A 5-Gbps FPGA prototype of a (31,29)2 Reed-Solomon Turbo Decoder

Camille Leroux, Gérald Le Mestre, Christophe Jego, Patrick Adde, Michel Jezequel

A 5-Gbps FPGA prototype of a $(31,29)^2$ Reed-Solomon Turbo Decoder

Camille Leroux, Gérald Le Mestre, Christophe Jégo, Patrick Adde and Michel Jézéquel
Institut TELECOM, TELECOM Bretagne, CNRS Lab-STICC
Electronics department, 29238 Brest
Email: firstname.surname@telecom-bretagne.eu

***Abstract*— In this paper, the use of single-error-correcting Reed-Solomon (RS) product codes are investigated in an ultra high-speed context. A full-parallel architecture dedicated to the turbo decoding process of RS product codes is described. An experimental setup composed of a Dinigroup board that includes six Xilinx Virtex-5 LX330 FPGAs is employed. Thus, a full-parallel turbo decoding architecture dedicated to the $(31, 29)^2$ RS product code has been designed and then implemented into a 5Gbps experimental setup. The purpose of this prototype is to demonstrate that RS turbo decoders can effectively achieve information rates above 1Gbps. The results show that the RS turbo product codes offer a good complexity/performance trade off for ultra-high throughputs. The major limitation in terms of data rate of our prototype is the data exchange between the FPGAs of the board. Indeed, the turbo decoder architecture enables decoding at information rates until 10Gbps onto FPGA devices.**

I. Introduction

In telecommunications, forward error correction (FEC) is a system of error control that improves digital communication quality. Since the early 90's, channel coding families based on iterative decoding process are considered to be the most efficient coding schemes. With the invention of turbo codes [1] followed by the rediscovery of low-density parity-check (LDPC) codes [2], it is now possible to approach the fundamental limit of channel capacity. The turbo code family refers to two classes of codes: Convolutional Turbo Codes (CTCs) [1] and Turbo Product Codes (TPCs) [3]. The general concept of TPC 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 TPC decoder architectures have been designed [4, 5]. Product codes using binary Bose-Chaudhuri-Hocquenghem (BCH) component codes have generally been chosen [6]. The turbo product code concept was recently successfully extended to RS component codes [7, 8]. Actually, 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 TPC architectures using RS component codes from the BCH- TPC architecture know-how [9].

At present, the main challenge for hardware implementation of turbo decoder is to meet the demand for even higher data rates. Indeed, the increasing demand of high data rate and reliability in modern communication systems is pushing next-generation standards toward error correction schemes allowing high throughput decoding with near Shannon limit performance. Thus, data rates above 1Gbps and 10Gbps will be required for the next generation standards of wireless technologies (WIFI, WIMAX, DVB-H…) and data transmission over Passive Optical Networks (PON), respectively. In order to achieve high throughput and to reduce latency, parallel implementations become mandatory. In 2006, a full-parallel architecture for the turbo decoding of product codes was proposed in [10]. The major advantage of this full-parallel architecture is that it enables the memory block between each half-iteration to be replaced by simple interconnection networks. Moreover, the latency of the turbo decoder is strongly reduced. In such a full-parallel architecture, most of the hardware complexity remains in the duplicated Soft Input Soft Output (SISO) decoders.

In this paper, the implementation onto FPGA devices of an ultra high throughput $(31,29)^2$ RS turbo decoder is presented. The purpose of this prototype is to demonstrate that an RS turbo decoder can effectively achieve data rates above 1Gbps. To the authors' knowledge, this is the first implementation of FEC decoder onto an FPGA target at information throughputs superior to 1Gbps. Previously, a BTC decoder included in a 12.4Gbps optical setup was described in [11]. Since only a part of the transmitted data is actually coded, the information throughput of the decoder is 156Mb/s. In 2007, a LDPC code decoder ASIC implementation at an information throughput equal to 2.4Gbps was proposed in [12].

The remainder of the paper is organized as follows. Section II briefly recalls the basic principles of turbo decoding for RS product codes. The challenging issue of designing high data rates full parallel turbo decoding architecture is developed in Section III. Section IV describes an implementation of the resulting turbo decoder and its experimental setup onto FPGA devices. Finally, characteristics and performance in terms of BER of the prototype are detailed in Section V.

