TURBO CODING FOR 4G SYSTEMS: DESIGN ISSUES AND SOLUTIONS

Alexandre Giulietti, Marius Strum, Bruno Bougard, Liesbet van der Perre

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 owning 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 10ms. This flexible architecture allows re-scaling towards faster low power implementations (beyond 1 Gb/s).

Keywords: Turbo codes, VLSI, low-power, broadband.

Resumo — Após o interesse inicial causado pelo surgimento dos códigos turbo em 1993, uma atenção especial para a implementação levou à sua utilização em alguns dos mais importantes padrões 3G. Entretanto, os futuros sistemas de faixa larga (com taxas de dados de até 155 Mb/s) requerem um desempenho de velocidade/latência/potência ainda melhor do que aquele encontrado nas atuais implementações. Esse artigo discute diversos aspectos de projeto de um codec turbo de baixa potência cujas características possibilitam a sua aplicação nos sistemas da próxima década. As idéias apresentadas foram aplicadas a um codec turbo operando a 80 Mb/s e 2nJ/bit com latência menor do que 10ms. Essa arquitetura flexível permite reescalonamento, na direção de se ter implementações de mais baixa potência e mais rápidas (acima de 1 Gb/s).

Palavras-chave: Códigos turbo, VLSI, baixa potência, faixa larga.

1. 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 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 third generation (3G) 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].

On the other hand, broadband applications like Wireless Local Area Networks (WLANs) defined in the IEEE 802.11 and Hiperlan2 [8][9] standards, with data rates up to 54 Mb/s, OFDM transmission 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 reconfigurable [31].

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]. Considering that the decoding algorithm is dominated by data transfers as will be shown later, special attention was given to the memory organization and scheduling. Throughout the design, algorithmic issues were tackled together with architectural issues, based on the paradigm that major gains in the final implementation are obtained in the earliest steps. As an example, a special interleaver was conceived where good spreading properties were combined with a structure suitable for parallel VLSI implementation.

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 (Applied Specific Integrated Circuit). 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

Alexandre Giulietti is with Genius Institute of Technology, Brazil. Marius Strum is with the University of São Paulo, Brazil. Bruno Bougard is with the Wireless Communications group of IMEC, Leuven, Belgium and is also a Ph.D. student in the E.E. Dept. of the K.U.Leuven, Belgium. Liesbet Van der Perre is with the IMEC's wireless program. This work has been supported in part by the Flemish Fund for Scientific Research (FWO, Flanders). E-mails: agiulietti@genius.org.br, strum@lme.usp.br, {bougardb, vdperrel}@imec.be.

in order to meet the requirements described above. Section 1 is this Introduction. Section 2 introduces the adopted turbo coding scheme with some details of the decoding algorithm. Section 3 presents the algorithmic exploration that was carried out in the early steps of the design in order to compare the BER (Bit Error Rate) performance of the adopted scheme with other possibilities (namely convolutional codes and block turbo codes). Section 4 presents the design methodology and final architecture. Section 5 details our solution regarding two major design issues: interleaving between decoding modules and trellis termination. Section 6 presents an optimized turbo decoding schedule that supports full-speed data transfer between decoding modules. Section 7 presents results regarding speed, area and power of the implemented ASIC. Section 8 presents our conclusions and future work.

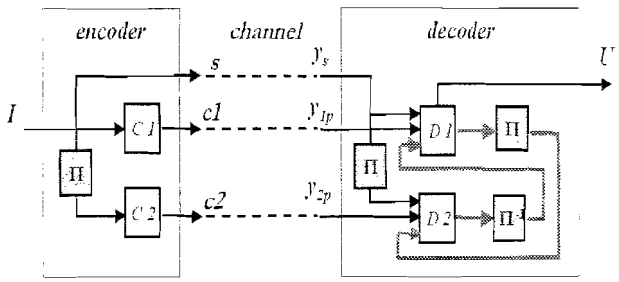

Figure 1. General turbo coding scheme.

2. ADOPTED TURBO CODING SCHEME

Turbo coding as presented in [1] is the short name for parallel concatenated convolutional coding (PCCC). The encoding stage consists of two constituent convolutional encoders operating in parallel, *C1* and *C2* as shown in Figure 1. *C1* encodes a direct version of the information, while *C2* encodes a permuted version, which is obtained through an interleaver π. In our implementation, we used the same 8-state encoders as defined in the UMTS standard [4]. Encoded bit streams *c1* and *c2* are transmitted through the channel together with a copy *s* of the original information (the 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 (soft inputs) 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 back to *D1* (second half-iteration). A deinterleaver π^{-1} between *D1* and *D2* returns the sequences to their original order. We adopted the SISO (soft-input, soft-output) max-log MAP (*Maximum a Posteriori*) algorithm as decoding 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 γ based on the channel values. Throughout successive iterations, the received information is continuously refined towards a final solution. State and branch metrics are calculated as shown in equations (1), (2) and (3):

$$\alpha_k(m) = \Pr\{S_k = m, \mathbf{R}_1^{\,k}\} \tag{1}$$

$$\beta_k(m) = \Pr\{\mathbf{R}_{k+1}^{\,N} \mid S_k = m\} \tag{2}$$

$$\gamma_k(m,m') = \Pr\{S_k = m, R_k \mid S_{k-1} = m'\} \tag{3}$$

Where:

- $\alpha_k(m)$ is the forward state metric at time instant k for state $S_k = m$ (the probability of being in state m at time instant k) given the received symbol sequence $\mathbf{R}_1^{\,k}$ from the beginning of the frame until time instant k.
- $\beta_k(m)$ is the backward state metric at time instant k for state $S_k = m$ given the received symbol sequence $\mathbf{R}_{k+1}^{\,N}$ from the end until time instant $k+1$.
- $\gamma_k(m)$ is the branch metric between states $S_k = m$ and $S_{k-1} = m'$ given the received symbol R_k at time instant k.

Figure 2 illustrates the symbols used in (1), (2) and (3) using a 4-state (an encoder with constraint length $K=2$) trellis representation.

