Efficient Architecture For Reed Solomon Block Turbo Code

Erwan PIRIOU, Christophe JEGO, Patrick ADDE, Raphael LE BIDAN, Michel JEZEQUEL
GET/ENST Bretagne, CNRS TAMCIC UMR 2872
Brest, France
firstname.lastname@enst-bretagne.fr

***Abstract*— Reed-Solomon codes are block-based error correcting codes with a wide range of applications in digital communications and storage. Recently, block turbo codes using Reed-Solomon component codes have been introduced. This was motivated by the highest code rate property of Reed-Solomon codes and their efficiency for burst error correction. In fact, the main advantage of Reed-Solomon block turbo codes is for high code rate applications. For these applications, the code length and consequently the decoder complexity are smaller than for usual Bose-Chaudhuri-Hocquenghem block turbo codes. This paper presents a block turbo decoder architecture using Reed-Solomon component codes. Our elementary soft input soft output decoder is dedicated to Reed-Solomon codes (31,29,3) with single error correction power. To the authors' knowledge, this is the first published architecture implementing this type of decoder. Experimentation has been done on a Stratix-based NIOS development board.**

I. INTRODUCTION

The use of error correction coding is one of the most powerful techniques available for the communication engineer to improve 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. The ability of forward error correction code to increase the signal to noise capability of a communication channel depends on the code chosen. Currently, the turbo code [1] family is considered to be the most efficient coding scheme for channel coding. This family of codes consists of two key design innovations: parallel concatenated encoding and iterative decoding. The iterative decoding is applied to Soft Input Soft Output (SISO) decoders. A SISO decoder both receives soft decision data and produces soft decision output. In this case, soft decision information is a probability or a quantized decimal value. The general concept of the iterative SISO decoding of concatenated convolutional codes has been extended to product codes [2] and Low Density Parity Check (LDPC) codes.

The turbo code family refers to two classes of codes: Convolutional Turbo Codes (CTC) and Block Turbo Codes (BTC). The general concept of BTC is based on product codes that were constructed by the serial concatenation of two (or more) systematic linear block codes. In the last few years, many block turbo decoder architectures have been designed [3,4]. Product codes using binary Bose-Chaudhuri-Hocquenghem (BCH) component codes have generally been chosen [5]. Currently, several companies such as TurboConcept in France provide industrial turbo product code decoders, which can handle BCH with single or double error correction power. The block turbo code concept was recently successfully extended to Reed-Solomon (RS) component codes [6]. In fact, RS codes are a widely used subclass of non-binary BCH codes. For this reason, the code construction approach dedicated to BCH codes can be naturally applied to RS codes. It is possible to design efficient BTC architectures using RS component codes from the BCH-BTC architecture know-how.

This paper presents an efficient architecture dedicated to block turbo code using RS component codes (31,29,3) with single error correcting power. It is structured in as follows: section 2 details the features of block turbo codes and some performance results. Then, in section 3, an original architecture for block turbo decoders is described. In fact, a novel implementation of the key equation solver, which helps to reduce the hardware complexity of the RS algebraic decoder, is shown. Finally, synthesis results of our design implementation are provided.

II. BLOCK TURBO CODES

The iterative soft input soft output decoding process can be used with product codes, first introduced by Elias [7] in 1954. Product codes are a series of concatenated systematic linear block codes. The product code inherits the properties of the elementary codes that it is composed of. RS codes are usually specified as RS (n,k,δ) with m_ary symbols. This means that encoder takes k data symbols and adds $(n-k)$ redundant symbols to generate an n symbol codeword. The minimum Hamming distance $\delta=n-k+1$ 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 $t=\lfloor (n-k)/2 \rfloor$ .The decoding of codes is based on Galois field arithmetic. So a (n,k,δ) code works in Galois field GF$(n-1)$ defined by its generator polynomial. Moreover, given a symbol size m, the maximum codeword length is $n = 2^m-1$.

