A Novel Architecture For Elementary Check Node Processing In Non-Binary LDPC Decoders

Oussama Abassi, Laura Conde-Canencia, Ali Al Ghouwayel and Emmanuel Boutillon

***Abstract*—This paper presents an efficient architecture design for Elementary Check Node processing in Non-Binary Low-Density Parity-Check decoders based on the Extended Min-Sum algorithm. This architecture relies on a simplified version of the Bubble Check algorithm and is implemented by the means of FIFOs. The adoption of this new design at the Check Node level results in a high-rate low-cost full-pipelined processor. A proof-of-concept implementation of this processor shows that the proposed architecture halves the occupied FPGA surface and doubles the maximum frequency without modifying the input/output behavior of the previous one.**

I. Introduction

Non-Binary (NB) Low-Density Parity-Check (LDPC) codes are now known to outperform both binary LDPC and Turbo-Codes when considering moderate or small code lengths [1], [2]. This family of codes retain the benefits of steep waterfall region (typical of convolutional turbo-codes) and low error floor (typical of binary LDPC). Compared to their binary counterparts, NB-LDPC codes generally present higher girths, which leads to better decoding performance. Moreover, their association with high-order modulations is advantageous as symbol likelihoods are calculated directly, without any marginalization [3]. Different works have also revealed the interest of NB-LDPC in MIMO systems ([4] [5] [6]).

However, these advantages entail the drawback of high computational complexity because NB-LDPC are defined over a Galois Field $\mathbb{GF}(q = 2^m)$ (where $q >> 2$ is the order of $\mathbb{GF}$), i.e. the non-zero entries of their parity-check matrices belong to high-order finite fields. Elements of $\mathbb{GF}(q)$ are called *symbols*, and each symbol is a set of m bits. Consequently, in the decoding process, each message exchanged between processing nodes in the associated Tanner graph is an array of values, each one corresponding to a $\mathbb{GF}$ element. From an implementation point of view, this leads to a highly increased complexity compared to binary LDPC.

In the last years, important effort has been dedicated to reduce complexity of NB-LDPC decoders and several algorithms with their associated architectures have been proposed. In this brief we propose a new design for the so-called L-Bubble Check architecture that is used to implement the Extended Min-Sum algorithm [7]. Without modifying the algorithm, we moved the data-dependent computations to the last stage of the architecture. This modification allows to relax the critical path and significantly simplifies the hardware design.

The brief is organized as follows. Section II provides a state of the art on NB-LDPC decoding algorithms and architectures. Section III presents notations and principles of NB-LDPC codes. Section IV describes the Min-Sum algorithm for NB-LDPC codes as well as the Elementary CN processing. Section V describes the algorithm and architecture of the FIFO-based Elementary CN processor. Section VI presents the FPGA post-synthesis results to compare the new design with the state-of-the-art. Finally, conclusion and perspectives are discussed in Section VII.

II. State of the art on NB-LDPC decoding

The direct application of the Belief Propagation (BP) algorithm to NB-LDPC codes leads to a computational complexity dominated by $\mathcal{O}(d_c \cdot q^2)$ [1] [8] for each check node update, which becomes prohibitive when considering values of $q > 16$. An important effort has thus been dedicated to develop reduced-complexity algorithms for NB-LDPC decoding. In order to reduce the prohibitive complexity of the BP algorithm for high-order NB-LDPC codes, the authors in [9] proposed to perform the BP algorithm in the logarithmic domain. This replaces all the products by the max^* operation, without any performance loss for $\mathbb{GF}(8)$. In [10] an FFT-Based BP decoding algorithm was proposed. The description of this algorithm in the log domain was presented in [11]. Note that the Fourier transform is easily computed when the $\mathbb{GF}$ is a binary extension field with order $q = 2^m$ and in this case the computational complexity of the BP algorithm is reduced to the order of $\mathcal{O}(d_c \cdot q \cdot m)$ per check node. Decoding of $\mathbb{GF}(256)$-LDPC codes using this method was described in [12]. However, although these algorithms considerably reduce the computational complexity of the decoding process, they are still far from being considered for hardware implementation. This implementation became feasible with the introduction of the Extended Min-Sum (EMS) [7] and the Min-Max [13] algorithms.

