Turbo product code decoder without interleaving resource: From parallelism exploration to high efficiency architecture

Camille Leroux, Christophe Jego, Patrick Adde, Deepak Gupta, Michel Jezequel

J Sign Process Syst
DOI 10.1007/s11265-010-0478-5

Turbo Product Code Decoder Without Interleaving Resource: From Parallelism Exploration to High Efficiency Architecture

Camille Leroux · Christophe Jego · Patrick Adde · Deepak Gupta · Michel Jezequel

Received: 13 November 2009 / Revised: 15 March 2010 / Accepted: 15 March 2010

Abstract This article proposes to explore parallelism in Turbo-Product Code (TPC) decoding through a parallelism level classification and characterization. From this design space exploration, an innovative TPC decoder architecture without any interleaving resource is presented. This architecture includes a fully-parallel SISO decoder capable of processing n symbols in one clock period. Syntheses results show the better efficiency of such an architecture compared with existing solutions. Considering a six-iteration turbo decoder of a BCH(32,26)^2 product code, synthesized in 90 nm CMOS technology, 10 Gb/s can be achieved with an area of 600 Kgates. Moreover, a second architecture enhancing parallelism rate is described. The throughput is 50 Gb/s while an area estimation gives 2.2 Mgates. Finally, comparisons with existing TPC decoders and existing LDPC decoders are performed. They validate the potential of proposed TPC decoder for Gb/s optical fiber transmission systems.

Keywords TPC decoding · Parallelism exploration · Ultra-high-speed integrated circuits

1 Introduction

Nowadays, high throughput telecommunication systems such as optical fiber transmission systems or passive optical networks require powerful error correcting codes in order to increase their optical budget. Iterative decoding [1, 2] provides effective solutions for next generation optical systems. Recently, a (660,480) LDPC code decoder ASIC implementation was proposed. The throughput is 2.4 Gb/s while it could be enhanced to 10 Gb/s with a (2048,1723) LDPC code [3]. Turbo product codes [4] also tend to be good candidates for emerging optical systems [5]. In [6], a TPC decoder is included in a 12.4 Gb/s optical experimental setup. Since only a part of the transmitted data is actually encoded, the throughput of the TPC turbo decoder is 156Mb/s.

The inherent parallel structure of the product code matrix confers to TPC a good ability for parallel decoding. Nevertheless, enhancing parallelism rate rapidly induces the use of a prohibitive amount of memory. Many solutions were proposed to efficiently exploit parallelism in TPC decoding. However, TPC decoding provides several level of parallelism and it is not always clear which level is the most efficient.

In [7], we proposed a fully parallel turbo product code decoder without interleaving resource. In this

This paper was presented in part at the IEEE workshop on Signal Processing Systems, October 8–10, Washington, D.C. Metro Area, U.S.A, 2008.

C. Leroux (✉) · C. Jego · P. Adde · D. Gupta · M. Jezequel
Institut TELECOM, TELECOM Bretagne, CNRS Lab-STICC, UMR 3192, Université Européenne de Bretagne, Technopôle Brest-Iroise, 83818-29238 Brest Cedex 3, France
e-mail: camille.leroux@telecom-bretagne.eu

C. Jego
e-mail: christophe.jego@telecom-bretagne.eu

P. Adde
e-mail: patrick.adde@telecom-bretagne.eu

D. Gupta
e-mail: deepak.gupta@telecom-bretagne.eu

M. Jezequel
e-mail: michel.jezequel@telecom-bretagne.eu

Published online: 14 April 2010

paper, we set this architecture in the more general context of parallelism level exploration. We propose a parallelism level taxonomy that helps to classify and characterize parallelism in TPC decoding. Similarly to [8], we provide insights on the benefits that each parallelism level can bring to the architecture performance. From this design space exploration, a parallelism level that has not been fully used in previous work is identified. Then, we propose an architecture of a highly-parallel TPC decoder that efficiently takes advantage of the exploited parallelism.

After a brief introduction of the TPC coding and decoding concept in Section 2, Section 3 defines and characterizes all the parallelism levels in TPC decoding. In Section 4, a review of existing solutions is given before the description of an innovative TPC decoder architecture without any interleaving resource. This original TPC decoder includes a novel fully-parallel SISO decoder architecture which is described in Section 5. Section 6 gives some synthesis results and demonstrates the efficiency of the proposed TPC decoder by comparison with current TPC and LDPC decoders. The interconnection issue is assessed and compared with an equivalent LDPC code decoder implementation.

2 TPC Coding and Decoding Principles

Product codes usually have high dimension which precludes Maximum-Likelihood (ML) soft-decision decoding. Yet, the particular structure of this code family lends itself to an efficient iterative "turbo" decoding algorithm offering close-to-optimum performance at high enough Signal-to-Noise-Ratios (SNRs).

2.1 Product Codes

The concept of product codes is a simple and efficient method to construct powerful codes with a large minimum Hamming distance d using cyclic linear block codes [9]. Let us consider two systematic cyclic linear block codes C_1 having parameters (n_1, k_1, d_1) and C_2 having parameters (n_2, k_2, d_2) where n_i, k_i and d_i $(i = 1, 2)$ stand for code length, number of information symbols and minimum Hamming distance respectively. The product code $P = C_1 \times C_2$ is obtained by placing $(k_1 \times k_2)$ information bits in a matrix of k_1 rows and k_2 columns, coding the k_1 rows using code C_2 and coding the n_2 columns using code C_1, as shown on Fig. 1.

Considering that C_1 and C_2 are linear codes, n_1 rows are codewords of C_2 exactly as all n_2 columns are codewords of C_1 by construction. Furthermore, the parameters of the resulting product code P are given

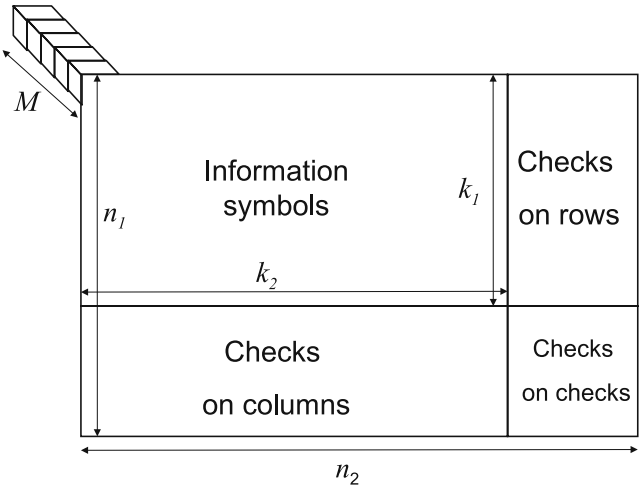

Figure 1 Product code matrix structure.

by $n_p = n_1 \times n_2$, $k_p = k_1 \times k_2$, and $d_p = d_1 \times d_2$ and the code rate R_p is given by $R_p = R_1 \times R_2$. Thus, it is possible to construct powerful product codes using linear block codes. In the following sections, we will consider a squared product code, meaning that $n_1 = n_2 = n$. The most commonly used component codes are Bose Chaudhuri Hocquenghem (BCH) codes. These codes are an infinite class of linear cyclic block codes that have capabilities for multiple error detection and correction. Product codes were adopted in 2001 as an optional correcting code system for both the uplink and downlink of the IEEE 802.16 standard (WiMAX) [10]. Reed-Solomon (RS) codes can also be used as component codes. RS codes are non-binary codes in which symbols are represented on $M_{RS} = log(n + 1)$ bits while $M_{BCH} = 1$. It has been shown that they provide similar performance for a low decoding complexity [11, 12].

