High-throughput block turbo decoding: from full-parallel architecture to FPGA prototyping

Camille Leroux, Christophe Jego, Patrick Adde, Michel Jezequel

J Sign Process Syst (2009) 57:349–361
DOI 10.1007/s11265-008-0316-1

High-throughput Block Turbo Decoding: From Full-parallel Architecture to FPGA Prototyping

Camille Leroux · Christophe Jégo · Patrick Adde · Michel Jézéquel

Received: 1 April 2008 / Accepted: 28 October 2008 / Published online: 4 December 2008

Abstract Ultra high-speed block turbo decoder architectures meet the demand for even higher data rates and open up new opportunities for the next generations of communication systems such as fiber optic transmissions. This paper presents the implementation, onto an FPGA device of an ultra high throughput block turbo code decoder. An innovative architecture of a block turbo decoder which enables the memory blocks between all half-iterations to be removed is presented. A complexity analysis of the elementary decoder leads to a low complexity decoder architecture for a negligible performance degradation. The resulting turbo decoder is implemented on a Xilinx Virtex II-Pro FPGA in a communication experimental setup which also includes an innovative parallel product encoder. The implemented block turbo decoder processes input data at 600 Mb/s. The component code is an extended Bose, Ray-Chaudhuri, Hocquenghem (eBCH(16,11)) code. Some solutions to reach even higher data rates are finally presented.

Keywords Block turbo decoding · Full-parallel architecture · Complexity analysis · FPGA implementation

1 Introduction

In telecommunications, forward error correction (FEC) is a system of error control that improves digital communication quality. Error correction encoding is the addition of redundancy into the binary information sequence that is to be transmitted over the communication channel. This redundancy allows the error correction decoder to detect and/or to correct the effects of noise and interference encountered in the transmission of the information through the communication channel. Turbo coding was a powerfull improvement in error correction systems [1]. This family of codes consists of two key design innovations: concatenated encoding and iterative decoding. Soft Input Soft Output (SISO) decoders are used in the iterative decoding process. A SISO decoder both receives soft decision data and produces soft decision output. The general concept of the iterative SISO decoding of concatenated convolutional codes has been extended to product codes [2] and LDPC codes [3]. Moreover, the principle of turbo processing has been extended into new receiver topologies such as turbo detection, turbo equalization, turbo-coded modulation, turbo MIMO, etc. Since 1999, turbo codes have been adopted by several applications. They are particularly attractive for cellular communication systems and have been included in the specifications for both the UMTS and CDMA2000 third-generation cellular standards. Similarly, turbo codes were chosen in the field of Digital Video Broadcasting (DVB) and Wireless Metropolitan Area Networks (IEEE 802.16) to increase transmission rates and/or to guarantee Quality of Service (QoS). Currently, research is under way to use turbo codes to protect data stored on hard drive or DVD and in fiber optical transmission. Fiber optical communications are an important technology in supporting broadband networking. The earliest FEC for optical communication [4] employed the well known Reed-Solomon (RS) codes to recover the

C. Leroux (✉) · C. Jégo · P. Adde · M. Jézéquel
TELECOM Bretagne, Institut TELECOM,
CNRS Lab-STICC FRE 3167, Brest, France
e-mail: camille.leroux@telecom-bretagne.eu

degradation in bit error rate (BER) due to the effects of fiber nonlinearity and polarization-dependent phenomena. The standard interfaces and rates are specified in the ITU-T G.709 for Optical Transport Networks (OTN). A G.709 OTN frame includes an RS(255,239) code that helps reduce the number of transmission errors on noisy links, which enables the increase in transmission capacity. A net coding gain of around 6dB is provided by the RS(255,239) code. During the last few years, more powerful FECs have been studied to increase the net coding gain. Block Turbo Codes (BTC) have a theoretical potential net coding gain of around 10 dB with a redundant overhead of less than 25 % [5, 6]. Typically, realistic block turbo codes can operate at less than 1dB from the Shannon limit for a binary symmetric channel. Very high-speed data transmission developed for fiber optical networking systems have necessitated the implementation of ultra high-speed FEC architectures to meet the continuing demands for even higher data rates. Currently, the RS(255,239) code can be used in ultra high-speed (40 Gb/s [7] and 80 Gb/s [8]) fiber optic systems. However higher coding gain can be reached using turbo product codes. In 2002, a new architecture for turbo decoding product codes was presented [9]. This architecture can theoretically achieve a throughput of 6.4 Gb/s on an ASIC target, but no concrete realisation were performed. In 2005, Mitsubishi Electric announced the development of the first block turbo decoder for 10 Gb/s optical transmission [10]. Since the BTC decoder is framed with a dummy sequence, effective input throughput is 193 Mb/s while the information throughput is 156 Mb/s (taking account of the code rate). In classical parallel architecture approaches [11], BTC decoders requires a large amount of memory between each half-iteration. In this paper, we show that Interleaving Memory (IM) can be replaced by a simple connection network. In such a parallel architecture, most of the hardware complexity remains in the duplicated elementary Soft Input Soft Output (SISO) decoders. That is the reason why a complexity analysis of the SISO decoder was performed, resulting in a low complexity parallel decoder. This low complexity architecture was finally implemented onto a low cost FPGA device. This paper is organized as follows. Section 2 recalls the basic principles of decoding for product codes: their construction and the turbo decoding process. In Section 3 an innovative, memoryless, full-parallel decoder architecture is presented. In Section 4, we propose a complexity and performance analysis of the elementary decoder which allows a rapid and efficient estimation of the decoder complexity. Section 5 describes an implementation of the resulting turbo decoder within an experimental setup onto an FPGA target.

2 BTC Coding and Decoding Principles

In this section, the concept of product codes, their construction and the principle of the decoding algorithm are presented.

2.1 Linear Cyclic Block Codes

Bose Chaudhuri Hocquenghem (BCH) codes are an infinite class of linear cyclic block codes that have capabilities for multiple error detection and correction. Reed-Solomon (RS) codes are a subclass of non-binary BCH codes. Actually, a code is linear if and only if the set of codewords forms a vector subspace over a finite field like a Galois field. Moreover, in a cyclic code, for every codeword $c = (c_0, c_1, ..., c_{n-2}, c_{n-1})$, $c' = (c_{n-1}, c_0, ..., c_{n-3}, c_{n-2})$ is also a codeword. Thus all n shifts of c are also codewords. BCH and RS codes are usually specified as BCH(n, k) with binary symbols and RS(n, k) with m-ary symbols. This means that the encoder takes k data symbols and adds $(n - k)$ redundant symbols to generate an n symbols codeword. The minimum Hamming distance d is the minimum distance between two distinct codewords, over all pairs of codewords. The decoder can correct up to t symbols that contain errors in a received word, where $t = \frac{d-1}{2}$. Concerning BCH codes, it is also possible to extend the code by adding a parity check bit at the end of the codeword. Extended BCH codes are denoted as eBCH codes. This simple overhead allows an increase of the code distance with a factor 1 and the additional complexity is very low.

