Stochastic multiple-stream decoding of Cortex codes

Matthieu Arzel, Cyril Lahuec, Christophe Jego, Warren J. Gross, Yvain Bruned

Stochastic Multiple Stream Decoding of Cortex Codes

Matthieu Arzel, Cyril Lahuec, Christophe Jego *Member, IEEE,*,
Warren J. Gross *Senior Member, IEEE,* and Yvain Bruned

Abstract

Cortex codes are short length block codes having a large Hamming distance. Their modular construction, based on simple and very short block codes, yield to difficulties in efficiently decoding them with digital decoders implementing the Sum-Product algorithm. However, this construction lends itself to analog decoding with performance close to ML decoding as was recently demonstrated. A digital decoding method close to analog decoding is stochastic decoding. This paper brings the two together to design a Cortex stochastic architecture with good decoding performance. Moreover, the proposed stochastic decoder architecture is simplified when compared to the customary one. Instead of edge or tracking forecast memories the proposed architecture uses multiple streams to represent the same probability and deterministic shufflers. This results in a more efficient architecture in terms of ratio between data throughput and hardware complexity. Finally, the proposed method offers decoding performance similar to a Min-Sum decoder with 50 iterations. **EDICS:** HDW-HDSP

Index Terms

Cortex codes, stochastic decoding, Sum-Product algorithm.

Matthieu Arzel and Cyril Lahuec are with Institut Telecom/Telecom Bretagne, CNRS Lab-STICC UMR 3192, Technopôle Brest-Iroise F-29238 BREST Cedex 3, France (email: qt.dong@telecom-bretagne.eu; matthieu.arzel@telecom-bretagne.eu).

Christophe Jego is with CNRS IMS, UMR 5218 351 Cours de la Libération F-33405 TALENCE CEDEX (email: christophe.jego@ims-bordeaux.fr).

Warren J. Gross is with the Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec, H3A 2A7, Canada (email: wjgross@ece.mcgill.ca).

Yvain Bruned is with ENS Cachan/Bretagne, Campus de Ker Lann, F-35170 Bruz, France (email: Yvain.Bruned@eleves.bretagne.ens-cachan.fr).

I. INTRODUCTION

Turbo [1] and LDPC [2], [3] codes allow near optimal decoding performance for codes of block lengths larger than a few hundreds of bits. However, these code families are not adapted to smaller blocks. Also based on iterative encoding and interleaving stages, Cortex codes [4], [5] were invented for this purpose. They are asymptotically good [6] and possess interesting properties such as a systematic construction of self-dual codes [4] and the possibility to associate every type-II self dual code with a Cortex code [7]. Although the Cortex construction offers short codes with large Hamming distances, they are generally difficult to decode with usual digital implementation techniques [7], [8] due to the complexity of the decoding graphs and the large number of hidden variables.

The first efficient Cortex decoder was implemented for codes based on the (4,2,2) Hadamard code with analog circuits [9] applying the Sum-Product Algorithm (SPA). But is it possible to achieve such results with digital circuits? The state-of-the-art digital implementations of the original SPA for LDPC codes consist of different optimized versions of the Min-Sum solution provided in [10], with the exception of stochastic circuits originally proposed in [11]. Stochastic decoders have been so optimized that they provide some of the most efficient implementations of LDPC decoders [12].

Principles of stochastic computation were elaborated in the 1960's [13], [14] as a method to carry out complex operations with a low hardware complexity. 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. Complex arithmetic operations on probabilities such as multiplication and division are transformed into operations on bits using elementary logic gates. Stochastic decoder architectures are designed with low computational complexity and a high level of parallelism [15] [16] achieving high throughputs [12], [17], [18]. Moreover, stochastic decoding is of high interest for low-power decoders [19] and fault-tolerant nanoscale circuits [20].

The main goal of this paper is to show that Cortex codes can be efficiently decoded, despite the complexity of their decoding graphs, by state-of-the-art techniques, *i.e.* stochastic processing. An original idea is also introduced to reduce the number of clock cycles required to decode one codeword and to replace the Edge Memories (EMs) [17] by simple deterministic shufflers.

The paper is organized as follows. Section II summarizes the principles of Cortex codes. A stochastic decoder architecture is presented to deal with them in Section III. This architecture is then compared with analog SPA and digital Min-Sum counterparts in terms of decoding performance in Section IV. Section V concludes the paper.

