EURASIP Journal on Applied Signal Processing 2005:6, 808–819

Design of Three-Dimensional Multiple Slice Turbo Codes

David Gnaedig

TurboConcept, Technopôle Brest-Iroise, 115 rue Claude Chappe, 29280 Plouzané, France
Email: david.gnaedig@turboconcept.com

LESTER, Université de Bretagne-Sud, BP 92116, 56321 Lorient Cedex, France

ENST Bretagne, Technopôle Brest-Iroise, CS 83818, 29238 Brest Cedex 3, France

Emmanuel Boutillon

LESTER, Université de Bretagne-Sud, BP 92116, 56321 Lorient Cedex, France
Email: emmanuel.boutillon@univ-ubs.fr

Michel Jézéquel

ENST Bretagne, Technopôle Brest-Iroise, CS 83818, 29238 Brest Cedex 3, France
Email: michel.jezequel@enst-bretagne.fr

Received 8 October 2003; Revised 8 November 2004

This paper proposes a new approach to designing low-complexity high-speed turbo codes for very low frame error rate applications. The key idea is to adapt and optimize the technique of multiple turbo codes to obtain the required frame error rate combined with a family of turbo codes, called multiple slice turbo codes (MSTCs), which allows high throughput at low hardware complexity. The proposed coding scheme is based on a versatile three-dimensional multiple slice turbo code (3D-MSTC) using duobinary trellises. Simple deterministic interleavers are used for the sake of hardware simplicity. A new heuristic optimization method of the interleavers is described, leading to excellent performance. Moreover, by a novel asymmetric puncturing pattern, we show that convergence can be traded off against minimum distance (i.e., error floor) in order to adapt the performance of the 3D-MSTC to the requirement of the application. Based on this asymmetry of the puncturing pattern, two new adapted iterative decoding structures are proposed. Their performance and associated decoder complexities are compared to an 8-state and a 16-state duobinary 2D-MSTC. For a 4 kb information frame, the 8-state trellis 3D-MSTC proposed achieves a throughput of 100 Mbps for an estimated area of 2.9 mm^2 in a 0.13 μm technology. The simulation results show that the FER is below 10^{-6} at SNR of 1.45 dB, which represents a gain of more than 0.5 dB over an 8-state 2D-MSTC. The union bound gives an error floor that appears at FER below 10^{-8}. The performance of the proposed 3D-MSTC for low FERs outperforms the performance of a 16-state 2D-MSTC with 20% less complexity.

Keywords and phrases: turbo codes, interleavers, multiple turbo codes, tail-biting codes, slice turbo codes.

1. INTRODUCTION

Turbo codes [1] are known to be very close to the Shannon limit. They are often constructed as a parallel concatenation of binary or duobinary [2] 8-state or 16-state recursive systematic convolutional codes. Turbo codes with 8-state trellises have a fast convergence at low signal-to-noise ratios (SNRs) but an error floor appears at high SNRs due to the weak minimum distance of these codes. For interactive, low-latency applications such as video conferencing requiring a very low frame error rate, an automatic repeat request (ARQ) system combined with a turbo code [3] cannot be used. Since this kind of application requires low latency, the block size cannot exceed few thousand bits. At constant block size, for very low frame error rate applications, several alternatives can be used. First, the more efficient 16-state trellis encoder can replace the 8-state trellis encoder at a cost of a double hardware complexity [4]. These encoders have the same waterfall region but the error floor region is considerably lowered due to the higher minimum distance. The second

alternative is to use a serial concatenation with an outer code, for example, either a Reed Solomon code [5] or a BCH code [6]. To achieve very good performance with these concatenations, a very large interleaver is needed to uniformly spread the errors at the output of the turbo decoder. The use of such an interleaver results in a long latency that is not acceptable for interactive services. Moreover, these serial concatenations decrease spectral efficiency.

The third alternative is to use multiple turbo codes. multiple turbo codes were introduced in [7] by adding a third dimension to the two-dimensional turbo code using reduced state trellises. It was shown in [8] that an increase of 50% of the minimum distance can be obtained by adding a third dimension to the turbo code. Using an 8-state trellis, this parallel code construction results in an equivalent or a higher minimum distance than for 16-state two-dimensional turbo codes. Other work on the constituent codes of three-dimensional turbo codes has been done by analyzing their convergence properties using extrinsic information transfer analysis [9], but these analyses do not handle the problem of designing good three-dimensional interleavers. Most of the designs of multiple turbo codes use random or *S*-random interleavers [7, 10], which are not efficient for scalable hardware implementation. Indeed, these types of interleavers are implemented in hardware as a table containing the interleaved address of all the symbols of the frame. Since practical applications generally require versatility, that is, several frame lengths and code rates, the storage of all the possible interleavers can represent a huge amount of memory.

In this paper, we generalize the multiple slice turbo codes (MSTCs) presented in [11] to the three-dimensional case. The idea is to propose a new and more efficient coding solution for high throughput, low hardware complexity, very low frame error rate applications. The use of MSTCs guarantees a parallel decoding architecture thanks to the way they are constructed. The careful design of a deterministic three-dimensional interleaver leads to a very simple address generation scheme and a high minimum distance for the code.

The paper is divided into five sections. In Section 2, multiple slice turbo codes are described, together with the interleaver construction. In Section 3, the design of three-dimensional MSTCs and of the interleavers is addressed. Then new decoding structures for three-dimensional codes are introduced in Section 4. Finally, the performance of the 2D-MSTC and 3D-MSTC is summarized in Section 5 and their complexities are compared in Section 6.

2. MULTIPLE SLICE TURBO CODES

The idea of multiple slice turbo codes (MSTC) was proposed by Gnaedig et al. [11]. The same idea has been mentioned independently in [12]. The aim of MSTCs is to increase by a factor P (the number of slices) the decoding parallelism of the turbo decoder without memory duplication. The principle of MSTCs is based on the following two properties.

(i) In each encoding dimension, the information frame of the N m-binary symbols is divided into P blocks (called "slices") of M symbols, where $N = M \cdot P$. Then, each slice is encoded independently by a convolutional recursive systematic convolution (CRSC) code. Finally, puncturing is applied to generate the desired code rate.

(ii) The permutation Π^i between the natural order of the information frame and the interleaved order of the ith dimension has a particular structure avoiding memory conflicts when a parallel architecture with P decoders is used.

The resulting MSTC is represented by the triplet (N, M, P).

After describing the construction of the interleaver, we will recall some simple rules for building efficient two-dimensional MSTCs (2D-MSTCs). Then, we will generalize these rules to the three-dimensional case (3D-MSTC). Note that all the results given in this paper are obtained with duobinary turbo codes [2]. The same results can also be used for classical turbo codes.

2.1. Multiple slice interleaver construction

The interleaver is designed jointly with the memory organization to allow parallel decoding of the P slices. 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 the P memory banks $\mathrm{MB}_0, \mathrm{MB}_1, \ldots, \mathrm{MB}_{P-1}$ without conflict. Since only one read can be made at any given time from a single-port memory, in order to access P data symbols in parallel, P memory banks are necessary.

The interleaver design is based on the one proposed in [13]: The interleaver structure presented in Figure 1 is mapped onto a hardware decoding architecture allowing a parallel decoding process.

The frame is first stored in the natural order in P memory banks, that is, the symbol with index j is stored in the memory bank $\lfloor j/M \rfloor$ at the address $j \bmod M$.

