High-level Transformations using Canonical Dataflow Representation

Maciej Ciesielski, Jeremie Guillot, Daniel Gomez-Prado, Emmanuel Boutillon

Submission to the special issue on the 3rd International Symposium on Turbo Codes and Related Topics, Brest, 4-7 September 2003.

Annals of Telecommunications

Authors:

David Gnaedig, Ph.D. student at TurboConcept
Emmanuel Boutillon, Professor at University of South Britany
Michel Jézéquel, Professor at ENST Bretagne
Vincent, C. Gaudet, Assistant Professor at University of Alberta
P. Glenn Gulak, Professor at University of Toronto

Corresponding author: David Gnaedig
TurboConcept
Technopôle Brest-Iroise
115, rue Claude Chappe
29280 Plouzané
France
Tel. +33 2 98 05 63 83
david.gnaedig@turboconcept.com

Authors biographies:

David GNAEDIG received the electrical engineering degree from the ENST Paris in 2001. Since 2002 he has been a Ph.D. student with TurboConcept, the LESTER laboratory and ENST Bretagne. His research interests include turbo code design and adapted hardware architectures for high throughput iterative decoders.

Emmanuel BOUTILLON (PhD in 1995) was Associate Professor at ENST Paris from 1995 to 2000. He is currently Professor at the University of South Brittany, Lorient. His research interests are the interactions between algorithms and architectures in the field of wireless communications.

Michel JEZEQUEL received the engineering degree in electrical engineering from ENSEA Paris in 1982. In 1988, he joined ENST Bretagne where he is currently Professor and head of the Electronics Department. His main research interests are in circuit design for digital communications.

Vincent GAUDET received the B.Sc. degree from the University of Manitoba in 1995, and the M.A.Sc. and Ph.D. from the University of Toronto in 1997 and 2003, respectively. During 2002, he was a Research Associate at ENST Bretagne. He is currently Assistant Professor at the University of Alberta.

Glenn GULAK holds the L. Lau Chair in Electrical and Computer Engineering at the University of Toronto. He served as Program Chair for the 2001 IEEE International Solid-State Circuits Conference. His research interests include circuits, algorithms and VLSI architectures for digital communications and signal processing.

On Multiple Slice Turbo Codes[(1)(2)]

Les Turbo-Codes à Roulettes

David Gnaedig[1,2,3], Emmanuel Boutillon[2], Michel Jézéquel[3],
Vincent C. Gaudet[4] and P. Glenn Gulak[5]

[1] TurboConcept -Technopôle Brest-Iroise-115 rue Claude Chappe - 29280 PLOUZANE - FRANCE
[2] LESTER-Université de Bretagne Sud - BP 92116 - 56321 Lorient Cedex - FRANCE
[3]GET/ENST Bretagne/PRACOM- Unité CNRS 2658- BP 832- 29285 BREST Cedex FRANCE
[4] Dept. of ECE - University of Alberta - Edmonton, Alberta, Canada T6G 2V4
[5] Dept. of ECE-University of Toronto-10 King's College Road-Toronto- Ontario-CANADA M5S 3G4

E-mail: david.gnaedig@univ-ubs.fr, emmanuel.boutillon@univ-ubs.fr
michel.jezequel@enst-bretagne.fr, vgaudet@ece.ualberta.ca, gulak@eecg.toronto.edu

Abstract: *The main problem with the hardware implementation of turbo codes is the lack of parallelism in the MAP-based decoding algorithm. This paper proposes to overcome this problem by using a new family of turbo codes called Multiple Slice Turbo Codes. This family is based on two ideas: the encoding of each dimension with P independent tail-biting codes and a constrained interleaver structure that allows the parallel decoding of the P independent codewords in each dimension. The optimization of the interleaver is described. A high degree of parallelism is obtained with equivalent or better performance than the DVB-RCS turbo code. For very high throughput applications, the parallel architecture decreases both decoding latency and hardware complexity compared to the classical serial architecture, which requires memory duplication.*

Résumé : Le problème majeur dans l'implémentation matérielle d'un turbo-décodeur réside dans le manque de parallélisme des algorithmes de décodage MAP. Cet article propose un nouveau procédé de turbo-codage basé sur deux idées: le codage de chaque dimension par *P* codes convolutifs

[1] Patent N° 0204764 (France), April 16th 2002.
[2] This work has been partially funded by the region of Brittany (Virtual Turbo, grant PRIR n°A2C622).

récursifs circulaires indépendants et des contraintes sur la structure de l'entrelaceur qui permet de décoder en parallèle les *P* codes convolutifs dans chaque dimension. La construction des codes constituants et de l'entrelaceur est décrite et analysée. Un haut degré de parallélisme est obtenu avec des performances équivalentes ou meilleures que le turbo-code du standard DVB-RCS. L'architecture parallèle du décodeur permet de réduire à la fois la latence de décodage et la complexité du turbo-décodeur pour des applications à très hauts débits.

***Keywords*:** turbo code, interleaver, parallelism, tail-biting code, multiple slice turbo code

1 Introduction

The throughput of a turbo-decoder is limited by the recursion involved in the Soft Input Soft Output (SISO) processing of each dimension. Because of this recursion, the maximum throughput (the symbol rate) of a SISO decoder is thus equal to the clock rate of the decoder. In this case, the decoding of two half-iterations (a decoding iteration) needs at least $2.N$ symbol cycles, where N is the length of the frame. Thus, the pipe-lining of I distinct decoders allows a throughput of one frame every $2.N$ symbol cycles with a minimum overall latency of $2.I.N$ symbols cycles. This scheme is relatively inefficient in terms of hardware complexity since the extrinsic information memories are duplicated I times [1]. Some authors propose to parallelize the decoding process of the convolutional code for each turbo code dimension. For this purpose, they arbitrarily divide the frame to be decoded into P segments of equal size, in order to handle each segment using a MAP decoder. In these architectures, side effects at the ends of the segments are treated in a sub-optimal way by the use of a learning period [2], or by the pointer method which gives the initial states of each segment between two iterations [3]. However, these approaches perform the decoding process

of one dimension without tackling the problem of memory conflicts that can arise from interleaving while decoding the second dimension.

In this paper, we propose a new family of turbo code where both constituent codes are constructed with P independent Circular Recursive Systematic Convolutional codes [4] (CRSC, also called tail-biting code), called "slices". The process of decoding P slices in parallel is made possible in the two code dimensions by using an adapted interleaver structure. It can be noted that another similar parallel interleaver has been independently studied by Dobkin *et al.*, but associated with a conventional turbo code [5]. The interleaver structure is constructed in a way similar to the one presented in this paper, but optimization of the permutation is not described.

This paper is divided into seven sections. In section 2, the global encoding strategy for Multiple Slice Turbo Codes is described. In section 3, the structure of the parallel interleaver is presented, which leads to a design based on two concatenated permutations. These permutations are designed and their influence on the performance are discussed in section 4. Finally, the complexity and latency reduction associated with the proposed scheme is evaluated in section 5 for several data rates and information block lengths. The performance is compared to a conventional turbo code in section 6 for various block lengths and code rates. Section 7 concludes and summarizes the performance results and complexity reductions.

2 Multiple Slice Turbo codes

Multiple Slice Turbo Codes are constructed as follows. An information frame of N m-binary symbols is divided into P blocks (called "slices") of M symbols, where $N = M \cdot P$. The resulting turbo code is denoted (N, M, P). As with a conventional convolutional two-dimensional turbo code, the coding process is first performed in the natural order to produce the coded symbols of the first

dimension. Each slice is encoded independently with a CRSC code. The information frame is then permuted by an N symbol interleaver. The permuted frame is again divided into P slices of size M and each of them is encoded independently with a CRSC code to produce the coded symbols of the second dimension. Puncturing is applied to generate the desired code rate.

The interleaver is constructed jointly with the memory organization of the decoder to allow the parallel decoding of the P slices. The architecture contains P Soft-In Soft-Out decoders (SISO units) working in parallel and P memory banks MB_0, MB_1,…, MB_{P-1} allowing parallel memory accesses for the P SISO units. In other words, at each symbol cycle k, the interleaver structure allows the P decoders to read and write the P necessary data symbols from/to the P Memory Banks MB_0, MB_1,…, MB_{P-1} without any conflict. With the solution described in the present paper, the degree of parallelism can be chosen according to the requirements of the application. The advantages of parallelism will be discussed in section 5.

The next section presents the interleaver construction, ensuring parallelism constraints while maintaining good performance.

3 Interleaver construction

The interleaver design is based on the approach proposed in [6]: The interleaver structure is mapped onto a hardware architecture allowing a parallel decoding process.

3.1 Interleaver structure

Figure 1 presents the interleaver structure used to construct a multiple slice turbo code constrained by decoding parallelism.