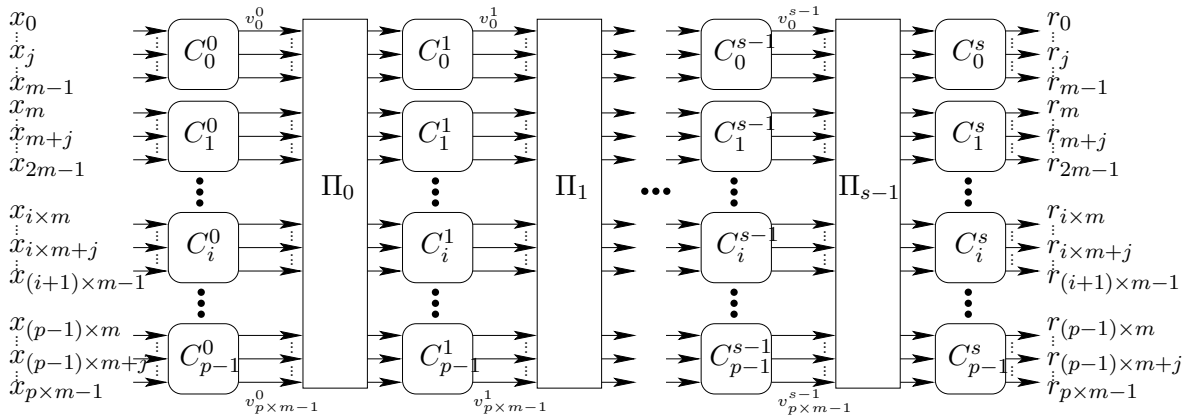


Fig. 1: Cortex encoder made up of s interleaving stages and $(s+1)\times m$ component encoders.

II. Cortex Codes

A. Design principles

Cortex codes were originally designed as systematic codes of rate one half [4]. As shown in Fig. 1, the systematic symbols are grouped in frames of k symbols denoted by x_i. Any frame $\mathbf{x}$ is sliced into p parts of same length m. Each sub-frame, or slice, is independently encoded using a component encoder C_j^0 of rate one half. The resulting redundant symbols are grouped into a frame of length k denoted by $\mathbf{v^0}$. Then, this frame is interleaved over its full-length k. This slicing-encoding-grouping-interleaving scheme is iterated s times. Finally, the resulting frame $\mathbf{v^s}$ is again sliced and encoded to yield the redundant frame $\mathbf{r}$ of length k. If the same self-dual code is used for any component code, then, depending on constraints on the interleavers [21], the resulting Cortex code is also self-dual. Thus, codes with Hamming distances multiple of two or four are built. Using this Cortex construction, many known and new extremal short codes can be built, as shown in [7].

B. C4 codes

Whereas the Cortex codes generally used the extended (8,4,4) Hamming code as component code, other codes can be used. For instance, in [9] the basic (4,2,2) Hadamard code, the smallest self-dual code, was proposed as component code with a simplified Tanner graph as illustrated in Fig. 2.

The resulting Cortex codes are referred to as C4 codes. Moreover, instead of constraining the design of Cortex codes with a reduced number of interleaving stages (one or two) as advised by [6], Perez-Chamorro showed in [22] that Cortex codes with more stages (up to six) have good decoding performance.

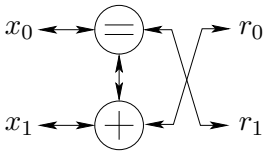


Fig. 2: Simplified bipartite graph of the (4,2,2) Hadamard code.

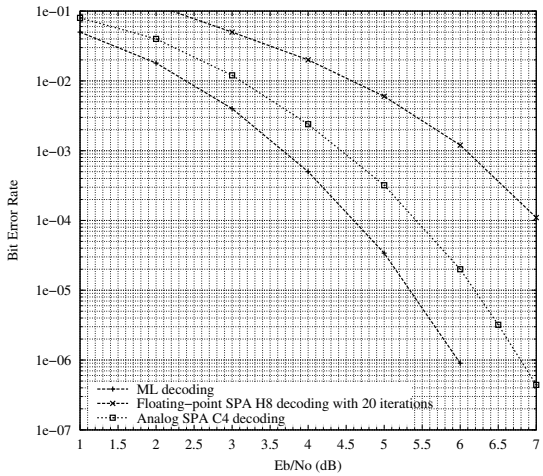


Fig. 3: BER performance of the (32,16,8) Cortex code provided in [7] and in [22].