3 Algebraic Codes Decoding

Algebraic codes are based on Galois field arithmetic. A (n, k) code works in a Galois field GF$(n - 1)$ defined by its generator polynomial. Moreover, given a symbol size m, the maximum codeword length is $n = 2^m - 1$. Decoding algebraic codes consists in resolving the key equation:

$$S(x)\sigma(x) = \Omega(x) mod(x^{n-k}). \quad (1)$$

$S(x)$ is the syndrome polynomial, $\sigma(x)$ is the error locator polynomial and $\Omega(x)$ is the error estimator poynomial. First, syndrome values of the to be decoded vector are computed. RS code decoding requires $2t$

syndromes while BCH codes decoding requires only t syndromes. Then, the roots of the error localisator polynomial is computed, giving the position of the errors. There are many algorithms which efficiently solve this equation. The best known methods are the iterative Berkelamp Massey [12, 13] algorithm and the direct Peterson Gorenstein Zierler (PGZ) algorithm [14]. Because of its complexity, the PGZ algorithm is mostly used when $t < 3$, otherwise Berkelamp Massey is mainly employed. Finally, in the case of RS codes, the roots of the error estimator polynomial are computed. It gives the magnitude of the errors. One of the conventionally used algorithms was designed by Forney [15]. In case of BCH codes, the magnitude of the error is always 1. Correction is then obtained by inverting the corrupted bit.

3.1 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 [16]. 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.

Considering that C_1 and C_2 are linear codes, it is shown that all 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 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 $R_p = R1 \times R2$. Thus, it is possible to construct powerful product codes using linear block codes. As a general rule, the more powerful a code, the more difficult the decoding process. Figure 1 schematizes the construction of a product code P. In the following sections, we will consider a squared product code, meaning that $n_1 = n_2 = n$.

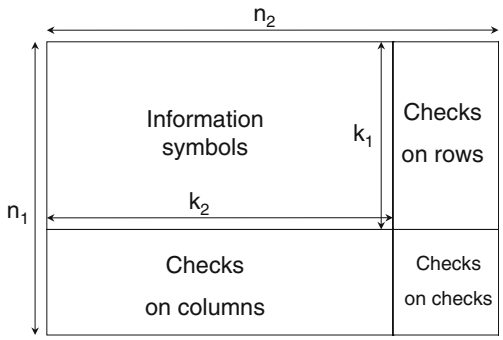

Figure 1 Structure of a product code matrix.

3.2 Iterative Decoding of Product Codes

Product code decoding involves sequentially decoding rows and columns using SISO decoders. Repeating this soft decoding during several iterations enables the decrease of the Bit Error Rate (BER). It is known as the block turbo decoding process. Each decoder has to compute soft information $[R']_{it+1}$ from the channel received information $[R]_{it}$ and the previous half-iteration computed information $[R']_{it}$ as shown on Fig. 2. Despite the existence of other decoding algorithms [17], the Chase-Pyndiah algorithm is known to give the best tradeoff between performance and decoding complexity [18]. Besides, this algorithm was adopted in 2001 as an optional correcting code system for both the uplink and downlink of the IEEE 802.16 standard (WiMAX) [19, 20].

The Chase-Pyndiah SISO algorithm [2, 21] is concisely summarized below:

1. Search for the Lr least reliable binary bits and compute the syndrome S_0 of $[R']_{it}$,
2. Generate Tp test patterns obtained by inverting some of the Lr least reliable binary symbols,
3. Algebraic decoding of each test pattern,
4. For each test vector, compute the square Euclidian distance (metric) $M_i (i=0, ..., Tp_{n-1})$ between $[R']_{it}$ and the considered test vector
5. Select the Decided Word (DW) having the minimal distance with $[R']$ and choose Cw concurrent words having the closest distance to $[R']_{it}$
6. Compute reliability $[F]_{it}$ for each symbol of the DW,

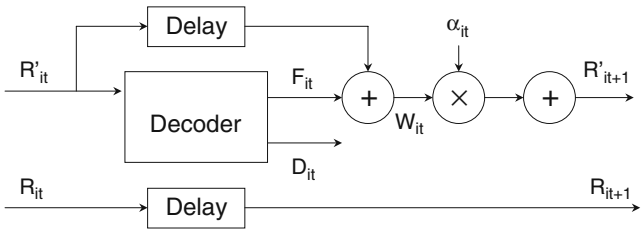

Figure 2 Elementary SISO decoder dedicated to product codes.

7. Compute extrinsic information $[W]_{it} = [F]_{it} - [R']_{it}$ for each symbol of the DW.
8. Add extrinsic information (multiplied by α_{it}) to the channel received word, $[R']_{it+1} = [R]_{it} + a_{it}[W]_{it}$

A α_{it} coefficient allows decoding decisions to be dampened during the first iterations. It should be noted that decoding parameters Lr, Tp, and Cw have a notable effect on decoding performance.

4 Parallel Decoding

4.1 Previous Work

Many block turbo decoder architectures have been previously designed. The classical high speed approach involves the use of a pipelined structure at the iteration level. Separate decoding resources are assigned for each half-iteration. In such an implementation, memory blocks are necessary between each half-iteration to store $[R']_{it}$ and $[R]_{it}$. Each memory block is then composed of four memories of $q \times n^2$ symbols where q is the number of bits to quantize the matrix symbols. This solution presents several drawbacks. First, a large amount of memory is required which increase the global latency of the system. Moreover, increasing the parallelism degree of each half-iteration produces memory conflicts when several data have to be addressed at the same time. In 2002, a new architecture was proposed [9] in order to increase the parallelism degree without any extra storage between half-iteration. The idea was to store several product code matrix symbols at the same address and to use elementary decoders able to process several symbols at the same time. This data organization does not require any particular memory architecture. The results obtained show that the turbo decoding throughput is increased by m^2 when m elementary decoders processing m data simultaneously are used and its latency is divided by m. The area of the m elementary decoders (m-e-dec) is increased by $\frac{m^2}{2}$ while the memory is constant. A m-e-dec decoder denotes an elementary decoder which decodes m symbols in one clock period. In [22], authors suggested to use a barrel shifter between decoding resources and the IM in order to avoid memory conflicts. This solution enables reaching an increased parallelism rate $P = n$ by rotating the to be stored data. The extra-complexity only consists in a barrel shifter with a complexity in $O(nlog(n))$. However, the IM requirement is still prohibitive. In the next section, we describe a innovative scheduling for block turbo decoding. In the corresponding architecture, the IM is replaced by a simple interconnection network.

