Faulty Stochastic LDPC Decoders Over the Binary Symmetric Channel

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*—The analysis of error correction decoders running on faulty hardware has attracted an increased interest in recent years, due to the inherent unreliability of emerging nanodevices. In this paper we investigate the performance of the stochastic decoder running on faulty hardware. To this end, we first introduce two error models to describe the noisy components of the decoder. We then provide a finite-length statistical analysis for each error model and, based on the obtained performance, we conclude that stochastic decoders have an inherent fault tolerant capability.**

I. Introduction

Over the past few years, there has been an increasing interest on error correcting decoders built out of unreliable components. The motivation is are twofold. First, it is widely accepted that future generation of electronic circuit will be inherently unreliable, due to increase in density integration and lower power supply voltage scaling. Second, error correcting decoders play a crucial role both in reliable transmission of information and in the design of reliable storage systems.

The asymptotic analysis of Gallager A and Gallager B LDPC decoders running on faulty hardware has been carried out in [1] and [2]. [3] studied the asymptotic and finite-length performance of faulty Min-Sum based decoders. The study has shown that the reliability of the most significant bit (sign bit) of exchanged messages has a critical impact on the performance of Min-Sum based decoders, while tolerates more errors can be tolerated on less significant bits. The sign bit should then be specifically protected to ensure the robustness of noisy Min-Sum based decoders. However, it can be difficult to properly achieve only the reliability of the most significant bit since errors occurring on less significant bits propagate to the next level and may affect the sign bit. On the contrary, in stochastic computation, each bit of a given stochastic stream has exactly the same significance than the others. Probabilities are represented by the ratio of 1's contained in the sequence of bits and the order of bits in that sequence does not affect the value of the encoded probability. Besides, if a small number of bits are in error, the resulting value of the probability will be close to the correct value, and errors are expected to have a limited impact on the decoder performance. Consequently, stochastic decoders may have an inherent fault tolerance capability. Moreover, [4] demonstrated that timing errors have a limited impact on the stochastic decoder performance. It is then important to study the behavior of the stochastic decoder in a more general hardware error scenario, in order to determine the amount of errors that can be tolerated, and which parts of the decoder can be built out of unreliable components.

In this paper we focus on stochastic decoder using edge-memories. We propose two error models and for each model we compare statistical results of noisy finite-length decoders with their corresponding noiseless versions. We show that in some cases the additional noise from the hardware can be used to lower the error floor of the stochastic decoder.

The remainder of the paper is organized as follows. Section II introduces LDPC codes and stochastic decoding. The error models of the faulty stochastic decoders are presented in Section III. The statistical analysis of finite-length faulty decoders are provided in Section IV. Section V concludes the paper.

II. LDPC Codes and Stochastic Decoding

A. LDPC Codes

LDPC codes [5] are linear block codes defined by sparse parity-check matrices. They can be advantageously represented by bipartite (Tanner) graphs [6] and decoded by message-passing iterative algorithms. The Belief-Propagation (BP) decoding – also referred to as Sum-Product (SP) – exchanges probability beliefs as messages between the nodes of the Tanner graph. However, its main drawbacks are its complexity ans its numerical instability.

In order to reduce the computation complexity of the exchanged messages, the Stochastic decoder converts the probabilistic belief in streams of stochastic bits representing the encoded probability. The resulting circuitry of the decoder has a small number of logic gates, thus a reduced hardware-complexity compared to the BP decoder.

B. Stochastic Decoding Principle

The stochastic decoder is the stochastic implementation of the Belief Propagation algorithm. Instead of propagating probability beliefs between the nodes of the factor graph, the stochastic decoder convert these probabilities in Bernoulli sequences of bits, each bit being equal to 1 with the probability to be encoded in the stochastic streams [7]. This representation is not unique since different sequences of a given length can contain the same number of 1's and then represent the same probability. The value of the encoded probability is the ratio of bits equal to 1 contained the stochastic sequence. This

stochastic computing allows the use of simple logic gates to perform complex arithmetic operations on probabilities such as multiplications and divisions.

