Analysis of Min-Sum based Decoders Implemented on Noisy Hardware

Christiane L. Kameni Ngassa*†, Valentin Savin*, David Declercq†
*CEA-LETI, MINATEC Campus, 17 rue des Martyrs, 38054 Grenoble, France
†ETIS ENSEA/UCP/CNRS UMR 8051, 95014 Cergy-Pontoise Cedex, France
{christiane.kameningassa, valentin.savin}@cea.fr, declercq@ensea.fr

***Abstract*—Motivated by the problem of designing fault-tolerant memories built out from unreliable components, this paper investigates the performance of two noisy Min-Sum-based decoders on Binary Symmetric Channels. We analyze the performance of the noisy Min-Sum decoder in terms of useful regions and target bit-error rate threshold values, derived by using "noisy" density evolution equations. We also present the finite length performance of two Min-Sum based decoders, and point out the excellent performance of the noisy Self-Corrected Min-Sum decoder, which exhibits almost the same performance as the noiseless decoder.**

I. Introduction

One of the most critical challenges for the next-generation electronic circuit design is the nanoscale integration of chips built out of unreliable components. The ineluctable integration density increase and the imperative requirements of low-energy consumption, for energy preservation, can only be sustained through low-powered components, which will be inherently unreliable. The design of storage, interconnect, and processing elements will require completely new approaches, which are inconceivable without the use of powerful fault tolerant techniques.

This paper is motivated by the problem of designing fault-tolerant memories built out from unreliable components. In traditional models of memory systems with error correction coding, it is assumed that the operations of an error correction encoder and decoder are deterministic and that the noise exists only in the storage (memory) channel. This assumption is certainly appropriate in systems designed so that the reliability of logic gates used in the encoder and decoder is many orders of magnitude higher than reliability of memory cells. However, if digital logic in the encoder and decoder is built from faulty components, then the errors occurring at the gate level do affect operations performed in the encoder and decoder, and reduce the reliability of a whole system.

Except the pioneered works by Taylor and Kuznetsov on reliable memories [1], [2], later generalized in [3] to the case of powerful hard-decision decoders, this new paradigm of noisy decoders has merely not been addressed until recently in the coding literature. It is worth noting that Taylor also proved that no decoding scheme other than iterative Low-Density Parity-Check LDPC decoding [4] can achieve non-zero *storage capacity*. Roughly, the storage capacity is the ratio between the amount of data stored in the memory and the number of faulty components of the memory (consisting of both noisy registers and the correcting circuit). Recently, the asymptotic behavior of the noisy Gallager-A and Gallager-B LDPC decoders (defined over binary or non-binary alphabets) has been investigated [5], [6]. However, these papers deal with very simple error models, which emulate the noisy implementation of the decoder, by passing each of the exchanged messages through a binary (or non-binary, for non-binary LDPC codes) symmetric channel.

This paper deals with more powerful iterative decoders, namely the Min-Sum (MS) and Min-Sum-based decoders. While we consider transmission over a Binary Symmetric Channel (which models the storage channel), we also propose probabilistic error models for the arithmetic components of the MS decoder, in order to emulate its noisy implementation.

We further analyze the asymptotic performance of the noisy Min-Sum decoder, and provide useful regions and target-BER-threshold values [5] for a wide range of parameters of the proposed error models. Finally, we investigate the finite length performance of the noisy Min-Sum and Self-Corrected Min-Sum decoders.

II. LDPC Codes and Min-Sum Algorithm

A. LDPC Codes

LDPC codes [4] are linear block codes defined by sparse parity-check matrices. They can be advantageously represented by bipartite (Tanner) graphs [7] and decoded by message-passing iterative algorithms.

We consider an LDPC code defined by a bipartite (Tanner) graph $\mathcal{H}$, with N variable-nodes and M check-nodes [7]. Variable-nodes and check-node are denoted, respectively, by $n \in \{1, 2, ..., N\}$ and $m \in \{1, 2, ..., M\}$. $\mathcal{H}(n)$ and $\mathcal{H}(m)$ denote the set of neighbor nodes of the variable-node n and of the check-node m, respectively.

