Turbo Coding for 4G Systems: Design Issues and Solutions

Alexandre Giulietti[12], Marius Strum[1], Bruno Bougard[2], Liesbet van der Perre[2]

[1]University of São Paulo (USP), Brazil

[2]Interuniversitary Microelectronics Center (IMEC), Belgium

Abstract - **After the initial interest caused by the appearance of turbo codes in 1993, special attention on the implementation has led to their adoption in some of the most important 3G standards. However, future broadband systems (with data rates up to 155 Mb/s) still require a better speed/latency/power performance than those found in current implementations. This paper discusses several design aspects of a broadband, low-power turbo codec owing features that enable its application in the incoming decade systems. The presented ideas were applied to an 80 Mb/s, 2 nJ/bit turbo codec core with latency smaller than 10µs. This flexible architecture allows re-scaling towards faster low power implementations (beyond 1 Gb/s).**

I. INTRODUCTION

Berrou, Glavieux and Thitmajshima showed in 1993 [1] for the first time a feasible channel coding scheme that managed to get within 1 dB from the channel capacity. The so-called turbo codes resulted from the combination of ideas already known by the coding community: convolutional codes, interleaving, and concatenation. These were combined using soft-input and soft-output iterative decoding in order to achieve the highest coding gains ever seen. However, turbo codes Bit-Error-Rate (BER) performance was shadowed by the high complexity of the decoding algorithm, allied to its low throughput and high latency. Several steps towards an efficient implementation have resulted in turbo decoding architectures that reach 10 Mb/s with reasonable power and latency [2][3]. The possibility of using turbo codes in real systems has triggered the interest of standardization bodies. A major landmark was the adoption of turbo codes as one of the IMT-2000 channel coding standards for the third generation (3G) of mobile communications systems, known as UMTS in Europe [4]. It was followed by the approval of turbo codes as the major coding scheme in the interactive channel for European digital video broadcasting (DVB-RCS) [5][6] and in the CCSDS standard for telemetry in space research missions [7]. However, broadband applications like the Wireless Local Area Networks (WLANs) defined in the IEEE 802.11 and Hiperlan2 [8][9] standards, with data rates up to 54 Mb/s and low latency did not adopt turbo coding. WLANs rely on high-order mapping constellations (16-QAM and 64-QAM) to achieve the desired data rate when the channel conditions allow it. Our research on turbo coding was driven by the belief that using a more powerful coding scheme would enable 64-QAM transmission most of the time, thus increasing the overall transmission efficiency. We aimed at optimizing turbo coding implementations as defined in 3G standards towards specifications that met broadband wireless communications. The main goal was to implement a turbo coding demonstrator on silicon that could be used in future 4G standards. Although some of the parameters (e.g. block size) were defined based on the Hiperlan2 standard, the final architecture showed to be very flexible and easily adapted to other specifications.

Starting with a thorough algorithm exploration that defined the best turbo coding scheme combined with the best set of parameters, we carried on with architecture exploration and optimization based on a systematic data transfer and storage exploration [10]. The result was a high-speed, low-power, easily reconfigurable VLSI architecture for turbo encoding and decoding that was mapped into a 0.18µm turbo codec ASIC. This paper presents an overview of the turbo decoder architecture and a detailed insight into some of the design issues that had to be tackled in order to meet the requirements described above.

Section 2 introduces the adopted turbo coding scheme and highlights the major design issues to be considered. Section

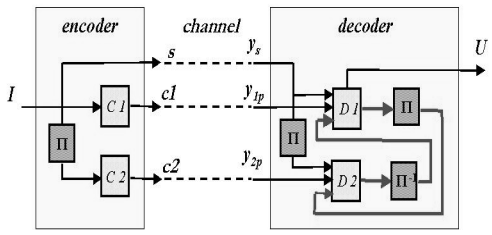

Figure 2.1 Turbo coding scheme.

3 presents the design methodology and final architecture. Section 4 details our solution regarding two major design issues: interleaving and termination. Section 5 presents results regarding speed, area and power of the implemented architecture, plus an implementation loss analysis. Section 6 presents our conclusions.

II. ADOPTED TURBO CODING SCHEME

Turbo coding as presented in [1] is the short name for parallel concatenated convolutional coding (PCCC). Its encoding stage consists of two constituent convolutional encoders operating in parallel, *C1* and *C2* as shown in figure 2.1. *C1* encodes a direct version of the information, whilst *C2* encodes a permuted version, which is obtained through an interleaver π. In our implementation, we used the same encoders as defined in the UMTS standard [5]. Encoded bit streams *c1* and *c2* are transmitted through the channel together with a copy of the original information *s* (systematic information). Therefore the overall code rate is k/n = 1/3, and the encoder outputs should normally be punctured when higher rates are necessary.