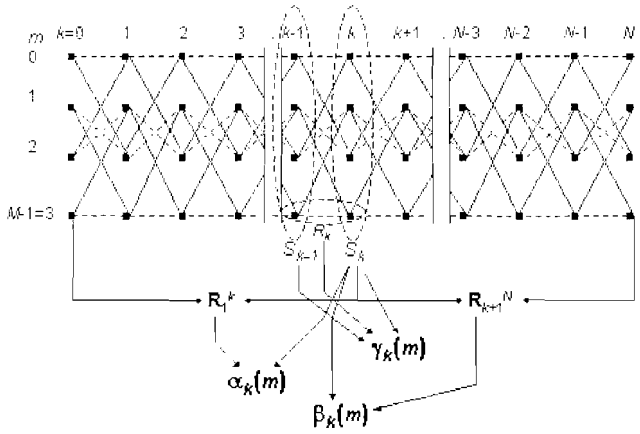

Figure 2. Metrics of the MAP decoding algorithm.

The branch metrics can be split into '1'(solid line) and '0'(dashed line) transitions according to the trellis. On top of that, each one of those '1' and '0' branches can be split into systematic, parity and *a priori* components [12] as shown in equations (4) and (5):

$$\gamma^1_k(m,m') = p(y^s_k \mid d_k = 1) \cdot p(y^p_k \mid d_k = 1, S_k = m, S_{k-1} = m') \cdot p(S_k = m \mid S_{k-1} = m') \tag{4}$$

$$\gamma^0_k(m,m') = p(y^s_k \mid d_k = 0) \cdot p(y^p_k \mid d_k = 0, S_k = m, S_{k-1} = m') \cdot p(S_k = m \mid S_{k-1} = m') \tag{5}$$

The first term in equations (4) and (5) represents the systematic soft-input information on uncoded bits received through the channel; the second term represents soft-input information on coded bits; the third term represents *a priori* information on the state transitions. The *a priori* component is set to zero in the first half-iteration and becomes an actual contribution after that, being replaced by the extrinsic information provided by each decoding stage.

From the Figure it is possible to infer that the APP (*A Posteriori Probability*) of decoded bit d_k is based on the observation of the whole trellis: the combination of metrics computed leftwards and rightwards of time instant k provide an optimal solution for branch and state probabilities at that time instant. The log-likelihood ratio for bit d_k is the log-ratio between state/branch metrics that correspond to '1' transitions in the trellis and state/branch metrics that correspond to '0' transitions in the trellis, as show in equation (6). Hence, the sign of $\Lambda(d_k)$ indicates the hard-decision ('1' for positive and '0' for negative) and the magnitude indicates the reliability of this decision. $\Lambda(d_k)$ is also called the *soft-output* of the MAP decoder.

$$\Lambda(d_k) = \log \frac{\sum_m \sum_{m'} \alpha_{k-1}(m').\gamma^1{}_k(m,m').\beta_k(m)}{\sum_m \sum_{m'} \alpha_{k-1}(m').\gamma^0{}_k(m,m').\beta_k(m)} \quad (6)$$

Splitting the branch metrics as shown in equations (4) and (5), this log-likelihood ratio can be factorized as shown in equation (7). The systematic information term in the branch metric $\gamma_k(m)$ is independent of the encoder states m and m' and therefore can be separated from the summation:

$$\Lambda(d_k) = \log \frac{p(y_k^s \mid d_k = 1)}{p(y_k^s \mid d_k = 0)} + \log \frac{\sum_m \sum_{m'} \alpha_{k-1}(m').\gamma^1{}_k(y_k^p, m, m').\beta_k(m)}{\sum_m \sum_{m'} \alpha_{k-1}(m').\gamma^0{}_k(y_k^p, m, m').\beta_k(m)} \quad (7)$$

The first term represents the received information about uncoded bits and the second term represents the extrinsic information that will feed the iterative process, extracted from an average procedure done over all possible state transitions.

The use of logarithmic calculations allows great simplification of the MAP algorithm, since multiplications are turned into additions and the ratio in equation (6) is turned into a subtraction. Moreover, an extra simplification is possible when using the most probable states in every transition instead of averaging over all possible ones (Figure 3). This yields the log-max MAP algorithm, much less complex than the original one with small effect on the coding gain [12]. Simplified state metrics obtained using the log-max approach are denoted by α' and β' in Figure 3.

Using the most probable states implies in pruning less probable branches as it is done in the ACS (*Add-Compare-Select*) computations of the Viterbi algorithm [28]. However, not only the best path is considered in the decision about the decoded bit. Since a ratio between '0' and '1' transitions is used, soft-output is generated instead. The decoding modules of the turbo decoder are referred to as SISO modules (*Soft-input Soft-output*), since they receive at the input a hard-decision plus a measure of the reliability of that decision and amplify this reliability. The analysis of the equations depicted in Figure 3 shows that the high computational complexity of the log-max MAP algorithm comes from the high number of operations instead of the complexity of the operations themselves. Indeed, only simple maximum operations and additions are needed. Figure 4 shows a schematic representation of a simple implementation of the log-max MAP algorithm. In the Figure, dashed lines represent α recursions and solid lines represent β recursions plus $\Lambda(d_k)$ calculations. The horizontal axis represents the time, where T is the total latency of the algorithm. The vertical axis represents the space, where N is the size of one block. The total state metrics storage is represented by the area of the triangle with base T and height N. For instance, α_0 should be stored from the beginning of the α recursion until the end of the β recursion in order to provide the correspondent $\Lambda(d_0)$.

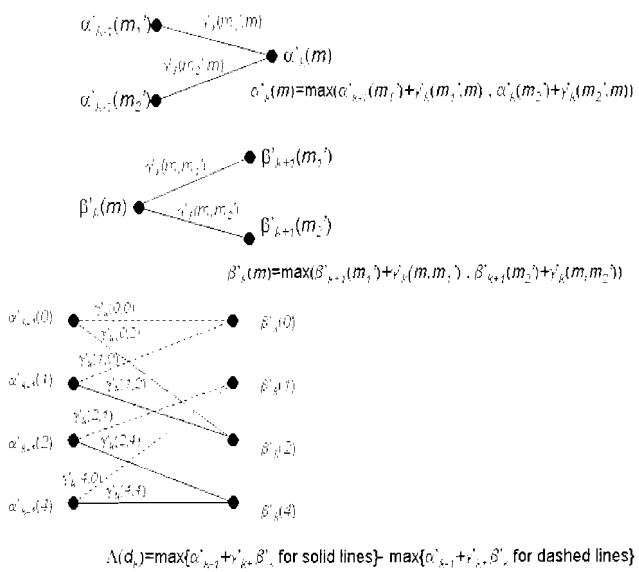

Figure 3. The log-max-MAP decoding algorithm.