We further consider a codeword $(x_1, \ldots, x_N)$ that is sent over a Binary Symmetric Channel (BSC) with crossover probability p_0, and denote by $(y_1, \ldots, y_N)$ the received word. The following notation will be used throughout the paper, with respect to message passing decoders:

- γ_n is the log-likelihood ratio (LLR) value of x_n according to the received y_n value; it is also referred to as the *a priori information* of the decoder concerning the variable-node n;
- $\tilde{\gamma}_n$ is the *a posteriori information* (LLR value) of the decoder concerning the variable-node n;

- $\alpha_{m,n}$ is the variable-to-check message sent from variable-node n to check-node m;
- $\beta_{m,n}$ is the check-to-variable message sent from check-node m to variable-node n.

B. Min-Sum Decoding

The MS decoding works as follows. First variable to-check-messages are initialized according to the corresponding a priori LLR values. Then each decoding iteration consists of three steps, namely the check-node (CN) processing step, the a posteriori (AP) information update, and the variable-node (VN) processing step.

Initialization

- $\gamma_n = \log\left(\frac{\Pr(x_n=0|y_n)}{\Pr(x_n=1|y_n)}\right) = (1-2y_n)\log\left(\frac{1-p_0}{p_0}\right)$;
- $\alpha_{m,n} = \gamma_n$, $\forall\ n \in \{1,\ldots,N\}$ and $m \in \mathcal{H}(n)$;

Iterations

- CN-processing: $\forall\ m \in \{1,\ldots,M\}$ and $n \in \mathcal{H}(m)$

$$\beta_{m,n} = \left(\prod_{n'\in H(m)\setminus n} \text{sgn}(\alpha_{m,n'})\right) \min_{n'\in H(m)\setminus n}(|\alpha_{m,n'}|)$$

- AP-update: for $\forall\ n \in \{1,\ldots,N\}$

$$\tilde{\gamma}_n = \gamma_n + \sum_{m\in H(n)} \beta_{m,n}$$

- VN-processing: for $\forall\ n \in \{1,\ldots,N\}$ and $m \in \mathcal{H}(n)$

$$\alpha_{m,n} = \tilde{\gamma}_n - \beta_{m,n}$$

Finaly, note that each iterations also comprises a *hard decision* step (not shown in the algorithm), in which each transmitted bit is estimated according to the sign of the a posteriori LLR value, and the syndrome of the estimated word is computed. Decoding algorithm stops when whether the syndrome is zero or a maximum number of iterations is reached.

C. Fixed-Point Min-Sum decoder

We consider a fixed-point Min-Sum decoder, in which the exchanged messages ($\alpha_{m,n}$ and $\beta_{m,n}$) are quantized on q bits. The a posteriori information $\tilde{\gamma}_n$ is quantized on $\tilde{q}$ bits with $\tilde{q} > q$ (usually $\tilde{q} = q+1$, or $\tilde{q} = q+2$). We further denote:

- $\mathcal{M} = \{-Q,\ldots,-1,0,+1,\ldots,Q-1\}$, where $Q = 2^{q-1}$, the alphabet of the exchanged messages;
- $\widetilde{\mathcal{M}} = \{-\widetilde{Q},\ldots,-1,0,+1,\ldots,\widetilde{Q}-1\}$, where $Q = 2^{\tilde{q}-1}$, the alphabet of the a posteriori information;
- $\mathbf{s}_{\mathcal{M}} : \mathbb{Z} \to \mathcal{M}$, the q-bit saturation map defined by:

$$\mathbf{s}_{\mathcal{M}}(z) = \begin{cases} -Q, & \text{if } z < -Q \\ z, & \text{if } z \in \mathcal{M} \\ Q-1, & \text{if } z \geq +Q \end{cases}$$

- $\mathbf{s}_{\widetilde{\mathcal{M}}} : \mathbb{Z} \to \widetilde{\mathcal{M}}$, the $\tilde{q}$-bit saturation map defined in a similar manner as the previous one

The decoder is initialized by $\gamma_n = 1-2y_n \in \{-1,+1\} \subset \mathcal{M}$, $\forall n \in \{1,\ldots,N\}$ (hence the multiplicative factor $\log\left(\frac{1-p_0}{p_0}\right)$ is omitted from the definition of γ_n).

