Statistical Data Correction for Unreliable Memories

Christoph Roth[*], Christoph Studer[†], Georgios Karakonstantis[‡], and Andreas Burg[‡]

[*] Integrated Systems Laboratory, ETHZ, Zürich, Switzerland
[†] School of Electrical and Computer Engineering, Cornell University, Ithaka, NY, USA.
[‡] Telecommunications Circuits Laboratory, EPFL, Lausanne, Switzerland
Email: rothc@iis.ee.ethz.ch, studer@cornell.edu, georgios.karakonstantis, andreas.burg@epfl.ch

***Abstract*—In this paper, we introduce a statistical data-correction framework that aims at improving the DSP system performance in presence of unreliable memories. The proposed signal processing framework implements best-effort error mitigation for signals that are corrupted by defects in unreliable storage arrays using a statistical correction function extracted from the signal statistics, a data-corruption model, and an application-specific cost function. An application example to communication systems demonstrates the efficacy of the proposed approach.**

I. Introduction

A main driver for the enormous success of modern digital signal processing (DSP) systems during recent years has been the ability of the underlying integrated circuits to perform computations and to store data in a 100% reliable and reproducible manner. Unfortunately, the growing effect of semiconductor-process parameter variation and the related reliability issues that come along with modern deep-submicron CMOS technologies are more and more putting an end to this ideal behavior. In particular, the small feature sizes combined with inaccuracies in the complex fabrication processes of technology nodes beyond 45 nm lead to a strong variability of transistor characteristics within and across different fabricated integrated circuits, which can lead to a variety of failures.

Embedded memories are specifically prone to process variations as they are typically implemented using the smallest feature sizes provided by the target process with the goal of maximizing storage density. Current approaches to maintain a high fabrication yield under these conditions rely on conservative design techniques, such as the use of larger bit-cells, or the use of circuit-level error-correction coding (ECC) schemes that add redundancy to stored data [1]–[3]. Unfortunately, these techniques entail a significant overhead in terms of silicon area and power consumption, thus contradicting with the low power and high memory density requirements of modern wireless communications and video processing system-on-chips [4]. Furthermore, these techniques are mostly a waste of resources as they merely serve as precaution to maintain correct functionality even for outliers in the manufacturing process. Consequently, realizing cost effective and energy-efficient DSP systems in the near future necessitates a paradigm shift toward fault-tolerant circuit and systems that are resilient to the various non-idealities and impairments caused by unreliable silicon [5].

To our advantage, a host of applications, such as wireless communications and video processing, are inherently fault tolerant as they deal with stochastic signals that are already distorted by noise and/or interference and naturally contain redundancy. It has been observed that the performance of such systems degrades only gracefully under a certain amount of hardware failures induced by unreliable silicon components, provided that the corresponding DSP algorithms and architectures are designed to take such failures into account [6], [7]. Recent works [8], [9] have also shown that the protection of only a small subset of carefully selected bits by using known circuit techniques such as larger memory bit-cells suffices to guarantee acceptable performance even under large failure rates.

In this paper, we propose an alternative system-level method for improving the robustness of fault-tolerant DSP systems against data-corruptions in embedded memories. Instead of adding redundancy via circuit-level ECC or using larger bit-cells, we propose a data correction approach that statistically corrects the output of unreliable memories using a correction function that is based on an application-specific, probabilistic cost function and on side information (Sec. II). As a proof-of-concept, we consider a hardware related memory failure model and show an application example for communication systems. Our results demonstrate that the deployment of statistical data correction can limit the error-rate performance under memory failures (Sec. III). We conclude by summarizing potential applications of the proposed framework (Sec. IV).

II. Robustness via Statistical Data Correction

We consider a fault-tolerant DSP system as depicted in Fig. 1. The system comprises two general signal processing blocks A and B that communicate via an unreliable memory. The K-valued discrete input data $d \in \mathcal{D} = \{d_1, \ldots, d_K\}$ of the unreliable memory is assumed to be distributed according to the probability mass function $P_d(d_k) = \Pr(d = d_k)$. To store the data in the unreliable memory, a mapping function $\Delta(d)$, which determines the *data representation*, maps each symbol d_k to a N-dimensional binary-valued label vector $\mathbf{s}_k$,

