of integrated circuits together with stringent energy-efficiency constraints result in an increased unreliability of today's semiconductor devices. As a result of supply voltage reduction and the process variations effects, a fully reliable operation of hardware components cannot be guaranteed [19]. Error control coding as a method for adding redundancy to ensure fault-tolerance of systems build of unreliable hardware was introduced in the late sixties and early seventies by Taylor [3] and Kuznetsov [4] in the context of reliable storage. In their memory system, the information sequence, encoded by a low-density parity-check (LDPC) code, is stored in unreliable memory registers, which are periodically updated using a "noisy" correcting circuit. It was proved that, under the so called von Neumann failure model, such a memory even with a number of redundant gates linear in memory size is capable of achieving arbitrary small error probability [3], i.e. has non-zero storage capacity.

The equivalence between Taylor-Kuznetsov (TK) fault-tolerant memory architectures and a Gallager-B decoder built from unreliable logic gates was first observed by Vasić *et al.* in [20] and [7], and developed by Vasić and Chilappagari [6] into a theoretical framework for analysis and design of faulty decoders of LDPC codes and using them to build reliable memories made of unreliable components. Recent research in this area has been aimed to two main directions: (*i*) achieving reliable storage and (*ii*) reliable transmission of information.

The modified TK scheme was introduced by Ivkovic *et al.* in [1], where it was shown that by adding a simple syndrome checker, build from reliable hardware, the performance of TK scheme can be significantly improved. The existence of a reliable memory based on a modified bit-flipping algorithm was shown by Chilappagari and Vasić in [21] and [20], who proposed a memory architecture that can tolerate a fixed fraction of errors in registers and logic gates based on expander LDPC codes.

Performance of the ensemble of LDPC codes under faulty iterative decoding was studied by Varshney in [14], who showed that, under the von Neumann failure model, the density evolution technique is applicable to faulty decoders which he used to examine the performance of faulty Gallager-A and belief-propagation algorithms. Density evolution analysis of the noisy Gallager-B decoder was presented in the series of complementing papers by Yazdi *et al.* in [22] and [23] and by Huang *et al.* in [24]. In [22] the authors studied the performance of the binary Gallager-B decoder used to decode irregular LDPC codes and proposed optimal resource allocation of noisy computational units, i.e. variable and check nodes of varying degrees, in order to achieve minimal error rate. The faulty decoder of non-binary regular LDPC codes was analyzed in [23] in the presence of von Neumann errors. In [24] a more complicated failure model was considered, which includes transient errors and permanent memory errors. Similar analysis was done by Leduc-Primeau and Gross in [25], where the faulty Gallager-B decoder improved by message repetition scheme was studied. More general finite-alphabet decoders were investigated by Huang and Dolecek in [26], while a noisy min-sum decoder realization was considered by Ngassa *et al.* in [27] and by Balatsoukas-Stimming and Burg in [15]. Dupraz *et al.* [28] have improved the notion of a noisy threshold by introducing the so called *functional threshold*, which accurately characterizes the convergence behavior of LDPC code ensembles under noisy finite-alphabet message passing decoding.

Although complex soft-decision iterative decoders build from reliable components typically outperform the low-complexity majority logic decoders, this is not necessarily true for faulty decoders. The complex Boolean functions are more sensitive to hardware unreliability, which may lead to pronounced vulnerability, as shown in a case of the noisy quantified min-sum decoder [27]. On the other hand, the probabilistic gradient decent bit flipping decoder, recently proposed by Al Rasheed *et al.* in [12], performs approximately the same on both perfect and faulty hardware. This resulted in an increased interest in hard-decision decoders.

One-step majority logic (O-S-MAJ) decoding introduced in the sixties by Rudolf [29] and Massey [30] is an important class of algorithms in the context of faulty decoding. In the O-S-MAJ decoder the decoding process is terminated after only one iteration, and the bit estimates are obtained by a majority vote on multiple parity check decisions. In contrast to iterative decoders, the bit error rate performance of these decoders can be evaluated analytically for finite-length codes as shown by Radhakrishnan *et al.* [31].

In all the above references a special type of so called *transient failures* is assumed. Transient failures manifest themselves at particular time instants but do not necessarily persist for later times. These failures have probabilistic behavior and we assume the knowledge of their statistics. The simplest such statistics is the von Neumann failure model [32], which assumes that each component of a (clocked) Boolean network fails at every clock cycle with some known probability. Additionally, the failures are not temporally nor spatially correlated. In other words, failures of a given component are independent of those in previous clock cycles and independent of failures of other components.

However, the von Neumann failure model is only a rough approximation of the physical processes leading to logic gate failures. The actual probability of failure of logic gates is highly dependent on a digital circuit manufacturing technology, and for high integration factors the failures are data-dependent and/or temporally correlated as it was shown by Zaynoun *et al.* in [33]. For example, timing errors are heavily dependent on data values processed by the gate in previous bit intervals and cannot be represented accurately by the von Neumann model.

The three main contributions of this chapter the following: (*i*) the memory architecture which uses O-S-MAJ decoders with faulty XOR gates for correcting failures in memory registers, (*ii*) a general approach to modeling of data-dependent gate faults based on Markov chains, and (*iii*) analysis of the effects of gate faults due to timing errors to memory reliability. Markov chain model captures the effects of data-dependent and correlated nature of gate failures. In this chapter we continue the analysis presented in Deliverable 4.1 where we derived a closed form expression of the residual bit error rate (BER) at the output of the one-step majority logic decoder for an ensemble of regular LDPC codes free of four-cycles. Here, we investigate influence of timing gate failures on the performance of finite geometry LDPC codes.

The rest of the chapter is organized as follows. In Section 4.2 the preliminaries on O-S-MAJ decoding are discussed. In Section 2.3 we give a description of our memory architecture and novel approach to gate failure modeling. Section 2.4 is dedicated to the theoretical analysis of the one-step majority logic decoder. The special case of the timing failure model, while the capability of the memory to store information over time is investigated in Section 2.5. The numerical results are presented in Section 2.6. Finally, some concluding remarks and future research directions are given in Section 4.5.

2.2 Preliminaries

2.2.1 Decoding of LDPC Codes

Let C be a (γ, ρ)-regular binary LDPC code of length n, with a parity check matrix $\mathbf{H}$ and code rate $r \geq 1 - \gamma/\rho$. The parity check matrix can be represented with a bipartite graph called the Tanner graph. Each column in the parity check matrix corresponds to a variable node and each row corresponds to a check node in the Tanner graph, and a variable node v and a check node c are adjacent if and only if $H_{c,v} = 1$. We denote a set of edges incident on a node x in a Tanner graph (x can be either variable or check node) as E_x. A vector $\mathbf{x} = (x_1, x_2, ..., x_n)$ is a codeword if and only if $\mathbf{H}\mathbf{x}^{\mathrm{T}} = \mathbf{0} \pmod 2$.

A codeword $\mathbf{x}$ is stored in a memory, and when read from the memory each bit x_v is flipped by probability α and observed as r_v. We refer to r_v as value of the variable node v. The number of flipped bits is called the Hamming distance between the stored codeword $\mathbf{x}$ and the read-back word $\mathbf{r}$, and is denoted as $d_H(\mathbf{x}, \mathbf{r})$.

The O-S-MAJ decoder is implemented as expressed in Algorithm 1.

Algorithm 1 O-S-MAJ Decoder:

```
Input: $\mathbf{r} = (r_1, r_2, \ldots, r_n)$
for $\forall v$ do
  $\hat{r}_v \leftarrow r_v$
  $\forall e \in E_v : m_e \leftarrow \bigoplus_{e' \in E_c \setminus \{e\}} r_{e'}$
  if $|\{e \in E_v : m_e = s\}| > \lfloor \gamma/2 \rfloor$ then
    $\hat{r}_v \leftarrow s$
  end if
end for
Output: $\hat{\mathbf{r}} = (\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_n)$
```

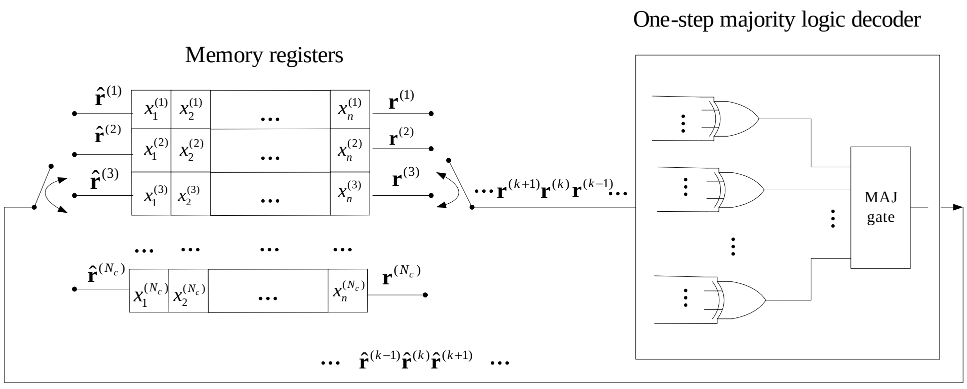

One-step majority logic decoder
Memory registers
$\hat{\mathbf{r}}^{(1)}$
$\hat{\mathbf{r}}^{(2)}$
$\hat{\mathbf{r}}^{(3)}$
$\hat{\mathbf{r}}^{(N_c)}$
$x_1^{(1)}$ $x_2^{(1)}$ $x_n^{(1)}$
$x_1^{(2)}$ $x_2^{(2)}$ $x_n^{(2)}$
$x_1^{(3)}$ $x_2^{(3)}$ $x_n^{(3)}$
$x_1^{(N_c)}$ $x_2^{(N_c)}$ $x_n^{(N_c)}$
$\mathbf{r}^{(1)}$
$\mathbf{r}^{(2)}$
$\mathbf{r}^{(3)}$
$\mathbf{r}^{(N_c)}$
$\dots\mathbf{r}^{(k+1)}\mathbf{r}^{(k)}\mathbf{r}^{(k-1)}\dots$
MAJ gate
$\dots\ \hat{\mathbf{r}}^{(k-1)}\hat{\mathbf{r}}^{(k)}\hat{\mathbf{r}}^{(k+1)}\ \dots$

Figure 2.1: The memory architecture scheme.

A check node makes an estimate of the variable node value based on the other variable nodes. The value of variable node itself is not used in its estimation. Majority of bit estimates determine the decoded bit value.

2.2.2 Transient Gate Failures

Let $f : \{0,1\}^m \to \{0,1\}$, $m > 1$, be an m-argument Boolean function. The relation between the input arguments $y_1, y_2, \dots y_m$ and the output z of a *perfect* gate realizing this function is $z = f(y_1, y_2, \dots, y_m)$. For a *faulty* gate, this input-output relation is $z = f(y_1, y_2, \dots, y_m) \oplus e$, where $\oplus$ is the Boolean XOR and the error $e \in \{0,1\}$ is a Bernoulli random variable. Denote by $\mathbf{y} = (y_1, y_2, \dots, y_m)$ the gate input vector, i.e. a vector of arguments. Denote by $\{\mathbf{y}^{(k)}\}_{k\geq 0}$ a time-sequence of input vectors, and by $\{e^{(k)}\}_{k\geq 0}$ the corresponding failure sequence. In the manuscript we will interchangeably use the terms “failure” and “error” meaning that the failures are “additive” errors. In a classical von Neumann transient failure model the error values $\{e^{(k)}\}_{k\geq 0}$ are independent of input sequence $\{\mathbf{y}^{(k)}\}_{k\geq 0}$.

2.3 System Model

2.3.1 The Memory Architecture

We analyze the O-S-MAJ decoding in the context of reliable information storage, as presented in Fig. 2.1. A collection of $N_c \gg 1$ codewords $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N_c)}$, denoted as $\{\mathbf{x}^{(k)}\}_{k\in[1,N_c]}$, is stored in an unreliable memory registers, which are periodically updated based on the error correction scheme. The memory unreliability is modeled by an n-dimensional binary random variable $\mathbf{E}$ defined over $\{0,1\}^n$ with independent entries E_j such that $\Pr\{E_j = 1\} = \alpha$, $1 \leq j \leq n$. The values read form the memory can be described by random variables $\mathbf{R}^{(k)} = \mathbf{E} \oplus \mathbf{x}^{(k)}$, where their particular realizations are denoted as $\mathbf{r}^{(k)}$, $1 \leq k \leq N_c$. The memory registers are updated by the round-robin scheduling principle, which means that k-th codeword registers $\mathbf{r}^{(k)}$ are updated in $(a-1)N_c + k$, $a \in \mathbb{N}$, memory update cycles. During an update cycle the sequence $\mathbf{r}^{(k)}$ is decoded by the faulty O-S-MAJ decoder and replaced by the newly estimated sequence $\hat{\mathbf{r}}^{(k)}$. The registers update mechanism is formally expressed in Algorithm 2.

The memory architectures that use error correcting schemes are characterized by two closely related parameters: *complexity* and *redundancy*. The memory *complexity* is defined as the total number of components within the memory required to store a single bit or perform 2-input Boolean function [3]. The *redundancy* of a memory architecture is the ratio of the complexity of the memory to the complexity of an irredundant memory built from perfectly reliable registers, which store the same number of information bits [3].

The O-S-MAJ decoder needs ρ $(\rho-1)$-input XOR gates at each check node, and γ-input majority logic (MAJ) gate at every variable node. It is known that the $(\rho-1)$-input XOR gate can be implemented as serial concatenation of $\rho-2$ 2-input XOR gates. As there are $n\gamma/\rho$ check nodes, the total number of 2-input XOR gates needed for the decoder implementation is equal to $n\gamma(\rho-2)$. We denote the complexity of the γ-input MAJ gate as D_γ.

Algorithm 2 The memory registers update in L cycles:

```
Input: ${\mathbf{r}^{(k)}}_{k\in[1,N_c]}$
$j \leftarrow 1$
while $j \leq L$ do
  $k \leftarrow \mod (j, N_C)$
  for $\forall v$ do
    $\hat{r}_v^{(k)} \leftarrow r_v^{(k)}$
    $\forall e \in E_v : m_e \leftarrow \bigoplus_{e'\in E_c\setminus\{e\}} r_{e'}^{(k)}$
    if $|\{e \in E_v : m_e = s\}| > \lfloor \gamma/2 \rfloor$ then
      $\hat{r}_v^{(k)} \leftarrow s$
    end if
  end for
  $j \leftarrow j + 1$
end while
Output: ${\hat{\mathbf{r}}^{(k)}}_{k\in[1,N_c]}$
```

The memory build from reliable registers does not use error correcting scheme and its complexity is equal to nrN_c, where r represents code rate of a LDPC code used in a memory with redundance. Then, the memory redundancy can be expressed as follows

$$R = n(N_c + D_\gamma + \gamma(\rho-2))/(rnN_c) \leq (N_c + D_\gamma + \gamma(\rho-2))/((1-\gamma/\rho)N_c). \tag{2.1}$$

The memory redundancy is independent of n and remains bounded for infinite code lengths.

