Implementation on FPGA of a high data rate turbo encoder-decoder of product code with a low complexity.

Thomas Q.K. TA, Pierre Leray, Annick Le Glaunec

Supélec - Rennes, ETSN team, Avenue de la Boulaie, BP 81127, 35511 Cesson-Sévigné Cedex, FRANCE.
Phone: (+33) 2 99 84 45 36, Fax: (+33) 2 99 84 45 99
Email : thomas.ta@supelec.fr, pierre.leray@supelec.fr, annick.leglaunec@supelec.fr

Abstract: *In this paper, we present an implementation on FPGA (Field Programmable Gate Array) of a turbo encoder-decoder of product code which is able to achieve high data rate (i.e. > 25 Mbit/s) and has a low complexity. For this implementation, we consider a product code BCH(32,26,4)⊗BCH(32,26,4). In order to reach a high data rate, we use the Von Neumann structure with block processing that has nevertheless a high complexity. We propose afterwards to apply a pipeline structure to reduce the complexity of the turbo decoder. The simulation in the C language and the synthesis with VHDL language shows that our implementation on a ALTERA APEXII can reach a data rate of 50 Mbit/s and have a complexity less than 4500 logic elements.*

Keywords: block turbo codes, iterative decoding, circuits and software, Von Neumann structure.

1. INTRODUCTION

The development of multimedia services, of the portable terminals and the associated needs in terms of network capacity and high data rate with a more and more spectrum cumbersome, requires to simplify the receivers complexity. On one hand, we have to limit the consumption of the terminals, i.e. raise their autonomy. On the other hand, we have to reduce their treatment time, i.e. raise their data rate. This constraint about complexity, even moderated by technology progress, is contradictory with the increase in data rate. At present, in digital systems, the channel coding and particularly the channel decoding are the costly operations in terms of the treatment time and complexity. At the moment, the most used Forward Error Corrector (FEC) codes are the convolutional codes, sometime associated with Reed-Solomon code. It is the case of the GSM (Global System for Mobile), of the DAB (Digital Audio Broadcasting), of the terrestrial and satellite DVB (Digital Video Broadcasting). In 1994, R. Pyndiah [1] proposed an innovative method for product code decoding, called Block Turbo Codes (BTC). For high code rate, the performance of BTC is very close to Shannon's limit [2] and then satisfy the constraint of high spectral efficiency.

In this paper, first, we recall the basic principles of coding for product codes and the low complexity encoder for product code BCH(32,26,4)⊗BCH(32,26,4). Then, we present the turbo decoder architecture according to Von Neumann structure with block processing. A method of optimization on the complexity and a memory organization for the block processing are proposed afterwards. Finally, we present the performance and the total complexity of TCB encoder-decoder.

2. PRODUCT CODES ENCODER

2.1 Concept of product codes

Product codes are serially concatenated codes [3] which were introduced by P. Elias in 1954 [4]. The concept of these codes is a simple and relatively efficient method to construct powerful codes (that is having large minimum Hamming distance d) using simpler linear block codes [3]. If we consider two systematic linear block codes $\mathbf{C}1$ with parameters (n_1,k_1,d_1) and $\mathbf{C}2$ with parameters (n_2,k_2,d_2) where n_i, k_i, and d_i (i = 1,2) stand for code word length, number of information bits and minimum Hamming distance respectively. The parameters of the resulting product code $\mathbf{P} = \mathbf{C}1 \otimes \mathbf{C}2$ are $n = n_1 \times n_2$, $k = k_1 \times k_2$ et $d = d_1 \times d_2$ and code rate $R = R_1 \times R_2$ where R_i is the code rate of the code $\mathbf{C}i$ (i = 1,2). If $\mathbf{C}1$ is a systematic linear code, then $(n_1 - k_1)$ last rows built by $\mathbf{C}1$ are code words of the code $\mathbf{C}2$ and can thus be decoded like the k_1 first rows. In the same way, $(n_2 - k_2)$ last columns are code words of code $\mathbf{C}1$. For an application in HIPERLAN/2 (High PERformance Radio Local Access Networks type 2) [10], we consider codes $\mathbf{C}1 = \mathbf{C}2$ = extended BCH(n_e=32,k=26,d_e=4) code. In this case, the product code $\mathbf{P}$ is the $n_e \times n_e$ matrix $[\mathbf{C}_e]$ (see figure 1) where :

