Parallel CN-VN processing for NB-LDPC decoders

Hassan Harb, Cédric Marchand, Ali Chamas Al Ghouwayel, Laura Conde-Canencia, E. Boutillon

Parallel CN-VN processing for NB-LDPC decoders

Hassan Harb, Cédric Marchand,
Laura Conde-Canencia and Emmanuel Boutillon
Lab-STICC, CNRS UMR 6285
Université Bretagne Sud, Lorient, France
Email: emmanuel.boutillon@univ-ubs.fr

Ali Chamas Al Ghouwayel
Embedded Systems Division
EFREI Paris
Villejuif, France.
Email: ali.ghouwayel@efrei.fr

***Abstract*—In this paper, a novel and innovative approach to implement the check node and variable node phases of the EMS algorithm is proposed. The novelty is not only from the hardware side, but also from the algorithmic point of view. An unusual manner of processing some steps of the check and variable nodes are shown. The performance and implementation results are promising to dig deeper in this work. Compared to its serial counterpart, the synthesis results of the proposed architecture show a factor gain greater than two in terms of area efficiency, with negligible performance loss.**

***Index Terms*—Channel coding, ASIC, NB-LDPC, Min-Sum.**

I. Introduction

Non-Binary (NB) Low-Density Parity-Check (LDPC) codes allow to close the performance gap with the Shannon limit [1] when using small or moderate frame lengths. They are defined on high-order Galois Fields (GF) of order q with $q > 2$ and have been proven to be more robust than convolutional turbo-codes and binary LDPC codes [2]. NB-LDPC codes are already adopted in two space communication standards [3], [4] but the high complexity of NB-LDPC decoders remains a major drawback. Fortunately, prior art has already shown the feasibility of NB-LDPC decoder implementation. In particular, Trellis-Extended Min Sum (T-EMS) based algorithms allow very high decoding throughput (greater than 10 Gbit/s) [5], [6]. In this paper, we show that it is also possible to reach decoding throughput of the same order with Extended Min-Sum (EMS) based algorithms. This is achieved with massive parallelization of a joint Check Node (CN)-Variable Node (VN) unit. Adopting the parallel approach in hardware design arises some difficulties that should be faced wisely. In this context, the main two contributions in this work are: 1) merging the CN and the VN blocks in a single block to avoid the costly sorting of the check-to-variable messages; 2) a method to determine, without any sorting, the default value associated with the check-to-variable messages. These two main contributions led to a significant saving in terms of hardware resources.

The paper is organized as follows: Section II shows the notations, the NB-LDPC codes and the EMS algorithm. Section III presents the proposed parallel pipelined CN-VN unit and its architecture. The performance and implementation results are shown in section IV and V, respectively. Finally, the conclusion and perspectives are presented in section VI.

II. NB-LDPC codes and EMS algorithm

This section introduces NB-LDPC codes, describes the calculation of the intrinsic messages and the principles of the EMS algorithm.

A. NB-LDPC codes defined over Galois Fields

A NB-LDPC code is a linear block code defined on a very sparse parity-check matrix H whose non-zero elements belong to a finite field GF(q), where $q > 2$. The elements of GF(q) are $\{0, \alpha^0, \alpha^1, \ldots, \alpha^{q-2}\}$. The dimension of matrix H is $M \times N$, where M is the number of parity-CNs and N is the number of VNs (i.e., the number of GF(q) symbols in a codeword). A codeword is denoted by $\mathbf{C} = (x_0, x_1, \ldots, x_{N-1})$, where x_k, $k = 0, \ldots, N-1$, is a GF(q) symbol represented by $m = \log_2(q)$ bits as $x_k = (x_{k,0}\ x_{k,1} \ldots x_{k,m-1})$. The construction of regular (d_v, d_c) NB-LDPC codes is expressed as a set of M parity-check equations C_j, $j = 0, 1, \ldots, M-1$, over GF(q), where the j^{th} parity check equation C_j is given as

$$C_j : \sum_{i=0}^{d_c-1} h_{j,k(j,i)} x_{k(j,i)} = 0, \tag{1}$$

where $\{k(j,i)\}_{i=0,1,\ldots,d_c-1}$ is the set of the d_c non-null positions in the j^{th} row of the matrix H and $h_{j,k(j,i)}$ its associated GF values. Each variable is connected to exactly d_v non-null elements in a column.