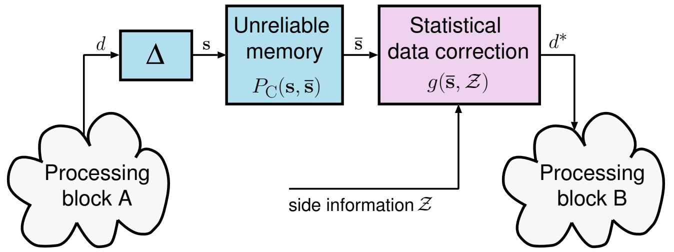

d
Δ
$\mathbf{s}$
Unreliable memory
$P_{\mathrm{C}}(\mathbf{s}, \bar{\mathbf{s}})$
$\bar{\mathbf{s}}$
Statistical data correction
$g(\bar{\mathbf{s}}, \mathcal{Z})$
d^*
Processing block A
side information $\mathcal{Z}$
Processing block B

Fig. 1. Statistical data correction in unreliable DSP systems: A data mapping function Δ maps digital information to binary-valued labels that are stored in an unreliable memory. The statistical data correction then corrects the memory output using a correction function g together with side information $\mathcal{Z}$ about the state of the system.

which we will refer to as *label*. We assume the function Δ to be bijective and thus, we have $K = 2^N$.

Commonly used data representations in DSP systems are the 2's complement and the sign-magnitude number formats, which enable the efficient implementation of basic arithmetic operations in digital integrated circuits [10]. In this paper, we solely focus on 2's complement—however, it is worth mentioning that by choosing different data representations one can further reduce the impact of reliability issues on the quality of fault-tolerant DSP systems (see [11] for more details).

A. Model for Unreliable Memories

Data corruption in unreliable memories is modeled as a probabilistic channel that maps input labels $\mathbf{s}$ to output labels $\bar{\mathbf{s}}$ according to a label cross-over probability mass function $P_{\mathrm{C}}(\mathbf{s}_k, \bar{\mathbf{s}}_{k'}) = \Pr(\bar{\mathbf{s}} = \bar{\mathbf{s}}_{k'} \,|\, \mathbf{s} = \mathbf{s}_k)$ (see Fig. 1). In what follows, we model the physical memory bit-cell errors using the stuck-at channel model [11], which matches the standard-failure model for embedded memories in nanometer CMOS technologies that are affected by process variations [12]. This model assumes that each bit-cell fails independently with a bit-cell error probability ε (which is known for a given technology node and circuit topology). In addition, a faulty bit cell is either stuck-at-0 or stuck-at-1 with equal probability. The resulting label cross-over probabilities are given by [11]

$$P_{\mathrm{C}}(\mathbf{s}_k, \bar{\mathbf{s}}_{k'}) = \sum_{\ell=0}^{N-d_H(\mathbf{s}_k,\bar{\mathbf{s}}_{k'})} \binom{N - d_H(\mathbf{s}_k, \bar{\mathbf{s}}_{k'})}{\ell} \times \left(\frac{\varepsilon}{2}\right)^{d_H(\mathbf{s}_k,\bar{\mathbf{s}}_{k'})+\ell} (1-\varepsilon)^{N-d_H(\mathbf{s}_k,\bar{\mathbf{s}}_{k'})-\ell}, \qquad (1)$$

where $d_H(\mathbf{s}_k, \bar{\mathbf{s}}_{k'})$ denotes the Hamming distance between the label vectors $\mathbf{s}_k$ and $\bar{\mathbf{s}}_{k'}$. We emphasize that the fault model can easily be replaced and adapted to the underlying memory type, e.g., to model the errors in multi-level Flash memories.

B. Statistical Data Correction

To mitigate the errors induced by unreliable memories, we propose to statistically correct the memory output-data based on *side information* $\mathcal{Z}$ obtained from the system.

1) Sets of side information: For a more clear distinction between system and hardware properties, we consider two disjoint sets of side information $\mathcal{Z}_{\mathrm{S}}$ and $\mathcal{Z}_{\mathrm{H}}$, respectively, with $\mathcal{Z} = \{\mathcal{Z}_{\mathrm{S}}, \mathcal{Z}_{\mathrm{H}}\}$. The first set $\mathcal{Z}_{\mathrm{S}}$ includes a-priori-known statistical properties of the DSP system at hand, such as the instantaneous signal-to-noise ratio (SNR) in communication systems or the distribution of a particular wavelet coefficient in video compression. The second set $\mathcal{Z}_{\mathrm{H}}$ includes information about the state of the unreliable memory, such as the state of individual bit-cells, e.g., whether they are stuck-at or not. This side information can be obtained from production tests of the fabricated dies, built-in self tests, or dedicated error-detection circuits within the memory readout logic.