3. BER PERFORMANCE ANALYSIS

The adopted turbo coding scheme was compared to other kinds of coding schemes, namely a simple $K = 7$ convolutional code and block turbo codes (from now on referred to as SCBCs, *Serially Concatenated Block Codes*) based on the Fang-Buda algorithm [20]. SCBCs, a variation of product codes, were introduced by Pyndiah *et al.* [19] as an alternative to convolutional turbo codes (from now on referred to as PCCCs, *Parallel Concatenated Convolutional Codes*). Based on the concatenation of Hamming codes [29] or BCH codes [30], they tend to achieve better performance than PCCCs for higher code rates, since no puncturing is needed.

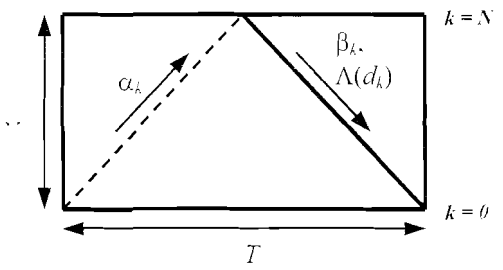

Figure 4. Schematic representation of the log-max MAP algorithm.

Figures 5, 6 and 7 show simulation results for PCCCs and simple convolutional codes for the AWGN channel and an OFDM indoor propagation channel as specified in the Hiperlan2 standard. For the turbo code scheme, the choice of the block size is very important mainly due to the influence of the interleaver in the coding performance. Choosing N= 288 was a good trade-off between interleaver size and latency requirements (< 10μs in Hiperlan2 and IEEE802.11a). Due to the relatively small block size, the gain in performance of turbo codes is only 1.4 dB for 6 iterations and BPSK modulation in an AWGN channel (Figure 5). This difference increases when analyzing the curve for 64-QAM modulated transmission. In this case the PCCC scheme is 3.5 dB better than the convolutional scheme (Figure 6).

For the coded indoor OFDM case, the meaningful information is the packet error rate (*PER* in Figure 7), which indicates the retransmission rate for the wireless network. In our model 50 coded frames containing 576 bits (2 OFDM symbols) were combined into one single packet of 28800 bits. The acceptable packet error rate for the existing standards is 10%, achievable with an Eb/N0 of 22.5 dB for the turbo code case and 26.9 dB for the convolutional code case and 64-QAM (4.4 dB gain for PCCCs when compared to simple convolutional codes).

Figures 8 and 9 compare the performances of PCCCs, SCBCs and convolutional codes for a Gaussian channel and mobile channels respectively [22]. In the Gaussian case almost no difference was noticed between PCCCs and SCBCs for the target *BER* (10^{-6}). The influence of the block size (in the example, 1 and 2 ATM, *Asynchronous Transfer Mode*, cells) was also the same in both schemes. A Rayleigh channel was simulated for a terminal moving at 100 km/h without LOS (Line-of-Sight) with the transmitter; a Ricean channel was also simulated, for the same speed and LOS (typical mobile outdoor satellite reception case). SCBCs performed better than PCCCs in the Rayleigh case, while almost no difference was noticed in the Ricean case. Further analysis showed that the performance of PCCCs in Rayleigh channels can be improved by providing the appropriate channel values to the log-max MAP algorithm [13]. Few conclusions could be taken from the simulations regarding the choice for a scheme to be implemented. The final decision was based on implementation issues as will be seen in the next section.

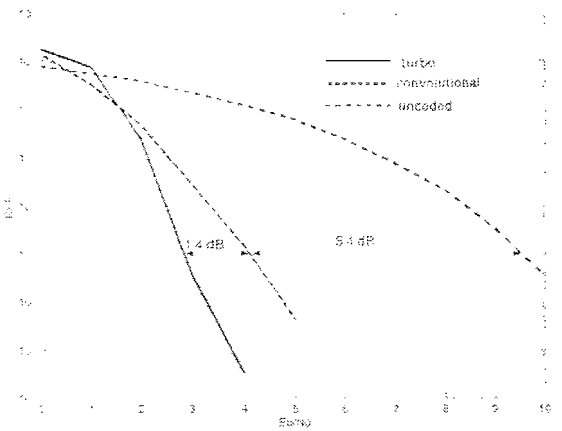

Figure 5. BER performance of PCCCs and convolutional codes in AWGN channel, BPSK, N=288.

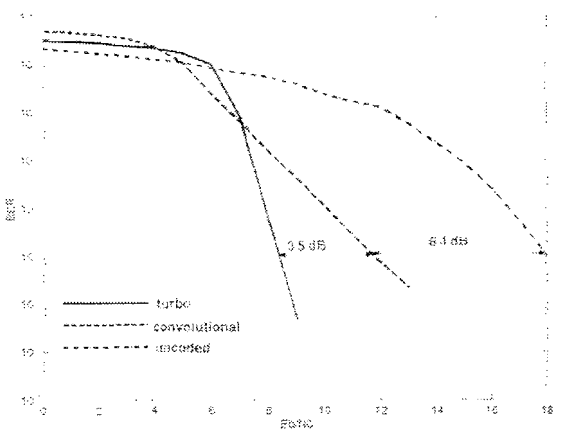

Figure 6. BER performance of PCCCs and convolutional codes in AWGN channel, 64-QAM, N=288.

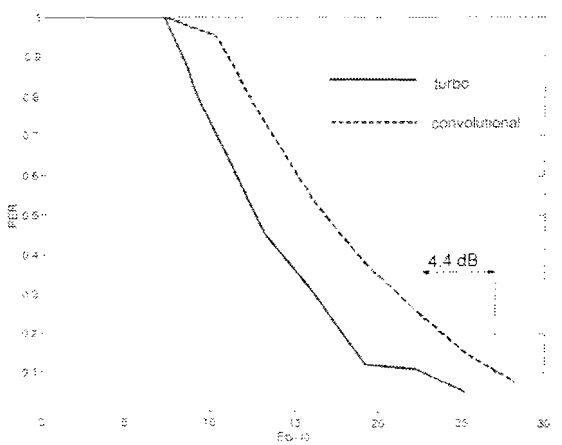

Figure 7. PER performance of turbo and convolutional codes in indoor channel, 64-QAM, N=288.