III. Digital decoder design

A. Choice of a case study

(32,16,8) Cortex codes have been built using different component codes and interleaving stages. First, three stages of (8,4,4) Hamming codes were proposed and digitally decoded with a floating-point SPA in [7]. Second, six stages of (4,2,2) Hadamard codes were introduced and decoded by means of an analog SPA decoder in [22]. Obtained from behavioral simulations, the Bit-Error Rate (BER) performance of both solutions is given in Fig. 3. Note that Perez-Chamorro's solution outperforms Otmani's one by 1.7dB at BER=10^{-5}, because of the girth of the decoding graph equal to 6 in the former case and equal to 4 in the latter case. Therefore, this code is chosen as case study for the proposed stochastic decoding method.

B. Cost of solving the latching problem

A C4 code can be decoded with the SPA by means of a stochastic architecture made up of random number generators (RNGs) and elementary logic gates as described in [17]. One major problem in stochastic decoding which deeply degrades the performance is known as the *latching* problem [16]. It is related to the sensitivity to the level of random switching activity [23], which is especially critical 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 [16], scaling the received Log-Likelihood Ratios (LLRs) up to a maximum value [23], Edge Memories (EMs)

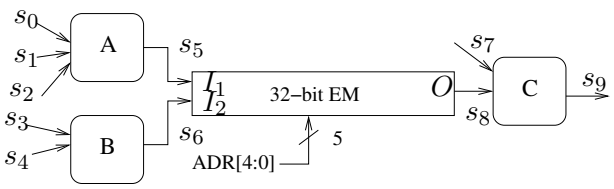


Fig. 4: EM-based architecture with probabilities represented by single stochastic streams and processed by logic units connected through Edge Memories.

insertion and Noise-Dependent Scaling (NDS) [17]. Basically, these solutions aim at re-randomizing and decorrelating stochastic streams.

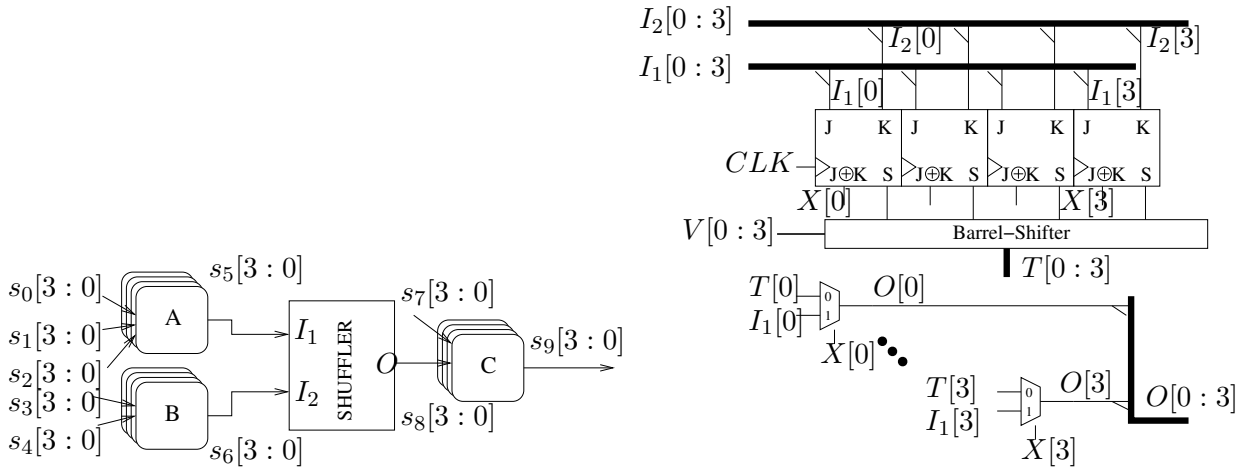
An EM is a complex unit based on a register in which only valuable bits referred to as *regenerative bits*, *i.e.* avoiding signals stuck at '0' or '1', are written and randomly read. Such a unit is efficient to solve the latching problem when the register depth is sufficient (typically between 32 and 64) and when it is duplicated for any vertex of the decoding graph, as detailed in [18]. Thus, EMs are costly in terms of silicon area. To improve that, some alternatives have been proposed such as the Tracking Forecast Memory (TFM) [18] and the Majority-based TFM (MTFM) [12]. A TFM-based decoder was shown to require 40% of the ASIC area of an EM-based decoder with no performance loss. The resulting circuits were shown in [12], [17], [18] to be competitive fully parallel decoders in terms of *architecture efficiency* (data throughput over circuit area). Nevertheless, this optimized solution still requires 8-bit registers, random address generators, adders and comparators.