II. Turbo decoding of RS product codes

A product code is a concatenation of systematic linear block codes. The product code inherits the properties of the elementary codes that it is composed of. Let us consider two identical systematic linear block codes C having parameters (n, k, d_{min}), where n, k and d_{min} stand for code length, number of information symbols and minimum Hamming distance, respectively. The parameters of the resulting product code are given by: $n_p = n^2$, $k_p = k^2$, $d_{min\ p} = d_{min}{}^2$ and $R_p = R^2$ (code rate). Thus, it is possible to construct powerful product codes using linear block codes. When C is an RS code over Galois field GF(2^m), we obtain an RS product code over GF(2^m). Digital communication systems usually employ some form of binary signaling. A binary expansion of the RS product code is then required for transmission. The extension field GF(2^m) forms a vector space of dimension m over GF(2). A binary image of the

This work was supported with funds from the Brittany Region under the competitiveness pole InterAccess

RS product code is thus obtained by expanding each code symbol of the product code matrix into m bits using a polynomial basis. The binary image is a product code with length mn^2, dimension mk^2 and minimum distance at least as large as the symbol level minimum distance.

Turbo decoding process involves sequentially decoding the rows and columns of the product code matrix and by exchanging soft information between the decoders until a reliable decision can be made on the transmitted bits. In this work, SISO decoding of the RS component codes is performed at the bit-level using the Chase-Pyndiah algorithm. First introduced in [3] for binary BCH codes and latter extended to RS codes in [13], the Chase-Pyndiah decoder consists of a soft-input hard-output Chase-2 decoder [14] augmented by a soft-output computation unit.

III. Full-Parallel RS turbo decoding architecture

With current technologies, parallel decoding architectures are the only solution to achieve information rates superior to 1Gbps. An easy architectural solution is to duplicate the elementary decoders and the corresponding interleaving memories in order to achieve the given throughput. This results in a turbo decoder with unacceptable cumulative area. Thus, smarter parallel decoding architectures have to be designed in order to better trade off performance and complexity under the constraint of a high-throughput.

A. Previous work

Many turbo decoder architectures for product codes have been previously designed. The classical approach involves decoding all the rows or all the columns of a matrix before the next half-iteration. When an application requires high speed decoders, an architectural solution is to cascade SISO elementary decoders for each half-iteration. In this case, memory blocks are necessary between each half-iteration to store channel data and extrinsic information. Each memory block is composed of four memories of mN^2 soft values. Thus, duplicating a SISO elementary decoder results in duplicating also the memory block which is very costly in terms of silicon area. In 2002, a new architecture for turbo decoding product codes was proposed in [5]. The idea is to store several data at the same address and to perform semi-parallel decoding to increase the data rate. However, it is necessary to process these data by row and by column. Let us consider l adjacent rows and l adjacent columns of the initial matrix. The l^2 data constitute a word of the new matrix that has l^2 times fewer addresses. This data organization does not require any particular memory architecture. The results obtained show that the turbo decoding throughput is increased by l^2 when l elementary decoders processing l data simultaneously are used. Turbo decoding latency is divided by l. The area of the l elementary decoders is increased by $l^2/2$ while the memory is kept constant.

B. Full-parallel decoding principle

All rows (or all columns) of a matrix can be decoded in parallel. If the architecture is composed of $2N$ elementary decoders, an appropriate treatment of the matrix allows the elimination of the reconstruction of the matrix between each decoding. Specifically, let i and j be the indices of a row and a column of the N^2 matrix. In full-parallel processing, the row decoder i begins the decoding by the soft value in the i^{th} position. Moreover, each row decoder processes the soft values by increasing the index by one modulo N. Similarly, the column decoder j begins the decoding by the soft value in the j^{th} position. In addition, each column decoder processes the soft values by decreasing the index by one modulo N. Actually, the full-parallel decoding of turbo product code is possible thanks to the cyclic property of RS codes. Indeed, every cyclic shift $\mathbf{c'} = (c_{N-1}, c_0, .., c_{N-3}, c_{N-2})$ of a codeword $\mathbf{c} = (c_0, c_1, .., c_{N-2}, c_{N-1})$ is also a valid codeword in a cyclic code. Therefore, only one clock period is necessary between two successive matrix decoding operations. The full-parallel decoding of a product code matrix N^2 is detailed in Figure 1. A similar strategy was previously presented in [15] where memory access conflicts are resolved by means of an appropriate treatment of the matrix. However, interleaving memory is still required.

