Shuffled Decoding of Serial Concatenated Convolutional Codes

Aomar Bourenane, Matthieu Arzel, Frédéric Guilloud, Alain Thomas

Shuffled Decoding of Serial Concatenated Convolutional Codes

Aomar Bourenane
Space and Communication Lab
Safran Data Systems
Les Ulis, France
aomar.bourenane@safrangroup.com

Matthieu Arzel, Frédéric Guilloud
IMT Atlantique
Lab-STICC, UMR CNRS 6285
F-29238 Brest, France
firstname.lastname@imt-atlantique.fr

Alain Thomas
Space and Communication Lab
Safran Data Systems
Les Ulis, France
alain-dominique.thomas@safrangroup.com

***Abstract*—Shuffled decoding enables to accelerate the extrinsic information exchange during iterative decoding of concatenated codes. It has already been applied to parallel convolutional codes or low-density parity-check codes. In this article, we propose to apply shuffled decoding to serial concatenation convolutional codes. We take advantage of their systematic encoding to propose an efficient shuffled decoding scheme. Compared to a standard iterative decoding scheme, the convergence of our shuffled implementation is obtained within fewer iterations, each one requiring also less time to be completed. This convergence acceleration yields doubling the throughput. We finally show that doubling the throughput comes at a lower cost than doubling the hardware resources, making this shuffled scheme efficient in term of implementation. For instance, the memory usage is 29% more efficient thanks to our proposal than a baseline scheme, which significantly reduces the power consumption of hardware decoders.**

***Index Terms*—Serial Concatenated Convolutional Codes (SCCC), iterative decoding, shuffled decoding, parallel implementations.**

I. Introduction

Serial Concatenated Convolutional Codes (SCCC) [1] has been proposed by S.Benedetto and G. Montorsi as an alternative scheme to the Parallel Concatenated Convolutional Codes (PCCC) [2] known as turbo-codes. Their error correcting performance is comparable to their parallel counterpart and to LDPC codes [3]–[5], while exhibiting a better error floor performance. Moreover, the low complexity of SCCC encoders make them suitable for on-board satellite implementations [6] [7]. SCCC encoding consists in the serial concatenation of two convolutional encoders linked by an interleaver, the first one corresponding to the outer code and the second one to the inner code [1]. Thus the iterative decoding scheme consists in decoding successively the inner and the outer codewords using Soft-Input Soft-Output (SISO) algorithms and exchanging extrinsic information with each other.

In the literature, several works have been carried out in order to achieve high throughput turbo-code decoders using different parallelism techniques. Most of the contributions concern PCCCs [8] and few are dedicated to SCCCs decoders even though their efficient implementation is an issue for satellite telemetry systems [9]. In [10], a parallel SCCC-decoder architecture with 16 concurrent SISO decoders was proposed based on sliding windows parallelism technique [11]. In [12], the hardware implementation of a high throughput Max-Log-Map SCCC turbo-decoder for an optical channel was proposed. Parallelism was introduced both in the state metrics computation within the SISO decoders and by using sliding windows. In these previous contributions, the parallelism introduced in SCCC decoders was intended to speed up the metric calculations but not the extrinsic information exchange which is crucial to further increase the decoder throughput.

In [13], the authors present a parallel Turbo Product Code (TPC) decoding approach to increase the throughput while keeping the same error performance level. This approach has been generalized on LDPCs and PCCCs under the name of *Shuffled decoding technique* in [14]. Shuffled decoding consists in accelerating the extrinsic information exchanges between concurrent decoders operating on the same codeword. More precisely, the aim is to exchange the extrinsic information as soon as it is produced and not once the component code decoding is over. The idea is to speed up the convergence of the iterative decoding, and hence the throughput of the decoder. However, care has to be taken to design the decoder since increasing the pace of extrinsic information exchange might also decrease the error rate performance. In [15], the authors presents a parallel decoding design for a PCCC decoder based on the work presented in [13]. In [16], the authors explore and analyze parallelism techniques in parallel turbo decoding, including the shuffled decoding technique.