When considering the encoding (or decoding) of the ith dimension of the turbo code, the encoding (decoding) process is performed on independent consecutive blocks of M symbols of the permuted frame: 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 $r \in \{0, \ldots, P-1\}$ and $t \in \{0, \ldots, M-1\}$. For the symbol with index k of the interleaved order, the permutation Π^i associates a corresponding symbol in the natural order with index $\Pi^i(k) = \Pi^i(t, r)$. To avoid memory conflict, the interleaver function is split into two levels: a spatial permutation $\Pi^i_S(t, r)$ and a temporal permutation $\Pi^i_T(t, r)$, as defined in the following:

$$\Pi^i(k) = \Pi^i(t, r) = \Pi^i_S(t, r) \cdot M + \Pi^i_T(t, r). \tag{1}$$

The symbol with index k in the interleaved order is read from memory bank $\Pi^i_S(t, s)$ at address $\Pi^i_T(t, r)$. When coding (or decoding) the noninterleaved dimension (or dimension 0), the frame is processed in the natural order. The spatial and temporal permutations are then simply replaced by *identity* functions ($\Pi^0_S(t, r) = r$ and $\Pi^0_T(t, r) = t$).

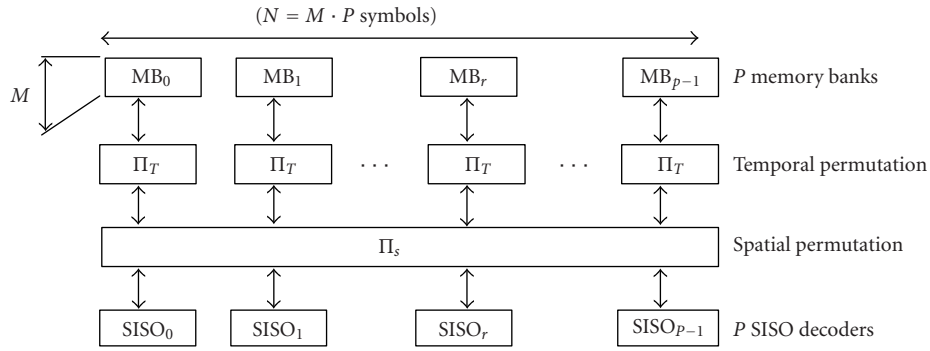

Figure 1: Interleaver structure for the (N, M, P) code.

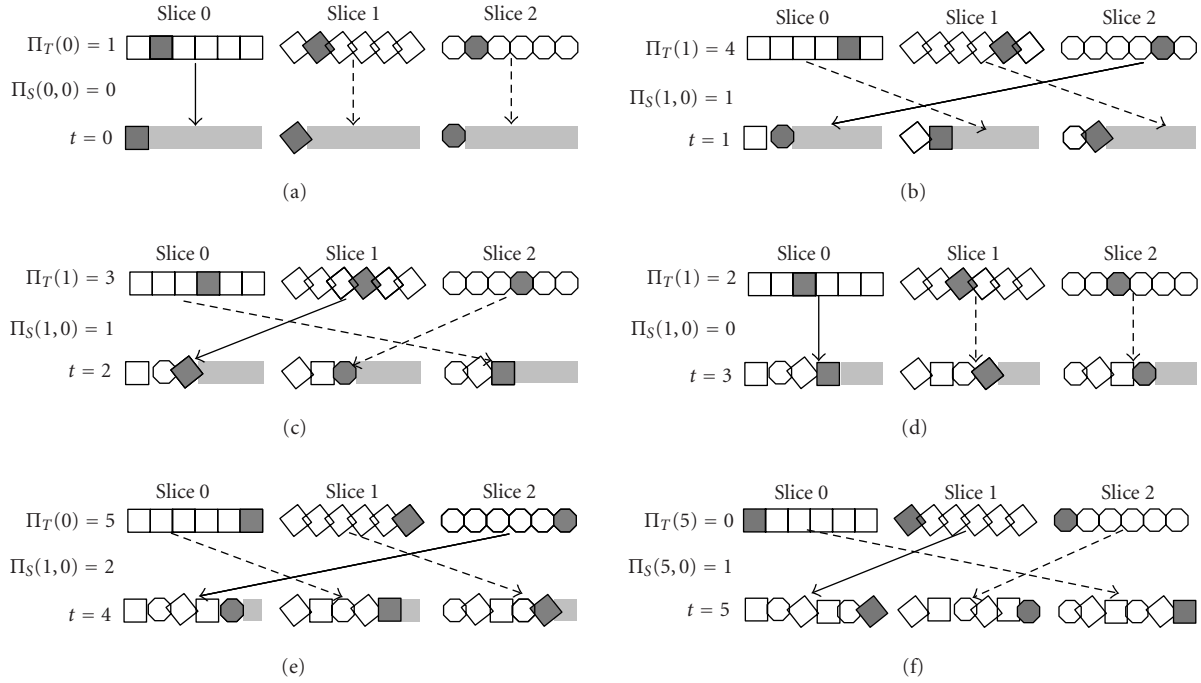

Figure 2: A basic example of an $(18, 6, 3)$ code with $\Pi_T(t) = \{1, 4, 3, 2, 5, 0\}$ and $A(t \bmod 3) = \{0, 2, 1\}$.

The spatial permutation allows the P data read-out to be transferred to the P decoders performing the max-log-MAP algorithm [14] (called the SISO algorithm). They are called SISO in Figure 1. Decoder r receives the data from memory bank $\Pi_S^i(t, s)$ at time t. For all fixed t, the function $\Pi_S^i(t, r)$ is then a bijection between the decoder index $r \in \{0, \ldots, P-1\}$ and the memory banks $\{0, \ldots, P-1\}$. To simplify the design, the shuffle network Π_S of Figure 1 is made with a simple barrel shifter, that is, for any given time t, $\Pi_S^i(t, r)$ is a rotation of amplitude $A^i(t)$. Furthermore, to maximize the shuffling between dimensions, we constrain function $\Pi_S^i(t, r)$ such that every P consecutive symbols of any slice come from P distinct memory banks. Thus, for a given r, the function $\Pi_S^i(t, r)$ is bijective and P-periodic. This means that for a given r, the function $\Pi_S^i(t, r)$ is a bijection between the temporal index $t \in \{0, \ldots, P-1\}$ and the set $\{0, \ldots, P-1\}$ of memory bank indices. Moreover, $\Pi_S^i(t, r)$ should also be P-periodic in the variable t, that is, $\Pi_S^i(t, r) = \Pi_S^i(t + P, r)$. The amplitude of the rotation $A^i(t)$ is then a P-periodic function and the spatial permutation is given by

$$\Pi_S^i(t, s) = \left(A^i(t \bmod P) + s\right) \bmod P. \tag{2}$$

2.2. A simple example of an interleaver

We construct a simple $(N, M, P) = (18, 6, 3)$ 2D-MSTC to clarify the interleaver construction. Let the temporal permutation be $\Pi_T(t) = \{1, 4, 3, 2, 5, 0\}$ (i.e., $\Pi_T(0) = 1$, $\Pi_T(1) = 4, \ldots$) and let the spatial permutation be a circular shift of amplitude $A(t \bmod 3)$, that is, 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.

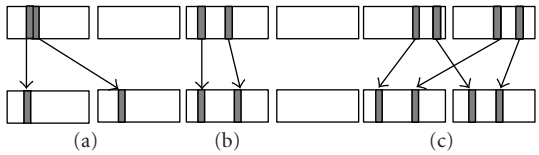

Figure 3: Primary and secondary cycles: (a) noncycle, (b) primary cycle, and (c) secondary cycle.

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.

2.3. Optimization of a two-dimensional interleaver

Optimization of an interleaver Π aims to fulfill two performance criteria: first, a good minimum distance for the asymptotic performance of the code at high signal-to-noise ratios (SNRs); second, fast convergence, that is, to obtain most of the coding gain performance in few decoding iterations at low SNRs. The convergence is influenced by the correlation between the extrinsic information, caused by the presence of short cycles in the interleaver. The cycles that are more likely to occur are primary and secondary cycles, as depicted in Figure 3. When these cycles correspond to combinations of low input-weight patterns leading to low weight codewords in the RSC constituent codes, they are called primary and secondary error patterns (PEPs and SEPs).