B. Intrinsic messages

Let $Y = (y_0, y_1, \ldots, y_{m-1})$ be the LLR intrinsic vector associated to a GF symbol x. Each value y_p is defined as: $y_p = \log\left(\frac{\mathrm{P}(x_p = 0/r_p)}{\mathrm{P}(x_p = 1/r_p)}\right) = \frac{2r_p}{\sigma^2}$, where $r_p = \mathrm{B}(x_p) + w_p$, $p = 0, \ldots, m-1$, is the received sample. In which, $\mathrm{B}(x_p) = (-1)^{x_p}$ is the Binary Phase-Shift Keying (BPSK) modulation and w_p is a realization of a Gaussian noise of variance σ^2. Considering the hypothesis that all the symbols in the GF(q) alphabet have equal probability, the expression of the LLR $I^+(x)$ of a symbol x knowing y is expressed as:

$$I^+(x) = \sum_{p=0}^{m-1} |y_p| \Delta(x_p, \bar{x}_p), \tag{2}$$

where $\Delta(x_p, \bar{x}_p) = 0$ if $x_p = \bar{x}_p$ and $\Delta(x_p, \bar{x}_p) = 1$ otherwise. Note that, by definition, $I^+(\bar{x}) = 0$ is the smallest LLR value. It will be denoted as $I[0] = (I^+[0], I^{\oplus}[0])$, with

the LLR $I^+[0] = I^+(\bar{x}) = 0$ and the associated GF value $I^\oplus[0] = \bar{x}$. Let $\Pi = \{\pi(0), \pi(1), \pi(2)\}$ be respectively the index of the smallest, the second smallest and the third smallest magnitude for $|y_p|$ values, $p = 0, 1, \ldots, m-1$. The set Π is retained in order to regenerate the $n_{m_{in}}$ intrinsic candidates as proposed in [7].

In the hardware architecture, we will consider $n_{m_{in}} = 4$. Thus, regenerating the 4 intrinsic candidates $I = (I^\oplus[0], I[1], I[2], I[3])$ is performed as: 1) $I^\oplus[0]$ is simply the hard decision of the observed Y; 2) $I^+[1] = |y_{\pi(0)}|$ and $I^\oplus[1]$ is equal to $I^\oplus[0]$ except at position $\pi(0)$ where its associated bit is flipped; 3) $I^+[2] = |y_{\pi(1)}|$ and $I^\oplus[2]$ is equal to $I^\oplus[0]$ except at position $\pi(1)$ where its associated bit is flipped and 4) $I^+[3] = \min(|y_{\pi(0)}| + |y_{\pi(1)}|, |y_{\pi(2)}|)$ and $I^\oplus[3]$ is equal to $I^\oplus[0]$ except at position $\pi(0)$ and $\pi(1)$ if $I^+[3] = |y_{\pi(0)}| + |y_{\pi(1)}|$, otherwise, if $I^+[3] = |y_{\pi(2)}|$, $I^\oplus[3]$ is equal to $I^\oplus[0]$ except at position $\pi(3)$. The intrinsic vector I associated to a given received symbol Y is thus composed as: $I = (I^\oplus[0], I[1], I[2], I[3])$. Associated to symbol Y, the pre-processed vector $\tilde{I}$ is defined as: $\tilde{I} = (|y_0|, |y_1|, \ldots, |y_5|, \Pi, I^\oplus[0])$, and allows to easily reconstruct I with few hardware resources. It also allows to compute $I^+[\alpha]$ for any $\alpha \in$ GF(64).

C. CN and VN EMS-based updates

We consider the EMS algorithm [8] with the following characteristics: VN degree $d_v = 2$, CN input messages truncated to a size $n_{m_{in}} \ll q$ and CN output messages to $n_{m_{out}} \ll q$. This leads to computation and storage reduction without necessarily performance loss [9] [10]. In this paragraph, we focus only on the CN and VN updates.

For the CN and VN descriptions, let:

- $I = \{I[0], \ldots, I[n_{m_{in}} - 1]\}$ be the intrinsic LLR vector (see previous section),
- $\{h_0, \ldots, h_{d_c-1}\}$ the d_c non-zero elements in H associated to a CN,
- $\{U_0, \ldots, U_{d_c-1}\}$ the group of messages sent to a CN from the d_c connected VNs, and
- $\{V_0, \ldots, V_{d_c-1}\}$ the group of messages sent to d_c VNs from a CN.