The final bit decision in a O-S-MAJ decoder is made on the basis of majority of the bit estimates, resulting in the probability of error of an estimated bit greater than or equal to the probability of failure of the MAJ gate. Since the error probability of the MAJ gate lower bounds the BER performance, MAJ gates must be made highly reliable. Otherwise, the probability of error is determined by this final gate, not the error control scheme. Thus, it is reasonable to make an assumption that MAJ gates are perfect and that only XOR gates are faulty. Reliable MAJ gates can be realized, for example, by using larger transistors. It should be noted that TK memory scheme also requires perfect MAJ gate implementation for the final step of extracting user information [3].

We are interested in finding the residual error rate after the decoding scheme. For that purpose we may assume that the sequence of codewords $\{\mathbf{x}^{(k)}\}_{k\geq 0}$ is transmitted through the communication Binary Symmetric Channel (BSC) with crossover probability α and then successively decoded by a single O-S-MAJ decoder built from perfect MAJ gates and faulty XOR gates. Before proceeding to this derivation we discuss a realistic model of XOR gate failures.

2.3.2 Data-Dependent Error Model

There are two types of hardware failures considered in this chapter: memory register failures and XOR logic gate failures. The memory registers failures can be modeled as spatially independent uniformly distributed random variables with the same probability of failure for each of the memory register [34]. On the other hand, the gate failures are dependent on gate input patterns and can not be represented in the same manner as memory failures [33].

In order to capture more accurately the data and time dependence of the failures we use Markov chains. Namely, we assume that $e^{(k)}$, the error at time k, is affected by the current and $M-1$ prior consecutive gate input vectors, i.e., its probability depends on the input vector sequence in the time interval $[k-(M-1),k]$, denoted as $\{\mathbf{y}^{(j)}\}_{j\in[k-(M-1),k]}$, where M is a positive integer. Denote this probability by $\Pr\{e|\mathbf{s}^{(k)}\}$, where the *gate state* $\mathbf{s}^{(k)}$ at time k is defined as $\mathbf{s}^{(k)}=\{\mathbf{y}^{(j)}\}_{j\in[k-(M-1),k]}$. As previously stated, in our O-S-MAJ decoder only XOR gates are unreliable. The number of states grows exponentially with M and ρ, i.e., for an $(\rho-1)$-input XOR gate used in our decoder there are $2^{M(\rho-1)}$ states.

The inputs of a (perfect) MAJ gate are the outputs of γ XOR gates in the neighboring check nodes. Thus, at time k these gates can be associated with a *state array* $\sigma^{(k)}=(\mathbf{s}_1^{(k)},\mathbf{s}_2^{(k)},\dots,\mathbf{s}_\gamma^{(k)})$, whose elements represent states of particular XOR gates. Based on $\sigma^{(k)}$, an *error probability vector* can be formed as $\epsilon^{(k)}=(\epsilon_1^{(k)},\epsilon_2^{(k)},\dots,\epsilon_\gamma^{(k)})$, $\epsilon_m^{(k)}=\Pr\{e|\mathbf{s}_m^{(k)}\}$, $1\le m\le\gamma$. The values of the error probability vector can be obtained by measurements or by simulation of the selected semiconductor technology. Thus, in our analysis we assume that these values are known.

2.4 Decoder Analysis under Timing Errors

Timing errors in logic gates are caused by so-called timing violations. The gate timing violation happens due to the supply voltage reduction, sampling clock fluctuations or signal propagation delays, when the output signal of a gate is sampled or used in the next stage before it reaches a steady value, leading to an incorrect output. Treating these errors represents a major challenge in new energy-efficient CMOS technologies [33]. Timing errors depend on a gate history, i.e. data values processed by the gate in previous bit intervals. As an erroneous output of a logic gate occurs only if the gate output changes its values, it is usually sufficient to consider the failure dependence on data in current and only one previous bit interval, i.e. $M=2$. Thus, two disjoint subsets of faulty XOR gate states, denoted as $\mathcal{S}_1$ and $\mathcal{S}_2$, can be identified. The first subset $\mathcal{S}_1$ is constituted by states in which gate output remains unchanged and for which the failure is impossible, i.e. $\Pr\{e|\mathbf{s}^{(k)}\}=0,\forall\mathbf{s}^{(k)}\in\mathcal{S}_1$. If a gate output changes its value, the failure is possible and the states from the subset $\mathcal{S}_2$ have non-zero error probabilities. For simplicity, we assume that these error probabilities are the same, i.e. $\Pr\{e|\mathbf{s}^{(k)}\}=\varepsilon,\forall\mathbf{s}^{(k)}\in\mathcal{S}_2$.

Let $\{\mathbf{x}^{(k)}\}_{k\ge0}$ be a codeword sequence stored in the memory registers. Clearly, decoding error of $\mathbf{x}^{(k)}$ depends on $M-1$ codewords previously read from the memory. Let $\mathbf{y}_{m,v}=\{\mathbf{y}_{m,v}^{(j)}\}_{j\in[k-(M-1),k]}$, $1\le m\le\gamma$, $1\le v\le n$, be sequence of code bits that, if stored with no errors, will appear at inputs of m-th XOR gate connected to node v, in the time interval $[k-(M-1),k]$.

The XOR gate output will remain unchanged if gate input vectors from two consecutive time points $k-1$ and k, $k>1$, are the same or differ in an even number of positions. Thus, for example, the m-th XOR used for decoding the bit x_v will produce correct output, at time k, if the vectors written in the memory $\mathbf{y}_{m,v}^{(k-1)}$ and $\mathbf{y}_{m,v}^{(k)}$ satisfy the relation $\bmod(d_H(\mathbf{y}_{m,v}^{(k-1)},\mathbf{y}_{m,v}^{(k)}),2)=0$ and no memory errors occur. Similarly, the gate output will be erroneous with the probability ε if all bits are stored without errors and $\bmod(d_H(\mathbf{y}_{m,v}^{(k-1)},\mathbf{y}_{m,v}^{(k)}),2)=1$. However, the gate input vectors parity can be changed due to memory failures, when an odd number of gate inputs from two consecutive time points are flipped. The probability of the collection of all such events is equal to

$$A=\sum_{j=0}^{\rho-2}\binom{2(\rho-1)}{2j+1}\alpha^{2j+1}(1-\alpha)^{2\rho-2j-3}. \tag{2.2}$$

Therefore, we can conclude that the gate output will be erroneous with the probability εA when the relation $\mod (d_H(\mathbf{y}_{m,v}^{(k-1)}, \mathbf{y}_{m,v}^{(k)}), 2) = 0$ is satisfied. Let all XOR gates with this property, used for decoding the bit x_v, form a set $\mathcal{G}_v$. Similarly, $\mathcal{H}_v$ is composed of all the gates for which $\mod (d_H(\mathbf{y}_{m,v}^{(k-1)}, \mathbf{y}_{m,v}^{(k)}), 2) = 1$.

Let $P_v(\alpha, \epsilon^{(t)})$ be the probability of miscorrection of code bit v under error probability vector $\epsilon^{(t)}$. The derivation of $P_v(\alpha, \epsilon^{(t)})$ was explained in Deliverable 4.1 in details. We now extend the previous discussion on faulty XOR gates and formulate the lemma that describes the data-dependence of O-S-MAJ decoding.

Lemma 1. *Let $\mathbf{x}^{(k-1)}$ and $\mathbf{x}^{(k)}$ be the codewords decoded in two consecutive bit intervals. The faulty O-S-MAJ decoder will operate worst if the cardinality of the set $\mathcal{G}_v$, $|\mathcal{G}_v| = 0$, $1 \leq v \leq n$, while the best performance corresponds to decoding of the consecutive codewords for which $|\mathcal{G}_v| = \gamma$, $1 \leq v \leq n$.*

Proof: The failures of XOR gates from the set $\mathcal{G}_v$ happen with probability $A\varepsilon$, while the failure rate under condition that a gate is an element of the $\mathcal{H}_v$ is equal to $(1-A)\varepsilon$. Since $A < 0.5$ a gate from $\mathcal{H}_v$ will be erroneous more often. The proof of lemma follows from the fact that the probability $P_v(\alpha, \epsilon)$ monotonically increases with increase of hardware unreliability, i.e. for every $\epsilon^{(t_1)}$ and $\epsilon^{(t_2)}$ with property $\epsilon_m^{(t_1)} \leq \epsilon_m^{(t_2)}$, $1 \leq m \leq \gamma$ holds $P_v(\alpha, \epsilon^{(t_1)}) \leq P_v(\alpha, \epsilon^{(t_2)})$. ■

The previous lemma reveals a fundamental property of the O-S-MAJ decoding performance under data dependent hardware failures: *dependence on codewords decoding order*. It can be seen that, for example, decoding of two the same codewords will result in the lowest error rate, while if two complementary codewords are consecutively decoded the decoder will operate worst.

The O-S-MAJ decoder built entirely from reliable components satisfy the symmetry theorem, which states that performance of the decoder are independent of codewords being decoded. We see that the symmetry condition does not hold for the O-S-MAJ decoding in the presence of timing errors. The negative effect of data-dependence can be reduced by optimizing the encoding process. However, this is out of scope of this paper and left for future research.

Let the cardinality of the set $\mathcal{G}_v$, be equal to $|\mathcal{G}_v| = t_v$. The bit miscorrection probability depends only on the number of non-zero elements of ϵ, but not on its order. Thus, we can simplify the notation by introducing $\varepsilon^{(t)} = (\varepsilon_1^{(t)}, \varepsilon_2^{(t)}, \ldots, \varepsilon_\gamma^{(t)})$: an error probability vector with t non-zero elements. This allows as to formulate the following corollary that gives the bit miscorrection probability under timing errors.

Corollary 1. *The probability that a code bit x_v of a (γ,ρ)-regular LDPC code is incorrectly decoded by the faulty O-S-MAJ decoder under timing errors is given by*

$$\bar{P}_v(t_v) = \sum_{t=0}^{\gamma} P_v\left(\alpha, \varepsilon^{(t)}\right) \sum_{j=t_{min}}^{t_{max}} \binom{t_v}{j} \binom{\gamma - t_v}{t-j} A^{\gamma+2j-t_v-t}(1-A)^{t_v+t-2j}, \tag{2.3}$$

where, $t_{min} = \max(t + t_v - \gamma, 0)$ and $t_{max} = \min(t_v, t)$.

Proof: The probability that j non-zero failure rates in $\varepsilon^{(t)}$ originated from the set $\mathcal{G}_v$ and $t-j$ from the set $\mathcal{H}_v$ is equal to $\binom{t_v}{j}\binom{\gamma-t_v}{t-j}A^{\gamma+2j-t_v-t}(1-A)^{t_v+t-2j}$. The sum of all possible ways that t non-zero failure rates can appear represents the contribution of $P_v\left(\alpha, \varepsilon^{(t)}\right)$ in the overall miscorrection probability value. The final summation for all $\gamma+1$ values of t gives the bit miscorrection probability. ■

Based on Lemma 1 and Corollary 1 we can measure the effect of data-dependence by bounding BER values, as described in the following lemma.

Lemma 2. *The conditional average bit error rate of a (γ, ρ)-regular LDPC code decoded by the faulty O-S-MAJ decoder in the presence of timing errors is bounded by*

$$\sum_{t=0}^{\gamma} \binom{\gamma}{t} A^t (1-A)^{\gamma-t} P_v\left(\alpha, \varepsilon^{(t)}\right) \leq \bar{P}_e(\mathbf{x}^{(k)}|\mathbf{x}^{(k-1)}) \leq \sum_{t=0}^{\gamma} \binom{\gamma}{t} A^{\gamma-t}(1-A)^t P_v\left(\alpha, \varepsilon^{(t)}\right). \tag{2.4}$$

Proof: According to Lemma 1 the lower bound is obtained by setting $t_v = \gamma$ in Eq. (2.3). Similarly, the upper bound can be calculated by setting $t_v = 0$. ■

The bounds presented in Eq. (2.4) are obtained under conditions described in Lemma 1, i.e. they represent the lowest and the highest possible BER values. These bounding values depend on all parameters γ, ρ, ε and α and can differ by orders of magnitude. The analysis of decoder performance under timing errors for the several classes of LDPC codes is presented in Section 2.6.

2.5 Analysis of TK-based Memories

In the previous section we have analyzed the memory system in which multiple codewords represent memory content, meaning that different codewords are decoded in the consecutive bit intervals. We found the residual error rate after one decoding cycle. Here, we investigate ability of the memory based on the O-S-MAJ to preserve information over time.

We assume that only one codeword is stored in our memory presented in Fig. 2.1 ($N_c = 1$). The bit values stored in the registers are updated at regular time instants ($\tau, 2\tau, \ldots, L\tau, L \in \mathbb{N}$), by the O-S-MAJ decoder presented in Sections 4.2 and 2.3. It is also assumed that the time for update is smaller compared to τ and that the memory content do not change while the update is in progress.

We investigate memory architecture under the adversarial failure model. In the adversarial model the number of components failures at any given time is upper bounded, i.e. only a fraction of components can be faulty [21]. Let α_m be the fraction of errors that can affect the memory registers between memory correcting cycles and let $\alpha_\oplus$ be the fraction of erroneous XOR gates used for decoding a single bit during the every correcting cycle. The majority logic gates are considered to be reliable as in the previous sections. We are interested in finding the fraction of errors that can be tolerated by our memory architecture. It is given by the following theorem.