Re-randomizing and decorrelating stochastic streams can be done without these complex units if all the probabilities are represented by multiple stochastic streams that can be shuffled, as explained in the next section.

C. Multiple stream architecture

An EM-based architecture is illustrated in Fig.4. Each probability P_i is carried by a stochastic stream s_i and is processed with other probabilities by a logic unit – A, B or C – or an EM to avoid latching.

This EM picks up a regenerative bit from a pool when correlation occurs. It is also possible to pick up a regenerative bit from another independent stochastic stream representing the same probability. To reduce the correlation between the concurrent streams, more than two are required. Moreover, the regenerative bit has to be randomly selected among them. Thus, the EM-based architecture in Fig. 4 can be replaced by the multiple stream architecture shown in Fig. 5b where all the streams and the logic gates are duplicated p times (p>2) and the random bit selection is done by a simple shuffler illustrated in Fig. 5b. The shuffler



(a) Multiple stream architecture.

(b) Shuffler handling $p = 4$ streams representing the same probability.

Fig. 5: Novel architecture with probabilities represented by four stochastic streams and processed by parallel logic units connected through shufflers.

is made of a register of p JK flip-flops – providing the stochastic normalization –, a p-bit barrel-shifter and p multiplexers. The barrel-shifter is purely combinatorial. Its shifting value V is updated at each decoding cycle (DC) – which corresponds to the output of a new bit from any Fixed-Point-to-Stochastic-Stream (FX2SS) module – and is the same for any barrel-shifter. FX2SS modules are made up of Random Number Generators (RNGs) and comparators as detailed in [11]. To simplify further the architecture, the shuffling rule can be deterministic to avoid additional RNGs, for instance circularly shifting bits of one position to the left at each DC.

In addition, representing a probability by p stochastic streams instead of one divides the number of DCs by p to insure the same precision. Thus, this novel architecture has a throughput similar to any other stochastic architecture parallelized at degree p as suggested in [11]. Therefore, the multiple stream architecture has to be compared to state-of-the-art solutions in terms of architecture efficiency, *i.e.* ratio between throughput and hardware complexity, as done in the next sections.

D. Stochastic multiple stream Sum-Product decoder

The proposed stochastic processing mimics that of the analog decoder in [9], *i.e.* the decoding is based on the encoding graph. Input data feed the decoding graph in which all the vertices are, as conventionnaly, bidirectionnal. Messages are exchanged along these vertices until the decoder converges and stops. Given the encoding graph of a Cortex code (Fig.1), Fig. 6 illustrates the architecture of a Cortex decoder using the