Each U_i message can be written as $U_i = \{U_i[0], \ldots, U_i[n_{m_{in}} - 1]\}$. Each element $U_i[j]$ corresponds to a couple $U_i[j] = (U_i^+[j], U_i^\oplus[j])$ where $U_i^\oplus[j]$ is the GF(q) symbol and $U_i^+[j]$ its corresponding LLR value, $i = 0, \ldots, d_c - 1$ and $j = 0, \ldots, n_{m_{in}} - 1$. The elements in these messages are sorted in ascending order of LLR values.

1) CN update: The CN update is processed as

$$V_i^+(x) = \min\left\{ \sum_{i'=0, i' \neq i}^{d_c-1} U_{i'}^+[j_{i'}] \Big| \bigoplus_{i'=0, i' \neq i}^{d_c-1} U_{i'}^\oplus[j_{i'}] = x \right\}, \quad (3)$$

where $j_{i'} \in \{0, 1, \ldots, n_{m_{in}} - 1\}$ for $i' = 0, 1, \ldots, d_c - 1$, $i' \neq i$ and $\oplus$ refers to GF addition (i.e., XOR gate).

The final stage is to partially sort in increasing order the set of values of $V_i^+(x)$ indexed by $x \in$ GF(q) to obtain an ordered set $V_i^\oplus = \{x_0, x_1, \ldots, x_{n_{m_{out}}-1}\}$ that verifies

$$\forall (j,k),\ j < k < n_{m_{out}} \Rightarrow V_i^+(x_j) \leq V_i^+(x_k),$$

and

$$\forall x \in \text{GF}(q),\ x \notin V_i^\oplus \Rightarrow V_i^+(x_{n_{m_{out}}-1}) \leq V_i^+(x).$$

The i^{th} output message is thus given as $V_i = \{V_i[0], V_i[1], \ldots, V_i[n_{m_{out}-1}]\}$, where $V_i[j] = (V_i^+[j] = V_i^+(x_j), V_i^\oplus[j] = x_j)$, $j = 0, 1, \ldots n_{m_{out}-1}$.

In the state of the art, the GF values outside V_i are associated with a default LLR value D_i, with $D_i = V_i^+[n_{m_{out}} - 1] + O$, i.e., the default value is equal to the highest LLR value of message V added with O, a positive offset value (see [10] for more details on the definition of the offset value). In section III.A, we propose a new method to determine the default value D_i to facilitate the hardware implementation of the full parallel CN architecture.

2) VN update: After processing CN a, the required inputs of a VN are: the intrinsic vector I of size $n_{m_{in}}$, the m values of Y to be able to compute $I^+(x)$ for any $x \in$ GF(q) using (2), the received message V^a of size $n_{m_{out}}$ coming from CN a, the default value D^a associated to message V^a and the message U^a sent to the CN a. The two outputs of the VN are the current message U^b of size $n_{m_{in}}$ to be sent to CN b and the VN decision $\hat{x}$ obtained by combining V^a and U^a.

The first step of the VN processing is the addition of the intrinsic LLR values on the incoming V^a message to generate the message $\bar{V}^a$ defined as:

$$\bar{V}^{a,+}[j] = V^{a,+}[j] + I^+(V^{a,\oplus}[j]),\ j = 0, 1, \ldots n_{m_{out}} - 1. \quad (4)$$

Since $\bar{V}^a$ associates LLR values for only a subset of GF values, a second message $\bar{I}$ is generated in parallel as:

$$\bar{I}^+[j] = I^+[j] + D^{a,+},\ j = 0, 1, \ldots n_{m_{in}} - 1. \quad (5)$$

Then, the $n_{m_{in}}$ smallest values of set $\bar{V}^a \cup \bar{I}$ in terms of LLR value are extracted along with their associated GF symbols to generate the vector messages $\bar{U}^b$. Note that by construction, $\bar{V}^{a,\oplus} \cap \bar{I}^\oplus$ may not be empty. In that case, the corresponding LLR element in $\bar{I}$ is saturated so that $\bar{U}^b$ contains the first $n_{m_{in}}$ smallest LLR values with distinct GF values. The last step to generate the final message is the normalization process that keeps the first LLR of the message equal to zero, i.e.,

$$U^{b,+}[j] = \bar{U}^{b,+}[j] - \bar{U}^{b,+}[0],$$
$$U^{b,\oplus}[j] = \bar{U}^{b,\oplus}[j],\ j = 0, \ldots, n_{m_{out}} - 1. \quad (6)$$

III. Pipelined CN-VN unit

This section presents a pipelined architecture able to perform a CN of degree $d_c = 12$ and its 12 associated VNs every Clock Cycle (CC). Based on the Hybrid (H)-CN architecture [11], an optimized version of the CN architecture for the 5/6-rate codes ($d_v = 2$) is described and merged to the VN to form the innovative CN-VN unit.