To the best of our knowledge, shuffled decoding for SCCC has never been addressed in the literature. In this paper, we propose a novel SCCC-based shuffled decoding scheme that increases the processing throughput while improving hardware efficiency.

The rest of the paper is organized as follows. In Section II, we introduce the notations used throughout the paper and we describe the encoding and decoding schemes of SCCC. In Section III, we propose an efficient shuffled implementation for systematic SCCC. In Section IV, the performance of the proposed scheme is addressed and compared to that of a baseline implementation. Conclusions and perspectives are discussed in Section V.

II. Model and notations

For the sake of simplicity, throughout the paper we shall refer to the communication channel shown in Fig.1 consisting of a systematic SCCC encoder, a binary input memoryless additive white Gaussian noise (Bi-AWGN) channel and an SCCC decoder.

The SCCC code consists of an outer convolutional recursive systematic code (RSC) C^o with rate $R^o = \frac{1}{2}$, an interleaver Π and an inner RSC C^i with rate $R^i = \frac{1}{2}$. In order to simplify the notations and without loss of generality, we assume that both component codes have the same constraint length K. The superscripts i and o refer to the inner and outer codes respectively. Let u_k represent the k^{th} information bit at the input of the SCCC encoder, for $k \in [1 \,..\, N]$, and encoded first by the outer encoder. Let c_l^o for $l \in [0 \,..\, 2N]$ represent the outer coded bits which are interleaved into bits v_l and input into the inner encoder. Let also c_m^i for $m \in [0 \,..\, 4N]$ represent the inner coded bits which are modulated and sent over a Bi-AWGN channel. Finally let y_m represent the channel output which is proportional to log-likelihood ratios (LLRs) of c_m^i and let $\hat{u}_k$ represent the estimated information bit. As the outer (resp. inner) encoder is assumed to be systematic, its output c_l^o (resp. c_m^i) is alternatively equal to the systematic bit u_k (resp. v_l) and the redundant bit r_k^o (resp. r_l^i).

The SCCC decoder is depicted in Fig. 2. It consists of two serially concatenated SISO decoders denoted outer SISO and inner SISO, an interleaver and a deinterleaver. The trellis length of the outer (resp. inner) code is N (resp. $2N$) sections. LLRs can be considered as information regarding the decision to make when estimating a bit. Hereafter, information regarding a bit b will be denoted $L(b)$ and decomposed according to: $L(b) = L_c(b) + L_a(b) + L_e(b)$, where L_c denotes the channel information, L_a denotes the *a priori* information and L_e denotes the extrinsic information. The inner SISO is fed by the channel LLRs $L_c(c_m^i)$. Since the inner encoder is systematic, they correspond alternatively to the inner code systematic bits $L_c(v_l)$ and to the inner code redundant bits $L_c(r_l^i)$. The inner SISO is also fed by *a priori* information about the inner code systematic bits, coming from the outer SISO. Only information related to the systematic bits $L(v_l)$ is output to feed through the de-interleaver the outer SISO since they correspond to the coded bits of the outer code. The outer SISO computes information on both the systematic bits $L(u_k)$ and the redundant bits $L(r_k^o)$ to feed back extrinsic information to the inner SISO. $L(u_k)$ is used at the end of the iterative decoding to estimate the transmitted bits. As illustrated in Fig. 2 by the blue dashed wires, we emphasize that part of the channel information, $L_c(v_l)$, is input to both the inner and the outer decoders, since the inner code is systematic.

III. Shuffled SCCC Decoder Implementation

A. Baseline decoder

A standard implementation for SCCC decoding is illustrated in Fig. 3. In this implementation, the hardware resources are shared between the inner and the outer decoders: a single SISO component is instantiated, as well as a single memory for the state metrics and a single memory for the extrinsic information. These resources are used alternatively to process the inner and the outer decoding, one trellis section being processed every clock cycle. The control unit schedules the computing and drives the memory access depending on the component code being decoded. These resources have thus to be sized to support the greedier case.