Finally, note that saturation maps $\mathbf{s}_{\mathcal{M}}$ and $\mathbf{s}_{\widetilde{\mathcal{M}}}$ define the fixed-point saturation of the exchanged messages and a posteriori LLR values, during the iterative decoding process.

III. Error Model for Noisy Arithmetic Components

In order to emulate the noisy implementation of the VN-processing step and the CN-processing step we propose error models for the two arithmetic components of the decoder, adders and the comparators, as well as an error model for the logic XOR gate needed to compute the sign of the check-to-variable node messages. We shall not consider here the case of errors that may occur due to the temporary storage of the exchanged messages in possibly noisy registers (in order not to depend on a specific decoder architecture). However, we note that the effect of noisy register on the exchanged messages can be integrated into the probabilistic models of the arithmetic components, in the sens that adding noise in those registers would modify the parameters of the probabilistic arithmetic components.

A. Model for the Noisy Adder

Adders are used in the decoder to compute the a posteriori information $\tilde{\gamma}_n$ (quantized on $\tilde{q}$ bits) and the variable-to-check node messages $\alpha_{m,n}$ (quantized on q bits). Given that $\tilde{q} > q$, we only consider $\tilde{q}$-bits adders (which also corresponds to practical implementations, since the value of $\alpha_{m,n}$ is derived from that of $\tilde{\gamma}_n$).

The noisy (probabilistic) adder is defined by the following parameters:

- p_a is the probability of the adder's output being in error;
- q_e is the number of bits of the adder output, starting from the least significant bit (LSB), that can be affected by errors. Hence, $q_e \leq \tilde{q}$, and it is referred as the *depth* of the probabilistic model.

The probabilistic model is further specified as follows. Let d be an integer, referred to as error-pattern, for which the positions of 1's (within its binary representation) indicate the locations of the erroneous bits in the adder's output. The output of the adder is error-free if and only if $d = 0$. The error-pattern d belongs to an alphabet $\mathcal{D}$, which is defined as follows.

- When $q_e < \tilde{q}$ the error-pattern d is an unsigned integer represented on q_e-bits, hence $\mathcal{D} = \{0,1,\ldots,Q_e\}$, with $Q_e = 2^{q_e}-1$.
- When $q_e = \tilde{q}$ the sign of the output may also be in error, hence d is a signed integer represented on $q_e = \tilde{q}$ bits and $\mathcal{D} = \widetilde{\mathcal{M}} = \{-\widetilde{Q},\ldots,-1,0,1,\ldots,\widetilde{Q}-1\}$

The output of the noisy adder is obtained by performing a bit-wise xor operation between the output of the noiseless adder and d. Furthermore, we consider that all non-zero error patterns have equal probability. Since $p_a = \sum_{d\neq 0}\Pr(d)$, it follows that for any $d \neq 0$, $\Pr(d) = \frac{p_a}{|\mathcal{D}|-1}$, while $p(0) = 1-p_a$. Thus, the probabilistic adder is defined by:

$$\mathbf{a}_{\text{pr}}(x,y) = \mathbf{s}_{\widetilde{\mathcal{M}}}(x+y) \wedge d,\ \forall (x,y) \in \widetilde{\mathcal{M}} \times \widetilde{\mathcal{M}},$$

where d is drawn randomly from $\mathcal{D}$ according to the above probabilities, and $\wedge$ symbol denotes the bitwise XOR operation.