Theorem 1. *The proposed memory architecture build from LDPC codes free of four cycles can preserve all stored information bits for an arbitrary long time period if*

$$n\alpha_m + \gamma\alpha_\oplus \leq \lfloor \gamma/2 \rfloor. \tag{2.5}$$

Proof: Since for four-cycle free LDPC codes the number of orthogonal parity check sums for each bit is equal to column weight of the code parity check matrix, the O-S-MAJ decoder can correct up to $\lfloor \gamma/2 \rfloor$ codeword errors. In other words, each bit will be decoded correctly if the number of incorrect estimates at inputs of a MAJ gate is no greater then $\lfloor \gamma/2 \rfloor$. In a noisy decoder incorrect estimate can also appear due to an unreliability of a XOR gate. In the worst case t_m memory errors can produce t_m incorrect estimates and, similarly, $t_\oplus$ XOR failures can lead to $t_\oplus$ incorrect estimates. Thus, each bit will be correctly decoded if $t_m + t_\oplus \leq \lfloor \gamma/2 \rfloor$. If total number of errors in the time interval τ is bounded by the previous condition, after every correcting cycle in the memory registers only correct values of bits are written. Therefore, Eq. (2.5) is obtained by noting that $t_m = n\alpha$ and $t_\oplus = \alpha_\oplus\gamma$. ■

If all the XOR gates were perfect, the fraction of memory failures that can be tolerated would correspond to a correcting capability of a code. The presence of gate failures reduces the correcting capability of a code by the factor $\gamma\alpha_\oplus$. It can be seen that, under the condition given by Theorem 1, the O-S-MAJ decoder can correct all memory errors that appeared between two update cycles. In such a way the number of erroneous registers is never larger then $n\alpha_m$. In the next section, we illustrate Eq. (2.5) with examples of memories based on projective geometry codes.

Additionally, under the data-dependent error model, gate failures are dependent on failures of the memory registers. This is especially pronounced in the case of timing errors where the fixed fraction of memory failures cause the fixed fraction of gate failures. Then, Eq. (2.5) can be simplified as in the following corollary.

Corollary 2. *The proposed memory architecture under timing errors can tolerate the fixed fraction of memory failures α_m if*

$$\alpha_m \leq \frac{1}{n}\left\lfloor \frac{\gamma}{6} \right\rfloor. \tag{2.6}$$

Proof: Consider the memory registers connected to a particular XOR gate. If in the previous correcting cycle all memory errors were corrected the output of a XOR gate during the current cycle can be erroneous only if the total number of memory failures in both current and previous decoding cycles is odd.

Let $\mathcal{F}_L$ be a set of all faulty memory registers between the $(L-1)$-th and L-th update cycle. From the initial condition follows $|\mathcal{F}_L| \leq \alpha_m n$, for any $L > 0$. It is clear that the maximal number of faulty XOR gates correspond to the case when $\mathcal{F}_{L-1} \cap \mathcal{F}_L = \emptyset$. Then, the total number of faulty XOR gates used for decoding a particular bit is bounded by $2\alpha_m n$. The Eq. (2.6) can be obtained by incorporating the previous observation into expression (2.5), from Theorem 1. ■

The previous corollary is important because it defines the condition under which memory failures can be corrected during the memory update cycle regardless of the XOR gate reliability. In other words, if the number of faulty memory registers is less then $\lfloor \gamma/6 \rfloor$, the memory content can be stored correctly under timing errors with arbitrary high XOR failure rates.

Note that Eq. (2.6) does not guarantee positive storage capacity since the fraction of memory failures reduces with code length increase. In order to prove that memory based on O-S-MAJ decoder has positive storage capacity under timing error model we use expander graph arguments. This is presented in the next chapter of the deliverable.

2.6 Numerical Results

2.6.1 Decoder Performance under Timing Errors

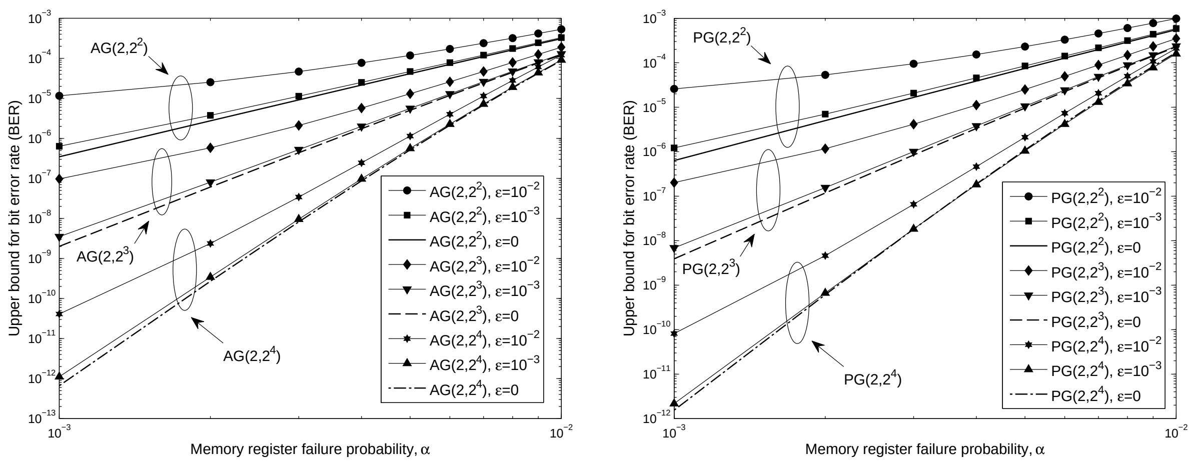
The codes designed from finite geometries are considered to be important one-step majority logic decodable codes [35–37]. It was proved that for a LDPC code derived from finite geometries, with column weight γ, the O-S-MAJ decoder can correct up to $\lfloor \gamma/2 \rfloor$ errors [31]. In this section we investigate 2-dimensional affine and projective geometry LDPC codes form over the Galois field GF(2^s), denoted as AG$(2, 2^s)$ and PG$(2, 2^s)$ codes, $s > 0$, respectively. The affine geometry codes, AG$(2, 2^s)$, have the parity check matrix with row weight $\rho = 2^s + 1$ and column weight $\gamma = 2^s$ and minimum distance $d_{min} = 2^s + 1$. The PG$(2, 2^s)$ code is characterized by column and row weights equal to $\rho = \gamma = 2^s + 1$ and minimum distance $d_{min} = 2^s + 2$.

The average bit error probabilities for several PG and AG codes under timing errors are presented in Fig. 2.2. The performance upper bounds are calculated using Eq. (2.4) for the case of two gate error rates $\varepsilon = 10^{-3}, 10^{-2}$ and compared to the case of $\varepsilon = 0$, i.e. with the perfect decoder. It should be noted that lower bounds values correspond to rare hardware failures and can be well estimated using

$$\sum_{t=0}^{\gamma} \binom{\gamma}{t} A^t (1-A)^{\gamma-t} P_v\left(\alpha, \varepsilon^{(t)}\right) \approx P_v\left(\alpha, (0, \ldots, 0)\right), \tag{2.7}$$

and for that reason they are omitted from Fig. 2.2.

It can be seen that frequent hardware failures can lead to significant performance degradation. This degradation is especially pronounced in the region with low memory failure rates. For example if $\alpha = 10^{-3}$, extremely unreliable XOR gates (with $\varepsilon = 10^{-2}$) can reduce the bit error



(a) AG codes (b) PG codes

Figure 2.2: Analytically calculated BER bounds for different LDPC codes constructed from finite geometries.

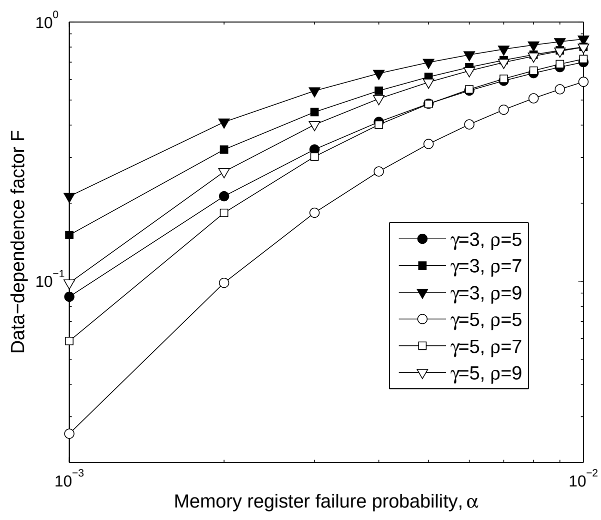


Figure 2.3: The data-dependence factor values for different (γ, ρ) classes of LDPC codes ($\varepsilon = 10^{-2}$).

rate for an order of magnitude for all the presented codes. On the other hand, hardware failures corresponding to $\varepsilon = 10^{-3}$, cause notably smaller performance loss. Performance loss is lower for higher s, which results in negligible BER degradation for codes with $s = 4$, i.e. AG$(2, 2^4)$ and PG$(2, 2^4)$. Since $\varepsilon = 10^{-3}$ is considered to be a large value of the gate failure probability, the O-S-MAJ is in general proved to be resistant to hardware unreliability. For smaller values of ε ($\varepsilon < 10^{-3}$), the BER degradation is negligible for all the analyzed codes.

As a convenient measure of performance variation caused by timing errors we define the *data-dependence factor*, F, as ratio of two border BER values, given by Lema 2, as follows

$$F = \frac{\sum_{t=0}^{\gamma} \binom{\gamma}{t} A^t (1-A)^{\gamma-t} P_v\left(\alpha, \varepsilon^{(t)}\right)}{\sum_{t=0}^{\gamma} \binom{\gamma}{t} A^{\gamma-t} (1-A)^{t} P_v\left(\alpha, \varepsilon^{(t)}\right)}. \tag{2.8}$$

The values of F for different (γ, ρ) classes of LDPC codes are presented in Fig. 2.3. It can be seen that the degradation is higher in codes with larger column weights. For example, when $\alpha = 10^{-3}$, for codes with $\gamma = \rho = 5$, the BER upper bound is two order of magnitude higher

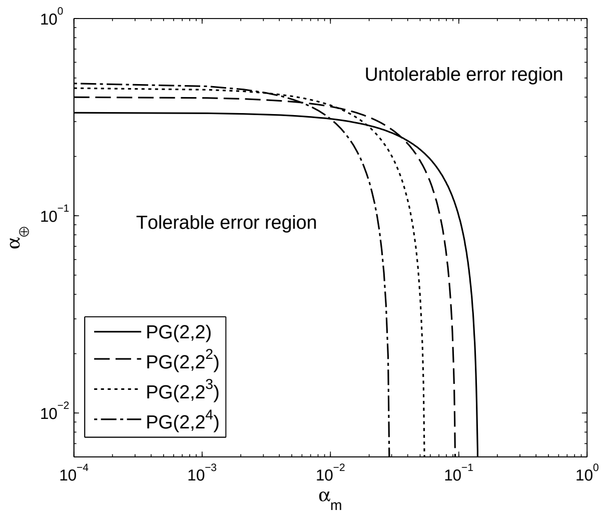


Figure 2.4: The graphical interpretations of regions in which successful information storage is possible.

then the corresponding lower bound. As correcting capability of the code increases with γ, it is interesting to notice that the better codes are more susceptible to negative effects of hardware failures, for the same ρ value. Additionally, it can be shown that the performance loss can be reduced by increasing row weight of $\mathbf{H}$.

2.6.2 Tolerable Error Region

The condition given in Theorem 1 is graphically interpreted in Fig. 2.4, for several PG codes. The region below the curves describes all fractions of errors that can be tolerated. It can be seen, for example, that memory architecture based on $\mathrm{PG}(2, 4)$ code can provide successful information storage even if fractions close to 10% of all memory registers and XOR gates are erroneous during the time interval τ. Although the codes with higher column weights can correct more errors in absolute scale, the number of memory components also increases resulting narrower tolerable region for these codes.

2.7 Conclusion

Although the von Neumann error model is suitable for theoretical evaluation of fault-tolerant systems, use of the results obtained under this error model is limited. In practice, unreliability of logic gates is usually data-dependent and correlated in time. Hence, in order to describe hardware unreliability phenomenon more accurately, the change of modeling paradigm is required. We advocate use of Markov chains, which provide more general modeling approach.

We proposed a low complexity memory architecture, build from simple XOR and MAJ logic gates, that can operate well under unreliable gate computation. Then, based on data-dependent gate failure model, we developed an analytical method for performance evaluation of the proposed memory architecture. Our method enables calculating the BER of any regular LDPC code of girth at least six. These BER values are highly dependent on the codewords stored in the memory and we have succeed to bound them for the case of timing errors. We have also studied the ability of the proposed memory to preserve information over time.

The future research includes the investigating fault-tolerant schemes which use other types of LDPC decoders, under data-dependent hardware failures. We are working on generalization of tolerable error region results to more complex memory architectures, such as, for example, finite-alphabet LDPC decoders. Based on the structural property of Tanner graphs of LDPC codes we are also investigating possibility of designing novel decoders that can work well under data-dependent hardware failures.

Chapter 3

Expander Graph Arguments for Bit-Flipping Decoders Made of Unreliable Gates

Abstract: *In this chapter we investigate the fault-tolerant decoders based on the bit flipping algorithm for low-density parity-check (LDPC) codes, in the presence of data-dependent logic gate failures. Based on the expander property of the Tanner graph of the LDPC code, we prove that the number of worst case errors that can be corrected by the bit flipping algorithm, built in part from unreliable logic gates, increases linearly with the code length. In addition, we give conditions under which a memory architecture based on the bit flipping algorithm can preserve the stored information for an arbitrary long time period.*

3.1 Introduction

Due to huge density integration increase, 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 reduction of energy consumption by several orders of magnitude, which can be done only by aggressive supply voltage scaling. Consequently, the signal levels are much lower and closer to the noise level, which 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 unreliable components [19]. Recently, there has been a surge in interest in error control schemes that can ensure fault-tolerance in unreliable hardware. The only known class of codes resilient to logic gate faults are low-density parity-check (LDPC) codes [4,20,38]. Their attractiveness 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 [3]. Such fault tolerant decoders are based on message-passing and bit-flipping decoding algorithms, which unlike more complex algorithms, limit the error propagation in a decoder caused by faulty logic gates.

This research is mostly motivated by the studies of fault-tolerant memories based on the LDPC decoders presented in the late sixties and early seventies by Taylor [3] and Kuznetsov [4]. In their pioneering works they proposed a memory architecture build entirely from unreliable components, capable of preserving stored information over arbitrary long time. Their memory is composed of an unreliable memory registers which are periodically updated using a faulty correcting circuit. It was later observed by Vasić *et al.* in [6,7,20] that an update cycle corresponds to one iteration of a Gallager-B decoder, built from unreliable logic gates.

