# Turbo decoding of product codes for Gigabit per second applications and beyond

Javier Cuevas Ordaz, Patrick Adde, Sylvie Kerouedan

# Turbo decoding of product codes for Gigabit per second applications and beyond

Javier Cuevas, Patrick Adde and Sylvie Kerouedan*

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

The main purpose of this paper is to present a new turbo decoding architecture for high data rates with strong error correction power. We present the latest results on the block turbo decoder designs of product codes, using extended BCH codes correcting one and three errors. We present the architecture for decoding the product code BCH (32,26,4) $\otimes$ BCH (32,26,4) as well as the architecture for decoding the product code BCH (128,106,8) $\otimes$ BCH (128,106,8). In both cases, we use the concept of a parallel decoding architecture which stores several data at the same memory address (RAM). The decoder designs are based on soft input-soft output (SISO) decoding, processing $m = 2$, 4 and 8 data simultaneously in order to increase the data rate. By using $m$ elementary decoders processing $m$ data at the same time, this new architecture increases the data rate by a factor $m^2$, the area of the $m$ elementary decoders being approximately increased by a factor $m^2/2$. But this new architecture uses only one RAM for decoding while a traditional design uses $m$ memories. In this paper, we compare the most important characteristics of the new design with those of a traditional architecture. To compare the performance and complexity of the decoders, we use C language for behavioural simulations, VHDL for functional simulations and Synopsys Design Compiler for the synthesis.

## 1. INTRODUCTION

In 1993, C. Berrou [1] proposed the concept of turbo codes. The coding scheme developed is based on the parallel concatenation of two recursive systematic convolutional codes separated by a non-uniform interleaver. The iterative decoding algorithm, introduced by Berrou, uses soft input-soft output (SISO) decoding of the component codes with performance close to the Shannon limit. This coding scheme is known as convolutional turbo coding.

Block turbo codes (BTC) are an alternative solution to convolutional turbo codes (CTCs). They use a product code architecture with linear block code for elementary code. To decode product codes, many iterative decoding algorithms have been proposed in the last few years [2, 3] which use algebraic decoders or hard decoders to decode the received samples. In 1994, R. Pyndiah [4–6] presented a new iterative decoding algorithm for decoding product codes, based on the iterative SISO decoding of concatenated block codes. This coding scheme uses the series concatenation of block codes, which was introduced in 1954 by P. Elias [7]. The minimum Hamming distance of the product code can be very high (9, 16, 24, 36 and higher) even for relatively small data blocks. BTC are especially attractive for applications requiring very high code rates (up to $R = 0.95$) and are used for data transmission or data storage in order to ensure a high quality of service. The coding gains were close to the theoretical coding gain expected from product codes when using maximum likelihood sequence decoding. From the hardware point of view there are many different solutions to implement the BTC depending on trade-off between performance, complexity, decoding delay and data rate. There is a large amount of flexibility for implementing the SISO decoder.

* Correspondence to: Sylvie Kerouedan, GET-ENST Bretagne, Technopôle Brest Iroise, CS 83818, 29238 Brest Cedex 3, France.
E-mail: sylvie.kerouedan@enst-bretagne.fr

Given the structure of the product code, a single elementary decoder can be used to perform all the iterations instead of the pipeline solution which duplicates elementary decoders and so increases complexity and decoding delay.

This paper presents a new architecture for turbo decoding which stores $m$ data at the same address and performs parallel decoding to increase the data rate. Thus, it is possible to process $m$ data simultaneously with only one memory (classical designs require $m$ memories). We detail the processing unit (PU) design with $m = 2, 4$ or 8 samples on the input (2-PU, 4-PU and 8-PU). For each decoder, the latency is decreased, the area is increased and the critical path is similar. The data rate increases by $m^2$ if we have $m$ parallel decoders.

In Section 2, we present the fundamental aspects of the theory related to BTCs: their construction and principles [8] of turbo decoding. Then we recall the iterative process of turbo decoding, the decoding algorithm and its performance. In Section 3, we introduce the elementary decoder architecture for decoding the product code BCH (32,26,4) $\otimes$ BCH (32,26,4) [9]. Then we present the elementary decoder architecture for decoding the product code BCH (128,106,8) $\otimes$ BCH (128,106,8). The BCH code used is able to correct three errors ($t = 3$, strong error correction power). Section 4 is dedicated to traditional architectures for the turbo decoding of product codes at high rates, as well as the principles of the architectures in the turbo decoding of product codes at high rates. Then, in Section 5 we present the new architecture for turbo decoding at high data rates, using the concept of parallel architecture which stores several data at the same memory address. To show the application of this new architecture we have decided to study the block code BCH (32,26,4) correcting one error ($t = 1$) [10] and the block code BCH (128,106,8) correcting three errors ($t = 3$). In both cases the architectures are able to process $m = 2$, 4 and 8 data symbols at the same time. Finally, in the conclusion, we discuss and compare the synthesis and simulation results of each block code studied in this paper.

## 2. PRODUCT CODES

In this section, we recall some concepts of BTC and we present the turbo decoding algorithm of product codes and their performance.

### *2.1. Principle concepts of product codes*

In 1954, P. Elias [7] proposed product codes which are a series concatenated coding scheme. For the construction

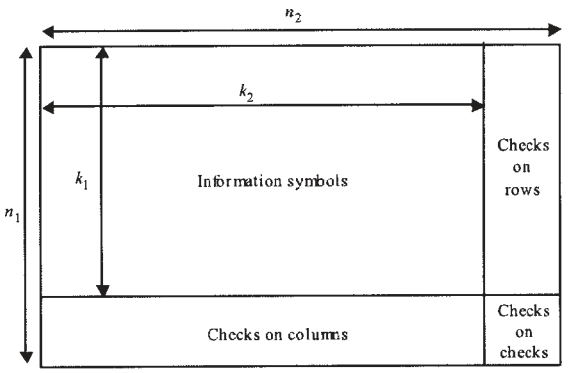


Figure 1. Construction of product code $P = C^1 \otimes C^2$.

of product codes, let us consider two systematic linear block codes $C^1$ and $C^2$ with parameters $(n_1, k_1, \delta_1)$ and $(n_2, k_2, \delta_2)$ where $n_i$, $k_i$ and $\delta_i$ $(i = 1,2)$ stand for codeword length, number of information bits, and minimum Hamming distance respectively. The product code $P = C^1 \otimes C^2$ is presented in the form of a matrix $\mathbf{C}$ with $n_1$ rows and $n_2$ columns as illustrated in Figure 1. The parameters of the resulting product code are given by: $n = n_1 \times n_2$, $k = k_1 \times k_2$, $\delta = \delta_1 \times \delta_2$ and code rate $R$ is given by $R = R_1 \times R_2$, where $R_i$ is the code rate of code $C^i$.

