New architecture for high data rate turbo decoding of product codes

Javier Cuevas Ordaz, Patrick Adde, Sylvie Kerouedan, Ramesh Pyndiah

NEW ARCHITECTURE for HIGH DATA RATE TURBO DECODING of PRODUCT CODES

Javier CUEVAS, Patrick ADDE, Sylvie KEROUEDAN and Ramesh PYNDIAH
ENST Bretagne, Technopôle Brest Iroise, BP 832, 29285 BREST Cedex , France.

E-mail: Javier.Cuevas@enst-bretagne.fr, Patrick.Adde@Enst-Bretagne.fr, Sylvie.Kerouedan@enst-bretagne.fr and Ramesh.Pyndiah@Enst-Bretagne.fr

Tel : +33 2 29 00 10 91 Fax : +33 2 98 00 11 84

Abstract - **This paper presents a new circuit architecture for turbo decoding, which achieves very high data rates when using product codes as error correcting codes. Although this architecture is independent of the elementary code (convolutional or block) used and of the corresponding decoding algorithms, we focus here on the case of product codes. This innovative circuit architecture stores several data at the same address and performs parallel decoding to increase the data rate. It is able to process several data simultaneously with one memory (classical designs require *m* memories); its latency decreases when the amount of data processed simultaneously is large. We present results on block turbo decoder designs of 2-data, 4-data and 8-data decoders (where 2, 4 and 8 are the number of data symbols processed simultaneously). For each decoder circuit, the latency is decreased, the area of the processing unit is increased by a factor *m* and the critical path and memory size are constant (the data rate is increased by m^2 if we have *m* parallel decoders).**

I. INTRODUCTION

The concept of turbo codes was introduced by C. Berrou [1] in 1993. The coding scheme is based on the parallel concatenation of two recursive systematic convolutional codes separated by a non-uniform interleaver. Decoding uses an iterative process with soft input-soft output (SISO) decoders. This principle, known as convolutional turbo codes (CTC), has exceptional performance close to the Shannon limit.

An alternative solution to CTCs are block turbo codes (BTC) proposed in 1994 by R. Pyndiah [2][3][4]. This coding scheme uses the series concatenation of block codes (product codes, introduced in 1954 by P. Elias [5]) and the iterative process is also obtained using SISO elementary decoders. Series concatenation guarantees a large minimum Hamming distance (9, 16, 24, 36 and higher) even for relatively small data blocks. The extrinsic information which has a significant role in turbo decoding is used not only for data bits but also for redundancy bits at each iteration.

The object of this paper is to recall the basic principles of decoding for product codes: their construction and the principles of turbo decoding. Then we present high data rate turbo decoding architectures, the classical and an innovative approach. Finally, we present the latest results on block turbo decoder design for high data rates based on massive parallel decoding.

II. BASIC PRINCIPLES FOR BTC DECODING

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

1. Construction of Product codes

We consider two systematic linear block codes C_1 with parameters (n_1,k_1,δ_1) and C_2 with parameters (n_2,k_2,δ_2), where n_i, k_i and δ_i (i=1,2) stand for codeword length, number of information bits, and minimum Hamming distance, respectively. The product code is presented in the form of a matrix **C** with n_1 rows and n_2 columns where:

1.- the binary information is represented by a sub-matrix **M** of k_1 rows and k_2 columns

2.- the k_1 rows of the matrix **M** are coded by the code C_2.

3.- the n_2 columns of the matrix **C** are coded by the code C_1.

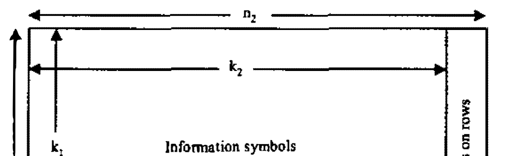

Figure 1 : Construction of product code $P = C_1 \otimes C_2$.