Table I gives an example of an erroneous adder output, for $\tilde{q} = 5$ and $q_e = 4$. In this case, since $q_e < \tilde{q}$, the sign bit of the output cannot be affected by errors. However, it worth

Table I
EXAMPLE OF AN ERROR INJECTION IN THE OUTPUT OF THE NOISY ADDER

	integer	2's complement bit representation				
exact output	-11	1	0	1	0	1
error pattern	6		0	1	1	0
erroneous output	-13	1	0	0	1	1
bit position		$\tilde{q}=5$	$q_e=4$	3	2	1

noting that in case of an "addition chain", as for instance $(x+y)+z$, the sign of the noiseless fixed-point output, given by $\mathbf{s}_{\widetilde{\mathcal{M}}}(\mathbf{s}_{\widetilde{\mathcal{M}}}(x+y)+z)$, may be different from the sign of the noisy output, given by $\mathbf{a}_{\text{pr}}(\mathbf{a}_{\text{pr}}(x,y),z)$.

Finally, we note that the depth parameter q_e is used to investigate the decoder behavior when errors may occur on increasing significantly bits. It can be used as a *guideline* for the hardware architecture of adders made from noisy components (logic gates). In order to ensure a target performance of the decoder, *adders should be specifically designed, such that to be compliant with the maximum admissible q_e value* (e.g. by using classical fault-tolerant solutions for the MSBs, as modular redundancy).

B. Model for the Noisy XOR Logic Gate

The probabilistic error model of the noisy XOR is much more simple, and it is specified only by a single error probability parameter, denoted by p_x.
For any $a, b \in \{0,1\}$ the probabilistic XOR of a and b, denoted by $\mathbf{x}_{\text{pr}}(a,b)$ is defined by:

$$\mathbf{x}_{\text{pr}}(a,b) = \begin{cases} \text{XOR}(a,b), & \text{with probability} 1-p_x \\ \overline{\text{XOR}(a,b)}, & \text{with probability} p_x \end{cases}$$

C. Model for Noisy Comparator

Similarly to the XOR operator, the probabilistic error model of the noisy comparator is simple, and it is specified only by a single error probability parameter, denoted by p_c. We consider q-bit comparators that are used at the CN-processing step. For any $x, y \in \mathcal{M}$ the probabilistic **minimum** of x and y, denoted by $\mathbf{m}_{\text{pr}}(x,y)$ is defined by:

$$\mathbf{m}_{\text{pr}}(x,y) = \begin{cases} \min(x,y), & \text{with probability} 1-p_c \\ \max(x,y), & \text{with probability} p_c \end{cases}$$

IV. NOISY FIXED-POINT MIN-SUM DECODER

A. Noisy Min-Sum decoding

Using the previous notation, the noisy fixed-point Min-Sum decoder can be described as follows (probabilistic components appear in red):

Initialization

- $\gamma_n = 1-2y_n \in \{-1,+1\} \subset \mathcal{M}$, $\forall\, n \in \{1,\dots,N\}$;
- $\alpha_{m,n}^{(0)} = \gamma_n$, $\forall\, n \in \{1,\dots,N\}$ and $m \in \mathcal{H}(n)$;

Iterations: for $\ell \geq 1$

- CN-processing: $\forall\, m \in \{1,\dots,M\}$ and $n \in \mathcal{H}(m)$

$$\text{sgn}\left(\beta_{m,n}^{(\ell)}\right) = \mathbf{x}_{\text{pr}}\left(\mathbf{x}_{\text{pr}}\left(\text{sgn}(\alpha_{m,n_1}^{(\ell-1)}), \text{sgn}(\alpha_{m,n_2}^{(\ell-1)})\right)\cdots, \text{sgn}(\alpha_{m,n_{d_c-1}}^{(\ell-1)})\right)$$

$$|\beta_{m,n}^{(\ell)}| = \mathbf{m}_{\text{pr}}\left(\mathbf{m}_{\text{pr}}\left(|\alpha_{m,n_1}^{(\ell-1)}|, |\alpha_{m,n_2}^{(\ell-1)}|\right)\cdots, |\alpha_{m,n_{d_c-1}}^{(\ell-1)}|\right)$$