4.2 Full-parallel Decoding Principle

The codewords of all rows (or all columns) of a matrix are independent. Consequently, they can be decoded in parallel. If the architecture is composed of n elementary decoders for each half-iteration, an appropriate treatment of the matrix enables the elimination of the reconstruction of the matrix between each decoding. Let i and j be the indices of a row and a column of the n^2 matrix. In full-parallel processing, the row decoder i begins the codeword decoding by the symbol in the i^{th} position. Moreover, each row decoder processes the codeword symbols by increasing the index by one modulo n. Similarly, the column decoder j begins the codeword decoding by the symbol in the j^{th} position. In addition, each column decoder processes the codeword symbols by decreasing the index by one modulo n. Therefore, each new decoded data is immediately used by the next half-iteration decoders and data only require to be properly routed. Consequently, no IM is required. The full-parallel decoding of a product code matrix is detailed in Fig. 3. The proposed decoding scheme is presented in the case of a squared product code ($n_1 = n_2 = n$). The method is also applicable to non-squared codes such as $n_1 \neq n_2$. Defining n_{min} as $n_{min} = min(n_1, n_2)$, it is possible to decode n_{min} rows and n_{min} columns in parallel.

The latency of an elementary decoder depends on the structure of the elementary decoder and the codeword length n. As the reconstruction matrix is removed, the latency between row and column decoding is null.

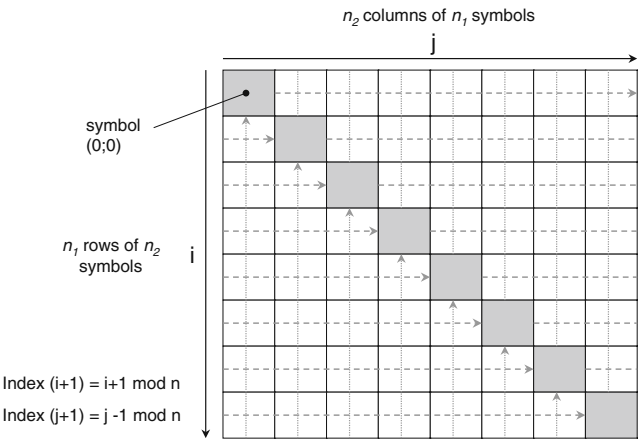

Figure 3 Full-parallel decoding of a product code matrix.

4.3 Full-parallel Architecture for Product Codes Turbo Decoder

The major advantage of our full-parallel architecture is that it enables the memory block of $4 \times q \times n$ symbols between each half-iteration to be removed. However, the codeword symbols exchanged between the row and column decoders have to be scheduled. One solution is to use a connection network for this task. In our case, we have chosen an Omega network. The Omega network is one of several connection networks used in parallel machines [23]. It is composed of $\log_2 n$ stages, each having $\frac{n}{2}$ exchange elements. In fact, a n-input Omega network contains $\frac{\log_2 n}{2}$ switch transfer blocks and $n \log_2 n$ connections. More details about the architecture can be found in Section 7.3. For example, the equivalent gate complexity of a network with 32 inputs quantized with 1 bit can be estimated to be 200 per exchange bit. The proposed full-parallel architecture for product codes is presented in Fig. 4. It is composed of cascaded modules for the block turbo decoder. Each module is dedicated to one iteration. However, it is possible to process several iterations by a same module. In our approach, $2n$ elementary decoders and 2 connection networks are necessary for one module. Actually, the full-parallel turbo decoder complexity essentially depends on the complexity of the elementary decoder and the code size n.

The proposed IM-less architecture eliminates the latency due to matrix reconstruction while maximizing the reachable throughput. It also reduces the interleaving resources from a large amount of memory to a simple omega network. The structure and the complexity of an omega network is detailed in Section 7.3. Figure 5 shows the complexity (in equivalent logic gate count) for the previous and proposed architectures. The curve A corresponds to the architecture proposed in [9], the curve B corresponds to [22] and curve C represents the interleaving logic complexity of the proposed architecture. Area estimation were plotted using a memory area estimation model provided by ST-Microelectronics. It clearly shows that the IM architecture drastically reduces the IM requirements, especially when the code size increases.

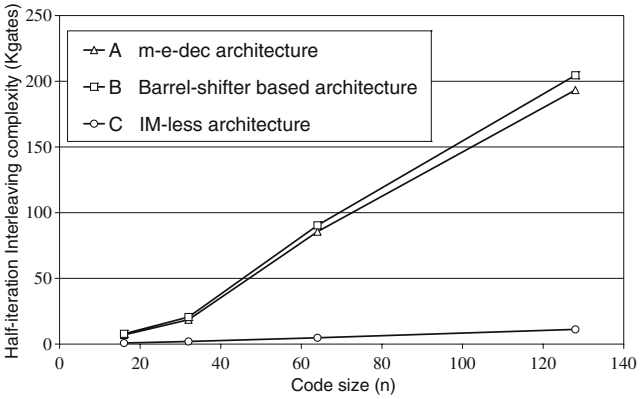

Figure 5 Comparison of internal memory complexity for a half-iteration.

Considering the proposed interleaving memoryless architecture, the remaining complexity is in the decoding processors. In such a parallel architecture, the number of required SISO decoders is $DEC_n = 2 \times it \times n$, where it and n are the number of decoding iterations and the code size respectively. A SISO decoder complexity can be reduced by a decrease of algorithmic parameters values. However, these simplifications lead to a loss in performance which have to be estimated. In the next section, we propose a method to

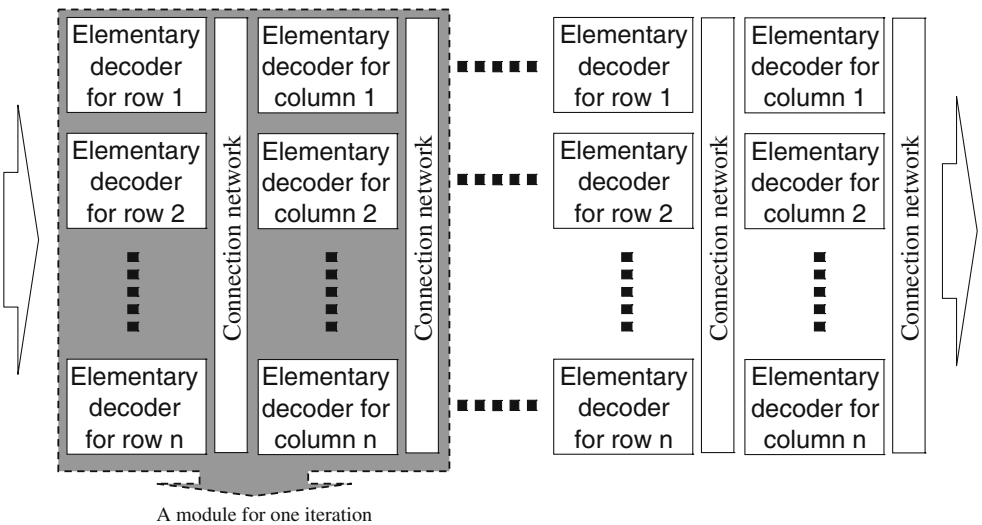

Figure 4 Full-parallel architecture of a block turbo decoder.

find a good tradeoff between complexity and decoding performance.