The EMS algorithm [7], [14] is based on a generalization of the Min-Sum algorithm (initially proposed for binary LDPC codes [15]). The EMS has the advantage of performing only additions while truncating the size of the messages from q to n_m ($n_m \ll q$). This sub-optimality introduces a performance degradation that is compensated by a correction factor that can be optimized so that the EMS algorithm can approach, or even in some cases slightly outperform, the BP-FFT decoder [10] [12]. Also, the complexity/performance trade-off can be

Oussama Abassi, Laura Conde-Canencia and Emmanuel Boutillon are with the Lab-STICC laboratory, CNRS UMR 6285, Université de Bretagne Sud, BP 92116, 56321 Lorient, France.

Ali Al-Ghouwayel is with the CCE Department, Lebanese International University (LIU), Beirut, Lebanon.

adjusted with the value of the n_m parameter, making the EMS decoder architecture flexible for both implementation and performance constraints. In the Min-Max algorithm, the extrinsic messages exchanged within the Min-Sum-based decoder are composed of a set of $\mathbb{GF}$ symbols with their corresponding reliability metrics measured with respect to the most likely one. By using appropriate metrics, the author in [13] derived a low-complexity quasi-optimal iterative algorithm as well its canonical selective implementation that reduces the number of operations at each decoding iteration. Different architectures for CN have been proposed based on Min-Max algorithm [**?**], trellis-based approach [**?**] and a basis construction [18]. Also, a simplified version of the Min-Sum algorithm and its associated architectures were presented in [19] and [20].

Two other alternative approaches have been proposed for NB-LDPC decoding. The first one is based on symbol flipping algorithms, characterized by their low complexity at the cost of performance degradation [21]. The second approach is based on stochastic computations [22].

Complexity reduction of the EMS-based CN processing has been investigated in [23], [24] and specifically the Elementary Check Node processor (ECN) which constitutes the core of the CN based on the Forward-Backward (FB) structure [9]. According to this FB model, the CN is composed of $3 \cdot (d_c - 2)$ ECNs, thus, simplifying the ECN architecture will considerably reduce the global decoder complexity. The Bubble Check and L-Bubble Check algorithms proposed in [23], [24] constitute two original approaches in the design of ECN allowing the reduction of hardware complexity from $\mathcal{O}(n_m^2)$ [14] to $\mathcal{O}(n_m \cdot \sqrt{n_m})$ and $\mathcal{O}(4 \cdot n_m)$, respectively. The L-Bubble Check only considers the first two columns and the first two rows of the $n_m \times n_m$ ECN matrix. In other words, the paths to follow are predetermined and the size of the bubble sorter is always fixed to 4. These characteristics significantly simplify the ECN architecture compared to the original Bubble Check. Based on the L-Bubble Check algorithm, a global architecture for a $\mathbb{GF}(64)$-LDPC EMS decoder was presented in [25]. In this architecture the ECN contains a feedback loop including the sorter that outputs valid candidates. This mechanism results in a long critical path that greatly impacts the latency of the architecture. Recently, the authors in [26] presented the Syndrome-Based algorithm which is a lower-complexity hardware approach for EMS-based CN processing that allows increased parallelism while achieving slightly better communication performance.

In this brief we propose a novel and efficient architecture of the ECN processor for the EMS based decoder. The proposed ECN architecture is based on the efficient use of FIFO buffers that directly output the candidates to the sorter. Also, we consider a Tree-based architecture for the CN processor, instead of the FB solution to reduce the numbers of ECN in the critical path. Implementation results on FPGA show that the CN processor area is divided by two, the maximum operating frequency is doubled and the hardware efficiency is enhanced by a factor of 6, compared to previous EMS implementations [25].

III. NB-LDPC CODES

This section presents NB-LDPC codes and provides some details and references on the matrix construction.