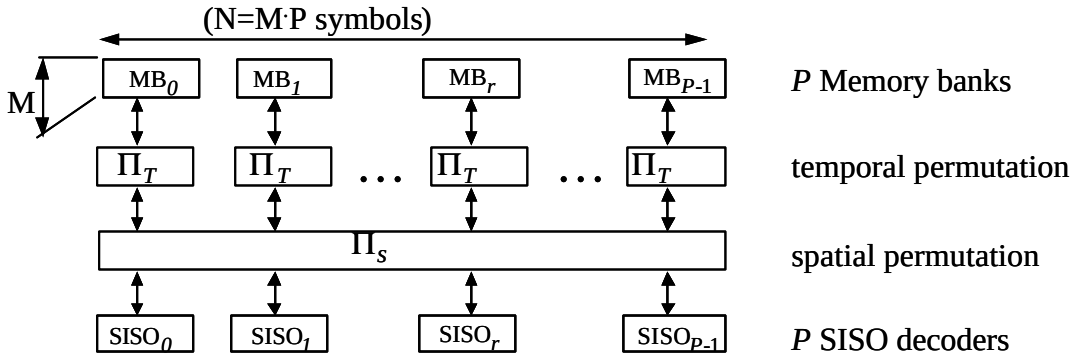

Figure 1: Interleaver structure for the (*N*,*M*,*P*) code

Figure 1: Structure de l'entrelaceur pour le code (*N*, *M*, *P*)

Let l and k denote the indices of the symbols in the natural and interleaved order, respectively. The coding process is performed in the natural order on independent consecutive blocks of M symbols. The symbol with index l is used in slice $\lfloor l/M \rfloor$ at temporal index time $l \bmod M$, where $\lfloor . \rfloor$ denotes the integral part function. Likewise, in the interleaved order, the symbol with index k is used in slice $r = \lfloor k/M \rfloor$ at temporal index $t = k \bmod M$. Note that $k = M \cdot r + t$, where $r \in \{0..P-1\}$ and $t \in \{0..M-1\}$. For each symbol with index k in the interleaved order, the permutation Π associates a corresponding symbol in the natural order with index $l = \Pi(k) = \Pi(t,r)$. The interleaver function can be split into two levels: a spatial permutation $\Pi_S(t,r)$ (ranging from 0 to P-1) and a temporal permutation $\Pi_T(t,r)$ (ranging from 0 to M-1), as defined in (1) and described in Figure 1.

$$\Pi(k) = \Pi(t,r) = \Pi_S(t,r) \cdot M + \Pi_T(t,r) \qquad (1)$$

The symbol at index k in the interleaved order is read from the memory bank $\Pi_S(t,r)$ at address $\Pi_T(t,r)$. While decoding the first dimension of the code, the frame is processed in the natural order. The spatial and temporal permutations are then simply replaced by *Identity* functions.

In order to simplify the hardware implementation, the same temporal permutation is chosen for all memory banks. Thus, $\Pi_T(t,r) = \Pi_T(t)$ depends only on the temporal index t. This solution has the advantage of requiring a single computation of the address to read P data symbols from the P memory banks. Moreover, the P memory banks can be merged into a single memory, which further reduces the area.

The spatial permutation allows the P data read out to be transferred to the P SISO units (denoted SISO in Figure 1). Decoder r receives the data from memory bank $\Pi_S(t,r)$ at instant t. For each fixed t, function $\Pi_S(t,r)$ is then a bijection from the decoder index $r \in \{0..P-1\}$ to the memory banks $\{0..P-1\}$.

Furthermore, to maximize the shuffling between the natural and the interleaved order, we constrain function $\Pi_S(t,r)$ such that every P consecutive symbols of a slice of the interleaved order comes from P distinct slices of the natural order, *i.e.* P distinct memory banks. Thus, for a given r, function $\Pi_S(t,r)$ is bijective and P-periodic. The bijectivity means that $\Pi_S(t,r)$ is a bijection from the temporal index $t \in \{0..P-1\}$ to the set $\{0..P-1\}$ of memory bank indices. The P-periodicity on the temporal index means that for $\forall t, \forall q$ satisfying $t + q \cdot P < M$, one obtains $\Pi_S(t + q \cdot P, r) = \Pi_S(t,r)$.

In the rest of the paper, bijective P-periodic functions are now considered for the spatial permutation. Let r be the division of the number of symbols in a slice by the number P of slices: $r = M / P$. In what follows we suppose, without loss of generality, that $r \geq 2$ is an integer. This implies that two distinct slices with index j in the first dimension and index i in the second dimension share exactly r symbols with temporal indices $t_{i,j} = \{T(i,j) + q \cdot P\}_{q=0..r-1}$ in the interleaved order, where $T(i,j) \in \{0,...,P-1\}$ is the temporal index that verifies $\Pi_S(T(i,j),i) = j$.

3.2 A simple example

Let us construct a simple (N, M, P) = (18,6,3) code to clarify the interleaver construction. Let the temporal permutation be $\Pi_T(t) = \{1,4,3,2,5,0\}$ (*i.e.* $\Pi_T(0) = 1$, $\Pi_T(1) = 4$, ...) and let the spatial permutation be a circular shift of amplitude $A(t \bmod 3)$, *i.e.* the slice of index r is associated with the memory bank of index $\Pi_S(t,r) = (A(t \bmod 3) + r) \bmod 3$, with $A(t \bmod 3) = \{0,2,1\}$. The spatial permutation is then bijective and 3-periodic.

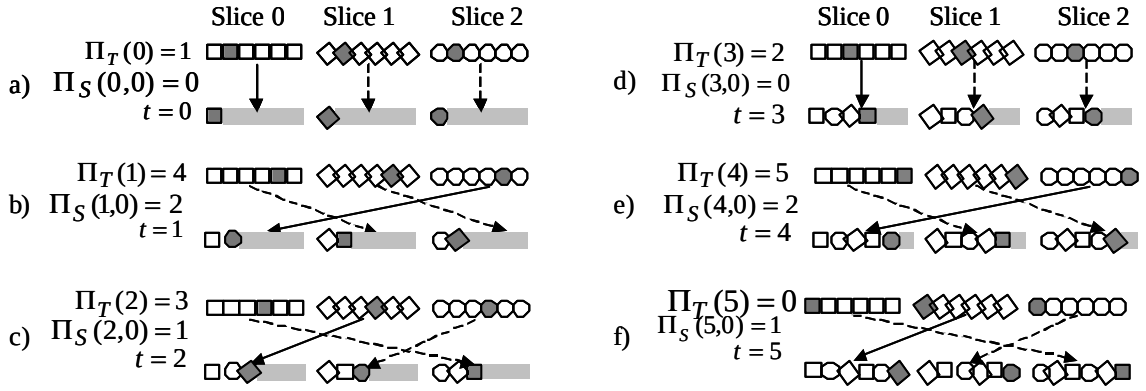

Figure 2 : A basic example of a (18,6,3) code with $\Pi_T(t) = \{1,4,3,2,5,0\}$ and $A(t \bmod 3) = \{0,2,1\}$.

Figure 2 : Un exemple simple d'un code (18,6,3) avec $\Pi_T(t) = \{1,4,3,2,5,0\}$ et $A(t \bmod 3) = \{0,2,1\}$.

The interleaver is illustrated in Figure 2, which shows the permutations for the 6 temporal indices t = 0 (2.a), t = 1 (2.b), t = 2 (2.c), t = 3 (2.d), t = 4 (2.d) and t = 5 (2.c). The 18 symbols in the natural order are separated into 3 slices of 6 symbols corresponding to the first dimension. In the second dimension, at temporal index t, symbols $\Pi_T(t)$ are selected from the 3 slices of the first dimension, and then permuted by the spatial permutation $\Pi_S(t,r)$. For example, at temporal index $t=1$ (b), symbols at index $\Pi_T(1)=4$ are selected. Then, they are shifted to the left with an amplitude $A(1 \bmod 3) = 2$. Thus, symbols 4 from slices 0, 1 and 2 of the first dimension go to slices 1, 2 and 0 of the second dimension respectively.

4 Interleaver design

The design of the interleaver aims to fulfill two performance criteria. The first is to obtain a good minimum distance for the asymptotic performance of the code at high signal to noise ratios (SNR). In fact, a high minimum distance is needed to lower the "error floor" induced by the presence of low weight codewords [7][8]. The second is to achieve the convergence of the code at low SNRs. The convergence is degraded by the correlation between the extrinsic information inputs, caused by the presence of short cycles in the interleaver [9]. After an analysis of the short cycles and low-weight codewords, we describe the design process of the interleaver based on three steps: optimization of the temporal permutation, optimization of the spatial permutation, and then, introduction of a local disorder in the temporal permutation.

4.1 Analysis of the cycles in an interleaver and related low-weight codewords

Definition 1: For an *m*-binary convolutional code, a Return To Zero (RTZ) sequence is defined as an ordered finite list of symbols that makes the encoder diverge from state 0 and then return to state 0. The number of symbols after the first symbol in the list defines the length of the sequence.

An RTZ input sequence produces a finite number of parity bits. The weight of an RTZ sequence is then equal to the number of 1s in the input bits and in the corresponding parity bits. These RTZ sequences represent the low weight error sequences of a convolutional code. As a first approximation, we will consider that their weight grows linearly with their length. For an 8-state recursive systematic binary convolutional code (rate 1/2) with memory *n* and with generator polynomials 15 (feedback) and 13 (redundancy), the basic RTZ sequence is the sequence of length 7: 10000001. The weight of its parity bits is equal to 4. For an 8-state recursive systematic duo-binary convolutional code (rate 2/3), it was shown in [10] that there are other basic RTZ sequences of different lengths. For the duo-binary turbo-code in the DVB-RCS standard [11][12], the input couple of the convolutional encoder can take 4 different values denoted **0** = (0,0), **1** = (0,1), **2** = (1,0), **3** = (1,1). For this code, the primitive RTZ sequences are 13, 201 and 30002 of weight before puncturing 4, 5 and 7, respectively (see [10] for more details about duo-binary turbo codes).

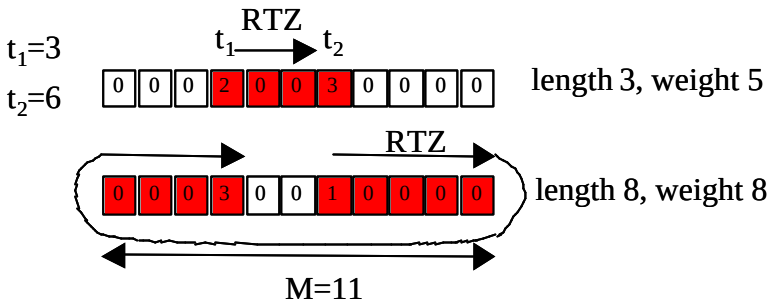

Figure 3 : Example of RTZ sequences for two given symbols with the tail-biting duo-binary code defined in [11] with a slice M = 11.

