Stochastic Decoding of Turbo Codes

Q. T. Dong, Matthieu Arzel, Christophe Jego, W. J. Gross

Stochastic decoding of Turbo Codes

Quang Trung DONG[†], Matthieu ARZEL[†]*, Christophe JEGO[†] and Warren J. GROSS[‡]
[†] Institut Telecom; Telecom Bretagne, CNRS Lab-STICC UMR 3192
Université Européenne de Bretagne, France
firstname.lastname@telecom-bretagne.eu
[‡] Department of Electrical and Computer Engineering, McGill University,Montreal, Quebec, H3A 2A7, Canada
wjgross@ece.mcgill.ca

Abstract—**Stochastic computation is a technique in which operations on probabilities are performed on random bit streams. Stochastic decoding of Forward Error-Correction (FEC) codes is inspired by this technique. This paper extends the application of the stochastic decoding approach to the families of convolutional codes and turbo codes. It demonstrates that stochastic computation is a promising solution to improve the data throughput of turbo decoders with very simple implementations. Stochastic fully-parallel turbo decoders are shown to achieve the error correction performance of conventional *A Posteriori* Probability (APP) decoders. To our knowledge, this is the first stochastic turbo decoder which decodes a state-of-the-art turbo code. Additionally, an innovative systematic technique is proposed to cope with stochastic additions, responsible for the throughput bottleneck.**

EDICS

HDW-HDSP

I. Introduction

Iterative Soft-Input Soft-Output (SISO) decoding was first presented by Berrou *et al* in 1993 [1] for the turbo decoding of two parallel concatenated convolutional codes, widely known as turbo codes. Since their invention, turbo codes have received considerable attention due to their performance close to the theoretical limits. They are especially attractive for mobile communication systems and have been adopted as part of several channel coding standards for high data rates such as UMTS and CDMA2000 (third-generation) or 3GPP-LTE (the last step toward the 4^{th} generation). The general concept of iterative SISO decoding has been extended to other families of error-correcting codes such as product codes. It also prompted the rediscovery of Low-Density Parity-Check (LDPC) codes. After many years of research, many decoding algorithms, decoder architectures and circuits were proposed. Although the industrial products were digitally designed, J. Hagenauer [2] and H.-A. Loeliger [3] simultaneously proposed to apply the SISO concept to a continuous-time continuous-value decoding scheme with analog circuits to provide high decoding speeds and/or low power consumptions with extremely simple computation units working in parallel. In 2003, V. Gaudet, a member of the analog decoding community, and A. Rapley, proposed a novel approach [4] based on stochastic computation.

Principles of stochastic computation were described in the 1960's by Gaines [5] and Poppelbaum *et al.* [6] as a method to carry out complex operations with a low hardware complexity. The main feature of this method is that the probabilities are converted into streams of stochastic bits using Bernoulli sequences, in which the information is given by the statistics of the bit streams. As a result, complex arithmetic operations on probabilities such as multiplication and division are transformed into operations on bits using elementary logic gates. This advantage allows architectures to be designed with low computational complexity and enables high data rates to be achieved.

Stochastic computations have been recently considered to decode FEC codes. Early stochastic decoding has been applied to some short error correcting codes such as the (7,4) Hamming code [4] and a (256,121) block turbo code based on two (16,11) Hamming codes [7]. The first implementation of a stochastic decoder with a (16,8) LDPC code was described in [8]. An improved stochastic decoding approach was then proposed to decode practical LDPC codes [9], [10]. This approach was also extended to well-known linear block codes with high-density parity-check matrices, namely BCH codes, Reed Solomon codes and product codes [11]. When compared with conventional Sum-Product implementations, stochastic decoding could provide near-optimal performance for practical LDPC codes. The potential of the stochastic technique for low complexity and high throughput was recently demonstrated by the FPGA implementation of a (1056,528) LDPC decoder [12] which achieved a throughput of 1.66Gb/s. Thus, state-of-the-art decoders combine high throughput and low complexity thanks to the stochastic approach.