If the codes $C^1$ and $C^2$ are linear and systematic, then the $(n_1 - k_1)$ rows of the matrix are codewords of $C^2$ and the $(n_2 - k_2)$ columns of the matrix are codewords of $C^1$. The product code is characterised by all of the columns of matrix $P$ being codewords of $C^2$ and all of the rows of matrix $P$ being codewords of $C^1$. The codes $C^1$ and $C^2$ can be obtained from elementary convolutional codes or linear block codes (extended BCH codes).

### *2.2. Iterative decoding algorithm*

The principle of the turbo decoding of product codes is illustrated in Figure 2 [11]. It involves carrying out the decoding of rows (half-iteration) of the matrix [$\mathbf{C}$] and after reconstructing the matrix, the decoding of columns is performed (half-iteration). This process is achieved by a structure which consists of cascaded soft input/soft output (SISO) decoders for rows and columns.

### *2.3. Block diagram of a half-iteration*

The SISO decoder, which is used for decoding one half-iteration (rows or columns) uses an algorithm based on the Chase and Pyndiah algorithms [4, 12]. Figure 3 illustrates the elementary decoder that performs a half-iteration, the current half-iteration being indicated by $k$.

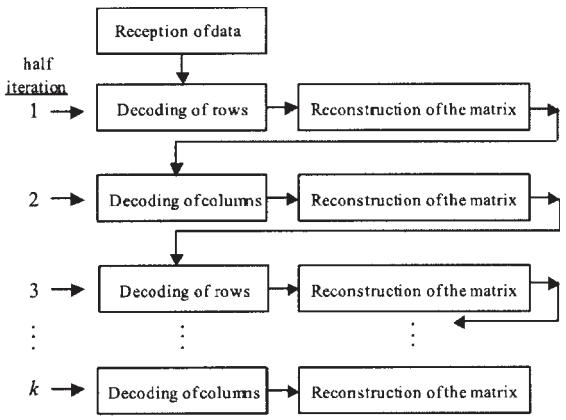


Figure 2. Iterative process of turbo decoding.

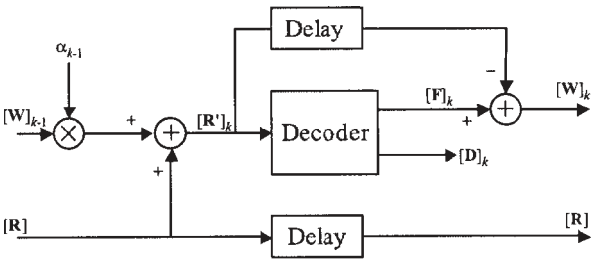


Figure 3. Block diagram of a half-iteration elementary decoder.

where:

(1) $[\boldsymbol{R}]$ is the information vector received from the channel.
(2) $[\boldsymbol{W}]_k$ is the extrinsic information vector (difference between the output information and the input information). This extrinsic information is given by the previous half-iteration concerning the reliability of the decoded bit.
(3) $[\boldsymbol{D}]_k$ is the result of binary decoding.
(4) $[\boldsymbol{R}']_k = [\boldsymbol{R}] + \alpha_{k-1} \times [\boldsymbol{W}]_{k-1}$ is the information from the previous half-iteration.

$\alpha_{k-1}$ is a constant varying with the current half-iteration (optimised by simulations).

### 2.4. The SISO algorithm

The SISO algorithm performed by the elementary decoder corresponds to the SISO decoding of a row (or column) of the matrix. The different steps of the SISO algorithm are given below [13, 14]:

(1) Determine the $p$ least reliable binary symbols of $[\boldsymbol{R}']_k$; their positions are called $\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_p$.
(2) Form $\tau$ test patterns $\boldsymbol{T}^{\mathrm{Q}}$ $\mathrm{Q} \in \{1, \ldots, \tau\}$ which are a combination of elementary test vectors $\boldsymbol{T}^j$ with '1' in position $\boldsymbol{I}_j$ and '0' elsewhere.
(3) Compute $\boldsymbol{Z}^{\mathrm{Q}} = \boldsymbol{T}^{\mathrm{Q}} \oplus (\text{sign of } \boldsymbol{R}'_k)$, for each test vector $\boldsymbol{T}^{\mathrm{Q}}$.
(4) Perform the binary decoding for each test vector.
(5) Select codeword $\boldsymbol{Y}^{\mathrm{D}}$ having the minimal Euclidean distance with $\boldsymbol{R}'_k$. Then $\mathbf{D} = \mathbf{Y}^{\mathrm{d}}$ is the result of the binary decoding.
(6) Choose the closest test vectors of $\mathbf{D}$, called competitors.
(7) Compute the reliability for each element $\mathrm{d}_j$ of $\mathbf{D}$.
(8) Compute the extrinsic information.

The binary decoding (step 4) determines the positions in the received word that must be corrected, and applies the correction. When the SISO algorithm corrects three errors ($t = 3$), the binary decoding consists of three steps, described below:

(1) Compute the syndromes $\mathbf{S}_1$, $\mathbf{S}_3$, $\mathbf{S}_5$.
(2) Compute the coefficients of the error-locator polynomial $\boldsymbol{\sigma}_1$, $\boldsymbol{\sigma}_2$, $\boldsymbol{\sigma}_3$ by Peterson's method [15]. We use standard techniques to solve sets of simultaneous linear equations for the coefficients of the error locator polynomial. The results of such solutions for $t = 3$ are as follows:

$$\boldsymbol{\sigma}_1 = \mathrm{S}_1$$
$$\boldsymbol{\sigma}_2 = (\mathrm{S}_1^2\ \mathrm{S}_3 + \mathrm{S}_5)/(\mathrm{S}_1^3 + \mathrm{S}_3) \text{ and}$$
$$\boldsymbol{\sigma}_3 = \left(\mathrm{S}_1^3 + \mathrm{S}_3\right) + \mathrm{S}_1\sigma_2$$

(3) Using Chien's algorithm [16], determine the roots of the error-locator polynomial $\boldsymbol{\sigma}(\mathrm{X})$, which are the error locators, and correct the errors indicated.