Figure 3: Exemple de séquences RTZ pour deux symboles avec un code duo-binaire circulaire défini dans [11] avec M = 11.

Definition 2: The norm $|x|_M$ is defined as $|x|_M = \min(x \bmod M, -x \bmod M)$.

Definition 3: The distance $d_M(t_1, t_2)$ between two symbols k_1 and k_2 of a same slice, and whose temporal indices are denoted t_1 and t_2 respectively, is defined by the length of the smallest path $|t_1 - t_2|_M = \min\left(|t_1 - t_2|, M - |t_1 - t_2|\right)$ between the two symbols (see Figure 3).

Because of the tail-biting technique, two RTZ sequences are possible for two given symbols of a slice (see Figure 3). In a first approximation, the RTZ sequence with the smallest length will be considered. In fact, this RTZ sequence may have the smallest Hamming weight and, thus, penalizes the minimum distance the most. Hence, for two symbols in a slice, the RTZ sequence of smallest weight (if there are any) is proportional to the distance (definition 3) between the two symbols.

In the general case, two symbols located in two distinct slices cannot generate a RTZ sequence: their distance will be assumed to be infinity. Thus, the definition of distance is generalized by:

Definition 4: The distance $\|k_1 - k_2\|_M$ between two symbols k_1 and k_2 of the frame is defined by:

$\|k_1 - k_2\|_M = |k_1 - k_2|_M$ if symbols k_1 and k_2 are in the same slice (see Figure 4.b).

$\|k_1 - k_2\|_M =$ infinity otherwise (see Figure 4.a).

In the following, for the sake of simplicity we shall not specify whether we refer to definition 3 or definition 4. The distance $\| \|_M$ is used for the distance between two symbols, whereas $| |_M$ is used for the distance between the temporal indices of the symbols in a slice. Moreover, $\|k_1 - k_2\|_M = |t_1 - t_2|_M$ if symbols k_1 and k_2 are both in the same slice with temporal indices t_1 and t_2.

Definition 5: A primary cycle is defined as a set of two symbols that are in a same slice in both the natural and interleaved order, as shown in Figure 4.b.

If the two symbols correspond to an RTZ sequence in the two dimensions, then the primary cycle is also called a Primary Error Pattern (PEP). Thus, it is natural to analyze the impact on the performance of a primary cycle by its spread defined as the sum of the distance between the symbols in the two dimensions:

$$S(k_1, k_2) = \|k_1 - k_2\|_M + \|\Pi(k_1) - \Pi(k_2)\|_M \qquad (2)$$

Note that a similar definition of spread has been proposed independently in [13]. The spread between two symbols that belong to a primary cycle is proportional to the minimum weight of a potential RTZ sequence. Thus primary error patterns are strongly related to primary cycles: short

primary cycles may lead to low weight codewords, while long primary cycles lead to higher weight codewords. Moreover, $S(k_1, k_2)$ is also related to the correlation between the extrinsic information inputs associated with symbols k_1 and k_2. The higher the spread, the lower the correlation. Thus, the minimum spread S among all possible primary cycles, $S = \min_{k_1,k_2}[S(k_1, k_2)]$, should be reasonably high in order to avoid both low-weight PEPs and high correlation.

Definition 6: A secondary cycle is defined as a set of 4 symbols that constitute two couples in the two dimensions of the code as shown in Figure 4.c, d.

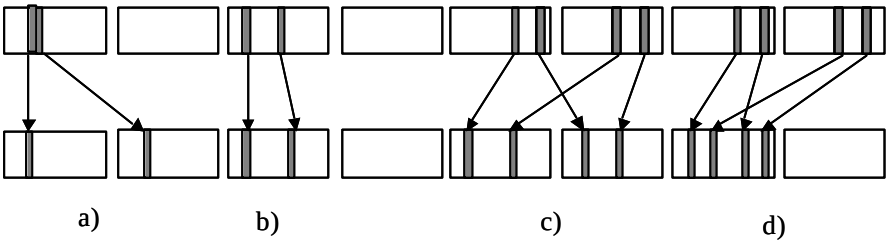

Figure 4: Primary and secondary cycles:

a) no cycle, b) primary cycle, c) external secondary cycle, d) internal secondary cycle.

Figure 4: Cycles primaires et secondaires:

a) pas de cycle, b) cycle primaire, c) cycle secondaire externe, d) cycle secondaire interne.

Definition 7: A secondary cycle is said to be internal when the four symbols are in the same slice in at least one of the two dimensions (Figure 4.d). It is said to be external in the other case (Figure 4.c).

Like for the primary cycle, we can define a secondary error pattern (SEP) as a secondary cycle whose symbols form two RTZ sequences in the two code dimensions. To characterize secondary cycles, we extend the notion of spread using the summary distance [12]. For each secondary cycle,

there are 4 couples of symbols that belong to four slices or less: slices i_1 and i_2 in the interleaved order, slices j_1 and j_2 in the natural order. Their temporal indices are denoted $\left(u_{i_1}^1, u_{i_1}^2\right)$ and $\left(u_{i_2}^1, u_{i_2}^2\right)$ in the slices of the interleaved order, and $\left(v_{j_1}^1, v_{j_1}^2\right), \left(v_{j_2}^1, v_{j_2}^2\right)$ in the slices of the natural order. The summary distance of a secondary cycle is then defined as the sum of the 4 pair-wise lengths between the temporal indices of the 4 couples:

$$d_{sum} = \left|u_{i_1}^1 - u_{i_1}^2\right|_M + \left|u_{i_2}^1 - u_{i_2}^2\right|_M + \left|v_{j_1}^1 - v_{j_1}^2\right|_M + \left|v_{j_2}^1 - v_{j_2}^2\right|_M \quad (3)$$

Definition 8: A secondary cycle is said to be "rectangular" if $\left|u_{i_1}^1 - u_{i_1}^2\right|_M = \left|u_{i_2}^1 - u_{i_2}^2\right|_M$ and $\left|v_{j_1}^1 - v_{j_1}^2\right|_M = \left|v_{j_2}^1 - v_{j_2}^2\right|_M$.

For a rectangular secondary cycle, the distances between the symbols of the two couples of the first dimension are equal, as presented in Figure 5. It is the same for the couples of the second dimension.

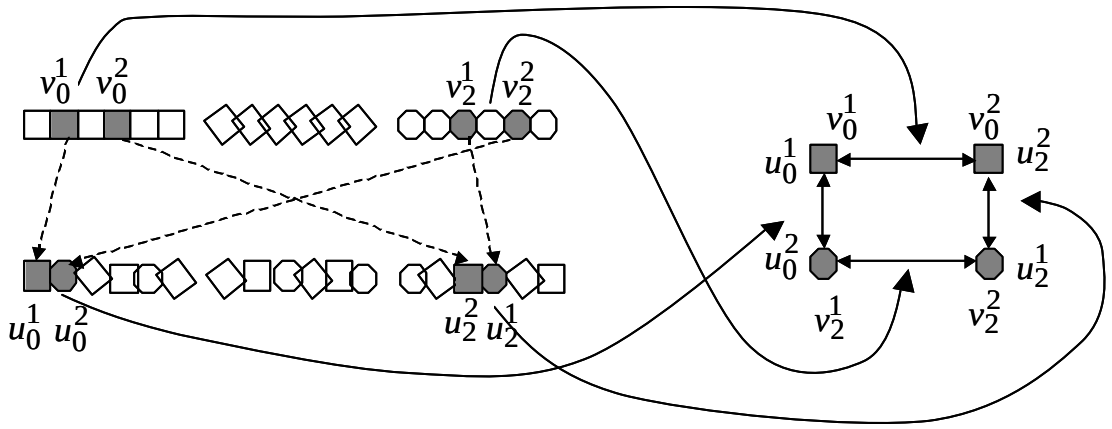

Figure 5: External rectangular secondary cycle representation on the example of section 3.2 with (i_1, j_1, i_2, j_2) = (0,0,2,2).

Figure 5: Cycle secondaire externe rectangulaire sur l'exemple de la section 3.2 avec (i_1, j_1, i_2, j_2) = (0,0,2,2).

The optimization of the interleaver aims to find appropriate temporal and spatial permutations. The design of these permutations is performed by analyzing primary cycles and patterns characterized by their spread, and secondary cycles and patterns characterized by their summary distance.

4.2 Temporal permutation

The aim of the temporal permutation is to maximize the minimum spread in order to eliminate low weight PEPs and to reduce correlation between the extrinsic information inputs.

As a first step, a simple regular temporal permutation is chosen in order to simplify the study of the interaction between the temporal and the spatial permutations. In this sub-section, it is optimized by analyzing primary cycles only. Next, in sub-section 4.4 more irregularity will be introduced in the temporal permutation to improve the performance furthermore. The temporal permutation used in this study is given by:

$$\Pi_T(t) = a \cdot t \bmod M, \tag{4}$$

where a is an integer, and where a and M are mutually prime. This choice is justified by the hardware simplicity of the temporal generator and the good performance that can be obtained. The only parameter to be optimized is then a.