2) Correction function: The *correction function* $g(\bar{\mathbf{s}}, \mathcal{Z})$ corrects the output of the unreliable memory based on side information $\mathcal{Z}$ and the potentially faulty observation $\bar{\mathbf{s}}$. This function is defined offline by an application-specific (probabilistic) cost function $\mathcal{C}$, which takes into account the impact of the unreliable memory on the system performance and is also conditioned on the observed memory output-label $\bar{\mathbf{s}}$ and on the side information $\mathcal{Z}$. Specifically, the correction function $d^* = g(\bar{\mathbf{s}}, \mathcal{Z})$ is given by the following optimization problem:

$$d^* = g(\bar{\mathbf{s}}, \mathcal{Z}) \triangleq \arg\min_{\bar{d}} \mathcal{C}(\bar{d} \,|\, \bar{\mathbf{s}}, \mathcal{Z}). \qquad (2)$$

3) Example cost function: As an example, consider the expected mean squared error (MSE) between the memory input d and the corrected output value $\bar{d}$ as a cost function

$$\mathcal{C}(\bar{d} \,|\, \bar{\mathbf{s}}, \mathcal{Z}) \triangleq \mathbb{E}\big\{(d - \bar{d})^2 \,|\, \bar{\mathbf{s}}, \mathcal{Z}\big\} \qquad (3)$$

with the expectation taken over the memory input d. Note that the cost function (3) accounts for the statistics of processing block A in Fig. 1 as well as the data corruption effects inside the unreliable memory modeled by $P_{\mathrm{C}}(\mathbf{s}, \bar{\mathbf{s}})$. A resulting correction function would depend on the distribution P_d of the memory input-data resulting from processing block A, which is typically known in dedicated VLSI circuits for DSP systems.

C. Implementation of the Correction Function

The total number of individual instances of $\mathcal{Z}_{\mathrm{S}}$ and $\mathcal{Z}_{\mathrm{H}}$ is denoted by Z_{S} and Z_{H}, respectively. For each instance of side information $\mathcal{Z}_i$ ($i = 1, \ldots, Z_{\mathrm{S}} \times Z_{\mathrm{H}}$) and for each observed memory output-label $\bar{\mathbf{s}}_k$, the corresponding corrected value $g(\bar{\mathbf{s}}_k, \mathcal{Z}_i)$ is obtained by minimizing the cost function $\mathcal{C}$ conditioned on $\bar{\mathbf{s}}_k$ and $\mathcal{Z}_i$ (as shown in Eq. 2). The minimization can be performed either analytically, or by means of Monte-Carlo simulations, if no closed-form evaluation and minimization of the cost function is possible. Once the $Z_{\mathrm{S}} Z_{\mathrm{H}}$ available correction functions $g(\bar{\mathbf{s}}_k, \mathcal{Z}_i)$ have been computed, they can be stored in the system. In practical systems, an application of the correction function simply amounts to table look-ups (parametrized by i and k), which can be implemented as conventional on-chip look-up tables with very low hardware complexity if $K Z_{\mathrm{S}} Z_{\mathrm{H}}$ (the total side information that need to be stored) is reasonably small [10].

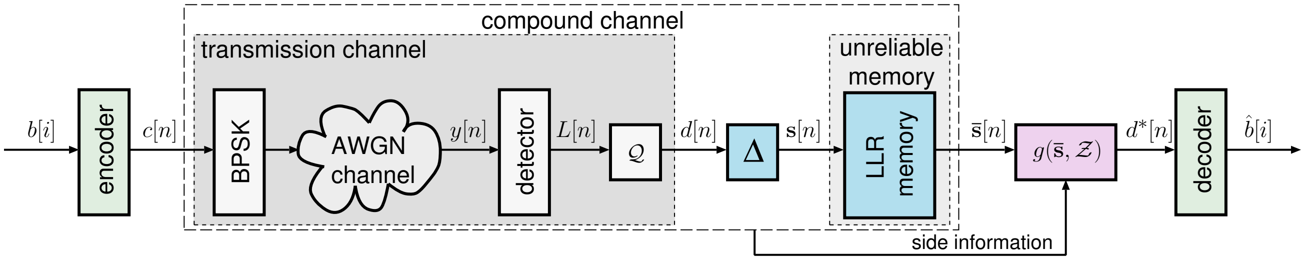


Fig. 2. Digital communication system employing BPSK transmission over an AWGN channel. The receiver consists of a soft-output detector, an LLR quantization block, a data mapping function Δ, and an unreliable LLR memory. At the memory output, corrected LLR values are computed from the observed label with the aid of the correction function g. The corrected LLRs are then passed to a soft-input decoder.