### 2.5. Performance of the algorithm

In order to study the performance of the algorithm it is necessary to consider some variables, the most important being: the number of bits to quantify data, the number of test vectors and the number of concurrent words [17, 18]. Figure 4 shows the performance of the turbo decoding algorithm when we use a QPSK modulation on a Gaussian channel.

The product code used is built from codes of size $N = 128$, correcting 1 error (BCH 128,120), 2 errors

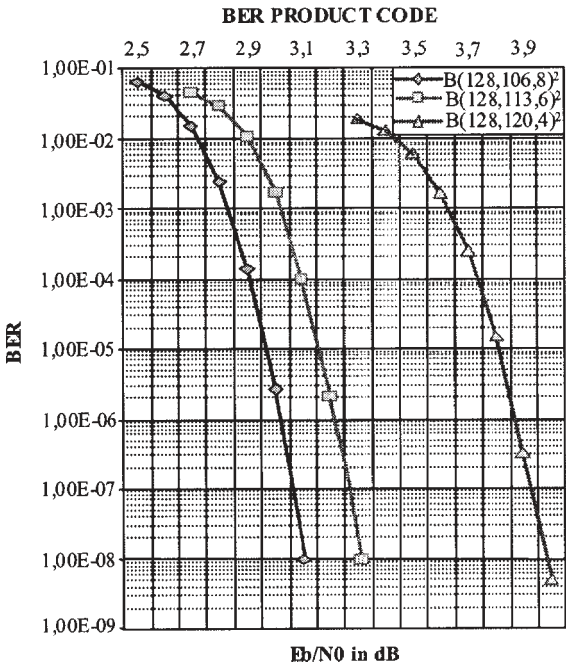


Figure 4. BER as a function of SNR using QPSK modulation on a Gaussian channel.

(BCH 128,113) and 3 errors (BCH 128,106). The processed data are coded using five bits, one bit for the sign and four bits for its reliability.

From the curves we can deduce that the error correction power is good. The slope of the curve is about 20 decades/dB for the turbo code built from the extended BCH (128,106) code. The code rate $R$ is 0.686. We note that the distance between the code performance and the theoretical performance limit is about 1.25 dB at BER $= 10^{-8}$.

## 3. ELEMENTARY DECODER ARCHITECTURE

In this section, we present the elementary decoder architecture used in the turbo decoder of product codes. We choose to study first the block code BCH (32,26,4) which corrects one error ($t = 1$) and then the block code BCH (128,106,8) which corrects three errors ($t = 3$).

### *3.1. The block code BCH (32,26,4), correcting one error ($t = 1$)*

The architecture design for decoding the product code, obtained from BCH codes, correcting one and three errors

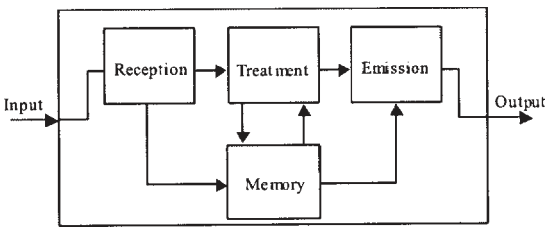


Figure 5. Functional blocks of the elementary decoder.

is based on the circuit of Figure 3. Let us consider this circuit for decoding a word (row or column) and following the steps of the algorithm described in Subsection 2.4. We organised the elementary decoder as three functional blocks and a RAM memory, shown in Figure 5. We call this elementary decodeur 1-PU$_{t=1}$.

These functional blocks can be detailed as follows:

(1) *Reception*: This block executes all the calculations at the rhythm of the input symbols.
(2) *Treatment*: This block determines the positions in the received word that must be corrected, and applies the correction. For BCH ($t = 1$), the Chase-Pyndiah algorithm is used [4, 13] and for BCH ($t = 3$) Chien's algorithm is used. In this block, we select the codeword that has the minimal distance with $\boldsymbol{R}'_k$ and the concurrent codeword when there is one.
(3) *Emission*: This block computes the extrinsic information at the rhythm of the output symbols.
(4) *Memory*: There are two memory blocks used to store the data ($\boldsymbol{R}'$ and $\boldsymbol{R}$).

In Figure 6, we present the elementary decoder or processing unit (PU), processing symbol by symbol, that from now on we call 1-PU$_{t=1}$. The BTC used is obtained from extended BCH codes, correcting one error ($t = 1$). The SISO decoding algorithm was implemented with 5 quantisation levels, 16 test vectors and 3 competitors.

In the *reception part*, the functions that execute the subblocks are: computing the syndrome $S_0$, determining the five least reliable binary symbols, computing the parity, and the binary counter (0–31) that is used for timing the circuit.

In the *treatment part*, the syndrome $S_0$ is processed during two clock periods as is the syndrome $S'_1$ of the test sequence $C^0$ and so on, up to $C^{15}$, computing the square Euclidean distance and to class $M^d$, $M^{c^1}$, $M^{c^2}$ and $M^{c^3}$ where the final choice is made during the last two clock periods.