In [11, 15], the influence of the temporal and spatial permutations on these error patterns has been studied, using the following:

$$\Pi_T(t) = \alpha \cdot t \bmod M, \tag{3}$$

$$\Pi_S(t, s) = A(t \bmod P) + s \bmod P, \tag{4}$$

where α and M are mutually prime, and $A(t \bmod P)$ is a bijection between $\{0, \ldots, P-1\}$ and $\{0, \ldots, P-1\}$. Equation (4) for the spatial permutation is a circular shift of amplitude $A(t \bmod P)$, which can be easily implemented in hardware.

In order to characterize primary cycles and PEPs, other authors have introduced spread [16, 17, 18] and used the spread definition to improve the interleaver gain. In [11], an appropriate definition of spread is used taking into account the slicing of the constituent code. The spread between two symbols is defined as $S(k_1, k_2) = |k_1 - k_2|_M + |\Pi(k_1) - \Pi(k_2)|_M$, where $|a - b|_C$ is equal to $\min(|a - b|, C - |a - b|)$ if $\lfloor a/C \rfloor = \lfloor b/C \rfloor$ (this condition implies that the symbols a and b belong to the same slice when $C = M$), and is equal to infinity otherwise. The overall minimum spread is then defined as $S = \min_{k_1,k_2}[S(k_1, k_2)]$. Low weight PEPs are eliminated with high spread. Since the spatial permutation is P-periodic and bijective, two symbols separated by less than P symbols in the interleaved order are not in the same slice in the natural order. Their spread is then infinite. Using the definition of spread, the optimal parameter α maximizes the spread of the symbols separated by exactly P symbols.

Since the weight of the SEPs does not increase with high spread, we choose the spatial permutation in order to maximize the weight of these patterns. This weight is maximized for irregular spatial permutations. For a regular spatial permutation (e.g., $A(t) = a \cdot t + b \bmod P$, where a and P are relatively prime and $b > 0$) many SEPs with low Hamming weight are obtained [11]. To characterize the irregularity of a permutation, dispersion was introduced in [17]. In [15], an appropriate definition of dispersion is proposed to characterize the irregularity of the spatial permutation. First, for a couple $t_1, t_2 \in \{0, \ldots, P-1\}^2$, a displacement vector $D_v(t_1, t_2)$ of the spatial permutation is defined as $D_v(t_1, t_2) = (\Delta t, \Delta A)$, where $\Delta t = |t_1 - t_2|M$ and $\Delta A = |A(t_1) - A(t_2)|_P$. Let $D = \{t_1, t_2 \in \{0, \ldots, P-1\}^2, D_v(t1, t2)\}$ be the set of displacement vectors. The dispersion is then defined as the cardinality of D, that is, the number of different couples. It can be observed that the number of low weight SEPs decreases with high dispersion. This simple property is explained in detail in [15]. Some other criteria about the choice of the spatial permutation are also given in [15].

The criteria of spread and dispersion maximization increase the weight of PEPs and SEPs and improve the convergence of the code. But, with increasing frame size, the study of PEPs and SEPs alone is not sufficient to obtain efficient interleavers. Indeed, more complex error patterns appear, penalizing the minimum distance. In practice, the analysis and the thorough counting of these new patterns are too complex to be performed. Thus, to increase the minimum distance of the code, four coefficients $\beta(i)_{i=0,\ldots,3}$, multiple of 4, are added to the temporal permutation:

$$\Pi_T(t) = \alpha \cdot t + \beta(t \bmod 4) \bmod M. \tag{5}$$

The minimum distance is evaluated using the error impulse method proposed by Berrou et al. [19], which gives a good approximation of the minimum distance. Its results can be used to compute the union bound of the code.

3. THREE-DIMENSIONAL MULTIPLE SLICE TURBO CODES

In order to lower the error floor of the two-dimensional 8-state MSTC, a third dimension is introduced into the code. The goal is to increase the weight of the low weight error patterns of the 2D-MSTC, while maintaining good convergence at low SNRs. The interleaver of the third dimension has the same structure as the interleaver of 2.1 in order to allow the parallel decoding of the slices in each of the three dimensions.

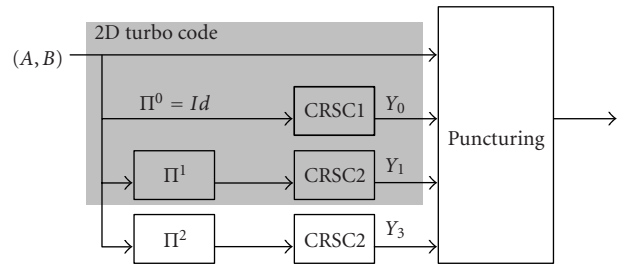

Figure 4: Three-dimensional multiple slice turbo code structure.

3.1. 3D slice turbo code construction

The generalization of the 2D slice turbo code to a 3D slice turbo code is straightforward (see Figure 4). Like for the second dimension, the third dimension is also sliced and its associated interleaver Π^2 has the same structure as that of Π^1:

$$\Pi^2(k) = \Pi^2(t,r) = \Pi^2_S(t,r) \cdot M + \Pi^2_T(t,r). \quad (6)$$

The generation of the addresses in the third dimension is simple and, due to the construction of Π^2, the architecture of Figure 1 can also compute the third dimension interleaver, with a degree P of parallelism, with negligible extra hardware (only interleaver parameters need to be stored).

Since the first initial paper on turbo codes in 1993 [1], much work has been done on efficient design methodologies for obtaining good 2D interleavers [16, 20, 21]. There are several papers dealing with multidimensional turbocodes [7, 9, 22, 23] but, unfortunately, very few papers consider the complex problem of the construction of good 3D interleavers. The 3D interleaver designs presented in [7, 10, 24] are based on random and S-random interleavers. Note that dithered relative prime (DRP) interleavers introduced by Crozier and Guinand in [25] have been used in [26] to design low-complexity multiple turbo codes. Moreover, none of these papers deals with the construction of good interleavers for duobinary codes. We have restricted our efforts to obtaining a class of couples (Π^1, Π^2) for 3D-MSTC verifying the following three properties.

(1) A 3D interleaver (Id, Π^1, Π^2) is said to be a "good" 3D interleaver if the three 2D interleavers defined by $(Id, \Pi^1), (Id, \Pi^2), (\Pi^1, \Pi^2) = (Id, \Pi^{1^{-1}} \circ \Pi^2)$ are "not weak," that is, the spreads for the temporal permutations and the dispersions for the spatial permutations are optimized by using the 2D interleaver construction developed in Section 2.3.

(2) There is a maximum global spreading of the message symbols over the slices, that is, the maximum number S_3 of common symbols between 3 distinct slices should be minimized.[1]

[1] The minimal value of S_3 is $\lfloor M/P^2 \rfloor$. When P^2 divides M, the appropriate choice of temporal permutation leads to $S_3 = M/P^2$.

Table 1: Puncturing patterns $P1$, $P2$, and $P3$ of different periods.

h	$h=1$	$h=3$	$h=4$	$h=8$	$h=16$
$P1$	1	011	0111	01111111	0111111111111111
$P2$	1	101	1101	11110111	1111111011111111
$P3$	0	110	0101	10001000	10000000100000000

(3) There is a regular intersymbol permutation ((A, B) becomes (B, A)): in the second dimension, all even indices are permuted; in the third dimension, all odd indices are permuted.

Note that these three conditions are an *a priori* choice, based on the authors intuition and on their work on 2D interleavers. Simulation results show that they effectively lead to efficient 3D turbo codes. Since the constituent codes are duobinary codes of rate 2/3, the overall rate of the turbo code without puncturing is 2/5. Puncturing is applied on the parity bits and on the systematic bits to generate the desired code rate. It will be shown, however, in the sequel that the puncturing strategy has a dramatic influence on performance. Moreover, the interleaver optimization process can use the properties of the puncturing strategy, as will be seen in the next section.