The main advantages of the stochastic decoder are the following.

- Only one bit is exchanged between computational elements, reducing the number of wire in the circuitry and allowing high clock frequency.
- The computation hardware area is significantly reduced.
- Computation accuracy can be trade for energy saving without hardware modification [8].

C. Notations

We consider an LDPC code defined by a bipartite (Tanner) graph $\mathcal{H}$, with N variable-nodes and M check-nodes [6]. 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-input memoryless noisy channel, and denote by $(y_1, \ldots, y_N)$ the received word. The following notation will be used throughout the paper:

- 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 LLR value* of the decoder concerning the variable-node n;
- $p_n = \Pr(x_n = 1|y_n) = \dfrac{1}{1+\exp(\gamma_n)}$;
- $\mathbf{\Pi}$ is a random bit generator used to generate stochastic streams;
- $\Gamma_n \leftarrow \mathbf{\Pi}(p_n)$ a random generated bit in $\{0, 1\}$, with probability of being 1 equal to p_n;
- $\alpha_{m,n}$ is the variable-to-check message (1 bit from the stochastic stream) sent from variable-node n to check-node m;
- $\beta_{m,n}$ is the check-to-variable message (1 bit from the stochastic stream) sent from check-node m to variable-node n.
- θ_n is a counter that counts the number of $\alpha_{m,n}$ messages equal to 1;

D. Stochastic Decoding Algorithm

As mentioned above, stochastic decoder is the stochastic computation of Belief-Propagation algorithm. At each decoding iteration, the BP decoder exchanges values of probability (whose binary representation contains several bits) between variable and check nodes. Stochastic decoder converts theses probability in sequences of bits and exchanges only one bit at each decoding iteration. Therefore iterations in stochastic decoder does not exactly corresponds to the message passing iterations and are referred to as *decoding cycles* instead of decoding iterations.

The Belief-Propagation algorithm in probability domain can be found in [8]. Stochastic decoder follows the same steps but performs multiplications by using AND gates and divisions using JK flip-flops.

The stochastic decoding algorithm can then be written as follows.

Initialization

- $\gamma_n = \log\left(\dfrac{\Pr(x_n = 0|y_n)}{\Pr(x_n = 1|y_n)}\right)$, $\forall\ n \in \{1, \ldots, N\}$;
- $p_n = \Pr(x_n = 1|y_n) = \dfrac{1}{1+\exp(\gamma_n)}$, $\forall\ n \in \{1, \ldots, N\}$;
- $\Gamma_n \leftarrow \mathbf{\Pi}(p_n)\ \forall\ n \in \{1, \ldots, N\}$
- $\alpha^{(0)}_{m,n} = \Gamma_n$, $\forall\ n \in \{1, \ldots, N\}$ and $m \in \mathcal{H}(n)$;
- $\theta^{(0)}_n = 0$, $\forall\ n \in \{1, \ldots, N\}$;

Iteration loop: at cycle ℓ

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

$$\beta^{(\ell)}_{m,n} = \underset{n' \in H(m)\backslash n}{\mathbf{XOR}} \left(\alpha^{(\ell-1)}_{m,n'}\right)$$

- Channel random bit update: $\forall\ n \in \{1, \ldots, N\}$,

$$\Gamma_n \leftarrow \mathbf{\Pi}(p_n)$$

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

$$\alpha^{(\ell)}_{m,n} = \begin{cases} \Gamma_n, & \text{if } \beta^{(\ell)}_{m',n} = \Gamma_n\ \forall m' \in H(n)\backslash m \\ \alpha^{(\ell-1)}_{m,n}, & \text{otherwise} \end{cases}$$

  In other words, if the input messages (including $Gamma_n$) agree, the output message is equal to their common value. Otherwise the output message is not modified with respect to the previous iteration. Note that when the inputs disagree the node is said to be in a *hold* state. When they agree the decoder is in a *nonhold* or *regular* state.