As computational resources are concerned, the outer SISO has to calculate 2 pieces of extrinsic information per trellis section: one related to systematic bits $L(u)$ and one related to redundant bits $L(r^o)$. On the other hand, the inner SISO has to calculate only one extrinsic information which is related to systematic bits $L(v)$: it means that only half of the resources are then required to calculate the extrinsic information related to the inner decoder.

As memory resources are concerned, since the trellis length of the inner code is twice the one of the outer code (the code rate R^o is set to $1/2$), so has to be the (α, β) State Metrics memory of the inner decoder compared to the one of the outer decoder. Therefore, in this baseline decoder, half of this memory is not used during the outer decoder processing.

B. Asymmetrical Shuffled Decoding (ASD)

The idea of a shuffled implementation is to speed up the extrinsic information exchange, if possible as soon as it is calculated. In the baseline decoder, extrinsic information is used in one component code decoder after it has been calculated for all the bits involved in the other code component decoder. In the proposed shuffled implementation, both component code decoders will proceed simultaneously, using the extrinsic information produced in parallel by each decoder as soon as it is output. Hence, two SISO decoders have to be instantiated as illustrated in Fig. 4.

The serial concatenation induces that the component decoders do not have the same trellis length to decode (hence the *Asymmetrical* term in ASD): considering an outer code rate $R^o = 1/2$, the trellis length of the inner code is twice the outer one's. So processing the component codes does not last the same duration. Let T denote the amount of time required to perform a complete inner code decoding. Then, the outer code is decoded within an amount of time equal to $T/2$ since the trellis length is halved.

As illustrated in Fig. 5, we propose to take advantage of the trellis length ratio to run the outer decoder twice while running a single inner decoder. Note that a replica butterfly scheme [17] is implemented to increase the extrinsic information exchange speed and the decoding throughput (as in [8], [18]). In fact, during the second execution of the outer decoder, it generates new extrinsic information $L_e(c^o)$ that allows to update up to 50% of the inner decoder *a priori* information $L_a(v)$ which can be used in the same iteration. Also, new inner extrinsic information $L_e(v)$ will provide a more reliable *a priori* information $L_a(c^o)$ to be used by the outer decoder in its second execution. Hence, a complete iteration of the proposed ASD requires an amount of time

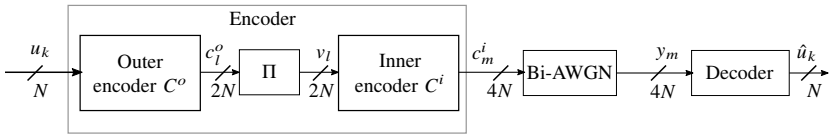

Fig. 1. SCCC encoder and transmission over a binary input AWGN (Bi-AWGN) channel model.

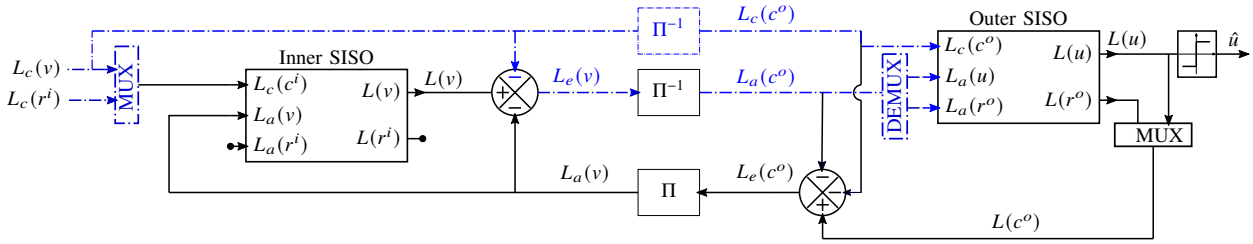

Fig. 2. Considered SCCC iterative decoder, where information (LLR) related to the channel observation is coloured in blue.