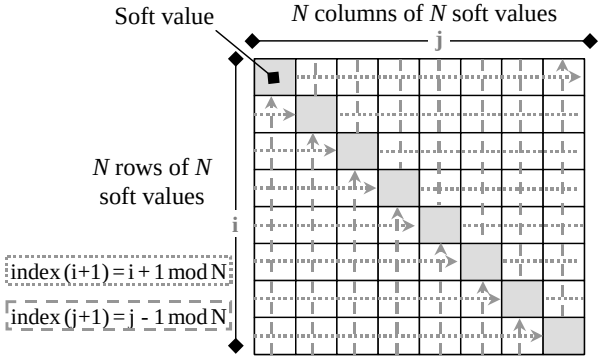

Fig. 1. Full-parallel decoding of an RS product code matrix.

The elementary decoder latency depends on the structure of the decoder (*i.e.* number of pipeline stages) and the code length N. Here, as the reconstruction matrix is removed, the latency between row and column decoding is null.

C. Full-parallel architecture for RS product codes

The major advantage of our full-parallel architecture is that it enables the memory block of $4mN^2$ soft values between each half-iteration to be removed. However, the codeword soft values exchanged between the row and column decoders have to be routed. One solution is to use an interconnection network for this task. In our case, we have chosen an Omega network. The Omega network is one of several interconnection networks used in parallel machines [16]. It is composed of $log_2 N$ stages, each having N=2 exchange elements. In fact, the Omega network complexity in terms of number of interconnections and of 2x2 switch transfer blocks is N x $log_2 N$ and $N/2$ x $log_2 N$, respectively. For example, the equivalent gate complexity of a 31x31 network can be estimated to be 200 logic gates per exchange bit. Figure 2 depicts a full-parallel architecture for the turbo decoding of product codes. It is composed of cascaded modules for the turbo decoder. Each module is dedicated to one iteration. However, it is possible to process several iterations by the same module. In our approach, $2N$ elementary decoders and two interconnection blocks are necessary for one module. An interconnection block is composed of two Omega networks exchanging the $\mathbf{R}$ and $\mathbf{R}_{iter}$ soft values. Since the Omega network has low complexity, the full-parallel turbo decoder

complexity essentially depends on the complexity of the elementary SISO decoder.

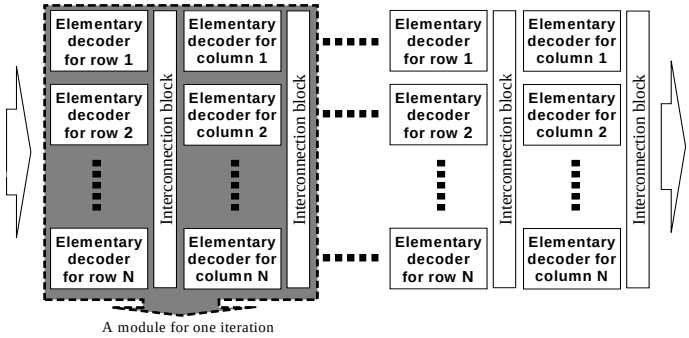

Fig. 2. Full-parallel architecture for decoding of RS product codes.

D. Elementary SISO decoder architecture

The block diagram of an elementary SISO decoder is shown in Figure 3, where *iter* stands for the current half-iteration number. $\mathbf{R}_{iter}$ is the soft-input matrix computed from the previous half-iteration whereas **R** denotes the initial matrix delivered by the receiver front-end ($\mathbf{R}_{iter}=\mathbf{R}$ for the 1st half-iteration). $\mathbf{W}_{iter}$ is the extrinsic information matrix. α_{iter} is a scaling factor that depends on the current half-iteration and is used to weight the influence of the extrinsic information during the first iterations.

