Simplified Trellis Min–Max Decoder Architecture for Nonbinary Low-Density Parity-Check Codes

Jesús O. Lacruz, Francisco García-Herrero, David Declercq, *Senior Member, IEEE*, and Javier Valls, *Member, IEEE*

***Abstract*— Nonbinary low-density parity-check (NB-LDPC) codes have become an efficient alternative to their binary counterparts in different scenarios, such as moderate codeword lengths, high-order modulations, and burst error correction. Unfortunately, the complexity of NB-LDPC decoders is still too high for practical applications, especially for the check node (CN) processing, which limits the maximum achievable throughput. Although a great effort has been made in the recent literature to overcome this disadvantage, the proposed decoders are still not ready for high-speed implementations for high-order fields. In this paper, a simplified trellis min–max algorithm is proposed, where the CN messages are computed in a parallel way using only the most reliable information. The proposed CN algorithm is implemented using a horizontal layered schedule. The overall decoder architecture has been implemented in a 90-nm CMOS process for a (N = 837 and K = 726) NB-LDPC code over GF(32), achieving a throughput of 660 Mb/s at nine iterations based on postlayout results. This decoder increases hardware efficiency compared with the existing recent solutions for the same code.**

***Index Terms*— Layered decoder, message passing algorithm, nonbinary low-density parity-check (NB-LDPC), trellis min–max (TMM).**

I. Introduction

NONBINARY low-density parity-check (NB-LDPC) codes have become an interesting alternative to their binary counterparts for applications requiring small to moderate codeword lengths and large rates. The main limitation of a wider use of NB-LDPC codes is that the complexity of the decoder limits the maximum throughput that can be achieved with their hardware implementations.

NB-LDPC are lineal block codes characterized by a sparse parity check matrix $\mathbf{H}$ with M rows and N columns. Each nonzero element $h_{m,n}$ of $\mathbf{H}$ belongs to the Galois field GF($q = 2^p$). In this paper, we only consider regular NB-LDPC codes with constant row weight d_c and column weight d_v. NB-LDPC codes can also be characterized by a bipartite graph called Tanner graph [1], where two types of nodes can be differentiated, the ones representing the rows of the parity check matrix called check nodes (CNs) and the ones that represent the columns in $\mathbf{H}$, called variable nodes (VNs). Decoding algorithms for NB-LDPC codes use iterative message exchange between the CNs and the VNs and vice versa to estimate the most reliable codeword from the noisy received sequence.

Different decoding algorithms have been proposed since the Q-ary sum product algorithm (QSPA) [2]. The complexity of QSPA is too large to be suitable for hardware implementations, and several approaches, such as fast Fourier transform-SPA [3], log-SPA, and max-log-SPA [4], were proposed to overcome its limitations. These solutions reduce the complexity of the CN processing equations without introducing any performance loss. In [5], an approximation of QSPA called extended min-sum (EMS) has been proposed, where the complexity of the CN is reduced considerably involving only comparisons and additions. Later, the min–max algorithm was proposed [6], which uses comparisons with compute the maximum reliability values instead of additions, unlike the EMS algorithm. This new solution helps preventing the growth of the data length of the decoder without introducing any performance loss with respect to the EMS algorithm.

On the other hand, EMS and min–max algorithms still suffer from a bottleneck at the CN caused by the use of forward-backward metrics for the extraction of check to variable messages. In [7], the trellis EMS (T-EMS) has been introduced, for computing the combination of the most reliable messages while avoiding the use of forward-backward metrics and, therefore, increasing the degree of parallelism. The decoder presented in [7] was improved in [8] where an extra column is added to the original trellis with the purpose of generating in a parallel way the check to variable messages. This algorithm allows to derive higher throughput architectures. The main drawback of the approach presented in [8] is that it requires a lot of area in its proposed structure, reducing the overall efficiency of the decoder.

To further improve the T-EMS efficiency, we propose in this paper, a simplification of this algorithm, by building the extra column of the trellis and generating the output messages of the CN using only the most reliable information. The extra column information and two most reliable messages are computed to generate the check to variable messages in an efficient way improving both area and latency of the decoder. Additionally, for each configuration path, we define the path reliability

Manuscript received December 6, 2013; revised March 21, 2014 and July 3, 2014; accepted July 27, 2014. This work was supported in part by the Spanish Ministerio de Ciencia e Innovación under Grant TEC2011-27916; in part by the Universitat Politècnica de València, Gandia, Spain, under Grant PAID-06-2012-SP20120625; and in part by the Institut Universitaire de France, Rennes, France. The work of F. García-Herrero was supported in part by the Spanish Ministerio de Educación under Grant AP2010-5178 and in part by the Institute Universitaire de France.

J. O. Lacruz is with the Department of Electrical Engineering, Universidad de Los Andes, Mérida 5101, Venezuela (e-mail: jlacruz@ula.ve).

F. García-Herrero and J. Valls are with the Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politècnica de Valéncia, Gandia 46730, Spain (e-mail: fragarh2@epsg.upv.es; jvalls@eln.upv.es).

D. Declercq is with the ETIS Laboratory, University of Cergy-Pontoise, Cergy-Pontoise F-95000, France (e-mail: david.declercq@ensea.fr).

Digital Object Identifier 10.1109/TVLSI.2014.2344113

using the maximum value instead of aggregating the symbol reliabilities. For this reason, we named our algorithm trellis min–max (TMM). The simplified CN algorithm is implemented using a horizontal layered scheduling, which establishes a compromise between overall area of the decoder and latency.

To show the efficiency of the proposed NB-LDPC decoder on codes over high-order Galois fields, a ($N = 837$ and $K = 726$) NB-LDPC code over GF(32) has been selected. This code has been used in many preceding papers and serves then as a benchmark for comparing different implementations of NB-LDPC decoders. To the best of our knowledge, the proposed architecture based on TMM is more area-throughput efficient than the existing proposals found in [8] and [9], for the same code. Moreover, the proposed design has lower latency and higher throughput than any proposed NB-LDPC decoder.

The rest of this paper is organized as follows. In Section II, we recall the principles of the T-EMS algorithm. The proposed TMM algorithm and its CN architecture are presented in Section III. Section IV describes the overall layered decoder architecture and the synthesis and postlayout results. Section V includes the comparisons with others proposed decoders and conclusion is outlined in Section VI.

II. T-EMS Algorithm

An NB-LDPC code is defined by its parity check matrix $\mathbf{H}$ with M rows and N columns. Each nonzero element $h_{m,n}$ of $\mathbf{H}$ belongs to a Galois field GF($q = 2^p$), which is often chosen as a field with characteristic 2, i.e., when the field order is a power of 2 [5], [10]. An NB-LDPC code can be either regular that is with constant row weight d_c and column weight d_v, or irregular, when the row and/or column weights differ. For ease of presentation of the equations and of the algorithm, we consider in this paper constant row and column weights, but our algorithm can be trivially generalized to irregular LDPC codes.