In the receiver side, probabilities for systematic information y_s and coded information y_{1p} are fed into decoder *D1*, which provides extrinsic information (a refinement of y_s) to decoder *D2* (first half-iteration). *D2* uses this extrinsic information, the probabilities for coded information y_{2p} and an interleaved version of y_s in order to provide extrinsic information for *D1* (second half-iteration). A deinterleaver π^{-1} between *D1* and *D2* returns the sequences to their orginal order. We adopted the SISO (soft-input, soft-output) max-log MAP algorithm [11][12]. It produces extrinsic information based on state metrics α and β that are calculated in forward and backward recursions over the received block using branch metrics based on the channel values. Throughout successive iterations, the received information is continuously refined towards a final solution.

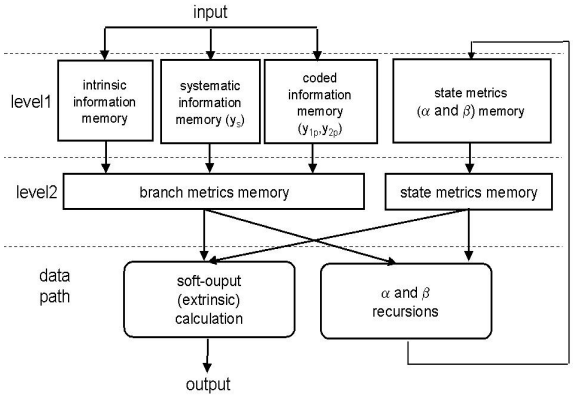

Figure 3.1 MAP decoder architecture: memory hierarchy level and data path.

In order to achieve throughputs of hundreds of Mb/s, the decoding modules *D1* and *D2* should be paralleled, what was done in our case using a windowed approach [13][14][15]. However, breaking the speed bottleneck inside the decoding modules had to be combined with designing a special interleaver that took into account maximizing the bandwidth between the decoder and the storage elements. This interleaver will be detailed in section 4.

III. DESIGN METHODOLOGY AND ARCHITECTURE DEFINITION

The first step of the design was the high-level definition of the main algorithmic parameters. BER performance simulations were done using a model developed in C and Matlab®. Random bit generation, channel simulator, mapping and demapping were implemented using Matlab® functions, while modules to be implemented in VLSI were modeled using C. The interface was implemented using the Mathwork's MEX® compiler. Such a simulator proved to be a fast way to simulate BERs down to 10^{-8} (with a throughput of 4 Kb/s on a PentiumIII/Linux machine). At this stage, serially concatenated block turbo codes (SCBC) [18][19] were also considered as a possible scheme to be implemented in VLSI. Previous analysis as described in [20] indicated PCCCs as the best coding scheme to be further optimized and transferred into silicon due to its more regular decoding algorithm and larger flexibility. Consequently convolutional turbo codes were adopted. Based on simulations and architecture complexity estimations [21], we chose block sizes ranging from 32 to 432 with code rates 1/3,1/2,2/3 and 3/4 (corresponding to 2 OFDM symbols in the Hiperlan2 standard) obtained via puncturing.

The second step was to optimize the C description of the SISO decoder using an IMEC in-house systematic data transfer and storage exploration (DTSE) methodology [21]. It consists first of global data-flow transformations, where data transfer operations are highlighted and ordered. Then global loop transformations are applied, where the inherent recursion of the MAP algorithm was broken and paralleled. This included the definition of the windows architecture and of parameters as window size and optimal number of windows according to block size [20][22]. After the definition of the windows scheme, a 2-level memory hierarchy was introduced. It contains one intrinsic/extrinsic static RAM memory (corresponding to the interleaver storage modules A, B, C and D to be described in the next section), 2 branch metrics memories (to store systematic

and coded information coming from the channel) and a 2-level register file for the storage of state and branch metrics α, β and γ [11]. The first level stores metrics that have to be kept throughout the whole recursion, while the second level stores metrics to be used in the immediate next step of the recursion.