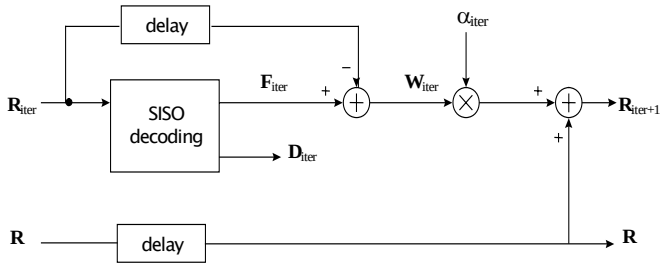

Fig. 3. Block diagram of the turbo-decoder at the *iter*-th half-iteration

The decoder architecture is structured in three pipelined stages identified as reception, processing and transmission units [5]. During each stage, the N soft values of the received word $\mathbf{R}_{iter}$ are processed sequentially in N clock periods. The reception stage computes the initial syndromes S_i and finds the L_r least reliable bits in the received word. The main function of the processing stage is to build and then to correct the N_{ep} error patterns obtained from the initial syndrome and to combine the least reliable bits. Moreover, the processing stage also has to produce a metric (Euclidean distance between error pattern and received word) for each error pattern. Finally, a selection function identifies the maximum likelihood codeword **d** and the competing codewords **c** (if any). The transmission stage performs different functions: computing the reliability for each binary soft value, computing the extrinsic information and updating the received soft values. The N soft values of the codeword are thus corrected sequentially. The decoding process needs to access the **R** and $\mathbf{R}_{iter}$ soft values during the three decoding phases. For this reason, these words are implemented in six Random Access Memories (RAM) of size qmN controlled by a finite state machine.

IV. Implementation of an RS turbo decoder in an ultra high rate communications setup

The purpose of this prototype is to demonstrate that an RS turbo decoder can effectively achieve information rates until 5Gbps.

A. 5Gbps experimental setup

The experimental setup is composed of a Dinigroup board [17] that includes six Xilinx Virtex-5 LX330 FPGAs. A Xilinx Virtex-5 LX330 FPGA contains 51,840 slices that can emulate up to 12 million gates of logic. It should be noted that Virtex-5 slices are organized differently from previous generations. Each Virtex-5 slice contains four Look Up Tables (LUTs) and four flip-flops instead of two LUTs and two flip-flops in previous generation devices. The board is hosted on a 64- bit, 66MHz PCI bus that enables communication at full PCI bandwidth with a computer.

Figure 4 shows the different components of the digital communication setup implemented onto six FPGAs. One FPGA is dedicated to the component implementation of the transmission part. The five other FPGAs are dedicated to the implementation of the RS turbo decoder. One decoding iteration was implemented onto each FPGA resulting in a 5 full-iteration turbo decoder. Each decoding module corresponds to a full-parallel architecture dedicated to the decoding of a matrix of 31 x 31 coded soft values. We recall here that a coded soft values over GF(32) is mapped onto 5 LLR values, each LLR being quantized on 5 bits. The decoding process needs to access the 31 coded soft values from each of the matrices **R** and $\mathbf{R}_{iter}$ during the three decoding phases of a half-iteration as explained in section III. For these reasons, 31 x 5 x 5 x 2 = 1,550 bits have to be exchanged between the decoding modules during each clock period f_0=37.5MHz. The Dinigroup board offers 200 chip to chip LVDS for each FPGA to FPGA interconnect. Unfortunately, this number of LVDS is insufficient to enable the transmission of all the bits between the decoding modules. To solve this implementation constraint, we have chosen to add SERializer/DESerializer (SERDES) modules for the parallel-to-serial conversions and for the serial-to-parallel conversions in each FPGA. Indeed, SERDES is a pair of functional blocks commonly used in high speed communications to convert data between parallel data and serial interfaces in each direction. SERDES modules are clocked with $f_1 = 8f_0$ = 300MHz and operate at 8:1 serialization or 1:8 deserialization. In this way, all data can be exchanged between the different decoding modules. Thanks to 200 chip to chip LVDS and the SERDES modules, data are exchanged between the different FPGAs at a throughput of 58,125Gbps while the working frequency and the information rates are only 37.5MHz and 5Gbps, respectively.