The parameters of the resulting product code P are $n= n_1 x\, n_2$, $k= k_1 x\, k_2$, $\delta=\delta_1 x \delta_2$ and code rate R is given by $R=R_1 x R_2$, where R_i is the code rate of code C_i illustrated in Figure 1. If the code C_1 is linear and systematic, then (n_1-k_1) rows built by C_1 are words of the code C_2 and can thus be decoded like the

k_1 first rows. In this case, the product code is characterized by n_1 words of code of C_2 along rows, and by n_2 words of code of C_1 along the columns. The codes C_1 and C_2 can be obtained from elementary convolutional codes or linear block codes (extended BCH code).

2. Principles of turbo decoding

The turbo decoding of product codes uses an iterative decoding structure consisting of cascaded soft input-soft output decoders for the rows and columns of matrix **C** [6][2]. Between each decoding, a reconstruction of the matrix is essential to recover the codewords when decoding is completed, as illustrated in Figure 2.

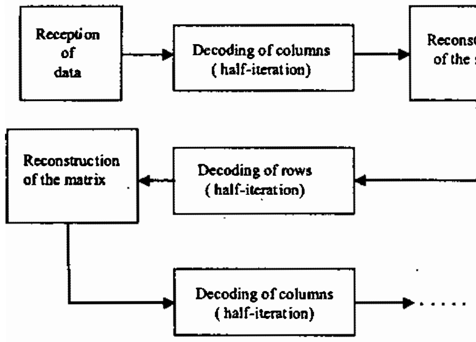
Reception of data
Decoding of columns (half-iteration)
Reconstruction of the matrix
Reconstruction of the matrix
Decoding of rows (half-iteration)
Decoding of columns (half-iteration)
.....

Figure 2: Iterative process of turbo decoding.

In general, the data processing between two half-iterations (a row and column decoding) is illustrated in Figure 3. Consider $\mathbf{R}_k$ the information received from the channel, $\mathbf{R'}_k$ the information that comes from the previous half-iteration and $\mathbf{R'}_k^+$ is the information sent to the following half-iteration. The output of each half-iteration is equal to $\mathbf{R}_k$ plus the extrinsic information, $\mathbf{W}_k$, multiplied by *alpha*. This extrinsic information corresponds to the contribution of the previous decoder to the data bits. It is obtained by the difference between the soft output $\mathbf{F}_k$ and the soft input of this decoder. *alpha* and β are constants varying with the current half-iteration.

Then we consider the SISO decoder as a block with $\mathbf{R}_k$ and $\mathbf{W}_k$ as inputs, delivering $\mathbf{R}_k^+$ et $\mathbf{W}_k^+$ with a latency L (delay to perform the decoding algorithm). In the following we call the SISO elementary decoder the Processing Unit (PU).

In the PU, the different steps of the algorithm are given below [7][8]:

1- Search for the p least reliable binary symbols of $[\mathbf{R'}_k]$; their positions are called $I_1, I_2, \ldots, I_p$ and the reliabilities of these symbols are $MF_1 = |r_{I1}|$, $MF_2 = |r_{I2}|, \ldots, MF_p = |r_{Ip}|$

2- Generate τ test vectors $[\mathbf{T}^Q]$ $Q \in \{1, \ldots, \tau\}$ which are a combination of elementary test vectors $[\mathbf{T}^j]$ having "1" in position $\mathbf{I}_j$ and "0" elsewhere.

3- For each test vector $[\mathbf{T}^Q]$, compute:

$$[\mathbf{Z}^Q] = [\mathbf{T}^Q] \oplus \text{sign}([\mathbf{R'}_k]+1)/2).$$

4- Decode $[\mathbf{Z}^Q]$ using the Berlekamp algorithm (result $[\mathbf{Y}^Q]$).

5- For each vector $[\mathbf{Y}^Q]$, compute the square Euclidean distance $M_Y{}^Q$ between $[\mathbf{R}_k]$ and $[\mathbf{Y}^Q]$.