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

$$\theta^{(\ell)}_n = \theta^{(\ell-1)}_n + \sum_{n' \in H(m)} \left(2\alpha^{(\ell)}_{m,n'} - 1\right)$$

  Meaning that the counter is incremented if $\alpha^{(\ell)}_{m,n'} = 1$ and decremented if $\alpha^{(\ell)}_{m,n'} = 0$

- Hard decision: for $\forall\ n \in \{1, \ldots, N\}$

$$\hat{x}_n = \begin{cases} 1, & \text{if } \tilde{\gamma}_n > 0 \\ 0, & \text{otherwise} \end{cases}$$

- Syndrome check: if $\underline{\hat{x}}$ is a codeword then exit iteration loop

E. Improving the Stochastic Decoder Performance

The problem with the above stochastic decoder is that it assumes that stochastic streams are independent Bernoulli sequences. However this is no longer the case when there are cycles in the factor graph. Besides, a low level of switching activity in the stochastic decoder can also cause groups of nodes to lock into fixed states which prevents proper decoding and leads to poor performance. In order to increase the switching activity in the circuitry, the noise-dependent scaling method have been introduced in [7] . Moreover, a couple of rerandomization methods have been proposed in the literature to reduce the correlation between stochastic bits. Edge-Memories will be used as rerandomization units in this works because they provide good performance and are commonly used in the literature.

1) Noise-Dependent Scaling: Noise-dependent Scaling consist of scaling the channel reliability (LLR) by a factor proportional to the channel noise power in order to ensure similar switching activity for different channel noise level. For the Binary symmetric Channel (BSC), this translates into making the LLR values independent from the channel error probability ϵ. This can be achieved by replacing ϵ by a constant μ in the expression of the a priori information [9]:

$$\gamma_n = (-1)^{y_n} \log\left(\frac{1-\epsilon}{\epsilon}\right) \Longrightarrow \gamma_n = (-1)^{y_n} \log\left(\frac{1-\mu}{\mu}\right)$$

2) Edge-Memories: Edge Memories (EMs) are memory-based re-randomization units used to decorrelate bits in stochastic streams. Each EM consists of a M-bit shift register and is assigned to one edge of the decoder. EMs are initialized according to the channel error probability (that is, each bit of an EM adjacent to variable node n is generated by $\mathbf{\Pi}(p_n)$).

The variable-node processing is modified as follows. At cycle ℓ, for $\forall\ n \in \{1, \ldots, N\}$ and $m \in \mathcal{H}(n)$
if $\forall m' \in H(n)\backslash m$, $\beta^{(\ell)}_{m',n} = \Gamma_n$

$$\begin{array}{rcl} \alpha^{(\ell)}_{m,n} & = & \Gamma_n \\ \Gamma_n & \rightarrow & \mathbf{EM} \end{array}$$

otherwise,

$$\alpha^{(\ell)}_{m,n} = \mathbf{EM}(i)$$

where $\Gamma_n \rightarrow \mathbf{EM}$ means that the bit Γ_n is stored in the Edge-Memory and i is a random position in the Edge-Memory. When a new Γ_n bit is stored in the EM, bits in the EM are shifted first from left to the right, and then Γ_n is stored in the left-most position.

III. Faulty Stochastic Decoders

In this section we propose two error models for the faulty stochastic decoders. In the first model, only the Edge-Memories are unreliable. In the second model the stochastic stream generators, the variable-node units and the check-node units are all considered to be noisy.

A. Stochastic Decoders with Noisy Edge-Memories

The main drawback of Edge-Memories as rerandomization method is their number. There are as many Edge-Memories in the decoder as the edges of the factor graph. The energy consumption of the stochastic decoder can be decreased by reducing the energy consumption of the Edge-Memories. The latter can be achieved by either voltage scaling or by using low-quality components for Edge-Memories. But this will result on a degradation of the reliability of EMs. The objective of this model is then to analyze the impact of the faulty Edge-Memories on the decoder performance and to further determine which level of noise in the EMs can the stochastic decoder tolerate.