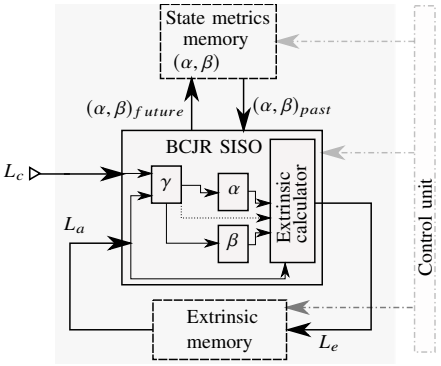

Fig. 3. Baseline decoding scheme: Inner or outer decoder according to the context.

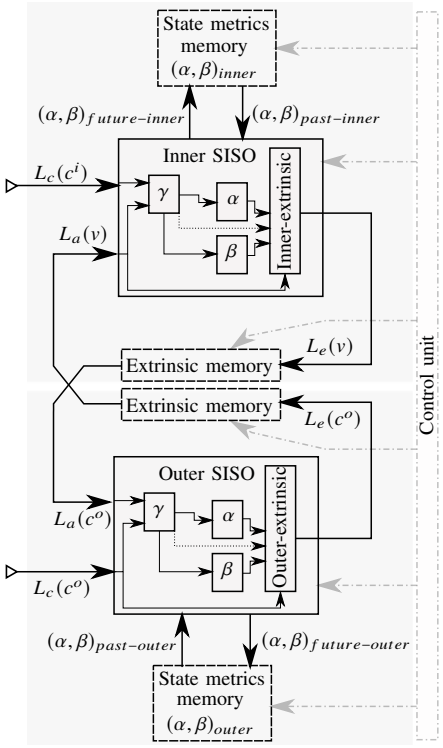

Fig. 4. Proposed ASD scheme.

T, when the baseline implementation with a SISO butterfly scheduling runs a complete iteration within an amount of time equal to $3T/2$: T for the inner code and $T/2$ for the outer one.

C. Hardware Resources

The decoding of the inner and outer codes will be performed using the Max-log-MAP algorithm [19] so as to reduce the computational complexity. To simplify the resource estimation both in terms of memory and of computing units, we assume hereafter that the LLRs and the forward-backward metrics (α,β) are quantized with the same bit-width. The hardware and time resources required by the baseline and proposed ASD schemes are compared in Table I and are detailed hereafter.

1) Memory Requirements: The amount of memory required to save each state metric is given by the number of states per trellis section, *i.e.* 2^{K-1}, times the number of sections plus one, that is $2N+1$ for the inner code and $N+1$ for the outer code. So the number of memory words to store the state metrics of an ASD is given by $2\times\left(2^{K-1}(2N+1)+2^{K-1}(N+1)\right) = 2^K(3N+2)$. For the baseline scheme, the largest trellis is the inner one and requires $2^K(2N+1)$ memory words to be saved.

The amount of memory required to save the extrinsic information in the inner decoder is given by the number of systematic bits which is $2N$. In the outer decoder, it is given by the total number of bits since the extrinsic information of both the systematic and the redundant bits have to be saved, that is $2N$. Note that the extrinsic information memory requirements are duplicated for the ASD scheme to enable the memory to be simultaneously accessed by the inner and the outer decoders as shown in Fig.4. In the baseline case, only one extrinsic memory of $2N$ words is required.

2) Computing Resources: We consider here the amount of resources required to perform the calculation of metrics (α,β) and the extrinsic information in a single trellis section.

The calculation of the state metrics in a trellis section of

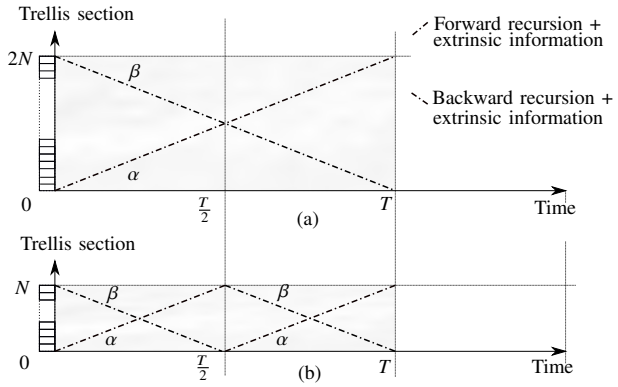

Fig. 5. BCJR computation with replica butterfly scheme : (a) inner decoder scheduling, (b) outer decoder scheduling.