A. Definition of NB-LDPC codes

An NB-LDPC code is a linear block code defined on a very sparse parity-check matrix H whose nonzero elements belong to a Galois field $\mathbb{GF}(q)$, where $q > 2$. A codeword is denoted by $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$, where $\mathbf{x}_k$, $k = 1 \ldots N$ is a $\mathbb{GF}(q)$ symbol represented by $m = \log_2(q)$ bits as follows: $\mathbf{x}_k = (x_{k,1}\ x_{k,2} \ldots x_{k,m})$. The construction of these codes is expressed as a set of parity-check equations over $\mathbb{GF}(q)$, where a single parity equation involving d_c codeword symbols is:

$$\sum_{k=1}^{d_c} h_{j,k}\mathbf{x}_k = 0 \tag{1}$$

with $h_{j,k}$ the nonzero values of the j-th row of H. The dimension of matrix H is $M \cdot N$, where M is the number of parity-CNs and N is the number of Variable Nodes (VN), i.e. the number of $\mathbb{GF}(q)$ symbols in a codeword.

B. Construction of NB-LDPC codes

The Tanner graph of a NB-LDPC code is usually much sparser than the one of its homologous binary counterpart for the same rate and binary code length. The ultra-sparse codes [27] achieve better performance with VN degrees $d_v = 2$ because this reduces the stopping and trapping sets effects and then performance of message-passing algorithms become closer to the optimal performance of Maximum Likelihood decoding. The protograph-based codes [28], [29], [30] obtain both good error correcting performance and hardware friendly decoder architecture by maximizing the girth of the Tanner graph and minimizing the multiplicity of the cycles with minimum length [31].

IV. MIN-SUM ALGORITHM FOR NB-LDPC DECODING

The EMS algorithm [7] is an extension of the Min-Sum ([32] [15]) algorithm from binary to NB-LDPC codes. The exchanged messages between the VN and CN processors consist of vectors of Log-Likelihood Ratio (LLR) values.

A. Definition of NB LLR values

The first step of the Min-Sum algorithm is the computation of the LLR value for each symbol of the codeword. With the hypothesis that the $\mathbb{GF}(q)$ symbols are equiprobable, the LLR value $L^k(\mathbf{x})$ of the k^{th} symbol is given by [13]:

$$L^k(\mathbf{x}) = \ln\left(\frac{P(\mathbf{y}_k|\tilde{\mathbf{x}}_k)}{P(\mathbf{y}_k|\mathbf{x})}\right) \tag{2}$$

where $\tilde{\mathbf{x}}_k$ is the symbol of $\mathbb{GF}(q)$ that maximizes $P(\mathbf{y}_k|\mathbf{x})$, i.e. $\tilde{\mathbf{x}}_k = \arg\max_{\mathbf{x}\in\mathbb{GF}(q)}, \{P(\mathbf{y}_k|\mathbf{x})\}$ and $\mathbf{y}_k$ is the received symbol.

Note that $L^k(\tilde{\mathbf{x}}_k) = 0$ and, for all $\mathbf{x} \in \mathbb{GF}(q)$, $L^k(\mathbf{x}) \geq 0$. Thus, when the LLR of a symbol increases, its reliability decreases. This LLR definition avoids the need to re-normalize

the messages after each node update computation and permits to reduce the effect of quantization when considering the finite precision representation of the LLR values.

B. CN processing in the Min-Sum algorithm

With the FB algorithm [9] a CN of degree d_c can be decomposed into $3(d_c-2)$ ECNs, where an ECN has two input messages U and V and one output message E. Each of the messages can be written as $U=[U(1),U(2),\cdots,U(n_m)]$ where each $U(i)$ represents a couple $(u_i^{\mathbb{GF}},u_i^L)$ where $u_i^{\mathbb{GF}}\in\mathbb{GF}(q=2^m)$ and u_i^L is its associated LLR value. The messages are sorted in increasing order as follows : $u_1^L=0$, $u_i^L\geq 0$, $i=2,\cdots n_m$ and $\forall i\leq j,\Rightarrow u_i^L\leq u_j^L$ [13] [25]. With the Min-Sum algorithm the values in message E = $\{(e_i^{\mathbb{GF}},e_i^L)\}$, $i=1,\ldots,n_m$, correspond to the n_m minimum values of the set $\{u^L+v^L\}$ such that $u^{\mathbb{GF}}\oplus v^{\mathbb{GF}}=e^{\mathbb{GF}}$ where $\oplus$ is the addition in $\mathbb{GF}(q)$.