- AP-update: for $\forall\, n \in \{1,\dots,N\}$

$$\tilde{\gamma}_n^{(\ell)} = \mathbf{a}_{\text{pr}}\left(\mathbf{a}_{\text{pr}}\left(\gamma_n, \beta_{m_1,n}^{(\ell)}\right)\cdots, \beta_{m_{d_v},n}^{(\ell)}\right)$$

- VN-processing: for $\forall\, n \in \{1,\dots,N\}$ and $m \in \mathcal{H}(n)$

$$\alpha_{m,n}^{(\ell)} = \mathbf{a}_{\text{pr}}\left(\tilde{\gamma}_n^{(\ell)}, -\beta_{m,n}^{(\ell)}\right)$$

In the CN-processing step, $\text{sgn}(x)$ denote the sign bit, in $\{0,1\}$, of the message x. Finally, we note that the hard decision and the syndrome computation step, mentioned at the end of Section II-B, are assumed to be *noiseless*.

B. Density Evolution for the noisy Min-Sum decoding

The objective of the density evolution technique is to recursively compute the probability mass functions of exchanged messages, through the iterative decoding process. This is done under the independence assumption of exchanged messages, holding in the asymptotic limit of the code length, in which case the decoding performance converges to the cycle-free case. Concentration and convergence properties for the asymptotic performance of noisy message-passing decoders have been proved in [5].

We derived density evolution equations for the noisy fixed-point MS decoding, not included here due to space limitations. However, density evolution equations for the noisy fixed-point MS decoding with parameters $p_a > 0$, $p_c > 0$, but $p_x = 0$ (*i.e.* XOR gates needed to compute the sign of the check-to-variable node messages are assumed to be noiseless) have been provided in [8]. These equations can be ready generalized to the case $p_x > 0$, and they will be included in an extended version of this paper.

1) Useful region: Since the input error probability can actually be increased when the decoder is run on noisy hardware, the first step is to evaluate the channel and hardware parameters yielding a final probability of error (after decoding) less than the channel error probability. Using the notation from [8], we denote by $P_e^{(\ell)}$ the error probability of the decoder at iteration ℓ. Following [5], decoder is said to be *useful* if the $\lim_{\ell\to\infty} P_e^{(\ell)}$ exists and:

$$\lim_{\ell\to\infty} P_e^{(\ell)} \leq p_0$$

where $P_e^{(\ell)}$ is the error probability at iteration ℓ, and p_0 is the crossover probability of the BSC. The ensemble of the parameters that satisfy this condition constitutes the *useful region* of the decoder.

2) Threshold region: For noiseless-decoders traditionally considered in classical coding theory, the decoding threshold is defined as the supremum p_0, such that the error probability converges to zero as the number of decoding iterations goes to infinity. However, for noisy decoders this error probability does not converge to zero, and an alternative definition of the decoding threshold has been introduced in [5]. Accordingly, for a target bit-error rate η, the η-threshold is defined by:

$$p_0^*(\eta) = \sup\{p_0 : \lim_{\ell\to\infty} P_e^{\ell} < \eta\}$$

where p_0 is the crossover probability of the BSC channel.

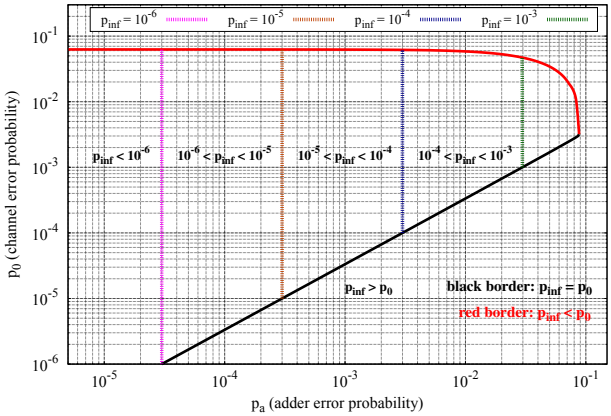

Figure 1. Useful region for $p_a > 0$ with $q_e = 4$ ($p_x = p_c = 0$)