The output of DTSE is a lower-level C description that reflects all elements that will actually be in the architecture (figure 3.1). We used this description and the OCAPI library [23], which makes possible to emulate fixed-point data-flow behavior at C level, to do implementation loss analysis and to generate data for VHDL testbenches. The last step of the design was to build the RTL-VHDL description for the turbo codec. Testing this VHDL was made easier by the exact correspondence between the RTL model and the data-flow OCAPI model. A more detailed description of the turbo codec architecture can be found in [24].

IV. INTERLEAVER DESIGN AND TRELLIS TERMINATION

The BER performance of turbo coding depends greatly on the choice of the interleaver. Interleaving introduces the random component necessary for good coding schemes, as stated by the principles of information theory. Although a lot of research has been carried out regarding design of good interleavers, the problem of integrating them into the decoding system is not normally considered because it does not represent a special problem for non-parallel decoders. However, in the specific case of paralleled MAP decoders, some regularity in the interleaver is necessary in order to avoid conflicts when accessing the storage elements between two decoding half-iterations. Considering single-port memories, that consume less power and are smaller than multi-port memories, maximum throughput is achieved when there are as many storage elements as parallel processors. In order to reach this maximum, we also have to guarantee that every storage element is being accessed just once every cycle. This is illustrated in figure 4.1. At time instant t = 0, four parallel windows W0, W1, W2 and W3 get values from the extrinsic/intrinsic memories A, B, C and D. After one decoding pass, at time t = T (where T is the time necessary to produce the first extrinsic information), W0, W1, W2 and W3 have to store four extrinsic values in A, B, C and D again. Where these values will be stored depends on the permutation introduced by the interleaver. In a random pattern, it is possible that two or more extrinsic values have to be stored into the same storage module at time T (in the example, windows W0 and W1 try to store their extrinsic output in C). Whenever such a *collision* occurs, the corresponding

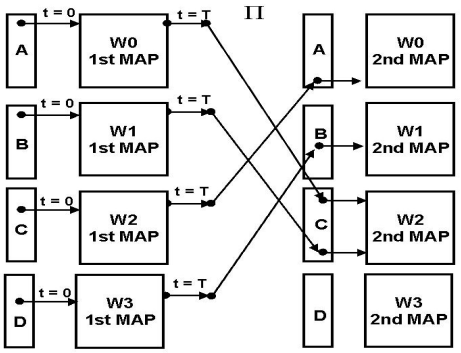

Figure 4.1 A collision when accessing extrinsic memory modules.

windows have to wait at least one cycle more before being able to store the information.

It becomes clear from the previous explanation that some regularity should be introduced in the interleaving pattern in order to avoid collisions.

We developed a systematic way to generate collision-free interleavers [16] that still keep good BER performance. An example is shown in figure 4.2 for N = 16 and W = 4 (where N is the interleaver size and **M** is the window size).

The first step consists in writing the elements of a liner table row by row in a 2-dimensional matrix **M**, with dimensions $(N/W) \times W$. Because the row size is the same as the window size, when reading these elements column by column as in a basic block interleaver the result will be a collision-full interleaver. It means that all extrinsic information produced by the MAP windows at a certain time t is stored in the same memory element. Such regularity can be exploited in order to get a collision-free pattern if progressive cyclic shifts are applied to every column in **M** (right part of figure 4.2): 0 positions for the first column, one position for the second column and so on,

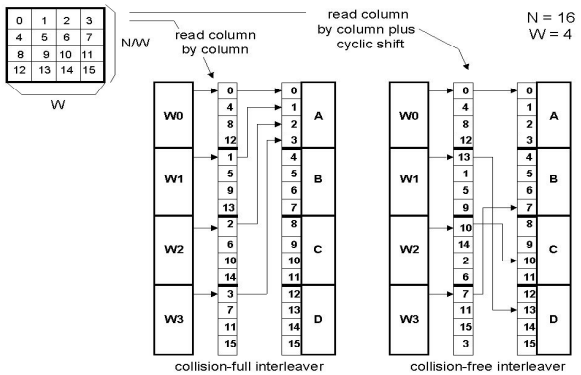

Figure 4.2 Collision-free interleaver generation.