Both Taylor and Kuznetsov as well as most of the related work [1–4, 6, 7, 12, 14, 15, 20–28] modeled logic gate unreliability as transient independent failures, originally introduced by von Neumann [32]. In the von Neumann failure model each component of a (clocked) Boolean network fails at every clock cycle with some known probability. Additionally, failures of a given component are independent of those in previous clock cycles and independent of failures of other components. Although the simplicity of this model makes it attractive for theoretical analysis, it is unrealistic. In practice, the unreliability of logic gates is strongly data-dependent and correlated in time. The most dominant effect in new energy-efficient CMOS technologies a logic gate unreliability comes from the so-called timing violations, which happen when a gate output changes its value [33]. The logic gate failures resulting from timing violations are called *timing failures*. Their statistics and effects to gate reliability has been recently studied by [5].

In this chapter, we consider timing failures. More specifically, we investigate the error correction capabilities of the bit flipping decoders, and show that expander graph arguments can be used to establish lower bounds on guaranteed error correction capability. Additionally, we extend results presented in [39] and prove that the error correction of a noisy bit flipping decoder can also be guaranteed when Tanner graph of a code has sufficient girth.

Furthermore, following the work presented in [21], we investigate the conditions under which the memory architecture based on the bit flipping decoder, as a correction circuit, can preserve information over time, and provide upper bounds on fraction of component failures that can be tolerated. Then, based on the Chernoff bounds, we prove that in the asymptotic code length our memory architecture achieves arbitrary small probability of failure under data-dependent failure model.

Guaranteed error correction of LDPC codes and iterative decoders have been studied in numerous papers, but with rather limited results. Although Gallager [40] proved the existence of $(\gamma \geq 3, \rho > \gamma)$-regular LDPC code, for which, under certain conditions, error rate approaches zero asymptotically, he failed to show that iterative decoders can correct fixed fraction of errors if there exist some α, $0 < \alpha < 1$ for which the decoder can correct αn worst case errors, where n is the code length.

Zyablov and Pinsker [41] analyzed parallel bit flipping algorithm and proved that asymptotically, almost all codes in the regular code ensemble with left degree $\gamma \geq 5$, can correct fixed fraction of error. Sipser and Spielman [42] showed that expander LDPC codes can be conveniently used for guaranteed error correction analysis. They proved that both serial and parallel bit flipping algorithms can correct fixed fraction of error if the underlying Tanner graph is a good expander. In the later work Burshtein [43] generalized results presented in [41] and [42] and proved that a linear number of errors can be corrected by parallel bit flipping algorithm with almost all codes in $(\gamma \geq 4, \rho > \gamma)$-regular ensemble. The expander graph arguments can be also used to provide guarantees of the message passing algorithms, at is was shown by Burshtein and Miller in [44]. Feldman *et al.* in [45] investigated linear programming (LP) and derived theoretic bounds for LP decoding of expander LDPC codes. Recently, Chilappagari *et al.* [39] provided another look on the guaranteed error correction of the bit flipping algorithms. They, found the relation between the girth of the Tanner graph and the guaranteed error correction capability of an LDPC code.

Recent research on noisy LDPC decoders has been aimed to two main directions: (*i*) achieving reliable storage (see [1, 6, 7, 20, 21]) and (*ii*) reliable transmission of information (see [2, 12, 14, 15, 22–28]).

The modified Taylor-Kuznetsov scheme was introduced by Ivkovic *et al.* in [1], where it was shown that by adding a simple syndrome checker, build from reliable hardware, the performance of the memory scheme can be significantly improved. The existence of a reliable memory based on a modified bit-flipping algorithm was shown by Chilappagari and Vasić in [21] and [20], who proposed a memory architecture that can tolerate a fixed fraction of error in registers and logic gates based on expander LDPC codes. Varshney in [14] used the density evolution tool to investigate the memory architecture based on Gallage-A decoder and obtained a tight bound on the memory storage capacity, under von Neumann gate failure model. His approach is limited to the data-independent gate failures, and it is not applicable for the case of timing failures. In this chapter we overcome this problem by proving that the number of component failures that can be tolerated increases with the code length, which for infinite code length leads to perfectly reliable memory. This represents the first prove of the reliable memories existence under data-dependent gate failures.

The rest of the chapter is organized as follows. In Section 3.2 the system model is presented, where preliminaries on LDPC codes, decoding algorithm and failure models are discussed. Section 3.3 is dedicated to the theoretical analysis of the noisy bit flipping decoder, while the capability of the memory to store information over time is investigated in Section 3.4. Finally, some concluding remarks and future research directions are given in Section 3.5.

3.2 System Model

3.2.1 LDPC Expander Codes and Decoding Algorithm

Let $G = (U, E)$ be a graph with set of nodes U and set of edges E. An edge e is an unordered pair (v, c) which connects two *neighborly* nodes v and c. The cardinality of the set U, denoted as $|U|$ represents the order of the graph and, while $|E|$ defines the size of the graph. The set of neighbors of a particular node u is denoted as $\mathcal{N}(u)$. The number of neighbors of a node u, denoted as $d(u)$ is called the degree of u. The average degree of a graph G is $\bar{d} = 2|E|/|U|$.

The girth g of a graph G, is the length of smallest cycle in G. A bipartite graph $G = (V \cup C, E)$ is a graph constructed from two disjoint sets of nodes V and C, such that all neighbors of nodes in V belong to set C, and vice versa. The nodes in V are called variable nodes and nodes from V are check nodes. A bipartite graph is said to be γ-left-regular if all variable nodes have degree γ, and similarly, a graph is ρ-right-regular if all check nodes have degree ρ.

Consider a (γ, ρ)-regular binary LDPC code of length n and its graphical representation given by γ-variable node-regular and ρ-check node-regular Tanner bipartite graph G, with $n\gamma/\rho$ check nodes and n variable nodes. Let E_v (E_c) be the set of neighbors of variable v (check c). In this paper we only consider expander codes, i.e. LDPC codes whose Tanner graphs satisfy expansion property defined as follows.

Definition 3. *[42] 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 an αn variable nodes, at least $\delta|S|$ check nodes are incident to S.*

The codeword of the expander code is transmitted through communication channel and decoded by the parallel bit-flipping decoder, which can be summarized as follows [42]:

- In parallel, flip each variable that is in more unsatisfied than satisfied parity checks.
- Repeat until no such variable remains.

In order to characterize the decoder build from unreliable components it is not sufficient to give only its functional description. The decoder performance are highly dependent on the way how the decoder is implemented at the circuit level. In this paper the hardware unreliability is modeled at logic gate level, which means that the correction capability of the decoder depends on the type of the gates used in the decoder and their mutual connections. Our decoder is divided into *processing units* that correspond to nodes in Tanner graph representation of the decoder. Let $\overrightarrow{m}_i(e)$ ($\overleftarrow{m}_i(e)$) be the messages passed on an edge e from (to) variable node to/from check node during i-th decoding iteration, respectively. Similarly $\overrightarrow{m}_i(F)$ and $\overleftarrow{m}_i(F)$ denote the set of all messages from/to a variable node over the set of edges $F \subseteq E$. Each variable node processing unit performs the majority voting on the binary messages received from its neighboring check nodes as follows

$$\Phi(\overleftarrow{m}_{i-1}(E_v)) = \begin{cases} 1 & \text{if } \sum\limits_{e \in E_v} \overleftarrow{m}_{i-1}(e) \geq \lceil \gamma/2 \rceil, \\ 0 & \text{if } \sum\limits_{e \in E_v} \overleftarrow{m}_{i-1}(e) < \lceil \gamma/2 \rceil, \end{cases} \tag{3.1}$$

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. The output of the majority logic (MAJ) gate, described by the function $\Psi(\cdot)$ is then passed to all neighboring check nodes, i.e $\overrightarrow{m}_i(e) = \Phi(\overleftarrow{m}_{i-1}(E_v))$, $\forall e \in E_v$.

During each iteration, the check node processing unit c performs ρ Exclusive OR (XOR) operations defined as follows

$$\Psi(\overrightarrow{m}_i(E_c \setminus \{e\})) = \bigoplus_{e' \in E(c) \setminus \{e'\}} \overrightarrow{m}_i(e'), \quad \forall e \in E_c. \tag{3.2}$$

The results of the XOR operations are passed to neighboring variable nodes by mapping $\overleftarrow{m}_i(e) = \Psi(\overrightarrow{m}_i(E_c \setminus \{e\}))$, $\forall e \in E_c$.

Note that in our decoder the check node calculates *estimates* of the neighboring variable nodes, rather then the parity check equation, since the value of the variable being estimated is not used in the calculation. However, it is functionally equivalent to the bit-flipping decoder.

It is clear that after each decoding iteration, a variable will be corrupt if receives more then $\gamma/2$ incorrect estimates from its neighbors. It should be emphasized that for γ-even, it is also possible that $\gamma/2$ incorrect estimates corrupt a variable. Our decoder implementations requires ρ $(\rho - 1)$-input XOR gates at each check node, and γ-input majority logic gate at every variable node.

Hardware unreliability in the decoder comes from unreliable computation of the operations $\Phi(\cdot)$ and $\Psi(\cdot)$ as logic gates performing these functions are prone to timing failures, which are described in the following subsection.

3.2.2 Failure Models

In a faulty bit-flipping decoder, the decoding operations performed by logic gates may result in a *perturbation* of the gate output bit with respect to its correct value. We use the term "perturbation" do denote an unintended bit flip due to faulty-computation/storage, while the term "flip" is used for intended operations performed by the decoding algorithm. We say that a bit or a variable is *corrupt* if its value is different from the value it would have during decoding a codeword when there were no errors/failures of any kind.

We consider two failure models, probabilistic and adversarial. We introduce the probabilistic model first.

Let $f : \{0,1\}^m \rightarrow \{0,1\}$, $m > 1$, be an m-argument Boolean function, which at time instant k produces the result $z^{(k)} = f(y_1^{(k)}, y_2^{(k)}, \ldots, y_m^{(k)})$, where $y_1^{(k)}, y_2^{(k)}, \ldots, y_m^{(k)}$ are input arguments at time k. Due to unreliability of the logic gate, the result of the function computation is $z^{(k)} \oplus \chi^{(k)}$, where $\chi^{(k)}$ is the error at time k. In the timing failure model $\chi^{(k)}$ is data-dependent and $\Pr\{\chi^{(k)} = 1 | z^{(k)} = z^{(k-1)}\} = 0$, i.e. when the gate output is unchanged during two consecutive time instants the function f is always correctly computed [5]. On the other hand, when $z^{(k)} \neq z^{(k-1)}$, $\Pr\{\chi^{(k)} = 1 | z^{(k)} \neq z^{(k-1)}\} = \varepsilon_g$, $\varepsilon_g > 0$. Furthermore, XOR gates fail independently, i.e., for any $e \neq f$, the perturbations in $\overleftarrow{m}_i(e)$ and $\overleftarrow{m}_i(f)$ due to failures of the corresponding XOR gates are assumed to be independent. The memory elements fail also independently with probability ε_m [34], and MAJ gates and the gates used to verify that all checks are satisfied are assumed to be perfect.

Note that although it is known that different input patterns cause gate failures with different probabilities [33], we simplify the analysis by using only the highest possible failure rate denoted by ε_g. The value ε_g can be obtained experimentally or by circuit-level simulation of gates implemented in a given semiconductor technology.

In order to prove that the memory architecture can tolerate a number of failures we use adversarial failure model. According to this model the fraction of component failures at any given time is bounded, i.e. only the fraction of component is allowed to be faulty. As number of component increases, so does the number of allowed failures. Moreover, the adversarial model assumes the worst case scenario in which the failures are inserted in a such way that they have the most negative effect on the decoding. In the adversarial timing-error model failure of a logic gate may occur when the gate output changes its value from the previous time instant.

3.3 Analysis of the Bit-Flipping Decoder

In this section we prove our main theorem which states that the correcting capability of the bit-flipping decoder built partially from unreliable gates increases linearly with expander code length. We assume that following two conditions are satisfied: (i) The MAJ gates used in the decoder are reliable, and XOR failures follow the timing-error mechanism introduced in Section 3.2.2, and (ii) no more than $|C_{XOR}|$ gates perturb their outputs in the first iteration. The need for previously described assumptions will be discussed shortly.

Now we formulate our main theorem. We also provide the proof as it reveals important facts on behavior of iterative decoders in presence of gate failures

Theorem 2. *Consider a $(\gamma, \rho, \alpha, (15/16 + \epsilon)\gamma)$ expander, $\epsilon \geq 0$. The bit-flipping decoder built from unreliable check nodes can correct any pattern of $|V_1| < \Big((3+16\epsilon)\alpha n - 4|C_{XOR}|\Big)/(5-16\epsilon)$ errors after at most $\lceil \log_{\frac{6}{\sqrt{13}+1}} [(1-16\epsilon)|V_1|/4 + |C_{XOR}|] \rceil + 3$ decoding iterations.*

Proof: Let V_i be the set of corrupt variables at the beginning of the i-th decoding iteration. The set of corrupt variables at the beginning of the $i+1$-th iteration (i.e., end of the i-th iteration), V_{i+1}, can be divided into two disjunct subsets: (i) $(V_{i+1} \cap V_i)$, the subset of corrupt variables that remained corrupt at the end of the i-th iteration, and (ii) $(V_{i+1} \setminus V_i)$, the subset of newly corrupted variables, i.e., variables that were correct in the $(i-1)$-th iteration but became corrupt during the i-th iteration. Let S_i be the set of variables that were corrected during $(i-1)$-th iteration and also stayed correct at the end of i-th iteration. Since the variables in S_i are flipped in the $(i-1)$-th iteration, from the definition of timing failure model, it follows that any variable in S_i may cause a perturbation of the neighboring XOR gate outputs in the i-th iteration and consequently the incorrect estimates of variables with whom it shares the neighbors. On the other hand, no perturbation of XOR-gate outputs occurs in the check nodes connected to only un-flipped variables in the $(i-1)$ iteration.