C. EMS ECN processing

In practice, the role of an EMS ECN processor is to select the n_m most reliable symbols in the 2D matrix T_Σ (Fig. 1), where $T_\Sigma(i,j)=U(i)+V(j)\quad\forall(i,j)\in[1,n_m]^2$. This addition represents the addition of the LLR values $u_i^L+u_j^L$ and the $\mathbb{GF}(q)$ values $u_i^{\mathbb{GF}}\oplus u_j^{\mathbb{GF}}$. The most reliable symbols correspond to the smallest LLR values in T_Σ. To simplify the selection of these symbols, the authors in [23], [24] showed that the exploration of only a portion of matrix T_Σ does not impact the decoder performance. To be specific, the L-Bubble Check algorithm divides this portion into four paths that constitute the so-called L-path [24].

V. S-Bubble Check algorithm and architecture

A. Principle

Let the four straight paths be (Fig. 1):

- $S_1=\{V(1)+U(j_1)\}_{j_1=1\ldots n_m}$
- $S_2=\{U(1)+V(1+j_2)\}_{j_2=1\ldots n_m}$
- $S_3=\{V(2)+U(1+j_3)\}_{j_3=1\ldots\frac{n_m}{2}}$
- $S_4=\{U(2)+V(2+j_4)\}_{j_4=1\ldots\frac{n_m}{2}}$

As each path is inherently sorted in increasing order, to extract the n_m most reliable symbols in T_Σ we only need to compare one candidate from each path during n_m clock cycles. At the end of each comparison, the most reliable candidate is updated by the next symbol on the same path, as shown by the arrows in Fig. 1. A detailed description is given in Algorithm 1. This algorithm is named S-Bubble Check to make reference to the straight paths. If q is high, typically, $q\geq 256$, the use of only four paths may introduce performance degradation. In that case, it is possible to add extra straight paths (starting from $U(3)$ and $V(3)$ for a 6-path ECN, for example) while keeping the same architecture structure.

B. Redundancy control and non-valid couples

Redundancy in $\mathbb{GF}(q)$ symbols occurs when the most recent selected candidate corresponds to a symbol that is already selected and contained in the output vector denoted by E. Then, if a redundant symbol is detected, only its first occurrence is considered valid and the followings are tagged as non-valid.

Algorithm 1 S-Bubble Check algorithm

Initialization step:
$E=\varnothing$
for $i=1$ **to** 4 **do**
  $C_i=S_i(1)$; $j_i=2$
end for
Note that $C_i=(c_i^{\mathbb{GF}},c_i^L)$
Updates:
for $l=1$ **to** n_m **do**
  $k=\arg\min\{c_i^L,i=1\ldots4\}$
  $C_k=S_k(j_k)$; $j_k=j_k+1$. This operation is equivalent to a **pull** operation in the FIFO buffers (see Figure 2).
  if $c_k^{\mathbb{GF}}\notin E^{\mathbb{GF}}$ **then**
    $E=E\cup C_k$
  end if
end for

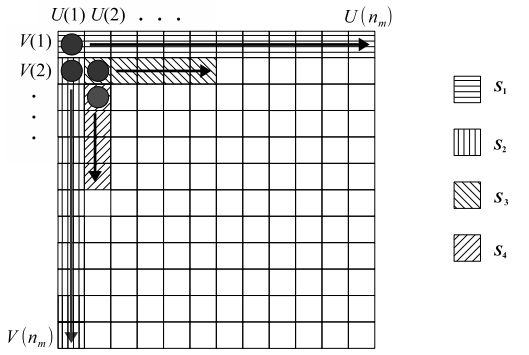

Fig. 1. Matrix T_Σ and the exploring strategy of the S-Bubble Check algorithm

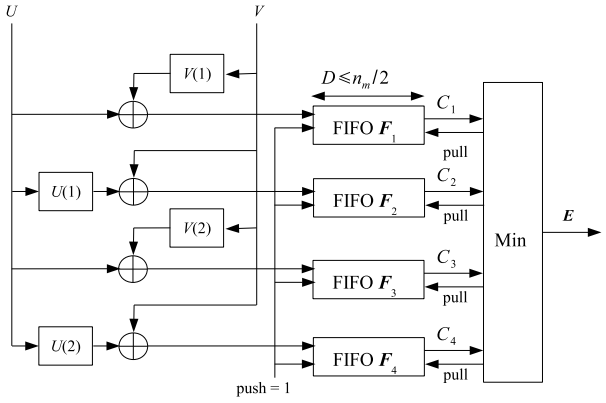

Fig. 2. S-Bubble ECN architecture

C. S-Bubble Elementary Check Architecture

Fig. 2 shows the architecture of the S-Bubble ECN. The adders receive vectors U and V and perform the LLR and $\mathbb{GF}(q=2^m)$ additions as previously described. The results are directly provided to the FIFOs on a clock cycle basis (**push** always equal to 1). Note that each FIFO is receiving the elements of the corresponding path in matrix T_Σ. The operator **Min** compares the outputs of the four FIFOs and selects the minimum LLR value with its associated $\mathbb{GF}$ symbol that will constitute a new element of message E. The selected

candidate will be freed from the relevant FIFO (**pull** = 1), and the FIFO will output a new candidate in the next clock cycle. This process is repeated n_m times. After $n_m/2$ clock cycles, $n_m/2$ symbols have been output and $n_m/2$ symbols still remain in the FIFOs. In the worst case, all those symbols are extracted from a FIFO not yet being read during the first $n_m/2$ clock cycles. Thus, the maximum size D of a FIFO can be bounded by $n_m/2$ if a mechanism preventing the input of new symbols, once a FIFO is full, is employed.

A detailed clock cycle examination combined with low level hardware FIFO behavior (not described here) leads to sizes $n_m/2$ for F_1 and F_2, $n_m/2-1$ for F_3 and $n_m/2-2$ for F_4. Note that, for the sake of simplicity, Fig. 2 does not show the control unit that tracks redundant symbols in the output message E. Also note that in the implementation described in [25], a number $n_{op} > n_m$ (typically, $n_{op} = n_m + \delta$, with $\delta = 2$) of messages are generated at the output of the ECN to compensate the fact that the redundant output symbols are discarded. In such case, the FIFO sizes should be lengthened by $\delta/2$.

As described in [25], the critical path of the L-Bubble Check ECN architecture contains a feedback loop including several elements: RAM access, adder, comparators and an index update operation, along with complex control. This mechanism results in a long critical path that greatly impacts the clock frequency. In the S-Bubble architecture, the critical path on the feedback loop (the right part of the FIFOs in Fig. 2)) is reduced to the **Min** processing and to the FIFO accesses (**pull** operation).

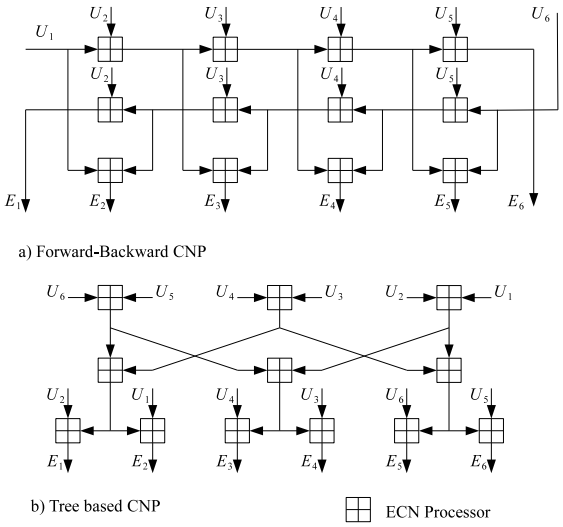

Fig. 3. Example of Forward-Backward and Tree-based CNP architecture for $d_c = 6$

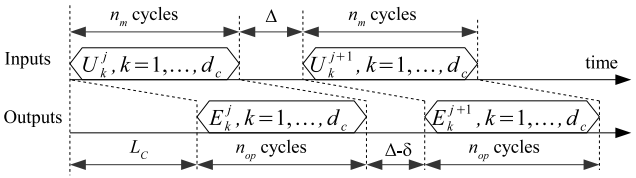

Fig. 4. Time diagram of a CNP. The d_c messages U of CN j are input in parallel in n_m clock cycles. After a latency of Δ clock cycles, CN $j+1$ performs.