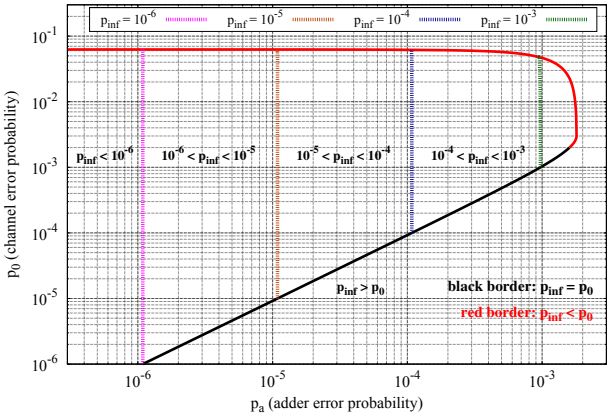

Figure 2. η-regions for $p_a > 0$ with $q_e = 5$ ($p_x = p_c = 0$)

V. Asymptotic analysis of the noisy Min-sum decoder

Density evolution equations for noisy fixed-point MS decoder were run on MATLAB for a regular $(3, 6)$ LDPC code over the Binary Symmetric Channel, with exchanged messages quantized on $q = 4$ bits, and the a posteriori information quantized on $\tilde{q} = 5$ bits.

A. Study of the impact of the noisy adder

In order to evaluate the influence of the noisy adder on the performance of the decoder, the useful region and the η-threshold regions have been computed, assuming that only the adders within the VN-processing step are noisy ($p_a > 0$), while the CN-processing step is noiseless ($p_x = p_c = 0$). This regions are represented in Fig. 1 and Fig. 2, for error depth parameters $q_e = 4$ and $q_e = 5$, respectively.

The border of the useful region is divided in two parts: the bottom border plotted in black (a straight line), and the right and top borders plotted in red. Although one would expect that $p_\infty = p_0$ on the border of the useful region, this equality only holds for the black (bottom) border. On the red border, one has $p_\infty < p_0$. The reason why the useful region does not extend beyond the red border is that for points located close to, but on the other side of the border, the sequence $(P_e^{(\ell)})_{\ell>0}$ becomes periodic, and hence p_∞ does not exist.

Another surprising result is that the decoder's error probability p_∞ seems to be independent of p_0, as long as p_0 is below the top boundary of the useful region. This is indicated by the vertical lines that divide the useful region on η-threshold regions, for $\eta = 10^{-n}$, $3 \leq n \leq 6$. On each line p_∞ is equal to the corresponding η-value. It should also be noted, that by increasing the error depth parameter from $q_e = 4$ (Fig. 1) to $q_e = 5$ (Fig. 2), the maximum p_a value for which $p_\infty \leq \eta$ is decreased by a a factor of ≈ 30. Moreover, for $q_e = 5$, the decoder error probability p_∞ seems to be very close (but slightly higher) than the p_a value.

B. Study of the impact of the noisy XOR gate

The useful region and the η-threshold regions of the decoder, assuming that only the XOR-gate used within the CN-processing step is noisy, are plotted in Fig. 3. Similar to the noisy-adder case, the border of the useful region is divided in two parts: the black border corresponding to points for which $p_\infty = p_0$ and the red border separating the useful region from the "non-convergence" region. It is worth noting that usually the value of p_x is much less than the value of p_a (given the high number of elementary gates contained in the adder). It practice it can be reasonably assumed that $p_x < 10^{-4}$. Moreover, since the XOR-gates used to compute the $\beta_{m,n}$ signs represent only a very small part of the decoder, this part of the circuit may be made reliable by using classical fault-tolerant methods, with a limited impact on the overall decoder design.