This paper proposes to extend stochastic computation to the design of turbo decoders. A major challenge in the implementation of turbo decoders is to achieve high-throughput decoding. Indeed, the next generations of mobile communication systems will require data rates of 1 Gb/s and beyond. Thanks to stochastic decoding, a fully-parallel architecture is a promising response to this challenge. In order to provide a typical study case, the investigation is limited to a single-binary turbo code similar to the ones adopted for the next generation of mobile systems (3GPP-LTE).

This paper is organized as follows. Section II provides a brief overview of the turbo codes, the APP algorithm and the principles of stochastic computation. Section III describes the APP-based stochastic processing applied to the iterative decoding of practical turbo codes. Section IV introduces a method to increase the stochastic decoding throughput. Some simulation results are given in section V to compare the stochastic processing with a conventional decoding using the

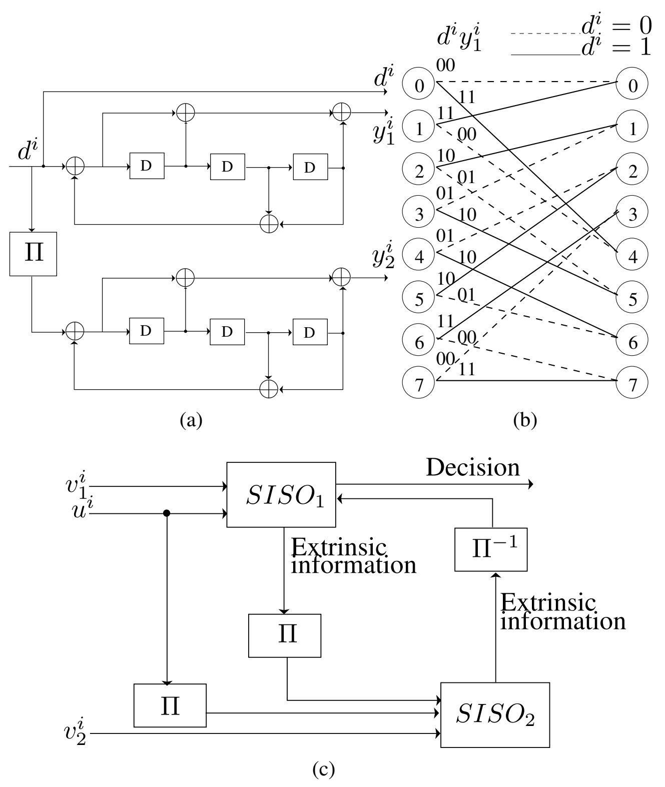


Fig. 1: (a) Turbo encoder; (b) Trellis diagram; (c) Turbo decoder architecture.

A Posteriori Probability algorithm.

II. Background

A. Turbo codes

Fig. 1a shows the structure of a turbo encoder made up of two tail-biting Recursive Systematic Convolutional (RSC) encoders concatenated in parallel thanks to an interleaver. Each RSC code has a coding rate R=1/2, a codeword length $n = 2k$ and a constraint length $\nu = 4$. It can be represented by means of a trellis diagram as shown in Fig. 1b. The overall code rate of the turbo code is $R = 1/3$. At each time i, the information bit (or systematic bit) d^i and two redundancies (or parity bits) y_1^i and y_2^i corresponding to the contributions of each RSC code are provided by the encoder.
The architecture of the turbo decoder illustrated in Fig. 1c is composed of two SISO decoders that exchange some probabilities thanks to an interleaver (Π) and a de-interleaver (Π^{-1}). Each SISO decoder is fed with three different inputs: the channel output corresponding to the systematic bit (u^i), the parity bit produced by the corresponding component encoder (v_1^i or v_2^i), and the extrinsic probabilities computed by the other component decoder. The iterative exchange of extrinsic probabilities between the SISO decoders greatly improves the error correction performance.