Errors can occur in EMs when a bit is written in the memory and when a bit is read from the memory. Denote w the probability that an error occurs during the writing and r the probability that an error occurs during the reading. The output of the EM will be in error if the selected bit was either written with error or read with error. Note that error during the writing and during the reading at the same address in the memory will compensate each other. Therefore the error probability of the Edge-Memory output is $p_{em} = w(1-r) + (1-w)r$.

1) Noisy Edge-Memory Model: Denote $\mathbf{EM}(i)$ and $\mathbf{EM}_{\text{pr}}(i)$ the bit read at address i from the noiseless EM and from the noisy EM respectively.

$$\mathbf{EM}_{\text{pr}}(i) = \begin{cases} \mathbf{EM}(i), & \text{with probability } 1-p_{em} \\ \overline{\mathbf{EM}(i)}, & \text{with probability } p_{em} \end{cases}$$

2) Variable Node processing with noisy EM: The VN-processing of noiseless stochastic decoder is modified as follows.

At cycle ℓ for $\forall\ n \in \{1, \ldots, N\}$ and $m \in \mathcal{H}(n)$
if $\forall m' \in H(n)\backslash m$, $\beta^{(\ell)}_{m',n} = \Gamma_n$

$$\begin{array}{rcl} \alpha^{(\ell)}_{m,n} & = & \Gamma_n \\ \Gamma_n & \rightarrow & \mathbf{EM} \end{array}$$

otherwise,

$$\alpha^{(\ell)}_{m,n} = \mathbf{EM}_{\text{pr}}(i)$$

B. Error model with noisy logic gates

In this paragraph we propose a more general error model for stochastic decoders. We suppose that the entire stochastic decoder is made of faulty components. Except the Counter update, Hard decision and Syndrome check steps. Note that syndrome check is not a compulsory step since the maximum number of cycles ensures the termination of iterations loops.

In order encompass all possible errors into a minimal number of error probability parameters, we inject errors at the output of the noisy components.

1) Noisy Stochastic Stream Generator: Recall that $\mathbf{\Pi}$ is the random bit generator and consider $p \in [0,1]$. $\mathbf{\Pi}(p) = 1$ with probability p. The noisy random bit generator (or stochastic stream Generator) denoted $\mathbf{\Pi}_{\text{pr}}$ is defined by:

$$\mathbf{\Pi}_{\text{pr}}(p) = \begin{cases} \mathbf{\Pi}(p), & \text{with probability } 1-p_\tau \\ \overline{\mathbf{\Pi}(p)}, & \text{with probability } p_\tau \end{cases}$$

p_τ is referred to as the stochastic stream error probability. It follows that $\mathbf{\Pi}_{\text{pr}}$ behaves like a (noiseless) bit generator with probability $p(1-p_\tau) + (1-p)p_\tau$.

2) Noisy Check Node Processing: Denote $\mathbf{C}$ the output of the noise-free check node unit. The output of the noisy check node unit denoted $\mathbf{C}_{\text{pr}}$ is defined by:

$$\mathbf{C}_{\text{pr}} = \begin{cases} \mathbf{C}, & \text{with probability } 1-p_c \\ \overline{\mathbf{C}}, & \text{with probability } p_c \end{cases}$$

p_c is referred to as the check node error probability.

3) Noisy Variable Node Processing: Denote $\mathbf{V}$ the output of the noise-free variable-node unit. The output of the noisy check node unit denoted $\mathbf{V}_{\text{pr}}$ is defined by:

$$\mathbf{V}_{\text{pr}} = \begin{cases} \mathbf{V}, & \text{with probability } 1-p_v \\ \overline{\mathbf{V}}, & \text{with probability } p_v \end{cases}$$

p_v is referred to as the variable node error probability.