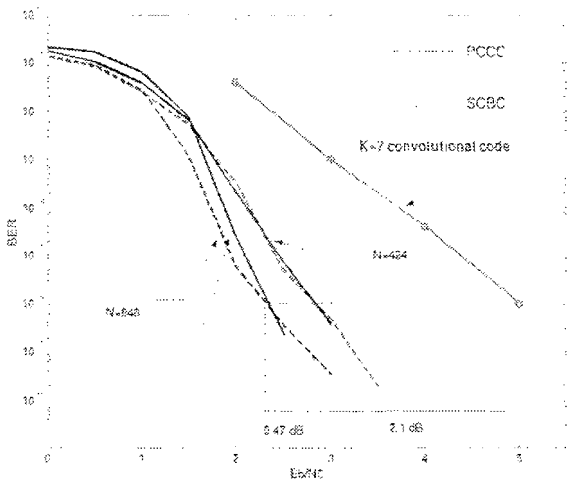

Figure 8. BER performance of SCBCs, PCCCs and convolutional codes in AWGN channel, BPSK.

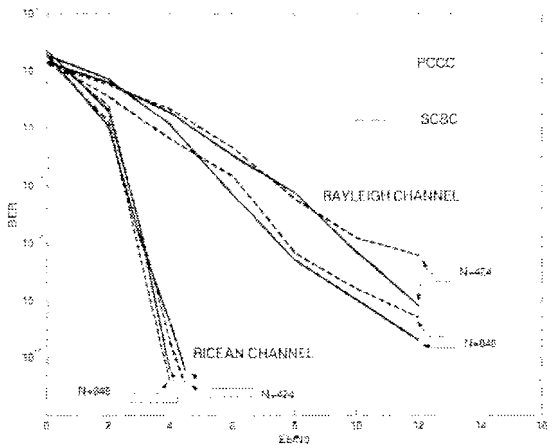

Figure 9. BER performance of SCBCs and PCCCs in mobile channels, BPSK.

4. DESIGN METHODOLOGY AND ARCHITECTURE DEFINITION

In order to achieve throughputs of hundreds of Mb/s, decoding modules *D1* and *D2* (see Figure 1) should be paralleled, what was done in our case using a windowed approach [14][15][16]. On top of that, breaking the speed bottleneck inside the decoding modules was combined with designing a special interleaver that took into account maximizing the data transfer rate between the decoder and the storage elements. Finally, a special memory hierarchy was introduced, increasing locality and minimizing the number of accesses.

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®. Part of the results were already shown in section 3. 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). PCCCs were chosen as the best coding scheme to be further optimized and transferred into silicon due to its more regular decoding algorithm and larger flexibility. Based on simulations and architecture complexity estimations [22][23], we chose block sizes ranging from 32 to 432 (corresponding to 2 OFDM symbols in the Hiperlan2 standard) with code rates 1/3,1/2,2/3 and 3/4 obtained via puncturing.

The second step was to optimize the C description of the SISO decoder using a systematic data transfer and storage exploration (DTSE) methodology [10][15][24]. It consisted first of global data-flow transformations, where data transfer operations were highlighted and ordered. Then global loop transformations were applied, where the inherent recursion of the MAP algorithm was broken and paralleled. This included the definition of the windowed architecture and of parameters as window size and optimal number of windows according to block size [22][23][24].

Using windows to parallelize the recursion shown in Figure 4 involved defining which finite extensions of dashed and solid lines would be processed by which parallel processors. As shown in Figure 10, normally dummy values should be used to initialize the recursion at arbitrary points. In our case, dummy values could be avoided by initializing the parallel recursions with metrics from the previous iteration (*NII. Next Iteration Initialization*) (Figure 11) [32].

Independent small pieces of the MAP recursions were named windows; the combination of windows were called workers. The combination of workers interchanging metric information constituted the parallel MAP. The scheme shown in Figure 10 is referred to as *double*-flow because there is extrinsic information production either in forward or backward directions.

After the definition of the windowed scheme, a 2-level memory hierarchy was introduced. It contained 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. 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 parallel MAP with intrinsic/extrinsic/channel (soft-values) memories is shown in Figure 12. The memory hierarachy scheme is depicted in Figure 13 [24].

This architecture was modeled using a low-level C description that reflected all elements that would actually be present in the VLSI implementation. We used this description together with the OCAPI library [25], which made 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.

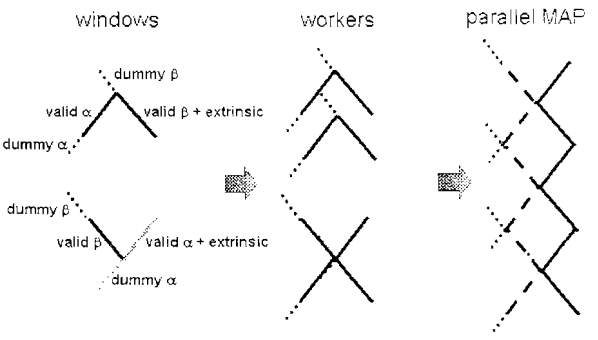

Figure 10. Parallel MAP : windows and workers.

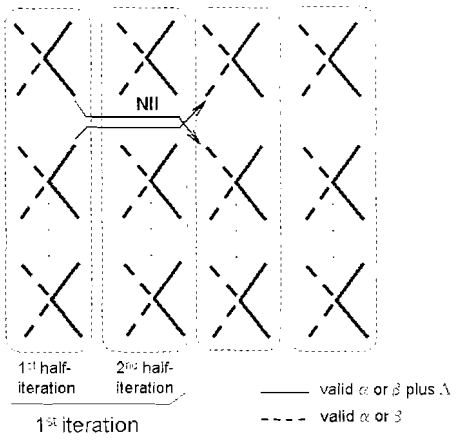

Figure 11. Parallel MAP: NII scheme.

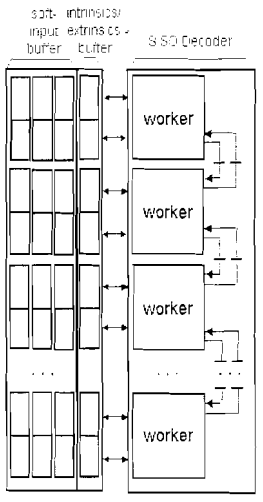

Figure 12. MAP decoder architecture: parallel processing elements and associated memory elements.