A. *Iterative decoding process of product codes* $(n,k,\delta)^2$

Let us consider an example with two identical RS (n,k,δ) codes as illustrated in Figure 1. So, the parameters of the resulting product code are given by: n^2 (code length), k^2 (number of information symbols), δ^2 (minimum Hamming

distance) and R^2 (code rate). In 1972, Chase [8] proposed an algorithm that approximates the optimum ML decoding of block codes with low computing complexity and small performance degradation. In 1994, R. Pyndiah [2] presented a new iterative decoding algorithm for decoding product codes, based on the iterative SISO decoding of concatenated block codes. In fact, the iterative decoding algorithm involves of performing the successive decoding of rows and columns (two half iterations). Note that each iteration provides decoding operations for all rows and all columns of a matrix (product code). Moreover, a reconstruction of the matrix is necessary between each half iteration. For RS codes, the matrices have n^2 m_ary symbols ($m*n^2$ bits).

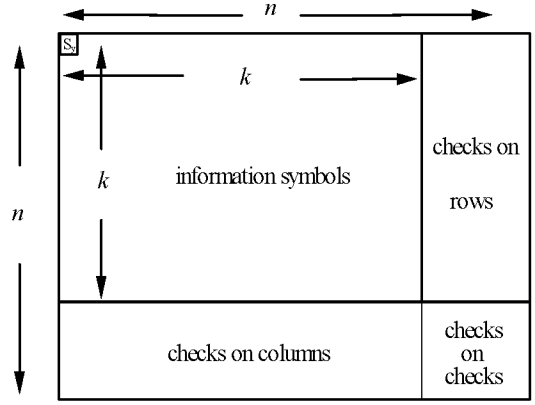

Figure 1. Principle of product codes

B. Elementary SISO decoder (n,k,δ)

RS codes are a special type of linear cyclic block codes. Actually, a code is linear if and only if the set of code words forms a vector subspace over a finite field like a Galois field. Moreover, in a cyclic code, for every code word $c=(c_0, c_1, \ldots, c_{n-2}, c_{n-1})$, $c'=(c_{n-1}, c_0, \ldots, c_{n-3}, c_{n-2})$ is also a code word. Thus all n shifts of c are also code words. A Galois field is a mathematical structure that contains a finite number of elements upon which addition and multiplication is defined. This means that the size of the Galois field (*i.e.* the number of elements in the field) uniquely determines the field.

The block diagram of an elementary SISO decoder is shown in Figure 2, where k stands for the number of half-iterations.

- R'_k is the word received from the previous half-iteration,
- R_k is the initial word received from the channel ($R'_k=R_k$ for the 1st half-iteration),
- W_k is the pattern that contains the extrinsic information, that is, the additional information given by the decoder concerning the reliability of the decoded symbol,
- D_k is the result of symbol decoding,
- α_k and β_k are constants that depend on the current half-iteration.

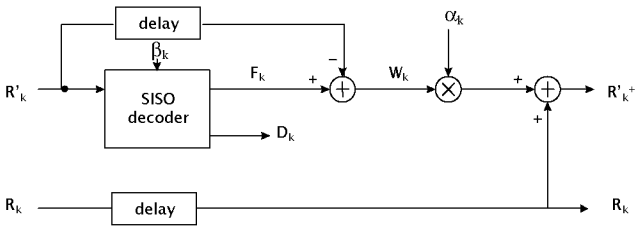

Figure 2. Block diagram of a half iteration elementary SISO decoder

C. Block turbo code performance

The reconfigurable elementary SISO decoder that we have chosen is dedicated to non-extended Reed-Solomon codes (31,29,3). In fact, the performance of the decoding depends on Hamming distance of the code. For linear cyclic block codes, it is possible to increase this distance by adding a parity symbol to the codeword and so the extended code parameters are $(n+1,k,\delta+1)$. In the case of BTC using Reed-Solomon component codes, extended codes are not efficient because the impact on code rate and consequently on the Shannon limit is not significant as shown in [5]. Thus, BTC using Reed-Solomon component codes are limited to non-extended codes. The main advantage of Reed-Solomon BTC using single correcting component codes is in high code rate applications. Indeed, for high code rates (0.8-0.9), the RS BTC require a much smaller code length than the corresponding BCH BTC. For example, table 5 shows that the RS BTC (31,29,3) enable the code length be reduced by 3.4. This feature is very interesting from the architecture point of view because it allows a reduction in memory complexity.