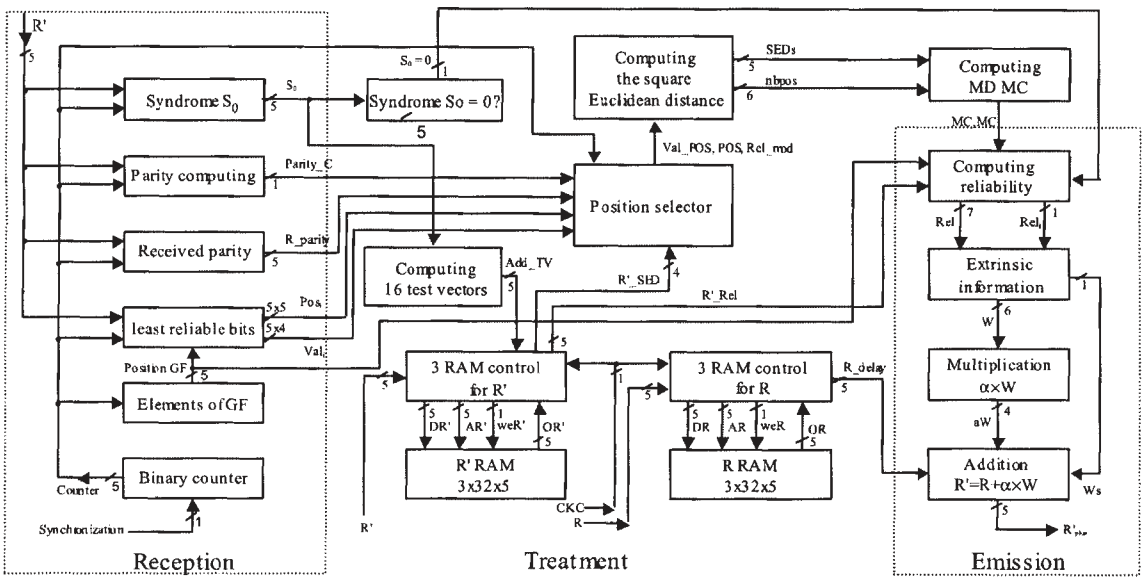


Figure 6. Elementary decoder or processing unit (1-PU$_{t=1}$).

In the *emission part*, the extrinsic information ($\boldsymbol{W}$) is computed, as well as the multiplication ($\alpha \times \boldsymbol{W}$) and addition ($\boldsymbol{R} + \alpha \times \boldsymbol{W}$).

Decoding a word (row or column) is carried out in three phases (reception, treatment and emission). Each phase requires 32 clock periods and the latency of the 1-PU$_{t=1}$ is 64 clock periods, equivalent to two words.

The time scale for decoding a word is illustrated in Figure 7.

The three phases of reception, treatment and emission are of equal duration (32 clock periods). To process the 16 test vectors, 2 clock periods are necessary for each one.

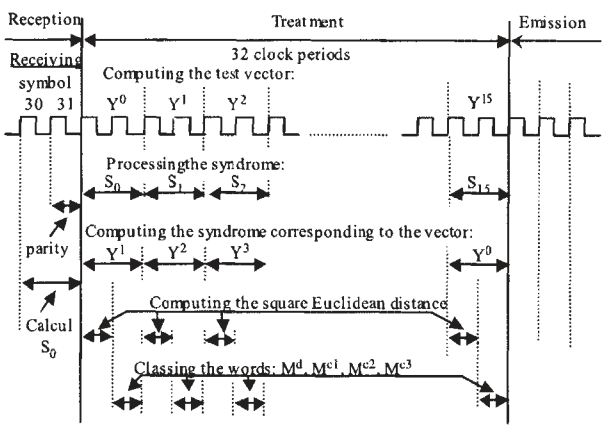


Figure 7. Steps of the decoding algorithm.

### *3.2. The block code BCH (128,106,8), correcting three errors ($t = 3$)*

Using the algorithm that we presented in Subsection 2.4 and considering the diagram of Figure 3, in Figure 8 we present the PU, processing symbol by symbol that from now on we call 1-PU$_{t=3}$. The BTC used is obtained from extended BCH codes, correcting three errors ($t = 3$). The SISO decoding algorithm was implemented with 5 quantisation levels, 16 test vectors and 3 competitors. The architecture organisation has the same principle as the circuit for $t = 1$ (reception, treatment and emission). Next we explain the three functional parts of the Processing Unit 1-PU$_{t=3}$.

The sub-blocks, that include the *reception part*, process the data at the rhythm of the input symbols and they work independently. Some of the functions that these sub-blocks perform are: computing the syndromes ($\mathbf{S}_1$, $\mathbf{S}_3$, $\mathbf{S}_5$), determining the positions of the least reliable bits, the RAM memory that stores $\boldsymbol{R}'$ and beta (reliability when there is no competitor vector). The sequencer (binary counter, 0–127) is used for timing the circuit.

In the *treatment part*, the operations cannot be executed until reception of the word is completed. The functions are known as sequential functions, since they use the results of the sub-blocks of the reception part. Some functions that the sub-blocks execute are the calculation of the coefficients of the error-locator polynomial ($\boldsymbol{\sigma}_1$, $\boldsymbol{\sigma}_2$, $\boldsymbol{\sigma}_3$), building the test sequences, the binary decoding, the selection of the

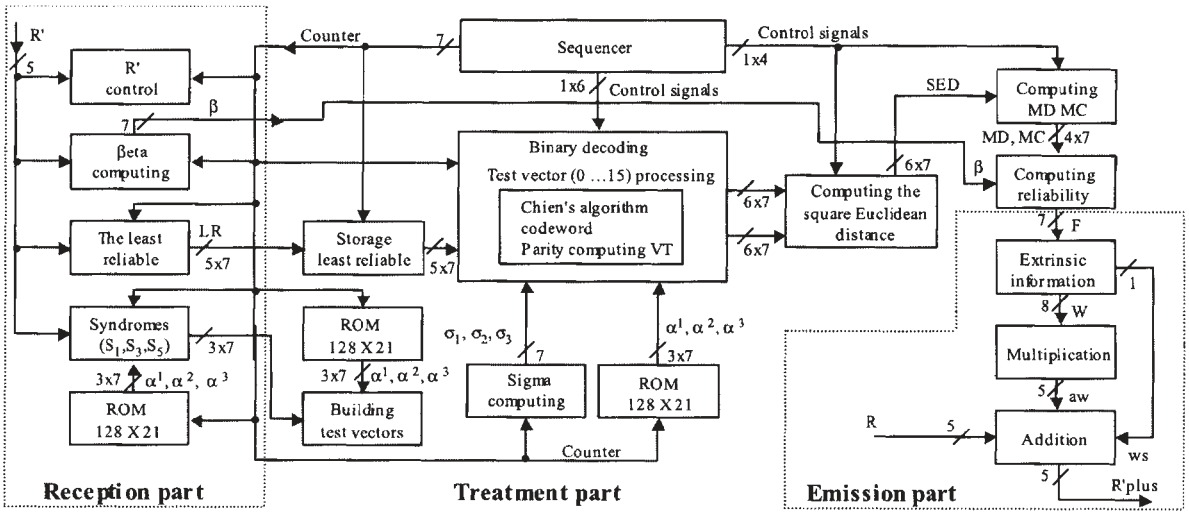


Figure 8. Elementary decoder or processing unit (1-PU$_{t=3}$).

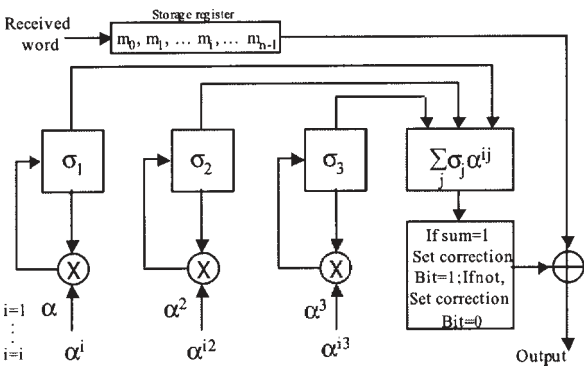


Figure 9. Block diagram of Chien's algorithm.

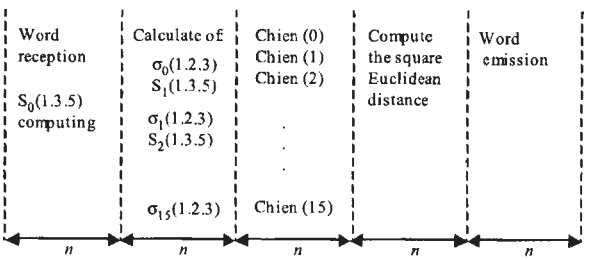


Figure 10. Steps for decoding a word.

maximum likelihood codeword and the concurrent codewords when there is at least one. To determine the roots of the error-locator polynomial $\boldsymbol{\sigma}(X)$, we use Chien's algorithm.

Figure 9 illustrates the block diagram of Chien's algorithm correcting three errors ($t = 3$) for the binary decoding for each test vector, where $\alpha$ is a primitive element of $\mathbf{GF}(2^x)$.

In the *emission part*, the functions are executed at the rhythm of the output symbols, also known as output sequential functions. The extrinsic information, the multiplication ($\alpha \times \boldsymbol{W}$) and addition ($\boldsymbol{R} + \alpha \times \boldsymbol{W}$) are computed in this part.

Decoding a word (row or column), is carried out in three phases (reception, treatment and emission). The reception and emission require $n = 128$ clock periods. The time scale for decoding a word is illustrated in Figure 10. The latency of the 1-PU$_{t=3}$ is 512 clock periods equivalent to four words.

## 4. TURBO DECODING ARCHITECTURES AT HIGH DATA RATES

Numerous applications in optical transmission and data storage require very high data rates. Also the sequential architecture (as Figure 2) must be used. The object of this section is to introduce the principles of the architectures in the turbo decoding of product codes at high rates. First, we describe the traditional architecture, then we introduce the principles for the new architecture at high data rates.

### *4.1. Traditional architecture*

In the traditional architecture, we consider that PUs work at a maximum frequency $F_{PUmax}$ given by the maximum operating frequency of the elementary decoders. If the

input frequency $F_{IN}$ is higher (high data rates), this means duplicating the turbo decoders in order to process more data simultaneously. With $m$ decoders in parallel it is possible to process $n_1$ (or $n_2$) data at the same time if access to the memory is carried out at different moments. To obtain a gain factor of $n_1$ (or $n_2$) in the ratio $F_{IN}/F_{PUmax}$ by processing $n_1$ (or $n_2$) data at the same time, it is required that the memory frequency operation is equal to $m \times F_{PUmax}$ which is a considerable disadvantage. This design requires $m$ memories to process $n_1$ (or $n_2$) data at the same time, which is another important disadvantage [19].

### *4.2. Principles for the new architecture at high data rates*

In the previous section, we concluded that it is indispensable to have $m$ decoders in parallel with $m$ memories for the turbo decoding of product codes at high data rates.

Now, we propose a memory splitting that allows us to read or to write several data at the same memory address [19]. With this concept of partition of the memory, several adjacent data can be stored at the same memory address to increase the data rate.

We take two adjacent rows and two adjacent columns of the initial matrix. Let us consider four data of the original matrix noted $(i,j)$, $(i+1,j)$, $(i,j+1)$ and $(i+1,j+1)$ where $i$ designates the row and $j$ the column. These four data constitute a word of the new matrix with four times fewer addresses. Figure 11 shows the representation of data in the new matrix with four times fewer addresses $(I,J)$.

If $n_1$ and $n_2$ are even then:

$$i = 2 \times I - 1 \quad \text{if} \quad (1 \leqslant I \leqslant n_1/2) \text{ and}$$
$$j = 2 \times J - 1 \quad \text{if} \quad (1 \leqslant j \leqslant n_2/2)$$

The data rows $(i,j)$, $(i,j+1)$ are decoded by the first PU and $(i+1,j)$, $(i+1,j+1)$ by the second PU while the data columns $(i,j)$, $(i+1,j)$ are decoded by the first PU and $(i,j+1)$, $(i+1,j+1)$ by the second PU.

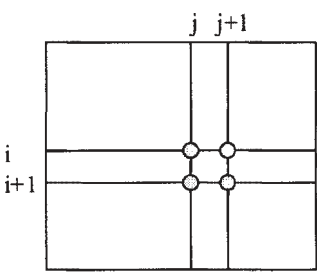


Figure 11. Representation of data in the new matrix.

We conclude that the reading/writing data rate is four times faster that before. In order to emphasise the main characteristic of the new architecture, we conclude by saying that if a word of the new matrix contains $m$ data per row and $m$ data per column, the reading (or writing) data rate is $m$ times faster than the reading (or writing) data rate of $m$ PU using the traditional architecture. This organisation of data does not require either particular memory architectures, or higher speed [19].

## 5. TURBO DECODING DESIGNS AT HIGH RATES

In this section, we present elementary decoder designs at high data rates and with strong error correction power, processing $m = 2$, 4, and 8 data simultaneously.

### *5.1. Reference prototype*

For the turbo decoding architecture designs of BCH (32,26,4) codes correcting one error ($t = 1$), we have taken as a reference the circuit implemented in 2000 [9, 17], which consists of one-chip turbo decoder of BCH (32,26,4) $\otimes$ BCH (32,26,4). The circuit processes data by data (1-$PU_{t=1}$) and the decoding algorithm was implemented with 5 quantisation levels, 16 test vectors and 3 competitors. With 7.5 iterations, the decoder achieves results close to the theoretical limit as described in [9, 17].