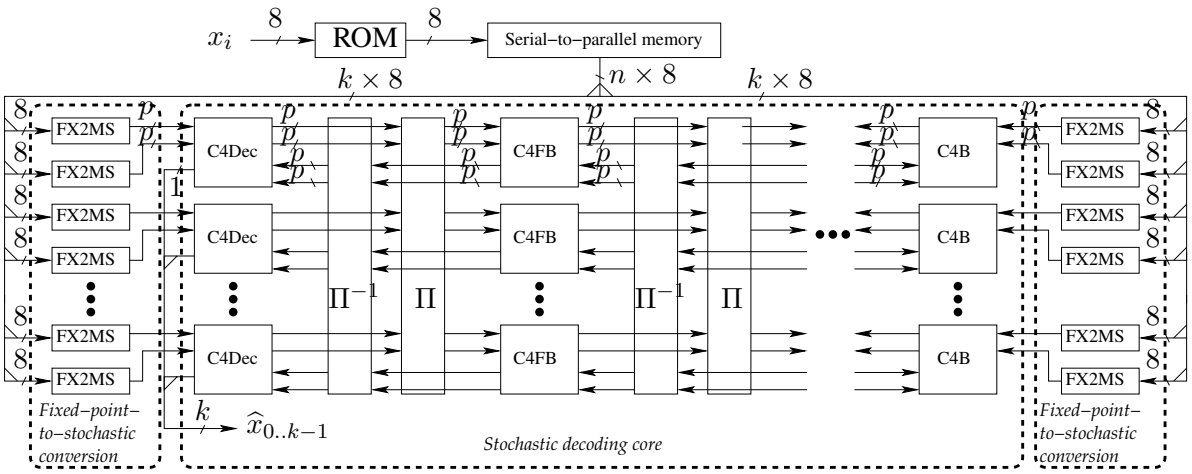


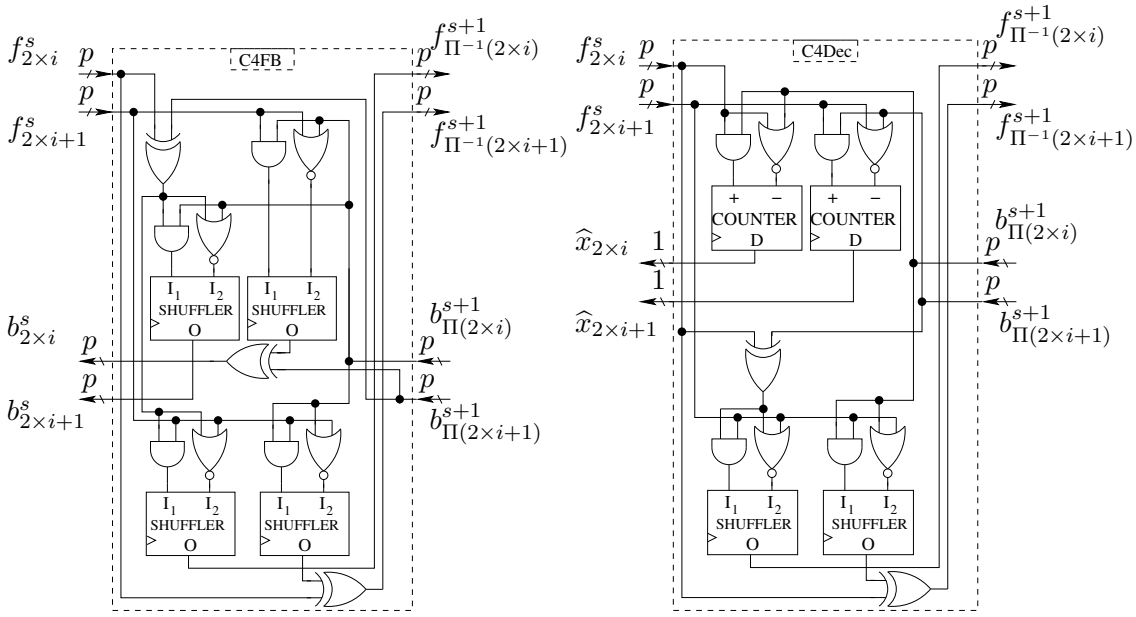
Fig. 6: Stochastic cortex decoder made up of a ROM, a serial-to-parallel memory and a fully parallel multiple stream stochastic core.

proposed multiple stream processing. The channel outputs are converted into symbol probabilities thanks to a ROM and parallelized to feed a stochastic computation core. This fully parallel core is composed of three types of units: converters from fixed-point probabilities into multiple stochastic streams (*FX2MS* units), bidirectional stochastic decoders of (4,2,2) Hadamard codes (*C4Dec*, *C4FB* and *C4B* units) and interleavers (Π and Π^{-1}). The *C4B* modules compute the messages sent to the *C4FB* modules on their left-hand side using the redundant information. The *C4Dec* modules compute the messages needed by the *C4FB* modules on their right-hand side using the systematic information. In addition, these modules take the decision on the decoded bits. The *C4FB* modules computes the messages needed by their left and right-hand side neighbors. This last module is detailed in Fig. 7a. All the interconnections are p-bit buses processed in parallel by the logic gates. For instance each p-bit XOR has two p-bit inputs and one p-bit output and is made up of p conventional XORs (each one has two 1-bit inputs and one 1-bit output).