Let $\mathcal{N}(m)$ ($\mathcal{M}(n)$) be the set of VNs (CNs) connected to a CN (VN) $m(n)$. Let $Q_{m,n}(a)$ and $R_{m,n}(a)$ be the messages from VN to CN and from CN to VN, respectively. For a symbol value $a \in$ GF(q), $Q_{m,n}(a)$ represents the ath entry in vector $Q_{m,n}$ and measures the extrinsic reliability of symbol n being equal to a, seen from the check-node m. Accordingly, $L_n(a)$ denotes the channel information for symbol n and $Q_n(a)$ its *a posteriori* information.

Let $\mathbf{c} = c_1, c_2, \ldots, c_N$ and $\mathbf{y} = y_1, y_2, \ldots, y_N$ be the transmitted codeword and received noisy symbol sequence, respectively. The log-likelihood ratio (LLR) for each received symbol is obtained as $L_n(a) = \log[P(c_n = z_n|y_n)/P(c_n = a|y_n)]$, where z_n is the symbol associated to the highest reliability. The previous definition ensures that all messages $L_n(a)$ are nonnegative and that the smaller the value, the more reliable the message.

We present in Algorithm 1, the T-EMS CN decoding unit where the first step consists in the delta domain transformation of input messages, which are denoted by $\Delta Q_{m,n}(\eta_j)$, being $\eta_j = a + z_{n_j}$ the delta domain index. This transformation ensures that the most reliable messages are always in the

Algorithm 1 T-EMS Algorithm

```
   Input: $\mathbf{Q_{m,n}}$ , $z_n = \arg\min_{a \in GF(q)} Q_{m,n}(a)\ \forall\ n \in \mathcal{N}(m)$
   for $j = 1 \to d_c$ do
1      $\Delta Q_{m,n_j}(\eta_j = a + z_{n_j}) = Q_{m,n_j}(a)$
   end
2  $\beta = \sum_{j=1}^{d_c} z_{n_j} \in GF(q)$
3  $\Delta Q(a) = \min_{\eta'_j(a) \in conf(n_r,n_c)} \sum_{j=1}^{d_c} \Delta Q_{m,n_j}(\eta'_j(a)), a \in GF(q)$
   for $j = 1 \to d_c$ do
4      $\Delta R_{m,n_j}(a + \eta'_j(a)) =$
       $\min(\Delta R_{m,n_j}(a + \eta'_j(a)), \Delta Q(a) - \Delta Q_{m,n_j}(\eta'_j(a)))$
5      $R_{m,n_j}(a + \beta + z_{n_j}) = \lambda \cdot \Delta R_{m,n_j}(a), a \in GF(q)$
   end
   Output: $\mathbf{R_{m,n}}$
```

first index of $\Delta Q_{m,n}(\eta_j)$ and the rest of the symbols are reordered and considered as deviations of the most reliable one, according to step 1. Step 2 involves the computation of CN syndrome β using the most reliable symbol z_n for each CN incoming message. For the syndrome computation, all nonzero elements of $\mathbf{H}$ are taken as $\alpha^0 = 1$ thanks to the preprocessing of the incoming messages outside of the node, as will be explained in following sections.

Step 3 makes use of the configuration sets originally proposed in [5] with the aim of building the output messages by just using the most reliable information. conf(n_r, n_c) is defined as the configuration set composed of the most reliable paths that satisfy the parity check equation. Each of these paths can be formed by the most reliable n_r messages for a symbol a deviating at most n_c times from the zero-order configuration [5], [11]. These combinations are usually named indifferently paths or configurations. Implementation of the step 3 requires the reordering of the delta messages in a trellis fashion considering all the d_c incoming messages as stages of the trellis and the reliability for each GF symbol η_j as the index per trellis stage, and the computation of an extra column $\Delta Q(a)$. $\Delta Q(a)$ is calculated by adding the reliability values of conf(n_r, n_c) with the highest reliability (minimum value).

Hereinafter, we only consider the case when $n_r = 2$ and $n_c = 2$ for T-EMS algorithm. In this case, combinations with the two most reliable symbols are analyzed to build the extra column reliability. This means that combinations of min1-min1, min1-min2, min2-min1, and min2-min2 must be analyzed (and combinations with the rest of corresponding messages are avoided) to extract the paths with higher reliability that deviate at most two times from the most reliable path. min1 and min2 represent the first and second most reliable messages, respectively, i.e., minimum values. In addition, for the same path, no more than one reliability from the same stage of the trellis is considered [8].

Output messages in delta domain $\Delta R_{m,n_j}(a)$ are generated subtracting the reliability of configurations to the information collected in the extra column of trellis $\Delta Q(a)$ (step 4 of Algorithm 1). When more than one configuration is associated to the same output message value, the minimum path metric is considered. The use of an extra column in the trellis allows to compute the output messages in parallel, which reduces the data dependency between the d_c elements involved in the CN and hence improves the overall throughput of the decoder.

Last step of T-EMS algorithm involves the inverse transformation from the delta domain to the normal domain, using the hard decision symbols z_n and the syndrome value β. Before the inverse transformation, a scaling factor λ can be applied to outgoing CN messages to improve the performance of the decoding algorithm.

Although very appealing in terms of throughput, the implementation of step 3 requires many computations in parallel, which increases the overall complexity of the decoder. This is especially true when high-order fields and large CN degree d_c are considered. In the following section, we introduce several simplifications to Algorithm 1, thereby greatly reducing the complexity of the CN unit and improving both latency and area of the global decoder architecture.

III. Simplified TMM Algorithm

In the T-EMS, the output message calculation (step 4 of Algorithm 1) involves $q \times d_c$ subtractions and also $q \times d_c$ minimum finders (min finders), which becomes the bottleneck for the CN processing. Considering this drawback, we propose a simplified algorithm, which considerably reduces the processing load of the CN messages (it avoids the use of subtractions and minimum finder) without inducing any performance loss.

A. Algorithm Description

The modified algorithm introduces a copy of the extra column reliability $\Delta Q(a)$ on the corresponding output message entry $\Delta R_{m,n_j}(a)$, when the configuration path has no deviation at column j for symbol a. On the other hand, when a given configuration is build with a deviation in column j, we have two choices: 1) if the configuration path for symbol a has only one deviation, output reliability is filled with the second most reliable value for the corresponding trellis index (second minimum); or 2) if the configuration path is build with more than one deviation, the output message is filled with the highest reliability in the corresponding trellis index (first minimum).

The simplification mentioned in the last paragraph takes advantage of the fact that only configurations with the most reliable message from each trellis index are considered. This reduces the possible paths by a factor of four with respect to taking configurations with the two most reliable messages for each trellis index. Therefore, only combinations of min1-min1 messages are considered leaving out combinations of min1-min2, min2-min1, and min2-min2. Although this simplification results in a large reduction of the number of considered paths to build the output messages, we did not see any performance loss on the NB-LDPC codes we used for the simulations.