2.2 Iterative Decoding of Product Codes

TPC decoding involves successively alternate decoding rows and columns using SISO decoders. Repeating this soft decoding during several iterations enables the reduction of the Bit Error Rate (BER). It is known as the TPC decoding process. Each decoder has to compute soft information $[R']_{it+1}$ from the channel received information $[R]$ and the information $[R']_{it}$ computed during the previous half-iteration.

Despite the existence of several other decoding algorithms [13], the Chase–Pyndiah algorithm is known to give the best tradeoff between performance and decoding complexity [14]. The Chase–Pyndiah SISO algorithm for a $t = 1$ BCH code [4, 15] is summarized

below. t represents the maximum number of correctable errors.

1. Search for the $L = |\lambda_i|$ least reliable bits from the previous half-iteration output vector $[R']_{it}$ such that $\min_i(|[R']_{it}|) = \lambda_i$. $\min_i(.)$ represents the i^{th} minimum function.
2. Compute the syndrome $S(t_0)$ of $[R']_{it}$,
3. Compute the parity of $[R']_{it}$,
4. Generate τ_p test patterns τ_i obtained by inverting some of the L least reliable bits ($\tau_p \leq 2^L$).
5. **For each** test pattern ($1 \leq \tau_i \leq \tau_p - 1$)
   - Compute the syndrome $S(\tau_i)$,
   - Correct the potential error by inverting the bit position $S(\tau_i)$,
   - Recompute the parity considering the detection of an error and the parity of $[R']_{it}$,
   - Compute the square Euclidian distance (metric) M_i between $[R']_{it}$ and the considered test pattern τ_i.
6. Select the Decided Word (DW) among test patterns having the minimal metric (M_{DW}) and choose Cw competitors codewords $\mathbf{c}_i$ ($1 < i < Cw$) having the second minimum metric.
7. **For each** symbol of the DW,
   - Compute the new reliability F_{it}:

$$F_{it} = \begin{cases} \beta_{it} = (|R'_{it}| + \sum_{i=1}^{L} \lambda_i) \\ \qquad - \min(M_i) \text{ when no competitor exists} \\ \\ F_{it} = \min_2(M_i) - \min(M_i) \text{ otherwise,} \end{cases}$$

   - Compute extrinsic information $W_{it} = F_{it} - R'_{it}$,
   - Add extrinsic information (multiplied by α_{it}) to the channel received word, $R'_{it+1} = R + \alpha_{it} W_{it}$.

As explained in [16], decoding parameters L, τ_p, Cw and the number of quantization bits of the soft information q have a considerable effect on decoding performance and complexity. The α_{it} coefficient allows decoding decisions to be damped during the first iterations. β_{it} is an estimation of F_{it} when no competitor exists. As detailed in [17], it is based on the least reliable bits value.

3 Parallel Processing Levels in TPC Decoding

An architecture can be characterized by different metrics such as throughput, latency, hardware complexity, power consumption, routing density, etc. In this study, we aim at high speed architectures with low hardware complexity. Consequently, the performance is measured by throughput (T) while the cost function is the hardware complexity (C). In such a context, the efficiency of an architecture is defined as the throughput/complexity ratio : $E = T/C$. An efficient architecture would process a high data rate at a low hardware complexity.

The parallelism of an architecture can be defined as "*the ability of the system to process several data in parallel*". We formerly define the parallelism P of a decoder as the number of bit that can be processed/decoded in a single clock cycle. Parallelism directly influences the performance of an architecture. In order to quantify the benefit/disadvantage brought by the application of a parallelism P_i to an architecture, we define three metrics, the *speed gain* G_S, the *computational ratio* R_C and the *efficiency gain* G_E:

$$\begin{cases} G_S(P_i = p) = \frac{T_{P_i=p}}{T_{P_i=1}} \\ \\ R_C(P_i = p) = \frac{C_{P_i=p}}{C_{P_i=1}} \\ \\ G_E(P_i = p) = \frac{E_{P_i=p}}{E_{P_i=1}} = \frac{G_S(P_i=p)}{R_C(P_i=p)} \end{cases}$$

A parallelism level P_i is considered to be *effective* if $G_S(P_i) > 1$, while it is *efficient* when $G_E(P_i) > 1 \iff G_S(P_i) > R_C(P_i)$. In the following of this section, all parallelism levels in TPC decoding are detailed and characterized.

3.1 Frame Parallelism

The highest level of parallelism can be observed at the frame level, this is known as *frame parallelism*. It is a form of spatial parallelism and is suitable to any decoding scheme. It consists in duplicating the processing resources, *e.g.* the turbo-decoder. By using this parallelism level in TPC decoding, P_{frame} matrices can be decoded at the same time. Considering P_{frame} turbo-decoders that have the same throughput T_0, the *speed gain* and *complexity ratio* are equivalent: $G_S = R_C = P_{frame}$. Consequently the efficiency does not increase with P_{frame}: $G_E = 1$. Actually, this level of parallelism is only limited by the affordable silicon area. Although frame parallelism do make TPC decoder architecture more effective, it does not improve its efficiency.

3.2 Iteration Parallelism

In a sequential TPC decoder implementation, each iteration is performed by the same decoder that reads

and writes data in the Interleaving Memories (IM). It is however possible to use the *iteration parallelism* by duplicating the elementary decoder in a pipelined structure. The maximum depth of such a structure equals to the maximum number of iteration it_{max}. Iteration parallelism is a type of temporal parallelism. Here again, the throughput benefit equals to the complexity ratio: $G_S = R_C = P_{it}$. It means that the iteration parallelism does not improve efficiency.

3.3 Sub-block Parallelism

In a product code matrix, each row (column) is obtained independently from the others (See Section 2.1). This interesting property can also be used during the decoding process, where each row (column) is decoded independently. In an implementation prospective, it means that more than one decoder can be assigned to row (column) decoding. Considering a product code matrix of size n^2, a maximum number of n decoders can be duplicated for row (column) decoding. We designate this parallelism level as *sub-block parallelism* P_{sb}. In a straightforward application of this parallelism level, one would simply duplicate the SISO decoders and the associated memory resources. It leads to a non-efficient parallelism exploitation, in particular due to the large size of the memory blocks that have to be duplicated P_{sb} times. In [18, 19] solutions are proposed to avoid interleaving resource duplication when P_{sb} increases. This makes the *complexity ratio* lower than the *speed gain*, which means that the *efficiency gain* of the architecture increases with P_{sb}. G_E can be expressed as:

$$G_E = \frac{P_{sb}(C_{SISO} + C_\pi)}{P_{sb} C_{SISO} + C_\pi}$$

$$G_E > 1 \iff P_{sb} > 1$$

C_{SISO} and C_π are the hardware complexity of the SISO decoder and the interleaving resource respectively.