D. Check Node Processor (CNP) Architecture

The CNP can be designed based on the FB architecture (Fig. 3.a) or alternatively, using a Tree-based structure [**?**] as illustrated in Fig. 3.b for $d_c = 6$. The main advantage of the Tree structure is that the number of ECN in the critical path is minimized and constant for all outputs. For these reasons, we considered the Tree structure in our work. Fig. 4 illustrates the timing diagram: L_C is the CNP latency and Δ is the time delay required to start a new processing. Note that the symbols of the messages entering the CNP must be multiplied by the non-zero entries of the parity-check matrix (the row corresponding to the CNP in the Tanner graph), as well as the output messages of the CNP that are divided by these non-zero entries. Therefore, the implemented CNP architecture uses the wired multipliers presented in [25] to perform multiplications over $\mathbb{GF}(q)$.

VI. FPGA PROTOTYPING

The proposed FIFO-based ECN was implemented on the Xilinx Virtex XC5VLX330T speed-2 FPGA device. For comparison purposes, we considered the same design parameters as in [25], which are: $q = 64$, $m = \log_2 q = 6$, $l = 6$ (l is the number of bits used to represent each LLR value), $n_m = 12$ and $n_{op} = 14$. The CN degree d_c is set to 4, 6, 8 and 12, corresponding to code rates of 1/2, 2/3, 3/4 and 5/6, respectively. Memory units were synthesized using distributed RAMs. In order to compare the architecture, we define the hardware efficiency E_n of the architecture as the ratio between the number of CNs processed in a second divided by the number S of slices in the CNP. In the proposed architecture, a new CN can be started every n_{op} clock cycles, thus, in a second, F/n_{op} CNs can be processed, with F the clock frequency of the design: $E_n = F/n_{op}/S$ CN/s/slice. If we denote by γ the number of ECNs contained in the CNP critical path and considering that the latency of an ECN is 2 clock cycles, the latency L_C can be expressed as: $L_C = 2 \cdot \gamma + 2$, where $\gamma \leq (d_c+1)/2$ in the Tree architecture and $\gamma = d_c - 2$ in the FB one. The two extra clock cycles of L_c are required for input and output $\mathbb{GF}$ multipliers.

TABLE I
POST-SYNTHESIS RESULTS COMPARISON FOR THE CN PROCESSOR

	[25]	Proposed S-Bubble with Tree architecture			
	$d_c = 6$	$d_c = 4$	$d_c = 6$	$d_c = 8$	$d_c = 12$
Slices	4287	1235	2288	3388	5046
F (MHz)	91	214	208	209	203
L_C	10	6	8	10	12
Δ	8	2	2	2	2
E_n	**1,061**	12,400	**6,493**	4,406	2,876

Table I presents the post-synthesis results obtained for the CN processor considering the S-Bubble ECN architecture for $d_c = 4$, 6, 8 and 12 and the FB-based architecture in [25] (L-Bubble Check ECN and $d_c = 6$). As shown in Table I, the FIFO-Based architecture increases the hardware efficiency E_n by a factor greater than 6 compared to available data in

the state of the art[1]. For completeness, we should indicate that the work presented in [35] also improves the implementation results for the EMS ECN, based on a pre-fetching technique. Nevertheless, the authors in [35] do not provide synthesis results at the CN level. However, in terms of frequency, our ECN implementation operates at 209 MHz, which is twice the frequency achieved in [35].

VII. Conclusion

This brief was dedicated to the design of an efficient Elementary Check Node architecture for NB-LDPC decoders based on the Extended Min-Sum algorithm. The proposed architecture enhances the hardware efficiency of the check node processor by a factor of 6, compared to previous work. This solution is based on the use of optimized FIFOs at the elementary level and on a Tree architecture at the check node level. Future work will be dedicated to the optimization of the Variable Node architecture and the implementation of the optimized global NB-LDPC decoder.

Acknowledgement

The authors would like to thank Yvan Eustache and Hassan Harb for the valuable discussion and insights on the implementation aspects of this work.