Each incorrect estimate of a particular variable in $V_{i+1} \setminus V_i$ is due to the variable's connection (through shared neighbors) to either variables from the set $V_i \cup S_i$ or some other variable in $V_{i+1} \setminus V_i$, which have caused the XOR gates perturbations in i-th iteration. Thus, each incorrect estimate indicates that a check is shared by two variables in $V_i \cup S_i \cup V_{i+1}$. On the other hand, there are no restrictions on possible neighbors of a check producing all correct estimates – they can be variables in $V_{i+1} \setminus V_i$ or variables outside of the set $V_i \cup S_i \cup V_{i+1}$. From Eq. 3.1, the number of correct estimates of each newly corrupt variable in $V_{i+1} \setminus V_i$ cannot be greater than $\gamma/2$, which means that the correct estimates are produced by at most $\gamma/2$ different neighboring check nodes. If the sets $V_{i+1} \setminus V_i$ and $V_i \cup S_i$ do not share any neighbors, each variable in $V_{i+1} \setminus V_i$ shares at least half of its neighbors with other variables in $V_{i+1} \setminus V_i$. This comes from the fact that the check node which sends an incorrect estimate to a node in $V_{i+1} \setminus V_i$ must be also connected to at least one other node in $V_{i+1} \setminus V_i$ which causes that incorrect estimate. Consequently, there are no more then $3\gamma/4$ checks connected to a variable in $V_{i+1} \setminus V_i$ that are not neighbors of $V_i \cup S_i$. Similarly, we can conclude that the number of neighbors of any variable in $V_{i+1} \cap V_i$ that are not the neighbors of $(V_i \setminus V_{i+1}) \cup S_i$ is also upper bounded by $3\gamma/4$. Now, we can bound the number of checks connected to $V_i \cup S_i \cup V_{i+1}$ as

$$\begin{aligned}|\mathcal{N}(V_i \cup S_i \cup V_{i+1})| &\leq \gamma(|V_i| - |V_i \cap V_{i+1}|) + \gamma|S_i| \\ &\quad + 3\gamma/4|V_{i+1} \setminus V_i| + 3\gamma/4|V_{i+1} \cap V_i| \\ &= \gamma|V_i| + \gamma|S_i| + 3\gamma/4|V_{i+1} \setminus V_i| \\ &\quad - \gamma/4|V_i \cap V_{i+1}|. \end{aligned} \tag{3.3}$$

If we assume that

$$|V_i \cup V_{i+1} \cup S_i| < \alpha n \tag{3.4}$$

for all $i > 0$, then, by the expansion property,

$$|\mathcal{N}(V_i \cup S_i \cup V_{i+1})| \geq (15/16 + \epsilon)\gamma(|V_i| + |S_i| + |V_{i+1} \setminus V_i|). \tag{3.5}$$

Combining the previous two inequalities we obtain

$$\begin{aligned}|V_i|(1 - 16\epsilon) &\geq (3 + 16\epsilon)|V_{i+1} \setminus V_i| + 4|V_{i+1} \cap V_i| \\ &\quad + (16\epsilon - 1)|S_i| \\ &\geq 3|V_{i+1}| - |S_i|. \end{aligned} \tag{3.6}$$

Because all elements of S_i were corrupt before $(i-1)$-th iteration we know that $|S_i| \leq |V_{i-1}|$, which based on previous inequality implies

$$|V_i| \geq 3|V_{i+1}| - |V_{i-1}|. \tag{3.7}$$

Let $|V_2| \leq \beta|V_1|$, $\beta > 0$. Then, based on Eq. (3.7), we can express upper bounds on the number of corrupt variables, for a few starting iterations as

$$\begin{aligned}|V_3| &\leq \frac{1+\beta}{3}|V_1|, \\ |V_4| &\leq \frac{1+4\beta}{3^2}|V_1|, \\ |V_5| &\leq \frac{4+7\beta}{3^3}|V_1| \ldots . \end{aligned} \tag{3.8}$$

It is not difficult to show that

$$|V_i| \leq \frac{a_{i-3} + \beta a_{i-2}}{3^{i-2}}|V_1|, \tag{3.9}$$

where $a_0 = a_1 = 1$ and

$$a_i = a_{i-1} + 3a_{i-2}, \tag{3.10}$$

for all $i \geq 2$. By solving the above difference equation we express the right side of Eq. (3.9) in a more convenient way. This is given by the following lemma.

Lemma 3. *The number of corrupt variables before i-th decoding iteration, $i > 1$, $|V_i|$ is bounded* *by*

$$|V_i| \leq \beta\Big(\frac{\sqrt{13}+1}{6}\Big)^{i-2}|V_1|. \tag{3.11}$$

■

In order to complete this part of the proof we have to analyze the first decoding iteration and bound $|V_2|$. In the following lemma we show that the upper bound of the value $|V_2|$ can be expressed in terms of $|V_1|$ and $|V_{XOR}|$, the number of XOR gate failures in the first iteration.

Lemma 4. *The number of corrupt variables after the first decoding iteration, $|V_2|$ is bounded by*

$$|V_2| \leq \frac{1-16\epsilon}{4}|V_1| + |C_{XOR}|. \tag{3.12}$$

Proof: From the analysis presented in [42] we know that the decoder build from reliable components reduces the number of corrupt variables to at most $(1-4\delta)|V_1|$, for all $\delta > 0$. The first summand in Eq. (3.12) is obtained noting that $\delta = 3/16 + \epsilon$. The second summand in Eq. (3.12) follows from the fact that each XOR gate failure can corrupt at most one additional variable. ■

Combining Eq. (3.11) and Eq. (3.12) we obtain

$$|V_i| \leq \Big(\frac{1-16\epsilon}{4}|V_1| + |C_{XOR}|\Big)\Big(\frac{\sqrt{13}+1}{6}\Big)^{i-2}. \tag{3.13}$$

The previous equation shows that the number of corrupt variables reduces over time, which after sufficient number of iteration leads to the correction of all initially corrupt variables.

Note that in our derivation we also assumed that $|V_i \cup V_{i+1} \cup S_i| < \alpha n$, for all $i > 0$ (Eq. (3.4)). We prove by mathematical induction stricter condition $|V_{i-1} \cup V_i \cup V_{i+1}| < \alpha n$, for all $i > 1$, since $|S_i| \leq |V_{i-1}|$.

Let assume that $|V_{i-2} \cup V_{i-1} \cup V_i| < \alpha n$. This means that Eq. (3.13) is satisfied for the first $i-1$ iterations and that we can use it to bound $|V_{i-1}|$ and $|V_i|$. Assume by the way of contradiction that $|V_{i-1} \cup V_i \cup V_{i+1}| \geq \alpha n$. Then, since we know that $|V_{i-1} \cup V_i| < \alpha n$, there must exists some $A \subset V_{i+1} \setminus (V_i \cup V_{i-1})$ for which $A \cup V_{i-1} \cup V_i = \alpha n$, and $|\mathcal{N}(A \cup V_{i-1} \cup V_i)| \geq (15/16+\epsilon)\gamma\alpha n$. On the other hand, for some δ, $15/16 + \epsilon < \delta \leq 1$, the number of checks connected to $A \cup V_{i-1} \cup V_i$ is bounded by

$$\begin{aligned}|\mathcal{N}(A \cup V_{i-1} \cup V_i)| &\leq \delta\gamma(|V_{i-1}| + |V_i|)\\ &\quad + 3\gamma/4(\alpha n - |V_{i-1}| - |V_i|).\end{aligned} \tag{3.14}$$

Combining previous relation with the lower bound given by the expansion we obtain

$$|V_{i-1}| + |V_i| \geq \frac{3+16\epsilon}{16\delta - 12}\alpha n \geq \frac{3+16\epsilon}{4}\alpha n, \tag{3.15}$$

which, based on Eq. (3.11), finally gives

$$|V_1| \geq \frac{4}{1-16\epsilon}\Big[\frac{3(3+16\epsilon)}{2(7+\sqrt{13})}\Big(\frac{6}{1+\sqrt{13}}\Big)^{i-3}\alpha n - |C_{XOR}|\Big], \tag{3.16}$$

The last inequality contradicts our initial assumption about $|V_1|$ given in the theorem formulation, and hence $|V_{i-1} \cup V_i \cup V_{i+1}| < \alpha n$ for all $i > 2$. When $i = 2$, Eq. (3.15) reduces to

$$|V_1| \geq \frac{(3+16\epsilon)\alpha n - 4|C_{XOR}|}{5-16\epsilon}, \tag{3.17}$$

which also contradicts our initial assumption. Finally, the condition $|V_1 \cup V_2| < \alpha n$ follows from the Eq. (3.12) and initial condition for $|V_1|$. This proves the theorem. ■

Corollary 3. *In the previous analysis we assumed that XOR logic are unreliable, but not the MAJ logic gates. If we allow MAJ logic gates to be prone to timing gate failures, error correction can not be guaranteed. This follows from the fact that in the worst case scenario correction of every variable can be annulled by the MAJ logic gate failure.*

Note that correcting capabilities of the decoder depends not only on the expansion property of the decoder, but also on the number of XOR failures in the first iteration ($|C_{XOR}|$). If there is sufficiently large number of XOR gates whose output is perturbed during the first iteration, the decoding process will not converge to the correct codeword. Recall from the timing failure model that $|C_{XOR}|$ depends on the XOR gates inputs at time instant prior to the first decoding iteration, on which we have limited knowledge. On the other hand, we can force all transistor-level transient processes in the decoding circuitry to reach a stationary state, and that before the decoding starts, there are no transitions at gate outputs nor âĂIJaccumulatedâĂİ errors. Practically, this can be done by slightly slowing down the clock in the first iteration and letting the signal level stabilize. Since the clock is slower, there are no-timing errors and the XOR computations are reliable, i.e. $|C_{XOR}|$=0.

We next compare our results with the results from [42] where a reliable decoder was considered. It can be observed that the presence of the XOR gate failures reduces the number of errors that can be tolerated by the bit-flipping decoder. Although in both cases the number of errors that can be corrected is upper bounded by αn, the noisy decoder requires higher graph expansion, to maintain the same correcting capability as that of the perfect decoder. For example, in order to correct $3\alpha n/5$ code bit errors, a perfect decoder would require expansion of $4\gamma/5$, while the graph of a noisy decoder needs expansion of at least $15\gamma/16$. Additionally, unreliable check node operations cause that less corrupt variables are corrected during decoding iterations, which increase the number of iterations needed for the correction of an error pattern.

The problem of explicit construction of expander graph with expansion arbitrary close to γ (called *lossless* expanders) was investigated by Capalbo *et al.* in [46], where it was shown that required expansion $15/16+\epsilon$ can be achieved with graph left degree $\gamma = \text{poly}(1-\gamma/\rho, 16/(1-16\epsilon))$. This proves the existence of a expander code that can tolerate fixed fraction of error under data-dependent gate failures.

Another proof of the guarantee error correction of LDPC codes was provided by Chilappagari *et al.* in [39], where the correction capabilities of a LDPC code was expressed in terms of girth of Tanner graph. In the following theorem we extend the results presented in [39] to the case of the noisy bit-flipping decoder.

Theorem 3. *Consider an LDPC code with γ-variable node-regular Tanner graph with $\gamma \geq 16$ and girth $g = 2g_0$. Then, bit-flipping decoder built from unreliable check nodes can correct any error pattern $|V_1|$ such that $|V_1| < (3n_0(\gamma/8, g) - 4|C_{XOR}|)/5$, where*

$$\begin{aligned} n_0(\gamma/8, g) &= n_0(\gamma/8, 2j+1) = 1 + \frac{\gamma}{8}\sum_{i=0}^{j-1}\left(\frac{\gamma}{8}\right)^i, \quad g \quad \textit{odd}, \\ n_0(\gamma/8, g) &= n_0(\gamma/8, 2j) = 2\sum_{i=0}^{j-1}\left(\frac{\gamma}{8}\right)^i, \quad g \quad \textit{even}. \end{aligned} \tag{3.18}$$

Proof: In order to prove the theorem we use the following lemma.

Lemma 5. *The number of checks connected to the set of V variable nodes in γ-variable node-regular Tanner graph with girth $g = 2g_0$ satisfies*

$$|N(V)| \geq \gamma|V| - f(|V|, g_0), \tag{3.19}$$

where $f(|V|, g_0)$ represents the maximal number of edges in an arbitrary graph with $|V|$ nodes and girth g_0.

Proof: See [39]. ■

Based on the Moore bound, we know that the number of nodes $n(\bar{d}, g_0)$ in a graph with the average degree $\bar{d} \geq 2$ and girth g_0 satisfies [47]

$$n(\bar{d}, g_0) \geq n_0(\bar{d}, g_0), \tag{3.20}$$

where $n_0(\bar{d}, g_0)$ is defined in Eq. (3.18). On the other hand, since $\gamma/8 \geq 2$ graph with $|V| < n_0(\gamma/8, g_0)$ nodes must have average degree smaller than $\gamma/8$. Then, based on the definition of average degree follows

$$f(|V|, g_0) < \gamma|V|/16. \tag{3.21}$$

Combining the previous expression with Eq. (3.5) we obtain

$$|N(V)| > 15\gamma/16. \tag{3.22}$$

■

Note that the authors in [39] showed that the $\gamma \geq 4$ represents sufficient condition that error correction is guaranteed on Tanner graph with girth g. However, due to logic gate failures higher expansions is required compared to error-free decoder implementation and we were able to come to the same conclusion only when $\gamma \geq 16$.

3.4 Application to Reliable Memories Build from Unreliable Components

In this section we investigate the problem of reliable storage of information in the unreliable memory cells. The information is stored as a codeword of an 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 $\tau, 2\tau, \ldots, L\tau, L \in \mathbb{N}$, based on the error correction scheme. In our memory architecture the update of the bit values is equivalent to one iteration of the bit-flipping algorithm, described in Section 3.2.

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 equivalent to the originally stored codeword. If we declare a *memory failure* every time that happens, our memory will be unreliable most of the time. Thus, if we want to recover the information, the final step of codeword extraction must be performed by reliable logic gates. In this paper we follow the system setup proposed by Taylor [3], 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 the finite number of iterations. We show that our memory architecture under certain conditions achieves arbitrary low memory failure probability.

We first investigate the memory reliability under the adversarial failure model, described in Section 3.2. Let α_m be the maximal fraction of failures that can affect memory cells between memory correcting cycles (iterations). In the Corollary 3, given in the previous section we stated that failures of MAJ gates can corrupt all variables corrected by the decoder, and hence, in order to preserve information over time we must bound the number of allowed failures per decoding iteration. Let α_γ be the maximal fraction of MAJ gate failures, during one decoding iteration. The following theorem gives the fraction of failures that can be tolerated by our memory architecture. We consider the case when $|C_{XOR}| = 0$, in which the signal is stabilized in the first iteration. This was justified in the Section 3.3.