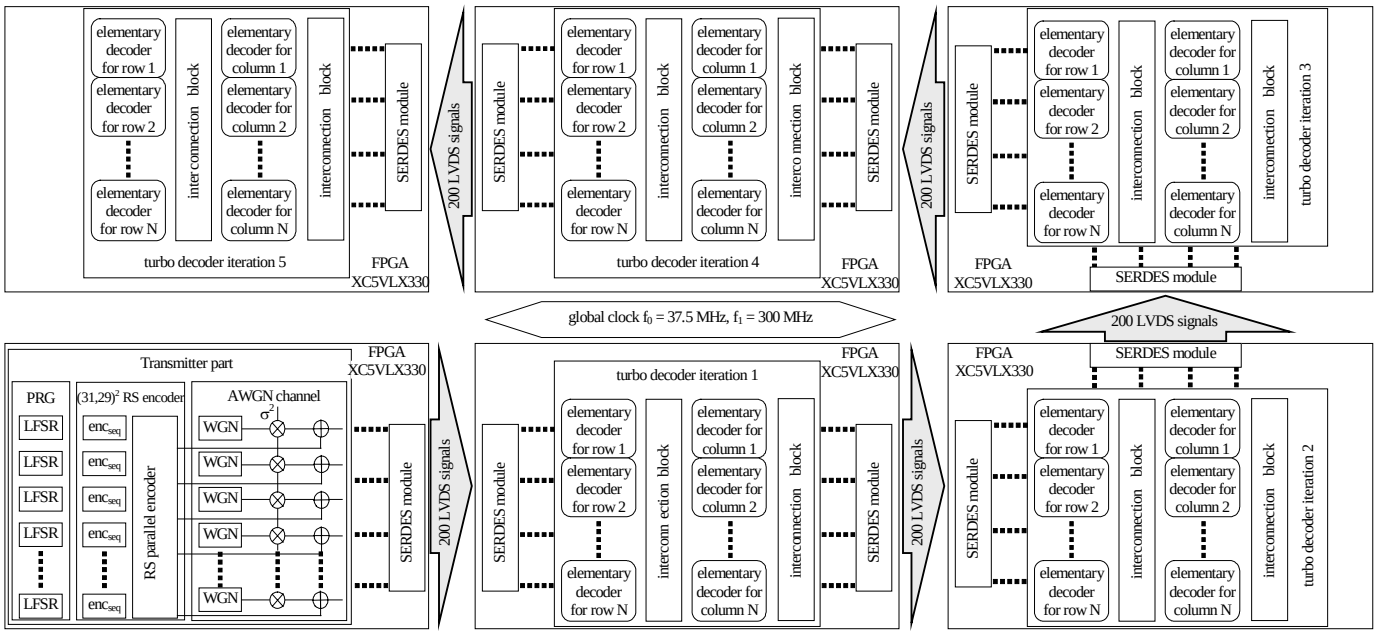

Fig. 4. 5Gbps experimental setup for turbo decoding of a $(31,29)^2$ RS product code

A. *The transmission part of the prototype*

A Pseudo Random Generator (PRG) sends out 145 (29*5) pseudo random data streams at each clock period (f_0). This module is composed of flip-flops and XOR gates that generate 145 parallel outputs. A $(31,29)^2$ RS encoder processes the 145 data streams in parallel. The original proposed encoding scheme uses both sequential and parallel encoders. *k* sequential encoders process *k* rows and a parallel encoder processes *n* columns. Sequential encoders are conventional implemented by a set of linear feedback shift registers (LFSR). Alternatively, a linear block code can be encoded by multiplying the information vector *k* by the generator matrix. This can be easily implemented by an XOR gate arborescence which is a parallel combinatorial encoder. The classical sequential approach requires a *kn* memory block between row and column encoding. The original proposed encoding scheme avoids the use of memory between row and column processing. Moreover, the depicted architecture decreases the latency from *kn* to only one symbol. 29 conventional sequential encoders are cascaded with one parallel encoder. 4805 encoded data (equivalent to a matrix 31 x 31 of 5-bit symbols) are generated in 31 clock periods (f_0). The noise generator models 155 uncorrelated White Gaussian Noise (WGN) samples and adds them to the previously encoded data. We have chosen to integrate an AWGN channel from a white Gaussian noise generator adapted to hardware implementation. High accuracy, speed and low-cost hardware are reached by combining the Box-Muller and Central limit methods [18]. Each output sample is a 5 bit vector resulting in 775 (155*5) bits to be sent in one clock period (f_0). The Signal to Noise Ratio (SNR) is controllable via a computer that calculates the equivalent variance square root σ and sends it to the transmission part thanks to a PCI bus.

B. *The transmission part of the prototype*

One decoding iteration was implemented onto one FPGA resulting in a prototype that contains 5 full-iteration turbo decoders as previously explained. Full-parallel decoder architecture has been detailed in section III. It was observed that most of the complexity remains in the duplicated SISO decoders. A complexity analysis detailed in [19] allowed us to select a configuration to be implemented on the SISO decoder. All the soft information within the decoder is quantized and processed with q = 5 bits (1 sign bit and 4 reliability bits). The SISO decoder architecture is structured in three pipelined stages identified as reception, processing and transmission units as described in subsection III.D. Each stage processes N = 31 symbols in N = 31 clock periods (f_0). In our architecture, the number of least reliable bits in a received word $\mathbf{R}_{iter}$ is equal to three (L_r = 3). The processing unit computes the syndrome of the N_{ep}=8 error patterns and their metric values. It also sequentially selects the decided word **d** and one competing codeword **c**. Finally, the transmission unit calculates extrinsic information $\mathbf{W}_{iter}$ and soft output $\mathbf{R}_{iter+1}$. The multi-stage dynamically reconfigurable interconnection network omega is based on a circular permutation principle. The connecting scheme consists of circularly shifted input data. One decoding iteration need $2N$ = 62 SISO decoders and 2 interconnection omega networks. Between two half-iterations, two omega networks are required to transmit **R** and $\mathbf{R}_{iter}$ values. However, the communication resources represent less than 1 % of the total complexity of one decoding iteration. A PRG module is also implemented onto each FPGA of the reception part. It generates the exact same data as in the transmitter in order to compare data before and after each decoding iteration. A BER block is used to measure the error rate comparing data from the PRG and the hard full-parallel decoder output $\mathbf{D}_{iter}$. The number of errors and the