- The binary information is represented by a sub-matrix [**M**] of k rows and k columns.
- the k rows of sub-matrix [**M**] are coded by the BCH(31,26,3) code,
- the n_2 columns are coded by the BCH(31,26,3) code,
- Last row and last column of the matrix $[\mathbf{C}_e]$ are respectively composed of the parity bits on the columns and of the parity bits on the rows.

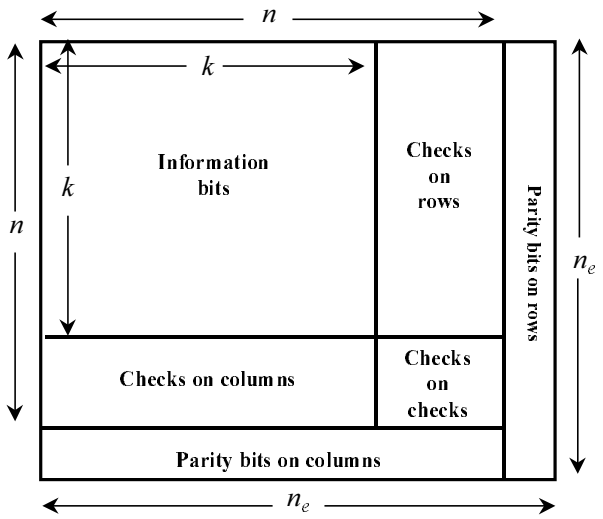

Figure 1 : Construction of product code P = C1⊗C2.

2.2. Encoder architecture

Let us consider that the transmission's order of the matrices [**M**] and [$\mathbf{C}_e$] bits is from left to right and from the top to the bottom. To respect this transmission's order and according to the construction of the product code BCH(32,26,4)⊗BCH(32,26,4) presented in the previous paragraph, we propose the encoder architecture illustrated in figure 2. Therefore, the product code encoder is composed of :

- a circuit that calculates the checks on rows,
- a circuit that computes the checks on columns,
- two parity calculation circuits (one for the rows and one for the columns),
- a multiplexer that selects the bits to transmit in transmission's order which is previously determined,
- a RAM that can store 31 words of 6 bits (checks and parity bits on columns),
- a sequencer circuit that control all the encoder's circuits.

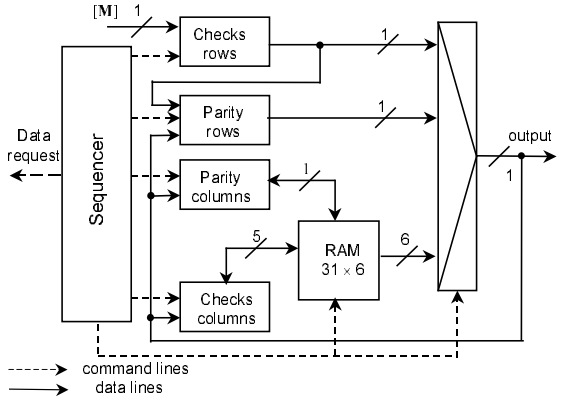

Figure 2 : block diagram of product code encoder.

The particularity of this encoder is its low complexity (140 Altera logic elements). We try to replace as far as possible all D flipflops by the RAM because the power consummation of one D flipflop is more important than one RAM bit [5]. Therefore this product code encoder has a low electrical power consummation and can reach a data rate of 62 Mbit/s.