A *C4B* module is half a *C4FB* one. The *C4Dec* and the *C4B* modules are similar except the former requires two 3-bit counters to provide decisions on the received codeword as illustrated in Fig. 7b. Figures 7a and 7b present the components required by the exact computing. Some flip-flops may be added to reduce the critical path, without any performance loss.

IV. Simulation results

Three digital decoder behavioral models were written in C language to assess the decoding performance of the three decoding methods, namely the proposed 16-stream stochastic decoder, a conventional



(a) C4FB module. (b) C4Dec module.

Fig. 7: Stochastic multiple stream processing modules computing forward/backward messages and decisions. All the shuffler inputs V, which give the shifting value and are driven by the same signal, are not mentioned.

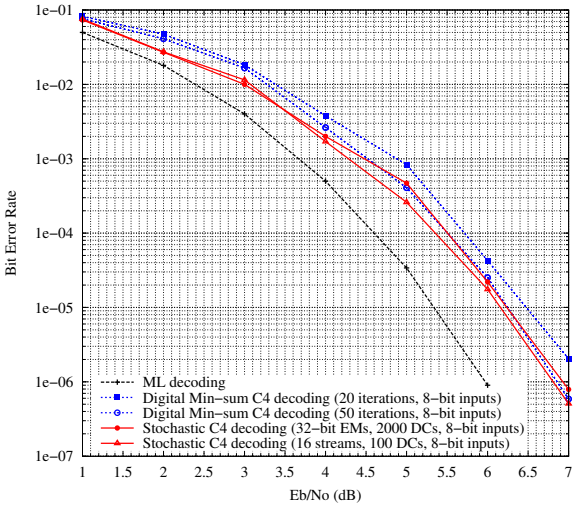
stochastic decoder with 32-bit EMs and a Min-Sum decoder using the flooding schedule and fixed-point 8-bit messages. Note that an iteration consists of updating all the edges of the decoding graph once.

A. Decoding performance

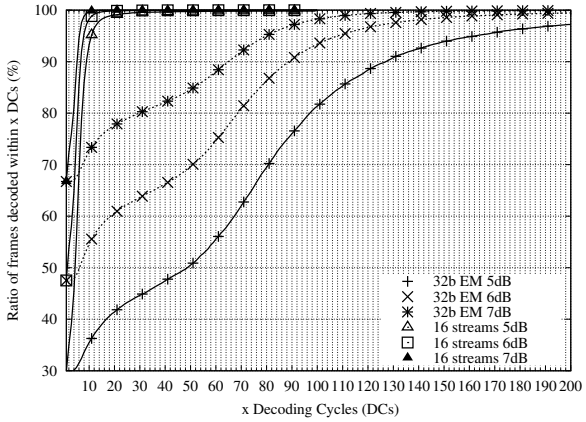
As shown in Fig. 8a, the 16-stream decoder with shufflers outperforms by 0.1dB the conventional stochastic decoder with 32-bit EMs and requires twenty times fewer DCs. Moreover, the 16-stream decoder achieves the BER performance of the 50-iteration Min-Sum decoder.

To achieve high throughput with adapted buffering, the decoding terminates when the decoded frame is a codeword or when a maximum number of DCs is reached. As shown in Fig. 8b, the amount of DCs needed to achieve this criterion depends on the signal-to-noise ratio and on the stochastic decoder used. Actually, while the 32-bit EM-based decoder requires many dozens of DCs to decode up to 90% of the frames, the 16-stream solution decodes 90% of the frames in less than ten DCs . This is correlated with the high ratio (90%) of regenerative bits feeding the shufflers within a few DCs while the EM-based decoder produces 90% of regenerative bits after many dozens of DCs, as illustrated in Fig. 8c.

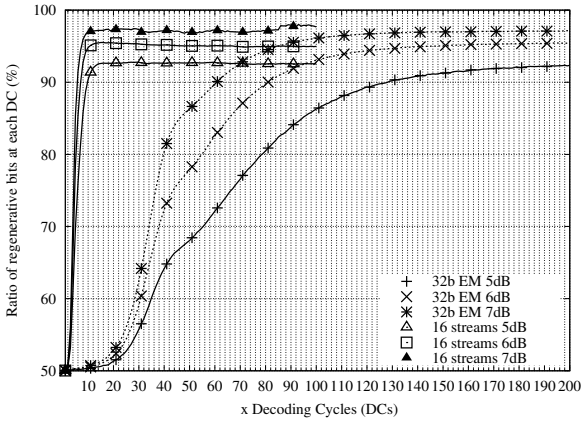
These results confirm stochastic computation as a practical approach for decoding Cortex codes and multiple stream decoding a promising innovation. However, reducing the number of DCs by a factor of



(a) BER performance.