Note that this error model also takes into account errors that can occur during each access to decoder memories.

IV. Statistical Analysis of Finite-Length Decoders

Density-evolution-like analysis is extremely difficult to perform for stochastic decoders, because variable-to-check node messages are computed as functions of *dependent* random variables. Precisely, in the VN-processing step of the stochastic decoding, it can be seen that $\alpha_{m,n}^{(\ell)}$ is a function of Γ_n, $(\beta_{m',n}^{(\ell)})_{m' \in H(n) \setminus m}$, and $\alpha_{m,n}^{(\ell-1)}$. But $(\beta_{m',n}^{(\ell)})_{m' \in H(n) \setminus m}$ and $\alpha_{m,n}^{(\ell-1)}$ are dependent random variables, since $\alpha_{m,n}^{(\ell-1)}$ depends on $(\beta_{m',n}^{(\ell-1)})_{m' \in H(n) \setminus m}$, and the computation tree [10] of $\beta_{m',n}^{(\ell-1)}$ is included in the computation tree of $\beta_{m',n}^{(\ell)}$.

This dependency between $\alpha_{m,n}^{(\ell-1)}$ and $(\beta_{m',n}^{(\ell)})_{m' \in H(n) \setminus m}$ has not been taken into account in the density evolution approach proposed in [11], which explains why the obtained threshold values in *loc. cit.* are even better than the Shannon limit! The problem remains, and is even compounded, with the use of edge-memories. Indeed when the inputs of the variable node disagree, the bit extracted from EMs is one of the values of the variable-node output at a previous non-hold state. The Markov-chain model for edge-memories proposed in [12] also neglects the dependency relation between messages sent on the same edge of the graph at different decoding iterations.

Therefore, to study the impact of hardware noise on the error correction capability of faulty stochastic decoders Monte-Carlo simulations have been carried out for a $(1008, 504)$ and $(3, 6)$-regular LDPC code [13]. Decoders have been simulated over the Binary Symmetric Channel (BSC). All decoders use 48-bit Edge-Memories, channel input probabilities are quantized on 8 bits and the maximum number of cycles is 1000. The scale factor $\mu = 0.12$.

Error models for the two noisy stochastic decoders verify the symmetric condition from [1] and thus ensure the concentration and convergence results around the all-zero codeword. For all the simulations, the all-zero codeword is transmitted through the channel.

The performance of the noisy decoders is compared with their noiseless version and with the floating-point Belief Propagation algorithm with maximum number of iterations equals to 100.

A. Numerical Results for the Noisy EM Stochastic Decoder

Fig. 1 shows the performance of stochastic decoders with noisy Edge-Memories for four values of the EM error probability: $p_{em} \in \{5.10^{-2}, 10^{-2}, 10^{-3}, 10^{-6}\}$. The black curve and the blue curve represent the noiseless Belief Propagation decoder and the noiseless stochastic decoder respectively.

The results show that when $p_{em} \leq 10^{-3}$, the noisy stochastic decoder performs very close to the noiseless decoder. However, when the level of the noise is high in EMs ($p_{em} = 10^{-2}$) the noisy decoder suffers from a small degradation compared to the noiseless decoder in the waterfall region but *outperforms* it in the error-floor region. This result can be explained by the fact that the additional noise from the hardware decreases the correlation between the bits within the stochastic streams, and thus reduces the occurrence of hold states during the decoding process. Since hold states occurs more often at high SNR, the positive impact of the hardware noise become significant at low values of the channel error probability p_0.

As a result, not only stochastic decoder is robust to noise coming from Edge-Memories, but also an appropriate level of this noise can be used to lower the error floor.

B. Numerical Results for Full Noisy Stochastic Decoder

Fig. 2 compares the performance of four noisy stochastic decoders from the second error model with the performance of BP decoder (black curve) and noiseless stochastic decoder (blue curve). To better understand the impact of each noisy unit on the decoder error correction capability, the following noisy stochastic decoders have been simulated.