5 Complexity and Performance Analysis for eBCH SISO Decoders

Syntheses for the complexity estimations were performed using the Synopsys tool with an STMicrolectronics 0.09 µm CMOS process target. Elementary decoders are clocked at $f = 500$ MHz. BER performance was simulated using C-ANSI models of a turbo decoder for the product codes eBCH(16,11)^2 and eBCH(32,26)^2.

5.1 Complexity Analysis of BCH SISO Decoders

A conventional SISO decoder is composed of twelve processing parts (see Fig. 8 in Section 7.2). Running these existing decoder designs through logic synthesis showed that only four parts were critical in terms of logical gate complexity (75% of the area). As a result, our study is focused on these parts. One of these parts is the α_{it} multiplication unit. In classical architectures, it is implemented as a conversion table. Input can be multiplied by $0.55 < \alpha_{it} < 0.75$, the value depending on the current iteration. Keeping $\alpha_{it} = 0.5$ for each iteration enables the unit to be removed since the multiplication becomes a simple bit shifting. Therefore, the elementary decoder area is decreased by 8%. The induced loss of performance Δ_α is very low ($0 < \Delta_\alpha < 0.1$ dB). Consequently, the complexity analysis will now be focused on the three remaining parts. By analyzing the architecture of these critical parts, five parameters appear to directly affect their complexities. Table 1 sums up the order of complexity of the three critical parts that depend on parameters introduced in Section 3.2.

Decoding parameters Lr, Tp, Cw, as well as the code size n and the number of quantization bits q have a direct impact on the decoder complexity (area or number of logical gates). Considering a code size n, let's define a set p_i of decoding parameters:

$$p_i = q_i, Cw_i, Tp_i, Lr_i. \tag{2}$$

Table 1 Order of complexity of the three critical parts of a SISO decoder

Blocks	n	q	Lr	Tp	Cw
Dw-Cw Sorting	$O(n\log(n))$	$O(q)$	$O(Lr)$	$O(Tp)$	$O(Cw)$
Lr sorting		$O(q)$	$O(Lr)$	$O(Tp)$	
Reliability computation	$O(n\log(n))$	$O(q)$	$O(Lr)$	$O(Tp)$	$O(Cw)$

Varying p_i, directly affects both the hardware complexity and the decoding algorithm performance. Increasing these parameter values improves performance while the complexity increases. The purpose of our analysis is to be able to compute easily the complexity of a SISO decoder for any set of parameters p_i with reasonable accuracy. Considering a code size n, the most favorable configuration is:

$$p_0 = Cw_0 = 3, q_0 = 5, Tp_0 = 16, Lr_0 = 5. \tag{3}$$

It gives the best decoding performance (0.1 dB from the optimal decoding limit). Moreover, increasing parameters values of p_0 does not significantly improve performance. For this reason, this configuration is the reference for our decoder architecture complexity. Synthesis showed that, in this case, the four critical parts represent 75% of the decoder area whatever the code size n. In addition, assuming that the remaining 25% of the decoder is almost not affected by the variation of a set p_i (as verified during syntheses), it can be demonstrated that the SISO decoder complexity $C_{pi}(n)$ can be expressed as:

$$C_{pi}(n) = (1/3)(C'_{p0}(n)) + C'_{pi}(n). \tag{4}$$

where $C'_{pi}(n)$ is the cumulated complexity of the four critical parts in terms of logical gates for a parameter set p_i. More details can be found in the Appendix. Synthesizing generic descriptions of the critical parts and carrying out a multiple linear regression analysis led to an expression of $C'_{pi}(n)$ only depending on the decoding parameters. This expression represents the cumulated complexity $C'_{pi}(n)$ of the four critical parts in terms of logical gates for a parameter set p_i.

$$\begin{aligned} C'_{pi}(n) = {} & 462(Cw_i - 1) + 261(\log(n) - 4) + 183(q_i - 4) \\ & + 55(Tp_i - 4) + 46(Lr_i - 2) + 1019 \end{aligned} \tag{5}$$

The model's accuracy was measured a posteriori. The maximum and average errors (between model and synthesis results) are 8% and 2.5% respectively. The total area of a half-iteration is given by:

$$C_T = nC_{p_i}(n) + 2C_\Omega(n, q) + C_{RAM}(n, q), \tag{6}$$

where $C_\Omega(n, q)$ is the area of a omega network routing n data of q bits and $C_{RAM}(n, q)$ is the area of the internal memory required in the SISO decoder as explained in Section 7.2.

5.2 A Complexity–Performance Tradeoff Case of Study

The previously presented model can be used in order find a good tradeoff between performance and

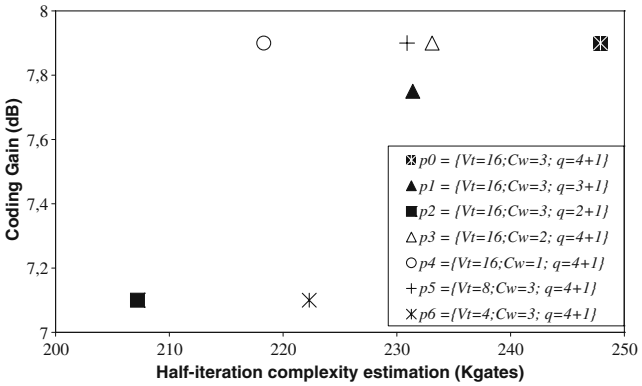

Figure 6 A complexity performance tradeoff for the eBCH(32,26)^2 code.

complexity. A case of study was performed for a code eBCH(32,26)^2. The coding gain at BER $= 10^{-6}$ and the complexity of a half-iteration were computed for different parameter sets. Figure 6 shows the impact of decoding parameters on the performance and the complexity.