3. TURBO DECODER FOR PRODUCT CODES

3.1. Decoding algorithm

The turbo decoding for product codes is an iterative decoding. It consists of cascaded SISO (Soft Input Soft Output) decoders for the rows and columns of the matrix [$\mathbf{C}_e$] [1]. The complete decoding of rows or columns corresponds to a half-iteration. The decoder uses the Chase-Pyndiah algorithm [6][1][2] and delivers the maximum likehook code word for a given input. It is shown that the product code decoder has a very good trade-off between performance and complexity when it uses a simplified Chase-Pyndiah (only one competitor) with four iterations [7][8]. Therefore, in our application, the whole product code decoder is composed of 8 half-iterations (four for the rows decoding and four for the columns decoding). Each half-iteration is illustrated in figure 3, where :

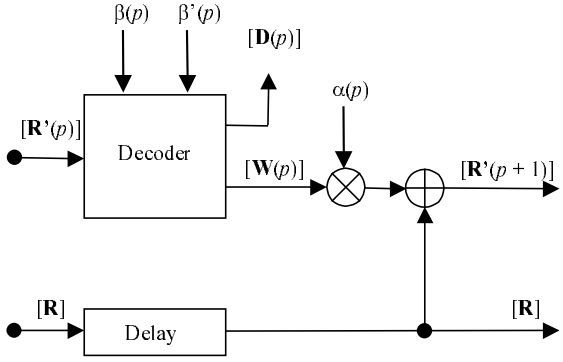

Figure 3 : block diagram of half iteration.

- [**R**] is the matrix initially received from the channel,
- [**R'**(p)] is the matrix received from the $(p-1)^{th}$ half-iteration,
- [**D**(p)] is the matrix that contains the result of binary decoding at the p^{th} half-iteration,
- [**W**(p)] is the p^{th} matrix of extrinsic information, which is the additional information given by the decoder concerning the reliability of the decoded bits
- [**R'**(p + 1)] is the matrix for the next half-iteration,
- $\alpha(p)$, $\beta(p)$, $\beta'(p)$ are three constants that depend on the p^{th} half-iteration; their values are optimized by simulations, with $\beta(p) < \beta'(p)$[7].

At the p^{th} half-iteration, the decoding of the i^{th} row (or column) $\mathbf{R'}_i(p)$ of the matrix $[\mathbf{R'}(p)]$ is decomposed as follows:

- selecting and storing the four less reliable binary symbols of $\mathbf{R'}_i(p)$. Their positions are called I_1, I_2, I_3, I_4.
- generating the test sequences $\mathbf{Y}^Q$ ($Q \in [1,7]$) by reversing one or two positions of the less reliable binary symbols in the $\mathbf{Y}^0$ which is a sign of $\mathbf{R'}_i(p)$,
- decoding $\mathbf{Y}^0$ and $\mathbf{Y}^Q$ with the hard decision decoding (syndrome decoding) and obtaining the $\mathbf{C}^k$ ($k \in [0,7]$) decoded code words,
- for each decoded code word $\mathbf{C}^k$, calculating the Euclidean distance between $\mathbf{C}^k$ and $\mathbf{R'}_i(p)$,
- selecting the code word $\mathbf{C}^d$ having a minimal Euclidean distance M^d with $\mathbf{R'}_i(p)$, then $\mathbf{D}_i(p) = \mathbf{C}^d$ is the result of binary decoding,
- computing the reliability for each element d_{ij} of $\mathbf{D}_i(p)$. For this, search for a second code word, called concurrent code word and noted $\mathbf{C}^c$, which has a minimal Euclidean distance M^c with $\mathbf{R'}_i(p)$. If a concurrent code word exists,

- if $d_{ij} \neq \mathbf{C}^c_j$ then $F_j = (M^c - M^d)$

- else $\begin{cases} F_j = \beta(p) \text{ if } M^d \neq 0 \\ F_j = \beta'(p) \text{ if } M^d = 0 \end{cases}$,

- and finally, computing extrinsic information

$$w_{ij} = F_j \, d_{ij} - r'_{ij}$$

3.2. Decoder architecture

Several structures can be adopted for hardware implementation of the decoder [8]. Between these structures, the Von Neumann structure with block processing is the most adapted for the application of high data rate thanks to its good compromise between performance and complexity.