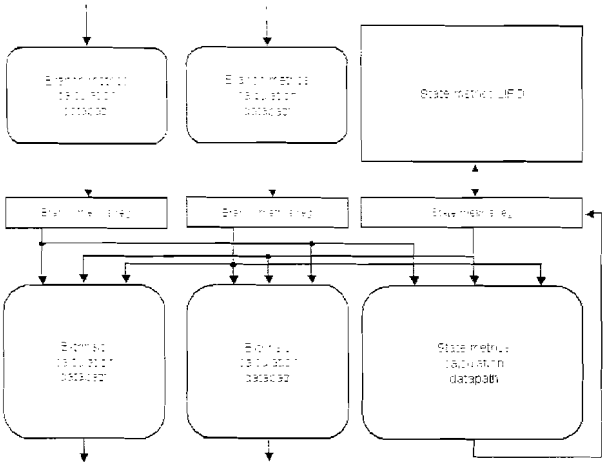

Figure 13. MAP decoder architecture: memory hierarchy levels and data path.

5. INTERLEAVER DESIGN AND TRELLIS TERMINATION

5.1 COLLISION-FREE INTERLEAVER DESIGN

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 [33]. Although a lot of research has been carried out regarding design of good interleavers, the problem of integrating them into the decoding implementation 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, which 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 14. 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 into the same extrinsic/intrinsic memory. In which modules (A, B, C, or D) these values will be stored depends on the permutation π introduced by the interleaver. In a random pattern or in a pattern not constrained by the parallel structure of the decoder, 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 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 [17] that still keep good BER performance. An example is shown in Figure 15 for $N = 16$ and $W = 4$ (where N is the interleaver size and W is the window size).

The first step consists in writing the elements of a linear table with N elements row by row in a 2-dimensional matrix **M**, with dimensions (N/W) x 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 instant t will be stored into 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**: 0 positions for the first column, one position for the second column and so on, until (N/W)-1 positions in the last column.

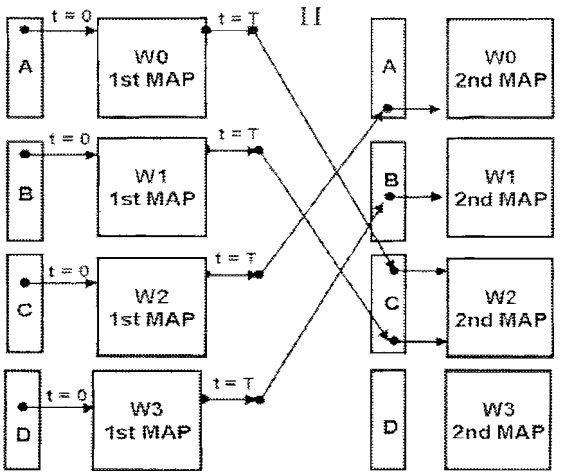

Figura 14. Representation of a collision when accessing storing elements in the MAP parallel decoder architecture.

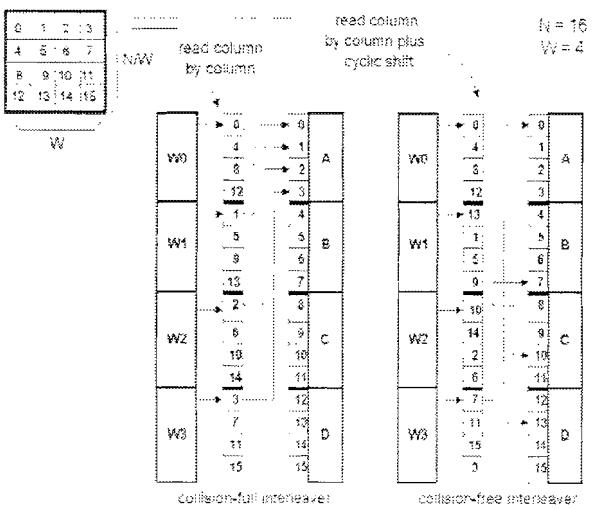

Figura 15. Collision-free interleaver generation from matrix **M**.

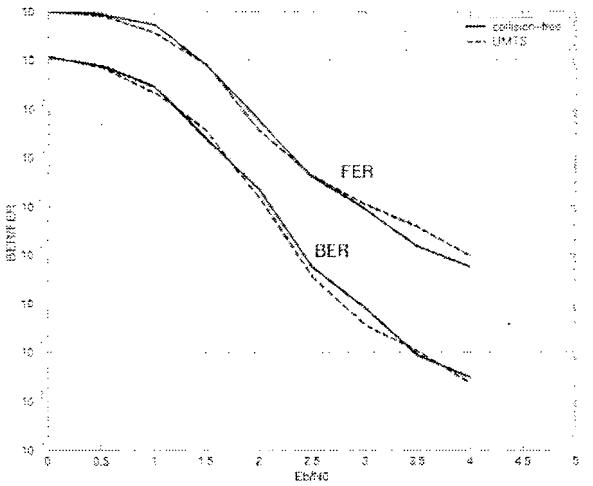

Figure 16. Comparison between collision-free interleaver and UMTS interleaver for N=432, k/n=1/3, BPSK.

Such a regular pattern has a poor BER performance (Figure 15 shows that the first position is not even permuted). Nevertheless, all elements in each row in matrix **M** belong to the same window, therefore 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 [18]. 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 the one in Figure 15, where N and W are multiples can be easily generated with on-the-fly address generators based on additions and modulo operations using as parameters the interleaver size and the window size. In our implementation, the existence of a large number of interleaver sizes (determined by the Hiperlan2 standard) with different relations between N and W led to a more careful study on the differents ways to implement the permutation table [17][34].

Figure 16 shows simulation results comparing the collision-free interleaver with an UMTS interleaver. Almost no difference can be noticed for the BER performance, while the collision-free slightly outperforms the UMTS for the FER (Frame Error Rate) performance.