total number of decoded data are then transmitted to the computer for a SNR value thanks to the PCI bus.

V. Characteristics and performance of the $(31,29)^2$ RS Turbo Decoder

A. Characteristics of the 5Gbps experimental setup

The total occupation rate of the FPGA that contains the transmission part is 79%. This corresponds to 164,953 Virtex-5 slice LUT-Flip Flop pairs. In addition, DSP resources for the transmission part take up 155 out of 192 (80%). No BlockRAM of 18Kbs are assigned. The PRG module needs only 1,900 slice Flip-Flops and 145 slice LUTs. The $(31,29)^2$ RS encoder module occupies 10,728 slice Flip-Flops and 4,471 slice LUTs. It represents only 5 % and 2% in terms of slice Flip-Flops and slice LUTs of the total design complexity, respectively. It means that the most important module of the transmission part in terms of complexity is the AWGN channel hardware emulator. This is due to the fact that the AWGN channel is composed of 155 uncorrelated White Gaussian Noise (WGN) elements. Thus, computational resources of the channel module take up 57,741 slice Flip-Flops and 148,560 slice LUTs. The total occupation rate are about 28% and 72% of a Xilinx Virtex-5 LX330 FPGA for slice Flip-Flops and slice LUTs, respectively. Moreover, 155 DSP resources are necessary for the 155 multiplication operations between each WGN sample and the variance square root value σ given by the computer. The rest of the FPGA is occupied by one SERDES module.

One decoding iteration was implemented on each of the five other FPGAs. These FPGAs are occupied by the same design. The total occupation rate of each FPGA that contains one decoding iteration module, two SERDES modules, one PRG module and one error estimation module is 46%. This corresponds to 97,383 Virtex-5 slice LUT-Flip Flop pairs. Note that the decoding module represents only 50 % in terms of slices of the total design complexity. Actually, a decoding module for one iteration is composed of 31 x 2 = 62 elementary decoders and 2 interconnection blocks. Each elementary decoder uses information quantized on 5 bits with N_{ep} = 8 error patterns and only 1 competing codeword. These reduced parameter values allow a decrease in the required area for a performance degradation which remains around 0.5dB as we will show in the next subsection. Thus a (31, 29) RS elementary decoder occupies 729 slice LUTs, 472 slice Flip-Flops and 3 BlockRAM of 18Kbs. An interconnection block occupies only 2,325 slice LUTs. Computation resources of one decoding iteration module take up 29,295 slice Flip-Flops and 49,848 slice LUTs. It means that the occupation rates are about 14% and 24% of a Xilinx Virtex-5 LX330 FPGA for slice registers and slice LUTs, respectively. In addition, memory resources for the decoding module take up 186 BlockRAM of 18kbits. It represents 32% of the total BlockRAM available in the Xilinx Virtex-5 LX330 FPGA. Note that one BlockRAM of 18kbits is allocated by the Xilinx tool ISE to memorize only 31x5x5 = 775 bits in our design. The occupation rate of each BlockRAM of 18kbits is then only about 4%. As input data are clocked with f_0=37.5 MHz resulting in an input data rate of T_{in}=5.715Gbps. However, taking into account the code rate R=0.875, the output information rate becomes T_{out}=5Gbps. In conclusion, the implementation results showed that a turbo decoder dedicated to the $(31, 29)^2$ RS product code can effectively be integrated to our 5Gbps experimental setup.