References

[1] M. Davey and D. MacKay, Low-density parity check codes over GF(q), IEEE Communications Letters, vol. 2, no. 6, pp. 165 - 167, June 1998.

[2] S. Pfletschinger, A. Mourad, E. Lopez, D. Declercq and G. Bacci, Performance evaluation of non-binary LDPC codes on wireless channels, in Proceedings of ICT Mobile Summit. Santander, Spain, June 2009.

[3] D. Declercq, M. Colas and G. Gelle, Regular GF(2^q)-LDPC coded modulations for higher order QAM-AWGN channel, in Proc. ISITA. Parma, Italy, Oct. 2004.

[4] F. Guo and L. Hanzo, Low-complexity non-binary LDPC and modulation schemes communicatins over MIMO channels, in IEEE Vehicular Technology Conference (VTC2004). Los Angeles, USA, Sept. 2004.

[5] X. Jiand, Y. Yan, X. Xia and M. Lee, Application of non-binary LDPC codes based on euclidean geometries to MIMO systems, in Int. Conference on wireless communications and signal processing, WCSP09. Nanjing, China, Nov. 2009, pp. 1 - 5.

[6] A. Haroun, C. Abdel Nour, M. Arzel and C. Jego, Low-complexity LDPC-coded iterative MIMO receiver based on belief propagation algorithm for detection, in Turbo Codes and Iterative Information Processing (ISTC), 2014 8th Int. Symp. on, Aug. 2014, pp. 213 - 217.

[7] D. Declercq and M. Fossorier, Decoding algorithms for nonbinary LDPC codes over GF(q), IEEE Trans. on Commun., vol. 55, pp. 633 - 643, April 2007.

[8] L. Conde-Canencia, A. Al-Ghouwayel and E. Boutillon, Complexity comparison of non-binary LDPC decoders, in Proceedings of ICT Mobile Summit. Santander, Spain, June 2009.

[9] H. Wymeersch, H. Steendam and M. Moeneclaey, Log-domain decoding of LDPC codes over GF(q), in Communications, 2004 IEEE International Conference on, vol. 2, June 2004, pp. 772 - 776 Vol.2.

[10] D. MacKay and M. Davey, Evaluation of Gallager codes for short block length and high rate applications, in In Codes, Systems and Graphical Models. Springer-Verlag, 1999, pp. 113 - 190.

[11] H. Song and J. R. Cruz, Reduced-complexity decoding of q-ary LDPC codes for magnetic recording, IEEE Trans. Magn., vol. 39, pp. 1081 - 1087, Mars 2003.

[12] L. Barnault and D. Declercq, Fast decoding algorithm for LDPC over GF(2q), in Proc. Inf. Theory Workshop. Paris, France, Mars 2003, pp. 70 - 73.

[13] V. Savin, Min-Max decoding for non binary LDPC codes, in Proc. IEEE Int. Symp. Information Theory, ISIT2008. Toronto, Canada, July 2008.

[14] A. Voicila, D. Declercq, F. Verdier, M. Fossorier and P. Urard, Low-complexity, low-memory EMS algorithm for non-binary LDPC codes, in Communications, 2007. ICC 07. IEEE International Conference on, 2007, pp. 671 - 676.

[15] M. Fossorier, M. Mihaljevic, and H. Imai, Reduced complexity iterative decoding of LDPC codes based on belief propagation, IEEE Trans. Commun., vol. 47, no. 5, pp. 673 - 680, May 1999.

[16] X. Zhang and F. Cai, Efficient partial-parallel decoder architecture for quasi-cyclic nonbinary LDPC codes, IEEE Trans. CAS-I, vol. 58, no. 2, pp. 402 - 414, February 2011.

[17] X. Zhang and F. Cai, Reduced-complexity decoder architecture for non-binary LDPC codes, IEEE Trans. VLSI, vol. 19, no. 7, pp. 1229 - 1238, July 2011.

[18] F. Cai and X. Zhang, Relaxed Min-Max decoder architectures for nonbinary low-density parity-check codes, IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 11, pp. 2010 - 2023, Nov 2013.