A. Principle of the hybrid CN architecture

The H-CN presented in [11] consists in three main functions: presorting, ECN processing and decorrelation using the Valid Syndrome Vector (VSV).

1) Presorting: The preliminary step of CN processing is the presorting block that leads to a significant reduction of the CN computations. In [12] the authors proposed the sorting of the CN input vectors based on the LLR value of the second GF element. This sorting polarizes the reliability of the input vectors and classifies them into two sets: high reliability and low reliability. Designing the size of the two sets is performed through a statistical study that determines the percentage of using an input message. Then, after selecting the most probable ones, the Monte Carlo (MC) simulation ensures the robustness of the selected ones. In addition, some messages can be pruned by hand after verifying that they do not weaken the performance. Consequently, presorting *helps* the CN to concentrate its processing effort on low-reliability vector messages.

Fig. 1 shows the architecture of the proposed parallel pipelined CN-VN. In this paragraph, we focus on the Presorting, Permutation (Perm.) Ψ and Perm. Ψ^{-1} blocks. $U^a = \{U_0^a, \ldots, U_{11}^a\}$ is of size $d_c = 12$, each $U_i^a = \{U_i^a[0], \ldots, U_i^a[3]\}$ is of length $n_{m,in} = 4$, $i = 0, \ldots, d_c - 1$. The Presorting receives $U^{a+}[1] = \{U_0^{a+}[1], \ldots, U_{11}^{a+}[1]\}$ to generate the $\Psi = \{\psi[0], \ldots, \psi[11]\}$ set. $\psi[0]$ is the index of the first minimum value in $U^{a+}[1]$, $\psi[1]$ is the index of the second minimum value in $U^{a+}[1]$, ..., etc. The Perm. Ψ block permutes U^a according to Ψ so the second LLR values of the new vector U'^a are sorted. Consequently, the high reliability messages are concentrated in one region so the CN concentrates its processing on them while the rest are suppressed. Next paragraph shows how the number of processed input symbols is reduced from $d_c \times n_{m,in} = 12 \times 4 = 48$ down to 27 after performing Perm. Ψ. After the CN and VN processing, the Perm. Ψ^{-1} block reorders the outputs to their original order. More details on the presorting technique are presented in [13], [12] and [14].

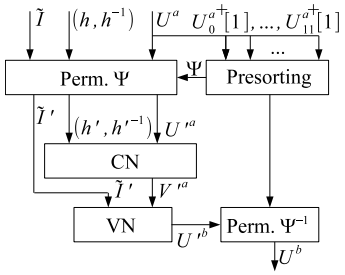

Fig. 1. CN-VN architecture.

2) ECN: For the implementation of the CN processing, equation (3) is implemented in a simplified way using the H-CN architecture defined in [11]. The whole CN architecture is characterized graphically in Fig. 2.a).

The number of elements of U_i', $i = 0, \ldots, 11$, that enter the CN is indicated by the number of circles below it. For example, only $U_0'[0] = (U_0'^{+}[0], U_0'^{\oplus}[0])$ out of $n_{m_{in}} = 4$ elements of message U_0' enters the CN and the first 3 elements $\{U_8'[0], U_8'[1], U_8'[2]\}$ of message U_8' enter the CN. This reduction of the number of messages, being fed to the CN, is achieved thanks to the presorting process. The line of multipliers on the top indicates that each GF value of U_k' is multiplied by the GF coefficient h_k'. The outputs of the multipliers enter a datapath composed of a network of ECNs. Each ECN performs the bubble check algorithm. Let us give the key to understand the processing performed by the generic ECN given in Fig. 2.b). An ECN receives two input vectors A and B of size n_a and n_b given by the number of circles (or bubbles) respectively in the first column and in the first row ($n_a = 4$ and $n_b = 3$ in Fig. 2.b)). It generates an output message C of size n_c. Note that in Fig.2.a), the output size is implicitly defined as the number of inputs (i.e., number of vertical bubbles) of the next ECN. A circle in position (t_0, t_1), $t_0 = 0, \ldots, n_a - 1$ and $t_1 = 0, \ldots, n_b - 1$, means that $U_a[t_0]$ and $U_b[t_1]$ are added to generate a couple $(U_a^+[t_0] + U_b^+[t_1], U_a^{\oplus}[t_0] \bigoplus U_b^{\oplus}[t_1])$. The n_c bubbles of minimum LLR sorted in increasing order constitute the output vector of the ECN. The VSV is appended with a boolean value that indicates whether $U_c[t_2]$, $t_2 = 0, \ldots, n_c - 1$, has been generated with $U_b[0]$ or with $U_b[t_1]$, $t_1 > 0$. Bubbles in dark color append a false Boolean value to the corresponding position in the VSV vector.