(b) Ratio of frames decoded within a given number of DCs by stochastic decoders using a stopping criterion.



(c) Ratio of regenerative bits at the inputs of the EMs or of the shufflers.

Fig. 8: Performance comparison of 32-bit EM-based and 16-stream decoders for a (32,16,8) Cortex code.

twenty with a BER improvement is not a sufficient metric to fully assess the efficiency of the proposed decoding solution, the circuit complexity and the achievable data throughput must also be compared.

B. Architecture efficiency

The 16-stream decoder with shufflers is compared with a stochastic decoder with 32-bit EMs. First, the efficiencies of the input fixed-point-to-stochastic-stream modules – FX2MS in the multiple stream architectures – has to be assessed. For any FX2MS of the 16-stream decoder, sixteen random bits are generated over one DC. This is sixteen times as complex as generating one random bit over one DC as done by any equivalent FX2SS module of a conventional stochastic decoder. However, since any

TABLE I: Comparison of stochastic Cortex decoders in terms of (a) hardware resources for codewords of length $2 \times k$ and s interleaving stages and of (b) mean number of DCs required at different Eb/N0s before stopping. These numbers of DCs include the initializing phase of one DC. The processing is stopped when the decoded frame is a codeword or the maximum number of DCs is reached.

(a)

	32-bit EM-based	16-stream
Random bits	$k\times$ $(26+10\times(s-1))$	$k \times 256$
Logic gates (normalized count)	$k\times$ $(1.2 + (s-1))$	$k\times$ $(3.5+0.9\times(s-1))$

(b)

Eb/N0	DC_{mean} for a 32-bit EM-based (up to 2000DCs)	DC_{mean} for a 16-stream (up to 100DCs)
5dB	62.5	6.2
6dB	32.9	4.4
7dB	17.2	3.3

16-stream FX2MS run one twentieth of the clock cycles required by any conventional counterpart, the 16-stream architecture improves the efficiency by a ratio of 20/16.

Second, the stochastic core is made of C4FB, C4B and C4Dec modules with either parallel logic gates and shufflers or simple logic gates and 32-bit EMs. The C4FB module of the former architecture requires seven percent less of logic gates than the C4FB module of the latter and runs twenty times less DCs. Moreover, many random bits have to be generated for the EMs whereas the shufflers do not require any. Naturally, efficient architectures as a distributed one [17] have been proposed to generate random bits. Nevertheless, the hardware resources cost for providing random numbers is still high. Thus, the proposed multiple stream stochastic C4FB is about twenty and ten times as efficient as EM-based and TFM-based cores, respectively. However, any C4Dec module has to accumulate one stream per output bit in a single-stream scheme and 16 streams in a 16-stream scheme. The latter requires so many adders that a 16-stream C4Dec is three times as complex as an EM-based single-stream C4Dec, and is finally 20/3 times as efficient. The total amount of random bits and logic gates required to decode C4 codewords of length $2 \times k$ with s interleaving stages was assessed and is given in Tab. Ia. If no stopping criterion is used for the (32,16,8) code with $s = 5$, the 16-stream architecture requires 3.9 times as many random bits and 1.6 times as many logic gates and 1/20 times as many DCs as an EM-based architecture. If the processing is stopped when the decoded frame is a codeword, then the mean DC number depends on the signal-to-noise ratio and can be reduced down to a few DCs as shown in Tab. Ib. In any case, the 16-stream decoder is the most efficient solution. Indeed, the hardware usage is between five and twelve times as efficient with a 16-stream decoder as with an EM-based decoder. Compared to a TFM-based