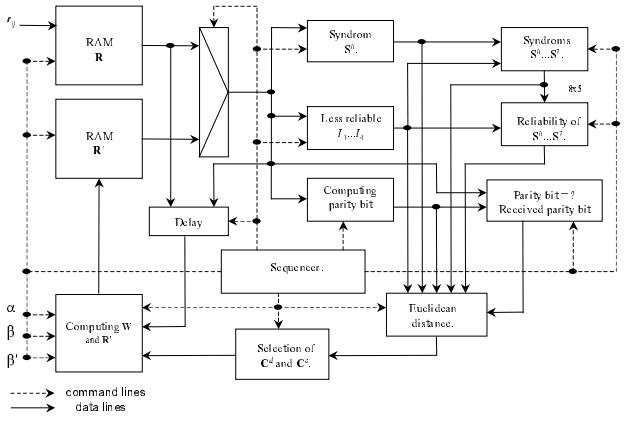

Figure 4 : block diagram of decoder according to the Von Neumann structure with block processing.

With block processing, only one elementary decoder (see figure 3) is indeed used for several iterations decoding. This reduces, on one hand, the decoder complexity and the decoding delay which is less than or equal to twice the time required to fill matrix [**R**] whatever the iterations number. On the other hand, this reduces the time required to decode one bit that raises the data rate. All stages of Chase-Pyndiah algorithm previously presented in sub-section 3.1 are described in VHDL as different blocks before to be gathered as shown in figure 4.

3.3. Memory block

To process and to memorize the matrices [**R**] and [**R**'], the Von Neumann structure with block processing requires a specific memory where the rows and columns of matrices [**R**] and [**R**'] can be read and written blockwise in one clock period. This double mode of row/column access can be realized by using n memory maps of standard RAM and by inserting a shift system on the inputs and outputs as illustrated in figure 5.

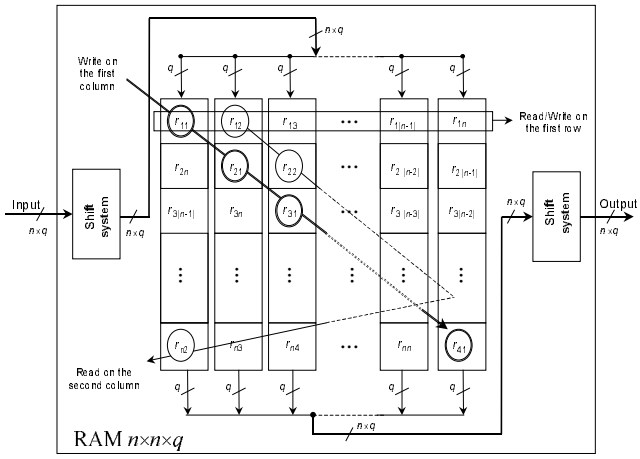

Figure 5 : Concept of memory for block processing.

In the figure 5, the ($n \times n \times q$) bits memory is composed of n memory maps. Each memory map is set as n words of q bits. In a such memory, the column or row reading is possible in one clock period if each memory map could give a word (of q bits) in one clock period. So the memory concept is based on the principle that the vector (column or row) words must not to be stored in the same memory map. To realize this, before writing, the data must be circularly shifted in one direction. And for getting back to the initial order, after reading, the data must be shifted in the other direction (except for the first row and for the first column). The complexity of the shift systems for the [**R**] and [**R**'] memories is very high and about equal to 4250 logic elements.