The three last ECNs (ECN11, ECN12 and ECN13) are slightly simplified compared to the other ECNs because all the bubbles are output without any sorting. In fact, one of the main ideas in the architecture is to save hardware complexity by postponing the sorting operation in the VN processing. Since no sorting is performed, the default value D of the check to variable message cannot be determined. Thus, we propose to empirically set it to the LLR of a fixed bubble position indicated by D_g, D_{10} and D_{11} in ECN11, ECN12 and ECN13, respectively. Note that the size of the output message S of ECN11 is $n_S = 20$ while the size of the message of ECN12 and ECN13 is $n_{FB} = 16$.

Any element of S is given by the summation of all the incoming messages. A decorrelation process is thus required to satisfy the parity check definition (see equation (3)). It suppresses the GF symbol $U_i'^{\oplus}[0]$ from $S^{\oplus}[t]$ if $U_i'^{\oplus}[0]$ contributes in computing $S^{\oplus}[t]$ to generate the i^{th} output thanks to the GF adder (addition and subtraction are equivalent in the GF domain), where $i = 0, \ldots, 11$, $t = 0, \ldots, n - 1$ and $n \in \{n_S, n_{FB}\}$. Otherwise, if $U_i'^{\oplus}[0]$ does not contribute in computing $S^{\oplus}[t]$, the Decorrelation Block (DB) associated to U_i' saturates the LLR value $S^{+}[t]$. This is done thanks to the VSV vector that is being checked by the DB. Subtracting only $U_i'^{\oplus}[0]$ from S is justified by the fact that when subtracting $U_i'^{\oplus}[j]$ from S, $j = 1, \ldots, 3$, leads to the same result if the position of messages taken from the neighborhood input vectors is the same. The final multiplication is applied on the GF value to compute the output message V_i'. In more details,

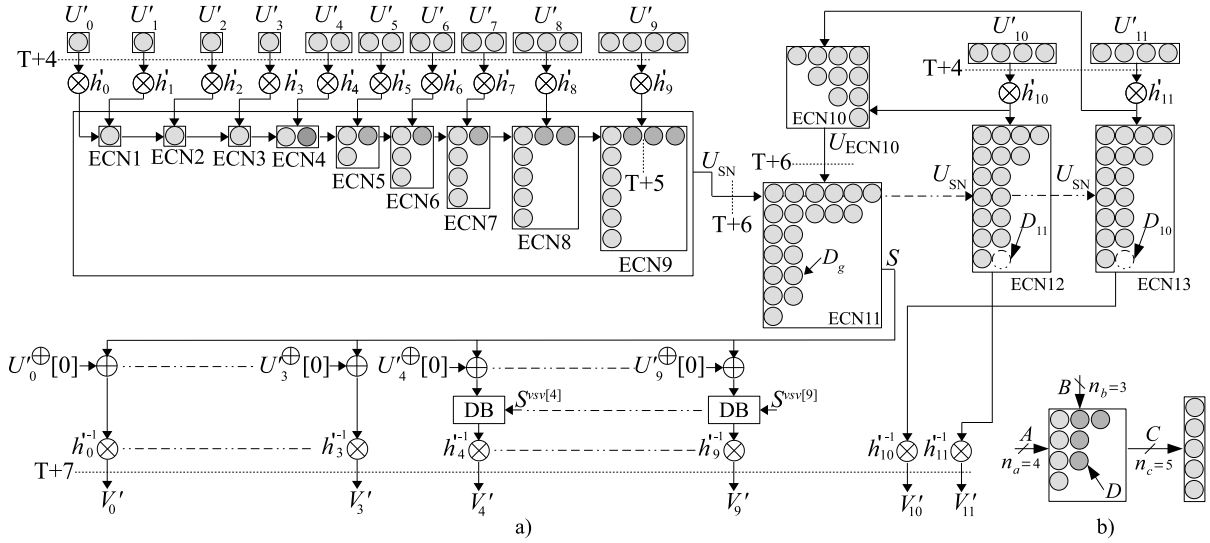

Fig. 2. a) High level CN architecture and b) ECN in case of $n_a = 4$, $n_b = 3$ and $n_c = 5$.