- A decoder with hardware noise coming only from stochastic stream generators ($p_\tau = 0.01$).
- A decoder with hardware noise coming only from check node processing ($p_c = 0.01$).
- A decoder with hardware noise coming only from variable node processing ($p_v = 0.01$). Recall that in this model the variable node processing also includes the edge memory.
- A decoder with all the above noisy units ($p_\tau = p_c = p_v = 0.01$).

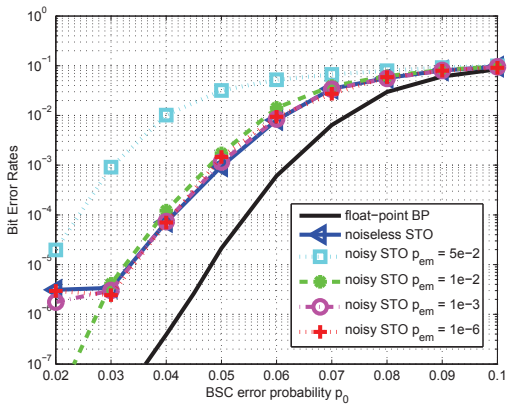

Figure 1. BER performance with noisy Edge-Memories

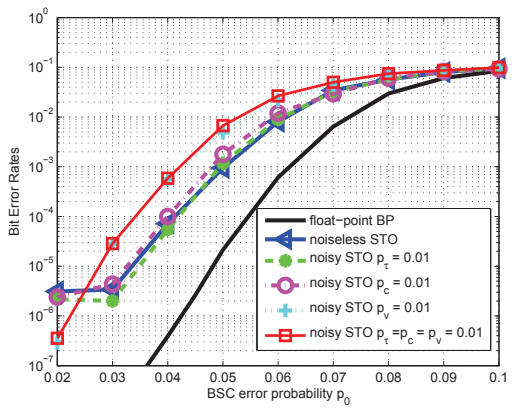

Figure 2. BER performance for the Full Noisy Stochastic Decoder

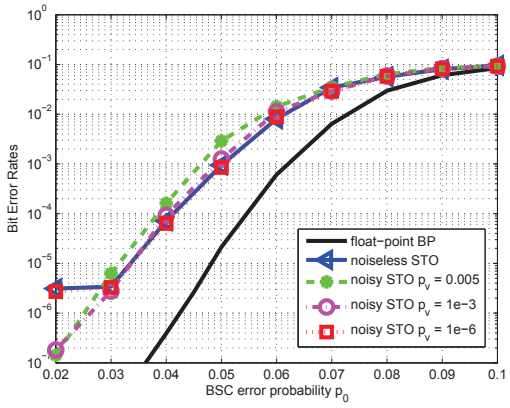

Figure 3. BER performance with only noisy Variable Node Units

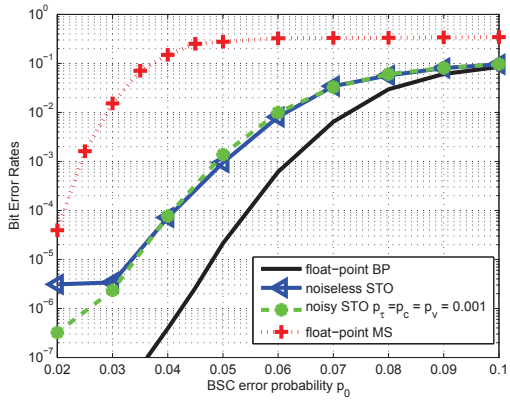

Figure 4. BER performance with $p_\tau = p_c = p_v = 10^{-3}$

According to the results, the hardware noise coming from stochastic stream generators and check node units does not degrade the performance of the decoder. The noise coming from variable node units leads to a sensible loss of performance in the waterfall region, but to a lower error floor (similar to the behavior observed for the stochastic decoding with noisy edge memories only). When $p_\tau = p_c = p_v = 0.01$, the decoder exhibits the same performance than the decoder with only $p_v = 0.01$. This proves that noise from variable node units have the most significant impact on the overall decoder performance.