First, it can be observed that the complexity variation can be significant: $C_T(p_6) - C_T(p_0) = 25$Kgates per half-iteration while the corresponding performance loss is 0.8dB. Comparing p_0, p_3 and p_4 show that, at BER=10^{-6}, Cw does not impact the decoding performance while the complexity linearly decreases with Cw. The effect of the quantization q is highlighted by parameter sets p_0, p_1 and p_2. The quantization level q has a notable impact on decoding performance and have to be chosen considering the potential performance loss in performance. The same remark can be done for Tp when considering parameter sets p_0, p_5 and p_6. This case of study does not have any pretention to generalize the impact of each parameter on decoding performance but it does show some trends of the complexity for the different parameter sets. When coupled with algorithmic performance considerations, such a complexity model can help the designer to efficiently explore the design space of the block turbo decoder.

6 Towards Prototyping

Currently, a typical hardware design approach is to use an FPGA development board to first prototype the turbo decoder design and its experimental setup. The low cost Virtex II-Pro XUP [24] development system from Digilent was selected for our experimentation. These boards contain a Xilinx Virtex II-Pro XC2VP30 FPGA device with 13,696 slices. Preliminary syntheses show that only a half-iteration of the eBCH(32,26)^2 block turbo decoder would fit onto the Virtex II-Pro XC2VP30 device. Indeed, some elements of the experimental setup have to fit onto the same FPGA device as the block turbo decoder. In the case of eBCH(16,11)^2, up to 3 half-iterations (48 SISO decoders) can be implemented onto the same low-cost Xilinx VII Pro FPGA device. For this reason, the eBCH(16,11)^2 was finally chosen for our experimentation. The main concern of the implementation phase was the limited resources available on the targeted component. The complexity model was transposed for a FPGA target and it provided an estimation of the complexity (slices number instead of gates count) for different parameter sets. Algorithmic simulations showed that for a small size of product code (eBCH(16,11)^2), the performance degradation associated with the particular set $p_7 = Cw = 1, q = 4, Tp = 4, Lr = 2$ was reasonnable. Replacing parameter set p_0 by p_7 degrades performance, but the gap is only 0.4 dB at BER=10^{-2} while it becomes insignificant (<0.1 dB) at BER=10^{-6}. Compared with p_0, the complexity is reduced by 30% and allows one full-iteration to fit onto the Virtex II Pro.

7 Full-parallel Encoder and Turbo Decoder Implementations on an FPGA Device

7.1 Full-parallel Encoder Architecture

Linear block codes encoding consists in dividing information vector k by the polynomial generator of the considered code. This is conventionally implemented by a set of shift registers with some feedback logic. In a conventional product codes encoding process, k_1 rows are firstly encoded using the code C_2. Therefore, a $k_1 \times (n_2 - k_2)$ redundancy block is obtained. Then the n_2 columns composed of $k_1 \times k_2$ information bits and $k_1 \times (n_2 - k_2)$ redundancy bits are encoded. This sequential approach requires a $k_1 \times n_2$ memory block between rows and columns encoding. Alternatively, a linear block code can be encoded by multiplying the information vector k by the generator matrix. This can be easily implemented by an XOR arborescence which is a parallel combinatorial solution. The original proposed encoding scheme uses both sequential and parallel encoders. k sequential encoders process k_1 rows and a parallel encoder processes the n_2 columns. Figure 7 shows the three first steps of the encoding process for a (7,4) code. It also depicts the corresponding architecture. During the first clock period, k_1 sequential encoders process the first bit of the k_1 rows. Next, the sequential encoder processes the second bit of each of the k_1 rows while the parallel encoder processes simultaneously the

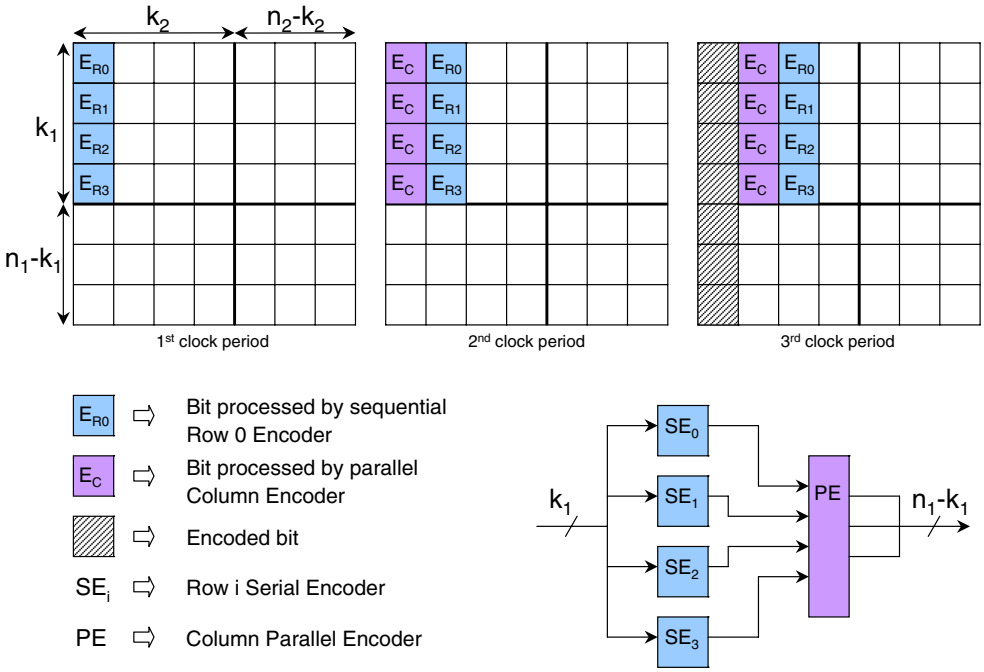

Figure 7 Product code parallel encoding scheme and architecture.

k_1 first bits of the first column. During the next period, redundancy of the first column is generated by the parallel encoder.

In the previous approaches, it was necessary to encode all k_1 rows before starting to encode the n_2 columns. A $k_1 \times n_2$ size memory was then required. The depicted architecture decreases the latency from $k_1 \times n_2$ to only 1 symbol and avoids the use of an intermediate $k1 \times n2$ memory block.

7.2 Full-parallel Decoder Architecture

Full-parallel decoder architecture has been detailed in Section 4.3. It was observed that most of the complexity remains in the duplicated SISO decoders. The complexity analysis performed in Section 5 allowed us to select a configuration to be implemented on the SISO decoder. The elementary SISO decoder architecture will now be detailed as depicted on Fig. 8.

All the soft information within the decoder is quantized and processed with $q = 4$ bits (1 sign bit and 3 reliability bits). The SISO decoder architecture is structured in three pipelined stages identified as reception, processing and emission units. Each stage processes $n = 16$ symbols in $n = 16$ clock periods. According to the latency definition given in Section 4.2, the resulting latency is then equal to $2n = 32$ clock symbols. The reception unit computes the syndrome S_0 and the $Lr = 2$ least reliable bits of the word received $[R']_{it}$. This unit is also in charge of the decoder scheduling by generating

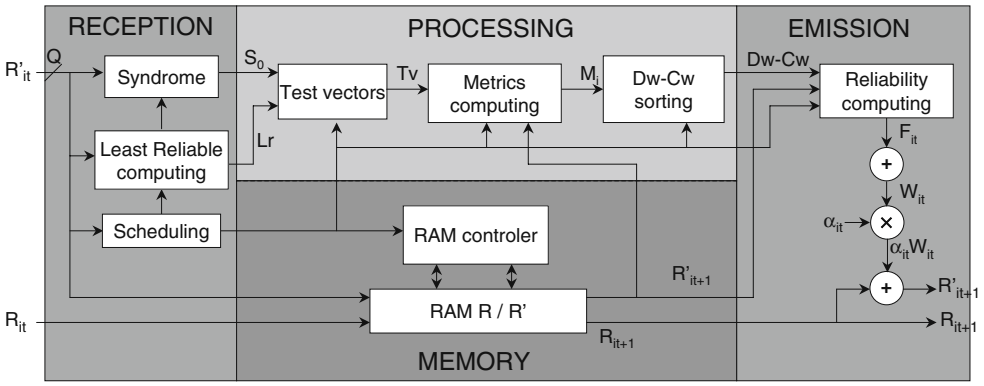

Figure 8 Architecture of a SISO decoder.

some control signals. The processing unit computes the syndrome of the $Tp = 4$ test patterns and their metric values. It also sequentially selects the decided word and the concurrent ones. The emission unit calculates new reliabilities from metrics of the decided word and the $Cw = 1$ concurrent word. Extrinsic information $[W]_k$ and soft output $[R']_{it+1}$ are also processed during the same clock period. A new q-bits symbol is then transmitted at each clock period. The decoding process needs to access the $[R]_{it}$ and $[R']_{it}$ values during the three decoding phases. For this reason, these words are implemented into Random Access Memories (RAM) of size $q \times n$ controlled by a finite state machine.

7.3 Omega Network Implementation

The multi-stage dynamically reconfigurable interconnection network omega is based on a circular permutation principle. The connecting scheme consists of circularly shifted input data. Figure 9 shows an example of communication between two sets of 8 SISO decoders. Switches are simple logic elements that have two possible controllable routing configurations. Such switches blocks are easily implementable in CMOS technology. Indeed, a 2-input switch presents an equivalent complexity of 2.5 logical gates.

The complexity of an omega network with n inputs and q-bits quantized data is then

$$C_\Omega(n, q) = 2.5 \times q \times \frac{n}{2} \times \log n \tag{7}$$

n control sequences of the omega network are generated by a configuration ROM in n cycles. The ROM is addressed by a modulo n counter. The implemented turbo decoder contains $2n = 32$ SISO decoders per iteration and two connection omega networks. Between two half-iterations, two omega network are required to transmit $[R']_{it+1}$ and $[R]_{it}$. The complexity of the implemented 16×16 omega network for 1 iteration is then: $2 \times C_\Omega(16, 8) = 320$ logical gates.

8 Implementation of a Block Turbo Decoder in an Ultra High Rate Communication Setup

The purpose of this first implementation is to show that a block turbo decoder can effectively work without memories between half-iterations at high throughput.

8.1 Experimental Setup

The experimental setup is composed of two identical development boards XUP linked with a Serial ATA communication bus. BER measurement facilities are implemented in order to rapidly verify the decoding performance. Each board contains a Xilinx XC2VP30 device that can transmit data up to 2.4 Gb/s rate. Indeed, encoded noisy data are sent from the transmitter FPGA to the receiver FPGA using the high speed Xilinx Aurora protocol. Each board has its own digital clock management system operating at 50 MHz. Synchronization between the two boards is carried out by Aurora protocol control signals. The Aurora protocol is clocked at $f_1 = 75$ MHz and the rest of the

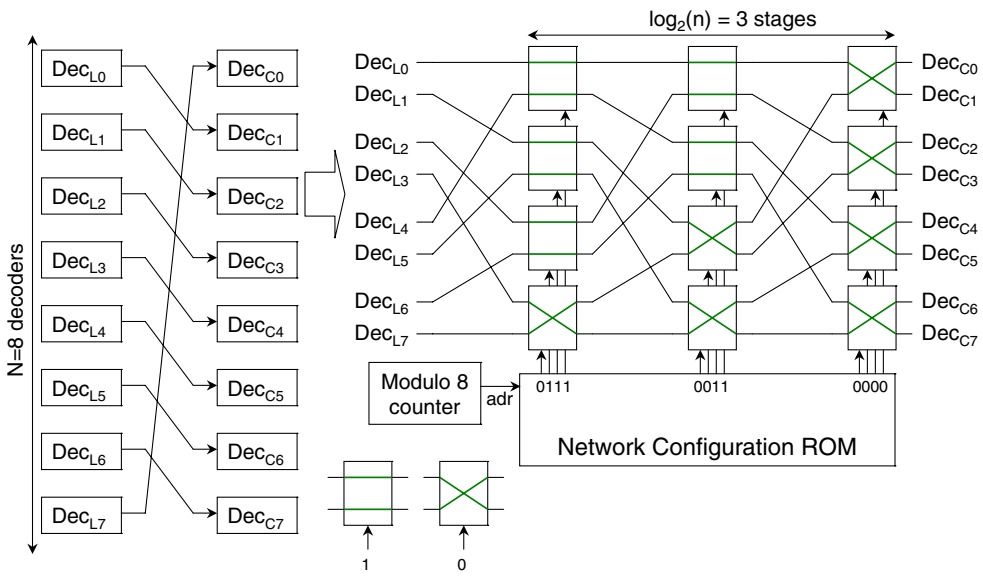

Figure 9 Architecture of a 8-input Ω network.