As explained in Section II, the extra column $\Delta Q(a)$ contains the paths formed by the most reliable combination of

Algorithm 2 Simplified TMM Algorithm

```
  Input: $\mathbf{Q_{m,n}}$ , $z_n = \arg\min_{a \in GF(q)} Q_{m,n}(a)\ \forall\ n \in \mathcal{N}(m)$
  for $j = 1 \rightarrow d_c$ do
1     $\Delta Q_{m,n_j}(\eta_j = a + z_{n_j}) = Q_{m,n_j}(a)$
  end
2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in GF(q)$
3 $[m1(a), m1_{col}(a), m2(a)] = \psi\{\Delta Q_{m,n_i}(a)\Big|_{i=1}^{d_c}\}$
4 $\Delta Q(a) = \min_{\eta'_k(a) \in conf^*(1,2)} \left\{ \max_{k=1,2} \left( m1(\eta'_k(a)) \right) \right\}$
  for $j = 1 \rightarrow d_c$ do
5     if $m1(a) \neq \Delta Q_{m,n_j}(a)$ and $m1(a) \neq \Delta Q_{m,n_j}(a)$ then
          $\Delta R_{m,n_j}(a) = \Delta Q(a)$
      else if $m1(a) = \Delta Q_{m,n_j}(a)$ then
          $\Delta R_{m,n_j}(a) = m2(a)$
      else
          $\Delta R_{m,n_j}(a) = m1(a)$
      end
6     $R_{m,n_j}(a + \beta + z_{n_j}) = \lambda \cdot \Delta R_{m,n_j}(a), a \in GF(q)$
  end
  Output: $\mathbf{R_{m,n}}$
```

symbols (step 3 of Algorithm 1). On the other hand, messages $\Delta Q_{\mathrm{mn}}(a)\ \forall\ a \neq 0$ can be treated as deviations from the most reliable symbol, so $\Delta Q(a)$ is the estimation of distance from the most reliable configuration when $a \neq 0$. Although this distance between the configurations is theoretically computed using sums of the local reliability in the trellis, it has been proposed in the min–max algorithm [6] to use the maximum value of the considered path as an alternative measure of distance. We also make use of this idea in our algorithm design, and propose the use of the maximum operator instead of the addition to compute the extra column $\Delta Q(a)$ reliability. Making use of the maximum value to measure distances prevents the data length growth associated to the summation, introducing an important area reduction due to the parallel processing of the trellis algorithm. In (1), the modifications made on step 3 of Algorithm 1 are presented, converting the T-EMS on TMM algorithm

$$\Delta Q(a) = \min_{\eta'_j(a) \in \mathrm{conf}(n_r, n_c)} \times \left\{ \max_{j=1 \rightarrow d_c} \left(\Delta Q_{m,n_j}\left(\eta'_j(a)\right) \right) \right\}, \quad a \in \mathrm{GF}(q). \quad (1)$$

The complete description of the proposed TMM algorithm is presented in Algorithm 2, where step 3 of Algorithm 1 has been split into two basic tasks. In step 3 of Algorithm 2, function ψ extracts the two most reliable messages for each symbol $a \in \mathrm{GF}(q)$ (considering the two most reliable symbols those having the least magnitude). First and second minimum are denoted as $m1(a)$ and $m2(a)$. The ψ function also extracts the position of the most reliable message ($m1_{\mathrm{col}}(a)$), so it can take values from one to d_c.

Step 4 of Algorithm 2 involves the processing of the trellis extra column reliability using information related to the most reliable symbol $m1(a)$. The configuration set $\mathrm{conf}(n_r, n_c)$

from Algorithm 1 [8] includes the set of symbols $\eta'_j(a)$, which contains information about all nodes through which pass the configuration. In this approach, we redefined the configuration set to conf*(n_r, n_c), where the difference is that $\eta'_k(a)$ only retains information from the n_c columns where deviations from the zero-order configuration are made instead of keeping information from all nodes. This simplified storage of configuration sets implies that k can take values from one to n_c, and lead to a significant area gain. In the rest of this paper, we will keep the same constraints for the decoder design, and restrict to the case in which only configurations with the most reliable message for each symbols a ($n_r = 1$) and a maximum of two deviations ($n_c = 2$) are considered.

Additionally, in the TMM algorithm, when $\Delta Q(a)$ is formed by only one deviation, the corresponding $\eta'_1(a)$ and $\eta'_2(a)$ will have the same values. This situation contributes to simplifications in the hardware implementation of Algorithm 2 as we will see in following sections.

Step 5 of Algorithm 2 presents a simplified way to obtain delta domain output messages using a simple assignation of $\Delta Q(a)$, $m1(a)$ or $m2(a)$ depending only on the deviation information from $\eta'_k(a)$. If no deviation for the most reliable path is made on column j for symbol a, then extra column information $\Delta Q(a)$ is directly assigned to the corresponding output message $\Delta R_{m,n_j}(a)$. On the other hand, if any deviation is made for column j and the corresponding path is build with only one deviation, then the second most reliable message for symbol a ($m2(a)$) is assigned to the corresponding output message. In the case of paths formed by more than one deviation, $m1(a)$ is assigned to the output message.

Step 6 of Algorithm 2 shows the transformation of delta domain messages to the normal domain, including a scaling factor λ to improve the performance of the decoder in the waterfall region. The scaling factor value λ is selected among the possible hardware friendly values that do not increase the area of the decoder.

B. Frame Error Rate Performance

For testing the performance of our simplified TMM algorithm, simulations were conducted for a ($N = 837$ and $K = 726$) NB-LDPC code over GF(2^5), where $\mathbf{H}$ is generated using the methods in [12], with $d_c = 27$, $d_v = 4$, and $M = 124$,[1] and using transmission over BPSK modulation and additive white Gaussian noise (AWGN) channel. We compare the TMM with the QSPA [2], and the recently published relaxed min–max (RMM) [9] and T-EMS algorithms [13]. In Fig. 1, we show the frame error rate (FER) simulation results for a layered scheduling.

The TMM algorithm in floating point (fp) simulation was made to be compared with T-EMS and QPSA performance. The configuration set parameters are $n_r = 1$ and $n_c = 2$ for the TMM algorithm although for T-EMS algorithm $n_r = 2$ and $n_c = 2$ were used. For T-EMS algorithm, the optimum λ was selected and 15 iterations (it) for the iterative decoding were used. For T-MM approach, λ value was set to 0.5, since this value do not requires an extra hardware for implementation purposes. Despite this, using the optimum λ value the performance of T-EMS of T-MM approach are very similar. In Fig. 1, we can see that the TMM has a negligible performance loss of 0.05 dB compared with the T-EMS, despite the proposed simplifications.

[1]Note that the $\mathbf{H}$ matrix from this code is not full rank.