6- Select codeword $[\mathbf{Y}^D]$ having the minimal distance with $[\mathbf{R'}_k]$. Then $[\mathbf{D}] = [\mathbf{Y}^d]$ is the result of binary decoding.

7- Select the closest test vectors of $[\mathbf{D}]$, called competitors.

8- Compute the reliability for each element d_j of $[\mathbf{D}]$.

9- Compute the extrinsic information.

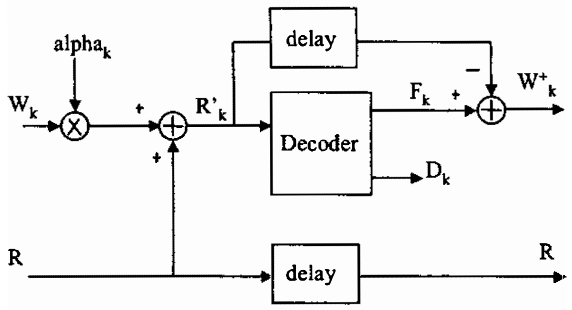

Figure 3: Block diagram of half-iteration (PU) (elementary decoder).

The hardware of the PU is split into five parts as illustrated in Figure 4:

1- the input sequential part concerns all the functions where calculations are performed at the rhythm of the input symbols (a counter used for the timing of the elementary decoder, the parity, the syndrome and the five least reliable binary symbols of the input sequence).

2- the algebraic decoding part determines the optimum codeword for a given input binary vector.

3- the selection part selects the maximum likelihood codeword (that is, the word at minimum distance of $[\mathbf{R'}_k]$) and the concurrent codewords when there is at least one (we choose to store a maximum of three concurrent code words).

4- the output sequential part computes the extrinsic information where calculations are performed at the same rate as the output symbols.

5- two storage elements are divided into three RAMs in order perform some operations in parallel (writing, computing, reading).

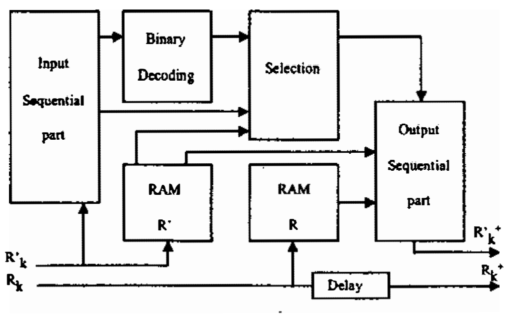

Figure 4: Block diagram of a half-iteration decoder

III. ARCHITECTURES FOR HIGH RATES

In this section we present both the classical and the new architectures for the turbo decoding of product codes at very high data rates.

1. Classical approach

The critical path in turbo decoders is located in the Processing Units PU. For a given technology, we will consider that the PU works at a maximum frequency F_{TUmax}, which is given by the maximum operating frequency of the elementary decoders. If the frequency F_{rate} is higher, it will be necessary to duplicate the turbo decoders to process a greater number of data at the same time, as illustrated in Figure 5.

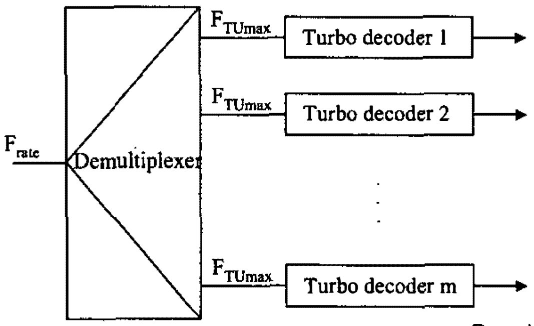

Figure 5: One architecture for high rates ($F_{rate} = m.\ F_{TUmax}$)

Product codes possess the property that codewords along the rows (or the columns) in the initial matrix **C** can be decoded independently. Thus, we can decode the rows (or columns) by duplicating the number of elementary decoders of the C_1 code (or C_2) as shown in Figure 6.