3.4 Symbol Parallelism

A finer-grained parallelism is the *symbol parallelism*. It can be defined as the ability of a SISO decoder, to process P_{sym} symbols of the same sub-block (row or column) in parallel. In a sequential SISO decoder, input data is shifted in a serial manner. Every incoming symbol implies some internal metrics to be updated. By increasing P_{sym}, some parts of the decoder datapath has to be duplicated, (*e.g.* the reliability computation stage). However, the other blocks, such as the test pattern metric computation, or the competitor vector determination block, remain identical when P_{sym} increases. Consequently, the *complexity ratio* is lower than the *speed gain*: $G_E < 1$. Increasing P_{sym} also means that the interleaving memory should be able to read/write more than one data during the same clock cycle. Solutions were provided in [20] to exploit this parallelism while avoiding interleaving memory duplication. Synthesis results confirm that the efficiency increases with P_{sym}. For an architecture that avoid interleaving resource duplication, G_E can be expressed as:

$$G_E > 1 \iff C_{DEC}(P_{sym} = p) < p \times C_{DEC}(P_{sym} = 1)$$

$C_{DEC}(P_{sym} = p)$ is the hardware complexity of a SISO decoder with a symbol parallelism equals to p. In [20], this inequality has been verified by synthesis for $P_{sym} = \{1; 2; 4; 8\}$ for a BCH(32,26)^2 TPC decoder. In this paper, we propose to verify this inequality for the same code and for $P_{sym} = n = 32$. This challenging architecture is described in Section 5.

3.5 Intra-symbol Parallelism

In TPC decoding, BCH codes are often used for their good decoding performance/complexity tradeoff. Recent work [11, 21] has shown that using RS codes as component codes, can provide similar decoding performance at a reasonable computational complexity.

From an architectural point of view, the non-binary structure of RS codes enables to exploit an extra parallelism level, the *intra-symbol parallelism* P_{is}. In a RS code of size n, a symbol consists in $\log(n+1)$ bits (see Fig. 1). A RS-SISO decoder can either shift-in symbols bit by bit or symbol by symbol. It provides a maximal parallelism rate of $\max(P_{is}) = \log(n+1)$.

Similarly to the symbol parallelism, the resource sharing within the RS-SISO decoder increases the efficiency. However the *efficiency gain* provided by P_{is} is hard to estimate because it is highly related to the internal architecture of the SISO decoder. Nevertheless, it is possible to give a condition that guarantees $G_E(P_{is}) > 1$:

$$C(P_{is} > 1) < P_{is} \times C(P_{is} = 1)$$

3.6 Parallelism Levels Comparison

Table 1 summarizes benefits of parallelism levels in TPC decoding. For each parallelism P_i, the maximum *speed gain*, the *efficiency gain* and the P_i value that maximizes the efficiency are given. Frame parallelism is only limited by technological issues (*e.g.* silicon area). This parallelism improves the effectiveness of the archi-

Table 1 Comparison of parallelism levels in TPC decoding.

P_i	$\max(G_S)$	G_E	$\arg(\max(E))$
P_{frame}	∞	$\simeq 1$	$[0; +\infty[$
P_{it}	IT_c	$\simeq 1$	IT_c
P_{sb}	n	≥ 1	n
P_{sym}	n	≥ 1	n
P_{is}	$\log(n+1)$	≥ 1	$\log(n+1)$

tecture; it is straightforward to implement but it does not improve efficiency. Iteration parallelism has the same impact but has an upper bound limited by the maximum number of iteration required by the decoding process.

Application of lower levels of parallelism (P_{sb}, P_{sym} and P_{is}) improves the architecture efficiency. It is even maximized for highest parallelism value. However, the use of these parallelism levels is not as straightforward as P_{frame} and P_{it}. It requires some specific scheduling and/or implementation strategies.

4 Parallel Decoding of Product Codes

Designing turbo-decoder architectures compatible with data rates higher than 10 Gb/s is a challenging issue. In this section we first introduce previous work. Then, the proposed Interleaving-Memory-free (IM-free) TPC decoder architecture is detailed. It jointly uses P_{sym} and P_{sb} and includes fully-parallel SISO decoders that are described in Section 5.

4.1 Previous Work

Many TPC decoder architectures were previously designed. In current architectures, the rebuilding of the product code matrix is necessary between each iteration: memory blocks are used at each half-iteration to read and store $[R']_{it}$ and $[R]$. Each interleaving memory block is then composed of four memories of $n^2 \times (M \times q)$-bits data . This solution has several drawbacks. First, a large amount of memory is required which increases the global latency and the hardware complexity of the design. In addition, increasing the degree of parallelism at the sub-block level produces memory conflicts when several data have to be accessed at the same time. In Table 2, existing architectures are reviewed in terms of achieved parallelism P_i and associated hardware complexity. The hardware complexity is given for interleaving resources (C_π) and decoding resources (C_{Dec}).

In [18], authors suggested to use a barrel shifter between decoding resources and the interleaving memory in order to avoid memory conflicts. This solution enables to use the sub-block parallelism at its highest rate: $P_{sb} = n$. The extra-complexity consists in a simple barrel shifter with a complexity of $O(n\log(n))$. However, it still includes a large amount of interleaving memory.

In [16, 19, 22], an IM-less architecture is detailed and prototyped onto an FPGA device. In this architecture, a particular scheduling of the product code matrix decoding enables the interleaving memories to be replaced by an interconnection network (omega network). This TPC decoder also has a maximal sub-block parallelism ($P_{sb} = n$), while the hardware complexity of the interleaving resources is drastically reduced.

Moreover, in [20] an architecture that uses symbol parallelism in conjunction with sub-block parallelism, was proposed. The idea is to store several product code matrix symbols at the same address and to design elementary decoders able to process $P_{sym} = m$ symbols during the same clock period (denoted as m-decoders). A half-iteration structure includes m decoders each decoding m symbols in one clock period and an interleaving memory of size $4 \times q \times M \times n^2$. This scheme actually exploits symbol parallelism on one dimension of the matrix and sub-block parallelism on the other dimension in such a way that $P_{sb} = P_{sym} = m$. The resulting throughput is $O(m^2)$ while the overhead factor of the decoder complexity is $\sim \frac{m^2}{2}$. In this previous work, the maximum reached parallelism rate was $m^2 = 64$, with $m = 8$ SISO decoders.

These three architectures can reach high parallelism degrees (*i.e.* high throughput) for different hardware cost. A TPC decoder is composed of interleaving resources and decoding resources. More than 50% of the complexity is in the memory for IM-based architecture, while it represents less than 10% for omega network-based structure [19, 22]. On the decoding resources side, increasing the parallelism rate by duplicating computation resources is inefficient since the reuse of available resources is not optimized. In the following section we propose a more efficient architecture that keeps a memory-less interleaver and uses both symbol and sub-block parallelism in the decoding stage.