To determine which value of p_v can provide results closed to the noiseless decoder, further simulations have been carry out with only noisy variable node units and several values of the variable node error probability ($p_v \in \{5.10^{-3}, 10^{-3}, 10^{-6}\}$). From Fig. 3 $p_v = 10^{-3}$ allows similar waterfall region performance than the noiseless decoder and a lower error floor. Note that as for decoder with only noisy edge-memories, a sufficient level of noise helps to avoid hold states in the decoding process, but too much noise ($p_v = 0.01$) will decrease the error correction capability of the decoder in the waterfall region.

Noisy stochastic decoder with $p_\tau = p_c = p_v = 10^{-3}$ has been simulated and compared to the noiseless stochastic decoder and the floating-point belief propagation decoder. According to Fig. 4 this noisy decoder is not only robust to hardware noise but also has a lower error floor. Furthermore, the noisy stochastic decoder outperforms the noiseless floating-point Min-Sum decoder (However, it is known that for the BSC channel, the floating-point min-sum decoder can also be outperformed by the finite-precision (quantized) Min-Sum decoder [14]).

V. Conclusion

In this paper we investigated the performance of stochastic decoders in presence of hardware noise. We proposed two error models to describe the noisy components of the decoders. Our results show that stochastic decoders are robust to hardware noise and that noise in variable node units can be use to improve the decoder performance in the error floor region. This is an important results since the number of edge-memories in stochastic decoders are the main drawback of the decoder. This work shows that their energy consumption can be reduced by diminishing their reliability without any degradation on the error correction capability of the stochastic decoder.

Acknowledgment

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

References

[1] L. R. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Trans. Inf. Theory*, vol. 57, no. 7, pp. 4427–4444, 2011.
[2] S. Yazdi, H. Cho, and L. Dolecek, "Gallager b decoder on noisy hardware," *IEEE Trans. on Comm.*, vol. 66, no. 5, pp. 1660–1673, 2013.
[3] 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.
[4] I. Perez-Andrade, X. Zuo, R. Maunder, B. Al-Hashimi, and L. Hanzo, "Analysis of voltage- and clock-scaling-induced timing errors in stochastic ldpc decoders," in *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, April 2013, pp. 4293–4298.
[5] R. G. Gallager, "Low density parity check codes," MIT Press, Cambridge, 1963, research Monograph series.
[6] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Inf. Theory*, vol. 27, no. 5, pp. 533–547, 1981.
[7] S. Sharifi Tehrani, W. J. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," *IEEE Communications Letters*, vol. 10, no. 10, pp. 716–718, 2006.
[8] S. Sharifi Tehrani, S. Mannor, and W. J. Gross, "Fully parallel stochastic LDPC decoders," *IEEE Trans. on Signal Processing*, vol. 56, no. 11, pp. 5692–5703, 2008.
[9] K.-L. Huang, V. Gaudet, and M. Salehi, "A scaling method for stochastic ldpc decoding over the binary symmetric channel," in *Information Sciences and Systems (CISS), 47th Annual Conference on*, 2013.
[10] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Likoping University, Sweden, 1996.
[11] V. Gaudet and W. Gross, "Switching activity in stochastic decoders," in *Multiple-Valued Logic (ISMVL), 2010 40th IEEE International Symposium on*, May 2010, pp. 167–172.
[12] K.-L. Huang, V. Gaudet, and M. Salehi, "A markov chain model for edge memories in stochastic decoding of ldpc codes," in *Information Sciences and Systems (CISS), 45th Annual Conference on*, 2011.
[13] D. J. MacKay. Encyclopedia of sparse graph codes. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html
[14] C. L. Kameni 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 (ITA)*, 2014.