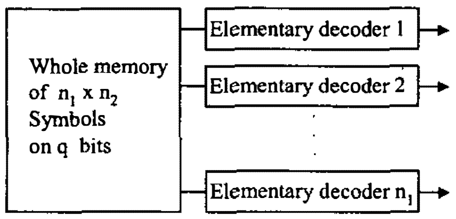

Figure 6: Architecture of high rate decoding several rows at the same time.

We can process a maximum number of n_1 (or n_2) words provided the access to the memory, in reading or in writing, takes place at different instants (several memory points of a matrix cannot be read or written at the same time, unless using "multi-ports" RAM). It is possible to gain a factor n_1 (or n_2) in the ratio F_{rate}/F_{TUmax} by processing n_1 (or n_2) binary elements simultaneously. The disadvantage of this architecture is that the memory must operate at a frequency equal to $m.F_{Tumax}$, where m is the number of decoders in parallel.

2. Innovative approach

To increase the data rate, we store several data at the same address. However, it is necessary to process these data by row and by column. At this address the data are adjacent for reading (or writing), by row and column. Let us consider two adjacent rows and two adjacent columns of the initial matrix, as illustrated in Figure 7.

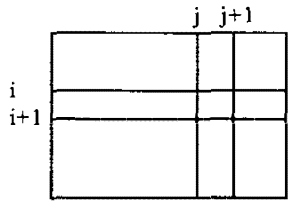

Figure 7: Representation of data in the new matrix

The 4 data (i, j), (i, j+1), (i+1, j) and (i+1, j+1) constitute a word of the new matrix that has 4 times fewer addresses (I, J). If n_1 and n_2 are even,

then if $1 \leq I \leq n_1/2$, $i=2*I-1$.
if $1 \leq J \leq n_2/2$, $j=2*J-1$.

To decode the rows, the data (i, j), (i, j+1) are processed by PU1, (i+1, j) and (i+1, j+1) by PU2. To decode the columns, the data (i, j), (i+1, j) are processed by PU1 and (i, j+1), (i+1, j+1) by PU2. The reading (or writing) data rate is 4 times faster than previously, as illustrated in Figure 8.

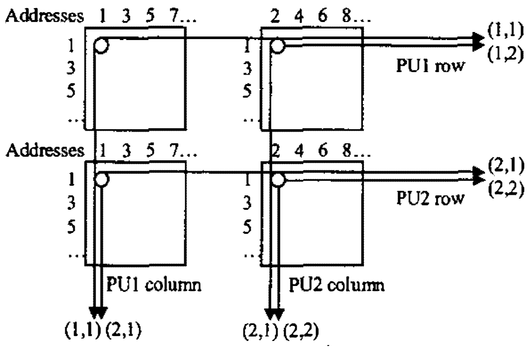

Figure 8: Example of memory splitting into 4 parts.

In short, if a word of the new matrix contains m data by row and m data by column, the reading (or writing) data rate is m^2

times faster than the reading (or writing) data rate of *m* PU using the classical approach. This organization of data does not require either particular memory architectures, or higher speed.

IV. TOWARDS THE IMPLEMENTATION OF A HIGH RATE TURBO DECODER.

By using the method described in section III, we can increase the data rate of BTC by a factor *m* if we use *m* parallel decoders for the rows or columns. One of the other advantages of this solution is that *m* data bits are read simultaneously from the memory for each of the rows (or columns). We shall now use this property to increase the data rate of the PU by processing *m* data bits per clock period.

1. Previous implementation: one-data-decoder

In our study, the reference circuit is the one-data-decoder implemented in 2000 [9]. This prototype consists of a one-chip turbo decoder of BCH(32,26,4)$\otimes$BCH(32,26,4). The algorithm was implemented with 5 quantization levels, 16 test vectors and 3 competitors. With 7.5 iterations, the decoder achieves results close to theoretical limit as described in [9] [10]. The decoding of a word is split into 3 phases: reception, processing and emission. Each phase requires 32 clock periods and the latency of the decoder is 64 clock periods.

2. New implementation

For the new architecture, we shall use the same product code BCH(32,26,4)$\otimes$BCH(32,26,4) and the same algorithm, but now we process *m* (2, 4 or 8) data simultaneously using 32/*m* clock periods (16, 8 and 4 respectively) for the decoding. The design of these 3 elementary decoders (PU) is an extension of the one-data-decoder. So we were able to compare the different solutions. The validation started with software simulations (C program), then VHDL functional simulations which were compared to the one-data-decoder simulations. The synthesis of the design was performed on a Synopsys Design Compiler for each decoder.

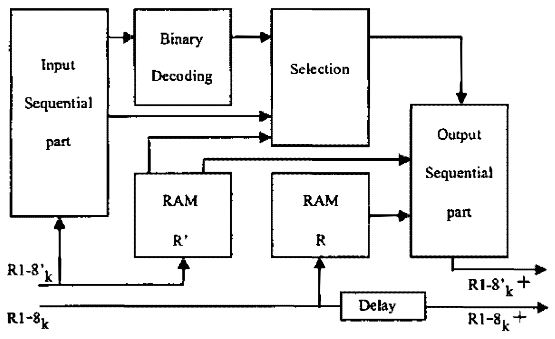

Figure 9: Block diagram of a half-iteration decoder processing 8 data simultaneously.

The design of each decoder is based on the concept of parallel decoding architecture. In figure 9 the 8-data-decoder architecture is illustrated.

The results summarized in Table 1 show the complexity of each PU. The area evaluation for each decoder is derived from the synthesis of the diagram in figure 9 with 2, 4 and 8 data, in the ST Microelectronics CMOS 0.18 µm target library. Table 1 shows the complexity in gate unit and their critical path for the different decoders.

	Number of gates excluding RAM	Critical path
1-data PU	5400	11.5 ns
2-data PU	6750	10.5 ns
4-data PU	11000	12.5 ns
8-data PU	18800	11.5 ns

Table 1: The area and its critical path for the different decoders.

The results obtained show that the critical path is similar for the four Processing Units. From then on, we synthesized the PU with F_{Tumax} = 100Mhz (clock period = 10ns). The main results are shown in table 2. The 1-data PU called PU1 is the reference, PU2 , PU4 and PU8 are respectively 2-data PU, 4-data PU and 8-data PU.

PU	*m*	Data rate	Latency	PU Area (gates)	*m* PU area (gates)
PU1	1	D_1	L	5500=S	S
PU2	2	4 D_1	L/2	7000	1.25x2xS
PU4	4	16 D_1	L/4	11200	2.0x4xS
PU8	8	64 D_1	L/8	21800	3.95x8xS

Table 2: Main characteristics of different decoders.

The latency of the PU*m* is divided by *m*. The data rate of PU*m* is increased by *m* and the necessary area of *m* PU*m* is approximately increased by *m.m*/2. The size of the memory (RAM), used to store $[R_k]$ and $[W_k]$, is the same in the all cases.

Thus, with PU8, the data rate is 6.4 Gbits/s and the area is multiplied by 32. To obtain the same data rate with PU1, we must use 64 PU1 and 64 RAMs. In table 3, we compare the characteristics of the decoder using PU*m* with the decoder using PU1 and operating at m^2 D_1.

	PUm	PU1
Data rate	$m^2 . D_1$	$m^2 . D_1$
Number of PU	m	m^2
Area	$\approx S . m^2/2$	$S . m^2$
Latency	L/m	L
RAM size	S_{RAM}	$m^2 \times S_{RAM}$

Table 3: Comparison of characteristics of the decoder using PUm with the decoder using PU1, and operating at m^2 D_1.