Since the spatial permutation is P-periodic and bijective, two symbols of a same slice having a distance which is not a multiple of P in the second dimension are not in the same slice in the first dimension. Their spread is then infinite. Reciprocally, two symbols (k_1, k_2) of the second dimension

that belong to a primary cycle have a distance verifying $\|k_1 - k_2\|_M = |q \cdot P|_M$, where q =1,..., r - 1. Their temporal indices are denoted t_1 and t_2 respectively, and thus $|t_1 - t_2|_M = |q \cdot P|_M$. With the temporal permutation (4), we obtain:

$$\|\Pi(k_1) - \Pi(k_2)\|_M = |a \cdot t_1 - a \cdot t_2|_M = |a \cdot q \cdot P|_M \quad (5)$$

Thus, there are r possible values for the spread between two symbols. Thus, the overall minimum spread of the interleaver is given by:

$$S = \min_{q=1,\ldots,\, r-1} \left(|q \cdot P|_M + |a \cdot q \cdot P|_M \right) \quad (6)$$

The parameter a is then chosen to maximize the spread S. Since $a < M$, an exhaustive search is performed in order to obtain the highest spread.

The temporal permutation also has an influence on the internal secondary cycles and on the corresponding SEPs. These cycles will be studied in section 4.4.

4.3 Spatial permutation

The choice of the spatial permutation is made by analyzing its influence on external secondary cycles and their associated error patterns. Hence in the following, we will implicitly refer to external cycles when speaking about secondary cycles. Note that the summary distance of secondary cycles does not increase with high spread, and therefore the weight of SEPs does not improve with high spread. For the temporal permutation given in (4), the spatial permutation aims to maximize the summary distance of the secondary cycles.

There is a whole range of possible spatial permutations that are bijective and P-periodic. To simplify hardware implementation, the spatial permutation is chosen to be a circular shift of amplitude $A(\bar{t})$, whose equation is given by:

$$\Pi_S(t,r) = \left(A(\bar{t}) + r\right) \bmod P, \tag{7}$$

where $\bar{t} = t \bmod P$ and A is a bijection of variable $\bar{t} \in \{0,...,P-1\}$ to $\{0,...,P-1\}$.

In order to obtain a graphical representation of the secondary cycles, the spatial permutation is represented by a square matrix T of size P. For this matrix, the i^{th} row is associated with the i^{th} slice of the second dimension and the j^{th} column is associated with the j^{th} slice of the first dimension. As defined at the end of section 3.1, $T(i,j)$ represents the temporal index verifying $\Pi_S(T(i,j),i) = j$, *i.e.*, it represents the set $t_{i,j}$ of the r symbols of the i^{th} slice of the interleaved order that also belong to the j^{th} slice of the natural order.

Let us now design the matrix T corresponding to equation (7). For fixed $\bar{t}$, the r symbols in slice r = 0 of the second dimension correspond to symbols from slice $A(\bar{t})$ of the first dimension. It is expressed mathematically as $T(0, A(\bar{t})) = \bar{t}$. Moreover, for fixed $\bar{t}$, $\Pi_S(\bar{t}, s)$ is a rotation. This means that the r symbols in slice r = 1 correspond to symbols from slice $(A(\bar{t}) + 1) \bmod P$, and thus, $T(1, (A(\bar{t}) + 1) \bmod P) = \bar{t}$. More generally, for i = 0..P-1, $T(i, (A(\bar{t}) + i) \bmod P) = \bar{t}$. Hence the P sets t_i of r temporal indices form a circular diagonal $d_{\bar{t}}$ in the matrix T (see Figure 6), which is then a Toeplitz matrix. The matrix T of example of section 3.2, where $A(\bar{t}) = \{0,2,1\}$ is given by (8):

$$T = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{bmatrix}. \qquad (8)$$

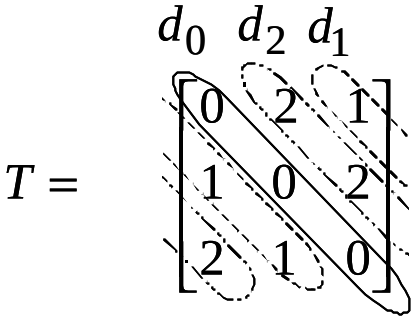

Figure 6: Representation of the circular diagonals d_0, d_1, d_2 for the matrix T corresponding to

$$A(\bar{t}) = \{0,2,1\}.$$

Figure 6 : Représentation des diagonales circulaires d_0, d_1, d_2 pour la matrice T correspondant à

$$A(\bar{t}) = \{0,2,1\}.$$

For $\bar{t} = 0$, the symbols in slices 0, 1 and 2 of the second dimension correspond to symbols in slices 0, 1, 2 of the first dimension, respectively (this diagonal d_0 corresponds to $A(\bar{t}) = 0$). Then for $\bar{t} = 1$, the symbols in slices 0, 1 and 2 of the second dimension correspond to symbols in slices 2, 0, 1 of the first dimension (this diagonal d_1 corresponds to $A(\bar{t}) = 2$). Finally, for $\bar{t} = 2$, the symbols in slices 0,1 and 2 of the second dimension correspond to symbols in slices 1, 2 and 0 of the first dimension (this diagonal d_2 corresponds to $A(\bar{t}) = 1$).

A secondary cycle is represented in the spatial permutation matrix T by a rectangle (i_1, j_1, i_2, j_2) as shown in Figure 7. Note that $T(i,j)$ represents the set $t_{i,j}$ of r distinct symbols. Thus the rectangle in Figure 7 represents r^4 (r to the fourth) distinct secondary cycles. For the sake of simplicity, we

consider only one of these secondary cycles, which does not change the results of our study. The temporal indices in the second dimension are denoted $t_{i_1 j_1}$, $t_{i_1 j_2}$ for the first slice and $t_{i_2 j_1}$, $t_{i_2 j_2}$ for the second slice. Let $\Pi_T(t_{i_1 j_1})$, $\Pi_T(t_{i_1 j_2})$, $\Pi_T(t_{i_2 j_1})$ and $\Pi_T(t_{i_2 j_2})$ be their corresponding indices in the slices of the first dimension. The summary distance of this secondary cycle is given by

$$d_{sum} = \left|t_{i_1 j_1} - t_{i_1 j_2}\right|_M + \left|t_{i_2 j_1} - t_{i_2 j_2}\right|_M + \left|\Pi_T(t_{i_1 j_1}) - \Pi_T(t_{i_2 j_1})\right|_M + \left|\Pi_T(t_{i_1 j_2}) - \Pi_T(t_{i_2 j_2})\right|_M \quad (9)$$

With the example of Figure 7, $t_{i_1 j_1} = 2$, $t_{i_1 j_2} = 1$, $t_{i_2 j_1} = 0$, $t_{i_2 j_2} = 2$ and $\Pi_T(t_{i_1 j_1}) = 4$, $\Pi_T(t_{i_1 j_2}) = 3$, $\Pi_T(t_{i_2 j_1}) = 1$, $\Pi_T(t_{i_2 j_2}) = 3$. The summary distance of this cycle is $d_{sum} = 6$.

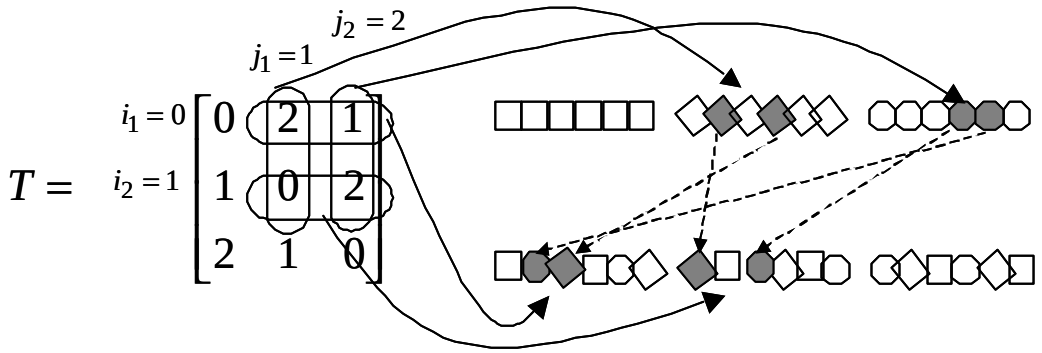

Figure 7: External secondary cycle representation on the spatial permutation: (i_1, j_1, i_2, j_2) = (0, 1, 1, 2) and $\Pi_T(t) = \{1,4,3,2,5,0\}$.

Figure 7 : Représentation d'un cycle secondaire externe sur la permutation spatiale : (i_1, j_1, i_2, j_2) = (0, 1, 1, 2) et $\Pi_T(t) = \{1,4,3,2,5,0\}$.

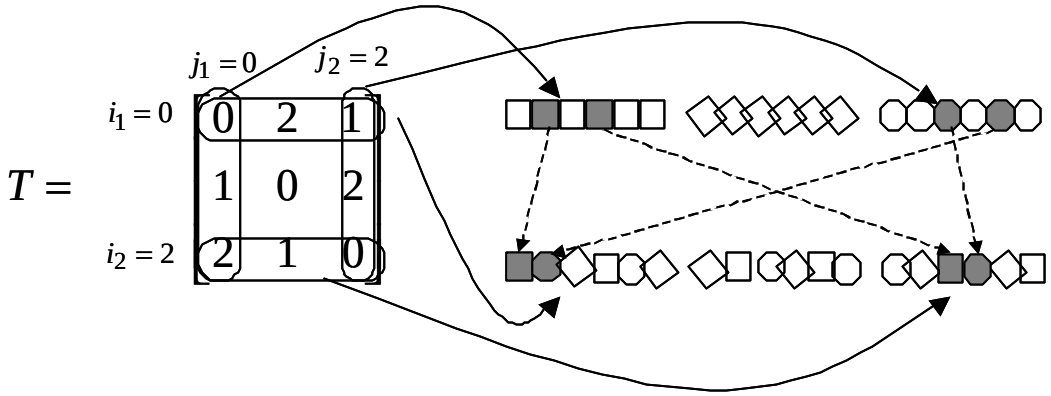

Figure 8: External rectangular secondary cycle representation on the spatial permutation (i_1, j_1, i_2, j_2) = (0,0,2,2), and $\Pi_T(t) = \{1,4,3,2,5,0\}$.