The prototyping board used for our experimental setup is the major limitation in terms of information rate. Indeed, the board offers only 200 chip to chip LVDS for each FPGA to FPGA interconnect. But, it is insufficient to exchange 1,550 bits between the FPGAs in one clock period. To solve this constraint, we have implemented SERDES modules that operate at 8:1 serialization or 1:8 deserialization. This configuration requires to clock the SERDES with a frequency $f_1=8f_0$. It means that we have to take the ratio between f_0 and f_1 into account to find the frequencies. Values equal to 37.5MHz and 300MHz have been finally selected for f_0 and f_1, respectively. However, one parallel decoding iteration decoding can be worked until a frequency f_0 equal to 85MHz onto a Xilinx Virtex-5 LX330 FPGA. As input data are clocked with f_0 = 85MHz, an information rate of 10Gbps is obtained. In particular, it means that a turbo decoder dedicated to the $(31,29)^2$ RS product code can effectively be integrated to the physical layer of a 10Gbps optical access network.

B. $(31,29)^2$ RS turbo product code performance

In [19], a complexity analysis of the BCH SISO elementary decoder leads to a low complexity decoder architecture for an acceptable performance degradation. As the architecture of an RS SISO elementary decoder is similar, an equivalent complexity analysis was done. Table 1 gives the characteristics of the prototype dedicated to the turbo decoding of a $(31,29)^2$ RS product code. The characteristics of an optimal Chase-Pyndiah algorithm in terms of BER performance are also given for comparison. By analysis the algorithm, five parameters appear to directly affect the complexity. The objective is to reach a better complexity/performance trade off in an ultra high-throughput context.

	Optimal algorithm features	Prototype features
arithmetic representation	floating-point	5bits fixed-point
number of iterations	iter = 8	iter = 5
number of error patterns	N_{ep} = 16	N_{ep} = 8
number of least reliable bits	L_r = 4	L_r = 3
number of competing codewords	**c** = 16	**c** = 1
value of the scaling factor α_{iter}	$0.2< \alpha_{iter} < 0.8$	$\alpha_{iter}=\{0.25;0.5;1\}$

Table 1. characteristic comparison between prototype and optimal algorithm

The value of the scaling factor α_{iter} depends on the current iteration in optimal Chase-Pyndiah algorithm. Keeping α_{iter} to a power of 2 for each iteration enables to remove a multiplication operation that becomes a simple bit shifting. Therefore, the elementary decoder area is decreased by 8% for a loss of BER performance inferior to 0.1dB. Similarly, eight iterations are classically done for the turbo process. However, significant gains are obtained during the first 5 iterations. The quantification level q has also a notable impact on decoding process and has to be chosen considering the potential loss in

terms of performance. The same remark can be done for N_{ep}, L_r and **c**. The complexity analysis led to a low complexity RS SISO elementary decoder (-30%) to be duplicated in the full-parallel turbo decoder.

Simulated performance of optimal Chase-Pyndiah algorithm and measured performance of full-parallel turbo decoder for the $(31, 29)^2$ RS product code are presented in Figure 5. Performance is evaluated by Monte-Carlo simulation and hardware emulation using our 5Gbps experimental setup, respectively. A cluster of computer working in parallel has given a reliable estimation for the optimal Chase-Pyndiah algorithm until a BER of 10^{-8}. As the hardware emulation speeds up the simulations by a few orders of magnitude, prototype performance has been measured until a BER of 10^{-13}. The impact of decoding parameters on the performance is clearly shown. Indeed, we observe a degradation of 0.5dB in terms of BER performance for the prototype at BER= 10^{-7}. Moreover, an error floor phenomenon occurs at lower SNR. It is not due to the asymptotic theoretical bound because we have applied the simple criterion introduced in [8]. This criterion improves by construction the binary minimum distance of the product code and thus the asymptotic performance. The parameter set that has been selected for our full-parallel turbo decoding architecture can explain this error floor. In particular, the number of N_{ep} error patterns affects the BER performance for lower SNRs.