Previous studies and implementation at ENST Bretagne consisted of the integration on silicon of the iterative decoding algorithm of extended BCH codes, correcting one error ($t = 1$) (Hamming codes) and two errors ($t = 2$) [9, 20, 21]. The circuit presented in Subsection 2.2, Figure 8 is a continuity of this study and it is our reference for the new architecture for the turbo decoding of product codes at high data rates with strong error correction power, $t = 3$ (minimal distance of 64).

### *5.2. Design generalities*

The most important sub-circuit of a decoder is the elementary decoder or processing unit which performs the half-iteration decoding (columns or rows) of the data matrix.

Let us consider the elementary decoder architecture for decoding a column or a row of any product code. It has two memory blocks, to store the information vectors of $[\boldsymbol{R}]$ and $[\boldsymbol{R}']$. Each block is composed of three RAMs of size ($n_1$ or $n_2$) $\times\, q$ bits where $q$ represents the quantisation level bits. These RAMs are used for the reception, treatment and emission of the word, working in circular form. The

organisation of these memories must allow us to read or to write $m = 2$, 4 and 8 data simultaneously.

It is possible to consider the use of parallel architecture, if we take as an example the counter which synchronises the rest of the elementary decoder.

Let us assume that $n = 32$ and $m = 1$, then the decoding time of reception, treatment or emission is $n/m$ (32/1) clock periods. Now, we assume that $n = 32$ and $m = 8$, so the decoding time is eight times faster than before.

In short, for decoding the product code BCH (32,26,4) $\otimes$ BCH (32,26,4), $t = 1$ the execution time for each decoding part (reception, treatment and emission) is $32/m$ and for the product code BCH (128,106,8) $\otimes$ BCH (128,106,8), $t = 3$ is $128/m$ for the reception and emission.

When the elementary decoder processes more than one data at the same time, the processing time of one word is divided by $m$ (number of data processed) which implies that the architecture must execute functions in parallel. We increase the speed in decoding but also the area of the elementary decoder.

We have described the way in which this new architecture stores several data at the same memory address and the concept of parallel architecture. These concepts are the focus of the designs of each elementary decoder. To decode the product code at high data rates with strong error correction power, we use the same algorithm as in Subsection 2.4, and the principle of the architectures of Figure 3, and Figure 5, but now for high data rates and strong error correction power. We present in the following sections the design complexity when processing $m = 2$, 4 and 8 data simultaneously [10].

### *5.3. Design complexity for the turbo decoding of product codes using extended BCH codes*

In this section, we present the design complexity of elementary decoders at high rates for extended BCH (32,26,4) codes correcting one error ($t = 1$) and at high rates with strong error correction power for extended BCH (128,106,8) codes correcting three errors, $t = 3$.

*5.3.1. Turbo decoding of product codes using extended BCH (32,26,4) codes, $t = 1$.* The design of each decoder ($m$-PU$_{t=1}$, $m = 2$,4 and 8) is based on the concept of a parallel decoding architecture, which is of great importance to obtain the high data rate. These designs are an extension of the reference circuit (1-PU$_{t=1}$); we used the same algorithm, the same product code but $m = 2$, 4 and 8. Thus, we were able to validate the results of the three new designs. With VHDL, we described the elementary decoder designs and with Synopsys VSS tools their simulations. Then, we compared the result of each elementary decoder and the C simulation. In this way we validated the correct operation of the designs. For each elementary decoder, the area evaluation, in the ST Microelectronics CMOS 0.18 µm target library (where one gate is about 12.5 µm$^2$), was determined from the synthesis with $m = 1$, 2, 4 and 8 data. For the four elementary decoders, Table 1 shows the area in gate units and their critical path.

Table 1(a). Area and critical path for $m$-PU$_{t=1}$).

| | Area in gate units | Critical path |
|---|---|---|
| 1-PU$_{t=1}$ | 5500 | 11.5 ns |
| 2-PU$_{t=1}$ | 7000 | 10.5 ns |
| 4-PU$_{t=1}$ | 11 200 | 12.5 ns |
| 8-PU$_{t=1}$ | 21 800 | 11.5 ns |

Table 1(b). Details on the area for the different process units.

| | 1-PU$_{t=1}$ | 2-PU$_{t=1}$ | 4-PU$_{t=1}$ | 8-PU$_{t=1}$ |
|---|---|---|---|---|
| Sequencer | 156 | 101 | 50 | 26 |
| Syndrome, $S_0$ | 82 | 89 | 103 | 136 |
| Parity | 72 | 60 | 63 | 82 |
| The least reliable | 866 | 1171 | 1556 | 3074 |
| Building test vectors | 426 | 406 | 550 | 702 |
| Computing square Euclidean distance | 192 | 192 | 394 | 782 |
| Computing MD_MC | 2188 | 2265 | 2732 | 3962 |
| Computing reliability | 627 | 1052 | 2022 | 3242 |
| Extrinsic information | 161 | 327 | 640 | 1242 |
| Multiplier ($\alpha \times \boldsymbol{W}$) | 487 | 853 | 1866 | 3435 |
| Computing of $\boldsymbol{R}'$ | 83 | 165 | 355 | 696 |
| Memory management ($f = 100$ MHZ) | 153 | 446 | 592 | 1202 |
| Total area | 5500 | 7000 | 11 200 | 21 800 |
| | S = 5500 | 1.25 × S | 2 × S | 4 × S |

Considering that the critical path is similar for the four $m$-PU$_{t=1}$. The results in Table 2 were synthesised with a frequency $f = 100$ MHz (clock period of 10 ns). This table shows that the latency of the $m$-PU$_{t=1}$ is divided by $m$. The

Table 2. Results of the $m$-PU$_{t=1}$ implementation.

| $m$-PU$_{t=1}$ | Data rate | Latency | $m$-PU$_{t=1}$ area (gates) | Traditional $m$-PU area (gates) |
|---|---|---|---|---|
| $m = 1$ | $D_1$ | L | 5500 = S | S |
| $m = 2$ | 4 $D_1$ | L/2 | 7000 | 1.25 × 2 × S |
| $m = 4$ | 16 $D_1$ | L/4 | 11 200 | 2.00 × 4 × S |
| $m = 8$ | 64 $D_1$ | L/8 | 21 800 | 4.00 × 8 × S |

Table 3. Comparison of $m$-PU$_{t=1}$ versus traditional $m$-PU operating at $m^2$ D$_1$.

| | $m$-PU$_{t=1}$ | Traditional $m$-PU |
|---|---|---|
| Data rate | $m^2 \times D_1$ | $m^2 \times D_1$ |
| Number of PU | $m$ | $m^2$ |
| Area | $\approx S \times m^2/2$ | $S \times m^2$ |
| Latency | $L/m$ | L |
| RAM size | $S_{RAM}$ | $m^2 \times S_{RAM}$ |

data rate of $m$-PU$_{t=1}$ is increased by $m$ and the area necessary for the traditional $m$-PU ($m$ decoders in parallel and $m$ memories, cf. 4.1) is approximately increased by $m \times m/2$, the memory being constant (the traditional architecture operating at the same data rate uses $m^2$ memories).