Theorem 4. *The proposed memory architecture based on $(\gamma, \rho, \alpha, (15/16 + \epsilon)\gamma)$ expander code can preserve all stored information bits for an arbitrary long time period if*

$$\alpha_m + \alpha_\gamma \leq \frac{3 + 16\epsilon}{2(13 - 16\epsilon)}\alpha. \tag{3.23}$$

Proof: At t=0 a codeword of our expander code is written into the memory. The memory cells are updated at time instants $i\tau$, $i > 0$, 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 $|V(\tau - \delta_0)|$, where δ_0 denotes denote an infinitesimal duration of time, is bounded by

$$|V(\tau - \delta_0)| \leq n\alpha_m. \tag{3.24}$$

After the update cycle we have

$$|V(\tau)| \leq (\beta\alpha_m + \alpha_\gamma)n, \tag{3.25}$$

where $\beta = (1 - 16\epsilon)/4$. In the time interval $[\tau, 2\tau)$ there can be at most $\alpha_m n$ memory cells failures, which in the wort case will lead to $\alpha_m n$ additional corrupt variables. Then,

$$|V(2\tau - \delta_0)| \leq (\beta\alpha_m + \alpha_m + \alpha_\gamma)n. \tag{3.26}$$

Based on Eq. (3.7) and the previous discussion for all $i > 1$ we obtain

$$|V(i\tau)| \leq \frac{1}{3}\Big(|V(i\tau - \delta_0)| + |V((i-1)\tau - \delta_0)|\Big) + \alpha_\gamma n, \tag{3.27}$$

which leads to

$$|V(i\tau - \delta_0)| \leq \frac{(a_{i-2}\beta + b_{i-2})\alpha_m + c_{i-2}\alpha_\gamma}{3^{i-2}}n, \tag{3.28}$$

where $a_0 = a_1 = 1$, $b_0 = 1$, $b_1 = 5$, $c_0 = 1$, $c_1 = 4$ and

$$a_i = a_{i-1} + 3a_{i-2},$$

$$b_i = b_{i-1} + 3b_{i-2} + 3^i,$$

$$c_i = c_{i-1} + 3c_{i-2} + 3^i.$$

We next show that the number of corrupt memory cells can be upper bounded as presented in the following lemma.

Lemma 6. *The number of corrupt memory cells before the i-th update cycle $|V(i\tau - \delta_0)|$ for all $i > 0$ satisfies*

$$|V(i\tau - \delta_0)| < (3 + \beta)(\alpha_m + \alpha_\gamma)n. \tag{3.29}$$

■

Note that previous lemma can be used only if $|V(i\tau) \cup V(i\tau - \delta_0) \cup V((i-1)\tau - \delta_0)| < \alpha n$, for $i > 0$. Finding the condition under which the previous relation is satisfied gives upper bound on fraction of failures that can be tolerated. This proof is analogous to one presented in Section 3.3. Similarly, as it was done in the Section 3.3 we can find the condition that contradicts the previous relation. For that purpose we can rewrite Eq. (3.15) as follows

$$|V(i\tau - \delta_0)| + |V((i-1)\tau - \delta_0)| \geq \frac{3+16\epsilon}{4}\alpha n, \tag{3.30}$$

which leads to

$$\alpha_m + \alpha_\gamma > \frac{3+16\epsilon}{2(13-16\epsilon)}\alpha. \tag{3.31}$$

Then, we can conclude that Lemma 6 holds if

$$\alpha_m + \alpha_\gamma \leq \frac{3+16\epsilon}{2(13-16\epsilon)}\alpha. \tag{3.32}$$

The memory failure occurs when, at every time instants, the memory content cannot be decoded successively by a perfect bit-flipping decoder. The following lemma describes correction capabilities of the perfect decoder.

Lemma 7. *The bit-flipping decoder built from reliable components can correct any $\alpha_0 < (7+16\epsilon)/8$ fraction of errors, if the $(\gamma, \rho, \alpha, (15/16+\epsilon)\gamma)$ expander code is used.*
Proof: See [42]. ■

Since the fraction of corrupt memory cells at any time instant does not exceeds the error capability of the bit-flipping decoder, the memory content is preserved. This proves the theorem. ■

We next show how the right side of the Eq. (3.23), denoted by $\alpha_{total}(\alpha, \epsilon) = (3+16\epsilon)/[2(13-16\epsilon)]\alpha$, can be upper bounded. For that purpose the following lemma is used.

Lemma 8. *Let α^* and ϵ^* be such that $\alpha_{total}(\alpha^*, \epsilon^*) \geq \alpha_{total}(\alpha, \epsilon)$, $0 < \alpha < 1$, $0 < \epsilon \leq 1/16$. Then, they satisfy the following relation*

$$\epsilon^* = (1-(1-\alpha^*)^\rho)/(\alpha^*\rho) - 15/16. \tag{3.33}$$

Proof: The previous relation follows from the [42, Theorem 25] where it was shown that the set of αn variables can have at most $n\gamma(1-(1-\alpha)^\rho)/\rho + O(1)$ neighbors and the fact that we look for graphs which expand by at least a factor of $(15/16+\epsilon)$. ■

Based on the Eq. (3.33) we can numerically express the upper bounds for fixed values of ρ, which is presented in Table 1. Note that this upper bound does not depend on the parameter γ. It can be observed that by increasing the check node degree ρ, the number of neighbors of a set with αn variable nodes reduces. On the other hand, we require that a set with αn variable nodes have expansion of at least 15/16, which can be only satisfied with a reducing of α. Consequently, α_{total} is inversely proportional to ρ, and its highest value is obtained for ϵ close to zero.

Table 3.1: The upper bound values of tolerable fractions of failures.

ρ	α^*	ϵ^*	$\alpha_{total}(\alpha^*, \epsilon^*)$
16	0.0087	2.64×10^{-5}	0.002
20	0.0069	1.46×10^{-5}	0.0016
30	0.0045	2.92×10^{-5}	0.001
40	0.0033	2.44×10^{-5}	7.71×10^{-4}

We now expand the above results to the probabilistic failure model defined in Section 3.2. Let $\Delta_m > 0$ and $\Delta_\gamma > 0$ be such that $\varepsilon_m + \Delta_m = \alpha_m$ and $\varepsilon_g + \Delta_\gamma = \alpha_\gamma$, where failure rates ε_m and ε_g are defined in Section 3.2.2 and represent the probability of perturbation of a memory cell and MAJ logic gate, respectively. Then, when condition given by Eq. (3.18) is satisfied, the following lemma can be formulated.

Lemma 9. *The probability that memory failure occurs after L update cycles, $P(L)$, is bounded by*

$$P(L) \leq L(e^{-2\Delta_m^2 n} + e^{-2\Delta_\gamma^2 n}). \tag{3.34}$$

Proof: The proof is analogous to one provided in [21] and follows from the fact that by Chernoff bounds the probability of failure of more then a fixed fraction of component at time interval τ is bounded. ■

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

3.5 Conclusion

In this paper we have shown that the concept of guaranteed error correction can be expanded to faulty bit-flipping decoders, when the check node operations are prone to data-dependent failures. We also proved the existence of the memory architecture that achieves arbitrary small probability of failure under timing failure model, which presents the first such result under failure models other then the von Neumann model.

This paper raises a number of open questions in analysis of faulty decoders. The expander arguments have proven to be suitable for the analysis of the message passing decoders and it would be interesting to continue Burshtein and MillerâĂŹs work [44] on the irregular LDPC codes, for the case when node operations are unreliable. Additionally, we show that high graph expansions is required for guaranteed error correction of the faulty bit-flipping decoders. This requirement usually leads to complex decoders realizations and improving the required condition represents a major challenge. On the other hand, analysis of the expansion properties of graphs that correspond to the known quasi-cyclic LDPC codes can help us revealing which codes are more resistent to gate failures.

Chapter 4

MUDRI: A Fault-Tolerant Decoding Algorithm

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.*

The work presented in this chapter has been published in P. Ivanis, O. Al Rasheed, and B. Vasic, "MUDRI: A fault-tolerant decoding algorithm", *IEEE International Conference on Communications (ICC)*, London, UK, June 2015 [P4].

4.1 Introduction

High integration factor of integrated circuits together with low power consumption requirements makes emerging semiconductor devices inherently unreliable [19]. 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 [6] 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 [6] or one-step majority logic (MAJ) decoding [2], [5]. Methods for performance analysis of more complex decoders built from unreliable hardware based on the sum-product algorithm (SPA) [14] and its suboptimal (min-sum algorithm) version [27] have been also developed for transient failure model. In the similar context, finite-alphabet decoders (FAID) were analyzed by Huang and

Dolecek in [26]. Density evolution analysis of the simplest massage-passing algorithm (Gallager-B) implemented in noisy hardware is given in [23] and [25].

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 [25], numerous ingenious improvements of the BF algorithms have been proposed in the literature (see [48] and references therein) with the aim to close this gap. Recently, we proposed an modification of Gradient Descent Bit Flipping (GDBF) [49], appropriate for binary symmetric channel (BSC). The algorithm incorporates the idea of Probabilistic Bit Flipping (PBF) [50] 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 [51].

A probabilistic analysis of the PGDBF performed in this chapter 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 [52]. 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 chapter 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 chapter.

4.2 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 [49, Eqn. (6)]

$$\Delta_v^{(l)}(\chi, \eta) = \chi_v^{(l)} \eta_v + \sum_{c \in \mathcal{N}_v} \prod_{u \in \mathcal{N}_c} \chi_u^{(l)}, \tag{4.1}$$

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 [51], 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) [51]

$$\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = \hat{x}_v^{(l)} \oplus y_v + \sum_{c \in \mathcal{N}_v} \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(l)}. \tag{4.2}$$

In the PGDBF algorithm, the refreshed estimation in the l-th iteration is calculated as [51]

$$\hat{x}_v^{(l+1)} = \begin{cases} a_v \oplus \hat{x}_v^{(l)}, \Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = b^{(l)}, \\ \hat{x}_v^{(l)}, \Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) < b^{(l)}. \end{cases} \tag{4.3}$$

where $b^{(l)}$ denotes the largest value of the MIF at the l-th iteration, i.e., $b^{(l)} = \max\limits_v(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$, 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 [12], 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 $\max\limits_v{}_2(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$, the second MIF maximum is reached.

4.3 The MUDRI Algorithm

In this section, we propose a modification of PGDBF algorithm presented in [51]. We begin with three illustrative examples, exhibiting a method used to analyze probabilistic algorithms and presenting the intuition behind our decoder.

4.3.1 Example 1

Due to their very nature, probabilistic BF algorithms render inapplicable the trapping set analysis method developed for their deterministic counterparts [48]. In order to analyze the correctability of low-weight error patterns, let us consider a a three-bit error pattern shown in Fig. 4.1a, 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. 4.1b. Note that these nodes have different value compared to the initial values. As it results in the fixed set [53], 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 $s_2 = p^3(1 - p)^2$.

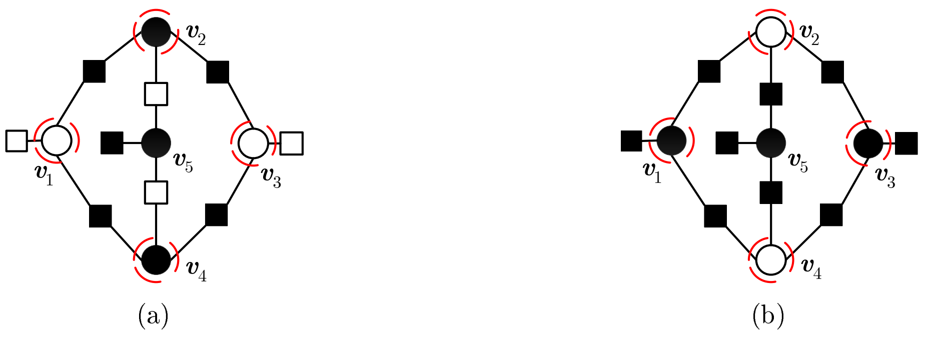


Figure 4.1: a) A three-bit pattern, b) the pattern in the second iteration of GDBF.

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

$$p_{PGDBF}(L) = 1 - \sum_{l=1}^{L} s_l. \tag{4.4}$$

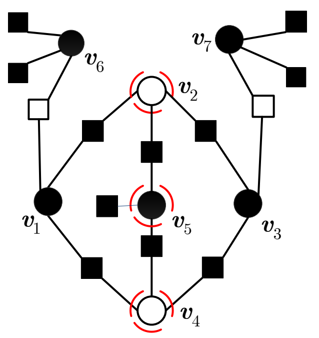
4.3.2 Example 2

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 [48] correct all triple errors for some codes). By using Eq. (4.4) 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 4.2a.

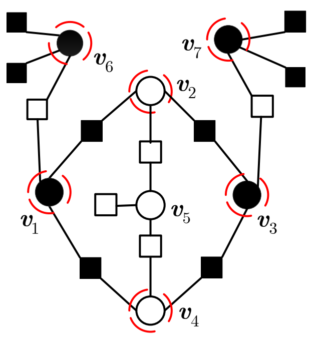
In the first iteration, $b^{(1)} = 3$ is associated with the variable nodes v_2, v_4 and v_5. The PGDBF update rule allows an independent flipping of all these variables (2^3 possible choices), but only some of them are actually flipped. If only v_5 is flipped (with the probability $p(1-p)^2$), the error pattern at the beginning of the second iteration looks like the one shown in Fig. 4.3b. 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

$$p_{MUDRI}(L, L_1) = \left(1 - \sum_{l=1}^{L_1} s_l\right)^{\lfloor L/L_1 \rfloor}. \tag{4.5}$$

In the special case when $L_1 = L$, we have a single attempt with L iterations, and the above expression reduces to Eq. (4.4). The probability of unsuccessful decoding can be minimized

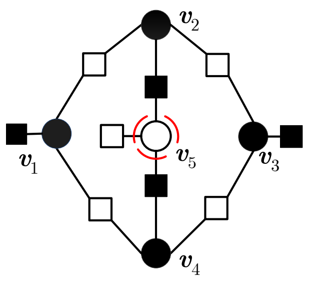


(a)

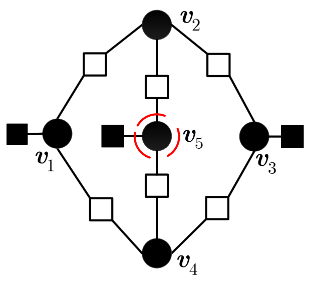


(b)

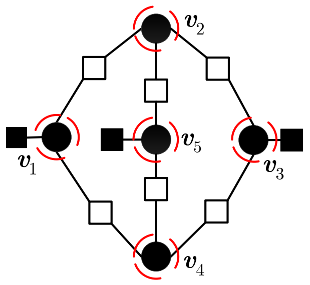
Figure 4.2: a) A five-bit error pattern uncorrectable by using GDBF, b) The second iteration of PGDBF, after the first iteration with the optimal choice.



(a)



(b)



(c)

Figure 4.3: a) A four-bit pattern critical in PGDBF, b) the second iteration if $b^{(l)} = 3$, c) the second iteration if $b^{(l)} = 2$.

with the proper choice of this parameter L_1.