Figure 8 : Représentation d'un cycle secondaire rectangulaire sur la permutation spatiale : (i_1, j_1, i_2, j_2) = (0,0,2,2), et $\Pi_T(t) = \{1,4,3,2,5,0\}$.

Theorem 1: For temporal permutation (4) and spatial permutation (7), if an external secondary cycle verifies $\left|t_{i_1 j_1} - t_{i_1 j_2}\right|_M = \left|t_{i_2 j_1} - t_{i_2 j_2}\right|_M = \Delta t$ then it is rectangular and its summary distance is given by $S_d = 2 \cdot \left(\Delta t + \left|a \cdot \Delta t\right|_M\right)$.

In fact, using the temporal permutation (4), $\left|t_{i_1 j_1} - t_{i_1 j_2}\right|_M = \left|t_{i_2 j_1} - t_{i_2 j_2}\right|_M = \Delta t$ implies $\left|\Pi_T(t_{i_1 j_1}) - \Pi_T(t_{i_2 j_1})\right|_M = \left|\Pi_T(t_{i_1 j_2}) - \Pi_T(t_{i_2 j_2})\right|_M = \left|a \cdot \Delta t\right|_M$.

Note that a summary distance S_d of an external rectangular secondary cycle is twice the spread between two symbols distant from Δt. But unlike to equation (6), Δt is no more necessarily a multiple of P. Hence, an inappropriate choice of the spatial permutation may lead to rectangular secondary cycles with low summary distances and to low SEPs. . For example, with the regular permutation $A(\bar{t}) = \{0,1,...,P-1\}$, all secondary cycles are rectangular. This case is the worst since

any value of Δt leads to a rectangular secondary cycle. The bijection A should be chosen as "irregular" as possible in order to minimize the number of rectangular secondary cycles. Since an exhaustive search of the best possible permutation is computationally too high, empirical rules have to be derived. Usually, the irregularity of an interleaver is characterized by its dispersion [15]. We introduce a new appropriate definition of dispersion (10).

Let us first define the displacement vector $\Delta = (\Delta\bar{t}, \Delta A) = \left(\left| \bar{t}_2 - \bar{t}_1 \right|_P, \left| A(\bar{t}_2) - A(\bar{t}_1) \right|_P \right)$ for any couple of diagonals $d_{\bar{t}_1}$ and $d_{\bar{t}_2}$. Due to the P-periodicity of the spatial permutation, $\Delta\bar{t}$ is defined with distance $\min(\left| \bar{t}_2 - \bar{t}_1 \right|, P - \left| \bar{t}_2 - \bar{t}_1 \right|)$. Similarly, due to the P-periodicity of the spatial permutation, ΔA is also defined with the $\left| \ \right|_P$ distance. Indeed, for each distance, there are two possible configurations. For example, in Figure 6, on the first line, diagonals d_0 and d_1 are separated by two positions but by only 3-2 = 1 position on the second line. For the sake of unicity of the displacement vector we consider only the one with smallest distance $\left| \ \right|_P$. Define the set of displacement vectors as:

$$D(A) = \left\{ (\Delta\bar{t}, \Delta A), \Delta\bar{t} = \left| \bar{t}_2 - \bar{t}_1 \right|_P, \Delta A = \left| A(\bar{t}_2) - A(\bar{t}_1) \right|_P, (\bar{t}_1, \bar{t}_2) \in \{0,...,P-1\}^2 \right\} \qquad (10)$$

The dispersion of the spatial permutation is then defined as $D = \left| D(A) \right|$ the cardinal of the set of displacement vectors. When $D(A)$ contains two displacement vectors $\Delta_1 = \Delta_2$, this means that there exists a secondary cycle which verifies $\left| t_{11} - t_{12} \right|_M = \left| t_{21} - t_{22} \right|_M$. From Theorem 1, this secondary cycle is "rectangular". Hence, the higher the dispersion is, the fewer "rectangular" secondary cycles exist. The choice of the bijection A is made by maximizing the dispersion D. Its

maximization reduces the number of rectangular secondary cycles, increases the minimum summary distance among all secondary cycles, and introduces randomness in the interleaver.

We have seen the influence of the spatial permutation on secondary cycles and secondary error patterns. Other error patterns that are combinations of secondary and primary cycles also benefit from its irregularity. Thanks to this optimization, both the convergence of the code and its minimum distance are increased compared to a regular spatial permutation.

4.4 Further optimization

With increasing frame size, the study of the PEPs and SEPs alone is not sufficient to obtain efficient interleavers. Indeed, other error patterns appear, penalizing the minimum distance. In practice, the analysis and the exhaustive counting of these new patterns are too complex to be performed. Moreover, the temporal permutation (4) is too regular and leads to many internal rectangular secondary cycles. Indeed, for an internal secondary cycle as depicted in Figure 4.d), 4 symbols belong to the same slice in the second dimension. Since the spatial permutation is P periodic, the distances between the temporal indices of any two couples are necessarily equal in the second dimension. Moreover, they are equal in the first dimension as well. These cycles may lead to low weight SEPs. Thus, to increase minimum distance, the temporal permutation given by (4) is modified following the structure of the DVB-RCS interleaver [11], which has been thoroughly explained in [16]. This optimization step involves introducing a local disorder in the regular temporal permutation (4). Every permuted index $\Pi_T(t)$ is shifted modulo M of an amplitude that is not the same for all indices t. This is achieved by adding four coefficients $\left(b(i)\right)_{0\leq i<4}$ (11) to the temporal permutation. These coefficients are inferior or equal to M and verify that their values modulo 4 are all different.

$$\Pi_T(t) = a \cdot t + b(t \bmod 4) \bmod M \qquad (11)$$

The a parameter is the same as the one chosen in sub-section 4.2. Clearly, this local disorder reduces the overall spread of the interleaver that has been optimized with the choice of a in the regular temporal permutation. But an interleaver with reasonably lower spread can also achieve good performance and might even improve it at low error rates. Moreover, the local disorder modifies the summary distance of the secondary cycles: rectangular secondary cycles may not remain rectangular and vice-versa. Nevertheless, this local disorder introduces more irregularity in the interleaver and may lead to a better interleaver with higher minimum distance and better convergence. The optimal coefficients $(b(i))_{0 \le i < 4}$ are those leading to the highest minimum distance. An exhaustive evaluation of the minimum distance for all possible parameters can be achieved by using the error impulse method proposed by Berrou *et al.* [17].

Since the local disorder is of period 4, the size of the slice should be a multiple of 4 in order to design a homogeneous code with even protection for all symbols of a slice. Hence for a duo-binary code, the size of the slice has byte (8-bit) granularity. In addition, for an (*N*,*M*,*P*) turbo code the granularity of the frame size is *P* bytes.

In this section, the interleaver design has been discussed and its optimization is performed in three steps: maximization of the spread through a regular temporal permutation, maximization of the dispersion of the spatial permutation and introduction of a local disorder in the temporal permutation to increase the randomness of the interleaver. In the next section, the properties of parallelism in the interleaver are used to implement a low-complexity high-throughput turbo decoder.

5 Hardware implementation

5.1 Decoding architectures

Let us consider that a SISO unit performing a classical Sliding Window algorithm has been designed and synthesized. The throughput achieved by this SISO unit is related to its maximal working frequency F and the structure of the convolutional code. Basically, for a duo-binary convolutional code the corresponding SISO unit achieves roughly $D_S = 2.F$ Mbits/s. For a single iteration using the same SISO unit twice (once for each half-iteration), the throughput amounts to F Mbits/s. This leads to a throughput of $F /$ I Mbits/s for the turbo decoder with I decoding iterations, which is rather low (a few tens of Mbits/s). For a high-throughput turbo decoder, the number of SISO units needs to be increased. In fact, achieving an overall throughput of D Mbits/s consisting of I iterations (and thus $2.I$ half-iterations) requires duplicating the SISO unit $2.I \cdot D / Ds$ times. For the classical serial architecture [1] the SISO units work in a pipe-lined way: the memories and the functional units are duplicated as shown in Figure 9. In this figure, the same SISO unit is used for two half-iterations. Hence, this architecture requires two extrinsic information memories and one memory for the channel outputs per iteration. The area A_{TD}^{S} of the serial architecture is given by:

$$A_{TD}^{S} = I \cdot \frac{D}{D_S} \cdot 2 \cdot \left[A_{SISO} + 2 \cdot Mem_E + Mem_I \right] \qquad (12)$$

where A_{SISO} represents the area of one SISO unit, Mem_I and Mem_E the area of the memories for storing the channel values and the extrinsic information respectively.

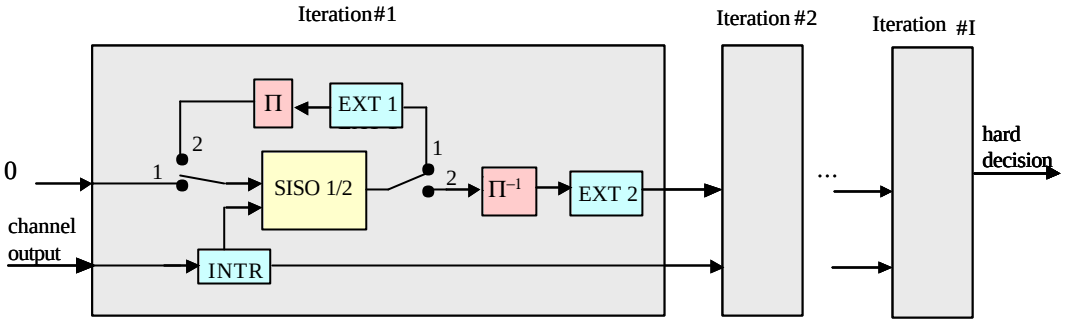

Figure 9: Description of the serial architecture as a pipe-line of SISO units (in this case D/D_S = 1/2).