In Table 3, we compare the results obtained with the new architecture versus the traditional architecture when the data rate is the same $m^2 \times D_1$. The 1-PU$_{t=1}$ is the reference circuit; 2-PU$_{t=1}$, 4-PU$_{t=1}$ and 8-PU$_{t=1}$ are the new designs.

From Table 3 we conclude that:

(1) For the same data rate ($m^2 \times D_1$), the number of $m$-PU$_{t=1}$ required is only $m$ versus $m^2$ in the traditional solution.
(2) The latency for $m$-PU$_{t=1}$ is divided by $m$.
(3) The area of the traditional $m$-PU is approximately increased by $m^2/2$.
(4) The new solution divides the required memory area (RAM) by a factor $m^2$.

Thus, with 8 decoders 8-PU$_{t=1}$ in parallel, the data rate is $8^2 \times 100$ Mbits/s $= 6.4$ Gbits/s and the area is multiplied by 32. To obtain the same data rate with 1-PU$_{t=1}$, we must use 64 1-PU$_{t=1}$ and 64 RAMs.

*5.3.2. Turbo decoding of product codes using extended BCH (128,106,8) codes, $t = 3$.* For elementary decoder design at high data rates with strong error correction power (2-PU$_{t=3}$, 4-PU$_{t=3}$ and 8-PU$_{t=3}$), we followed the SISO algorithm presented in Subsection 2.4 and as a reference the circuit presented in Subsection 3.2, Figure 8. Thus the new designs are an extension of the reference circuit (1-PU$_{t=3}$). We used the same algorithm and the same product code but now with $m = 2$, 4 and 8.

In Table 4, we have summarised the results, which show the complexity of four elementary decoders: 1-PU$_{t=3}$, 2-PU$_{t=3}$, 4-PU$_{t=3}$ and 8-PU$_{t=3}$. An important characteristic of the result is the critical path (100 ns) which is similar for the four elementary decoders. With this characteristic we synthesised the four elementary decoders with a frequency $f = 100$ MHz.

Table 4. Area for the different decoders.

| | Processing unit | | | |
|---|---|---|---|---|
| | 1-PU$_{t=3}$ | 2-PU$_{t=3}$ | 4-PU$_{t=3}$ | 8-PU$_{t=3}$ |
| Number of gates excluding RAM | 40 800 = S | 56 500 | 81 800 | 120 300 |
| Normalised surface | S | $1.4 \times S$ | $2.0 \times S$ | $3.0 \times S$ |

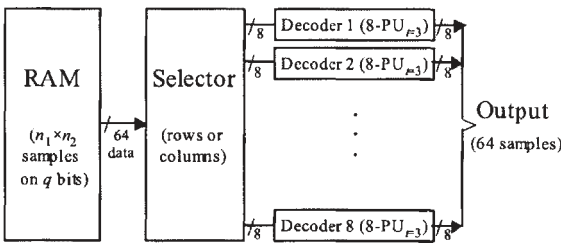


Figure 12. Block architecture decoding 64 data simultaneously (data rate = 6.4 Gbits/s).

Table 4 shows that when the 8-PU$_{t=3}$ decoder works in bytes, it uses three times more area with a data rate eight times greater.

With eight decoders in parallel it is possible to obtain a data rate of 6.4 Gbits/s with a silicon surface 24 times greater than 1-PU$_{t=3}$. This architecture uses only one memory; the classical architecture operating at the same data rate uses $m^2$ memories. Figure 12 illustrates the block architecture with a whole memory of $n_1 \times n_2$ symbols on $q$ bits (RAM), the selector block which selects the rows or columns and the eight decoders that allow 64 data to be decoded at the same time.

In Table 5, we explain in detail the area of 1-PU$_{t=3}$ blocks.

## 6. CONCLUSION

Two new turbo decoding architectures for very high data rate and strong error correction power have been presented in this paper. The first one is designed using product codes based on extended BCH (32,26,4) codes, correcting one error $t = 1$ and the second one based on extended BCH (128,106,8) codes, correcting three errors $t = 3$. These innovative architectures can be applied to convolutional codes or linear block codes. We have analysed only the case of linear block codes. An important characteristic of

Table 5. Different block areas of 1-PU$_{t=3}$.

| Blocks | Gates |
|---|---|
| Reception part | |
| Séquencer | 266 |
| Management $\boldsymbol{R}'$ | 360 |
| Beta computing | 414 |
| The least reliable symbols | 1188 |
| Syndrome computing | 408 |
| | **2638** |
| Treatment part | |
| Computing of the square Euclidean distance | 968 |
| Sigma computing | 1694 |
| Chien (16 times) | 14 562 |
| Codeword (16 times) | 4119 |
| Parity computing VT (16 times) | 245 |
| Gestion des LBM | 6767 |
| Metric computing | 785 |
| Computing MD_MC | 5346 |
| | **34 486** |
| Emission part | |
| Reliability computing | 3395 |
| Extrinsic information | 197 |
| Multiplier: $\alpha$ W | 487 |
| Adder: $\boldsymbol{R}' = \boldsymbol{R} + \alpha \boldsymbol{W}$ | 83 |
| | **4162** |
| Total | $\sim$40 800 |

these new designs is that the decoder uses only one RAM for decoding while a traditional architecture uses $m$ memories. We have presented the way in which these architectures are able to process several data ($m = 2$, 4 and 8) simultaneously, storing these data at the same memory address and using the concept of parallel processing architecture in order to achieve high data rates in decoding. The decoder designs presented in this paper are based on the SISO decoding algorithm, which has been implemented with 5 quantisation levels, 16 test vectors and 3 competitors. We have compared the more important characteristics of the new designs with those of a traditional architecture.