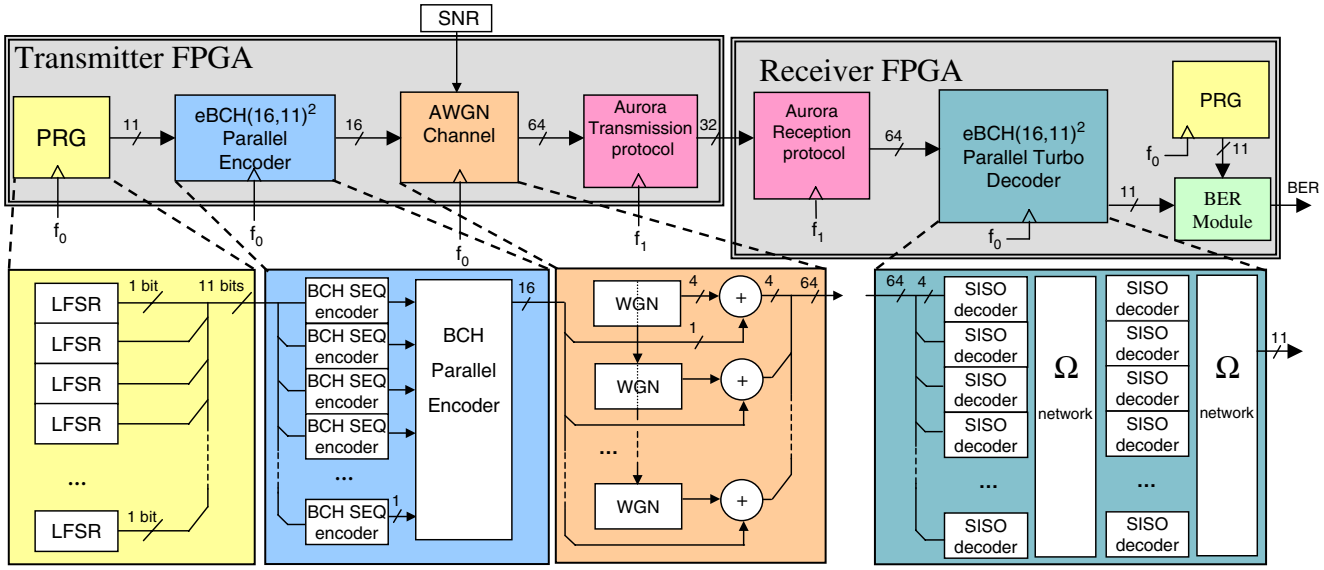

Figure 10 Mult-gigabit experimental setup.

setup is clocked at $f_0 = 37.5$ MHz. Figure 10 shows the different components of the communication setup implemented onto the FPGA.

8.2 The Components Implemented on the Transmitter FPGA Device

A Pseudo Random Generator (PRG) sends out eleven pseudo random data streams at each clock period (f_0). It is composed of eleven different LFSR. Each LFSR has a different primitive polynomial and a different size which insure a non correlation between the eleven outputs. An eBCH(16,11)^2 encoder processes the eleven data streams in parallel. This innovative architecture avoids the use of memory between rows and columns encoding as mentionned in Section 7.1. Eleven conventional sequential encoders are cascaded with one parallel encoder. 256 encoded data (equivalent to a matrix 16×16) are generated in 16 clock periods (f_0). The noise generator models 16 uncorrelated White Gaussian Noise (WGN) samples and adds them to the previously encoded data [25]. Each output sample is a 4 bit vector resulting in 64 bits to be sent in 1 clock period (f_0). The Signal to Noise Ratio (SNR) is controllable via on-board switches $0<\text{SNR}<15.75$ dB with a pitch of 0.25 dB. The Aurora protocol emission module handles a set of control signals. It receives 64 data in 2 clock cycles and sends 32 data every clock cycle (f_1). The output rate is then 2.4 Gb/s.

8.3 The Components Implemented on the Receiver FPGA Device

The Aurora protocol reception module receives data at 2.4 Gb/s and sends out 64 bits every two clock cycles (f_1). The turbo decoder is composed of 32 SISO (16 SISO decoders per half-iteration) and two omega networks used to route data between half-iterations. The decoder architecture and the omega network are described in Sections 4.3 and 7.3. Data arrive at 2.4 Gb/s while the working frequency is only 37.5 MHz. The same PRG is also implemented in the receiver. It generates the exact same data as in the transmitter in order to compare data before and after decoding. A BER block is finally used to measure the error rate comparing data from the PRG and the decoder output. It guarantees a minimum of 1,000 errors before outputting the BER value. This value is then displayed on an LCD module. The minimum reliable BER value is 10^{-9}. Turbo decoder BER was measured after two half-iterations. Comparing measured BER values with C reference simulations, we observed a slight difference. At BER = 10^{-4}, we have a gap of 0.2 dB. This difference tends to slightly increase with SNR. However, bit-true simulations showed that the turbo decoder strictly performs like the C reference program. Indeed, when providing the same soft inputs to both decoders, soft outputs are exactly the same. Consequently the divergence is most probably due to the Gaussian channel implementation. The same trend was observed using this channel implementation with a simple convolutional

SISO decoder which again excludes the decoding fault possibility.

8.4 Characteristic and Performance

Clocked at only 37.5 MHz, the turbo decoder processes input data at 600 Mb/s. This frequency is limited by the communication protocol. The turbo decoder can actually perform up to 50 MHz on this target, which corresponds to 800 Mb/s. Using an FPGA device optimized for high-performance logic would lead to even higher frequency. The output throughput, it is defined as:

$$T_{out} = P \times f \times R. \tag{8}$$

P is the parallelism rate ($\max(P) = n$), f the elementary decoder frequency and R the code rate. In our case, $P = 16$, $f = f_0 = 37.5$ MHz and $R = 0.473$, the resulting throughput is then 284 Mb/s. Several solutions exist to increase the throughput, the most straightforward is to use a larger code (in order to increase P) with a larger rate R. For instance, assuming we are using an eBCH(32,26)[2] at $f_0 = 37.5$ MHz, the input and output throughput become 1.2 Gb/s and 792 Mb/s respectively. In our architecture, SISO decoders process data sequentially. Designing SISO decoders that decode several data in one clock period, as in [9], would again improve throughput and with a limited complexity overhead. Moreover, enhancing our study to non binary component codes like RS codes [26] can increase throughput even more. The turbo decoder was synthesized and implemented on a Virtex II Pro FPGA using Xilinx ISE 7.1i tools. The decoder occupied 7300 slices. So far, one iteration (32 SISO decoders and 2 omega networks) has been fully implemented. The available target (xc2vp30) was insufficient to implement several iterations. Duplicating the decoders simply requires a larger FPGA target. Implementing a 6-iteration full-parallel turbo decoder represents 43800 FPGA slices with a maximum throughput. Such a design can for instance, fit onto a Xilinx Virtex 4. Another solution would be to use as much experimental board as required iterations.