3.2. Puncturing

With irregular spatial rotations, the influence of puncturing on the performance of the rate 1/2 three-dimensional multiple slice turbo code has been studied. For a duobinary 3D-MSTC, every incoming symbol (2 bits) generates three parity bits y_1, y_2, y_3. Thus, to obtain a rate 1/2, one third of the parity bits has to be punctured. We define the puncturing patterns $P1$, $P2$, and $P3$ of parity bits y_1, y_2, y_3 as a stream of bit of periodicity h, where a "1" means that the parity bit is transmitted and a "0" means that the parity bit is discarded. In our test, the puncturing is uniformly distributed among the different symbols, that is, one and only one parity bit is discarded for each information symbol.

Table 1 gives different puncturing patterns for $h = 1, 3, 4, 8$, and 16. The case $h = 1$ corresponds, in fact, to a standard 2D code. The case $h = 3$ is a regular three-dimensional code with uniform protection for the three dimensions. For $h = 4, 8, 16$ (h a power of 2), $P1$ is constructed with a "0" in first position, "1" elsewhere, $P2$ is constructed with a "0" in the $h/2 - 1$ position, "1" elsewhere, and finally $P3$ is constructed with a "1" in the first and the $h/2$ position and "0" elsewhere. This construction guarantees that a single parity bit is punctured for every incoming symbol.

The performance of different puncturing patterns of period $h = 1, 3, 4, 8, 16, 32$, and 64 is given in Figure 5 for a duobinary 8-state MSTC with parameters (2056, 256, 8). The decoding is a floating point max-log-MAP algorithm [14], with 10 decoding iterations. The iterative decoding method used by the decoder is the extended serial decoding structure proposed in [27]. More details on this method are given in Section 4.

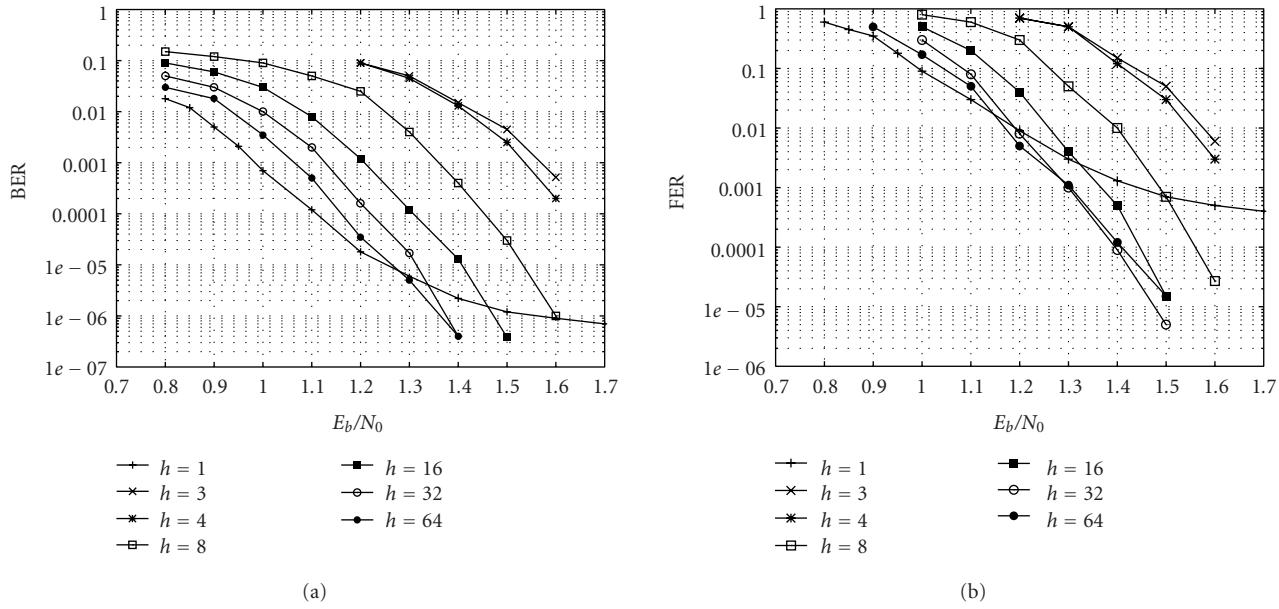

Figure 5: Performance of (2048, 256, 8) duobinary 8-state codes with different puncturing patterns with QPSK modulation on an AWGN channel.

The regular 3D code ($h = 3$) has no error floor due to its very high minimum distance: the impulsive estimation method [19] gives a minimum distance greater than 50. The drawback is a convergence loss of 0.5 dB at a bit error rate of 10^{-4} compared to the 2D code ($h = 1$). This loss of convergence has already been noted in the literature for binary turbo codes [8]. It can be explained by the fact that for every information symbol, the information relative to this symbol is spread over three trellises, instead of two for a 2D code. Hence, the waterfall region of the 3D code is at a considerably higher SNR than that of the 2D code.

As shown in [28], a high minimum distance is not needed to achieve a target FER of 10^{-7}. Indeed, for a 4 kb frame, the matched Hamming distance (MHD) is around 35. To reduce the minimum distance of the 3D code and to improve its convergence, we tend towards a 2D code by puncturing the third dimension more and more, while the first two dimensions are evenly and far more protected. The 3D codes with increasing puncturing period h given in Table 1 tend towards the 2D code: convergence at lower SNR and lower minimum distance. The 3D code is thus designed to trade off the loss of convergence with a minimum distance close to the MHD. The 3D code with puncturing period $h = 64$ has a convergence loss of less than 0.1 dB, but an error floor appears. The code of puncturing period $h = 32$ seems to have a reduced convergence loss and a high minimum distance.

The asymmetry in the protection of the three dimensions will be used in the decoder to reduce its complexity and improve its performance. Moreover, the optimal interleaver design depends on the puncturing patterns of the three dimensions. Thus, the interleaver optimization process has been redefined in order to take into account the asymmetric puncturing of the code: the first interleaver is chosen to obtain a good 2D-MSTC with the two highly protected dimensions. Then, the parameters of the second interleaver, that is, the third dimension, are selected in order to maximize the minimum distance of the code. The minimal distance is evaluated using the error impulse method [19]. This optimization process in two steps converges easily to an optimal solution and leads to an estimated minimum distance of 34, given by the error impulse method.

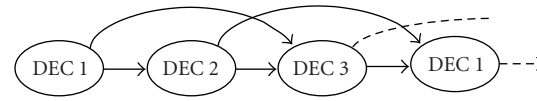

Figure 6: Extended serial decoding structure.

4. DECODING STRUCTURE

After describing the classical extended serial method [27], we study the impact of the scaling factors used to scale the extrinsic information during the decoding process [29]. We will show that an appropriate choice of scaling factor can increase performance while reducing decoding complexity (hybrid extended serial method). Then, we propose a suboptimal decoding method (partial serial method and hybrid partial serial method) that allows the third dimension to be decoded with a classical two-dimensional turbo decoder thanks to negligible additional hardware. Performance of the different coding schemes is also given.