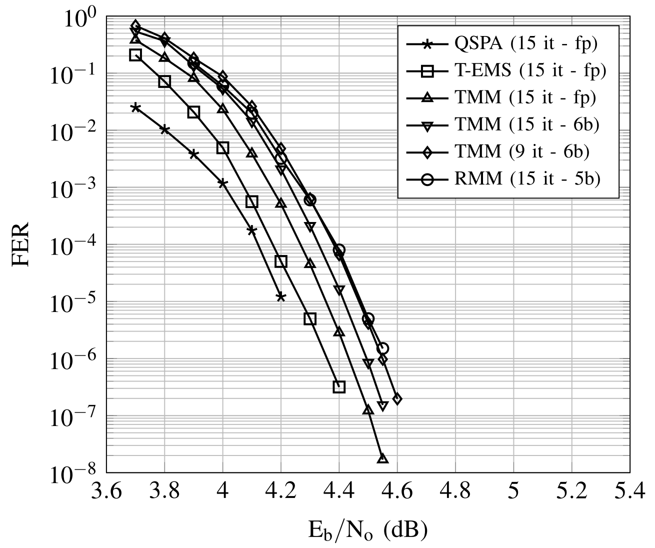

Fig. 1. FER of (837, 726) NB-LDPC over GF(32) under AWGN channel. Layered schedule is used for all algorithms. $\lambda = 0.375$ for T-EMS algorithm and $\lambda = 0.5$ for TMM algorithm (6 bits are required for the messages from VN to CN and 5 bits is the size of the quantized LLR values and the CN to VN messages).

As for the comparison with the QSPA algorithm [2], it can be observed that the TMM algorithm has only 0.2 dB of performance loss, which is a reasonable loss if we consider the huge complexity reduction of TMM.

The quantized version of the TMM algorithm was also simulated where: 1) 6 bits (6b) has been used for the message exchanged from VN to CN; 2) 5 bits is the size of the quantized LLR values; and 3) the messages from CN to VN require only 5 bits due to the use of a scaling factor $\lambda = 0.5$. The cases with 9 and 15 iterations were considered for the iterative decoding. The case with 15 iterations has 0.05 dB of performance loss with respect to fp implementation with the same number of iterations.

The proposed quantized approach with nine iterations was compared with RMM [9]. For RMM algorithm, 5 bits are used for the datapath and the number of iterations are set to 15. In Fig. 1, we can see that both algorithms perform equally, but the reduced number of iterations and the TMM specific features improve a lot the throughput and latency compared with [9], as we will see in the following sections.

IV. CN Architecture

In this section, the design of the CN unit based on TMM algorithm is explained. The CN architecture is shown in Fig. 2, where parallel processing is adopted to generate the output messages $R_{m,n}(a)$.

The first step in the CN processing requires transformation from the normal domain to the delta domain. This delta domain transformation is made using a permutation

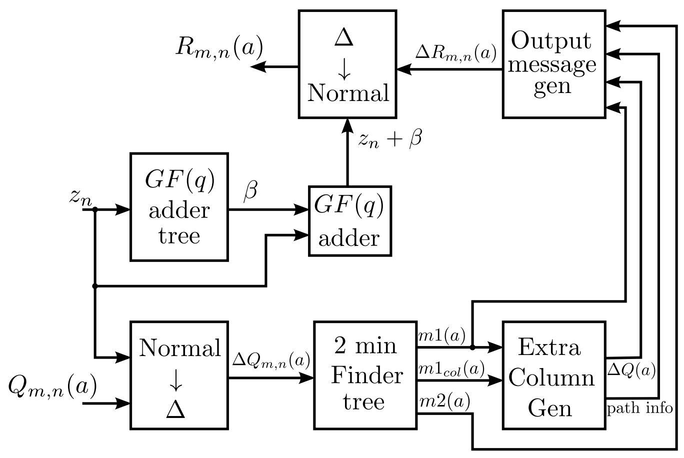

Fig. 2. Proposed top level CN structure.

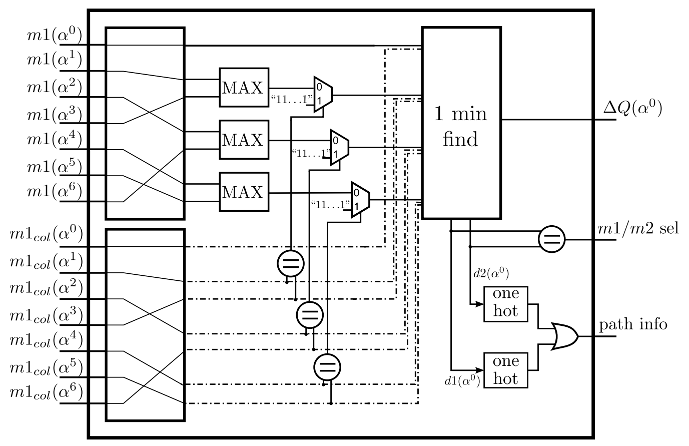

Fig. 3. Architecture for extra column extraction. Example for generation of message $\Delta Q(\alpha^0)$ over GF(8).

network similar to the one proposed in [14]. This network requires $q \cdot \log_2(q)$ multiplexors of two inputs to perform the delta domain transformation of each input vector message $\mathbf{Q_{m,n}}$. Therefore, the CN requires d_c permutation networks where multiplexors are addressed by tentative hard decision symbols z_n. The same structure is used for inverse transformation to normal domain applied to output messages $\Delta R_{m,n}(a)$, where instead of addressing multiplexors using tentative hard decisions symbols, $z_n + \beta$ sum is applied. The CN syndrome β is calculated adding all d_c tentative hard decision symbols. This is performed using a GF adder in a tree structure fashion.

The next step of the CN processing involves the implementation of the function ψ, which extracts the two most reliable messages for each symbol $a \in \mathrm{GF}(q)$. This function is implemented using a 2-min finder tree structure where also the position of the first minimum is extracted [15]. Only $q-1$ cells are required to implement all ψ functions, because in delta domain messages the most reliable symbols remains on $\eta_j = 0$ in the delta domain and their magnitudes are equal to zero. Each ψ function requires d_c inputs because of the processing based on the trellis reordering of the delta messages. The approach followed to implement the ψ function is the tree structure proposed on [15] since it provides a good tradeoff between the area and latency.

Extra column reliability $\Delta Q(a)$ are generated using configurations composed by the most reliable message of each symbol $a \in \mathrm{GF}(q)$, as explained in Section III. The architecture designed for building the extra column is shown in Fig. 3. As an example, $\Delta Q(\alpha^0)$ is obtained for GF(8). The entire cell is similar for all $\mathrm{GF}(q)$ symbols except for the reordering networks in the left side of Fig. 3, which are particularized for each $\mathrm{GF}(q)$ symbol. Since a maximum of two deviations have been considered in the CN implementation, symbols are wired in a way that the GF sum of the symbols, in conjunction with symbol a, meet the parity check equation. For each symbol $a \in \mathrm{GF}(q)$, there are $q/2 - 1$ pair of symbols such that the result of the addition is the symbol a. For example, in Fig. 3, the corresponding pair of symbols are $\alpha^1 + \alpha^3$, $\alpha^2 + \alpha^6$, and $\alpha^4 + \alpha^5$. Since the paths with only one deviation have been also considered, the reliability values corresponding to symbol a (symbol α^0 on Fig. 3) is passed to the block in charge of finding the most reliable path for the corresponding symbol a (1 min find block of Fig. 3).