5.2 AVOIDING TRELLIS TERMINATION

The UMTS standard adopted a rather complicated double-termination scheme that results in 6 tail positions appended to the information block to be encoded. Our interleaver intra-row and inter-row permutations took into account a

non-terminated scheme by minimizing the correlation between extrinsic values produced at the end of the block by the two decoding modules. In other words, mapping positions at the end of the block as far as possible during interleaving garantees that 'good quality' extrinsic values (those not affected by a non-terminated trellis) will be used in the decoding process to improve the quality of extrinsic values closer to the end.

In the SNR calculation for the simulation comparing the double terminated and the non-terminated scheme (Figure 17), a correction factor was applied in order to take into account the effect of introducing tail bits into the information to be transmitted. 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 17 and thus justified the adoption of no termination in our demonstrator.

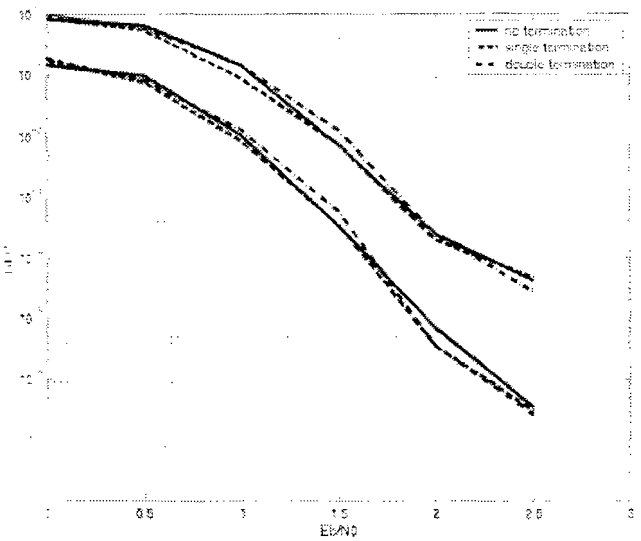

Figure 17. Comparison between no termination, single and double termination for N=432, k/n=1/3, BPSK.

6. PARALLEL TURBO DECODING SCHEDULE

In the previous section, we discussed 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 18, 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 solid 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 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 (Figure 19). 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 into the same position as their correspondent intrinsic values, thus guaranteeing collision-free deinterleaving if the interleaving is collision-free.

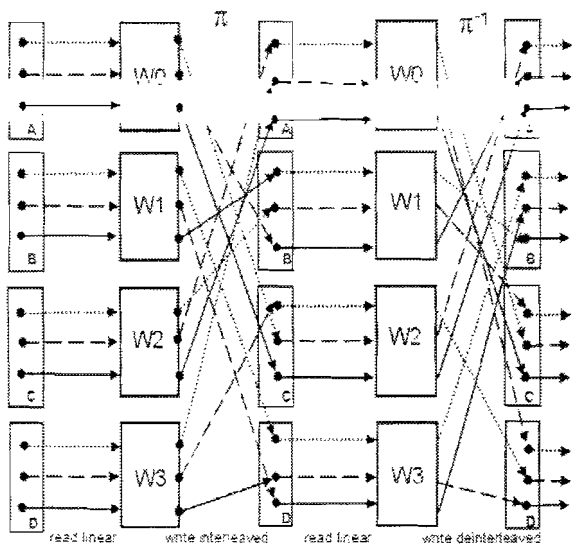

Figure 18. Typical parallel turbo decoding schedule.

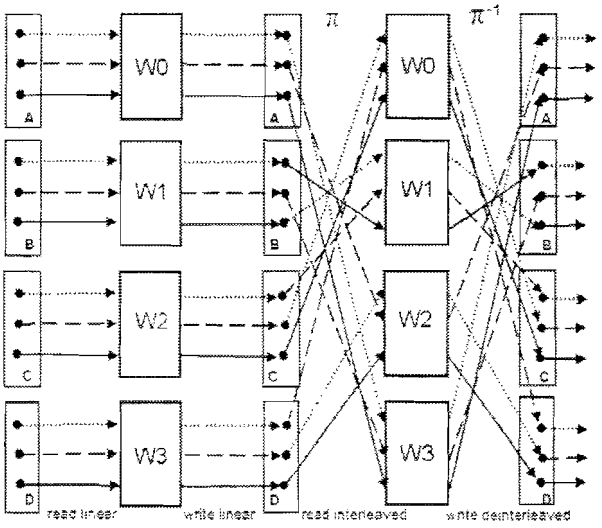

Figure 19. Optimized parallel turbo decoding schedule.

6. ADAPTIVE NUMBER OF ITERATIONS

The fixed-point implementation of the MAP allows using the amplitude of the log-likelihood ratio (LLR) (see equation 6) information measured at the output of each half-iteration in order to stop the turbo decoding iterations according to the measured quality of the process. With negligible loss in BER performance, this method can reduce the number of iterations and consequently the power consumption of the turbo decoder in more than 50 % when compared to the full number of iterations. The main idea of this criterion is to compare the magnitude of the LLR values produced by the MAP with a threshold. The larger the LLR's, the larger will be the probability of correct decoding. Therefore, the LLR threshold can be set in order to achieve a pre-defined frame error rate (FER) after a certain number of iterations (smaller or equal to a fixed maximum number set externally). This relation allows tuning the quality of service (QoS) in order to fit the user requirements with minimal energy.

7. RESULTS

We have implemented a turbo codec ASIC based on the proposed architecture [26]. VHDL code was written to model at RTL level the functionality of the encoder and turbo-decoder. This HDL code was verified and tested using test pattern and expected values directly generated with the OCAPI data flow model, which was used as design reference (see section 4). The gate level netlist was synthesized using the UMC 0.18µm technology. The critical path, located in the state metrics calculation datapath that cannot be pipelined, has been evaluated to be lower than 5.85 ns, enabling 170.9 MHz clock. Table 1 shows general features of the turbo codec core obtained with the Design Compiler™ tool from Synopsys™. Figure 20 shows the eletromiography of the processed ASIC, highlighting the parallel structure of datapath and memory elements. The BER performance of the processed turbo codec is shown in Figure 21. This analysis was made based on a board containing a AWGN channel, modulation and demodulation (soft-input generation from received symbols) built on a FPGA (Figure 22). The IC host and microprocessor interfaces are connected to a PC via a PCI bus. Data transfers in the board were implemented using FIFOs (also implemented on the FPGA). Host software determined the effective throughput and the actual BER performance shown. The complete evaluation setup description can be found in [27].