4.1. Extended serial structure

The extended serial (ES) decoding structure is depicted in Figure 6. The three-dimensions are decoded sequentially,

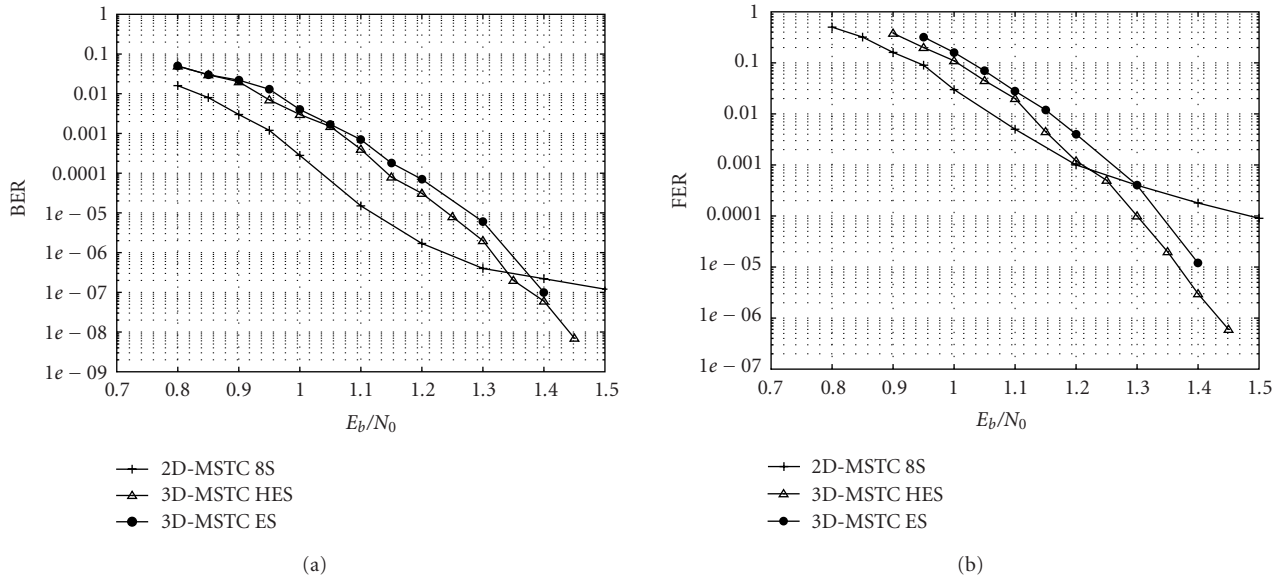

Figure 7: BER and FER comparison between the conventional ES structures and the hybrid HES structures for (2048, 256, 8) duobinary turbo codes of rate 1/2 over an AWGN channel with the max-log-MAP algorithm (10 full iterations) and $h = 32$.

Figure 8: Partial serial decoding structure and extrinsic memory content (E1, E2, and E3 correspond to extrinsic information produced by dimension 1, 2, and 3, respectively).

and each dimension receives information from the other two dimensions. This structure is repeated periodically with a period of 3.

With the ES architecture, each decoder uses the extrinsic information from the other two decoders, and therefore at least two extrinsic data per symbol must be stored. Thus the extrinsic memory is doubled.

4.2. Optimization of the decoding structure

Since the three-dimensional code is asymmetric and the third dimension is weak, during the first iterations, the reliability of the extrinsic information of the third dimension is low. In other words, during the first decoding iterations, the computation of the third dimension is useless and can thus be discarded. Moreover, for a max-log-MAP algorithm, to avoid performance degradation, extrinsic information is usually scaled by an SNR-independent scaling factor [29]. This scaling factor increases along the iterations.

By simulation, two different sets of scaling factors were jointly optimized, one for the first two equally protected dimensions and one for the weaker third dimension. For the third dimension, typically, the scaling factor is 0 for the first iterations (the third dimension is not decoded in practice) then it grows from 0.2 to 1 during the last iterations.

Thus, during the first iterations, the turbo decoder iterates only between the first two more protected dimensions (conventional two-dimensional serial structure S). Then, during the last iterations, an ES structure is used. This new structure will be called the hybrid extended serial (HES) structure.

Figure 7 compares the performance of the conventional nonoptimized ES structure with the performance of the optimized hybrid HES structure for 10 decoding iterations. For the hybrid structure, the third dimension is not decoded during the first 5 iterations. Then its scaling factor grows from 0.2 to 1 during the 5 last iterations. For the ES structure, the same scaling factor growing from 0.7 to 1 is used for the three dimensions. The simulation results show that the optimized structure slightly improves the performance for less computational complexity and negligible additional hardware. Unlike the classical structure, the optimized structure takes into account this unequal protection to improve decoding performance.

4.3. Partial serial structure

The ES decoder requires an additional extrinsic memory in order to store the extrinsic information from the last two decoding iterations. This extra memory leads to additional

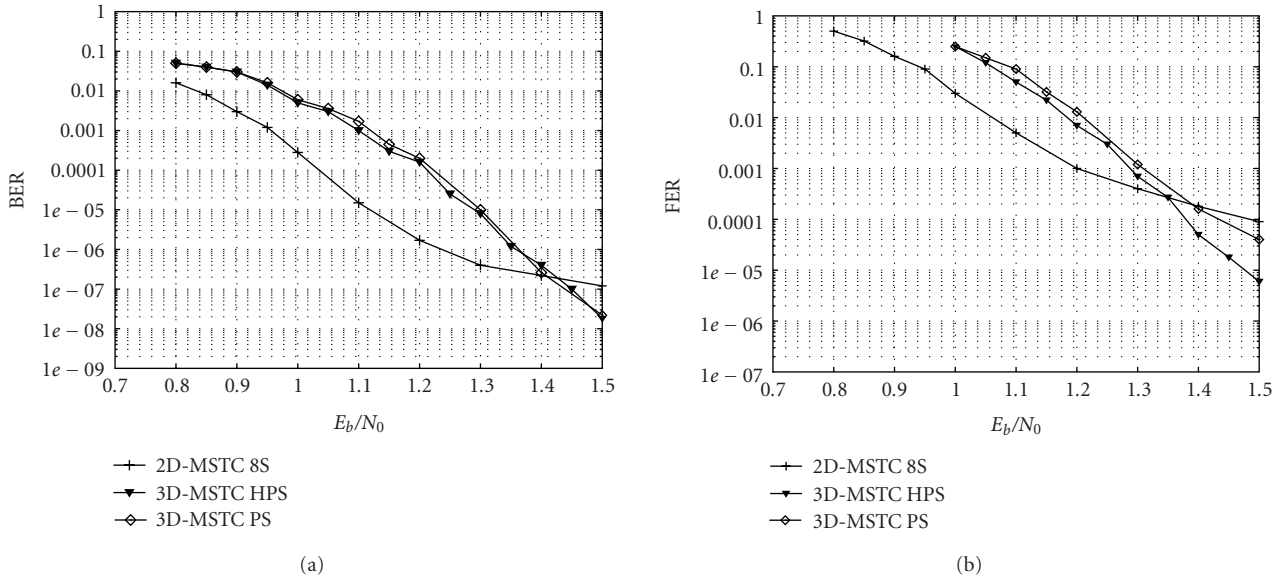

Figure 9: BER and FER comparison between the conventional PS structures and the hybrid HPS structures for (2048, 256, 8) duobinary turbo codes of rate 1/2 over an AWGN channel with the max-log-MAP algorithm (10 full iterations) and $h = 32$.

complexity compared to the 2D-MSTC. In order to adapt the 2D-MSTC decoder to the case of 3D-MSTC, with no significant additional hardware, a new serial decoding, called partial serial (PS) decoding structure, is introduced. This decoding structure is given in Figure 8, and its period is 4: during one period, dimensions 1 and 2 are decoded once, while dimension 3 is decoded twice. This structure is said to be "partial" because the third dimension only benefits from the extrinsic information of a single dimension: dimension 1 or dimension 2. Dimension 1 (or dimension 2) benefits from the extrinsic information of both dimension 2 (dimension 1, respectively) and dimension 3. These two data are added in a single memory, which will be read by the second dimension (first dimension, respectively). Hence, this structure only requires one extrinsic value per symbol to be stored. This structure reduces the memory requirements but increases the computational complexity. Compared to the ES structure, it is suboptimal, because the third dimension benefits from one dimension directly, and from the other indirectly. In terms of complexity, each decoding iteration of the partial serial (PS) method requires 4 subiterations.

Like for the HES structure, the same two sets of scaling factors are used with the PS structure leading to the hybrid partial serial (HPS) structure. As shown in Figure 9, the conclusions are the same as for the comparison between the ES and HES structures: performance improvements for the hybrid structure with less complexity.