Once the symbols have been wired, the maximum of the corresponding reliabilities is derived. Next, a validation process is made, in which the reliability arising from the same trellis stage are discarded, since only deviations from different stages are considered. The method used for discarding invalid reliabilities is through comparing the origin of the most reliable messages for a symbol a. If the source trellis stage of both reliabilities is the same, then the maximum value for the quantization scheme is assigned to the corresponding 1 min finder input.

When one and two deviations are considered, the one minimum finder must have $q/2$ inputs and three outputs, which correspond to the reliability for the symbol a of the extra column and the two more outputs that correspond to the trellis stage where deviations were made, called $d1(a)$ and $d2(a)$. For generating the path info for extra column $\Delta Q(a)$, the 1 min find outputs $d1(a)$ and $d2(a)$, with $\lceil \log_2 d_c \rceil$ bits each, are passed through two binary to one hot converters. The outputs of the converters are combined using an OR gate to obtain a unique signal of d_c bits, which contains the total information of the trellis stages where deviations were made. Each bit of this signal is used as a control signal for the output message generation (step 5 of Algorithm 2) in conjunction with the signal $m1/m2$ sel. This signal contains information about the number of deviations taken in each path (one or two deviations).

Once the extra column reliabilities have been obtained, the output messages in delta domain must be generated. The process for building the output messages $\Delta R_{m,n}(a)$ has been greatly simplified with respect to the approach presented in [13].

In [13], $\Delta R_{m,n}(a)$ generation is performed by subtracting from the extra column $\Delta Q(a)$ the contribution of symbols in which deviations were taken. On the other hand, when more than one configuration corresponds to the same output message value, the minimum value is considered, as explained in Section II. As can be seen, the output message

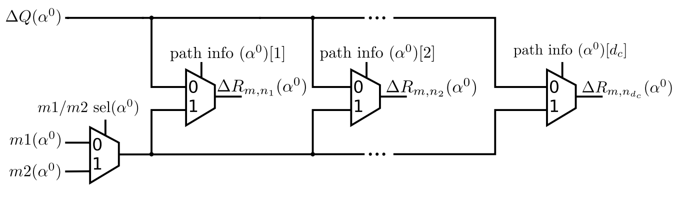
$\Delta Q(\alpha^0)$
path info $(\alpha^0)[1]$
path info $(\alpha^0)[2]$
path info $(\alpha^0)[d_c]$
$\Delta R_{m,n_1}(\alpha^0)$
$\Delta R_{m,n_2}(\alpha^0)$
$\Delta R_{m,n_{d_c}}(\alpha^0)$
$m1/m2$ sel(α^0)
$m1(\alpha^0)$
$m2(\alpha^0)$
0
1

Fig. 4. Output message generation in delta domain. Example for symbol α^0.

generation requires minimum finders and subtractions that increase hardware requirements, and limit the maximum throughput.

The proposed simplified structure for the output message generation is shown in Fig. 4. In it, $d_c + 1$ multiplexors of two inputs are only needed to obtain the output messages for each symbol a. The structure shown in Fig. 4 implements step 5 of Algorithm 2, where each $\Delta R_{m,n}(a)$ message takes its value from $m1(a)$, $m2(a)$, or $\Delta Q(a)$ depending on the control signals obtained during extra column generation. The scaling factor (λ) applied to the output messages can be incorporated in the input messages to the multiplexors shown in Fig. 4. The multiplexors used for the output message generation and for the delta domain inverse transformation can reduce the width of datapath depending on the scaling factor value. As an example, for $\lambda = 0.5$ (value used for generation of FER curves in Fig. 1), these multiplexors can reduce the datapath in 1 b. This has an important impact in the area saving since parallel processing has been adopted in this design.

In the following section, the TMM CN unit is integrated with the rest of the blocks that depicts the entire decoder architecture based on a nonbinary layered scheduling.

V. Architecture for the Complete Decoder

In this section, the top level design of the NB-LDPC decoder, which includes the CN proposed in Section III, is presented. The proposed decoder has been designed for QC-NB-LDPC codes over GF(q) constructed using the methods included in [12], where **H** is formed by $(q-1) \times (q-1)$ circulant submatrices that can be composed of zero elements or cyclic shifted identity matrix with nonzero elements from GF(q).

A. Decoder Schedule

For the proposed decoder, horizontal layered schedule is adopted due to its inherent hardware efficiency. This schedule requires less decoding iterations to achieve a desired performance compared with the flooding schedule. In Algorithm 3, the layered schedule for the proposed decoder is presented, where the CN processor corresponds to the simplified TMM (Algorithm 2).

The decoding process starts loading the channel information $L_n(a)$ on the VNs memories $Q_n(a)$ and then the iterative message passing algorithm continues with steps 1–5 until the maximum iteration number (MaxIter) is reached.

Implementation of the simplified TMM, in the same way as T-EMS, requires processing the CN incoming and outgoing messages avoiding GF multipliers inside the CN processor, similar to the one proposed in [14] for the flooding schedule. In this paper, we address this idea for the horizontal layered schedule. To do this, in step 1 messages $Q_n(a)$ are permuted depending on the corresponding nonzero **H** element h_{mn} to obtain $Q_n(h_{mn}a)$. The permuted VN messages and the last iteration CN outgoing messages $R_{mn}^{(t-1)}(a)$ are processed to obtain $Q'_{mn}(a)$, which correspond to the outgoing VN messages.

Algorithm 3 Layered Schedule for Proposed Decoder

```
Input: $L_n(a) = \log[\frac{P(c_n=z_n|y_n)}{P(c_n=a|y_n)}]$

Inicialization:
    $Q_n^{(1)}(a) = L_n(a), \quad R_{mn}^{(0)}(a) = 0, \quad t = 1$

Main Loop:
    while $t \leq MaxIter$ do
        for $m = 1$ to $M$ do
1           $Q'^{(t)}_{mn}(a) = Q_n^{(t)}(h_{mn}a) - R_{mn}^{(t-1)}(a)$
2           $Q'^{(t)}_{mn} = \min\{Q'^{(t)}_{mn}(a)\} \quad \forall\, a \in GF(q)$
3           $Q_{mn}^{(t)}(a) = Q'^{(t)}_{mn}(a) - Q'^{(t)}_{mn}$
4           $R_{mn}^{(t)}(a) = \text{TMM}\{Q_{mn}^{(t)}(a) \in \mathcal{N}(m)\}$
5           $Q_n^{(t+1)}(h_{mn}^{-1}a) = R_{mn}^{(t)}(a) + Q_{mn}^{(t)}(a)$
        end
6       $t = t + 1$
    end

Output: $\tilde{c}_n = \arg\min\left(Q_n^{(t)}(a)\right) \quad \forall\, a \in GF(q)$
```

Steps 2 and 3 involve normalization of the VN outgoing messages. This process is necessary to ensure the numerical stability of the algorithm and, on the other hand, guarantees that all messages are positive and the most reliable symbol of each message has an associated reliability value equal to zero. Moreover, normalization avoids the growth of the decoder datapath.