V. CONCLUSION

In this paper, we have proposed a new architecture for implementing very high data rate turbo decoders. This solution can be applied to convolutional codes and linear block codes. We have used the concept of parallel processing architectures as well as that of parallel data storage (at the same address) in order to increase the decoding speed. To illustrate the concept we have considered the example of a turbo decoder for product code BCH(32,26,4)$\otimes$BCH(32,26,4). The basic processing unit (row or column decoding) has the following characteristics: 5 quantization bits, 16 test vectors and 3 competitors and previous results have already shown that it is very efficient and can achieve near asymptotic performance [6][11]. It requires 32 clock periods for decoding a code word (data rate equals clock rate) and the latency is of 64 clock periods.

By using m decoders (or Processing Units: PU) processing m bits simultaneously with the new solution proposed in this paper, we have shown that the overall data rate of the turbo decoder can be increased by a factor m^2 and that the latency is divided by a factor m. The overall area of the m processing units is increased by a factor $m^2/2$ while that of the memory is constant. Note that the critical path in the processing units is nearly constant for the values of m considered here (2,4 and 8). 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 latency is not modified, the total area of the processing units is multiplied by m^2 and furthermore, the total area of the memory is also multiplied by m^2. Thus the new solution proposed here saves 50% in terms of the total area of the processing units. But what is even more important is that the new solution divides the required memory area by a factor m^2. For large turbo codes (usually high code rate product codes) the memory area can be much bigger than that of the processing unit and thus bring a tremendous saving on memory area.

By considering 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. By performing 4 iterations with the same turbo decoder (4 row and 4 column decoding) the decoder can achieve a data rate of 800 Mbit/s, which is to our knowledge the highest data rate achieved with turbo codes today. These results open the way to numerous applications in optical transmission, data storage and many others.

REFERENCES

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding : Turbo-codes (1), " *IEEE Int. Conf. on Comm. ICC' 93*, vol 2/3, May 1993, pp. 1064-1071.

[2] R. Pyndiah, A.Glavieux, A. Picart and S.Jacq, "Near optimum decoding of product codes, "in proc. of *IEEE GLOBECOM* '94 *Conference*, vol. 1/3, Nov.- Dec. 1994, San Francisco, pp. 339-343 .

[3] R. Pyndiah, "Near optimum decoding of product codes : Block Turbo Codes" *IEEE Trans. on Comm.*, vol 46, n° 8, August 1998, pp. 1003-1010.

[4] R. Pyndiah, "Iterative decoding of product codes : block turbo code," *International Symposium on turbo codes and related topics*, Brest, Sept. 1997, pp. 71-79.

[5] P. Elias, "Error-free coding," *IRE Trans. on Inf. Theory*, vol. IT-4, pp. 29-37, Sept. 1954.

[6] D. Chase, " A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory*, vol IT-18, Jan. 1972, pp 170-182.

[7] P. Adde, R. Pyndiah, O. Raoul, " 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.

[8] P. Adde, R. Pyndiah, O. Raoul and J.R. Inisan, "Block turbo decoder design," *International Symposium on turbo codes and related topics*, Brest, Sept. 1997, pp.166-169.

[9] S. Kerouédan and P. Adde, "Implementation of a Block Turbo Coder on a single chip" *2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, 2000.

[10] S. Kerouédan, P. Adde, and R. Pyndiah "How we implemented Block Turbo Codes?", *Annals of telecommunications*, 56, n° 7-8, 2001, pp 447-454.

[11] P. Adde, R. Pyndiah and F. Buda "Design and performance of a product code turbo encoding-decoding prototype" *Annals of telecommunications*, 54, n° 3-4, 1999, pp 214-219.

[12] S. Kerouédan, P. Adde and P. Ferry "Comparaison performance/complexité de décodeurs de codes BCH utilisés en turbo-décodage", *Gretsi'* 99, 13/17 Sept. 1999, pp. 172-175.