5. PERFORMANCE

The performance of the proposed 3D-MSTC with $h = 32$ is compared to the performance of the 2D 8-state MSTC and 2D 16-state MSTC. All the codes presented in this section have a length of 4096 bits constructed with 8 slices of 256 duobinary symbols in every code dimension. They are compared through Monte Carlo simulations over an AWGN channel using a floating point max-log-MAP algorithm. Two comparisons are made at a constant decoder throughput and delay. First, the asymptotic performance of the different coding schemes is compared. For this comparison at constant throughput, the computational complexity of one decoder increases as the number of required subiterations increases. Then, in order to obtain a fair comparison, the performance is compared for the same computational complexity.

5.1. Asymptotic performance of the codes

Simulation results show that, for the MSTC codes used here, 10 iterations are sufficient to obtain most of the error-correction capabilities of the codes. In fact, additional iterations can only increase the SNR for a given BER than less than 0.1 dB. 2D-MSTCs with 8-state and 16-state trellises are compared in Figure 10 with 3D-MSTCs, for both HES and HPS decoding structures.

The FER results show that the 2D codes converge faster (0.1 dB) than the 3D code in the waterfall region. However, the union bound curves (denoted by UB) show that the error floor of the 3D code (minimum distance of 34) is slightly lower than of the 16-state code (minimum distance 32) at high SNRs. Compared to the HES decoding structure, the HPS decoding structure has a loss of less than 0.1 dB over the whole range of SNRs.

For the asymptotic performance of the codes, the computational complexity of one decoder increases as the number

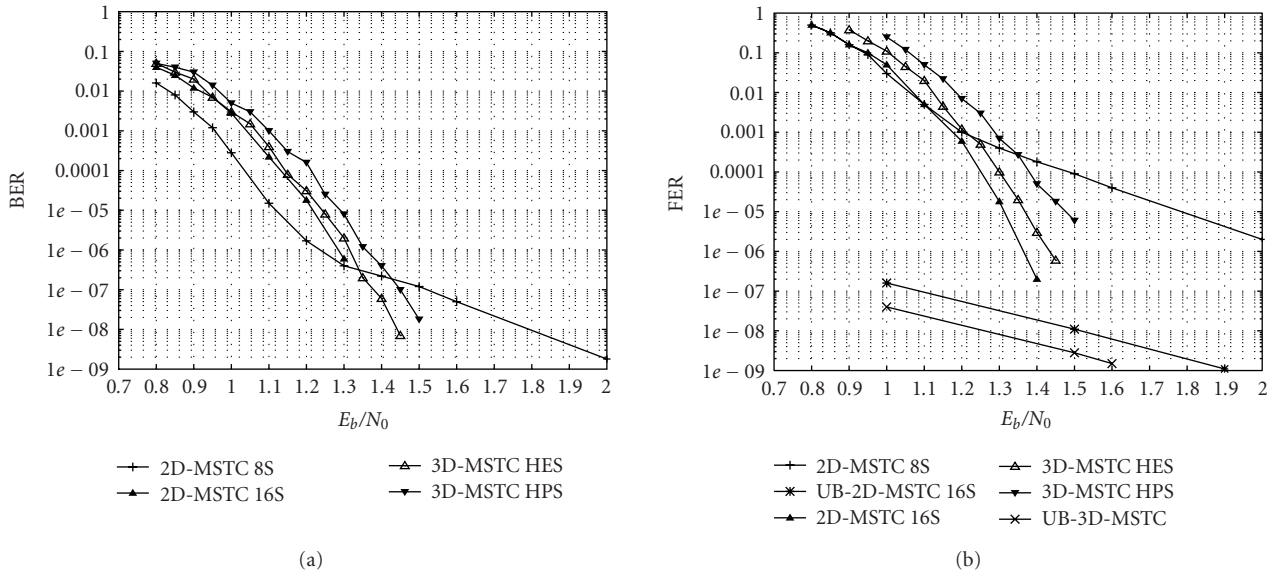

Figure 10: BER and FER of (2048, 256, 8) duobinary turbo codes of rate 1/2 over an AWGN channel with the max-log-MAP algorithm (asymptotic performance for 10 decoding iterations).

of required subiterations increases. Hence, to achieve a constant decoding throughput and delay, the corresponding decoder complexity increases as the number of required subiterations increases. This complexity comparison is analyzed in Section 6.2.

5.2. Comparison at constant computational complexity

It is obvious that the computational complexity for one decoding iteration differs between the different codes. In order to make a fair comparison, simulation results are given at constant computational complexity, that is, the same number of subiterations (decoding one dimension of the code). Thus, the decoding delay is the same for the different codes. The complexity of a 16-state trellis is assumed to be twice the complexity of an 8-state trellis. Figure 11 compares the performance for a total J of 20 subiterations of an 8-state trellis. It can been seen that, at constant complexity, HES is more efficient than the 2D 16-state MSTC over the whole range of SNRs. Moreover, HES becomes much more efficient than 2D 8-state MSTC for an FER below 10^{-4}. In addition, at a target FER of 10^{-6}, it achieves a gain of more than 0.5 dB over the 2D 8-state code. The 3D-MSTC code with HPS decoding structure shows an "error floor" at high SNRs, and therefore this decoding structure does not seem to be appropriate for this frame size and computational complexity.

When designing turbo codes, it is necessary to trade off complexity and performance. Thus, before drawing conclusions about the superiority of one over another, a comparison of the complexity of the different decoding schemes is required.

6. COMPLEXITY COMPARISON

The performance comparison of Section 5.2 pointed out that for a given computational complexity, the 3D-MSTC with HES decoding structure outperforms both 2D 8-state and 16-state codes at an FER below 10^{-4}. In terms of area, the memories of the different solutions have also to be taken into account. A generic model of area is developed in this section. This model is used to compare the different codes (of Section 5.1) with 10 full decoding iterations for their asymptotic performance.

6.1. Complexity modeling

A simple hardware complexity model is given to compare the area of the different coding schemes described in this paper. This model assumes an ASIC implementation in a 0.13 μm technology with a clock frequency F. It only takes into account the computational complexity, that is, the number P of SISOs working in parallel, and the memory area required by the turbo decoder. Moreover, there is an additional key assumption: the decoding latency of a codeword is equal to, or below, the time required to receive a new codeword (the frame duration). Thus, a parallel decoder architecture [11, 15, 30] is needed to perform the required number of iterations during the frame duration.

Since the codes are duobinary, a SISO decoder working at a frequency F can achieve a throughput of $2F$ Mbps [11]. Let J be the total number of subiterations decoded during the turbo decoding process and let D (in Mbps) be the throughput of the code. During one frame duration, every SISO decoder can decode $2 \cdot F/D$ slices. The real-time constraint implies that the SISO decoders can decode J subiterations within the reception of one frame, that is, $P \geq J \cdot D/(2 \cdot F)$.