Step 4 involves the CN processor where simplified TMM has been used. CN outgoing and incoming messages $R_{mn}(a)$ and $Q_{mn}(a)$ are used for the $Q_n(a)$ message update on step 5 of Algorithm 3. In this step, inverse permutation of $Q_n(a)$ messages have to be done before processing of a new row of **H**. The decoding process stops when the maximum number of iterations is reached, then the output codeword $\tilde{\mathbf{c}}$ is formed by the most reliable symbols associated to the VN messages $Q_n(a)$.

B. Decoder Architecture

The block diagram of the complete architecture for the proposed decoder is shown in Fig. 5, where the datapath for each one of the d_c inputs in the CN is presented. Since the decoder addresses the case of quasi-cyclic NB-LDPC codes build from [12], the $Q_n(a)$ messages can be grouped on sets with $q-1$ messages. In total, assuming w quantification bits for the messages, d_c memories with $q-1$ positions of $q \cdot w$ bits are required for $Q_n(a)$. Only one message is read and one

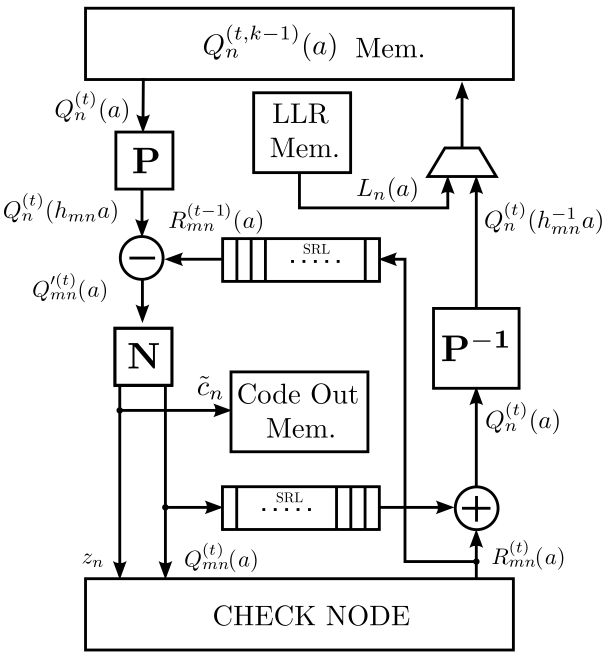

Fig. 5. Top level decoder architecture based on the horizontal layered schedule.

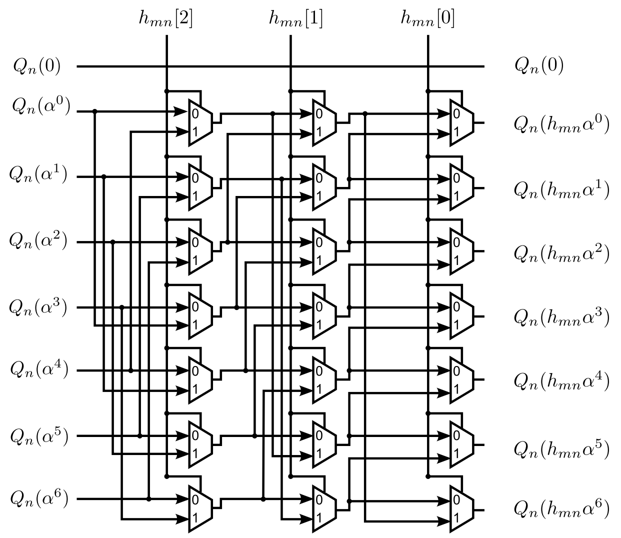

Fig. 6. Permutation network implemented for GF(8).

is written in the same clock cycle from each memory during the processing of one row of **H**.

Blocks **P** and $\mathbf{P^{-1}}$ in Fig. 5 perform direct and inverse permutation of messages $Q_n(a)$ as can be seen in steps 1 and 5 of Algorithm 3, respectively. The permutations are implemented using multiplexor networks as the ones presented in Fig. 6 for the block **P** over GF(8). For the block $\mathbf{P^{-1}}$, the only differences are the connections between multiplexors. Each network requires $(q-1)\cdot\log_2(q)$ multiplexors of w bits. For the entire decoder, $2\cdot d_c$ networks are required to implement the blocks **P** and $\mathbf{P^{-1}}$.

Block **N** in Fig. 5 implements the normalization included in steps 2 and 3 of Algorithm 3. This block includes a one minimum finder which searches the most reliable value to derive $Q'_{mn}(a)$ as explained before. In our approach, we take advantage of the one minimum finder to obtain the most reliable symbol of each $Q'_{mn}(a)$ message using the position to recover the minimum. The recovered symbol is used as input for the CN (z_n). On the other hand, the same symbol corresponds to the estimated hard decision symbol $\tilde{c}_n$ at the end of the decoding process.

To generate the last iteration information for the outgoing CN messages $R_{mn}^{(t-1)}(a)$, it is necessary to include shift registers (SRL) that synchronize with the permuted VN messages. The decoder requires d_c shift registers with M stages and $q\cdot(w-1)$ bits per register.

The incoming CN messages $Q_{mn}(a)$ also require passing through a SRL for synchronizing them with $R_{mn}(a)$ messages (to add them correctly due to pipeline stages used in the decoder). For this purpose d_c SRL are required.

LLR of the received sequence is initially stored in LLR Mem. memories (Fig. 5) and then extracted to be loaded on VN memories ($Q_n(a)$) when the decoding process starts. d_c memories are required with $q-1$ positions and $q\cdot(w-1)$ or $q\cdot w$ bits, respectively. To store the output codeword ($\tilde{c}_n$), d_c memories are also included, each of them with $q-1$ positions of p bits each one. In addition to these memories, parity check matrix nonzero coefficients h_{mn} need to be stored. Due to the structure of **H**, only the coefficients of the first row of each circulant submatrix need to be saved. For doing this, d_c small memories with d_v elements of p bits are added.

C. Decoder Timing

The decoding process starts loading the channel information on $Q_n(a)$ memories, this process consumes $q-1$ clock cycles. Simultaneously, $\tilde{c}_n$ is taken out of $Q_n(a)$ memories and stored on Code Out Mem., as can be observed in Fig. 5. This last process requires that the permutation block **P** and the subtractor do not modify the $Q_n(a)$ messages. Control signals are included to this end.

One decoding iteration starts processing $q-1$ rows of **H**, one at a time. Then, the decoder adds seg clock cycles for emptying the pipeline, where seg corresponds to the number of pipeline stages of the decoder. After that, the next $q-1$ rows of **H** can be processed and seg additional clock cycles are required. The process continues until all the M rows of **H** are processed. In total, one decoding iteration spends $(M/(q-1))*((q-1)+\text{seg}) = M+\text{seg}\cdot d_v$ clock cycles, considering that each circulant matrix has $q-1$ rows as explained in first paragraph of this section. After that, a new decoding iteration begins until the maximum number of iterations finishes (MaxIter).