Block Turbo Code		code rate R	N (bits)	K (bits)
Reed-Solomon	$(31,29,3)^2$ t=1	0.875	4805	4205
BCH	$(128,120,4)^2$ t=1	0.878	16384	14400

Table 1.Comparison of BCH and RS block turbo codes

The performance of block turbo codes with single error correction power is given in Figure 3. Frame error rates were done on the AWGN channel using 16 test patterns and after 8 iterations. A comparison can be done from theoretical limit. In fact, the numerical theoretical performance limit can be quickly determined by using a tool available at the following address: *http://www-elec.enst-bretagne.fr/turbo/*. At a frame error rate of 10^{-4}, the distance between the code performance and the theoretical limit is lower than 1 dB for the two BTC: 0.75 dB for BCH$(128,120,4)^2$ and 0.8 dB for RS $(31,29,3)^2$.

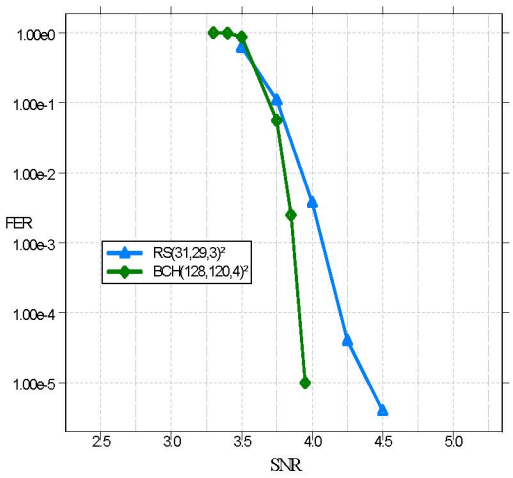

Figure 3. Performance of block turbo codes on the AWGN channel

III. Efficient Architecture Dedicated To Reed-Solomon Block Turbo Code

The SISO decoding algorithm is a modified Chase algorithm [8]. It searches for the p least reliable binary symbols of received m-ary word R'_k. Then τ test patterns, which are a combination of R'_k with inversions for the least reliable binary symbols, are built. Each test pattern is separately decoded with an algebraic decoding algorithm. The algebraic decoding that is used by each test pattern involves different major steps. First, the syndrome values are calculated from the symbols of the test pattern. These represent the sequence of errors in the frequency domain. Decoding algorithms need $2t$ syndromes for Reed-Solomon codes. Then the key equation is solved by determining the error locator polynomial $\sigma(x)$ from the syndrome values. The iterative techniques frequently used to solve the key equation include the Berlekamp-Massey algorithm or the Euclidian algorithm. But, for low error correction powers ($t<3$), the error locator polynomial coefficients can directly be calculated by the Peterson Gorenstein Zierler (PGZ) algorithm [9]. The next step is the computation of the error locator polynomial roots in order to provide the error locations in a test pattern. The most useful approach is the Chien search algorithm. But this iterative technique needs to take into account all the symbols of the test pattern for the root computation. Another approach that is based on a look up table can be used for low error correction powers ($t<3$). In fact, Berlekamp [10] has shown that a simple look up table of $m2^m$ bits with 2^m inputs is necessary to find the error locator polynomial roots for codes defined in Galois field GF(n-1). Finally, it is necessary to calculate the error magnitudes before the pattern test correction. A mathematical description of the different steps of the algebraic decoding can be found in [10].