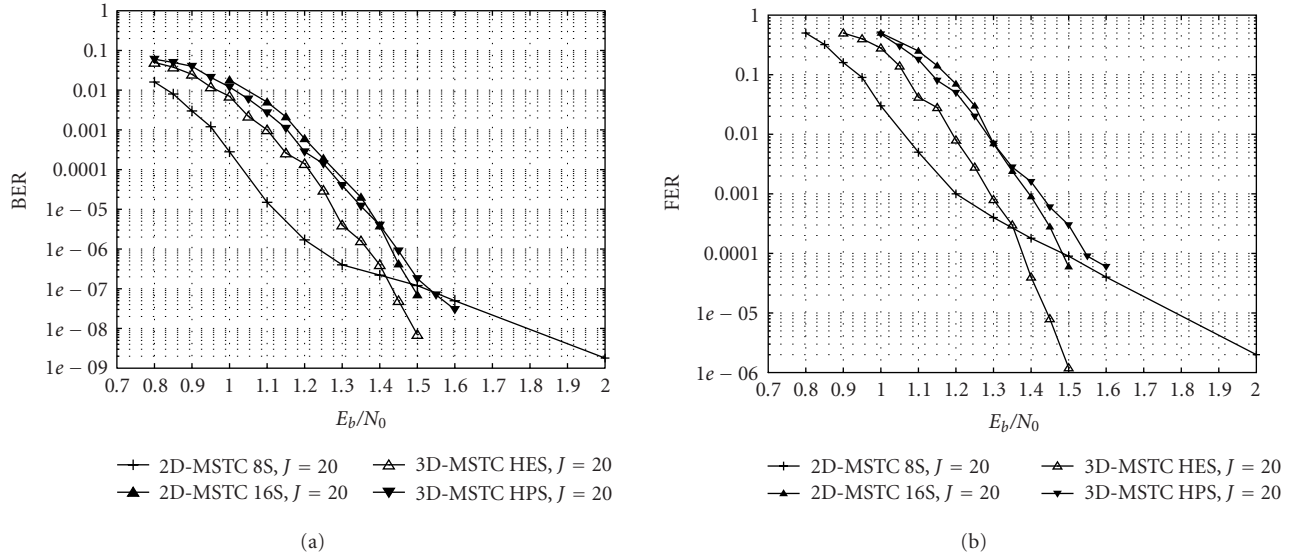

Figure 11: BER and FER of $(2048, 256, 8)$ duobinary turbo codes of rate 1/2 over an AWGN channel with the max-log-MAP algorithm (constant computational complexity $J = 20$ subiterations).

The memories required for the decoding process are composed of the intrinsic memory, which contains the output of the channel, the extrinsic memory of the current decoded frame, and an additional buffer to store the next frame while decoding the current one. The number of extrinsic memories η_E is equal to 2 for extended serial structures, 1 otherwise. The equation representing the estimation model is given in

$$A_{TD} = \left\lceil J \cdot \frac{D}{2 \cdot F} \right\rceil \cdot A_{\text{SISO}}(s) + \eta_E \text{Mem}_E(k) + 2 \cdot \text{Mem}_I(k), \tag{7}$$

where k is the number of information bits and where $\lceil x \rceil = \lfloor x \rfloor + 1$ and $\lfloor \cdot \rfloor$ denotes the integral part function. The areas $A_{\text{SISO}}(s)$ of an s-state SISO decoder are given by RTL synthesis in a 0.13 μm technology. The SISO algorithm used to compute its area is a sliding window algorithm. The areas obtained are 0.3 and $0.6mm^2$ for 8-state and 16-state SISO decoders, respectively.

6.2. Comparison

Table 2 gives the values for η_E and J for the different simulated codes presented in Section 5.1. Note that the areas of memories are also obtained by VHDL synthesis.

The values given in Table 2 show that the 3D 8-state decoders can outperform the 2D 16-state decoder in terms of complexity. Indeed, the size and the number of memories are the same for the HPS decoding structure, whereas the complexity of all SISOs is 30% higher for the 2D 16-state code. The HES decoding structure is 60% less complex in terms of SISO complexity, but the number of memories is doubled compared to the 2D 16 state code. To conclude on the relative complexity of this latter example and to choose between the HES and HPS decoding structure, we need to compare the total turbo decoder area, for a given throughput D and information frame size k. The results of this comparison are given in Table 2 for a 4096-bit information frame size and $D/2 \cdot F = 0.25$. To achieve a rate $D/2 \cdot F = 0.25$, for a 50 Mbps turbo decoder, the SISO should achieve a throughput of 200 Mbps. Since the SISO decoder is duobinary, the required clock frequency is 100 MHz, which is rather conservative. For a 100 Mbps turbo decoder, the clock frequency is doubled.

Table 2 shows that for a frame size of 4 kb, the 2D 16-state is the most complex structure. Its complexity is 75% higher than that of the 2D 8-state code. The complexities of the HES and HPS decoding structures are equivalent and 40% higher than that of the 2D 8-state code.

With increasing frame size, the size of the memory to store the extrinsic information increases. For small frame sizes up to 5000 bits, the HES decoding structure is less complex than the HPS structure. For longer block sizes, the HPS structure becomes less complex than the HES structure. The simulated performance of the HPS decoding structure shows an "error floor" for small to medium frame sizes and for a reduced number of iterations. Hence, this decoding structure may only be attractive for very long frame sizes (above 10 kb) for a number of iterations close to its asymptotic performance.

6.3. Discussion

These complexity results are preliminary results, but they show that three-dimensional turbo decoders can be efficiently implemented with considerably less complexity than the two-dimensional decoder with 16-state trellises. The architectures of the three-dimensional decoders may be improved by optimizing the tradeoff between performance and the total number of subiterations. Moreover, the extrinsic memories for the HES can be reduced by using scaling of the

Table 2: Area of the code of Section 5.1. for a ratio $D/2F = 0.25$ and $k = 4096$ bits.

Code	Decoding structure	η_E	J	SISO area	Memory area	Total area
2D 8-state	S	1	20	1.45 mm^2	0.38 + 0.24 mm^2	2.08 mm^2
2D 16-state	S	1	20	2.97 mm^2	0.38 + 0.24 mm^2	3.60 mm^2
3D 8-state	ES	2	30	2.32 mm^2	0.38 + 0.49 mm^2	3.19 mm^2
3D 8-state	PS	1	40	2.90 mm^2	0.38 + 0.24 mm^2	3.53 mm^2
3D 8-state	HES	2	25	2.03 mm^2	0.38 + 0.49 mm^2	2.90 mm^2
3D 8-state	HPS	1	30	2.32 mm^2	0.38 + 0.24 mm^2	2.95 mm^2

extrinsic information throughout the iterations as described in [31]. Another possible improvement to decrease the complexity of the 3D-MSTC is to use another constituent code for the third dimension. For example, a 4-state duobinary trellis can be used to considerably reduce the hardware complexity.

7. CONCLUSION

A new approach to designing low-complexity turbo codes for very low frame error rates and high throughput applications has been proposed. The key idea is to adapt and optimize the technique of multiple turbo codes to obtain the required error frame rate combined with a family of turbo codes, called multiple slice turbo codes (MSTC), that allow high throughput for low hardware complexity. The proposed coding scheme is based on a versatile three-dimensional multiple slice turbo code (3D-MSTC) using 8-state duobinary trellises. Simple deterministic interleavers have been used for the sake of hardware simplicity. An efficient heuristic optimization method of the interleavers has been described, leading to excellent performance. Moreover, by a novel asymmetric puncturing pattern, we have shown that convergence can be traded off against minimum distance (i.e., error floor) in order to adapt the performance of the 3D-MSTC to the requirement of the application. With the asymmetry of the puncturing pattern, it has been shown that the decoding of the third dimension (i.e., the most punctured) should be idled during the first decoding iterations. Two new adapted iterative decoding structures HES and PES have been proposed. Their performance and associated decoder complexities have been compared to 8-state and 16-state duobinary 2D-MSTC. For frame sizes up to 5000 bits, the HES decoding structure achieves lower frame error rates for less complexity than the HPS decoding structure. For a frame size of 4 kb, it has been shown that compared to a two-dimensional turbo code with 8-state trellises, the 3D-MSTC with HES decoding structure achieves a coding gain of more than 0.5 dB at a frame error rate of 10^{-6}, at the cost of a complexity increase of 50%. In addition, compared to a two-dimensional turbo code with 16-state trellises, the proposed asymmetric scheme achieves a better tradeoff performance against complexity. Future work will study the generalization of these promising results to other code rates and other frame sizes.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. IEEE International Conference on Communications (ICC '93)*, vol. 2, pp. 1064–1070, Geneva, Switzerland, May 1993.

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

[3] S. Lin and D. J. Costello, *Error Control Coding Fundamentals and Application*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1983.

[4] C. Berrou, "The ten-year-old turbo codes are entering into service," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 110–116, 2003.

