A Novel Architecture For Elementary Check Node Processing In Non-Binary LDPC Decoders
Oussama Abassi, Laura Conde-Canencia, Ali Al Ghouwayel, Emmanuel Boutillon

To cite this version:
Oussama Abassi, Laura Conde-Canencia, Ali Al Ghouwayel, Emmanuel Boutillon. A Novel Architecture For Elementary Check Node Processing In Non-Binary LDPC Decoders. IEEE Transactions on Circuits and Systems II: Express Briefs, 2017, 64 (2), pp.136-140. <10.1109/TCSII.2016.2551550>. <hal-01298785>

HAL Id: hal-01298785
https://hal.archives-ouvertes.fr/hal-01298785
Submitted on 6 Apr 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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 $GF(q = 2^m)$ (where $q \gg 2$ is the order of $GF$), i.e. the non-zero entries of their parity-check matrices belong to high-order finite fields. Elements of $GF(q)$ 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 $GF$ 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 $O(d_c \cdot q^2)$ [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 $GF(8)$. 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 $GF$ 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 $O(d_c \cdot q \cdot m)$ per check node. Decoding of $GF(256)$-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
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 [16], trellis-based approach [17] 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 parity-CNs and \( \mathbb{GF} \) symbols are equiprobable, the LLR of \( x_k \) is given by [13]:

\[
\ln \frac{P(x_k | \tilde{x}_k)}{P(x_k | \tilde{x}_k)} = \ln \frac{P(x_k | \tilde{x}_k)}{P(x_k | \tilde{x}_k)} = \ln \frac{P(y_k | x_k)}{P(y_k | \tilde{x}_k)},
\]

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.

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 \( X = (x_1, x_2, \ldots, x_N) \), where \( x_k, k = 1 \ldots N \) is a \( \mathbb{GF}(q) \) symbol represented by \( m = \log_2(q) \) bits as follows: \( x_k = (x_{k,1} \ x_{k,2} \ldots \ x_{k,m}) \). 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:

\[
\sum_{k=1}^{d_c} h_{j,k} x_k = 0
\]

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 \( L^k(x) \) of the \( k \)-th symbol is given by [13]:

\[
L^k(x) = \ln \left( \frac{P(y_k | \tilde{x}_k)}{P(y_k | x_k)} \right)
\]

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

Note that \( L^k(\tilde{x}_k) = 0 \) and, for all \( x \in \mathbb{GF}(q) \), \( L^k(x) \geq 0 \). 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 \(E = [U(1), U(2), \ldots, U(n_m)]\)
where each \(U(i)\) represents a couple \((u_i^L, u_i^T)\) where \(u_i^L, u_i^T \in \mathbb{GF}(q = 2^m)\) and \(u_i^T\) is its associated LLR value. The
messages are sorted in increasing order as follows: \(u_i^T = 0,\)
\(u_i^L \geq 0, i = 2, \ldots, n_m\) and \(\forall i \leq j \Rightarrow u_i^T \leq u_j^T\) [13]
[25]. With the Min-Sum algorithm the values in message \(E = \{(e_i^L, e_i^T), i = 1, \ldots, n_m\}\), correspond to the \(n_m\) minimum
values of the set \(\{u^L_i + u^T_i\}\) such that \(u^T_i \oplus u^L_i = e^T_i\) 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 \(T_S\) (Fig. 1),
where \(T_S(i, j) = U(i) + V(j) \forall (i, j) \in [1, n_m]^2\). This
addition represents the addition of the LLR values \(u_i^L + u_j^T\)
and the \(\mathbb{GF}(q)\) values \(e_i^T \oplus e_j^L\). The most reliable symbols
are considered valid and the followings are tagged as non-valid.

As each path is inherently sorted in increasing order, to
extract the \(n_m\) most reliable symbols in \(T_S\) we only need to
come up with one candidate from each path for a valid path.
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:
\[E = \emptyset\]
for \(i = 1\) to \(4\) do
\[C_i = S_i(1) : j_i = 2\]
end for
Note that \(C_i = (e_i^L, e_i^T)\)
Updates:
for \(l = 1\) to \(n_m\) do
\[k = \text{arg min}\{e_i^L, i = 1 \ldots 4\}\]
\[C_k = S_k(j_k) : j_k = j_k + 1\]
if \(C_k \in E\) then
\[E = E \cup C_k\]
end if
end for

Fig. 1. Matrix \(T_S\) 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 \(T_S\). 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).

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 [33] 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 \( \Delta \) 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 \( \mathrm{GF}(q) \).

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: \( E_n = F / n_{op} / S \) CNs/slice. If we denote by \( \gamma \) 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 \( \mathrm{GF} \) multipliers.

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. 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**