A. Reed-Solomon SISO decoding architecture

This section details the features of an elementary soft input soft output decoder that is dedicated to single error correcting RS codes (31,29,3). Our elementary decoder consists of four blocks: reception, processing, transmission and control. In the reception part, the n=31 m-ary symbols of the received word are processed progressively. So, this block computes initial syndromes and positions of the p least reliable binary symbols. The number of syndromes depends on the error correction power (two for single error correction power). A construction of two parallel syndrome cells is used in our architecture.

The main function of the processing part is to build and then to correct the test patterns obtained from the initial syndrome and to combine the least reliable bits. The number of tests chosen in our architecture is sixteen because more patterns do not offer a significant performance gain for an increase in design complexity. Decoding the test patterns is done by an algebraic decoder that was previously detailed. The architecture that carries out the algebraic decoder is described in Figure 4. It can decode RS codes with single error correction power. The PGZ algorithm estimates the error position. Error value is computed in parallel because it only depends on the two syndromes. Finally, the algebraic decoder verifies if the corrected test pattern is a codeword. Moreover, the processing block has to produce a metric (Euclidean distance between test pattern and received word) for each test pattern. Finally, a selection function allows the maximum likelihood code word w_d and three concurrent words w_c to be selected.

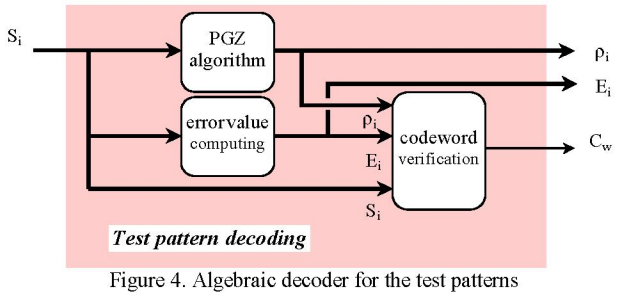

Figure 4. Algebraic decoder for the test patterns

In the transmission part, the block performs different functions: computing the reliability for each binary symbol, computing the extrinsic information and correction of the received symbols. The n=31 m-ary symbols of the codeword are corrected progressively.

The three previous blocks are supervised by a control part. In our design, this task is achieved by a Nios embedded processor. This processor enables the data communication through an Avalon system bus between the matrix memory and the elementary decoder. Moreover, the processor sequences the iterative decoding process of all the matrix codewords.

The RS (31,29,3) architecture uses 3 pipeline blocks for decoding a word of 31 m-ary (m=5) symbols. The three blocks of the decoder have the same latency. 10 (2m) clock periods are necessary for the algebraic decoding of each one of the 16 test patterns. The latency of the elementary decoder is 320 (64m) clock periods which corresponds to two words.

B. *Block turbo decoder prototype*

A system level design flow using SystemC language was applied to validate the elementary decoder description and to simultaneously involve hardware and software design elements [11]. The SystemC description of the elementary decoder functional blocks was synthesized with SystemC Compiler tool from Synopsys and then integrated into a Stratix EP1S40 FPGA device by the Altera Quartus® II design software. The RS block turbo decoder features are: 31 codeword length, 8 iterations and single error correction powers. In fact, the same elementary decoder is used for the sixteen half-iterations of the decoding process. The reconstruction of the 31 x 31 m_ary symbols matrix between each decoding is carried out by the Nios processor that assigned four memory blocks of 24,025 bits. Table 2 gives complexity and characteristics of the different blocks of the elementary RS decoder (31,29,3). 1998 and 4563 logic elements are necessary for the integration of the three functional blocks and the control unit respectively. A complexity comparison with a block turbo decoder architecture using BCH (128,120,4) can be done. The number of logic elements increases by 13 % for the elementary RS decoder (31,29,3). On the other hand, the number of memory bits decreases by 67 % for the elementary RS decoder (31,29,3). This result is very interesting from the architecture point of view because it allows a considerable reduction in memory complexity.