2^{K-1} states is made possible by the use of 2^{K-1} Add-Compare-Select (ACS) units for α or β state metrics, that is 2^K ACS, both for the inner or the outer decoders since we assume the same constraint length K for the two component codes.

The extrinsic information requires first to add the two state metrics to the branch metrics, which requires 2 ADD (Addition operator) per branch, and so $2 \times 2^K = 2^{K+1}$ ADD per trellis section. Then, in the case of the inner decoder (extrinsic information on the systematic bits), the maximum sum over the branches associated to a systematic bit equal to 0 and also over the branches associated to a systematic bit equal to 1 has to be selected. Finding the maximum over $2^K/2$ values requires $2^{K-1} - 1$ Compare-and-Select (CS) units, hence a total of $2^K - 2$ CS. Finally the subtraction between the two previous maximums and subtracting the *a priori* information and the channel information from the result (cost of 3 ADD) will provide the extrinsic information for the systematic bit on the considered trellis section of the inner decoder. To sum up, the inner trellis calculation requires 2^K ACS, $2^{K+1}+3$ ADD and $2^K - 2$ CS. Grouping the CS and ADD operator into a single ACS operator results finally in a total of $2^{K+1}-2$ ACS and 2^K+5 ADD. As for the outer decoder, the calculation is the same, except that 2 extrinsic information values have to be calculated for each trellis section: one for the systematic bit and one for the redundant bit. Hence the number of operations dedicated to the inner extrinsic information is doubled, yielding a total of $2^K+2\times(2^K-2)$ACS and $2\times(2^K+5)$ADD. Finally the overall complexity of the proposed ASD scheme is obtained by adding the complexity of the inner and outer decoders, resulting in $5\times2^K-6$ ACS and $3\times2^K+15$ ADD, whereas for the baseline scheme, the complexity is the one of the outer decoder ($3\times2^K-4$ ACS and $2^{K+1}+10$ ADD).

IV. Performance Comparison

In Fig. 6 both schemes are compared in term of the Bit Error Rate (BER) as a function of signal-to-noise ratio for several number of iterations. The component code is the one described in [20] where $K = 3$ (4-state trellis) and $N = 4320$. It is also illustrated in Fig. 7 as a function of the number of iterations for several signal-to-noise ratios. It can be noticed that the

TABLE I
HARDWARE AND TIME CONSUMPTION OF THE BASELINE DECODING SCHEME VS THE ASD SCHEME

Criteria	Baseline scheme	Proposed ASD scheme
SISO processing time	$3T/2$	T
# Memory words	$2^K(2N+1)+2N$	$2^K(3N+2)+4N$
# ACS	$3\times2^K-4$	$5\times2^K-6$
# ADD	$2^{K+1}+10$	$3\times2^K+15$

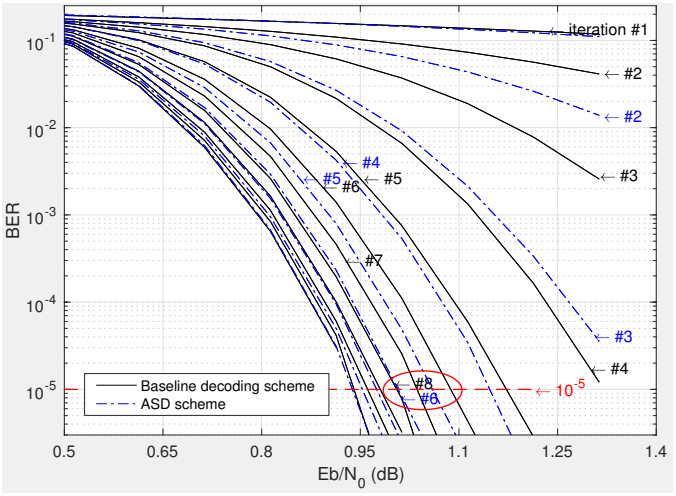

Fig. 6. BER performance comparison as a function of E_b/N_0.