Figure 9: Description de l'architecture série par une mise en cascade des décodeurs à entrée souple – sortie souple (dans ce cas D/D_S = 1/2).

Conversely, in our parallel architecture (see Figure 10), with a memory organization allowing $2 \cdot I \cdot D / D_S$ SISO units to operate in parallel, the areas of the memories can be extracted from the parenthesis in equation (12). Thus, only the functional units have to be duplicated and each SISO unit works independently on one slice of the frame. After each iteration, since the trellises of the slices are circular, the final states of the trellises become the initial states of the same slice[3]. There is therefore no extra processing for a learning period [2]. The SISO decoding algorithm for slices[4] is efficient in this sense since it requires M clock cycles to perform the decoding of one slice of M symbols. In the proposed parallel architecture, the same hardware (memory and SISO units) is used twice for each iteration in order to perform decoding in the two code dimensions. An additional buffer for the output of the channel is required for storing the next frame while decoding the current one. The area A_{TD}^{P} of the parallel architecture is given by:

$$A_{TD}^{P} = 2.Mem_I + Mem_E + I \cdot \frac{D}{D_S} \cdot 2 \cdot A_{SISO} \qquad (13)$$

[3] This is, in fact, the adaptation of the pointer method of [3] to the slice turbo-code.
[4] The SISO algorithm for slices is derived from the techniques proposed in [18]

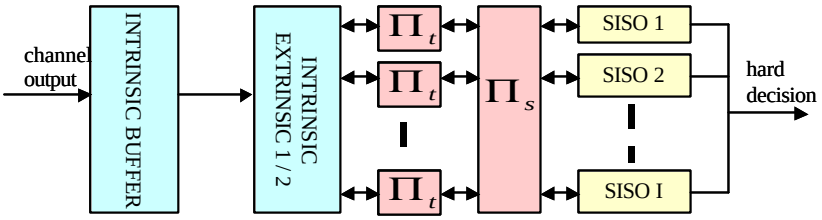

Figure 10: Description of the parallel architecture with SISO units working in parallel (in this case $D/D_S = 1/2$): the SISO units are used $2.I$ times for the I iterations, and the extrinsic memory stores the extrinsic information of the first dimension and of the second dimension alternately.

Figure 10 : Description de l'architecture parallèle avec une exécution simultanée des décodeurs à entrée souple – sortie souple (dans ce cas $D/D_S = 1/2$) : Les décodeurs à entrée souple – sortie souple sont utilisés $2.I$ fois pour effectuer les I itérations, la mémoire extrinsèque mémorise alternativement les informations extrinsèques de la première dimension et de la deuxième dimension.

5.2 Latency comparison

In this sub-section, the latency of both the serial and the parallel architectures are compared by assuming a sliding window SISO unit that requires M clock cycles to perform the decoding of one slice of M symbols. The latency of the turbo decoder for an information block length of N symbol is then defined as the number of clock cycles required to perform the I decoding iterations. Each iteration consists of two half-iterations of N cycles. The latency in number of clock cycles of the serial architecture is given by (14).

$$L_{TD}^{S} = 2 \cdot N \cdot I \qquad (14)$$

For the parallel architecture, the P SISO units are working simultaneously on a slice of M symbols and therefore the associated latency is given by (15).

$$L_{TD}^{P} = 2 \cdot \frac{N}{P} \cdot I = 2 \cdot M \cdot I \qquad (15)$$

Equations (14) and (15) show that the latency of the parallel architecture is P times shorter than the latency of the serial architecture.

5.3 Complexity results for ASIC implementation.

Table 1 compares the complexity of the two architectures for a throughput of 100 Mbits/s, using equations (12) and (13) and based on areas for a SISO unit and memories given by RTL (Register Transfer Level) synthesis in a 0.13µm technology. The complexity comparison is made for a turbo code with 8-state duo-binary trellises. The same constituent codes as those proposed by the DVB-RCS code [12] are used. The (2048,256,8) duo-binary code (*i.e.* 4096 bits) is decoded in 8 iterations using the Log-MAP algorithm [19]. For a clock frequency of 100 MHz, which is rather conservative, the SISO units achieve a throughput of 100 Msymbols/s (i.e. 200 Mbits/s), and hence D / D_S = 1/2. Because of memory duplication, which represents 80% of the total area, the classical serial architecture is more than twice as complex as the parallel architecture presented in this paper. The gap is even larger with more iterations.

Table 1 : Complexity evaluation (0.13µm technology) for a 4 kbit code with a code rate of 0.5, using 8 decoding iterations and D = 100 Mbits/s, D_S = 200 Mbits/s at 100 MHz.

Table 1 : Evaluation de la complexité (technologie 0.13µm) pour une code de 4 kbits de rendement 0.5 avec 8 itérations de décodage et D = 100 Mbits/s, D_S = 200 Mbits/s à 100 MHz.

		A_{SISO}	Mem_E	Mem_I	A_{TD}
	Area for 1 SISO units / 1 memory	0.3 mm²	0.25 mm²	0.2 mm²	
Serial	Number of SISO units / Memories	8	16	8	8 mm²
	Total Area	2.4 mm²	4.0 mm²	1.6 mm²	
Parallel	Number of SISO units / Memories	8	1	2	3.05 mm²
	Area	2.4 mm²	0.25 mm²	0.4 mm²	

5.4 Influence of the information block length and of the throughput.

Figure 11 presents the comparison of the complexities between the serial and the parallel architectures for a throughput of 100 Mbits/s and 8 decoding iterations. The gap between the two architectures increases as the information block length increases. For a block length of 16384 symbols and a throughput of 100 Mbits/s, the complexity of the serial architecture is more than four times that of the parallel architecture. Moreover, the gain of the parallel architecture increases with the throughput. Indeed, for a small throughput, there is less memory duplication for the serial architecture and hence its complexity is reduced, and is comparable to the parallel architecture. For example, for a block length of 16384 symbols, the complexity reduction amounts 80 %, 70 % and 50 % for throughputs of 100, 50 and 25 Mbits/s, respectively.

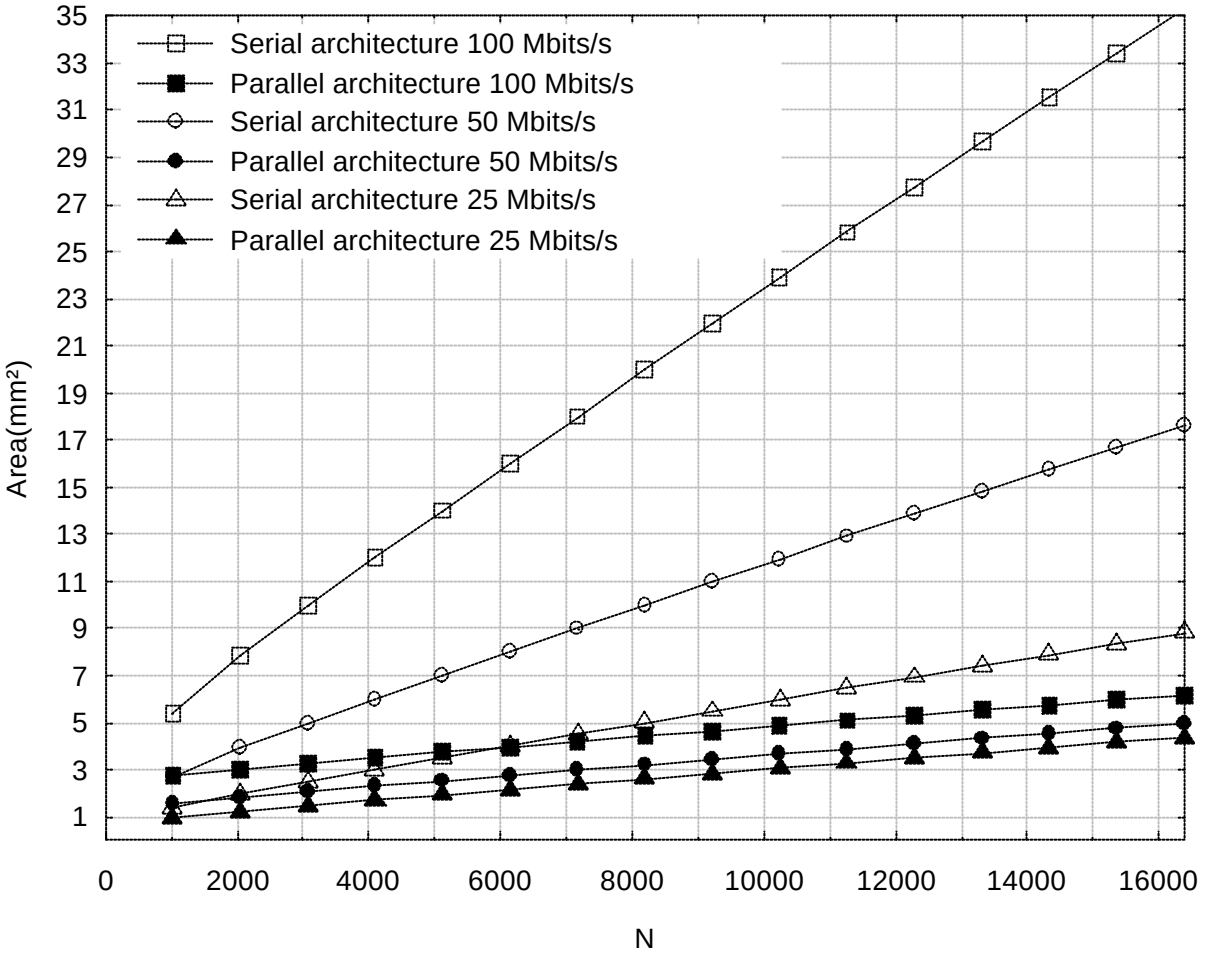

Figure 11: Comparison of the complexities of the serial and parallel architectures as a function of the information block size, for D = 25, 50 and 100 Mbits/s with 8 decoding iterations.