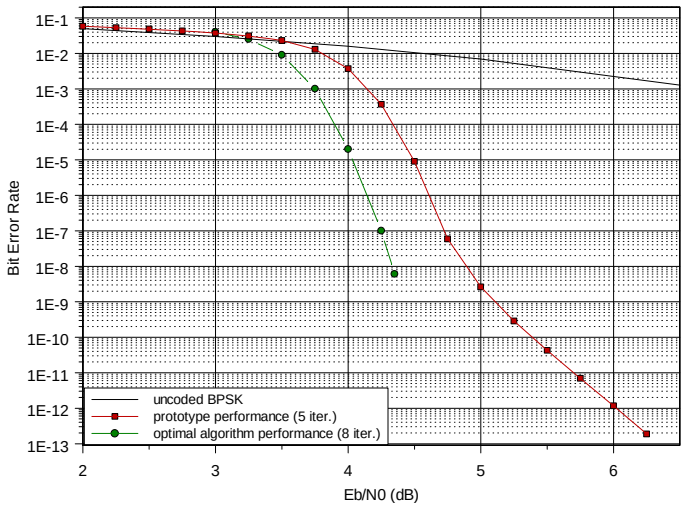

Fig. 5. $(31,29)^2$ RS turbo product code performance

VI. Conclusion

The use of RS product codes for forward-error correction in ultra high-speed context has been investigated in this article. We have designed a full-parallel turbo decoding architecture dedicated to the $(31, 29)^2$ RS product code and a 5Gbps experimental setup. The purpose of this prototype is to demonstrate that an RS turbo decoder can effectively achieve information rates above 1Gbps. The turbo decoder architecture enables decoding of TPCs at information rates until 10Gbps onto FPGA devices. A first application can be envisioned in the fiber optic communication systems: 10Gbps data transmission over passive optical networks. Moreover, enhancing the study to a larger code $((63, 61)^2$ RS product code) can increase throughput until 40Gbps and also be envisioned for 40Gbps line rate transmission over optical transport networks.

Acknowledgment

The authors thank E. Boutillon for providing the WGN IP. They are grateful to R. Le Bidan for providing the optimal bit error performance of iterative decoding of a $(31, 29)^2$ RS product code.

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. G. Gallager, "Low Density Parity Check Codes", IRE Trans. Inform. Theory, Jan. 1962, pp. 21-28.

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

[4] P. Adde, R. Pyndiah, and O. Raoul, "Performance and complexity of block turbo decoder circuits," in ICECS'96 Third International Conference on Electronics, Circuits and Systems, Oct. 1996.

[5] J. Cuevas Ordaz, P. Adde, S. Kerouédan, R. Pyndiah, "New architecture for high data rate turbo decoding of product codes", GLOBECOM02, Nov. 17-21, Vol. 2, 2002.

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

[7] R. Zhou, A. Picart, R. Pyndiah, A. Goalic, "Reliable transmission with low complexity Reed Solomon block turbo codes", ISWCS Conference, 2004.

[8] R. Le Bidan, R. Pyndiah, and P. Adde, "Some results on the binary minimum distance of Reed-Solomon codes and block turbo codes," in Proc. IEEE Int. Conf. Commun. ICC'07, Glasgow, Scotland, June 2007.

[9] E. Piriou, C. Jego, P. Adde, R. Le Bidan, M. Jezequel, "Efficient architecture for Reed Solomon block turbo code", in IEEE Int. Symp. on Circuits and Systems ISCAS'06, 21-24 May 2006.

[10] C. Jego, P. Adde, and C. Leroux, "Full-parallel architecture for turbo decoding of product codes", in Electronics Letters, vol. 42, no. 18, 31 August 2006.

[11] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "A 3.3-gbps bit-serial block-interlaced min-sum ldpc decoder in 0.13-um cmos," in Custom Integrated Circuits Conference, 2007. CICC '07. IEEE, 16-19 Sept. 2007.

[12] T. Mizuochi and all, "Experimental demonstration of net coding gain of 10.1 db using 12.4 Gbps block turbo code with 3-bit soft decision," in Optical Fiber Communications Conference, 23-28 March 2003.

[13] O. Aitsab and R. Pyndiah, "Performance of Reed-Solomon block turbo codes," in Proc. IEEE Global Telecommun. Conf. GLOBECOM96, London, UK, Nov. 1996, pp. 121–125.

[14] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," IEEE Trans. Inform. Theory, vol. 18, no. 1, pp. 170–182, Jan. 1972.

[15] Zhipei Chi; Parhi, K.K., "High speed VLSI architecture design for block turbo decoder", ISCAS 2002, IEEE International Symposium on Volume 1, May 2002, pp. 901-904.

[16] D. H. Lawrie, "Access and alignment of data in an array processor", IEEE Trans. Cornput**.,** vol. C-24, no. IO, pp. 1145-1155, Dec. 1975

[17] [Online] Available: http://www.dinigroup.com/DN9000k10PCI.php

[18] J. L. Danger, A. Ghazel, E. Boutillon, and H. Laamari, "Efficient FPGA implementation of gaussian noise generator for communication channel emulation," ICECS 2000, IEEE International Symposium on, Dec. 2000.

[19] C. Leroux, C. Jego, P. Adde, and M. Jezequel, "Towards Gbps turbo decoding of product code onto an FPGA device," in IEEE Int. Symp. on Circuits and Systems ISCAS'07, 27-30 May 2007, pp. 909–912.