Table 2 Current TPC decoder architecture comparison.

Architecture	P_i	$C_\pi(1/2iter)$	$C_{Dec}(1/2iter)$
[18]	$P_{sb} = n$	$O(4qn^2) + O(n\log(n))$	$O(n)$
[16, 19]	$P_{sb} = n$	$O(n\log(n))$	$O(n)$
[20]	$P_{sb} = m$; $P_{sym} = m$	$O(4qn^2)$	$O(m^2/2)$

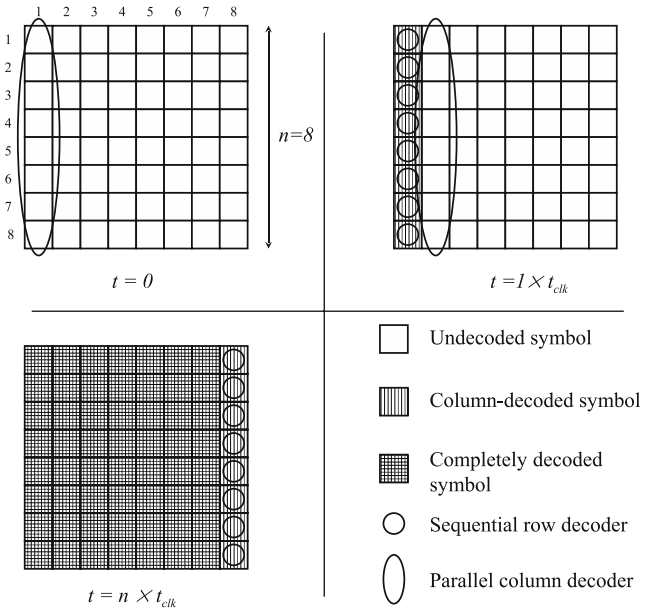

Figure 2 Proposed parallel decoding scheduling of a product code matrix.

4.2 Proposed IM-Free Architecture Using Fully-Parallel SISO Decoder

Considering that one can design a $P_{sym} = n$ SISO decoder, a product code matrix can be decoded without any interleaving resource as shown in Fig. 2.

At $t = 0$, the fully-parallel SISO decoder processes the column 1. During the next clock period, n sequential SISO decoders ($P_{sym} = 1$) start decoding the first symbol of each row while the parallel decoder process the column 2. During the n^{th} clock period, sequential decoders complete matrix decoding while the parallel decoder is already decoding the next matrix.

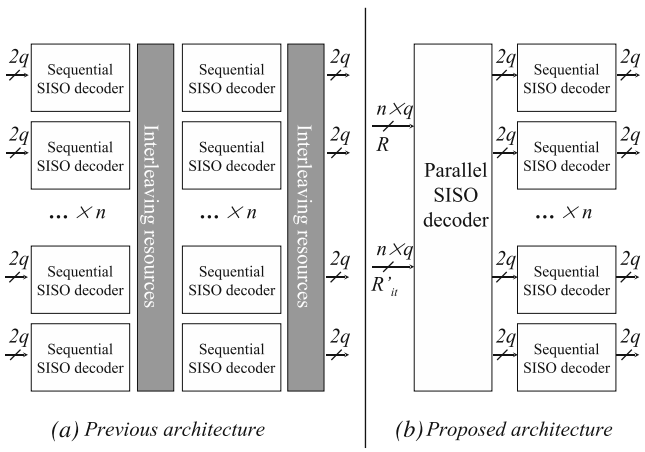

Figure 3 Previous TPC decoder architecture (**a**) and proposed fully-parallel SISO based TPC decoder architecture (**b**).

Data generated by the parallel decoder is immediately used by a sequential decoder. Consequently, no IM or data routing resources are required between the fully-parallel decoder and sequential decoders. The resulting proposed architecture and the previous architectures [18, 19] for one iteration are depicted on Fig. 3.

This architecture uses row-wise P_{sb} and column-wise P_{sym}. More specifically, we have:

$$\begin{cases} P_{sym}(col) = P_{sb}(row) = n \\ P_{sb}(col) = P_{sym}(row) = 1 \end{cases}$$

One should notice that $P_{sb}(col) = P_{sym}(row)$ can be further exploited.

4.3 Towards a Maximal Parallelism Rate

Starting from the IM-free architecture presented in the previous section, parallelism can be further enhanced. Figure 4 shows the alternate product code matrix parallel decoding scheme in which $P_{sb}(col) = P_{sym}(row) = m$ and $P_{sym}(col) = P_{sb}(row) = n$. The TPC decoder consists in $m\times$ n-decoders for column decoding and $n\times$ m-decoders for row decoding. A m-decoder can process m symbols in one clock period and $1 \leq m \leq n$. In such an architecture, the maximum reachable parallelism rate $P = n^2$ can be achieved by using n fully-parallel SISO decoders for column decoding and n fully-parallel SISO decoders for row decoding. Intra-symbol parallelism can also be used to increase the total parallelism to $P = P_{sb} \times P_{sym} \times P_{is} = n^2 \log(n)$.

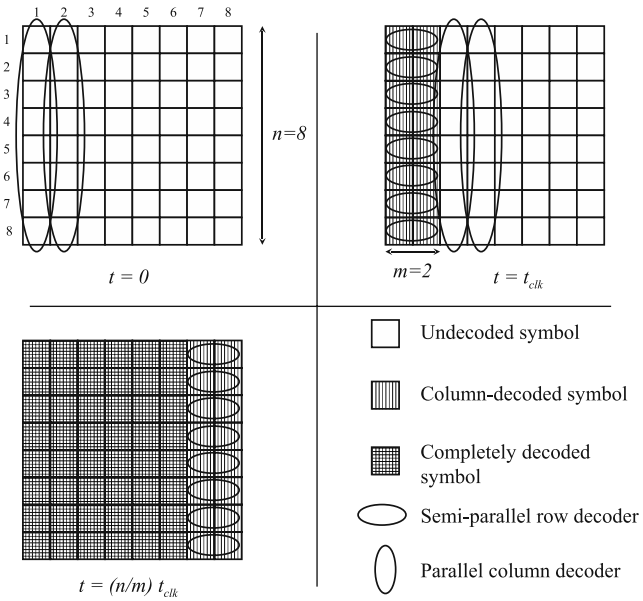

Figure 4 Alternative turbo decoding scheduling for enhanced parallelism rate.

All these new schemes however require to design a SISO decoder capable of processing n symbols in one clock period.

5 Architecture of a Fully-Parallel Combinatory SISO Decoder

The proposed IM-free TPC decoder architecture requires a fully-parallel combinatorial SISO decoder. To the best of our knowledge, only sequential SISO decoders able to process $m \leq n$ symbols in one clock period have been previously designed. The design of a fully-parallel combinatorial SISO decoder is a challenging issue for designer. In the following section, such an architecture is proposed and described.

5.1 Algorithmic Parameter Reduction

As explained earlier in Section 2, the Chase–Pyndiah algorithm includes parameters (L, τ_p, Cw, q) which impact on both the performance and the complexity of the turbo decoding. BER simulation were performed with different parameters: $L = \{2; 3; 4; 5\}$, $\tau_p = \{4; 8; 16\}$, $Cw = \{0; 1; 2; 3\}$, $q = \{3; 4; 5\}$. Performing eight iterations, the parameter set $\mathcal{P}_0 = \{L = 5, \tau_p = 16, Cw = 3, q = 5\}$ gives the best performance for a maximal complexity [14]. However, algorithmic simulations showed that the reduced parameter set $\mathcal{P}_1 = \{L = 3, \tau_p = 8, Cw = 0, q = 5\}$ only induce a performance loss of 0.25dB at BER= 10^{-6} while it becomes null below BER = 10^{-9}. Further reducing these parameters would induce a notable performance loss. For example by simply reducing the number of test patterns: $\mathcal{P}_2 = \{L = 2, \tau_p = 4, Cw = 3, q = 5\}$, the performance loss reaches 0.5dB. Consequently, using $\mathcal{P}_1$ enables the architecture to be simplified at very low performance lost below BER=10^{-9}.

5.2 Fully-Parallel SISO Decoder Architecture

Figure 5 depicts the architecture of the fully-parallel SISO decoder. In the first attempt a purely combinatorial designed was conceived. Later, a critical path study mandated the insertion of pipeline stages within the structure. The SISO decoder is split in three stages, namely the reception stage, the test pattern processing stage and the soft output computation stage.

5.2.1 Reception Stage

The reception stage corresponds to steps (1–3) of the Chase–Pyndiah algorithm detailed in Section 2.

The syndrome of the incoming vector R'_{it} can be derived as $S(R'_{it}) = H \times sign(R'_{it})$ where H is the parity check matrix of the BCH code. A straightforward implementation of such a matrix multiplication is depicted on Fig. 6. The H matrix, the corresponding parity check equations and the syndrome $S(t_0) = [s_2, s_1, s_0]$ implementation of a BCH(7,4) code are detailed.

It can be noticed that some parity check equations have similar terms. For instance, the term $(x_1 \oplus x_0)$ is used in both s_1 and s_2 computation. This enables a reuse of computation resources for an even more efficient implementation. The parity of the incoming vector R'_{it} is computed with a similar structure by "xoring" $(n-1)$ incoming bits. Selecting the least reliable bits among the incoming vector in parallel requires a sorting network. Such structures are composed of interconnected Compare and Select operators (CS). The interconnection scheme depends on the considered sorting algorithm. Many parallel sorting algorithm are conceivable [23]. However, most of them are optimized for a complete sorting, while the Chase–Pyndiah algorithm only requires a partial sorting (*i.e.* extracting L minima). Consequently we devised a network optimized, in terms of area and critical path, for the partial sorting of $L = 3$ values among $n = 32$, as depicted on Fig. 7. The

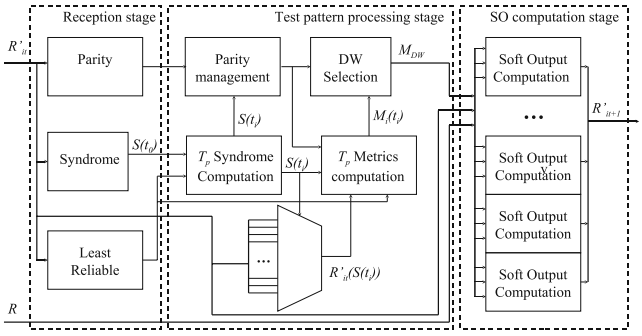

Figure 5 Combinatorial version of the fully-parallel SISO decoder.

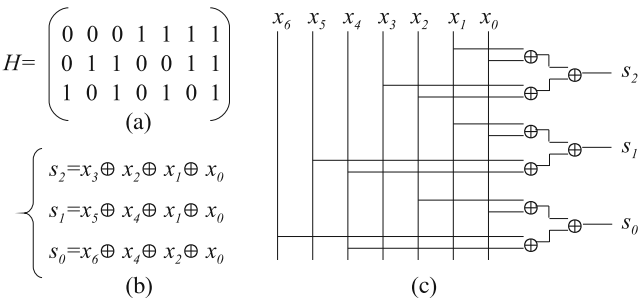

Figure 6 BCH(7,4) code: **a** Parity check matrix **b** Parity check equations **c** Syndrome parallel computation implementation.

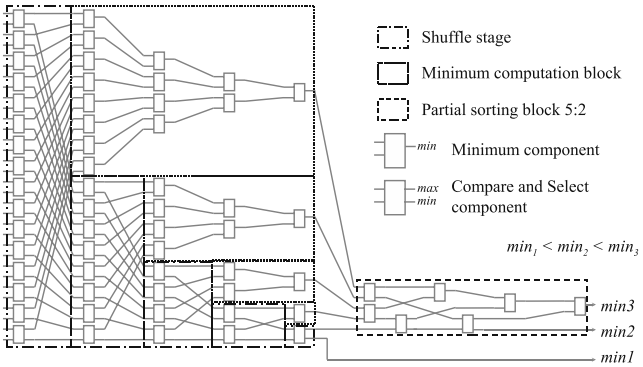

Figure 7 Optimized sorting network for least reliable bits selection.

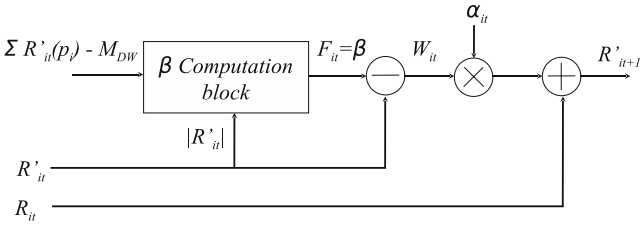

Figure 8 Soft output computation stage.

structure is based on shuffle networks coupled with local minima computation blocks. After the first shuffle stage, min_1 is in the lower section while the upper section can either contain min_2 or min_3 or no minimum. The same reasoning is applied recursively. After five shuffle stages, the minimum is determined while five values can still be min_2 and min_3. A local sorting of this five values enables the determination of min_2 and min_3 value. This partial sorting network requires 35 CS and 29 minimum elements. The critical path consists of nine comparison stages.

5.2.2 Test Pattern Processing Stage

The test pattern processing stage corresponds to steps (4–5) in the Chase–Pyndiah algorithm detailed in Section 2. Instead of being processed sequentially, test patterns are processed in parallel. The syndrome of each test pattern is computed by adding $S(t_0)$ with the position of the inverted reliable bits. The parity management block computes the parity of R'_{it+1} considering the parity of R'_{it} and the detection of an error which is the case when $S(t_i) \neq 0$. Metrics of each test pattern is then computed by adding the contribution of each inverted bit in the current test pattern (least reliable bits, syndrome corrected bits and, the new parity bit). The minimum metric is determined in the DW selection block. The structure is a simple minimum selection tree. The multiplexer selects $R'_{it}(S(t_i))$ in order to compute test pattern metrics.

5.2.3 Soft Output Computation Stage

The last stage is a duplication of n soft output computation blocks. As shown in Fig. 8, this block first computes the new reliability F_{it} of each symbol. Since, no competitor word is considered, the β value is automatically assigned. The β value is based on an estimation of the competitor word metric value. It is calculated from the reliability of the corrected bit and the least reliable bits. Then, the extrinsic information is computed and damped by the coefficient α_{it} which is devised to be a power of 2 making the multiplication a simple bit shifting. Finally, the channel information is added to generate the soft output R'_{it+1}. Within this block, all computation are performed in sign and magnitude format. Other arithmetic format were explored but the chosen one requires less computation resources than others.

6 Comparison with Existing TPC and LDPC Decoders

In classical iterative decoders, the hardware complexity corresponds to the cumulative area of computational resources and memory resources. In proposed IM-free architecture, only the hardware complexity of the SISO decoders have to be considered. Consequently, the following synthesis results will only focus on hardware complexity of SISO decoders. Moreover, some comparisons with current TPC and LDPC decoders are given to demonstrate the potential of the proposed TPC decoder.

6.1 BCH(32,26) SISO Decoder Logic Synthesis Results

In Section 3, we demonstrated that exploiting symbol parallelism is efficient if $C_{DEC}(P_{sym} = p) < p \times C_{DEC}(P_{sym} = 1)$.
In order to verify this inequality, we compare one parallel ($P_{sym} = n$) BCH SISO decoder *vs* $n \times$ sequential ($P_{sym} = 1$) SISO decoders. Five versions of the BCH(32,26) parallel SISO decoder were designed and have from one to five pipeline stages. The one-pipeline stage version is a fully-combinatorial architecture with register banks only at the input and output stages. Table 3 summarizes synthesis results of the five different parallel SISO decoders and compare them with $n = 32$ duplicated sequential SISO decoders. s is the number of pipeline stages inserted in the SISO decoder, $f_{\max}$ is the maximum frequency reached dur-

Table 3 Comparison of parallel and sequential BCH(32,26) SISO decoder performance.

s	Parallel SISO decoder ($P_{sym} = 32$)					32 sequential SISO decoders ($P_{sym} = 1$)
	1	2	3	4	5	3
$f_{\max}$(Mhz)	125	333	500	500	714	700
T(Gb/s)	4.0	10.7	16.0	16.0	22.9	22.4
A (Kgates)	18	26	31	26	34	200
E (Mb/s/gate)	0.15	0.27	0.34	0.41	0.44	0.07
G_E	2.1	3.9	4.9	5.9	6.3	1

ing synthesis, the throughput T is calculated such as $T = P \times f_{max}$, A represents the area of the design in equivalent gate count and E is the efficiency: $E = \frac{T}{A}$. Logic syntheses were performed using Synopsys Design Compiler with a ST-microelectronics 90 nm CMOS process. The area is transposed in logic gate count. One equivalent logic gate corresponds to the area of a two-input NAND gate. It enables a more technology-independent measure of the hardware complexity.

As expected, the maximum frequency of the combinatorial decoder ($s = 1$) is lower than a sequential version. However, by inserting pipeline stages inside the combinatorial structure, the same frequency is reached with $s = 5$. For this last version, the throughput is even higher than n sequential SISO decoders. The hardware cost of the pipeline stages insertion depends on registers location in the decoder. This is the reason why $A(s = 4) < A(s = 3)$. In this particular case, having $s = 4$ pipeline stages enables register stages to be assigned at regular intervals, for a lower hardware cost. In terms of efficiency, a parallel SISO decoder can reach the same throughput as n sequential SISO decoders with a six times lower complexity. The efficiency gain increases with s.

These synthesis results demonstrate the higher efficiency of parallel SISO decoding for the code BCH(32,26). Now, if one consider larger code with the same correction power (*i.e.* BCH(64,57), BCH(128,120)), the complexity of the reception stage and the soft output computation stage would grow linearly with the code size n. However the complexity of the test pattern processing stage would only increase linearly with $\tau_p < n$. Consequently, the overall complexity of the parallel SISO decoder is lower than a duplication of n sequential SISO decoders. It confirms that a fully-parallel SISO decoder enables a better reuse of computation and memory resources and makes the whole TPC decoder more efficient.

One should notice that, for higher correction power ($t > 1$), the algebraic decoding requires more complex algorithms such as Berlekamp–Massey algorithm [24, 25] which make the decoder complexity significantly higher. This is the reason why $t = 1$ codes were selected is this study.

Table 4 Comparisons with current TPC decoders and LDPC decoders.

Decoder features	Code	P_i	P_{total}	it_{max}	T (Gb/s)	Area (Mgates)	E (Kb/s/gate)	Coding gain (dB) @BER=10^{-9}
This work	BCH(32,26)^2	$P_{sym} = 4$, $P_{sb} = 32$	128	6	10.7	0.4	26.8	8.0
	BCH(32,26)^2	$P_{sym} = 32$, $P_{sb} = 32$	1,024	6	85.3	2.0	42.7	8.0
Barrel shifter + IM [18]	BCH(32,26)^2	$P_{sb} = 32$, $P_{frame} = 4$	128	6	10.7	2.6	4.1	8.4
Omega network + no IM [19]	BCH(32,26)^2	$P_{sb} = 32$, $P_{frame} = 4$	128	6	10.7	1.6	6.7	8.4
	BCH(64,57)^2	$P_{sb} = 64$, $P_{frame} = 2$	128	6	10.7	2.0	5.4	8.6
	BCH(128,120)^2	$P_{sb} = 128$	128	6	10.7	2.7	4.0	8.7
Multi-data access IM [20]	BCH(32,26)^2	$P_{sym} = 8$, $P_{sb} = 8$, $P_{frame} = 2$	128	6	10.7	3.5	3.1	8.4
Omega network + no IM [11]	RS(15,13)^2	$P_{is} = 4$, $P_{sb} = 15$, $P_{frame} = 2$	120	6	10	0.3	33.3	8.4
	RS(31,29)^2	$P_{is} = 5$, $P_{sb} = 31$	155	6	12.9	0.8	16.1	8.4
	RS(63,61)^2	$P_{is} = 3$, $P_{sb} = 63$	378	6	15.8	1.3	12.1	7.5
Commercial RS decoder (ASICS ws)	RS(255,239)	$P_{is} = 8$, $P_{frame} = 4$	32	X	10.7	0.12	89	5.0
Omega network + no IM [11]	RS(31,29)^2	$P_{is} = 5$, $P_{sb} = 31$	155	1	35	0.4	95	5.2
Commercial TPC decoder (Mitsubishi)	BCH(144,128) ×BCH(256,239)	?	?	4	10.0	18.0	0.6	10
LDPC decoder [3]	LDPC(2048,1723)	?	64	8	16.0	2.2	7.2	7.5
LDPC decoder [26]	LDPC(1440,1344)	?	360	8	6.1	0.4	15.3	5.5

6.2 Comparison with Existing TPC Decoder Architectures

Table 4 compares performance of the proposed solution with current architectures in a ultra-high-throughput context ($T > 10$ Gb/s). For each solution, the decoder architecture main features, the targeted code, the levels of parallelism that were used in order to reach $T = 10$ Gb/s, the resulting total parallelism ($P_{total} = \prod_i P_i$), the maximum number of iteration it_{max} are given. We consider that one iteration is actually implemented. The resulting throughput is $T = P_{total} \times f_{max}/it_{max}$. Finally, the gate count (A), the efficiency ($E = T/A$) and the achieved coding gain at BER=10^{-9} are given. Such a low BER is usually targeted in very high speed application (*e.g.* data transmission over Passive Optical Networks).

For a fair comparison, architectures described in [11, 18–20] were synthesized with the same technology: ST Microelectronics, CMOS 90 nm with a target frequency $f_{max} = 500$ MHz. For the remaining architectures, we gathered information from the published papers and technical reports.

Two versions of the proposed turbo decoder were synthesized. The first one consists in 4 parallel SISO decoders together with 32 $P_{sym} = 4$-SISO decoders. The reached throughput is then sufficient for 10 Gb/s applications. The second version uses only fully-parallel SISO decoder, 32 of such decoders are duplicated for each half-iteration. The maximum throughput is 85 Gb/s for the best efficiency.

The barrel-shifter-based solution [18] can achieve 10 Gb/s with 2.6 Mgates. In order to reach a sufficient parallelism level, it was necessary to use frame parallelism. The efficiency of this solution is six times lower than the proposed architecture. This low efficiency is mainly due to the use of interleaving memory.

For the same reason, the TPC decoder with multi-access data [20] has a low efficiency and also requires the use of frame parallelism to achieve 10 Gb/s.

In [19], the elimination of interleaving memories improves the efficiency but the maximum parallelism rate is limited by the code size n, which makes the use of frame parallelism mandatory in a ultra high speed context. However, the proposed solution provides a maximum parallelism rate of n^2.

The study in [11] shows that RS-TPC are a practical solution for 10 Gb/s transmission over optical networks. As we mentioned in Section 3, using RS codes enables the use of intra-symbol parallelism. With an omega-network-based architecture, this solution also presents good efficiency gain for similar decoding performance. One should notice that the proposed fully-parallel architecture is applicable to RS decoding as well. We expect that the application of intra-symbol parallelism would further increase the overall efficiency of the TPC decoder. Moreover. when comparing a single iteration of RS-TPC decoding with a commercial RS(255,239) code decoder, one can observe that superior efficiency is achieved for slightly better decoding performance.

Mitsubishi has recently proposed a TPC decoder for 10 Gb/s optical transmissions. The component code is a BCH(144,128)×BCH(256,239). These codes are more powerful than $t = 1$ BCH codes that are used in this study, however the implementation is very costly in terms of hardware complexity. Indeed, 18 Mgates are necessary to implement such a decoder, which makes the efficiency very small. This is the cost that have to be paid for a 2dB extra coding gain provided by this TPC decoder.

6.3 Comparison with Current LDPC Decoders

As a matter of comparison, we gathered results from recent academic implementation of LDPC decoders. Since the code rates are different, the decoding performance comparison is not straightforward. However the architectural results help to locate the different decoders in the design space. Synthesis results show that a fully-parallel TPC decoder achieves higher throughput (85 Gb/s) than comparable LDPC decoders with a lower hardware complexity (2 Mgates).

However, placing and routing $2n$ parallel SISO decoders onto the same chip would reduce the maximum working frequency of the parallel SISO decoder. Nevertheless, with P_{total} =1,024, throughput T=10 Gb/s is reached when $f > 88$ MHz which is a most probably achievable frequency on an ASIC target. Furthermore, a reasonable working frequency of f=300 MHz leads to a BCH(32,26)^2 product code turbo decoder with an throughput T =50 Gb/s. The total area of such a parallel turbo decoder is $A = 2.2$ Mgates= 10 μm^2. In parallel LDPC code decoder, one of the main issues is the routing congestion induced by the Tanner graph implementation. The number of interconnections among the TPC decoder is $I_n(\text{TPCD}) = 2 \times n_{BCH}^2 \times q$ while an equivalent fully parallel 1024-LDPC decoder would have $I_n(\text{1024-LDPC}) = n_{LDPC} \times q \times d_v$ where d_v represents the variable node degree. Consequently, as long as $d_v > 2$, the following inequation is verified: $I_n(TPCD) < I_n(1024 - LDPC)$. Consequently, despite the high parallelism level that can be reached in TPC decoding, both area and routing congestions are in favor of TPC decoders.

7 Conclusion

TPC decoding is a realistic solution for next generation high throughput optical communications such as long-haul optical transmissions or passive optical networks. The structure of the product codes makes them very suitable for parallelisation, however the exploitation of some parallelism levels may not be efficient in terms of throughput/complexity ratio. This is particularly true when interleaving memory has to be duplicated.

In this paper we proposed to review and characterize all parallelism levels in TPC decoding. This analysis helps to better understand and classify existing TPC decoders. In previous TPC decoders, high throughput architecture complexity is made prohibitive by the amount of memory usually required for data interleaving and pipelining.

After this design space exploration, we propose an innovative architecture that jointly exploit sub-block parallelism and symbol parallelism. This novel structure enables any interleaving resource to be removed. The proposed TPC decoder requires a fully-parallel SISO decoder capable of processing n symbols in one clock period. Such a SISO decoder architecture is described and includes a new optimized parallel sorting network.

ASIC-based logic syntheses confirm the better efficiency of the proposed IM-free TPC decoder architecture compared to existing TPC decoders and recent LDPC decoders. Actually, when compared to previous work, the area is reduced while the same throughput is achieved. A BCH$(32,26)^2$ product code can be decoded at 33.7 Gb/s with an estimated silicon area of $10\mu m^2$.

References

1. Gallager, R. G. (1962). Low density parity check codes. *IRE Transactions on Information Theory, IT*, 21–28.
2. Berrou, C., Glavieux, A., & Thitimajshima, P. (1993). Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *IEEE international conference on communications 1993, ICC 93, Geneva, 23–26 May 1993*.
3. Darabiha, A., Carusone, A. C., & Kschischang, F. R. (2007). A 3.3-Gbps bit-serial block-interlaced min-sum LDPC decoder in 0.13-um CMOS. In *IEEE conference on custom integrated circuits, 2007, CICC '07*.
4. Pyndiah, R., Glavieux, A., Picart, A., & Jacq, S. (1994). Near optimum decoding of product codes. In *IEEE global telecommunications conference, 1994, GLOBECOM '94*.
5. Mizuochi, T., Kubo, K., Yoshida, H., Fujita, H., Tagami, H., Akita, M., et al. (2003). Next generation FEC for optical transmission systems. In *Optical fiber communications conference, 2003, OFC 2003*.
6. Mizuochi, T., Ouchi, K., Kobayashi, T., Miyata, Y., Kuno, K., Tagami, H., et al. (2003). Experimental demonstration of net coding gain of 10.1 dB using 12.4 Gb/s block turbo code with 3-bit soft decision. In *Optical fiber communications conference, 2003, OFC 2003, 23–28 March*.
7. Leroux, C., Jego, C., Adde, P., & Jezequel, M. (2008). A highly parallel turbo product code decoder without interleaving resource. In *SiPS 2008: IEEE workshop on signal processing systems, 8–10 October, Washington, D.C. Metro Area, U.S.A*.
8. Muller, O., Baghdadi, A., & Jezequel, M. (2006). Exploring parallel processing levels for convolutional turbo decoding. In *ICCTA'06: IEEE international conference on information and communication technologies: From theory to applications, 24–28 April, Damas, Syria* (pp. 2353–2358).
9. Elias, P. (1954). Error-free coding. *IEEE Transactions on Information Theory, 4*(4), 29–37.
10. IEEE Standard 802.16-2001 (2001) IEEE standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems.
11. Bidan, R. L., Leroux, C., Jego, C., Adde, P., & Pyndiah, R. (2008). Reed–Solomon turbo product codes for optical communications: From code optimization to decoder design. *EURASIP Journal on Wireless Communications and Networking, 2008*, 909–912.
12. Leroux, C., Jego, C., Adde, P., & Jezequel, M. (2008). On the higher efficiency of parallel Reed–Solomon turbo-decoding. In *ICECS'08: 15th international conference on electronics, circuits and system, 31st August–3rd September*.
13. Forney, J. G. (1966). Generalized minimum distance decoding. *IEEE Transactions on Information Theory, IT-12*, 125–131.
14. Adde, P., Pyndiah, R., & Raoul, O. (1996). Performance and complexity of block turbo decoder circuits. In *Proceedings of the third IEEE international conference on electronics, circuits, and systems, 1996. ICECS '96, 13–16 October 1996* (Vol. 1, pp. 172–175).
15. Chase, D. (1972). A class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory, IT*, 170–182.
16. Leroux, C., Jego, C., Adde, P., & Jezequel, M. (2007). Towards Gb/s turbo decoding of product code onto an FPGA device. In *IEEE international symposium on circuits and Systems, 2007. ISCAS 2007, 27–30 May 2007* (pp. 909–912).
17. Adde, P., & Pyndiah, R. (2000). Recent simplifications and improvements in block turbo codes. In *2nd international symposium on turbo codes & related topics, 4–7 September, Brest, France* (pp. 133–136).
18. Chi, Z., & Parhi, K. (2002). High speed VLSI architecture design for block turbo decoder. In *IEEE international symposium on circuits and systems, 2002. ISCAS 2002, 26–29 May 2002* (Vol. 1, pp. I-901–I-904).
19. Jego, C., Adde, P., & Leroux, C. (2006). Full-parallel architecture for turbo decoding of product codes. *Electronics Letters, 42*, 55–56.
20. Cuevas, J., Adde, P., Kerouedan, S., & Pyndiah, R. (2002). New architecture for high data rate turbo decoding of product codes. In *Global telecommunications conference, 2002. GLOBECOM '02. IEEE, 17–21 Nov. 2002* (Vol. 2, pp. 1363–1367).
21. Piriou, E., Jego, C., Adde, P., Le Bidan, R., & Jezequel, M. (2006). Efficient architecture for Reed Solomon block turbo code. In *2006 IEEE international symposium on circuits and systems, 2006. ISCAS 2006. Proceedings, 21–24 May 2006* (4pp.).

22. Leroux, C., Jego, C., Adde, P., & Jezequel, M. (2009). High-throughput block turbo decoding: From full-parallel architecture to FPGA prototyping. *Journal of Signal Processing Systems, 57*, 349–361.
23. Akl, S. G. (1985). *Parallel sorting algorithms*. New York: Academic.
24. Berlekamp, E. R. (1984). *Algebraic coding theory, revised edition*. Laguna Hills: Aegean.
25. Massey, J. L. (1969). Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory, IT*, 122–127.
26. Yamagishi, H., & Noda, M. (2008). High throughput hardware architecture for (1440,1344) low-density parity-check code utilizing quasi-cyclic structure. In *2008 5th international symposium on turbo codes and related topics* (pp. 78–83).

Christophe Jego was born in Auray, France, in 1973. He received the M.S. and Ph.D. degrees from the Université Rennes 1, Rennes, France, in 1996 and 2000, respectively. He joined the Electronic Engineering Department of TELECOM Bretagne as a full-time Associate Professor in 2001. He was a visiting professor in the Department of Electrical and Computer Engineering at McGill University during 10 months (Sept. 2006 – June 2007). In 2009, he received Research Habilitation from University of Bretagne Sud. It is the highest French university diploma passed after a few years of active research and student supervision. His research activities are concerned with analysis and design of architectures for iterative processing in the digital communication systems.

Camille Leroux was born in Vannes, France, in 1981. He received his M.S. in Electronics Design and Systems Architecture from the University of South Brittany in 2005. He performed his Ph.D. (2005–2008) in the Electronic Engineering Department at TELECOM Bretagne, France. He is currently a Postdoc fellow in the Department of Electrical and Computer Engineering at McGill University, Montréal, Canada. His research interest focus on hardware implementation of iterative decoding algorithms.

Patrick Adde was born in Caen, France, in 1953. He received the degree of "Ingénieur" from the "École Nationale d'ingénieur de Brest", Brest, France in 1974. In 1979, he joined the École Nationale Supérieure des Télécommunications de Bretagne, where he is currently Professor. His research interests are about the design of efficiency architectures for turbo decoding of product codes.

Deepak Gupta was born in Haryana, India in 1983. He received his Bachelor of Technology degree from Guru Gobind Singh Indraprastha University, New Delhi, India in 2005. In 2008, he received M.Sc. degree in IC design and communications from TELECOM Bretagne, France during which he worked on channel coding and frequency multiplier architectures. Currently, he is working towards a Ph.D. in ultra-low power SoC design for biomedical applications in the Electronics department, Telecom Bretagne, France.

Michel Jezequel (M'02) was born in Saint Renan, France, on February 26, 1960. He received the degree of "Ingénieur" in electronics from the "École Nationale Supérieure de l'Électronique et de ses Applications", Paris, France in 1982. In the period 1983–1986 he was a design engineer at CIT ALCATEL in Lannion, France. Then, after an experience in a small company, he followed a one year course about software design. In 1988, he joined the TELECOM Bretagne, where he is currently Professor, head of the Electronics Department. His main research interest is circuit design for digital communications. He focuses his activities in the fields of Turbo codes, adaptation of the turbo principle to iterative correction of intersymbol interference, the design of interleavers and the interaction between modulation and error correcting codes.