Figure 11 : Comparaison des complexités des architectures série et parallèle en fonction de la longueur du code pour D = 25, 50 et 100 Mbits/s avec 8 itérations des décodage.

The parallel execution of SISO units allowed by the parallel interleaver leads to a very efficient implementation for high throughput applications and requires no memory duplication compared to the classical serial architecture. In the previous section, the parallel interleaver optimization method has been described. In the next section, the validity of the optimization process is verified through

simulations and the performance is compared to a conventional two-dimension convolutional turbo code.

6 Simulation Results

6.1 Validation of the optimization process

Applying the different methods developed in section 3, a (2048,256,8) 8-state duo-binary code is designed. An intra-symbol permutation is also applied to the increase minimum distance [11]. These 1/2 rate codes are compared in Figure 12 with a (2048,2048,1) code, constructed with one slice using the equations for the DVB-RCS code [11] with interleaver parameters $P_0 = 45$ and $\{P_1, P_2, P_3\} = \{36,52,16\}$. The latter is also optimized with the same strategy as for the multiple slice turbo code: a large number of interleaver parameters are tested with the error impulse method. This optimization leads to an evaluated minimum distance of 20. Let us illustrate the effects of the different optimization steps of the interleaver. First, only the temporal permutation is optimized, assuming a regular spatial permutation (curve TO, in Figure 12). The minimum weight of the PEP is then above 30 but the SEP introduces a low minimum distance of 14 for the code. After the optimization of the spatial permutation, low weight SEPs are eliminated and the minimum distance increases to 18 (curve SO). Then, when the local disorder is added to the temporal permutation, the minimum distance is raised to 21 (curve LD).

The performance curves of the above codes are given in Figure 12. All performance curves presented in this paper are obtained by Monte-Carlo simulation with a minimum of 25 frames in error. The results are in accordance with the evaluated minimum distances of the codes. The (2048,256,8) code performs 0.3 dB better at a bit error rate (BER) of 10^{-7} than the DVB-RCS code, with a complexity reduction of more than 50% (see Table 1).

Table 1).

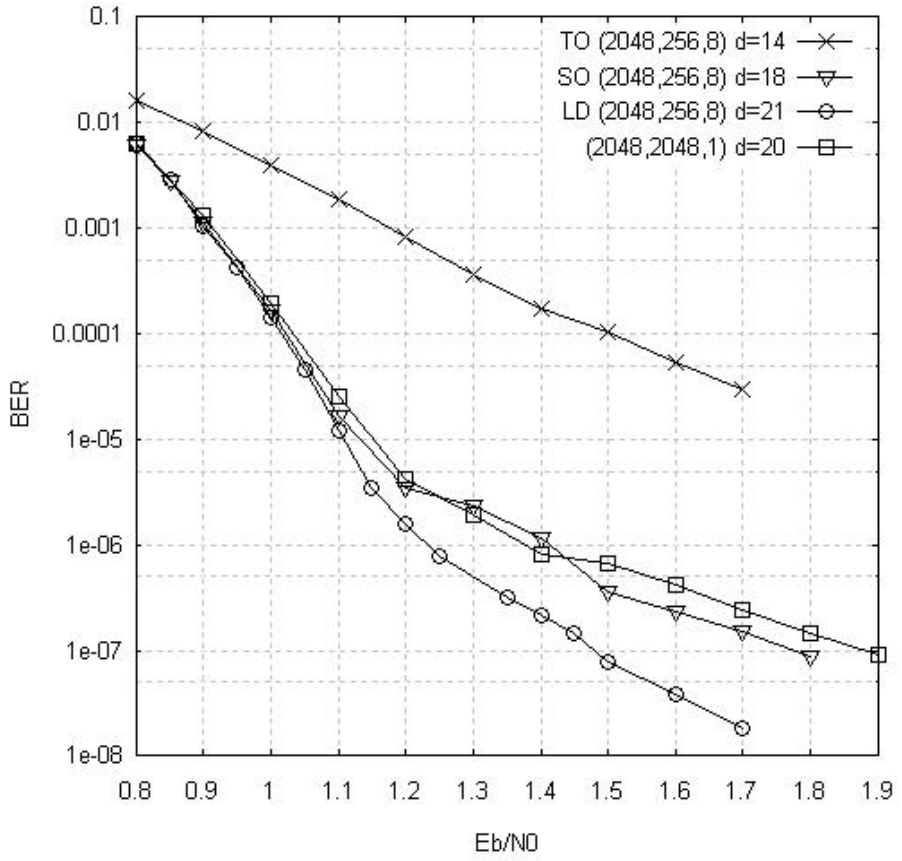

Figure 12: Performance of the (2048, 256, 8) and (2048,2048,1) duo-binary codes for 8 iterations.

Figure 12 : Performances des codes duo-binaires (2048, 256, 8) et (2048,2048,1) pour 8 itérations.

6.2 Influence of the code rate and frame size

In this sub-section we extend the results of rate 1/2 to higher codes rates and for an information block size of 1024 duo-binary symbols. Figure 13, Figure 14 and Figure 15 compare the bit error rate (BER) and frame error rate (FER) performance of codes constructed with 8 slices to codes with one slice for rates 1/2, 2/3 and 4/5, respectively. Two frame sizes are considered: 2048 and 1024

duo-binary symbols (only rates 1/2 and 2/3). The simulated performance shows that the gain of the multiple slice turbo code decreases when the code rate increases, and when the frame size decreases. For example, with a frame size of 2048 duo-binary symbols, for a target FER of 10^{-4} the FER of the multiple slice turbo code shows a gain of about 0.2 dB, 0.1 dB and 0.05dB for rates 1/2, 2/3 and 4/5, respectively. For a frame size of 1024 duo-binary symbols, the gain is reduced to less than 0.1 dB. Hence, for shorter information block lengths, the two turbo codes have the same performance in terms of convergence and minimal distance. It seems that the gain of the 8-slice multiple slice turbo code over the DVB-RCS-like code increases with the length of the code. This gain for longer block lengths can be explained by the good interleaver design for multiple slice turbo codes and especially its greater irregularity. Indeed, the spatial permutation of the code with 8 slices introduces another degree of freedom, *i.e.* irregularity, into the interleaver design, which is beneficial for longer block sizes. Moreover, the codes with one frame proposed in the DVB-RCS standard are between 48 and 1728 bits long and the interleaver was not designed for longer frames. Nevertheless, there is less flexibility in choosing the frame size for the multiple slice turbo code. Indeed, its granularity in bytes is a multiple of the number of slices, whereas the DVB-RCS code has a byte granularity.

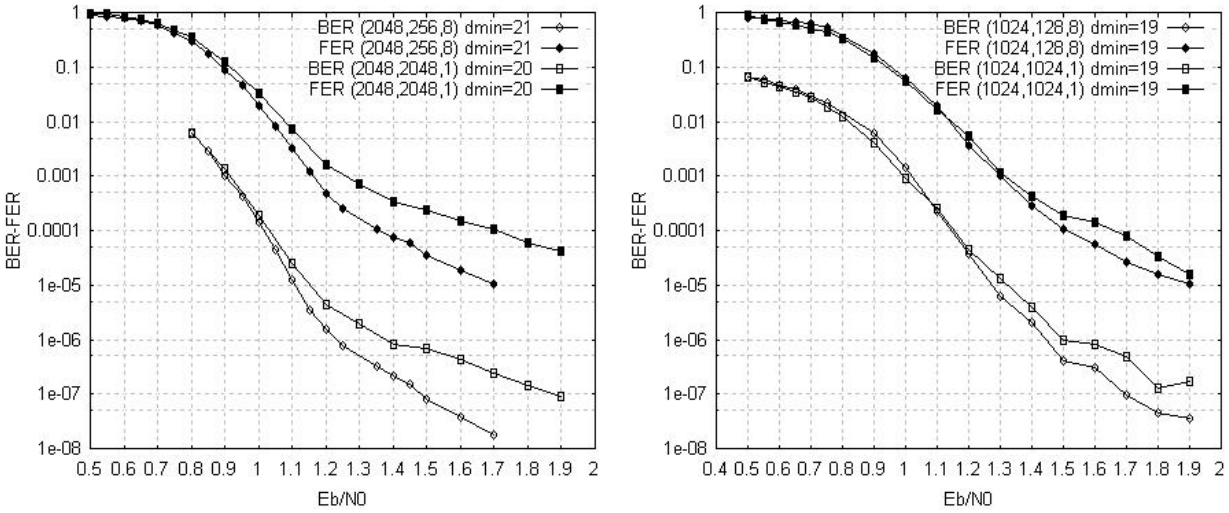

Figure 13: Performance comparison of the (2048, 256, 8) and (1024,128,8) multiple slice duo-binary codes to the (2048,2048,1) and (1024,1024,1) duo-binary codes for R=1/2.

Figure 13 : Comparaison des performances des codes duo-binaires (2048, 256, 8) et (1024,128,8) avec les codes duo-binaires (2048,2048,1) et (1024,1024,1) pour R=1/2.