proposed ASD scheme converges in less iterations than the baseline scheme. For instance, a BER of 10^{-5} is reached at $E_b/N_0 \cong 1.41$dB in 8 iterations with the baseline scheme and in only 6 iterations with the proposed ASD scheme. Taking also into account that one iteration of the proposed ASD scheme lasts an amount of time T whereas one iteration of the baseline lasts $3T/2$, the total decoding time of each scheme is given by $T_{\text{ASD}} = 6\times T$ for the proposed ASD scheme and $T_{\text{BD}} = 8\times 3T/2 = 12T$ for the baseline decoding scheme. So, the ASD scheme offers a throughput twice that of the baseline decoding scheme. This faster convergence and higher throughput is obtained at the cost of more hardware resources than implemented for the baseline scheme as summed up in Table II with $K = 3$ and $N = 4320$. So the question is to state if this increase in resources is worth the gain in throughput. To this aim, we suggest to calculate the processing efficiency E of the proposed architecture when compared to the baseline. For each type of resource, E is defined as the ratio between the processing cost (amount of resources times the amount of time they are used) of decoding one codeword with the baseline decoding scheme and that with the ASD scheme:

$$E = \frac{T_{\text{BD}} \times R_{\text{BD}}}{T_{\text{ASD}} \times R_{\text{ASD}}} \tag{1}$$

where R_{ASD} (resp. R_{BD}) is the amount of hardware resources of the ASD scheme (resp. of the baseline decoding scheme), that is the number of memory words, the number of ADD or the number of ACS used. According to Equation (1) and Table II, the efficiency of the ASD scheme is

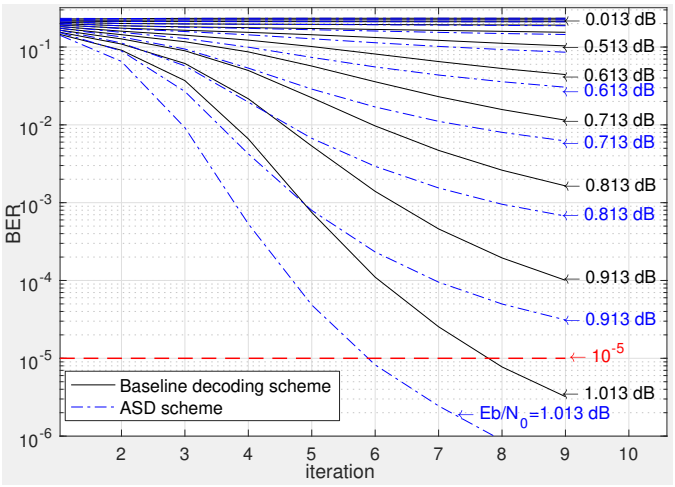

Fig. 7. BER performance comparison as a function of the number of iterations.

TABLE II
NUMERICAL VALUES CONSIDERED FOR THE ASD SCHEME COMPARISON

Criteria	Baseline scheme	Proposed ASD scheme
BER at 10^{-5}	1.41 dB	1.41 dB
Number of iterations	8	6
SISO processing time	$\frac{3T}{2}$	T
# memory words	77768	120976
# ACS	20	34
# ADD	26	39

- 129% in term of memory resources,
- 118% in term of ACS units,
- 133% in term of remaining ADD units.

In other words, when compared to the baseline scheme for a targeted throughput, the proposed ASD scheme offers considerable savings in terms of memory and computing resources.

V. CONCLUSION

In this paper, we have proposed a shuffled scheduling of serial concatenated convolutional codes called Asymmetrical Shuffled Decoding (ASD). Shuffled scheduling makes exchange of extrinsic information faster and our results show a significant reduction in the number of iterations required to achieve a given bit error rate. Moreover, the shuffled scheduling enables a parallel implementation of the inner and outer convolutional code decoders. As a consequence, one iteration of our proposed ASD scheme is faster than one of a conventional iterative decoding scheme. We show that the overall throughput is doubled in our ASD implementation at a lower cost than doubling the hardware resources, making ASD efficient in term of implementation. For instance, the memory usage is 29% more efficient thanks to our proposal than the baseline scheme, which significantly reduces the energy consumption of hardware decoders. The generalization of ASD scheme to any code rates will be considered in a future work.