until *(N/W)-1* positions in the last column.
Such a regular pattern has a poor BER performance, and figure 4.2 shows that the first position is even not permuted. Nevertheless, all elements in each row in matrix **M** belong to the same window, so that the collision-free property is not lost if any intra-row permutation is applied. Moreover, the order in which cyclic shifts are applied in every column is not important, provided that two different rows are shifted by a different number of positions. This implies that the collision-free property is not lost if any inter-row permutation is applied. Such freedom can be exploited in order to design interleavers that have good spreading properties and also can mask the expected loss when using non-terminated encoders by avoiding edge effects [17]. In our implementation, we applied the same inter-row permutation as in the UMTS interleaver and a random-like intra-row permutation that mapped the last elements in the permutation table as far as possible to the last positions. The result was a collision-free interleaver with BER performance roughly the same as the UMTS interleaver. On top of that, the effect of using no termination was overcome. Collision-free interleavers as such can be easily generated with on-the-fly address generators based on additions and modulo operations.
The UMTS standard adopted a rather complicated double-termination scheme that results in 6 tail positions appended to the information block to be encoded. In the SNR calculation for the simulation comparing the double terminated and the non-terminated scheme (figure 4.3), a correction factor was applied in order to take into account the effect of introducing tail bits into the information to be transmitted.

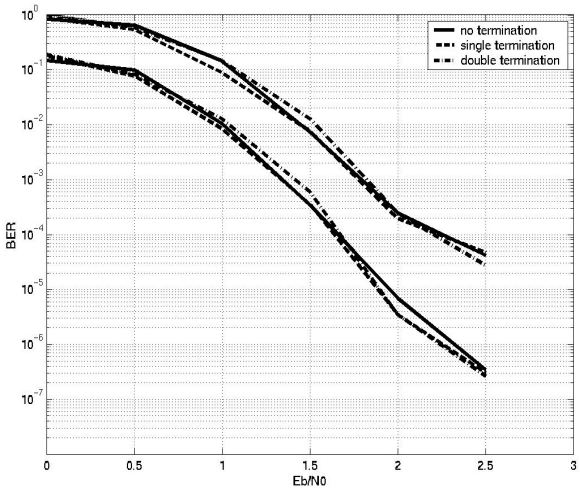

Figure 4.3 Comparison between no termination, single and double termination for N=432, k/n=1/3, BPSK.

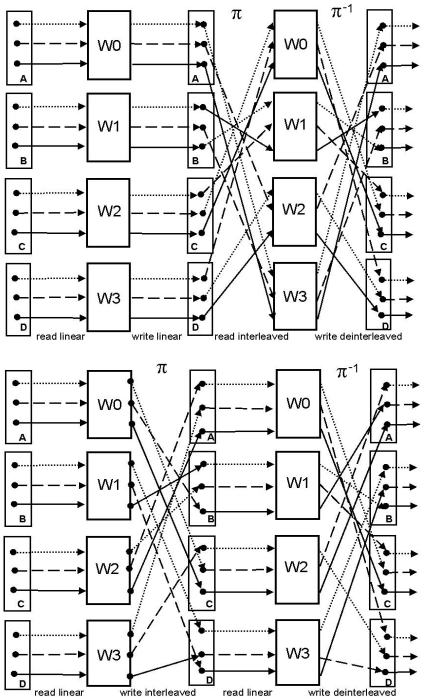

Figure 5.2 Typical and optimized parallel turbo decoding schedule.

Both effects (special interleaver design and correction factor) shrank the difference between the three curves (a single termination simulation was also added for comparison) in figure 4.3 and thus justified the adoption of no termination in our demonstrator.

5 - PARALLEL TURBO DECODING SCHEDULE

In the previous section, we discussed about the collision problem when interleaving output data from the MAP windows. However, in a general turbo decoding scheme as the one depicted in the top part of figure 5.1, avoiding collisions in the interleaving step π does not guarantee a collision-free behavior in the deinterleaving step π^{-1}. In the showed example, a dotted line indicates a data transfer occurring at time instant t = 0; a dashed line indicates a data transfer occurring at time t = 1; a full line indicates a data transfer occurring at time t = 2. Whenever there are two arrows of the same kind arriving at the same storage element (A, B, C or D) there is a collision when writing data. Whenever there are two arrows of the same kind leaving the same storage element there is a collision when reading. In the parallel decoding schedule shown above, interleaving and deinterleaving operations are done when writing into the storage elements. In this scheme, an extrinsic value will not necessarily be stored in the same

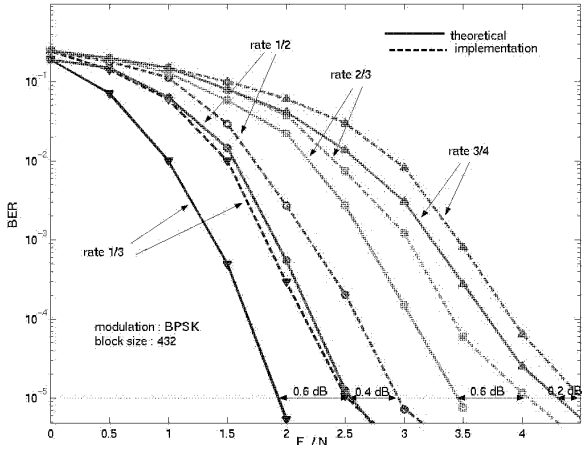

Figure 6.1 Implementation loss analysis for different code rates.

position from where its correspondent intrinsic value was read after one decoding step. The consequence of this fact is that, although being π collision-free, there are 2 collisions in π^{-1} (in memory elements B and D).