The throughput of the decoder can be obtained applying (2) where $q-1$ clock cycles are added for loading the channel information for a new decoding process and the output codeword is estimated

$$\text{Throughput} = \frac{f_{\text{clk}}[\text{MHz}]\cdot N\cdot p}{\text{MaxIter}\cdot(M+d_v\cdot\text{seg})+(q-1)}\left[\frac{\text{Mb}}{\text{s}}\right]. \quad (2)$$

D. Decoder Complexity and Implementation Results

As it has been explained before, the decoder was implemented for a ($N = 837$, $K = 726$) NB-LDPC code over

TABLE I
COMPLEXITY ANALYSIS OF THE PROPOSED DECODER FOR THE (N = 837 AND K = 726) NB-LDPC CODE OVER GF(32)

	Logic Gates (NAND)	Memory bits
Check Node	222K	31K
Permutations (**P** and $\mathbf{P^{-1}}$)	76K	-
Normalization (**N**)	58.2K	-
Add/Sub	46.7K	-
$Q_n(a)$	-	160.7K
$R_{mn}^{(t-1)}(a)$	-	535.7K
LLR mem	-	133.9K
Code Out mem	-	4.1K
$Q_{mn}(a)$	-	51.8K
Total	**402.9K**	**917.2K**

GF(32), with parity check matrix **H** and parameters $d_c = 27$ and $d_v = 4$. The CN processor is based on simplified TMM, thus the CN design is entirely combinational logic and has an equivalent area of 222 K NAND gates, using $w = 6$ bits for the datapath. Additionally, 10 pipeline stages have been used in the decoder to increase the maximum frequency of the decoder requiring 31-K registers. The registering points have been selected to balance the critical path, to this end: the segmentation points are listed below.

1) At the output of the P block (Fig. 5).
2) Inside the N block (Fig. 5) that normalizes the incoming messages at the CN.
3) At the input of the CN processor (Fig. 5).
4) At the output of the normal domain to delta domain converter (Fig. 2).
5) In the third stage of the two minimum finder (Fig. 2).
6) At the output of the two minimum finder (Fig. 2).
7) In the second stage of the one minimum finders (Fig. 3).
8) At the output of the extra column reliability values calculation (Fig. 3).
9) At the output of the CN processor (Fig. 5).
10) At the output of the P^{-1} block (Fig. 5).

Outside the CN, the permutation networks **P** and $\mathbf{P^{-1}}$ need 76-K NAND gates, and the normalization blocks (**N**) uses 58.2-K NAND gates. The logic resources of the decoder implementation are summarized in Table I. VHSIC Hardware Description Language was used for the description of the hardware. Cadence register-transfer level Compiler was used for the synthesis and system-on-a-chip encounter for place and route of the design employing a 90-nm CMOS process of nine layers with standard cells and operating conditions of 25 °C and 1.2 V.

After routing, the maximum frequency achieved is 238 MHz and the total area of the decoder is 14.75 mm^2 with a core occupation of 70% and a power consumption of 5.2 W.

Since one iteration of the decoding algorithm takes $M+d_v\cdot$seg clock cycles and considering 10 pipeline stages and 164 clock cycles per iteration are needed. On the other hand, to achieve the same performance as the approach proposed in [9], as shown in Fig. 1, nine iterations are required for the proposed decoding algorithm, which implies that the entire iterative decoding takes $1476+31 = 1507$ clock cycles, where $q - 1$ additional clock cycles are added for the channel information loading. The proposed decoder achieves a throughput of 660 Mb/s using (2), which is very much higher than all existing solutions from the SoA, as shown in the following section, except from the T-EMS [8], [13].

As can be seen, the proposed decoder has low latency without using excessive logic resources, even when higher order Galois fields have been considered. This advantage makes the proposed decoder suitable for high-speed communications systems, where latency is an important requirement.

VI. COMPARISONS WITH OTHER NB-LDPC DECODERS

The proposed decoder has been compared with the most efficient NB-LDPC decoder designs. Table II summarizes the results of different architectures found in the literature. The results of the proposed decoder included in Table II are computed considering that all the memory bits were implemented as RAM memories, except the ones from $R_{mn}^{(t-1)}(a)$ and $Q_{mn}(a)$, which are implemented as registers. According to [9], [16], and [17], the equivalence of 1 B of RAM memory is equal to 1.5 NAND gates and one register equals to 4.5 NAND gates. Considering this equivalence, the total area is equal to 3.5-M NAND gates. On the other hand, we also obtained the number of equivalent NAND gates based on the layout area after the place and route. The area of the decoder is 14.75 mm^2 and the core occupation 70%, using the libraries of our CMOS process the number of equivalent gates can be computed as $(0.7 \times 14.75\ \text{mm}^2)/3.136\ \mu\text{m}^2 = 3.28 - \text{M}$ NAND gates,[2] which is similar to the estimation based on synthesis. Before performing the comparisons with the existing decoders, it is important to remark two facts: 1) the area of the memory bits for $R_{mn}^{(t-1)}(a)$ (535.7 K), the CN (31 K) and the $Q_{mn}(a)$ (51.8 K) is implemented as registers, which have larger equivalent area, because these memories are wider than larger and we do not have fully customized memories for these sizes and 2) area results for the rest of designs that we compare with, except the one in [17], are provided assuming that all the memory bits are RAM not registers, which leads to an underestimation of area. Therefore, if we would consider the same metric, with all memory bits as RAM memory, the improvement of our design in terms of area would increase. Despite this, we just provide postlayout results as they are closer to reality than any other metric or estimation.

Since the decoders of Table II are implemented under different CMOS technologies, we scale the technology to show results over a 90-nm CMOS process using first-order approximations [18] based on the ratio of the maximum achievable frequency for the different processes. To this end, the scaling factors used for deriving the comparisons shown in Table II are 1.66 and 2.33 for 130/90 nm and 180/90 nm scaling, respectively. Note that, we compare different algorithms under the same performance, so each one has a different number of iterations.

[2]The equivalent area of a NAND gate is 3.136 μm^2.

TABLE II
COMPARISON OF THE PROPOSED NB-LDPC LAYERED DECODER WITH OTHER WORKS FROM THE LITERATURE. FOR THE NB-LDPC CODE (837, 726) OVER GF(32)

Algorithm	Simplify-MS [16]	Trellis Max-log QSPA [17]	Min-Max [18]	RMM [9]	T-EMS [13]	Simplified TMM [This Proposal]
Report	Synthesis	Post-layout	Synthesis	Synthesis	Synthesis	Post-layout
Technology	180 nm	90 nm	130 nm	180 nm	90 nm	90 nm
Quantization (w)	5 bits	7 bits	5 bits	5 bits	6 bits	6 bits
Gate Count (NAND)	1.29M[3]	8.51M	2.1M[3]	871K[3]	2.75M[3]	3.28M
f_{clk} (MHz)	200	250	500	200	250	238
Iterations	15	5	15	15	12	9
Latency (clock cycles)	12995	4460	28215	12675	2160	1507
Throughput (Mbps)	64	223	64	66	484	660
Throughput (Mbps) 90 nm	149	223	107	154	484	660
Area (mm^2)	-	46.18	-	-	-	14.75

[3] The gate count of the memory element was obtained by assuming that each memory bit is RAM and equates to 1.5 NAND gates

Considering that only the approach presented in [17] includes postlayout results, only comparisons with [17] can be made for the total decoder area given in mm^2. The TMM outperforms the area by a factor of three compared with the one presented in [17]. On the other hand, our approach has three times higher throughput than the decoder in [17].