4.3.3 Example 3

Finally, we show that the four bit error pattern presented in Fig. 4.3a 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. 4.3b) 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. $b^{(l)}_{mod} = \max_{v}{}_2(\Lambda^{(l)}_v(\hat{\mathbf{x}}, \mathbf{y}))$. In our example $b^{(2)}_{mod} = 1$, the nodes with $\Lambda^{(l)}_v(\hat{\mathbf{x}}, \mathbf{y})) \geq 1$ are flipped and the decoding is successful after the second iteration (Fig. 4.3c).

The above modifications are combined with the PGDBF algorithm [51] to obtain the MUDRI decoding algorithm, formally given in Algorithm 3. 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 $v_f^{(l)}$).

Algorithm 3 **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: $\hat{\mathbf{x}}^{(l)}$
```

4.4 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.

4.4.1 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 4.1a and 4.2a, 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. 4.4.

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. 4.2a 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

Algorithm 4 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
```

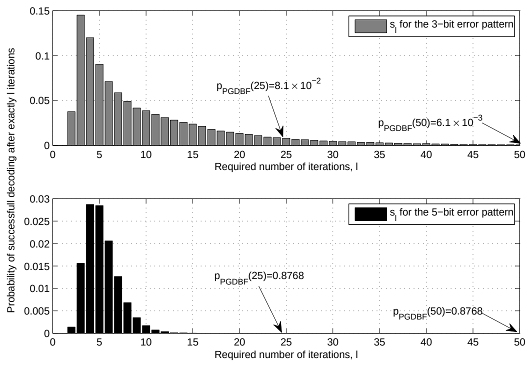


Figure 4.4: Probability distribution of the successful decoding in the l-th iteration of PGDBF, three-bit and five-bit error pattern, Tanner (155,64) code, $p = 0.7$.

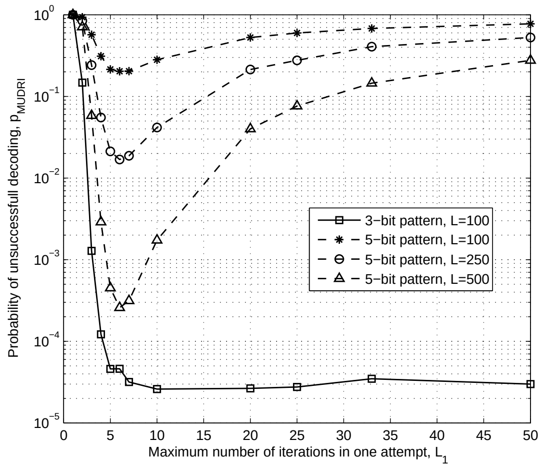


Figure 4.5: Probability of unsuccessful decoding for the three-bit and five-bit error pattern, MUDRI with $\lfloor L/L_1 \rfloor$ attempts per L_1 iterations each, $p = 0.7$.

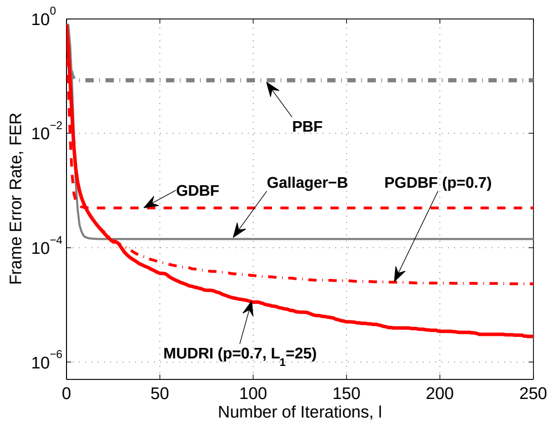


Figure 4.6: FER as a function of number of iterations l, Tanner $(155, 64)$ code, $\alpha = 0.01$, various decoding algorithms.

$p_{PGDBF}(14) = 0.8768$. 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. 4.5. It can be noticed that the best results are obtained for approximately $L_1 = 6$ decoding iterations per attempt.

In Fig. 4.6, the frame error rate (FER) as a function of number of iterations is presented for $\alpha = 0.01$. Although 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$=6 \times 10^{-7}$ when $L = 2000$.

4.4.2 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 [53] 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. 4.7. 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 the parameter p for both

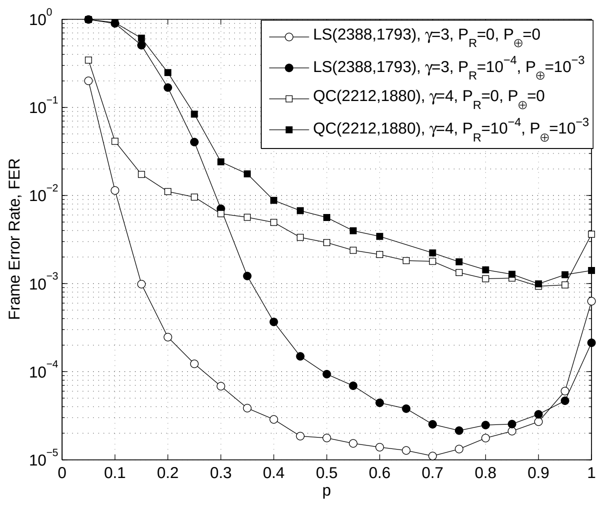


Figure 4.7: FER as a function of parameter p, LDPC codes with $\gamma = 3$ and $\gamma = 4$, $\alpha = 0.004$.

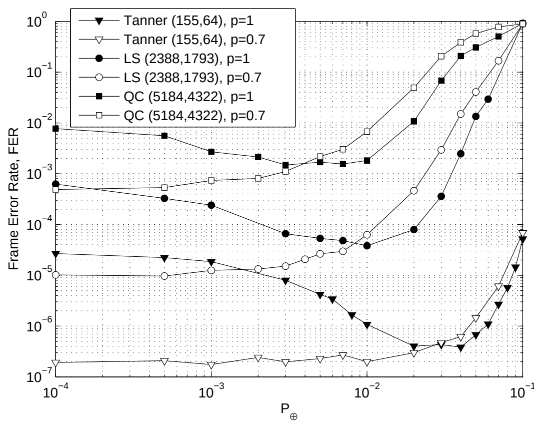


Figure 4.8: FER as a function of probability of error in XOR gates, $\alpha = 0.004$, $P_R = 0$, LDPC codes with $\gamma = 3$ and girth-8, with various code rates and codeword lengths.

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 $\gamma = 3$ [51]. 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. 4.8, 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.

In Fig. 4.9, we present the FER performance for five LDPC codes with various code constructions (QC, PEG, LS), column weights and codeword lengths (available in [54]), 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

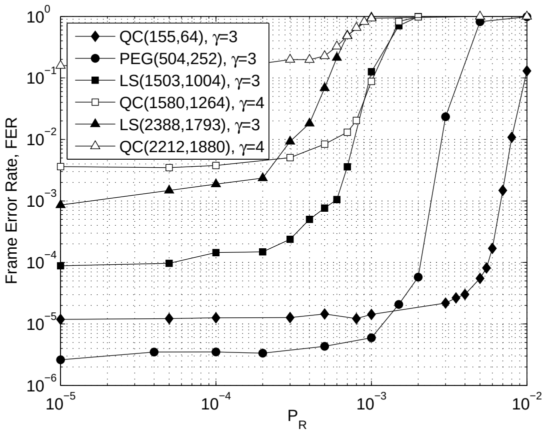


Figure 4.9: FER as a function of probability of error in registers, $\alpha = 0.008$, $P_{\oplus} = 10^{-3}$, LDPC codes with $\gamma = 3$ and girth-8, various codeword lengths.

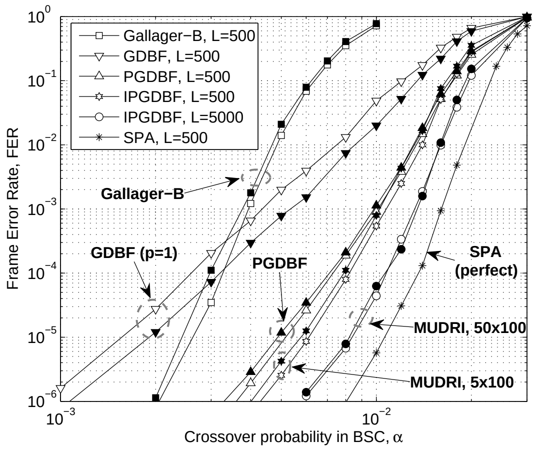


Figure 4.10: FER as a function of crossover probability in BSC channel. The code is LS(2388, 1793) ($\mathcal{C}_1$), the empty markers corresponds to perfect hardware and full markers to faulty hardware with $P_{\oplus} = 10^{-3}$, $P_R = 10^{-4}$, $p = 0.7$ in the PGDBF and MUDRI and the other decoding algorithms.

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. 4.10. 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 water-fall 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.

4.5 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.

Chapter 5

Robust Interconnect for Near/Sub-Threshold Region

Abstract: *In this chapter we present our initial results towards the construction of energy effective reliable data transport structures. Due to its intrinsic low power consumption we targeted the Near/Sub-Threshold operating region and proposed a dual-rail interconnection strategy, which outperforms the single-rail counterpart in terms of energy consumption at the expense of some area overhead. One remarkable feature of our proposal is the fact that it almost completely eliminates the overshot, which is very detrimental for signal integrity, thus the dual-rail interconnect we propose exhibits a built-in level of fault tolerance.*

5.1 Introduction

Within the context of deep-submicron (DSM) semi-conductor fabrication technologies era the contribution of interconnects to the Integrated Circuit (IC) performance became increasingly crucial. This is mostly associated with the fact that wire parasitic effects started to dominate digital ICs important figures of merit, such as speed, energy consumption, and reliability. This situation is unavoidable and is becoming more aggravated as the technology scales towards tens of nm feature size. Fig. 5.1 represents a simplified model of a 3-bit interconnect, which will be used within this section as a discussion vehicle, where CI is the inter-line capacitance and CL is the line-to-ground capacitance.

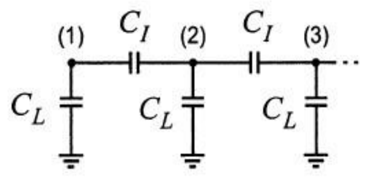


Figure 5.1: The RC model of a 3-bit line

5.1.1 Delay

It is well accepted that interconnect delay has a huge impact on the overall delay of state of the art digital ICs implemented in DSM technologies. Due to the existence of big parasitic elements, such as resistors, capacitors the circuit performance can dramatically degrade. Moreover, with technology scaling the crosstalk among adjacent wires, i.e., the situation when a signal transition on one line induces traveling waves on adjacent lines, is becoming a significant contributor to data transmission delay value and its variation. When crosstalk occurs transition on neighbouring wires (aggressors), e.g., line (1) and (3) in Fig. 5.1, can either speed up or slow down the transition on victim wires, line (2) in our example. We note that the worst-case delay occurs when the victim line and the two aggressors are switching in opposite directions, as illustrated in Fig. 5.2. In this case, both coupling capacitances (between the victim and the two aggressors) need to be charged. The coupling (crosstalk) effect can result in overshoot for the signals, which is harmful to the reliability. Because all the circuits work within a certain range of operating voltage, big offset from that range causes high peak current and also low noise margin, which could most likely lead to some transient error and/or shorter lifetime.

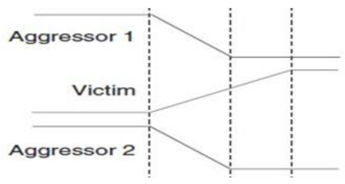


Figure 5.2: The worst-case delay for a 3-bit line.

The mitigation of the crosstalk consequences has been investigated and up to date some approaches have been proposed. Signal line duplication (wire redundancy) has been introduced in [55] and can fundamentally avoid the worst delay case at the expense of a significant area overhead. Bus encoding is another approach to optimize the interconnect delay and to eliminate the worst switching cases. Several encoding algorithms have been introduced, such as bus inverter algorithm [56], one to one mapping scheme [57], and some data dependent methodologies [57]. One can also choose to manipulate the space between adjacent wires, thus not to follow the layout design rules, to further reduce the inter-line capacitances and hence the delay. Moreover, in conjunction with the previously mentioned techniques, one may try to maintain the signal integrity and speed on long wires by breaking them into shorter segments connected by repeaters.

5.1.2 Energy

Apart of inducing IC delay increase long interconnects also contribute a significant amount to the power dissipated in digital ICs. It is well known that energy consumption is highly related to the value of the capacitances present in circuits. Thereby, for long interconnects, high capacitance is unavoidable. Additionally, given that the space between wires is following the fabrication process feature size scaling the value of the Miller capacitor is increasing. Thus it is clear that reducing crosstalk effects is also beneficial for energy efficiency. However, unlike the delay normally determined by the worst case, the bus energy dissipation is affected by the overall crosstalk across the bus lines. Furthermore, for the consideration of performance and signal integrity, repeaters are necessary for long wires, which contribute significantly or even in a major part to the total bus energy consumption. As for diminishing the energy consumption, bus-encoding schemes, which take advantage of spatial and temporal redundancy or use weighted opposite-transition-forbidden codes have been introduced in [58]. These coding schemes require extra hardware at both encoding and decoding side of the circuits, which can be quite expensive in terms of area and themselves can be major energy consumption contributors. Also, such schemes are not "panacea universalis" as their effectiveness is limited to certain data distribution types. One straightforward method to save energy is to dramatically decrease the voltage supply VDD, down to Near/Sub-Threshold Region (NSTR), which is in line with the i-RISC project philosophy. We note that the conventional repeater insertion technique commonly used in super-threshold domain does not provide any performance improvement in case of long interconnect length under sub-threshold conditions. The state of the art in this region focuses on the material choice, such as single-wall carbon nanotube, Cu, and mixed carbon nanotube bundle, etc. [59].