We divided the elementary decoder architecture into three functional parts: reception, treatment and emission. This allowed us to visualise the function of each block, facilitating their design.

To decode the product code BCH $(32,26,4)^2$ each part requires 32 clock periods with a latency of 64 clock periods (data rate equals clock rate). The results obtained show that the turbo decoding is increased by a factor $m^2$ when $m$ elementary decoders ($m$-PU$_{t=1}$, $m = 1$, 2, 4 and 8) processing $m$ data simultaneously are used and its latency is divided by a factor $m$. The area of the $m$ processing units is increased by a factor $m^2/2$ while the memory is constant. To increase the data rate of a turbo decoder by duplicating decoders (classical solution) by a factor $m^2$, we must use $m^2$ turbo decoders in parallel combined with a demultiplexer. The new solution divides the required memory area by a factor $m^2$.

In this paper, we have also presented the latest results on block turbo decoder design, using BCH (128,106,8) codes with strong error correction power ($t = 3$) and high code rate ($R \approx 0.8$). To decode the product code BCH $(128,106,8)^2$, the reception and emission each require 128 clock periods, and the data rate is equal to the clock rate. This architecture uses only one RAM for decoding while the traditional architecture uses $m$ memories. For this architecture, we have presented the design and complexity of three elementary decoders for very high data rates with strong error correction power, which use BCH (128,106,8) codes correcting three errors. These decoders can process $m = 2$, 4 and 8 data simultaneously (2-PU$_{t=3}$, 4-PU$_{t=3}$ and 8-PU$_{t=3}$,). After the synthesis of the elementary decoders, we have found that the surface of 1-PU$_{t=3}$ is $S1 = 40\,800$ gates, thus the elementary decoder surface of 2-PU$_{t=3}$ is $S_2 = 1.4 \times S_1$, of 4-PU$_{t=3}$ is $S_4 = 2 \times S_1$ and that of 8-PU$_{t=3}$ is $S_8 = 3 \times S_1$.

In both architecture designs, we consider a clock rate of 100 MHz with $m = 8$. The turbo decoder can achieve a data rate of 6.4 Gbit/s with cascaded decoders (using only one RAM). These results open the way to numerous applications and the possibility of future integration on silicon of these circuits.

On the other hand, as a continuity to the work presented in this paper, the researchers at 'ENST-Bretagne' have started the implementation of a turbo decoder architecture to decode the product code constructed with elementary code BCH (128,92,12) correcting five errors ($t = 5$), with a minimum distance of 144 and a code rate close to ($R = 0.51$).

## REFERENCES


1. Berrou C, Glavieux A, Thitimajshima P. Near Shannon limit error-correcting coding and decoding: turbo-codes (1). *IEEE International Conference on Communication ICC' 93* 1993; **2/3**:1064–1071.
2. Reddy SM. On decoding iterated codes. *IEEE Transactions on Information Theory* 1970; **IT-16**:624–627.
3. Reddy SM, Robinson JP. Random error and burst correction by iterated codes. *IEEE Transactions on Information Theory* 1972; **IT-18**:182–185.
4. Pyndiah R, Glavieux A, Picart A, Jacq S. Near optimum decoding of product codes. In *Proceedings of IEEE GLOBECOM'94 Conference*, Vol. 1/3, November–December 1994, San Francisco; pp. 339–343.

5. Pyndiah R. Near optimum decoding of product codes: block turbo codes. *IEEE Transactions on Communications* 1998; **46**(8): 1003–1010.
6. Pyndiah R. Iterative decoding of product codes: block turbo code. *International Symposium on Turbo Codes and Related Topics*, Brest, September 1997; pp. 71–79.
7. Elias P. Error-free coding. *IRE Transactions on Information Theory* 1954; **IT-4**:29–37.
8. Macwilliams FJ, Sloane NJA. *The Theory of Error Correcting Codes*. North-Holland publishing company: Amsterdam, 1978; pp. 567–580.
9. Kerouédan S, Adde P. Implementation of a Block Turbo Coder on a single chip. *2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, 2000.
10. Cuevas J, Adde P, Kerouédan S, Pyndiah R. New architecture for high data rate turbo decoding of product codes. *GLOBECOM 2002*, Taipei, Taiwan, 17–21 November 2002; pp. 139–143.
11. Adde P, Pyndiah R, Buda F. Design and performance of a product code turbo encoding-decoding prototype. *Annals of Telecommunications* 1999; **54**(3–4):214–219.
12. Chase D. A class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory* 1972; **IT-18**:170–182.
13. Adde P, Pyndiah R, Raoul O. Performance and complexity of block turbo decoder circuits. *Third International Conference on Electronics, Circuits and System* ICECS'96, 13–16 October 1996, Rodos, Greece; pp. 172–175.
14. Adde P, Pyndiah R, Raoul O, Inisan JR. Block turbo decoder design. *International Symposium on Turbo Codes and Related Topics*, Brest, September 1997; pp. 166–169.
15. Peterson WW. Encoding and error correcting procedures for the Bose-Chaudhuri codes. *IRE Transformation Theory* 1960; **IT-6**:459–470.
16. Chien RT. Cyclic decoding procedures for the Bose-Chaudhuri Hocquenghen codes. *IEEE Tranformation Theory* 1964; **IT-10**: 357–363.
17. Kerouédan S, Adde P, Pyndiah R. How we implemented block turbo codes? *Annals of Telecommunications* 2001; **56**(7–8):447–454.
18. Kerouédan S, Adde P, Ferry P. Comparaison performance/complexité de décodeurs de codes BCH utilisés en turbo-décodage. *Gretsi*'99, Vannes, France, September 1999; pp. 172–175, 13–17.
19. Adde P, Pyndiah R. Architecture de décodeur de code produit haut débit. *Gretsi'01*, Toulouse, France, 10–13 September 2001.
20. Raoul O, Adde P, Pyndiah R. Architecture et conception d'un circuit turbo-décodeur de codes produits. *Gretsi'95*, Tome 2, Juan les Pins, France, 18–21 September 1995; pp. 981–984.
21. Adde P, Kerouédan S, Inisan JR. Conception d'un décodeur BCH (30,19,6) à entrées et sorties pondérées: Application au turbo décodage. *Gretsi'99*, Vannes, France, 13–17 September 1999; pp. 103–106.