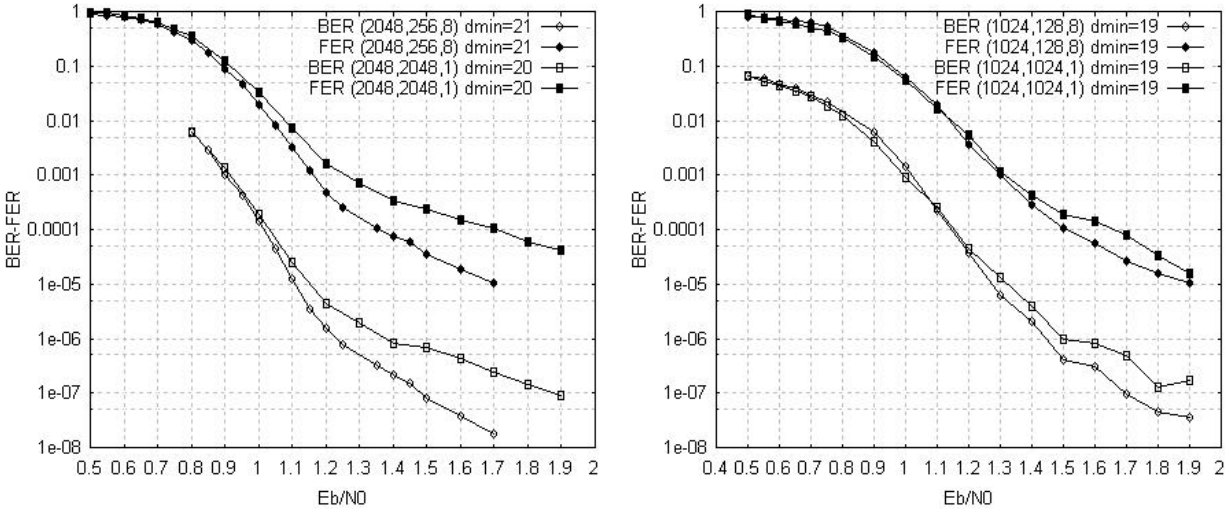

Figure 14: Performance comparison of the (2048, 256, 8) and (1024,128,8) multiple slice duo-binary codes to the (2048,2048,1) and (1024,1024,1) duo-binary codes for R=2/3.

Figure 14 : Comparaison des performances des codes duo-binaires (2048, 256, 8) et (1024,128,8) avec les codes duo-binaires (2048,2048,1) et (1024,1024,1) pour R=2/3.

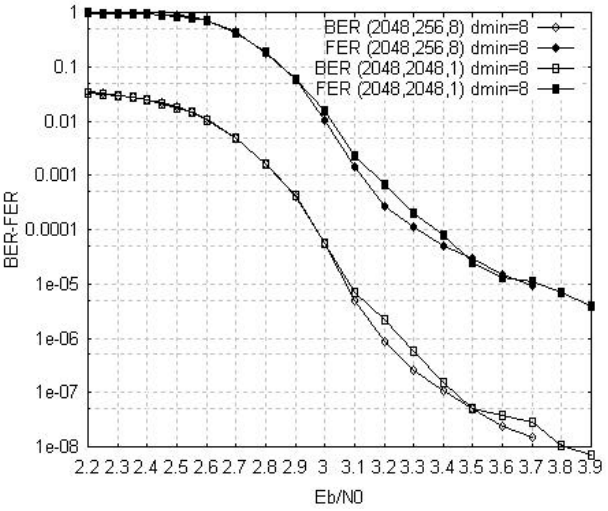

Figure 15: Performance comparison of the (2048, 256, 8) and (1024,128,8) multiple slice duo-binary codes to the (2048,2048,1) and (1024,1024,1) duo-binary codes for R=4/5.

Figure 15 : Comparaison des performances des codes duo-binaires (2048, 256, 8) et (1024,128,8) avec les codes duo-binaires (2048,2048,1) et (1024,1024,1) pour R=4/5.

7 Conclusion

A new family of convolutional turbo codes has been proposed. The slicing of the frame allows the decoder to work in parallel on independent slices. A study of an interleaver construction has been conducted, ensuring decoding parallelism, simple hardware implementation and good performance. The interleaver design is carried out in three steps: high spread through the regular temporal permutation, maximization of the dispersion of the spatial permutation and introduction of a local disorder in the temporal permutation to increase the randomness of the interleaver. The proposed scheme allows a hardware complexity reduction of more than 50% for a 100 Mbits/s turbo-decoder over a serial architecture for a 4 kbit information block length. This complexity

reduction increases with the block length and with the throughput of the decoder. Moreover with this parallel architecture, the latency is divided by the number slices decoded in parallel. The performance simulations show that the parallelism constraint in the interleaver construction introduces no degradation in performance, and a good interleaver construction can even improve it. However, the interleaver size has a coarser granularity than the conventional interleaver design. It has been shown that the interleaver gain increases with the length of the information block size and decreases for higher code rates.

This work on slice turbo codes has been extended to multiple turbo codes. In [20], we show that the addition of a third dimension can lower the error floor by two or three decades with a small convergence degradation (around 0.1 dB). Moreover, using slices, the notion of first and second dimension can be relaxed: the code can be seen as a LDPC-like code where the parity check constraints are simply replaced by tail-biting codes. Initial simulation results with this type of turbo code are very promising.

8 Acknowledgment

The authors wish to thank the anonymous reviewers of this paper for their positive criticism and for their valuable comments, which greatly helped to improve the paper and to make it more readable.

References:

[1] BERROU (C.), GLAVIEUX (A.) and THITIMAJSHIMA (P.), "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes", Proceedings of ICC'93, *IEEE International Conference on,* Geneva, Switzerland, pp. 1064-1070, May 1993.

[2] WANG (Z.), CHI (Z.) and PARHI (K.K.), "Area-Efficient High Speed Decoding Schemes for Turbo/MAP Decoders", IEEE Transactions on VLSI Systems, vol. 10, No. 6, pp. 902-912, Dec. 2002.

[3] WORM (A.), LAMM (H.) and WEHN (N.), "VLSI architectures for high speed MAP decoders", in Proceedings of the 14th International Conference on VLSI Design, pp 446-453, 2001.

[4] BERROU (C.), DOUILLARD (C.) and JÉZÉQUEL (M.), " Multiple parallel concatenation of circular recursive systematic codes ", Annales des Télécommunications, tome 54, n°3-4, pp 166-172, 1999.

[5] DOBKIN (R.), PELEG (M) and GINOSAR (R.), "Parallel VLSI Architecture for MAP Turbo Decoder", Proceedings of PIMRC 2002, Lisboa, Portugal, Sept. 2002.

[6] BOUTILLON (E.), CASTURA (J.) and KSCHISCHANG (R.F.), "Decoder-first code design", in Proceedings of ISTC2000, pp 459-462, Sept 2000.

[7] PEREZ (L.C.), SEGHERS (J.), and COSTELLO Jr. (D.J.), "A distance spectrum interpretation of turbo codes," IEEE Transactions on Information Theory, vol. 42, pp. 1698-1709, Nov. 1996.

[8] BENEDETTO (S.) and MONTORSI (G.), "Unveiling turbo-codes: some results on parallel concatenated coding schemes", IEEE Transactions on Information Theory, vol. 42, no. 2, pp. 409-429, Mar. 1996.

[9] HOKFELT (J.), EDFORS (O.) and MASENG (T.), "Interleaver Design for Turbo Codes Based on the Performance of Iterative Decoding", *ICC '99. IEEE International Conference on Communications*, Vol 1 , pp.93-97, 1999.

[10] BERROU (C.), DOUILLARD (C.) and JÉZÉQUEL (M.), "Designing turbo codes for low error rates", in Proceedings of *IEE Colloquium : "Turbo codes in digital broadcasting - Could it double capacity?"*, London, Nov. 1999.

[11] DVB-RCS Standard, "Interaction channel for satellite distribution systems", ETSI EN 301 790, V1.2.2, pp. 21-24, Dec 2000.

[12] DOUILLARD (C.), JÉZÉQUEL (M.), BERROU (C.), BRENGARTH (N.), TOUSCH (J.), Pham (N.), " The Turbo Code Standard for DVB-RCS", *2nd* International Symposium on Turbo Codes and Related Topics, pp. 535-538, Brest, France, Sept. 2000.

[13] CROZIER (S.), "New High-Spread High-Distance Interleavers for Turbo-Codes", *20th biennial Symposium on Communications*, pp. 3-7, Kingston, Canada 2000.

[14] THRUHACHEV (D.V.), LENTMAIER (M.), ZIGANGIROV (K. Sh.), "Some Results Concerning Design and Decoding of Turbo-Codes", *in Problems of Information Transmission*, vol. 37, vol. 37, No. 3, pp. 190-205, 2001.

[15] HEEGARD (C.), WICKER (S. B.), "Turbo Coding", *Kluwer Academic Publishers*, pp.50-53, 1999.

[16] BERROU (C.), SAOUTER (Y.), DOUILLARD (C.), KEROUEDAN (S.), JEZEQUEL (M.), "Designing good permutations for turbo codes: towards a single model", ICC'04, *International* Conference on Communications, Paris, June 2004.

[17] BERROU (C.), VATON (S.), JEZEQUEL (M.) and DOUILLARD (C.), "Computing the minimum distance of linear codes by the error impulse method", Proceedings of GLOBECOM 2002, Taipei, Taiwan, Nov 2002.

[18] BOUTILLON (E.), GROSS (W.J.), GULAK (P.G), "VLSI Architectures for the MAP Algorithm", IEEE Trans. On Communications., vol.51, No.2, pp. 175-185, 2003.

[19] ROBERTSON (P.), VILLEBRUN (E.) and HOEHER (P.), "A Comparison of Optimal and Sub-optimal Decoding Algorithm in the Log Domain", Proceedings of ICC, *IEEE International Conference on*, Seattle, WA, pp. 1009-1013, June 1995.

[20] GNAEDIG (D.), BOUTILLON (E.), JÉZÉQUEL (M.), "Design of Three-Dimensional Multiple Slice Turbo Codes", *submitted to the special issue on Turbo Processing, EURASIP Journal on Applied Signal Processing.*

List of figures:

List of tables: