

## A HIGHLY PARALLEL TURBO PRODUCT CODE DECODER WITHOUT INTERLEAVING RESOURCE

*Camille Leroux, Christophe Jego, Patrick Adde, Michel Jezequel, Deepak Gupta*

Institut TELECOM, TELECOM Bretagne, CNRS Lab-STICC  
firstname.lastname@telecom-bretagne.eu

### ABSTRACT

This article presents an innovative Turbo Product Code (TPC) decoder architecture without any interleaving resource. This architecture includes a full-parallel SISO decoder able to process  $n$  symbols in one clock period. Syntheses show the better efficiency of such an architecture compared with existing previous solutions. Considering a 6-iteration turbo decoder of a  $(32,26)^2$  BCH product code, synthesized in a 90nm CMOS technology, the resulting information throughput is 2.5Gb/s with an area of 233K gates. Finally a second architecture enhancing parallelism rate is described. The information throughput is 33.7Gb/s while an area estimation gives  $A = 10\mu m^2$ .

**Index Terms**— Product codes, Parallel architectures, Ultra-high-speed integrated circuits

### 1. INTRODUCTION

Nowadays, high throughput telecommunication systems such as optical fiber transmission systems or passive optical networks require powerful error correcting codes in order to increase their optical budget. Iterative decoding provides effective solutions for next generation optical systems. Recently, a (660,480) LDPC code decoder ASIC implementation was proposed. The information throughput is 2.4Gb/s while it could be enhanced to 10Gb/s with a (2048,1723) LDPC code [1]. Turbo product codes [2] also tend to be good candidates for emerging optical systems [3]. In [4], a BTC decoder is included in a 12.4Gb/s optical experimental setup. Since only a part of the transmitted data are actually coded, the information throughput of the BTC turbo decoder is 156Mb/s. The inherent parallel structure of the product code matrix confers to TPC good ability for parallel decoding. Nevertheless, enhancing parallelism rate rapidly induces the use of a prohibitive amount of memory. Some solutions were proposed in [5][6][7] to efficiently use the interleaving resources (IR), reaching high parallelism rates. However, a particular scheduling of the product code matrix enables to propose a turbo decoder architecture without any interleaving resource.

After a brief introduction of the TPC coding and decoding concept in section 2, section 3 reviews previous works and

proposes an innovative TPC decoder architecture without any interleaving resource. This new TPC decoder includes a new full-parallel SISO decoder architecture which is described in section 4. Section 5 gives some synthesis results and demonstrates the better efficiency of the proposed BTC decoder. Finally, a solution is proposed to enhance parallelism rate while highering the architecture efficiency. The interconnection issue is assessed and compared with an equivalent LDPC code decoder implementation.

### 2. TPC CODING AND DECODING PRINCIPLES

#### 2.1. Product codes

The concept of product codes is a simple and efficient method to construct powerful codes with a large minimum Hamming distance  $d$  using cyclic linear block codes [8]. Let us consider two systematic cyclic linear block codes  $C_1$  having parameters  $(n_1, k_1, d_1)$  and  $C_2$  having parameters  $(n_2, k_2, d_2)$  where  $n_i, k_i$  and  $d_i$  ( $i = 1, 2$ ) stand for code length, number of information symbols and minimum Hamming distance respectively. The product code  $P = C_1 \times C_2$  is obtained by placing  $(k_1 \times k_2)$  information bits in a matrix of  $k_1$  rows and  $k_2$  columns, coding the  $k_1$  rows using code  $C_2$  and coding the  $n_2$  columns using code  $C_1$ .

Considering that  $C_1$  and  $C_2$  are linear codes, it is shown that all  $n_1$  rows are codewords of  $C_2$  exactly as all  $n_2$  columns are codewords of  $C_1$  by construction. Furthermore, the parameters of the resulting product code  $P$  are given by  $n_p = n_1 \times n_2$ ,  $k_p = k_1 \times k_2$ , and  $d_p = d_1 \times d_2$  and the code rate  $R_p$  is given by  $R_p = R_1 \times R_2$ . Thus, it is possible to construct powerful product codes using linear block codes. In the following sections, we will consider a squared product code, meaning that  $n_1 = n_2 = n$ . The most commonly used component codes are Bose Chaudhuri Hocquenghem (BCH) codes. These codes are an infinite class of linear cyclic block codes that have capabilities for multiple error detection and correction. Product codes were adopted in 2001 as an optional correcting code system for both the uplink and downlink of the IEEE 802.16 standard (WiMAX) [9].

## 2.2. Iterative decoding of product codes

TPC decoding involves successively decoding rows and columns using SISO decoders. Repeating this soft decoding during several iterations enables the decrease of the Bit Error Rate (BER). It is known as the TPC decoding process. Each decoder has to compute soft information  $[R']_{it+1}$  from the channel received information  $[R]$  and the previous half-iteration computed information  $[R']_{it}$ .

Despite the existence of other decoding algorithms [10], the Chase-Pyndiah algorithm is known to give the best trade-off between performance and decoding complexity [11]. The Chase-Pyndiah SISO algorithm for a  $t = 1$  BCH code [12][2] is summarized below.  $t$  represents the maximum number of correctable errors.

1. Search for the  $p$  least reliable binary symbols of the previous half-iteration output vector  $[R']_{it}$ ,
2. Compute the syndrome  $S(t_0)$  of  $[R']_{it}$ ,
3. Compute the parity of  $[R']_{it}$ ,
4. Generate  $T$  test patterns  $t_i$  obtained by inverting some of the  $p$  least reliable binary symbols ( $T \leq 2^p$ ).
5. **For each** test pattern ( $1 \leq t_i \leq T - 1$ )
  - Compute the syndrome  $S(t_i)$ ,
  - Correct the potential error by inverting the bit position  $S(t_i)$ ,
  - Recompute the parity considering the detection of an error and the parity of  $[R']_{it}$ ,
  - Compute the square Euclidian distance (metric)  $M_i$  between  $[R']_{it}$  and the considered test pattern  $t_i$ .
6. Select the Decided Word (DW) among test patterns having the minimal metric ( $M_{DW}$ ) and choose  $Cw$  competitors codewords  $c_i$  ( $1 < i < Cw$ ) having the second minimum metric.
7. **For each** symbol of the DW,
  - Compute the new reliability  $F_{it} = \min_2(M_i) - \min(M_i)$  or  $F_{it} = \beta$  when no competitor exists,
  - Compute extrinsic information  $W_{it} = F_{it} - R'_{it}$ ,
  - Add extrinsic information (multiplied by  $\alpha_{it}$ ) to the channel received word,  $R'_{it+1} = R + \alpha_{it}W_{it}$ .

The  $\alpha_{it}$  coefficient allows decoding decisions to be damped during the first iterations. As detailed in [13], decoding parameters  $p$ ,  $T$ , and  $Cw$  have a notable effect on decoding performance. The number of quantization bits of the soft information  $q$  also impacts on performance.

## 3. PARALLEL DECODING OF PRODUCT CODES

### 3.1. Previous work

Many TPC decoder architectures were previously designed. The classical high speed approach involves the use of a pipelined structure at the iteration level. Separate decoding resources are assigned for each half-iteration. In existing architectures, the reconstruction of the matrix is necessary between each iteration: memory blocks are used between each half-iteration to store  $[R']_{it}$  and  $[R]$ . Each interleaving memory block is then composed of four memories of  $q \times n^2$  symbols. This solution presents several drawbacks. First, a large amount of memory is required which increases the global latency and the complexity of the design. In addition, increasing the parallelism degree of each half-iteration produces memory conflicts when several data have to be addressed at the same time.

In 2002, a new architecture was proposed [5] in order to increase the parallelism degree without any extra storage between half-iteration. The idea was to store several product code matrix symbols at the same address and to use elementary decoders able to process  $m$  symbols during the same clock period (denoted as  $m$ -decoders). A half-iteration structure includes  $m$  decoders each decoding  $m$  symbols in one clock period and an interleaving memory of size  $4qn^2$ . The resulting throughput is  $O(m^2)$  while the overhead factor of the decoder complexity is  $\sim \frac{m^2}{2}$ .

In [6], authors suggested to use a barrel shifter between decoding resources and the interleaving memory (IM) in order to avoid memory conflicts. This solution enables reaching the parallelism rate  $P = n$  by rotating the to be stored data. The extra-complexity only consists in a barrel shifter with a complexity in  $O(n \log(n))$ . However, the IM requirement is still prohibitive.

In [7][13], an IM-less architecture is detailed and prototyped on an FPGA device. In this architecture, a particular scheduling of the product code matrix decoding enables the interleaving memory to be replaced by an interconnection network (omega network). The complexity of the interleaving resources is then drastically reduced and highly-parallel structure can be implemented onto low-cost targets.

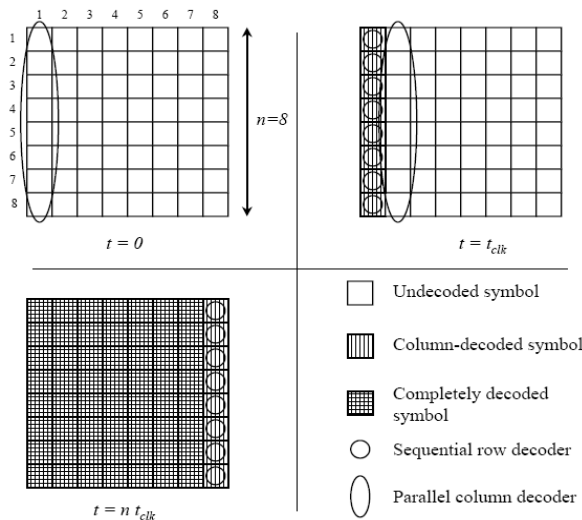
	Throughput	Memory (1/2 iteration)	
		IM	Internal
[5]	$O(m^2), m < n$ $m_{\max} = 8$	$O(4qn^2)$	$O(2sqnm)$
[6]	$O(n)$	$O(4qn^2)$ $+O(n \log n)$	$O(2sqn^2)$
[7][13]	$O(n)$	$O(n \log n)$	$O(2sqn^2)$

**Table 1.** Throughput and complexity order of previous high-speed architectures

These three architectures can reach high parallelism degrees (*i.e.* high throughput). However, as illustrated on in table 1, the internal memory complexity remains an important issue in the design of high-throughput TPC decoders.  $s$  is the number of pipeline stages inside the SISO decoder. Moreover, in these architectures, decoding resources consist in a duplication of sequential decoders. Increasing the parallelism rate by duplicating computation resources is inefficient since the reuse of available resources is not optimized. In the next section, we propose to merge the duplicated sequential SISO decoders into one full-parallel SISO decoder.

### 3.2. Proposed IM-free architecture using full-parallel SISO decoder

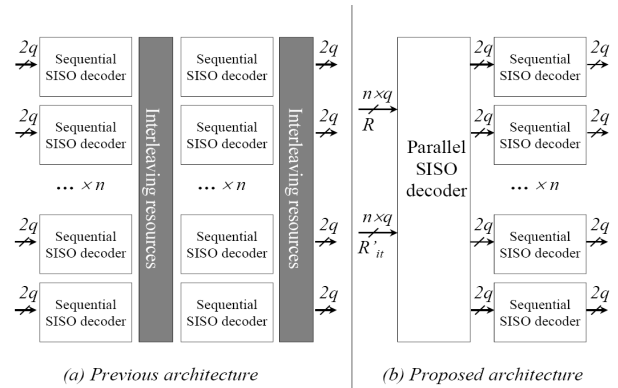
Considering that one can design a SISO decoder able to process  $n$  symbols in parallel in one clock period, a product code matrix can be decoded without any interleaving resource as shown in figure 1.



**Fig. 1.** Proposed parallel decoding scheduling of a product code matrix

At  $t = 0$ , the full-parallel SISO decoder processes the column 1. During the next clock period,  $n$  sequential SISO decoders start decoding the first symbol of each row while the parallel decoder process the column 2. During the  $n^{th}$  clock period, sequential decoders complete matrix decoding while the parallel decoder is already decoding the next matrix.

Data generated by the parallel decoder are immediately used by a sequential decoder. Consequently, no IM or data routing resources are required between the full-parallel decoder and sequential decoders. The resulting proposed architecture and the typical previous architecture for one iteration are depicted on figure 2.



**Fig. 2.** Previous TPC decoder architecture (a) and proposed full-parallel SISO based TPC decoder architecture (b)

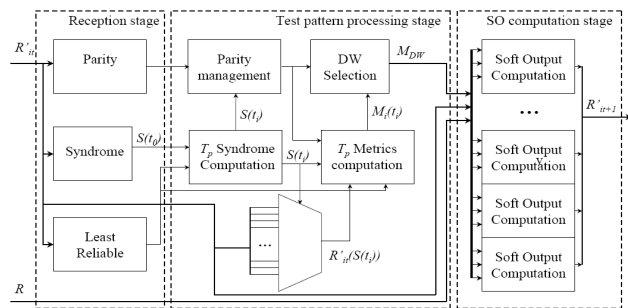
## 4. ARCHITECTURE OF A FULL-PARALLEL COMBINATORY SISO DECODER

### 4.1. Algorithmic parameter reduction

As explained in section 2, the Chase-Pyndiah algorithm includes parameters ( $p, T, Cw, q$ ) which impact on both the performance and the complexity of the turbo decoding. Performing 8 iterations, the parameter set  $p_0 = \{p = 5, T = 16, Cw = 3, q = 5\}$  gives the best performance for a reasonable complexity [11]. However, algorithmic simulations showed that the reduced parameter set  $p_1 = \{p = 3, T = 8, Cw = 0, q = 5\}$  only induce a performance loss of 0.25dB at BER=  $10^{-6}$  while it becomes nul below BER=  $10^{-9}$ . Consequently, using  $p_1$  enables the architecture to be simplified for a limited performance lost.

### 4.2. Full-parallel SISO decoder architecture

Figure 3 depicts the architecture of the full-parallel SISO decoder. It was firstly designed totally combinatorial, then, a critical path study enabled the insertion of pipeline stages within the structure.

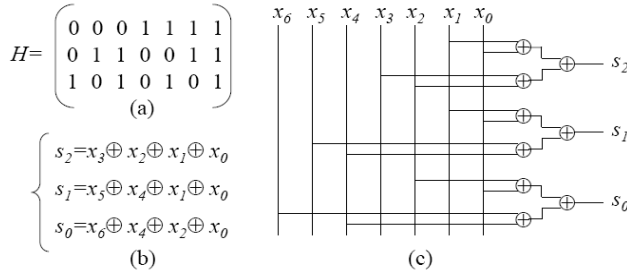


**Fig. 3.** Combinatorial version of the full-parallel SISO decoder

#### 4.2.1. Reception stage

The reception stage corresponds to steps (1-3) of the Chase-Pyndiah algorithm detailed in section 2.

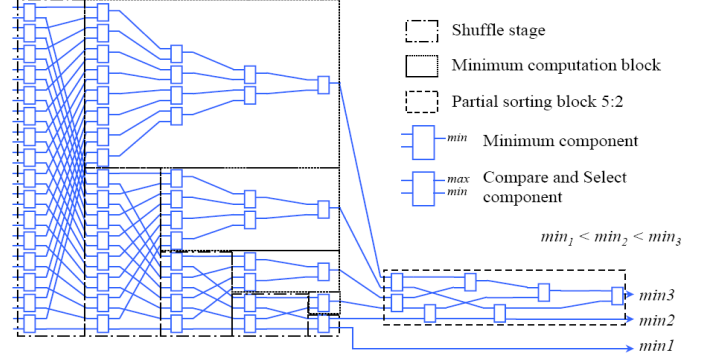
The syndrome of the incoming vector  $R'_{it}$  can be derived as  $S(R'_{it}) = H \times \text{sign}(R'_{it})$  where  $H$  is the parity check matrix of the BCH code. A straightforward implementation of such a matrix multiplication is depicted on figure 4. The  $H$  matrix, the corresponding parity check equations and the syndrome  $S(t_0) = [s_2, s_1, s_0]$  implementation of a (7,4) BCH code are detailed.



**Fig. 4.** BCH(7,4) code: (a) Parity check matrix (b) Parity check equations (c) Syndrome parallel computation implementation

It can be noticed that some parity check equations have similar terms. For instance, the term  $(x_1 \oplus x_0)$  is used in both  $s_1$  and  $s_2$  computation. This enables a reuse of computation resources for an even more efficient implementation. The parity of the incoming vector  $R'_{it}$  is computed with a similar structure by "xoring"  $(n - 1)$  incoming bits.

Selecting the least reliable bits among the incoming vector in parallel requires a sorting network. Such structures are composed of interconnected Compare and Select operators (CS). The interconnection scheme depends on the considered sorting algorithm. Many parallel sorting algorithm are conceivable [14]. However, most of them are optimized for a complete sorting, while the Chase-Pyndiah algorithm only requires a partial sorting (*i.e.* extracting  $p$  minima). Consequently we devised a network optimized, in terms of area and critical path, for the partial sorting of  $p=3$  values among  $n=32$ , as depicted on figure 5. The structure is based on shuffle networks coupled with local minima computation blocks. After the first shuffle stage,  $\text{min}_1$  is in the lower section while the upper section can either contain  $\text{min}_2$  or  $\text{min}_3$  or no minimum. The same reasoning is applied recursively. After 5 shuffle stages, the minimum is determined while 5 values can still be  $\text{min}_2$  and  $\text{min}_3$ . A local sorting of this 5 values enables the determination of  $\text{min}_2$  and  $\text{min}_3$  value. This partial sorting network requires 35 CS and 29 minimum elements. The critical path consists in 9 comparisons stages.



**Fig. 5.** Optimized sorting network for least reliable bits selection

#### 4.2.2. Test pattern processing stage

The test pattern processing stage corresponds to steps (4-5) in the Chase-Pyndiah algorithm detailed in section 2. Instead of being processed sequentially, test patterns are processed in parallel. The syndrome of each test pattern is computed by adding  $S(t_0)$  with the position of the inverted reliable bits. The parity management block computes the parity of  $R'_{it+1}$  considering the parity of  $R'_{it}$  and the detection of an error which is the case when  $S(t_i) \neq 0$ . Metrics of each test pattern is then computed by adding the contribution of each inverted bit in the current test pattern (least reliable bits, syndrome corrected bits and, the new parity bit). The minimum metric is determined in the DW selection block. The structure is a simple minimum selection tree. The multiplexer selects  $R'_{it}(S(t_i))$  in order to compute test pattern metrics.

#### 4.2.3. Soft output computation stage

The last stage is a duplication of  $n$  soft output computation blocks. As shown on figure 6, this block first computes the new reliability  $F_{it}$  of each symbol. Since, no competitor is considered, the  $\beta$  value is automatically assigned. The  $\beta$  value is based on an estimation of the competitor word metric value. It is calculated with the reliability of the corrected bit and the least reliable bits. Then, the extrinsic information is computed and damped by the coefficient  $\alpha_{it}$  which is devised to be a power of 2 making the multiplication a simple bit shifting. Finally, the channel information is added to generate the soft output  $R'_{it+1}$ . Within this block, all computation are performed in sign and magnitude format. Other arithmetic format were explored but the chosen one requires less computation resources than others.

## 5. AREA VS THROUGHPUT TRADEOFF

In this section, we compare the parallel and the sequential SISO decoders in terms of throughput, complexity and ef-

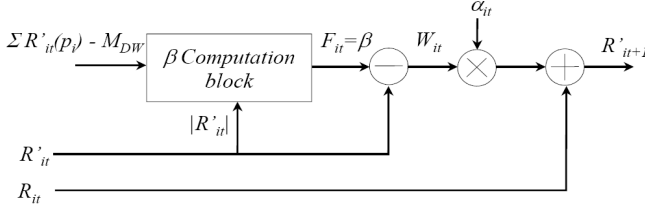


Fig. 6. Soft output computation stage

efficiency. Then a pipelined version of the parallel SISO decoder is proposed. Logic syntheses were performed using Synopsys Design Compiler with a ST-microelectronics 90nm CMOS process. The area is transposed in logic gate count. One equivalent logic gate corresponds to the area of a 2-input NAND gate. It enables a more technology-independent measure of the complexity.

### 5.1. Logic synthesis results

We designed 5 versions of the (32,26) BCH parallel SISO decoder having from 1 to 5 pipeline stages. The 1-pipeline stage version is a full-combinatorial architecture with register banks only in input and output. Table 2 summarizes synthesis results of the 5 different full-parallel SISO decoder and compare them with  $n$  duplicated sequential SISO.

$s$  is the number of pipeline stages,  $f_{\max}$  is the maximum working frequency reached during synthesis,  $A$  represents the area of the design in equivalent gate count, the parallelism rate  $P$  corresponds to the number of processed symbols per clock period.  $T_{in}$  is the input throughput such as  $T_{in} = P \times f$  and  $T_{out}$  is the information throughput  $T_{out} = R_p \times P \times f$ . Increasing throughput regardless of turbo decoder complexity is not relevant. In order to compare the throughput and complexity of SISO decoders, the efficiency of each decoder is defined as :  $\eta = \frac{T_{out}}{A}$ .

$s$	Parallel SISO decoder					Seq-based
	1	2	3	4	5	3
$f_{\max}$ (Mhz)	222	385	526	690	741	700
$A$ (Kgates)	47	32	33.8	39	36.2	200
$T_{in}$ (Gb/s)	7.1	12.3	16.8	22.1	23.7	22.4
$T_{out}$ (Gb/s)	4.7	8.1	11.1	14.6	15.6	14.8
$\eta$ (Mb/s/gate)	0.1	0.25	0.33	0.37	0.43	0.07

Table 2. Comparison of sequential and parallel (32,26) BCH SISO decoder performances

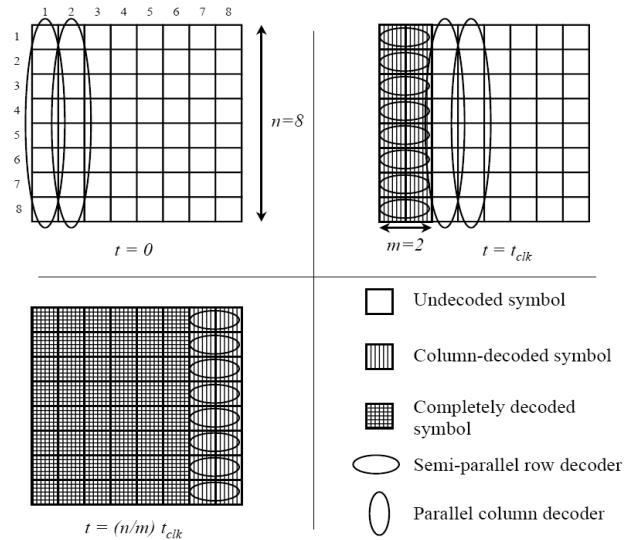
Table 2 shows that the parallel SISO decoder can reach high throughput with low complexity. This clearly shows the higher efficiency of the parallel SISO decoder compared with

a duplication of  $n$  sequential decoders. Actually a fully parallel structure enables a better reuse of computation and memory resources. For instance, in a parallel SISO decoder, the memory requirement for  $R$  and  $R'_{it}$  are  $O(2sqn)$  while in a sequential-based parallel decoder, it is  $O(2sqn^2)$ .

Since the resulting throughput is limited by the slower stage, the maximum throughput of a full-iteration module is 14.8Gb/s. In this case, a full-iteration of the proposed architecture (see figure 2b) takes 236Kgates while the previous architecture (see figure 2a) takes 400Kgates.

### 5.2. Towards a maximal parallelism rate

Previous synthesis results showed the better efficiency of parallel architecture compared with duplicated architectures.



Alternative turbo decoding scheduling for enhanced parallelism rate

Figure 5.2 proposes an alternate product code matrix parallel decoding scheme where  $m$  parallel decoders are used for column decoding while row decoding is performed by  $n$   $m$ -decoders. A  $m$ -dec decoder can decode  $m$  symbols in one clock period and  $1 < m < n$  [5]. In such an architecture, the maximum reachable parallelism rate  $P = n^2$  can be achieved by using  $n$  full-parallel SISO decoder for column decoding and  $n$  full-parallel SISO decoder for row decoding. Considering that the TPC decoder includes  $2n$  parallel SISO decoders (with  $s = 5$  pipeline stages) for  $it$  decoding iterations, the turbo decoder output throughput is

$$T_{out} = \frac{P \times f \times R_p}{it}$$

Placing and routing  $2n$  parallel SISO decoders would reduce the maximum working frequency of the parallel SISO decoder. Nevertheless, using the previous equation, an information throughput  $T_{out}=10\text{Gb/s}$  is reached when  $f > 88\text{MHz}$  which is a most probably achievable frequency. Furthermore,

a reasonable working frequency of  $f=300\text{MHz}$  leads to a  $(32,26)^2$  BCH product code turbo decoder with an information throughput  $T_{out}=33.7\text{Gb/s}$ . The total area of such a parallel turbo decoder is  $A=10\mu\text{m}^2$ . The achieved parallelism rate ( $P=n^2=1024$ ) is similar to a full parallel  $n=1024$  LDPC decoder. The number of interconnection among the product code turbo decoder is  $I_n(\text{TPCD})=2\times n_{BCH}^2\times q$  while an equivalent full parallel 1024-LDPC decoder would have  $I_n(1024\text{-LDPC})=n_{LDPC}\times q\times d_v$  where  $d_v$  represents the variable node degree. Consequently, as long as  $d_v>2$ , the following inequation is verified:  $I_n(\text{TPCD})<I_n(1024\text{-LDPC})$ .

## 6. CONCLUSION

High throughput TPC decoder architecture complexity is made prohibitive by the amount of memory usually required for data interleaving and pipelining. In this article, we proposed an innovative product code matrix decoding scheduling which enables any interleaving resource to be removed. The resulting architecture requires a full-parallel SISO decoder able to decode  $n$  symbols in one clock period. Such a SISO decoder architecture is described and includes a new optimized parallel sorting network. ASIC based-logic synthesis showed the better efficiency of the proposed architecture. Actually, compared to a previous architecture, the area is reduced while the throughput is the same. Finally, in order to enhance parallelism rate and throughput, a slightly different scheduling is proposed. With such a scheduling, the maximum parallelism rate  $O(n^2)$  can be achieved. An area estimation showed that a  $(32,26)^2$  BCH code can be decoded at  $33.7\text{Gb/s}$  with an estimated silicon area of  $10\mu\text{m}^2$ .

## 7. REFERENCES

- [1] 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, pp. 459-462.
- [2] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes," in *Global Telecommunications Conference, 1994. GLOBECOM '94. 'Communications: The Global Bridge'. IEEE*, 28 Nov.-2 Dec. 1994, pp. 339-343vol.1.
- [3] T. Mizuochi, K. Kubo, H. Yoshida, H. Fujita, H. Tagami, M. Akita, and K. Motoshima, "Next generation fec for optical transmission systems," in *Optical Fiber Communications Conference, 2003. OFC 2003*, 23-28 March 2003, pp. 527-528vol.2.
- [4] T. Mizuochi, K. Ouchi, T. Kobayashi, Y. Miyata, K. Kuno, H. Tagami, K. Kubo, H. Yoshida, M. Akita, and K. Motoshima, "Experimental demonstration of net coding gain of 10.1 db using 12.4 gb/s block turbo code with 3-bit soft decision," in *Optical Fiber Communications Conference, 2003. OFC 2003*, 23-28 March 2003, pp. PD21-P1-3vol.3.
- [5] J. Cuevas, P. Adde, S. Kerouedan, and R. Pyndiah, "New architecture for high data rate turbo decoding of product codes," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, 17-21 Nov. 2002, vol. 2, pp. 1363-1367vol.2.
- [6] Zhipei Chi and K.K. Parhi, "High speed vlsi architecture design for block turbo decoder," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 26-29 May 2002, vol. 1, pp. I-901-I-904vol.1.
- [7] C. Jego, P. Adde, and C. Leroux, "Full-parallel architecture for turbo decoding of product codes," in *Electronics Letters*, 31 August 2006, vol. 42, pp. 55-56.
- [8] P. Elias, "Error-free coding," *Information Theory, IEEE Transactions on*, vol. 4, no. 4, pp. 29-37, Sep 1954.
- [9] IEEE Std 802.16-2001, "Ieee standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems," December 2001.
- [10] Jr. Forney, G., "Generalized minimum distance decoding," *Information Theory, IEEE Transactions on*, vol. IT-12, pp. 125-131, April 1966.
- [11] P. Adde, R. Pyndiah, and O. Raoul, "Performance and complexity of block turbo decoder circuits," in *Electronics, Circuits, and Systems, 1996. ICECS '96., Proceedings of the Third IEEE International Conference on*, 13-16 Oct. 1996, vol. 1, pp. 172-175vol.1.
- [12] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory*, vol. IT, pp. 170-182, january 1972.
- [13] C. Leroux, C. Jego, P. Adde, and M. Jezequel, "Towards gb/s turbo decoding of product code onto an fpga device," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 27-30 May 2007, pp. 909-912.
- [14] S. G. Akl, *Parallel sorting algorithms*, Academic Press, 1985.