TABLE 1
SYNTHESIZED CORE MAIN CHARACTERISTICS

max clock frequency	**170.9 MHz**
max throughput	80.7 Mbps
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)

Table 2 shows measured results concerning coding gain, throughput and decoding energy. The energy consumption was measured by probing the current at the 1.8V core power supply. The coding gain at BER 10^{-8} ranges from 5.5 to 8.25 dB. It is possible to see that the decoding energy per block increases with the size of the block. On the other hand, the decoding energy per bit remains roughly constant, proving the modularity of the proposed architecture. The actual throughput is a bit smaller than the one evaluated using the layout before processing, due to physical limitations imposed by the board. The decoding energy consumption is shown in Figure 23 as a function of the channel signal to noise ratio, demonstrating the effect of the early stop criterion (see section 6). The decoding energy to reach a BER of 10^{-8} is in all cases lower than 10 nJ/bit.

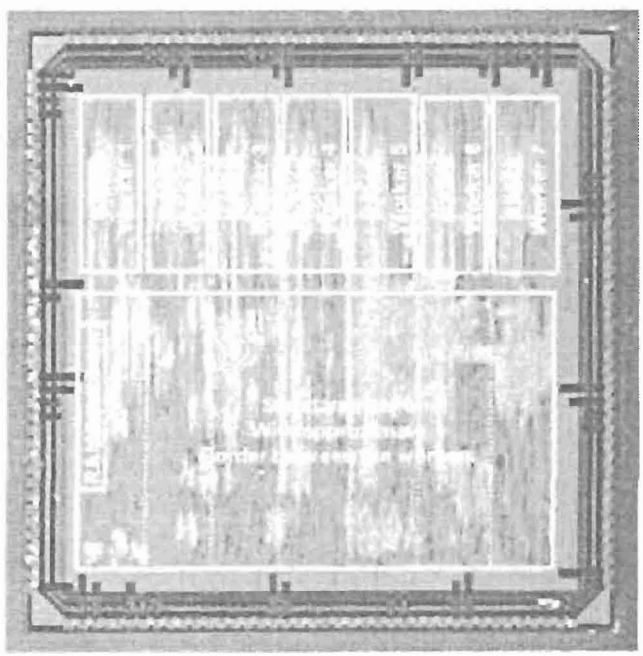

Figure 20. Eletromiography of the processed core showing the turbo codec main building blocks.

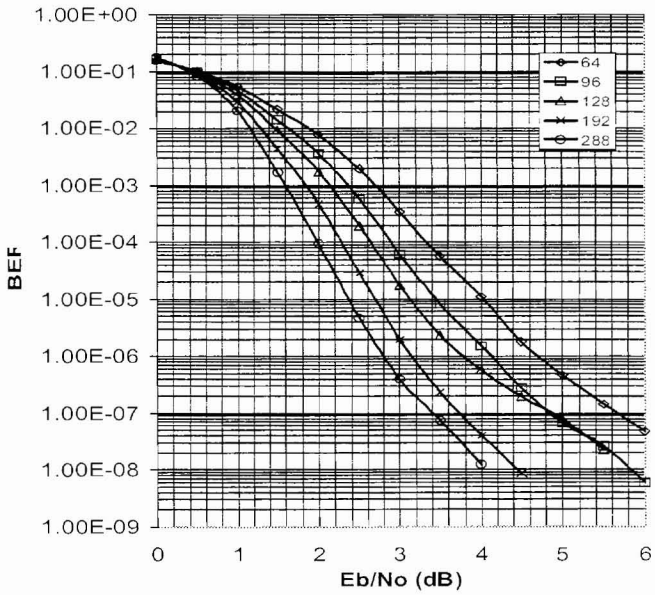

Figure 21. Measured BER performance.

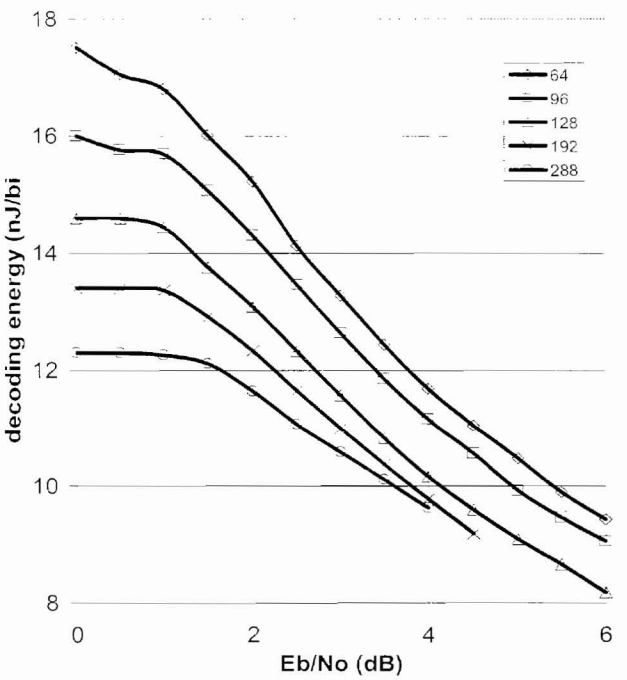

Figure 23. Measured core energy consumption

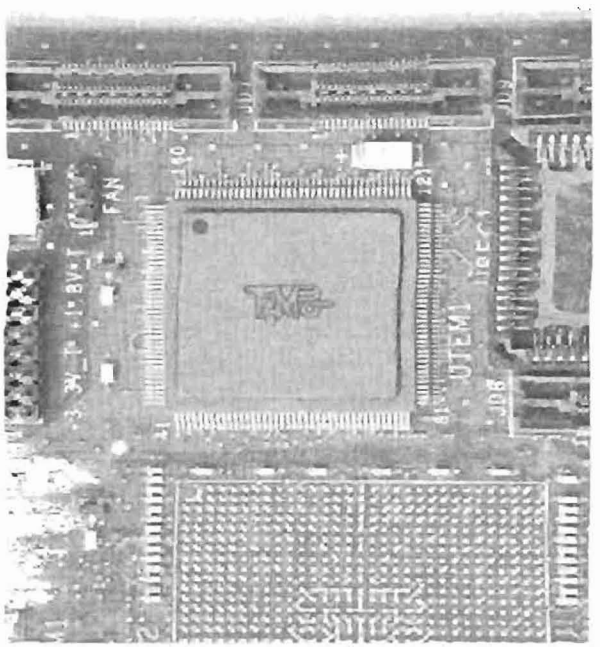

Figure 22 The turbo codec ASIC (nick-named 'T@MPO' Turbo @ Minimum Power) inserted on the test board.