We propose a novel decoding schedule in which a collision-free interleaving results in a collision-free deinterleaving (bottom part of figure 5.1). In the first half-iteration intrinsic values are read linearly from the storage elements and the correspondent extrinsic values are stored linearly in the correspondent position. In the second half-iteration intrinsic values are read in interleaved order and extrinsic values are stored in deinterleaved order. This implicates in storing extrinsic values in the same position as their correspondent intrinsic values, thus guaranteeing collision-free deinterleaving if the interleaving is collision-free.

VI. RESULTS

The BER performance of the described architecture is shown in figure 6.1 (based on the OCAPI C++ description), together with the results obtained with the theoretical turbo codec (based on the C/Matlab® environment already mentioned). The implementation loss shown is currently being tested by running simulations with the ASIC testing board. Table 1 shows general features of the implemented 0.18µm turbo codec core (figure 6.2).

TABLE I
CORE MAIN CHARACTERISTICS

Max. clock frequency	170.9 MHz
Max. throughput	80.7 Mbit/s
Latency	< 10 µs
Number of gates	373 K
Active area	7.16 mm^2
Total die size	14.7 mm^2
SRAM size	66 two-port (36 Kbit)

VII. CONCLUSIONS AND FUTURE WORK

This paper aimed mainly at describing some solutions that can be adopted in turbo decoders in order to make them suitable for future 4G systems, namely a parallel interleaver, an optimized parallel decoder schedule and no termination. These solutions combined with a systematic data transfer and storage exploration made possible designing a high throughput, low power architecture mapped into a turbo codec ASIC as described in [24]. The resulting architecture based on parallel windows can be easily tuned to different applications. The presented turbo codec proves that it is possible to have turbo coding with low latency, low power and high-speed. Its BER performance is going to be further evaluated in an IMEC wireless demonstrator consisting of OFDM baseband processing and integrated RF front-end. This set up will also allow power measurements.

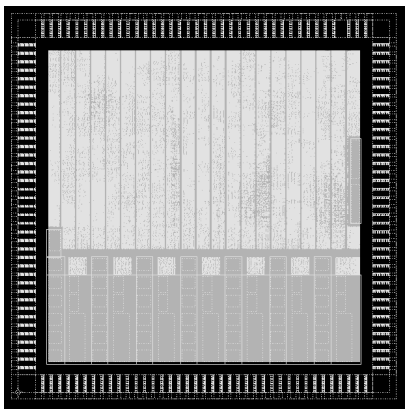

Figure 6.2 Turbo codec ASIC layout.

ACKNOWLEDGMENTS

This work was carried out in the context of a cooperation program between IMEC in Belgium and LME-EPUSP (Brazil). It was partially sponsored by the European Space Agency (ESA) in Europe and CAPES in Brazil. The authors would like to thanks the team that made possible the steps towards the implementation of the turbo codec ASIC: Francien Maessen, Veerle Derudder, Jan-Willem Weijers, Steven Dupont, Johan David and João Paulo Martins.

REFERENCES

[1] C. Berrou, A. Glavieux and P. Thitimajshima, “Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes”, *IEEE International Conference on Communications (ICC’93)*, Vol. 2/3, pp. 1064-1071, Geneva, Switzerland, May 1993.

[2] Small Word Communications, "MAP04T Very High Speed MAP Decoder", data sheet. http://www.sworld.com.au/products.

[3] Turbo Concept, "TC1000 turbo encoder/decoder", data sheet. http://www.turboconcept.com.

[4] Universal Mobile Telecommunications System (UMTS); Multiplexing and Channel Coding (FDD) (3G TS 25.212 version 3.3.0 release 1999. http://www.etsi.org.

[5] C. Douillard, M. Jézéquel, C. Berrou, N. Brengarth, J. Tousch and N. Pham, "The Turbo Code Standard for DVB-RCS", *2nd International Symposium on Turbo Codes and Related Topics"*, Brest, France, September 2000.