decoder, the proposed architecture is two to five times as efficient. Thus, the multiple stream architecture offers the best efficiency for the implementation of stochastic Cortex decoders. A TFM-based ASIC LDPC decoder [18] was designed for the ST Microelectronics 90nm process, and clocked at 400MHz. Assuming a similar clock frequency, the 16-stream Cortex decoder can provide a data throughput of 64Mb/s if no stopping criterion is used and up to 1.9Gb/s @ 7dB with a stopping criterion and adapted buffering. If a Cortex Min-Sum decoder could be clocked at 300MHz, as a recent 65nm Min-Sum ASIC decoder is [24], it would process data at 14Mb/s at fifty iterations, with no performance loss, and 34 Mb/s at twenty iterations, with a loss larger than 0.3dB if no stopping criterion is used. Even with a single iteration, this Min-Sum decoder could not operate at more than 680 Mbit/s. Based on these considerations and since previous stochastic architectures were already serious challengers to conventional digital decoders [17], it can be concluded that the multiple stream stochastic architecture is the most valuable solution for a digital ASIC implementation of an iterative Cortex decoder based on the belief-propagation algorithm.

V. Conclusion

An innovative stochastic architecture using multiple streams and deterministic shufflers has been proposed to digitally decode Cortex codes proposed by Perez-Chamorro in [22]. Compared to stochastic decoders using single streams and Edge Memories or TFMs, the proposed stochastic architecture presents the best architecture efficiency defined as the ratio between data throughput and circuit area. Thus, Cortex codes are optimal short codes which can be built and digitally decoded with good BER performance with a belief-propagation algorithm. In the case of a (32,16,8) C4 code, the architecture efficiency of a multiple stream decoder is up to five times as large as the one achieved using state-of-the-art techniques, i.e. with TFMs. Moreover, this multiple stream design could be applied to design LDPC decoders or turbo decoders with some possible architecture-efficiency improvements.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: turbo codes," in *Proc. IEEE International Conference on Communications*, Geneva, May 1993, pp. 1064–1070.

[2] R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. IT-8, pp. 21–28, January 1962.

[3] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, p. 1645, Aug. 1996.

[4] J. Carlach and C. Vervoux, "A new family of block turbo codes," in *Proceedings of 13th Applicable Algebra in Engineering Communication and Computing (AAECC 13)*, Hawaï, USA, November 1999, p. 15.

[5] J. Carlach, A. Otmani, and C. Vervoux, "A new scheme for building good self-dual block codes," in *Information Theory, 2000. Proceedings. IEEE International Symposium on*, 2000, pp. 476–.

[6] G. Olocco and J. Tillich, "A family of self-dual codes which behave in many respects like random linear codes of rate ," in *Information Theory, 2001. Proceedings. 2001 IEEE International Symposium on*, 2001, pp. 15–.

[7] A. Otmani, "Codes cortex et construction de codes auto-duaux optimaux," Ph.D. dissertation, UNIVERSITÉ DE LIMOGES, December 2002.

[8] E. Cadic, "Construction de turbo codes courts possédant de bonnes propriétés de distance minimale," Ph.D. dissertation, UNIVERSITÉ DE LIMOGES, October 2003.

[9] J. Perez-Chamorro, F. Seguin, C. Lahuec, M. Jezequel, and G. Le Mestre, "Decoding a family of dense codes using the sum-product algorithm," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, May 2009, pp. 2685–2688.

[10] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *Communications, IEEE Transactions on*, vol. 47, no. 5, pp. 673 –680, May 1999.

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

[12] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," *Signal Processing, IEEE Transactions on*, vol. 58, no. 9, pp. 4883 –4896, Sep. 2010.

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

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

[15] 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*, 28 2005-Nov. 1 2005, pp. 713–717.

[16] 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.

[17] 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.

[18] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Mannor, and W. Gross, "Tracking forecast memories in stochastic decoders," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, April 2009, pp. 561–564.

[19] V. C. Gaudet and W. J. Gross, "Switching activity in stochastic decoders," in *Proceedings of the 40th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2010)*, May 2010, pp. 167 –172.

[20] C. Winstead and S. Howard, "A probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 56, no. 6, pp. 484 –488, June 2009.

[21] J.-C. Carlach and A. Otmani, "A systematic construction of self-dual codes," *Information Theory, IEEE Transactions on*, vol. 49, no. 11, pp. 3005–3009, Nov. 2003.

[22] J. Perez-Chamorro, "Analogue decoding of the cortex codes," Ph.D. dissertation, TELECOM Bretagne, March 2009.

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

[24] H. Zhong, W. Xu, N. Xie, and T. Zhang, "Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording," *Magnetics, IEEE Transactions on*, vol. 43, no. 12, pp. 4117 –4122, Dec. 2007.