[19] X. Chen and C.-L. Wang, High-throughput efficient non-binary LDPC decoder based on the simplified Min-Sum algorithm, IEEE Trans. on Circuits and Systems I: Regular Papers, vol. 59, no. 11, pp. 2784 - 2794, Nov 2012.

[20] C.-L. Wang, X. Chen, Z. Li, and S. Yang, A simplified Min-Sum decoding algorithm for non-binary LDPC codes, IEEE Trans. on Communications, vol. 61, no. 1, pp. 24 - 32, January 2013.

[21] F. Garcia-Herrero, D. Declercq, and J. Valls, Non-binary LDPC decoder based on symbol flipping with multiple votes, IEEE Communications Letters, vol. 18, no. 5, pp. 749 - 752, May 2014.

[22] A. Ciobanu, S. Hemati, and W. Gross, Adaptive multiset stochastic decoding of non-binary LDPC codes, IEEE Trans. on Signal Processing, vol. 61, no. 16, pp. 4100 - 4113, Aug. 2013.

[23] E. Boutillon and L. Conde-Canencia, Bubble check: a simplified algorithm for elementary check node processing in extended Min-Sum non-binary LDPC decoders, Electronics Letters, vol. 46, no. 9, pp. 633 - 634, 2010.

[24] E. Boutillon and L. Conde-Canencia, Simplified check node processing in nonbinary LDPC decoders, 6th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2010, pp. 201 - 205.

[25] E. Boutillon, L. Conde-Canencia, and A. Al Ghouwayel, Design of a GF(64)-LDPC decoder based on the EMS algorithm, IEEE Trans. on Circuits and Systems I: Regular Papers, vol. 60, no. 10, pp. 2644 - 2656, 2013.

[26] P. Schlafer, N. Wehn, M. Alles, T. Lehnigk-Emden, and E. Boutillon, Syndrome based check node processing of high order NB-LDPC decoders, in Telecommunications (ICT), 22nd International Conference on, April 2015, pp. 156 - 162.

[27] C. Poulliat, M. Fossorier, and D. Declercq, Design of regular (2,d_c)-LDPC codes over GF(q) using their binary images, IEEE Trans. Commun., vol. 56, no. 10, pp. 1626 - 1635, Oct. 2008.

[28] L. Zeng, L. Lan, Y. Tai, S. Song, S. Lin, and K. Abdel-Ghaffar, Constructions of nonbinary quasi-cyclic LDPC codes: A finite field approach, IEEE Trans. Commun., vol. 56, no. 4, pp. 545 - 554, April 2008.

[29] R. Peng and R. Chen, Design of nonbinary quasi-cyclic LDPC cycle codes, in Information Theory Workshop. Tahoe City, USA, Sept. 2007, pp. 13 - 18.

[30] D. Declercq, C. Poulliat, and E. Boutillon, Report on robust and hardware compliant design of non-binary protographs, DAVINCI Deliverable 4.5, avalaible at http://www.ict-davinci-codes.eu, 2009.

[31] A. Venkiah, D. Declercq, and C. Poulliat, Design of cages with a randomized progressive edge growth algorithm, IEEE Commun. Letters, vol. 12(4), pp. 301 - 303, April 2008.

[32] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, On implementation of Min-Sum algorithm and its modifications for decoding LDPC codes, IEEE Trans. Commun., vol. 53, no. 4, pp. 549 - 554, April 2005.

[33] M. Awais, A. Singh, E. Boutillon, and G. Masera, A novel architecture for scalable, high throughput, multi-standard LDPC decoder, in Digital System Design (DSD), 2011 14th Euromicro Conference on, Aug. 2011, pp. 340 - 347.

[34] NB-LDPC codes webpage of the Lab-STICC laboratory, http://wwwlabsticc. univ-ubs.fr/nb_ldpc/, accessed: 2015-09-03.

[35] Y. Tao, Y. Park, and Z. Zhang, High-throughput architecture and implementation of regular (2, dc) nonbinary LDPC decoders, in IEEE Int. Symp. Circuits and Systems (ISCAS). Seoul, Korea, May 2012, pp. 2625 – 2628.

[1]The VHDL code is available under request [34]