Comparing our decoder with the approach presented in [16], we can see that the first one has more than eight times less latency and more than six times higher throughput, although our decoder requires 2.5 times more logic elements (NAND gates).

Li *et al.* [18] have presented a very efficient implementation of the min–max decoder. Our solution outperforms it in latency (almost 20 times lower) and throughput (almost 10 times higher) at a cost 1.56 more area. The min–max decoder has also been modified to improve its hardware efficiency in [9]. Although the TMM requires three times more area than the decoder presented in [9], our approach is 4.28 times faster. In addition, our TMM decoder shows eight times less latency than [9], which makes it suitable for high-speed implementations.

Finally, a T-EMS decoder was presented where Li et al. [13] introduced a low latency decoder achieving 484 Mb/s of throughput. Thanks to the simplifications presented in this paper, our proposed decoder outperforms [13] in latency (43% less) and throughput (36% higher), under the same FER performance.

VII. CONCLUSION

In this paper, we have presented a simplified TMM algorithm, which improves both area and latency with respect to the most efficient decoders included in literature for high-order fields. The outgoing CN messages are calculated in a parallel way using only the most reliable symbols, reducing the overhead of the CN by a factor of four compared with the T-EMS decoder. Using the layered schedule with the proposed CN algorithm reduces the required maximum number of iterations to achieve a desired performance. The improvements proposed in this paper on the hardware implementation of the T-EMS, including the replacement of the addition by a maximum operator (to derive the TMM algorithm) intends to keep the feature of a very high-speed implementation, but with a maximum hardware complexity reduction.

ACKNOWLEDGMENT

The authors would like to thank Dr. E. Li and Dr. K. Gunnam for their discussions on this topic, which helped to improve this paper.

REFERENCES

[1] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[2] M. C. Davey and D. MacKay, "Low-density parity check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, Jun. 1998.

[3] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *Proc. IEEE Inf. Theory Workshop*, Mar./Apr. 2003, pp. 70–73.

[4] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc. IEEE Int. Conf. Commun.*, vol. 2. Jun. 2004, pp. 772–776.

[5] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[6] V. Savin, "Min-max decoding for non binary LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2008, pp. 960–964.

[7] E. Li, K. Gunnam, and D. Declercq, "Trellis based extended min-sum for decoding nonbinary LDPC codes," in *Proc. 8th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Nov. 2011, pp. 46–50.

[8] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary LDPC codes and its hardware structure," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2600–2611, Jul. 2013.

[9] F. Cai and X. Zhang, "Relaxed min-max decoder architectures for nonbinary low-density parity-check codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 11, pp. 2010–2023, Nov. 2012.

[10] 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.

[11] E. Li, “Décodeurs haute performance et faible complexité pour les codes LDPC binaires et non-binaires,” Ph.D. dissertation, Lab. Des Équipes Traitement De L’Inf. Et Syst., Cergy-Pontoise Univ., Cergy-Pontoise, France, 2012.

[12] B. Zhou, J. Kang, S. Song, S. Lin, K. Abdel-Ghaffar, and M. Xu, “Construction of non-binary quasi-cyclic LDPC codes by arrays and array dispersions,” *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1652–1662, Jun. 2009.

[13] E. Li, D. Declercq, K. Gunnam, F. García-Herrero, J. O. Lacruz, and J. Valls, “Low latency T-EMS decoder for NB-LDPC codes,” in *Proc. 47th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Nov. 2013, pp. 831–835.

[14] J. Lin, J. Sha, Z. Wang, and L. Li, “Efficient decoder design for nonbinary quasicyclic LDPC codes,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 5, pp. 1071–1082, May 2010.

[15] C.-L. Wey, M.-D. Shieh, and S.-Y. Lin, “Algorithms of finding the first two minimum values and their hardware implementation,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3430–3437, Dec. 2008.

[16] X. Chen and C.-L. Wang, “High-throughput efficient non-binary LDPC decoder based on the simplified min-sum algorithm,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 11, pp. 2784–2794, Nov. 2012.

[17] Y.-L. Ueng, K.-H. Liao, H.-C. Chou, and C.-J. Yang, “A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using max-log-QSPA,” *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 2940–2951, Jun. 2013.

[18] J. Lin and Z. Yan, “Efficient shuffled decoder architecture for nonbinary quasi-cyclic LDPC codes,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 9, pp. 1756–1761, Sep. 2013.

[19] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ, USA: Pearson Education, 2003.

Jesús O. Lacruz received the B.S. degree in electrical engineering from the Universidad de Los Andes, Merida, Venezuela, in 2009 and the M.S. degree in electrical engineering from the Universitat Politècnica de València, Valencia, Spain, in 2013, where he is currently pursuing the Ph.D. degree in electrical engineering.

He has been an Assistant Professor with the Department of Electrical Engineering, Universidad de Los Andes, since 2010. His current research interests include design of VLSI architectures for digital communications, and in particular, error correction coding.

Francisco García-Herrero received the B.S. degree in telecommunication engineering from the Escuela Politecnica Superior de Gandia, Valencia, Spain, in 2008, and the M.S. and Ph.D. degrees in electrical engineering from the Universitat Politècnica de València, Valencia, in 2010 and 2013, respectively.

He is currently with the Institute of Telecommunications and Multimedia Applications, Valencia. His current research interests include hardware and algorithmic optimization of error-control decoders.

David Declercq was born in 1971. He received the Ph.D. degree in statistical signal processing from the University of Cergy-Pontoise, Cergy, France, in 1998.

He was involved several years in the particular family of LDPC codes, both from the code and decoder design aspects. Since 2003, he has developed a strong expertise on nonbinary LDPC codes and decoders in high-order Galois fields GF(q). A large part of his research projects are related to nonbinary LDPC codes. He mainly investigated two aspects, the design of GF(q) LDPC codes for short and moderate lengths, and the simplification of the iterative decoders for GF(q) LDPC codes with complexity/performance tradeoff constraints. He holds a junior position at the Institut Universitaire de France. He is currently a Full Professor with the Ecole Nationale Supérieure de l’Electronique et de ses Applications, Cergy. He has authored more than 30 papers in major journals of the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON INFORMATION THEORY, the IEEE COMMUNICATIONS LETTERS, the *EURASIP Journal on Wireless Communications and Networking*, and more than 100 papers in major conferences in information theory and signal processing. His current research interests include digital communications and error correction coding theory.

Dr. Declercq is the General Secretary of the National GRETSI Association.

Javier Valls (M’01) received the B.S. degree in telecommunication engineering from the Universidad Politecnica de Cataluna, Barcelona, Spain and the Ph.D. degree in telecommunication engineering from the Universidad Politecnica de Valencia, Valencia, Spain, in 1993 and 1999, respectively.

He has been with the Department of Electronics, Universidad Politecnica de Valencia, since 1993, where he is currently an Associate Professor. His current research interests include design of FPGA-based systems, computer arithmetic, VLSI signal processing, and digital communications.