FPGA device: Stratix EP1S40F780C5	block turbo decoder functions		number of logic elements
reception block	syndrome		49
	least reliable bits		288
processing block	test pattern processing		640
	test pattern decoding	PGZ algorithm	88
		error value	
		codeword	41
	w_d and w_c selection		430
transmission block	reliability		372
	extrinsic information		62
	codeword correction		28
control unit	Nios processor: 32-bits data paths and 16-bits instruction set		4563
decoder complexity	6561 / 41,250 (16 %)		

FPGA device: Stratix EP1S40F780C5	block turbo decoder functions	number of memory bits
memory unit	Galois field ROM	32 x 5 x 2
	symbols-codeword RAM	6 x 31 x 5 x 5
control unit	boot germs monitor	2,000 x 8
	block turbo decoder RAM	4 x 31 x 31x 5x 5
	Nios processor CPU	123,136
decoder complexity	240,206 / 3,423,744 (7 %)	

Table 2: Complexity of the block turbo decoder (31,29,3)[2]

The clock frequency of the Nios processor is 100 MHz but the data system bus transfer throughput between the processor and the hardware blocks is 25 Mbps. However, specific applications require high-speed decoders. Innovative solutions have been proposed for increasing the block turbo decoding speed [4]. The idea is to use several elementary decoders in parallel that correct different rows or columns independently. An appropriate data organization of the memory allows simultaneous reading of the required symbols by all the decoders. This approach can be applied in our architecture. The RS elementary decoder is duplicated and a Nios processor is used to organize the symbol communication between the memory and the decoders. In this case, several elementary decoders can be integrated into a Stratix EP1S40 device. Moreover, high-density FPGA devices can be used for high-speed block turbo decoding.

CONCLUSION

A block turbo decoder architecture using Reed-Solomon (31,29,3) component codes with a single error correction power has been presented in this paper. This architecture consists of two design innovations: novel implementation of the key equation solver for the algebraic decoding of Reed-Solomon codes and iterative soft input soft output decoding of Reed-Solomon product code. To the authors' knowledge, this is the first published architecture implementing a Reed-Solomon block turbo decoder. This family of error correction coding is a very promising solution especially for high code rate applications. In fact, many potential applications such as optical transmission, data storage and mobile communications, can benefit from this new class of block turbo code. Moreover, architectural solutions are currently investigated for increasing the throughput of our block turbo decoder. The objective is to design parallel turbo decoding of product code for gigabit per second applications.

REFERENCES

[1] C. Berrou, A. Glavieux, P. Thitimajshima, *Near Shannon limit error correcting coding and decoding: Turbo Codes*, IEEE International Conference on Communication ICC93, vol. 2/3, May 1993.

[2] R. Pyndiah, A. Glavieux, A. Picart, S. Jacq, *Near optimum decoding of product codes*, GLOBECOM94, November 1994.

[3] S. Kerouedan, P. Adde, R. Pyndiah, *How we implemented Block Turbo Codes*, Annals of Telecommunication, Vol. 56, N° 7-8, pp. 447-454, July-August 2001.

[4] J. Cuevas, P. Adde, S. Kerouédan, R. Pyndiah, *New architecture for high data rate turbo decoding of product codes,* GLOBECOM 2002, pp. 139-143, November 2002.

[5] R. Pyndiah, P. Adde, R. Zhou, *Block Turbo Codes: Ten years later*, IEE Seminar on Sparse Graph Codes, October 2004.

[6] R. Zhou, A. Picart, R. Pyndiah, A. Goalic, *Reliable transmission with low complexity Reed Solomon block turbo codes,* ISWCS Conference, pp 193-197, September 2004.

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

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

[9] W. W. Peterson, *Encoding and error correcting procedures for the Bose-Chaudhuri codes*, IRE Transf. Theory, vol IT-6, pp. 459-470, September 1960.

[10] E. R. Berlekamp, *Algebraic Coding Theorie,* revised edition, Aegean Park Press, 1984.

[11] E. Piriou C. Jégo, P. Adde, M. Jézéquel, *System level design using SystemC: a case study of block turbo decoder*, XIX Conference on DCIS2004, November 2004.