B. SISO decoding algorithm

In order to decode convolutional codes, an algorithm known as BCJR was introduced by Bahl *et al.* [13]. It was adapted by Anderson and Hladik to deal with tail-biting codes [14]. The APP decoding process performed by each SISO component decoder can be summarized by the following steps.

1) Branch metric computation: First, the branch metrics $\gamma^i(s', s)$ can be expressed as:

$$\begin{aligned}\gamma^i(s', s) \quad = \quad & \Pr{}^a\left(d^i = j\right) \\ & \times \Pr{}_{in}^{ex}\left(d^i = j | u, v_2\right) \\ & \times \Pr\left(u^i, v_1^i | d^i, y^i\right) \qquad (1)\end{aligned}$$

where d^i is the information bit for the transition from state s' to state s of the trellis at time i. $\Pr^a(d^i = j)$ is the *a priori* probability corresponding to the transition $d^i = j$. If a uniform source is considered, all the symbols have the same probability during the transmission, then $\Pr^a(d^i = j) = 1/2$. $\Pr_{in}^{ex}\left(d^i = j|u, v_2\right)$ is the incoming extrinsic probability computed by the other component decoder. It is calculated from the input sequences u and v_2. In the case of an Additive White Gaussian Noise (AWGN) channel, the third factor is given by:

$$\Pr\left(u^i, v_1^i | d^i, y^i\right) = \exp\left(\frac{\langle u^i, d^i\rangle + \langle v_1^i, y_1^i\rangle}{\sigma^2}\right) \qquad (2)$$

where σ^2 is the variance of the AWGN and $\langle a, b\rangle$ represents the scalar product of two symbols a and b.

2) State metric computation: Second, the forward and backward metrics are recursively calculated as follows:

$$\alpha^{i+1}(s) \quad = \quad \sum_{s'=0}^{2^{\nu-1}-1} \alpha^i(s')\gamma^i(s', s) \qquad (3)$$

$$\beta^i(s') \quad = \quad \sum_{s=0}^{2^{\nu-1}-1} \beta^{i+1}(s)\gamma^i(s, s') \qquad (4)$$

The state metric values are initialized at the same probability, *i.e.* $\alpha^0(s) = \beta^k(s) = 1/2^{\nu-1}$ for any $s \in [0..2^{\nu-1}-1]$. During the decoding process, the state metrics have to be kept in a given range and therefore are normalized regularly.

3) Extrinsic probability computation: Third, in the context of an iterative process, the component decoders exchange extrinsic probabilities calculated as:

$$\Pr{}_{\text{out}}^{\text{ex}}\left(d^i = j | u, v_1\right) = \frac{\displaystyle\sum_{(s',s)/d^i(s',s)=j} \phi_e^i(s', s)}{\displaystyle\sum_{(s',s)} \phi_e^i(s', s)} \qquad (5)$$

where

$$\phi_e^i(s', s) \quad = \quad \alpha^i(s')\beta^{i+1}(s)\gamma_e^i(s', s) \qquad (6)$$

$$\gamma_e^i(s', s) \quad = \quad \exp\left(\frac{\langle v_1^i, y_1^i\rangle}{\sigma^2}\right) \qquad (7)$$

4) A posteriori probability computation: Finally, *a posteriori* probabilities are computed so that:

$$\Pr(d_i = j | u, v_1) = \sum_{(s',s)/d^i(s',s)=j} \phi^i(s', s) \qquad (8)$$

where

$$\phi^i(s', s) = \alpha^i(s')\beta^{i+1}(s)\gamma^i(s', s) \qquad (9)$$

The decoded symbol $\widehat{d^i}$ at time i is equal to the value j that maximizes this *a posteriori* probability.
A sub-optimal version in the logarithmic domain with an acceptable loss of performance referred to as Max-Log-MAP (or Sub-MAP) algorithm was introduced by Robertson *et al.* [15].

C. Stochastic decoding principles

1) Stochastic computation: In a stochastic computing process, the probabilities are converted into Bernoulli sequences using random number generators and comparators [5]. The number of bits at "1" in a stream represents the corresponding probability. For instance, a 10-bit sequence with 4 bits equal to "1" represents a probability of 0.4. Therefore, different stochastic streams may represent the same probability. In order to obtain a good precision, the length of a sequence has to be large. The conventional arithmetic operations, such as multiplication or division, are thus processed by simple logic gates. For instance, the multiplication of a set of N probabilities $p_0, p_1, \ldots, p_{N-1}$ can be achieved by an N-input AND logic gate fed with N mutually independent stochastic streams. The output probability of the AND logic gate is exactly equal to $\prod_{i=0}^{N-1} p_i$. At each time, the bit of each input sequence contributes directly to the output bit. Similarly, the normalisation of two Bernoulli sequences is carried out by means of JK flip-flops [4].

2) The thorny addition: From the equations of the APP algorithm, it can be noted that besides the multiplication and division operations, a huge number of additions is necessary. Since the addition of N values in the interval [0,1] may take values bigger than 1, this operation cannot be done directly with stochastic streams. The addition operands can be scaled equally so that the sum always lays in the interval [0,1]. In practice, a multiplexer that randomly selects one of the N inputs with probability $1/N$ will produce an output stream that is the scaled sum of the input probabilities $\sum_{i=0}^{N-1} \frac{1}{N} p_i$. At each time, each input bit does not contribute directly to the output bit. Consequently, the output sequence length has to be about N times larger than the input sequence lengths to achieve the same precision. This constraint is particularly problematic for the APP-based decoding process. Indeed, many additions are necessary to normalize the state metrics and to compute the extrinsic probabilities. Thus, processing addition operations with multiplexers severely slows down the decoding convergence speed of a turbo decoder.

III. Stochastic decoding applied to turbo codes

A. SISO component decoder architecture

The stochastic decoding of turbo codes requires the stochastic computation to be applied to a tail-biting APP algorithm, which relies on the trellis representation. Fig. 2 details the exchange of information between the various sections of a tail-biting APP decoder. There are as many sections as symbols to decode and each section is made up of four modules. A Γ module is fed by the channel outputs u^i and v^i, which are associated with the i^{th} transmitted symbol d^i and its parity bit y^i. This module converts u^i and v^i into *a priori* probabilities, represented by two stochastic streams to compute the branch metrics and then the forward metrics in an A module and the backward metrics in a B module. These modules are involved in a recursive process since they use the forward and backward metrics α^i and β^{i+1} from their neighbors and provide them α^{i+1} and β^i. A Dec module decides the final value of each binary symbol, $\widehat{d}^{\,i}$ for the transmitted symbol d^i. A last module is also required if the APP decoder is part of a turbo decoder: the Ext module. This module computes the output extrinsic probability Pr_{out}^{ex} which is then used by a Γ module of the second APP decoder as the input Pr_{in}^{ex}. All the modules exchange stochastic streams over a logic gate network based on the code trellis representation. Each stochastic decoding step is referred to as a *decoding cycle* (DC) and corresponds to the output of one new bit for each stochastic unit. The decoding process terminates when a maximum number of DCs is reached.

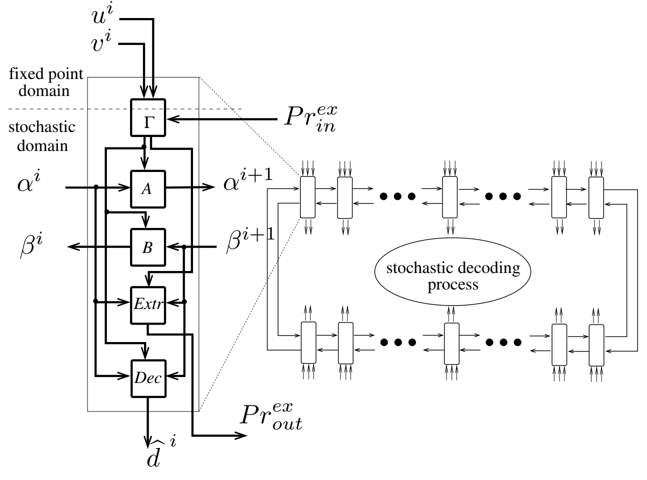


Fig. 2: Stochastic tail-biting APP decoder.

B. Hardware complexity

One major problem in stochastic decoding that deeply degrades the decoding performance is known as the *latching problem* [7]. It is related to the sensitivity to the level of random switching activity (bit transition) [16]. This problem can be easily observed at high Signal-to-Noise Ratios (SNRs). Different solutions have been suggested to solve the latching problem, and thus, to improve the BER performance of stochastic decoding, such as : using supernodes [7], scaling the received Log-Likelihood Ratios (LLRs) up to a maximum value [16], Edge Memories (EMs) insertion and Noise-Dependant Scaling (NDS) [12]. The APP decoders proposed in this paper take advantage of EMs and NDS. In particular, EMs are assigned to stochastic streams that represent forward and backward metric values α^i and β^i to break the correlation using re-randomization. Similarly, EMs are assigned to stochastic streams used for the output extrinsic computation in the module Ext. Overall, ten 32-bit EMs are necessary for each section of the stochastic SISO decoder

TABLE I: Complexity of one section of a stochastic single-binary 8-state turbo decoder with multiplexers for additions.

Module	Elementary hardware resources								Random bits	
	NAND2	AND2	OR2	XOR2	Mux2:1	Mux8:1	3-bit counter	D Flip-flop	7 bits	1 bit
Γ		34	12						2	
A / B		32	8	8	24	8		256		96
Ext		32	2	2	4	2		64		16
Dec		37	2			2	2			6
Total		167	32	18	52	20	2	576	2	214

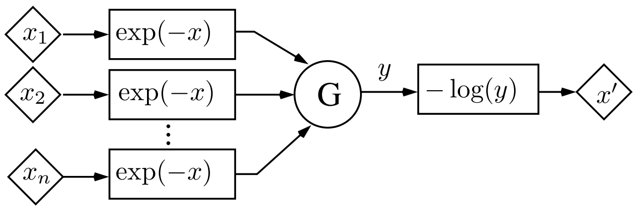


Fig. 3: Principle of exponential-domain computation.

to circumvent the *latching problem*. The complexity of one section of the stochastic decoder in terms of elementary hardware resources and random bits is detailed in Table I. Vectors of 7 random bits are used by the stochastic SISO decoder to convert the channel outputs into stochastic streams. As already mentioned in this paper, the main drawback of this architecture is the need of N-to-1 multiplexers to perform additions. For this reason, solutions have to be investigated to replace these large multiplexers.

IV. Stochastic addition in the exponential computation domain

In order to remove N-to-1 multiplexers for stochastic addition operations, a novel approach is proposed. The main idea is to carry out a critical operation **F** in the exponential domain thanks to the $\exp(-x)$ function. The output values of $\exp(-x)$ modules are then processed by using a simple operation **G**. Then, the result is converted back into a probability thanks to the function $-\log(x)$ as illustrated in Fig. 3. If **F** is the addition operation, then **G** is the multiplication operation, processed by an AND logic gate. Therefore, no large multiplexer is required to perform the stochastic addition operation.

A. Exponential and logarithmic transformations

The idea of processing stochastic streams in the exponential domain was first introduced by *Janer et al.* [17]. The $\exp(-x)$ function is chosen instead of $\exp(x)$ so that the output value can be represented by stochastic streams. In practice, the exponential function can be easily approximated by the first terms of its Taylor's expansion. In [17], the authors described some circuits for the first-, second- and third-order approximations. They also demonstrated that the accuracy of this approximation does not depend on the number of input probabilities that are being added. Therefore, this stochastic exponential transformation opens an efficient way to carry out the conversion of stochastic additions into stochastic multiplications.

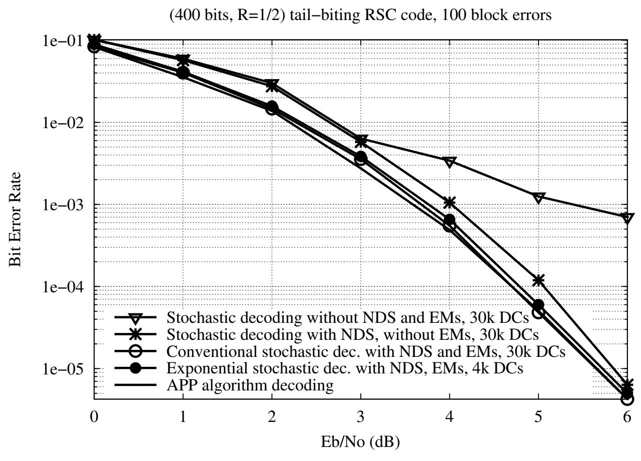
In [17], the result in the exponential domain was sufficient to end the data processing. Unfortunately in a turbo decoder architecture, the result of the addition operation has to be used by another module. Thus, the exponential stochastic stream has to be converted back into a conventional stochastic stream that corresponds to the addition of n terms. A logarithm function is necessary to perform this transformation. A Taylor's expansion is also considered in this case.

B. Hardware complexity

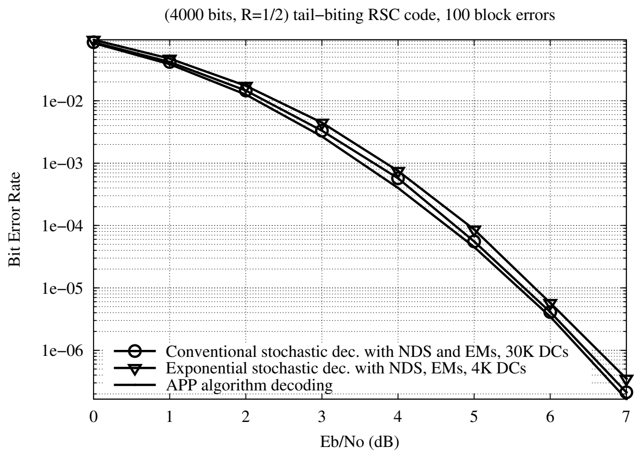
Table II gives a summary of the complexity of one section of the stochastic single-binary 8-state turbo decoder with additions in the exponential domain. Expanding the Taylor series to the second order is sufficient for both exponential and logarithmic modules. The additional cost of addition operations in the exponential domain in terms of hardware resources is reasonable. Indeed, 162 NAND2 logic gates, 205 AND2 logic gates, 98 D Flip-flops and eighteen 2-to-1 multiplexers are necessary to replace the twenty 8-to-1 multiplexers used for addition operations in the probability domain. The hardware complexity of one section of a stochastic single-binary 8-state turbo decoder has to be compared with an fixed-point Sub-MAP counterpart. For such a SISO decoder, the received symbols are 5-bit quantized while the extrinsic information and state metrics are both 7-bit quantized to achieve almost ideal performance [18]. A conventional Sub-MAP decoder is composed of three main parts, namely processing, memory and control. The major problem of the turbo decoders is the memory bottleneck. In order to reduce the state metric memory size, the sliding window principle can be applied, where each received frame has to be divided into several sliding windows. Such a sub-block processing is constrained by the sliding window initialization. To solve this constraint, additional costs in terms of resources and/or latency have to be considered. For a stochastic decoder, a randomization engine is necessary for providing random bits. These random bits are used in 2-to-1 multiplexers and as the addresses of stochastic stream generators. Although this amount of random bits for one section might seem large, as shown in Table II, random bits can be significantly shared by different modules without having an impact in terms of BER performance [12]. Moreover, random number generators using unreliable device behavior have to be considered since they require less hardware resources than conventional linear feedback shift registers. It means that a direct comparison between the two decoding techniques can only be done for the processing unit of one section. The FPGA implementation cost of a fixed-point Sub-MAP decoder must

TABLE II: Complexity of one section of a stochastic single-binary 8-state turbo decoder with additions in the exponential domain.

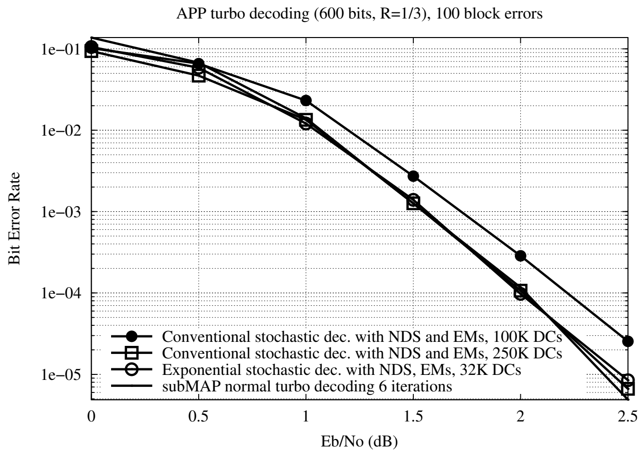
Module	Elementary hardware resources								Random bits	
	NAND2	AND2	OR2	XOR2	Mux2:1	Mux8:1	3-bit counter	D Flip-flop	7 bits	1 bit
Γ		34	12						2	
A / B	48	120	8	8	32			288		88
Ext	34	52	2	2	6			82		16
Dec	32	46		1			1	16		16
Total	162	372	30	19	70		1	674	2	208



(a)



(b)



(c)

Fig. 4: Performance of the stochastic decoding of a rate-1/2 convolutional code for codewords of 400 bits (a) and 4000 bits (b) and of a rate-1/3 turbo code for codewords of 600 bits (c).

be compared with the results given in Table II. One LUT is allocated for each elementary hardware resources of the Table II. In this case, 633 and 638 LUTs are necessary for the fixed-point Sub-MAP and the stochastic versions, respectively. In contrast, the stochastic decoding of one section is less costly in terms of flip-flops. Indeed, the flip-flop number can be decreased from 1398 down to 680 if a stochastic decoder is considered. It means that stochastic decoding is competitive in terms of hardware complexity for turbo codes.

V. Simulation results

In this section, the decoding performance is given for different versions of stochastic decoders for both convolutional and turbo codes. Fig. 4a shows the BER performance of the stochastic decoding of a tail-biting RSC code ($n = 400$ bits, code rate $R = 1/2$) with 30K DCs and different optimizations. A decoder combining NDS and EMs provides a BER performance similar to the one of a conventional APP floating-point algorithm. Moreover, processing additions in the exponential domain enables a decrease of the number of DCs from 30K to 4K with an acceptable performance loss of 0.1dB when compared with the floating-point APP algorithm. Similar conclusions are obtained with a 4000-bit RSC code as shown in Fig. 4b. Thus, the extension of the stochastic decoding to convolutional codes is possible. The BER performance of the proposed stochastic decoding method is provided also for a $(n = 600, R = 1/3)$ turbo code in Fig. 4c. The turbo code is designed with an S-Random interleaver [19]. The EM and NDS techniques are required to achieve good decoding performance. Stochastic turbo decoding needs 250K DCs to achieve the performance of the floating-point Sub-MAP decoding with 6 iterations. Fortunately, the exponential stochastic approach proposed in this paper enables the number of DCs to be reduced from 250K to 32K without any performance degradation. Thus, the proposed summation is a necessary step toward the implementation of high-speed stochastic turbo decoders. To compete with state-of-the-art turbo decoders, a stochastic decoder requires a higher level of parallelism. Two ways have to be explored. First, parallel processing of larger frames of a few thousands of bits – as in wireless communications standards – would be of major interest. Second, representing any probability with p parallel independent streams could divide the number of DCs by p and multiply the throughput by p. Naturally, a higher parallelism will impact the decoder complexity, which is the price to pay for high throughput devices.