9. 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, no termination and adaptive number of iterations. 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. The resulting architecture based on parallel windows can be easily tuned to different applications [31]. The presented turbo codec proves that it is possible to have turbo coding with low latency, low power and high-speed.

As future research work, two main branches could be tackled. Firstly a comparative analysis between LDPC codes [35] decoded iteratively and convolutional turbo codes regarding coding performance (a common research topic nowadays) and specially implementation cost (which has not been thoroughly studied). Secondly the development of a reconfigurable platform (using, for instance, FPGAs) that would be able to switch between different target throughputs, power consumption and block sizes on the fly, using completely the adaptive characteristics of the described architecture.

TABLE 2
ASIC MEASURED RESULTS

block size	net coding gain @ BER=10^{-6} (k/n = 1/3)	data throughput (mbps)	average number of iterations @ BER=10^{-6}
64	5.5	11.9	31
96	6.25	17.9	34
128	6	23.6	32
192	7.75	35.8	4
288	8.25	53.8	4.31
384	8	71.7	4.43
432	-	75.6	-

block size	decoding energy per block (μJ)	decoding energy per bit (nJ)
64	0.58	9
96	0.89	9.1
128	0.98	8.7
192	2.5	9.8
288	2.67	9.28
384	3.27	9.7
432	-	-

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] E. K. Hall, S. G. Wilson, "Design and Performance of Turbo Codes on Rayleigh Fading Channels", in Proceedings of CISS'96, Mar. 1996.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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

[20] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] B. Bougard, A. Giulietti, V. Derudder, J-W. Weijers, S. Dupont, L. Hollevoet, F. Catthoor, L. v. der Perre, H. de Man, R. Lauwereins, "A scalable 8.7 nJ/b 75.6 Mb/s Parallel Concatenated (Turbo-) CODEC", in IEEE International Solid State Circuit Conference (ISSCC), San Francisco, Feb. 2003.

[28] A.J. Viterbi, "Error bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", IEEE Transactions on Information Theory, v.13, p.260-269, 1967.

[29] R.W. Hamming, "Error Detecting and Error Correcting Codes", Bell Systems Technical Journal, v. 29, p. 147-160, 1950.

[30] R.C. Bose, D.K. Ray-Chauduri. "On a Class of Error Correcting Binary Group Codes". Inf. Control. v.3. p. 68-79. 1960.
[31] B. Bougard. A. Giulietti. L. Van der Perre. F. Catthoor. "A class of power efficient VLSI architectures for high-speed Turbo decoding". Proc. of IEEE Globecom. November 2002.
[32] J.Dielissen and J.Huisken. "State vector reduction for initialization of sliding windows MAP". 2nd International Symposium on Turbo Codes and Related Topics. Brest. France. September 2000.
[33] Battail. G. "A Conceptual Framework for Understanding Turbo Codes". IEEE Journal of Selected Areas in Communications. v.16. n.2. p. 245-254. 1998.
[34] Giulietti. A. "Uma arquitetura VLSI de baixa potência e alta velocidade para turbo decodificadores". PhD thesis. March 2003.
[35] R. G. Gallager. "Low-density parity-check codes." IRE Trans. Inform. Theory, vol. IT-8. pp. 21-28. Jan. 1962.

Alexandre Giulietti received the M. Sc. degree in Electrical Engineering from the Escola Politécnica da Universidade de São Paulo in 1998. He did his PhD at the Laboratory of Microelectronics of the University of São Paulo, Brazil, while working at IMEC, Leuven,Belgium, in the Wireless Communications group. He is currently with Genius Institute of Technology in Brazil. His research focus on VLSI implementation of communication systems, mainly turbo codes and speech recognition engines.

Marius Strum received his Electrical Engineering degree in 1971, his master and doctor in Electrical Engineering degrees in 1977 and 1983 respectively and the "Livre Docência" degree in 1990, all from São Paulo University - Brazil. In 1990 he became associate professor at the Electronics Engineering Department in the same University. He is responsible for the VLSI Design and CAD group of the Microelectronics Laboratory (LME) in the same University. He was a visiting scientist at the Interuniversity Microelectronics Center (IMEC) in Belgium (1986-1987). He has been responsible for several scientific projects in Brazil and two international cooperation agreements with European laboratories in France and Belgium. His research interests include architectures, design methodologies and CAD tools for digital IC and SOC. Dr. Strum has over 50 publications in conference proceedings and scientific magazines. He is also the author of chapters in three books about IC design and behavioral synthesis. He is one of the founders of the Brazilian Microelectronics Society.

Bruno Bougard received the M. Sc. degree in Electrical Engineering from the Polytechnic Institute of Mons, Belgium, in 2000. He carried out his M. Sc. thesis work with Alcatel ETCA, Charleroi, Belgium. He joined the Inter-university Microelectronics Center (IMEC), Leuven, Belgium in June 2000 as Research Engineer in the Wireless System Group. His current research focuses on channel coding for OFDM-based systems and on design methodologies for low power wireless communication system. He contributes as system architect to the design, the optimisation and the characterisation of a low power, high data-rate turbo decoder ASIC. Since 2002, he is also Research Assistant of the Fund for Scientific Research – Flanders (Belgium) and Ph.D. student at the E.E. Department of the Katholiek Universiteit Leuven, Belgium.

Liesbet Van der Perre received the M. Sc. degree in Electrical Engineering from the Katholieke Universiteit Leuven, Belgium, in 1992. She performed her M. Sc. thesis at the ENST in Paris, France. She received a Ph. D. degree from the Katholieke Universiteit Leuven in 1997 and jointed the Inter-university Microelectronics research center (IMEC), Leuven, Belgium. Currently, she is the director of IMEC's wireless program and part-time professor at the University of Antwerp, Belgium. Her work focuses on system design and digital modems for high-speed wireless communications. She was a system architect in IMEC's OFDM ASICs development, and the leader of the turbo coding team.