[6] European Telecommunication Standard Institute; "Digital Video Broadcasting: Interactive Channel for Satellite Distribution Channel" (DVB-RCS) EN 301 790. http://www.etsi.org

[7] Consultative Committee for Space Data Systems, "Recommendation for Space Data Systems Standards: Telemetry Channel Coding", CCSDS 101.0-B-5. http://www.ccsds.org.

[8] IEEE Std 802.11a – 1999, part 11; Wireless LAN Medium: Access Control (MAC) and Physical Layer (PHY) Specifications.

[9] Broadband Radio Access Networks (BRAN); HIPERLAN Type 2: Physical (PHY) layer (ETSI 101 475 version 1.2.2 release 2001-02).

[10] Cathoor, S. Wuytack, E. de Greef, F. Balasa, L. Nachtergaele and A. Vandecapelle, "Custom Memory Management Methodology, Exploration of Memory Organization for Embedded Multimedia System Design", Kluwer Academic Publishers, 1998.

[11] L.R. Bahl, J. Cocke, F.Jelinek, and J.Raviv, 'Optimal decoding of linear codes for minimizing symbol error rate', in *IEEE Transactions on Information Theory*, vol.IT-20,pp.284-287, Mar. 1974.

[12] P. Robertson, E. Villebrun and P. Hoeher, “A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain”, *IEEE International Conference on Communications*, pp. 1009-1013, 1995.

[13] H. Dawid and H. Meyr, “Real-time algorithms and VLSI Architectures for Soft Output MAP Convolutional Decoding”, *6th IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC’95*), vol. 3, Toronto, pp. 193-197, September 1995.

[14] C. Schurgers, F. Cathoor and M. Engels, "Memory Optimization of MAP Turbo Decoder Algorithms", *IEEE Transactions on VLSI*, vol.9, no. 2, April 2001.

[15] A. Giulietti, M. Strum, B. Gyselinckx, L. van der Perre, F. Maessen, "A Study on Fast, Low-Power VLSI Architectures for Turbo Codes", in *XV International Conference on Microelectronics and Package*, Manaus, September 2000.

[16] A. Giulietti, L. Van der Perre, M. Strum, "Parallel turbo code interleavers : avoiding collisions in accesses to storage elements ", *Electronics Letters*, vol.38, no.5, February 2002.

[17] J. Hokfelt, O. Edfors, T. Maseng, 'On the theory and performance of trellis terminations methods for turbo codes', in *IEEE Journal on Selected Areas in Communications*, vol.19, no5, May 2001.

[18] R. Pyndiah, A. Glavieux, A. Picart and S. Jacq, "Near optimum decoding of product codes", in *Proc. IEEE Globecom Conference* (San Francisco, CA, Nov. 1994), pp. 339-343

[19] J. Fang. F. Buda. E. Lemois, “Turbo Product Code: A Well Suitable Solution To Wireless Packet Tranmission For Very Low Error rates” in *2nd International Symposium on Turbo Codes & Related topics* (Brest, France, 2000), pp. 101-111.

[20] A. Giulietti, J.Liu, F. Maessen, A. Bourdoux, L. Van der Perre, B. Gyselinckx, M. Engels, M. Strum, "A trade-off study on concatenated channel coding techniques for high data rate satellite communications", in *2nd International Symposium on Turbo Codes and Related Topics*, Brest, September 2000.

[21] J. Martins, A. Giulietti, M. Strum, "Performance comparison of convolutional and block turbo codes for WLAN applications", in *4th IEEE International Conference on Devices, Circuits and Systems (ICCDCS)* , Aruba, April 2002.

[22] F. Maessen, A. Giulietti, B. Bougard, L. Van der Perre, F. Catthoor, M. Engels, "Memory power reduction for the high-speed implementation of turbo codes", in *IEEE Workshop on Signal Processing Systems (SIPS) Design and Implementation*, Antwerp, pp.16-24, September 2001.

[23] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde and I. Bolsens, "A Methodology and Design Environment for DSP ASIC Fixed Point Refinement", in *IEEE Design Automation Conference (DATE)*, Munchen, 1999.

[24] A.Giulietti, B.Bougard, V. Derudder, S. Dupont, J.W. Weijers, L. van der Perre, " A 80 Mb/s Low-power Scalable Turbo Codec Core", in *IEEE Custom Integrated Systems Conference (CICC)*, Orlando, May 2002.