VI. Conclusion

This paper extends the application of the stochastic decoding to the families of convolutional codes and turbo codes. Simulation results show performance close to the floating-point Sub-MAP decoding algorithm for $(n = 600, R = 1/3)$ turbo codes. One major problem of a conventional stochastic decoding of turbo codes is the large number of decoding cycles. To reduce the number of cycles, a novel technique for implementing the stochastic addition operation has been investigated. It consists in transforming the stochastic additions into stochastic multiplications in the exponential domain. The number of decoding cycles is thus considerably reduced with no performance degradation. The results provided in this paper validate the potential of stochastic decoding as a practical approach for high-throughput turbo decoders and encourage to keep on investigating in this way.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," in *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[2] J. Hagenauer and M. Winklhofer, "The analog decoder," in *Proc. 1998 IEEE Int. Symp. on Information Theory*, 16-21 Aug 1998, p. 145.

[3] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarköy, "Probability propagation and decoding in analog VLSI," in *Proc. 1998 IEEE Int. Symp. on Information Theory*, 16-21 Aug 1998, p. 146.

[4] V. Gaudet and A. Rapley, "Iterative decoding using stochastic computation," *Electronics Letters*, vol. 39, no. 3, pp. 299–301, Feb. 2003.

[5] B. Gaines, "Stochastic computing," in *AFIPS SJCC*, no. 30, 1967, pp. 149–156.

[6] W. Poppelbaum, C. Afuso, and J. Esch, "Stochastic computing elements and systems," in *AFIPS FJCC*, no. 31, 1967, pp. 635–644.

[7] C. Winstead, V. Gaudet, A. Rapley, and C. Schlegel, "Stochastic iterative decoders," in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, Sept. 2005, pp. 1116–1120.

[8] W. Gross, V. Gaudet, and A. Milner, "Stochastic implementation of LDPC decoders," in *Signals, Systems and Computers, 2005. Conference Record of the Thirty-Ninth Asilomar Conference on*, Oct. 28 - Nov. 1 2005, pp. 713–717.

[9] S. Sharifi Tehrani, W. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," *Communications Letters, IEEE*, vol. 10, no. 10, pp. 716–718, Oct. 2006.

[10] S. Sharifi Tehrani, S. Mannor and W. Gross, "Survey of stochastic computation on factor graphs," in *Multiple-Valued Logic, 2007. ISMVL 2007. 37th International Symposium on*, May 2007, pp. 54–59.

[11] S. Sharifi Tehrani, C. Jego, B. Zhu, and W. Gross, "Stochastic decoding of linear block codes with high-density parity-check matrices," *Signal Processing, IEEE Transactions on*, vol. 56, no. 11, pp. 5733–5739, Nov. 2008.

[12] S. Sharifi Tehrani, S. Mannor, and W. Gross, "Fully parallel stochastic LDPC decoders," *Signal Processing, IEEE Transactions on*, vol. 56, no. 11, pp. 5692–5703, Nov. 2008.

[13] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *Information Theory, IEEE Transactions on*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[14] J. B. Anderson and S. M. Hladik, "Tailbiting MAP decoders," *IEEE Journal on selected areas in communications*, vol. 16, no. 2, Feb. 1998.

[15] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," in *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on*, vol. 2, Jun 1995, pp. 1009–1013 vol.2.

[16] C. Winstead, "Error-control decoders and probabilistic computation," *Tohoku Univ. 3rd SOIM-COE Conf., Sendai, Japan*, Oct. 2005.

[17] C. Janer, J. Quero, J. Ortega, and L. Franquelo, "Fully parallel stochastic computation architecture," *Signal Processing, IEEE Transactions on*, vol. 44, no. 8, pp. 2110–2117, Aug 1996.

[18] G. Montorsi and S. Benedetto, "Design of fixed-point iterative decoders for concatenated codes with interleavers," *Selected Areas in Communications, IEEE Journal on*, vol. 19, no. 5, pp. 871–882, May 2001.

[19] S. Dolinar, D. Divsalar, and F. Pollara, "Weight distributions for turbo codes using random and non-random permutations," *TDA Progress Report 42-122*, August 1995.