5.2 Dual-Rail Interconnect for Near/Sub-Threshold Operation

In this section we introduce an approach to enhance performance and robustness of sub-powered interconnect while not losing the benefit of low energy consumption. The basic idea is to transmit signals in the dual-rail format while using cross-coupled inverters rather than conventional buffers as drivers to strengthen the signals, as depicted in Fig. 5.3. This structure has the capability to magnificently supressed the overshoot magnitude because of the fact that no inverter is directly involved in the propagation line. Moreover, the transmission delay can also be reduced due to the positive feedback capability of the cross-coupled inverters. However, it should be understood that power consumption would be increased inevitably subject to the number of lines involved, which means larger resistor and capacitor, but an overall lower energy consumption can still be achieved even for wider interconnects. A down side of this approach is the area overheard as such interconnect can consume up to two times more area than the single-rail counterpart. We note however that in the nano era chip real estate is perceived like a commodity thus it is not that important any longer while energy consumption and reliability are at premium. To

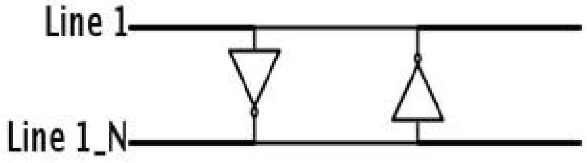


Figure 5.3: Dual-rail interconnect for NSTR operation.

get a first inside on the practical potential of our proposal we built three 3-bit $500\mu m$ long interconnects as follows: (i) single-rail with standard repeater insertion strategy, (ii) dual-rail using normal repeater, and (iii) dual-rail according to the proposed scheme. The distributed RC line model has been implemented and to explore the worst-case scenario, we considered the case when line (1) and (3) are switching in the same direction while line (2) switches in the opposite direction (see Fig. 5.1). We utilized the 45nm PTM CMOS model in our simulation. We set the temperature to 75°C and to cover the NSTR operation range we made the voltage supply, VDD, sweep from 0.2V to 0.5V with a 50mV step. The corresponding power, delay, energy, and maximal overshoot value obtained for the three designs are summarized in Table 5.1. Fig. 5.4 depicts the corresponding power, delay, and energy, respectively. It can be observed that, when compared with the single-rail counterpart, the proposed approach brings significant delay improvement with some penalty in power consumption but it is always reducing the overall power delay product, which is the energy consumption. When compared with the dual-rail using the conventional repeater insertion scheme, the proposed interconnect has slightly smaller area requirements and consume less power/energy at the expense of about 20% more delay. It should be noticed that we only investigate the worst case here rather than the average case while no encoding scheme is utilized. The most impressive improvement of our proposal is the suppression of the overshoot. When compared with the two conventional approaches, the overshoot is negligible in our design, which can be quite beneficial to the signal integrity. Note that the overshoot magnitude ranges from 14% to 32% and from 7% to 18% of the VDD value, for the conventional single-rail and dual-rail design, respectively, while it is about two orders of magnitude smaller for the proposed dual-rail approach. The low energy consumption can also be related to this characteristic due to the fact that the charge on inter-wire capacitance is significantly reduced.

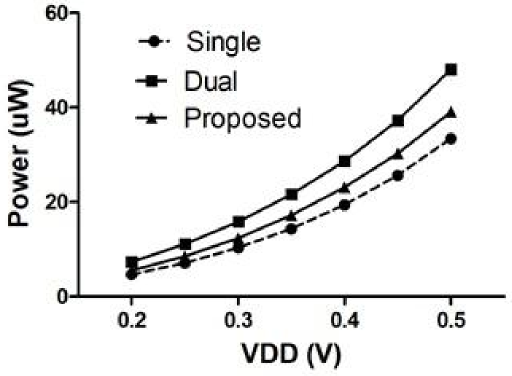
It can be concluded that the proposed dual-rail cross-coupled inverter methodology for NSTR interconnect is promising for its delay improvement against the conventional single-rail case, and its power efficiency against the conventional dual-rail model, and more important, the substantial overshoot reduction. As a follow up work we plan to investigate the further potential of this approach by combining it with other encoding schemes in order to achieve even higher energy efficiency.

Table 5.1: Power, delay, energy, and maximum overshoot.

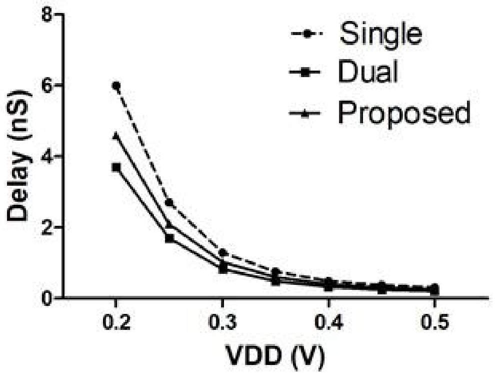
3-bit (500um)	POWER (μW)			Delay (nS)		
$V_{DD}(V)$	Single	Dual	Proposed	Single	Dual	Proposed
0.2	4.7	7.3	5.5	6	3.7	4.6
0.25	7.1	11.1	8.5	2.7	1.7	2.1
0.3	10.3	15.8	12.4	1.3	0.83	1.03
0.35	14.3	21.6	17.2	0.76	0.49	0.6
0.4	19.4	28.6	23.1	0.49	0.33	0.4
0.45	25.6	37.3	30.3	0.38	0.25	0.3
0.5	33.4	48.1	39.1	0.31	0.21	0.25
3-bit (500um)	Energy (fJ)			Max Overshoot (mV)		
$V_{DD}(V)$	Single	Dual	Proposed	Single	Dual	Proposed
0.2	28.2	27.0	25.3	28.5	25.7	0.5
0.25	18.9	18.9	17.8	38.9	34.9	1.4
0.3	13.4	13.1	12.8	53.9	47.6	2.6
0.35	10.9	10.6	10.3	75.9	62.1	2.8
0.4	9.64	9.4	9.24	121.6	61.9	2.9
0.45	9.66	9.3	9.1	137.9	50.7	3.2
0.5	10.3	10.1	9.6	162.9	36.5	2.7

5.3 Conclusion

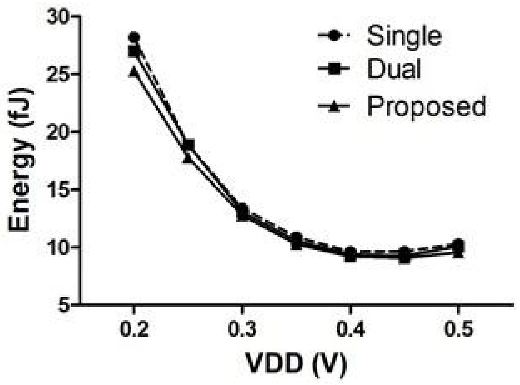
In this section we made some preliminary steps towards the construction of energy effective reliable data transport structures, i.e., interconnects. To this end we targeted the Near/Sub-Threshold operating region and proposed a dual-rail interconnection strategy, which outperforms the single-rail counterpart in terms of energy consumption at the expense of some area overhead. One remarkable feature of our proposal is the fact that it almost completely eliminates the overshot, which is very detrimental for signal integrity, thus the dual-rail interconnect we propose exhibits a certain level of fault tolerance by construction. In the last 3rd year of the project we plan to continue this research avenue and investigate the impact of various coding schemes, e.g., constrained coding, on the energy consumption and reliability of the proposed interconnect scheme.



(a) Power



(b) Delay



(c) Energy

Figure 5.4: Power, delay, and energy for single-rail, dual-rail, and the proposed dual-rail.

Chapter 6

General Conclusion and Next Steps

During the M13-M24 period we have made considerable progress beyond the state of the art on the analysis of hard decision decoders, under data-dependent gate failure models. We have also designed probabilistic bit-flipping decoders robust to hardware unreliability, and initiated research on reliable data transport. The main contributions during this period for WP4 can be summarized as follows:

- We developed the analytic tool for the finite code length analysis of the one-step majority logic decoders, subjected to data-dependent logic gate failures. This tool enable us to evaluate the performance of memory architectures that are based on one-step majority logic decoding.
- We demonstrated that the guaranteed error correction concept can be extended to the faulty iterative decoders. We proved that the error correction capability of the faulty bit-flipping decoder increases linearly with code length, even when logic gates used in the decoder are prone to data-dependent failures. In addition we proved the existence of reliable memories under correlated gate failures, which represents the first such result for failure models other than the simple independent failure model.
- We designed a fast convergent bit-flipping decoder robust to hardware unreliability. Our decoder outperforms all known bit-flipping decoders in terms of error correction capabilities.
- We proposed a dual-rail interconnection strategy tailored for Near/Sub-Threshold operation, which outperforms traditional the single/dual-rail counterparts in terms of energy consumption.

The topics presented in this deliverable will be further investigated in the third year of the project. New challenges for the rest of the project in WP4 include the following:

- Investigate the guaranteed error correction analysis of the Gallger-B decoder, as well as novel designs of hard decision message passing decoders robust to hardware unreliability.
- Continue the work towards the understanding of the decoding failures of the probabilistic bit-flipping decoder, so that to further improve its error correction performance. We also intend to use this work as a guideline for the development of deterministic decoders with approximately the same performance but lower complexity levels.
- Analyze the novel hard decision decoders proposed in this project under data-dependent failure models.
- Investigate the design of constraint codes in order to avoid crosstalk error prone data patterns, which potentially enables robust and low energy intra/inter-chip data transmission.

Bibliography

[1] M. Ivkovic, S. K. Chilappagari, and B. Vasic, "Construction of memory circuits using unreliable components based on low-density parity-check codes," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'06)*, San Francisco, CA, USA, Nov. 2006, pp. 1–5.

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

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

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

[5] 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.

[6] 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.

[7] 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.

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

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

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

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

[12] O. Al 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.

[13] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[14] L. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, July 1973.

[15] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," *IEEE Communications Letters*, vol. 18, no. 5, pp. 849–852, May 2014.

[16] C. K. Ngassa, V. Savin, E.Dupraz, and D. Declercq, "Density Evolution and Functional Threshold for the Noisy Min-Sum Decoder," *Accepted at IEEE Transactions on Communications*, December 2014.

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

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

[19] 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.

[20] B. Vasic, S. Chilappagari, S. Sankaranarayanan, and R. Radhakrishnan, "Failures of the Gallager B decoder: analysis and applications," in *Proceedings of 2nd Information Theory and Applications Workshop (ITA 2006)*, San Diego, CA, Feb. 2006, paper 160, [Online Available:] http://ita.ucsd.edu/workshop/06/papers/160.pdf.

[21] S. K. Chilappagari and B. Vasic, "Reliable memories built from unreliable components based on expander graphs," *arXiv:0705.0044v1 [cs.IT]*, May 2007.

[22] S. M. Sadegh Tabatabaei Yazdi, C. H. Huang, and L. Dolecek, "Optimal design of a Gallager B noisy decoder for irregular LDPC codes," *IEEE Communications Letters*, vol. 16, no. 12, pp. 2052–2055, Dec. 2012.

[23] S. M. Sadegh 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.

[24] C. H. Huang, Y. Li, and L. Dolecek, "Gallager B LDPC decoder with transient and permanent errors," *IEEE Transactions on Communications*, vol. 62, no. 1, pp. 15–28, Jan. 2014.

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

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

[27] C. Kameni Ngassa, V. Savin, and D. Declercq, "Min-Sum-based decoders running on noisy hardware," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'13)*, Atlanta, USA, Dec. 2013, pp. 1–5.

[28] 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.

[29] L. D. Rudolf, “A class of majority logic decodable codes,” *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 305–307, Apr. 1967.

[30] J. L. Massey, *Threshold Decoding.* Cambridge, MA, USA: MIT Press, 1963.

[31] R. Radhakrishnan, S. Sankaranarayanan, and B. Vasic, “Analytical performance of one-step majority logic decoding of regular LDPC codes,” in *Proceedings of IEEE International Symposium on Information Theory (ISIT 2007)*, Nice, France, June 2007, pp. 231–235.

[32] 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.

[33] 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, Sept.–Oct. 2012, pp. 322–328.

[34] 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.

[35] S. Lin and D. J. Costello, *Error control coding, 2nd Edition.* Englewood Cliffs, NJ: Prentice-Hall, 2004.

[36] Y. Kou, S. Lin, and M. Fossorier, “Low-density parity-check codes based on finite geometries: A rediscovery and new results,” *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[37] R. Lucas, M. P. C. Fossorier, Y. Kou, and S. Lin, “Iterative decoding of one-step majority logic deductible codes based on belief propagation,” *IEEE Transactions on Communications*, vol. 48, no. 6, pp. 931–937, June 2000.

[38] N. Pippenger, “On networks of noisy gates,” in *Proceedings of 26th Annual Symposium on Foundations of Computer Science*, Portland, OR, USA, Oct. 1985, pp. 30–38.

[39] S. K. Chilappagari, D. V. Nguyen, B. Vasic, and M. W. Marcellin, “On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes,” *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1600–1611, Apr. 2010.

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

[41] V. V. Zyablov and M. S. Pinsker, “Estimation of the error-correction complexity for Gallager low-density codes,” *Probl. Inf. Transm.*, vol. 11, no. 1, pp. 18–28, 1976.

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

[43] D. Burshtein, “On the error correction of regular LDPC codes using the flipping algorithm,” *IEEE Transactions on Information Theory*, vol. 54, no. 2, pp. 517–530, Feb. 2008.

[44] D. Burshtein and G. Miller, “Expander graph arguments for messagepassing algorithms,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 782–790, Feb. 2001.

[45] J. Feldman, T. Malkin, R. A. Servedio, C. Stein, and M. J. Wainwright, "LP decoding corrects a constant fraction of errors," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 82–89, Jan. 2007.

[46] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson, "Randomness conductors and constant-degree lossless expanders," in *STOC'02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, New York, NY, USA: ACM Press, 2002, pp. 659–668.

[47] N. Alon, S. Hoory, and M. Linial, "The moore bound for irregular graphs," *Graphs and Combinatorics*, vol. 18, no. 1, pp. 53–57, 2002.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

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

[55] D. Rossi, C. Metra, A. K. Nieuwland, and A. Katoch, "Exploiting ECC redundancy to minimize crosstalk impact," *IEEE Design and Test of Computers*, vol. 22, no. 1, pp. 59 – 70, Jan.-Feb. 2005.

[56] K. S. Sainarayanan, C. Raghunandan, and M. B. Srinivas, "Bus encoding schemes for minimizing delay in VLSI interconnects," in *Proceedings of the 20th annual conference on Integrated circuits and systems design*, 2007, pp. 184–189.

[57] V. Bret and K. Keutzer, "Bus encoding to prevent crosstalk delay," in *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, 2001, pp. 57–63.

[58] A. R. Brahmbhatt, J. Zhang, Q. Qiu, and Q. Wu, "Adaptive low-power bus encoding based on weighted code mapping," in *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, 2006, pp. 4 pp– 1742.

[59] S. D. Pable and M. Hasan, "Interconnect design for subthreshold circuits," *IEEE Transactions on Nanotechnology*, vol. 11, no. 3, pp. 633 – 639, Feb. 2012.