[5] M. C. Valenti, "Inserting turbo code technology into the DVB satellite broadcasting system," in *Proc. 21st Century Military Communications Conference Proceedings (MILCOM '00)*, vol. 2, pp. 650–654, Los Angeles, Calif, USA, October 2000.

[6] J. D. Andersen, "Turbo codes extended with outer BCH code," *Electronics Letters*, vol. 32, no. 22, pp. 2059–2060, 1996.

[7] D. Divsalar and F. Pollara, "Multiple turbo codes for deep-space communications," TDA Progress Report 42-120, pp. 66–77, Jet Propulsion Laboratory, February 1995.

[8] C. Berrou, C. Douillard, and M. Jézéquel, "Multiple parallel concatenation of circular recursive systematic convolutional (crsc) codes," *Annals of Télécomunications*, vol. 54, no. 3-4, pp. 166–172, 1999.

[9] S. Huettinger and J. Huber, "Design of multiple-turbo-codes with transfer characteristics of component codes," in *Proc. Conference on Information Sciences and Systems (CISS '2002)*, Princeton, NJ, USA, March 2002.

[10] N. Ehtiati, M. R. Soleymani, and H. R. Sadjadpour, "Interleaver design for multiple turbo codes," in *Proc. IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '03)*, vol. 3, pp. 1605–1607, Montreal, Quebec, Canada, May 2003.

[11] D. Gnaedig, E. Boutillon, M. Jézéquel, V. C. Gaudet, and P. G. Gulak, "n multiple slice turbo codes," in *Proc. 3rd International Symposiumon Turbo Codes and Related Topics*, pp. 343–346, Brest, France, September 2003.

[12] Y. X. Cheng and Y. T. Su, "On inter-block permutation and turbo codes," in *Proc. 3rd International Symposiumon Turbo Codes and Related Topics*, pp. 107–110, Brest, France, September 2003.

[13] E. Boutillon, J. Castura, and F. R. Kschischang, "Decoder-first code design," in *Proc. 2nd International Symposium on Turbo Codes and Related Topics*, pp. 459–462, Brest, France, September 2000.

[14] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal decoding algorithm in the log domain," in *Proc. IEEE International Conference on Communications (ICC '95)*, vol. 2, pp. 1009–1013, Seattle, Wash, USA, June 1995.

[15] D. Gnaedig, E. Boutillon, V. C. Gaudet, M. Jézéquel, and P. G. Gulak, "On multiple slice turbo codes," published in *Annales des Télécommunications*, January 2005.

[16] S. N. Crozier, "New high-spread high-distance interleavers for turbo-codes," in *Proc. 20th Biennial Symposium on Communications*, pp. 3–7, Kingston, Ontario, Canada, May 2000.

[17] C. Heegard and S. B. Wicker, *Turbo Coding*, Kluwer Academic Publishers, Boston, Mass, USA, 1999, pp. 50–53.

[18] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," TDA Progress Report 42-122, pp. 66–77, Jet Propulsion Laboratory, August 1995.

[19] C. Berrou, S. Vaton, M. Jézéquel, and C. Douillard, "Computing the minimum distance of linear codes by the error impulse method," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM '02)*, vol. 2, pp. 1017–1020, Taipei, Taiwan, November 2002.

[20] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for turbo codes," *Electronics Letters*, vol. 30, no. 25, pp. 2107–2108, 1994.

[21] J. Hokfelt, O. Edfors, and T. Maseng, "Interleaver design for turbo codes based on the performance of iterative decoding," in *Proc. IEEE International Conference on Communications (ICC '99)*, vol. 1, pp. 93–97, Vancouver, BC, Canada, June 1999.

[22] P. C. Massey and D. J. Costello Jr., "New low-complexity turbo-like codes," in *Proc. IEEE Information Theory Workshop*, pp. 70–72, Cairns, Qld., Australia, September 2001.

[23] C. He, A. Banerjee, D. J. Costello Jr, and P. C. Massey, "On the performance of low complexity multiple turbo codes," in *Proc. 40th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Ill, USA, October 2002.

[24] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for three dimensional turbo codes," in *Proc. IEEE International Symposium on Information Theory*, p. 37, Whistler, BC, Canada, September 1995.

[25] S. Crozier and P. Guinand, "Distance upper bounds and true minimum distance results for turbo-codes with DRP interleavers," in *Proc. 3rd International Symposiumon Turbo Codes and Related Topics*, pp. 169–172, Brest, France, September 2003.

[26] C. He, D. J. Costello Jr., A. Huebner, and K. S. Zigangirov, "Joint interleaver design for low complexity multiple turbo codes," in *41st Annual Allerton Annual Allerton Conference on Communications, Control, and Computing*, Monticello, Ill, USA, October 2003.

[27] J. Han and O. Y. Takeshita, "On the decoding structure for multiple turbo codes," in *Proc. IEEE International Symposium on Information Theory*, p. 98, Washington, DC, USA, June 2001.

[28] C. Berrou, E. Maury, and H. Gonzalez, "Which minimum hamming distance do we really need," in *Proc. 3rd International Symposiumon Turbo Codes and Related Topics*, pp. 141–148, Brest, France, September 2003.

[29] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM '89)*, vol. 3, pp. 1680–1686, Dallas, Tex, USA, November 1989.

[30] R. Dobkin, M. Peleg, and R. Ginosar, "Parallel VLSI architecture for MAP turbo decoder," in *Proc. 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '02)*, vol. 1, pp. 384–388, Lisboa, Portugal, September 2002.

[31] R. Hoshyar, A. R. S. Bahai, and R. Tafazolli, "Finite precision turbo decoding," in *Proc. 3rd International Symposiumon Turbo Codes and Related Topics*, pp. 483–486, Brest, France, September 2003.

David Gnaedig was born in Altkirch, France, on August 28, 1978. He received the Engineering Diploma from the École Nationale Supérieure des Télécommunications (ENST), Paris, France, in 2001. Since 2002, he has been working towards his Ph.D. degree with TurboConcept, Brest, France, the Laboratoire d'Electronique et des Systèmes Temps Réels (LESTER), Lorient, France, and the École Nationale Supérieure des Télécommunications (ENST) Bretagne, Brest, France. His Ph.D thesis deals with parallel decoding techniques for high throughput turbo decoders and especially with joint code/architecture design. His research interests also include high throughput iterative decoding techniques in the field of digital communications.

Emmanuel Boutillon was born in Chatou, France, on November the 2nd, 1966. He received the Engineering Diploma from the École Nationale Supérieure des Télécommunications (ENST), Paris, France, in 1990. In 1991, he worked as an Assistant Professor in the École Multinationale Supérieure des Télécommunications in Africa (Dakar). In 1992, he joined ENST as a Research Engineer where he conducted research in the field of VLSI for digital communications. While working as an engineer, he obtained his Ph.D in 1995 from ENST. In 1998, he spent a sabbatical year at the University of Toronto, Ontario, Canada. Since 2000, he has been a Professor at the University of South Britany, Lorient, France. His current research interests are on the interactions between algorithm and architecture in the field of wireless communications. In particular, he works on turbo codes and LDPC decoders.

Michel Jézéquel was born in Saint-Renan, France, on February 26, 1960. He received the "Ingénieur" degree in electronics from the École Nationale Supérieure de l'Électronique et de ses Applications, Paris, France, in 1982. During the period 1983–1986, he was a Design Engineer at CIT ALCATEL, Lannion, France. Then, after gaining experience in a small company, he followed a one-year course about software design. In 1988, he joined the École Nationale Supérieure des Télécommunications de Bretagne where he is currently a Professor Head of the Electronics Department. His main research interest is in circuit design for digital communications. He focuses his activities on the fields of turbo encoding/decoding circuits modelling (behavioural C and synthesizable VHDL), Adaptation of the turbo principle to iterative correction of intersymbol interference, the design of interleavers, and the interaction between modulation and error correcting codes.