III. Application Example: Communication Systems

We next show an application example of the proposed statistical data correction framework to a coded digital communication receiver containing an unreliable memory.

A. System Model

We consider a communication system as shown in Fig. 2 and thoroughly introduced in [11], [13]. A sequence of information bits $b[i] \in \{0,1\}$ is encoded into a sequence of coded bits $c[n]$ using ECC. The coded bits are then mapped to BPSK symbols and transmitted over an AWGN channel, which is characterized by its instantaneous signal-to-noise ratio *SNR*.

1) Detection and quantization: At the receiver, a soft-output detector computes log-likelihood ratio (LLR) values for each coded bit $c[n]$ based on the received signal $y[n]$ as

$$L[n] = \log\left(\frac{\Pr(c[n]=0\,|\,y[n])}{\Pr(c[n]=1\,|\,y[n])}\right).$$

The computed LLR values $L[n]$ are then passed through a uniform[1] N-bit scalar quantizer $\mathcal{Q}$.

2) Unreliable LLR memory: The quantized LLR values $d[n]$ are mapped to labels using the mapping function Δ and stored in the unreliable LLR memory. In what follows, we assume the label cross-over probabilities $P_{\mathrm{C}}(\mathbf{s},\bar{\mathbf{s}})$ defined in (1), following the stuck-at channel model. In practice, the unreliable memory is typically used for data (de-)interleaving or as a large buffer that stores the LLR values of several data (re-)transmissions in modern wireless communication systems employing hybrid-ARQ (short for automatic repeat-request).

B. Statistical Data Correction for Unreliable LLR Memories

Following the framework from Sec. II, we statistically correct the output of the unreliable LLR memory (see Fig. 2).

1) MMSE-based correction function: As a baseline, we consider g_{MMSE} as defined in (3), which, when applied to the system in Fig. 2, minimizes the expected MSE between memory input LLR value $d[n]$ and the corrected output value $\bar{d}[n]$ conditioned on the observed memory output-label and the side information $\mathcal{Z}$. Basic arithmetic manipulations show that the resulting correction function is given by $g_{\text{MMSE}} = \mathbb{E}\{d[n]\,|\,\bar{\mathbf{s}},\mathcal{Z}\}$. We note that minimizing the bit error-rate (BER) rather than the MSE between LLR-memory input and output it is more relevant in communication systems. Hence, we next show a superior choice of the correction function that is tailored to the specifics of communication systems.

2) Application-specific correction function: Rather than correcting the LLR values directly, we propose to use a correction function that approximates the (coded) input bits $c[n]$ of the compound channel instead, i.e., we define

$$\mathcal{C}(\bar{c}\,|\,\bar{\mathbf{s}},\mathcal{Z}) \triangleq \mathbb{E}\big\{(c[n]-\bar{c})^2\,|\,\bar{\mathbf{s}},\mathcal{Z}\big\}.$$

This application-specific cost function leads to

$$\rho_1 = \Pr(c[n]=1\,|\,\bar{\mathbf{s}},\mathcal{Z}) = \arg\min_{\bar{c}} \mathcal{C}(\bar{c}\,|\,\bar{\mathbf{s}},\mathcal{Z}), \tag{4}$$

which corresponds to the probability that $c[n]=1$ given the output of the compound channel and side information. In uncoded systems, one can directly use the probability ρ_1 to extract binary (or hard) estimates of the uncoded bits. In *coded* systems, however, directly feeding such binary estimates to the (soft-input) decoder would result in sub-optimal performance. Hence, to compute LLR-values for the decoder, it is important to realize that the probability ρ_1 from (4) can be used to directly extract corrected LLR values as follows:

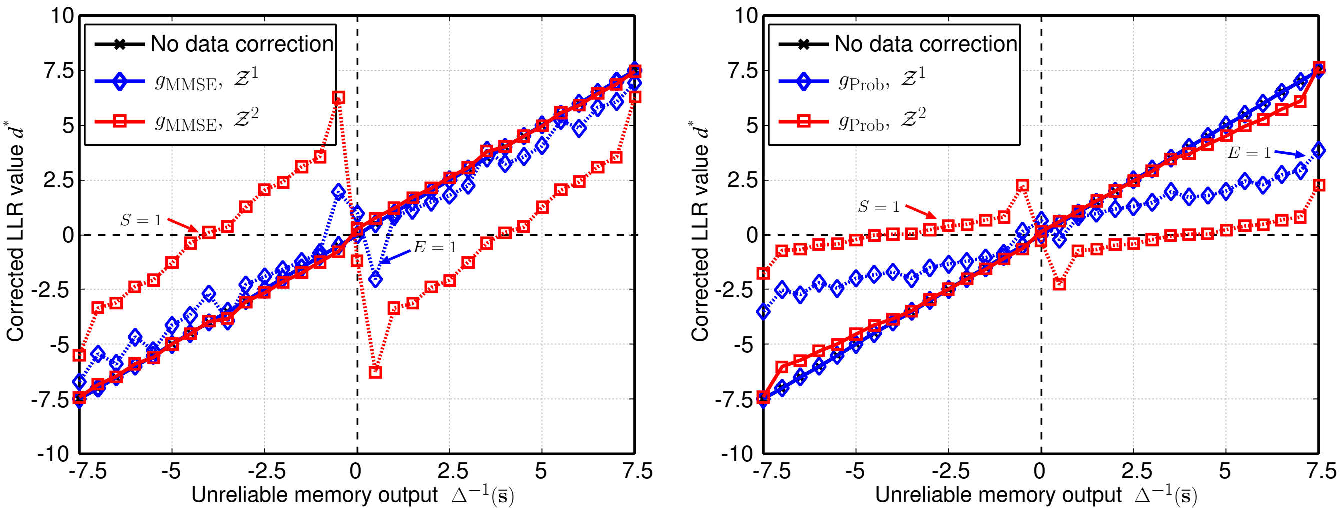
$$d^*[n] = g_{\text{Prob}}(\bar{\mathbf{s}}[n],\mathcal{Z}) \triangleq \log\left(\frac{1-\rho_1}{\rho_1}\right). \tag{5}$$

Interestingly, we find that this alternative LLR correction function coincides with the correction method proposed in [16], [17] to compensate mismatches in LLRs resulting from approximate data detection algorithms. We finally note that (4) and, hence, the alternative correction function (5), can be computed via Monte-Carlo simulations (averaged over noise realizations and unreliable memory effects).

3) Impact of side information: Fig. 3(a) and Fig. 3(b) show the two proposed correction functions g_{MMSE} and g_{Prob}; we assume an $N=5$ bit quantizer and 2's complement data representation.[2] We consider the following two sets of side

[1] Uniform quantization is the de-facto standard in digital integrated circuits for communication systems, e.g., [14], [15], but is not necessarily optimal.

[2] $N=5$ bit is a commonly used word width in channel decoders implemented in application-specific integrated circuits (see, e.g., [15]).



(a)

(b)

Fig. 3. Statistical data correction in a coded communication system assuming a bit-cell error probability $\varepsilon = 0.05$, $N = 5$ bit quantization, and 2's complement data representation. (a) Correction function g_{MMSE} for different sets of side information at $SNR = 1$ dB. (b) Correction function g_{Prob} for different sets of side information at $SNR = 1$ dB.

information $\mathcal{Z}^{1|2} = \{\mathcal{Z}_{\mathrm{S}}^{1|2}, \mathcal{Z}_{\mathrm{H}}^{1|2}\}$ with

$$\mathcal{Z}^1 = \{SNR, E\} \quad \text{and} \quad \mathcal{Z}^2 = \{SNR, S\}.$$

Here, $E \in \{0,1\}$ indicates for each LLR value stored in the unreliable memory whether it is corrupted by stuck-at faults ($E = 1$) or not ($E = 0$). The quantity $S \in \{0,1\}$ indicates for each LLR value whether its *sign-bit* is corrupted by a stuck-at fault ($S = 1$) or not ($S = 0$), ignoring information about the remaining bits. The specific choice of side information $\mathcal{Z}^2$ is motivated by the fact that the sign-bit of an LLR value is the most critical bit as it determines whether the corresponding code bit is more likely a 0 or a 1.

a) Observations for $\mathcal{Z}^1$: From Fig. 3(a) and Fig. 3(b), we see that for $\mathcal{Z}^1$ with $E = 0$, the corresponding LLR value stored in the unreliable memory is not corrupted by any stuck-at faults and, thus, no correction has to be performed. In contrast, for $E = 1$ we see that our statistical data correction function reduces the magnitude (i.e., the reliability) of the LLR values to account for the data corruption in the memory. We emphasize, that g_{Prob} leads to a stronger reliability reduction than g_{MMSE}. Interestingly, the two non-zero LLR values with smallest magnitude are corrected to values with opposite sign.

b) Observations for $\mathcal{Z}^2$: For $\mathcal{Z}^2$ with $S = 0$, almost no correction is performed for both g_{MMSE} and g_{Prob}, which indicates that the memory output-label is considered reliable as long as its sign-bit is not corrupted. For $S = 1$ on the other hand, the LLR values are strongly corrected in their magnitude and their sign. Comparing the two correction functions, we observe that g_{Prob} strongly reduces the reliability of all LLR values, while g_{MMSE} leads to a larger spread of LLR reliabilities. Intuitively, we expect a suitable statistical correction function to reduce the LLR reliabilities, which is clearly achieved for g_{Prob}.

C. Error-rate Performance Results

We next show that the proposed statistical data correction framework significantly improves the robustness of communication systems against unreliable memories. As an example, we assume that the encoder in Fig. 2 implements a convolutional code, which is decoded on receiver side using a soft-input Viterbi algorithm.[3] In such systems, the unreliable LLR memory typically corresponds to the (usually large) memory required for the de-interleaving of LLRs. It is worth mentioning that convolutional codes are still prominently used in many modern communication standards and hence, the robustness of the Viterbi decoder in presence of unreliable memories is of significant practical interest.

Fig. 4 shows the BER performance of the considered system, assuming a bit-cell error probability $\varepsilon = 0.05$ and $N = 5$ bit quantizer with 2's complement data representation.

Generally, it can be observed that both the choice of the correction function as well as the set of side information have a strong impact on the efficacy of statistical data correction. Specifically, we find that the application-specific correction function g_{Prob} yields better BER performance than the MSE-based function g_{MMSE}. This behavior highlights the importance of tailoring the cost function to the application at hand. Furthermore, we observe that the gains of data correction are more pronounced for $\mathcal{Z}^2$ than for $\mathcal{Z}^1$, demonstrating that information about corrupted sign-bits of the memory output-labels is extremely valuable to increase the robustness of the system against memory reliability issues. In this example, statistical data correction based on g_{Prob} and side information $\mathcal{Z}^2$ improves the SNR by more than 1 dB compared to a reference

[3] We consider the rate-1/2, 256-state convolutional code as specified for the 3G cellular communication standards of 3GPP (see [18]).

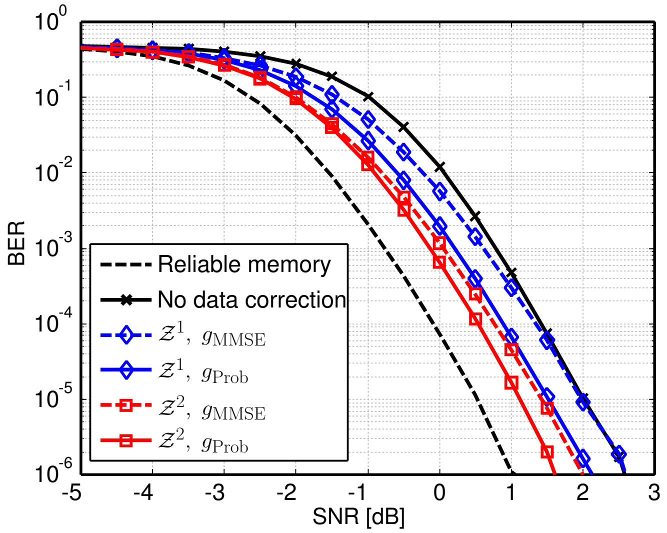


Fig. 4. BER performance of convolutional coding for different correction functions.

system having no statistical data correction.