REFERENCES

[1] S. Benedetto and G. Montorsi, "Serial concatenation of block and convolutional codes," *Electronics Letters*, vol. 32, no. 10, pp. 887–888, 1996.

[2] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[3] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[4] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 909–926, 1998.

[5] A. Graell I Amat, G. Montorsi, and F. Vatta, "Design and performance analysis of a new class of rate compatible serially concatenated convolutional codes," *IEEE Transactions on Communications*, vol. 57, no. 8, pp. 2280–2289, Aug. 2009. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00498307

[6] S. Benedetto, R. Garello, G. Montorsi, C. Berrou, C. Douillard, D. Giancristofaro, A. Ginesi, L. Giugno, and M. Luise, "Mhoms: high-speed acm modem for satellite applications," *IEEE Wireless Communications*, vol. 12, no. 2, pp. 66–77, 2005.

[7] B. BOOK Recommended Standard CCSDS 131.2-B-1, "Flexible advanced coding and modulation scheme for high rate telemetry applications," March 2012. [Online]. Available: https://public.ccsds.org/Pubs/131x2b1e1.pdf

[8] S. Weithoffer, C. Abdel Nour, N. Wehn, C. Douillard, and C. Berrou, "25 Years of Turbo Codes: From Mb/s to beyond 100 Gb/s," in *10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC 2018)*, Hong Kong, Hong Kong SAR China, Dec. 2018. [Online]. Available: https://hal-imt-atlantique.archives-ouvertes.fr/hal-01869012

[9] M. Bertolucci, F. Falaschi, R. Cassettari, D. Davalle, and L. Fanucci, "A comprehensive trade-off analysis on the ccsds 131.2-b-1 extended modcod (sccc-x) implementation," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020, pp. 126–132.

[10] M. Martina, A. Molino, F. Vacca, G. Masera, and G. Montorsi, "High throughput implementation of an adaptive serial concatenation turbo decoder," *Journal of Communications Software and Systems*, vol. 2, no. 3, pp. 252–261, 2006.

[11] C. Schurgers, F. Catthoor, and M. Engels, "Memory optimization of map turbo decoder algorithms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 2, pp. 305–312, 2001.

[12] R. Shoup, "Hardware implementation of a high-throughput 64-ppm serial concatenated turbo decoder," vol. 6311, 08 2006, pp. 63 110S–63 110S.

[13] C. Argon and S. W. McLaughlin, "A parallel decoder for low latency decoding of turbo product codes," *IEEE Communications Letters*, vol. 6, no. 2, pp. 70–72, 2002.

[14] Juntan Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, 2005.

[15] Y. Lu and E. Lu, "A parallel decoder design for low latency turbo decoding," in *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, 2007, pp. 386–386.

[16] O. Muller, A. Baghdadi, and M. Jezequel, "Exploring parallel processing levels for convolutional turbo decoding," in *2006 2nd International Conference on Information Communication Technologies*, vol. 2, 2006, pp. 2353–2358.

[17] Juntan Zhang, Yige Wang, M. Fossorier, and J. S. Yedidia, "Replica shuffled iterative decoding," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, 2005, pp. 454–458.

[18] S. Weithoffer, F. Pohl, and N. Wehn, "On the applicability of trellis compression to turbo-code decoder hardware architectures," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, pp. 61–65.

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

[20] S. Benedetto, R. Garello, G. Montorsi, C. Berrou, C. Douillard, D. Giancristofaro, A. Ginesi, L. Giugno, and M. Luise, "Mhoms: high-speed acm modem for satellite applications," *IEEE Wireless Communications*, vol. 12, no. 2, pp. 66–77, 2005.