$V_i'^{\oplus}[t] = (S^{\oplus}[t] - U_i'^{\oplus}[0]).h_i'^{-1}$ and $V_i'^{+}[j] = S^{+}[t]$. These operations are performed in 1 CC to have 3 CC latency in total to perform the CN.

B. CN-VN Processing

Let us focus again in the CN-VN architecture shown in Fig. 1. The inputs of CN-VN are the intrinsic information $\tilde{I} = \{I, \tilde{I}_e\}$ ($I = \{I_0, \ldots, I_{11}\}$ and $\tilde{I}_e = \{\{I_0[0], \ldots, I_{11}[0]\}, \{|Y_0|, \ldots, |Y_{11}|\}, \{\Pi_0, \ldots, \Pi_{11}\}\}$), the GF coefficients and their inverse $(h, h^{-1}) = \{(h_0, h_0^{-1}), \ldots, (h_{11}, h_{11}^{-1})\}$ and the extrinsic messages $U^a = \{U_0^a, \ldots, U_{11}^a\}$. The first stage of the proposed joint CN-VN unit starts in a similar way than the hybrid architecture, except that all messages are received in parallel: first, all the inputs are permuted using the indexes Ψ obtained by the presorting block, then the H-CN is performed.

The presorting receives $\{U_0^+[1], \ldots, U_{11}^+[1]\}$ to generate the indexes $\Psi = \{\psi[0], \ldots, \psi[11]\}$ for $d_c = 12$ based on the presorting principle explained in section III.A, thus $U_{\psi[0]}^+[1] \leq U_{\psi[1]}^+[1] \leq \cdots \leq U_{\psi[11]}^+[1]$. The architecture of the parallel pipelined presorting block is inspired from [15].

Based on the Ψ values, the inputs of the CN-VN are switched using the permutation (Perm. Ψ) block. After that, the CN is performed. The specification of the CN was defined in section III.A.

Fig. 3 illustrates the parallel architecture of the VN. The eLLR block generates the intrinsic LLR value $I^+(V_i'^{a\oplus})$, thanks to $|Y_i'|$ and $I_i'^{\oplus}[0]$, then the value is added to $V_i'^{a+}$ to generate $\bar{V}_i'^{a+}$. The Regeneration of Intrinsic Candidates (RIC) block regenerates the intrinsic candidates $\{I_i'[0], \ldots, I_i'[3]\}$. Then, the offset value $D^a \in \{D_g, D_{10}, D_{11}\}$ associated to V_i' is added on $\{I_i'[0], \ldots, I_i'[3]\}$ to generate $\bar{I}_i^+$. In