9 Conclusion

This article presents an innovative architecture for both encoding and turbo decoding of product codes. After introducing the full-parallel decoding principle, we also show how we implemented this memory free, high-throughput, full-parallel, block turbo decoder on an FPGA device. In such parallel architectures, it is necessary to use low complexity SISO decoders. Thus, we proposed a complexity analysis for the eBCH(n, k) SISO decoder. The complexity expression gives a rapid estimation of the SISO area, for a fixed set of decoding parameters. Then, depending on the required level of performances, it becomes easy to decide on a set of parameters to implement. This analysis led to a low complexity SISO decoder (−30%) to be duplicated in the parallel turbo decoder. Next, we describe the experimental setup designed to test the turbo decoder also including a full-parallel encoder. Using a more efficient communication protocol the turbo decoder can process input data at 3.2 Gb/s. Using a larger code, with a higher rate and parallel SISO decoders, would again, increase the data rate. Moreover non binary codes like RS codes enable even higher throughputs to be reached. Currently, a study for proposing a BTC implementation on an FPGA device for the physical layer of a 10 Gb/s optical access network is in progress.

Acknowledgements The authors would like to thank Gerald Le Mestre for his significant help during the experimental setup design phase.

Appendix: SISO Complexity Approximation Proof

$C'_{pi}(n)$ is the cumulated complexity of the four critical parts in terms of logical gates for a parameter set p_i and a code size $n.C_{pi}(n)$ is the SISO decoder complexity in terms of logical gates. Let's first assume that,

$$C_{p0}(n) - C_{p_i}(n) \approx C'_{p0}(n) - C'_{p_i}(n) \tag{9}$$

(verified during syntheses) Since the 4 critical blocks represent 75% of the complexity, we have

$$C_{p0}(n) = ((C'_{p0}(n))/(0.75)) \tag{10}$$

Combining (9) and (10), we obtain,

$$C_{p_i}(n) = (1/3)(C'_{p0}(n)) + C'_{pi}(n) \tag{11}$$

References

1. Berrou, C., Glavieux, A., & Thitimajshima, P. (1993). Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Communications, 1993. ICC 93. Geneva. Technical program, conference record, IEEE international conference on* (Vol. 2, pp. 1064–1070), 23–26 May.

2. Pyndiah, R., Glavieux, A., Picart, A., & Jacq, S. (1994). Near optimum decoding of product codes. In *Global telecommunications conference, 1994. GLOBECOM '94. 'Communications: The global bridge', IEEE* (Vol. 1, pp. 339–343), 28 Nov.–2 Dec.
3. Gallager, R. G. (1962). Low density parity check codes. *IRE Transactions on Information Theory, IT, 8*, 21–28, January.
4. Azadet, K., Haratsch, E., Kim, H., Saibi, F., Saunders, J., Shaffer, M., et al. (2002). Equalization and fec techniques for optical transceivers. *IEEE Journal of Solid-State Circuits, 37*(3), 317–327, March.
5. Ait Sab, O., & Lemaire, V. (2000). Block turbo code performances for long-haul dwdm optical transmission systems. In *Optical fiber communication conference, 2000* (Vol. 3, pp. 280–282), 7–10 March.
6. Mizuochi, T. (2006). Recent progress in forward error correction and its interplay with transmission impairments. *IEEE Journal of Selected Topics in Quantum Electronics, 12*(4), 544–554, July–Aug.
7. Song, L., Yu, M.-L., & Shaffer, M. (2002). 10- and 40-gb/s forward error correction devices for optical communications. *IEEE Journal of Solid-State Circuits, 37*(11), 1565–1573, Nov.
8. Lee, H. (2005). A high-speed low-complexity reed-solomon decoder for optical communications. *IEEE Transactions on Circuits and Systems II: Express Briefs [see also IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing], 52*(8), 461–465, Aug.
9. 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* (Vol. 2, pp. 1363–1367), 17–21 Nov.
10. Tagami, H., Kobayashi, T., Miyata, Y., Ouchi, K., Sawada, K., Kubo, K., et al. (2005). A 3-bit soft-decision ic for powerful forward error correction in 10-gb/s optical communication systems. *IEEE Journal of Solid-State Circuits, 40*(8), 1695–1705, Aug.
11. Kerouedan, S., Adde, P. (2000). Implementation of a block turbo decoder on a single chip. In *2nd international symposium on turbo codes & related topics* (pp. 243–246).
12. Berlekamp, E. R. (1984). *Algebraic coding theorie, revised edn*. Laguna Hills: Aegean.
13. Massey, J. L. (1969). Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory, IT-15*, 122–127, January.
14. Peterson, W. (1960). Encoding and error correcting procedures for the bose chaudhuri codes. *IEEE Transactions on Information Theory, IT-6*, 459–470, September.
15. Forney, J. G. (1965). On decoding bch codes. *IEEE Transactions on Information Theory, 11*(4), 549–557, Oct.
16. Elias, P. (1954). Error-free coding. *IEEE Transactions on Information Theory, 4*(4), 29–37, Sept.
17. Forney, J. G. (1966). Generalized minimum distance decoding. *IEEE Transactions on Information Theory, IT-12*, 125–131, April.
18. Pyndiah, R. (1998). Near-optimum decoding of product codes: Block turbo codes. *IEEE Transactions on Communications, 46*(8), 1003–1010, Aug.
19. IEEE (2001). I. S. 802.16-2001 IEEE standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems, December.
20. IEEE (2002). I. S. 802.16a 2003 IEEE standard for local and metropolitan area networks—part 16: Air interface for fixed broadband wireless access systems—amendment 2: Medium access control modifications and additional physical layer specifications for 2–11 ghz. April.
21. Chase, D. (1972). A class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory, IT*, 170–182, January.
22. 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* (Vol. 1, pp. I–901–I–904), 26–29 May.
23. Lawrie, D. (1975). Access and alignment of data in an array processor. *IEEE Transactions on Computers, C-24*(12), 1145–1155, Dec.
24. Xilinx (2005). *Xilinx university program Virtex-II pro development system*. Hardware reference manual. http://www.xilinx.com/univ/XUPV2P/Documentation/XUPV2P_User_Guide.pdf.
25. Danger, J. L., Ghazel, J. L., Boutillon, E., & Laamari, H. (2000). Efficient fpga implementation of gaussian noise generator for communication channel emulation. In *ICECS*, December.
26. Piriou, E., Jégo, C., Adde, P., Le Bidan, R., & Jézéquel, M. (2006). Efficient architecture for reed solomon block turbo code. In *Circuits and systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE international symposium on* (p. 4), 21–24 May.

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, Montreal, Canada. His research interest focus on VLSI and FPGA implementation of iterative decoding algorithms.

Christophe Jégo 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). His research activities are concerned with analysis and design of architectures for iterative processing in the digital communication systems.

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.

Michel Jézéquel (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 École Nationale Supérieure des Télécommunications de 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.