IV. Conclusion

Digital signal processing techniques can improve the output quality of fault-tolerant DSP systems implemented with unreliable silicon. The statistical data correction approach proposed in this paper reduces the quality-degradation due to reliability issues of memories by exploiting side information on the unreliable system to minimize the expected cost. The proposed DSP technique is suitable for applications requiring large memories, where the complexity of data correction is low compared to the overhead of adding redundancy to the memory array for conventional error correction.

In addition to the provided application example in a coded communication system, the generality of the proposed statistical correction framework finds potential use in many other DSP systems for image-, video-, or audio-processing, for example. The design, analysis, and VLSI implementation of corresponding statistical correction methods is part of ongoing research.

V. Acknowledgments

This work was partially supported by the EU OPEN-FET SCoRPiO project (grant no. 323872), the EU Marie-Curie DARE project (grant no. 304186) and the ICYSoC RTD project (no. 20NA21 150939) funded by the Nano-tera.ch.

References

[1] J. Rabaey, *Low Power Design Essentials*. Springer, 2009.

[2] S. Bhunia and S. Mukhopadhyay, *Low-Power Variation-Tolerant Design in Nanometer Silicon*. Springer, 2010.

[3] S. Borkar, T. Karnik, and V. De, "Design and reliability challenges in nanometer technologies," in *Proc. 41st ACM/IEEE Design Automation Conf.*, Jul. 2004, p. 75.

[4] "Everything you wanted to know about SOC memory (white paper)," Tensilica, USA, 2014, available online May 1st, 2014 at http://www.tensilica.com/uploads/white_papers/SOC_Memory_Tensilica.pdf.

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

[6] A. Hussien, M. Khairy, A. Khajeh, K. Amiri, A. Eltawil, and K. F., "A combined channel and hardware noise resilient viterbi decoder," in *Proc. 44th IEEE Conf. Signals, Systems and Computers*, Nov. 2010, pp. 395–399.

[7] C.-H. Huang, Y. Li, and L. Dolecek, "Gallager b LDPC decoder with transient and permanent errors," *IEEE Trans. Communications*, vol. 61, no. 1, pp. 15–28, Jan. 2014.

[8] G. Karakonstantis, C. Roth, C. Benkeser, and A. Burg, "On the exploitation of the inherent error resilience of wireless systems under unreliable silicon," in *Proc. 49th ACM/IEEE Design Automation Conf.*, Jun. 2012, pp. 510–515.

[9] M. May, M. Alles, and N. N. Wehn, "A case study in reliability-aware design: a resilient LDPC code decoder," in *Proc. Design, Automation and Test in Europe*, Mar. 2008, pp. 456–461.

[10] H. Kaeslin, *Digital Integrated Circuit Design*. Cambridge University Press, 2008.

[11] C. Roth, C. Benkeser, C. Studer, G. Karakonstantis, and A. Burg, "Data mapping for unreliable memories," in *Proc. 50th Annual Allerton Conf. Commun., Control, and Computing*, Oct. 2012, pp. 679–685.

[12] R. Dekker, F. Beenker, and L. Thijssen, "A realistic fault model and test algorithms for static random access memories," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 9, no. 6, pp. 567–572, Jun. 1990.

[13] C. Novak, C. Studer, A. Burg, and G. Matz, "The effect of unreliable LLR storage on the performance of MIMO-BICM," in *Proc. 44th IEEE Conf. Signals, Systems and Computers*, Nov. 2010, pp. 736–740.

[14] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 8–17, Jan. 2011.

[15] C. Roth, P. Meinerzhagen, C. Studer, and A. Burg, "A 15.8 pJ/bit/iter quasi-cyclic LDPC decoder for IEEE 802.11n in 90 nm CMOS," in *Proc. IEEE Asian Solid State Circuits Conf.*, Nov. 2010, pp. 1–4.

[16] M. van Dijk, A. J. E. Janssen, and A. G. C. Koppelaar, "Correcting systematic mismatches in computed log-likelihood ratios," in *Europ. Trans. Telecomm.*, vol. 14, no. 3, Jul. 2003, pp. 227–244.

[17] C. Studer and H. Bölcskei, "Soft-input soft-output single tree-search sphere decoding," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 4827–4842, 2010.

[18] *Multiplexing and channel coding (TDD)*, Third Generation Partnership Project TS 25.222, Rev. 12.0.0, Dec. 2013.