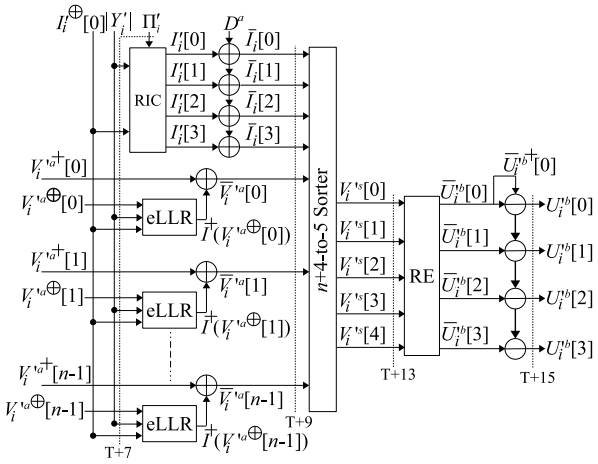

Fig. 3. VN architecture

section II.C, we showed that the output is generated by detecting the most reliable not redundant GF symbols from $\{V_i'^a[0], \ldots, V_i'^a[n-1], I_i'[0], \ldots, I_i'[3]\}$. To reduce the complexity, the sorting and the Redundant Elimination (RE) operations are separated. First, the vector $V_i'^s$ of $n_{m_{in}} + n_\delta$ couples having the lowest LLR values are detected from $\{V_i'^a[0], \ldots, V_i'^a[n-1], I_i'[0], \ldots, I_i'[3]\}$, then the outputs are generated by detecting, from $V_i'^s$, the $n_{m_{in}}$ couples without redundant GF symbols. In this work $n_\delta = 1$. Thus, the $n+4$-to-5 Sorter block, $n \in \{n_{FB}, n_s\} = \{16, 20\}$, generates $V_i'^s$ that is having the $4+1=5$ couples of lowest LLR values among $\{V_i'^a[0], \ldots, V_i'^a[n-1], I_i'[0], \ldots, I_i'[3]\}$. The RE block generates $U_i'^b$ by detecting the 4 couples from V'^s that are different in terms of GF values and having lowest LLR values. Finally, the normalization of $\bar{U}^{b,+}$ (6) is performed. Eight pipeline stages are inserted in the VN architecture.

The last operation in CN-VN is the inverse permutation performed using Ψ^{-1} to reorder $U'^b = \{U'^b_0, \ldots, U'^b_{11}\}$ and $\hat{X}' = \{\hat{x}'_0, \ldots, \hat{x}'_{11}\}$ to their original order. This block is having one pipeline stage and hence 16 CCs and 10 CCs are the total latency to generate U^b and $\hat{X}$ respectively.

IV. Simulation Results

We consider MC simulations under the AWGN channel, BPSK modulation and 6 bits-LLR values quantization. In order to compensate the performance degradation due to the elimination of some bubbles, the maximum number of iterations in the proposed decoder is increased to 30 with sub-layered decoding schedule, while the results of the serial H-CN and T-EMS are obtained considering 10 iterations and full-layered decoding schedule. Fig. 4 shows simulation results for the H-CN EMS decoder [11] and the proposed CN-VN approach with the following parameters: $K = 120$, $N = 144$ GF(64) symbols and CR $= 5/6$.

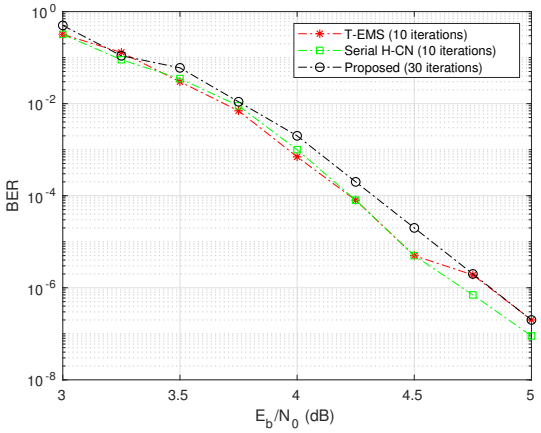

Fig. 4. FER performance for a (144, 120) NB-LDPC code over GF(64): Proposed decoder vs FB CN-based decoder and (864, 720) B-LDPC code over GF(2) SP decoder.

Compared to H-CN, the proposed decoder presents a performance loss of only 0.1 dB. This amount of degradation comes as a cost of the low computational operations achieved by eliminating some bubbles. The same loss that appears compared to T-EMS [16] when $E_b/N_0 \leq 4.5$ dB, it starts to disappear when $E_b/N_0 > 4.5$ dB. Although the proposed decoder is implemented with a maximum number of iterations equal to 30, it is the average number of iterations that will be involved to determine the average decoding throughput. This will be discussed in more details in the next section

V. Implementation Results

This section discusses the throughput calculation and the post-synthesis results on 28-nm FDSOI technologies. Since the decoding throughput is highly dominated by the average number of iterations $n_{av,it}$, we only focused on $E_b/N_0 = 4.5$ dB. At this point, Table I shows the number of processed CNs per second $T_{CN} = \dfrac{F_{clk} \times 10^6}{L_{CN} \times n_{av,it}}$ (CNs/s) and the hardware efficiency $E_{CN} = T_{CN}/C$ (CNs/s/Mgate) of the CN-VN block in this work and [11] ($d_c = 12$), where L_{CN} is the latency of the CN-VN and F_{clk} the clock frequency of the design expressed in MHz. Synthesis results give a maximum clock frequency $F_{clk} = 900$ MHz with a latency $L_{CN} = 1$. For the serial hybrid architecture, a new synthesis of the hybrid architecture performed by the authors increased the maximum clock frequency up to 1000 MHz. The latency of the serial hybrid architecture is equal to $L_{CN} = 41$. The parallel CN-VN block provides much higher T_{CN} than the serial CN-VN presented in [11]. Even though the area consumption of this serial version is about 10 times lower than the parallel one, the hardware efficiency of the latter is higher as shown in Table I.