3.4. Decoder optimization

Block processing requires not only a specific memory but also raises the complexity of some decoder blocks. Indeed, the "less reliable $I_1...I_4$" (which selects the 4 less reliable binary symbols of $\mathbf{R}'_j$) and "computing **W** and **R'**" blocks are specially cumbersome (27% of total complexity of the decoder). Moreover, their treatment time is long and penalizes the decoder data rate. In order to solve these two problems, we propose to partition the decoder functioning and to use a pipeline structure to reduce the critical time. For the BCH(32,26,4)⊗BCH(32,26,4) code, as 4 clock periods are reserved for decoding one data vector, the 4 stage pipeline is used. Thanks to this partitioning, we proposed afterwards to apply the reiterated structure to the decoder blocks in order to reduce the decoder complexity without raising the critical time. For the implementation of BCH(32,26,4)⊗BCH(32,26,4) code with data quantizing on 4 bits, 8 test sequences, 1 competitor (simplified Chase-Pyndiah algorithm), and 4 decoding iterations, we obtain the performance and complexity of the encoder-decoder as presented in table 1.

	Maximum Data rate Mbit/s	Logic Elements	Memory RAM
Encoder	64	140	192
Decoder	50	4349	18032

Table 1 : performance and complexity of the BCH(32,26,4)⊗BCH(32,26,4) encoder-decoder .

With our solution, the complexity of block turbo decoder according to the Von Neumann with block processing decreases from 8804 down to 4178 logic elements and the data rate rises from 10 Mbit/s to 50 Mbit/s.

4. CONCLUSION

In this paper, we have shown that the implementation of the product code BCH(32,24,4)⊗BCH(32,24,4) decoder according to the Von Neumann structure with block processing can satisfy the constraints of high data rate (e.g. > 25 Mbit/s) and low complexity. This would not have been possible without applying the pipeline and the Von Neumann structure to the decoder blocks. Indeed, the block processing requires not only a specific memory but also increases the decoder complexity comparing to the sequential processing [8][9]. In this paper, we have also presented a low product code BCH(32,24,4)⊗BCH(32,24,4) encoder. We are implementing the circuit presented in this paper on the Altera FPGA from the APEXII family. The synthesis is performing on LeonardoSpectrum.

ACKNOWLEDGMENTS

The authors wish to present their gratitude to Mitsubishi Electric, Information Technology Europe - Telecommunications Laboratory (ITE-TCL) for their financial support to this work. Furthermore, we would like to thank Y. Louët and S. Tertois for their useful comments of this paper.

REFERENCES

[1] R. Pyndiah, A.Glavieux, A. Picart, S. Jacq, "Near optimum decoding of product codes", Globecom'94, San Fransisco, 1994.

[2] R. Pyndiah, , "Near optimum decoding of product codes : Block Turbo Codes", IEEE Trans. on Comm., vol. 46, No. 8, August 1998.

[3] F.J. Macwilliams and N.J.A. Sloane, "The theory of error correcting codes", North-Holland publishing company, pp. 567-580, 1978.

[4] P. Elias, "Error free coding", IRE Trans. On Inf. Theory, vol. IT-4, pp. 29-37, September 1954.

[5] A. Garcia, "Etude sur l'estimation et l'optimisation de la consommation de puissance des circuits logiques programmables du type FPGA", Thesis of l'ENST, 2000.

[6] D. CHASE, "A class of algorithms for decoding block codes with channel measurement information", IEEE Trans. Inform. Theory, vol. IT-18, pp. 170-182, no.1, January 1972.

[7] P. ADDE, R. PYNDIAH, "Recent simplifications and improvements in Block Turbo Codes", Proceedings of the 2nd International Symposium on Turbo Codes, pp. 133-136, France, 2000.

[8] O. RAOUL, "Conception et performances d'un circuit intégré turbo décodeur de codes produits", thesis of Université de Bretagne Occidentale, November 1997.

[9] S. Kerouédan, P. Adde, "Implementation of a block turbo decoder on a single chip", Proceedings of the 2nd International Symposium on Turbo Codes, pp. 243-246, France, 2000.

[10] N. Chapalain-Le Floc'h, "Application des Turbo Codes en Blocs pour les réseaux locaux sans fil à haut débit", thesis of Université de Bretagne Occidentale, 2002.