C. Study of the impact of the noisy comparator

This section investigates the case when comparators used within the CN-processing step are noisy ($p_c > 0$), but $p_a = p_x = 0$. Contrary to the previous cases, this case exhibits a threshold phenomenon: for a given $p_c > 0$, the exists a p_0-threshold value, denoted by p_{th}, such that for any $p_0 < p_{\text{th}}$, one has $p_\infty = 0$. Tis threshold value is plotted as a function of p_c in Fig. 4. Although such a threshold phenomenon might seem surprising for a noisy decoder, it can be easily explained. As it can be seen from Fig. 4, if $p_0 = 0.01$ then for any $p_c > 0$ one has $p_\infty = 0$. The idea behind is that in this case the crossover probability of the channel is small enough, so that in the CN-processing step only the sign of check-to-variable messages is important, but not their amplitudes. In other words a decoder that only computes the signs of $\beta_{m,n}$ messages and randomly chooses their amplitudes, would be able to perfectly decode the received word (in the asymptotic length of the codeword).

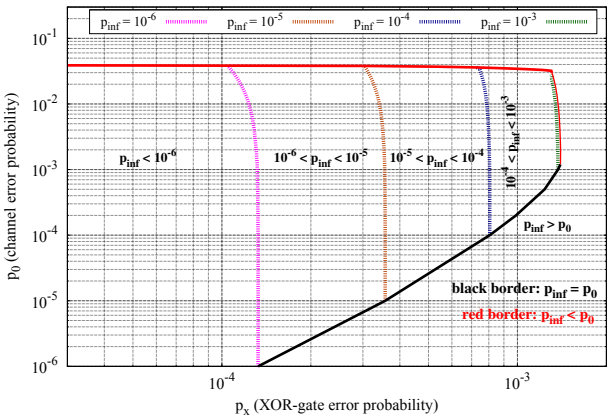

Figure 3. η-regions for $p_x > 0$ ($p_a = p_c = 0$)

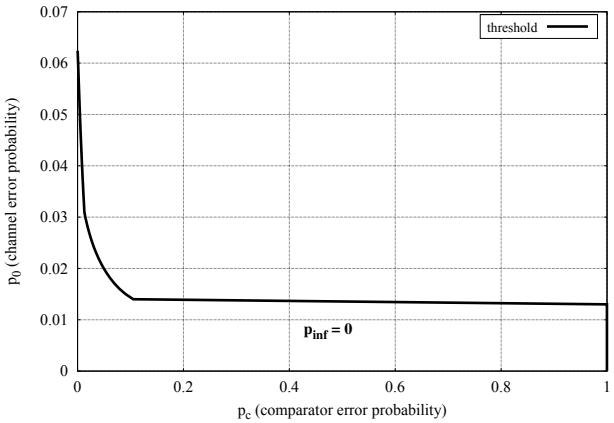

Figure 4. η-regions for $p_c > 0$ ($p_a = p_x = 0$)

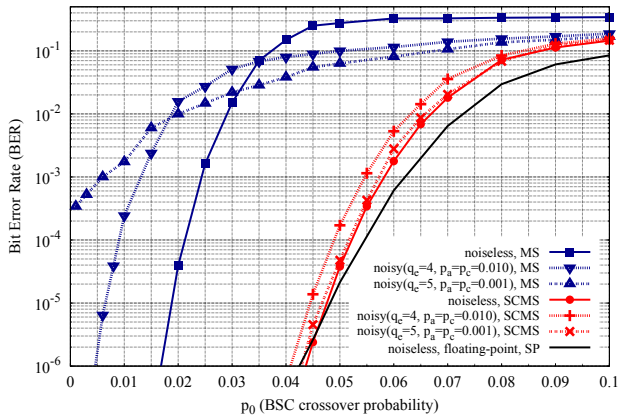

Figure 5. BER performance of noisy vs. noiseless MS and SCMS decoders

VI. Finite Length Performance of Min-Sum Based Decoders

In this section we evaluate the finite-length performance of two noisy Min-Sum based decoders: the Min-Sum (MS) and the Self-Corrected Min-Sum (SCMS) [9]. The objective of the SCMS is to determine if a correction circuit "plugged into" the noisy MS decoder can improve the robustness of the decoder to hardware noise.