TABLE I
COMPARISON OF THE PARALLEL AND SERIAL CN-VN BLOCK (ASICs).

$E_b/N_0 = 4.5$ dB	CN-VN this work	CN-VN [11]
$n_{av,it}$	1.5	1.26
C (NAND)	0.38 M	0.032 M
$T_{CN} \times 10^6$ (CNs/s)	600	19.4
$E_{CN} \times 10^6$ (CNs/s/Mgate)	1579	607

VI. Conclusion and Perspectives

This work presented a parallel-pipelined version of the serial H-CN. The EMS algorithm is rethought and implemented in a novel manner enabling an important complexity reduction along with an increase in the throughput rate. The H-CN and the T-EMS algorithms with 10 iterations and layered decoding schedule are considered as references to compare performance. Knowing that the average number of iterations will be finally impacting the throughout rate, the maximum number of iterations used in the proposed decoder has been increased from 10 to 30 to allow the reduction of the required computation resources in the EMS algorithm and thus the global decoder area. In more details, to achieve the goal of reducing the area consumption, some computations are suppressed by pruning some extra bubbles. This procedure leads to performance loss, compensated by increasing the maximum number of iterations. MC simulation showed a low performance loss, and ASIC implementation showed that this work more than doubles the hardware efficiency of the serial H-CN. Even though the complexity and simulation results evaluation is performed on the pre-mentioned code, the proposed CN-VN can be used for any code of $d_c = 12$ and can be extended for codes that are having $d_c > 12$ without the need of modifying the first 12 ECNs. On the other hand, for codes of $d_c < 12$, some additional bubbles should be included in the decoding process in order to maintain a good level of performance. In this perspective, future work will be designing a flexible parallel pipelined H-CN that can be adjusted for different NB-LDPC codes.

REFERENCES

[1] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF(q)," *IEEE Communications Letters*, vol. 2, no. 6, pp. 159–166, 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] "Beidou navigation satellite system, signal in space, interface control document, open service signals b1c (version 1.0)," *China Satellite Navigation Office*, 2017.

[4] "Short block length ldpc codes for tc synchronization and channel coding," *Consulative Committee for Space Data Systems (CCSDS)*, 2015.

[5] J. O. Lacruz, F. Garca-Herrero, M. J. Canet, and J. Valls, "Reduced-complexity nonbinary ldpc decoder for high-order galois fields based on trellis minmax algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 8, pp. 2643–2653, Aug 2016.

[6] H. Pham Thi and H. Lee, "Basic-set trellis minmax decoder architecture for nonbinary ldpc codes with high-order galois fields," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 496–507, March 2018.

[7] H. Harb, A. C. Al Ghouwayel, and E. Boutillon, "Parallel generation of most reliable LLRs of a non-binary symbol," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1761–1764, Oct 2019.

[8] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633–643, April 2007.

[9] ——, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Comm.*, vol. 55, no. 4, pp. 633–643, April 2007.

[10] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low complexity, low memory EMS algorithm for non-binary LDPC codes," in *IEEE Intern. Conf. on Commun., ICC'2007*. Glasgow, England, June 2007.

[11] C. Marchand, E. Boutillon, H. Harb, L. Conde-Canencia, and A. Al Ghouwayel, "Hybrid check node architectures for NB-LDPC decoders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 869–880, Feb 2019.

[12] H. Harb, C. Marchand, A. Ghouwayel, A. Conde-Canencia, L., and E. Boutillon, "Pre-sorted forward-backward NB-LDPC check node architecture," in *SIPS*, 2016.

[13] C. Marchand and E. Boutillon, "NB-LDPC check node with pre-sorted input," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sept 2016, pp. 196–200.

[14] C. Lin, S. Tu, C. Chen, H. Chang, and C. Lee, "An efficient decoder architecture for nonbinary LDPC codes with extended min-sum algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 9, Sept 2016.

[15] M. J. S. Amin Farmahini-Farahani, Henry J. Duwe III and K. Compton, "Modular design of high-throughput, low-latency sorting units," *IEEE TRANSACTIONS ON COMPUTERS*, vol. 62, no. 7, pp. 1389–1402, July 2013.

[16] E. Li, K. Gunnam, and D. Declercq, "Trellis based extended min-sum for decoding nonbinary ldpc codes," in *2011 8th International Symposium on Wireless Communication Systems*, 2011, pp. 46–50.