The noisy SCMS decoder performs the same computations as the noisy MS (Section IV), except that the VN processing step further includes a *correction step*, as follows:

- VN-processing: for $\forall\ n \in \{1, \ldots, N\}$ and $m \in \mathcal{H}(n)$

$$\alpha_{m,n}^{(\ell)} = \mathbf{a}_{\text{pr}}\left(\tilde{\gamma}_n^{(\ell)}, -\beta_{m,n}^{(\ell)}\right);$$
if $\text{sgn}\left(\alpha_{m,n}^{(\ell)}\right) \neq \text{sgn}\left(\alpha_{m,n}^{(\ell-1)}\right)$ and $\alpha_{m,n}^{(\ell-1)} \neq 0$
$\quad \alpha_{m,n}^{(\ell)} = 0\,;$
end

The body enclosed between the **if** condition and the matching **end** is referred to as correction step. Note that the correction step is implemented with reliable circuitry, which can be reasonably assumed, since the required circuitry is very simple (see also the discussion concerning the sign of check-to-variable messages in Section V-B).

Fig. 5 shows the Bit Error Rate (BER) performance of the two decoders for the $(N = 504, K = 252)$ and $(d_v = 3, d_c = 6)$-regular LDPC code available online at [10]. The solid curves with filled markers correspond to the performance of the fixed-point ($q = 4, \tilde{q} = 5$) noiseless MS and SCMS decoders. For comparison purposes, the performance of the floating-point noiseless Belief-Propagation decoder is also shown (solid curve, no markers). The four dashed curves correspond to the performance of the noisy MS and SCMS decoders, with parameters $(q_e = 4, p_a = p_c = 0.01, p_x = 0)$ and $(q_e = 5, p_a = p_c = 0.001, p_x = 0)$.

While it can be seen that hardware noise alter the performance of the MS decoder, the noisy SCMS decoder exhibits very good performance, very close to that of the noiseless decoder. Therefore, one can think of the *self-correction* circuit as a *noiseless patch* applied to the noisy MS decoder, in order to improve its robustness to hardware noise.

VII. Conclusion

In this paper we investigated the performance of noisy MS-based decoders over the BSC. We proposed probabilistic error models for the arithmetic components of the MS-based decoders, in order to emulate their implementation on noisy hardware. We analyzed the performance of the MS decoder in terms of useful regions and target-BER thresholds, derived by using "noisy" density evolution. We further evaluated the finite length performance of the MS and SCMS decoders, for various parameters of the hardware noise models. We highlighted the excellent performance of the SCMS decoder, which is due to its intrinsic ability to detect and discard unreliable messages during the iterative decoding process. Finally, the results of our work may serve as guidelines for the design of noisy arithmetic components for Min-Sum-based decoders on BSC.

Acknowledgment

This work was supported by the Seventh Framework Programme of the European Union, under Grant Agreement number 309129 (i-RISC project).

References

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

[2] A. V. Kuznetsov, "Information storage in a memory assembled from unreliable components," *Problemy Peredachi Informatsii*, vol. 9, no. 3, pp. 100–114, 1973.

[3] B. Vasic and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 54, no. 11, pp. 2438–2446, 2007.

[4] R. G. Gallager, "Low density parity check codes," MIT Press, Cambridge, 1963, research Monograph series.

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

[6] S. Yazdi, H. Cho, and L. Dolecek, "Gallager b decoder on noisy hardware," *IEEE Trans. on Comm.*, 2013.

[7] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Inf. Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[8] C. L. Kameni Ngassa, V. Savin, and D. Declercq, "Min-sum-based decoders running on noisy hardware," in *proc. of IEEE Global Communications Conference (GLOBECOM)*, 2013.

[9] V. Savin, "Self-corrected min-sum decoding of LDPC codes," in *Proc. of IEEE Int. Symp. on Information Theory (ISIT)*, 2008, pp. 146–150.

[10] D. J. C. MacKay. Encyclopedia of sparse graph codes. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html