High-performance NB-LDPC decoder with reduction of message exchange

Jesús O. Lacruz, Francisco García-Herrero, María José Canet, Javier Valls

Member, IEEE

Abstract

This paper presents a novel algorithm based on Trellis Min-Max for decoding NB-LDPC codes. This decoder reduces the number of messages exchanged between check node and variable node processors, which decreases the storage resources and the wiring congestion and, thus, increases the throughput of the decoder. Our Frame Error Rate (FER) performance simulations show that the proposed algorithm has a negligible performance loss for high-rate codes with GF(16) and GF(32), and a performance loss smaller than 0.07dB for high-rate codes over GF(64). Additionally, a layered decoder architecture is presented and implemented on a 90nm CMOS process for the following high-rate NB-LDPC codes: (2304, 2048) over GF(16), (837, 726) over GF(32) and (1536, 1344) over GF(64). In all cases the achieved throughput is higher than 1Gbps.

Index Terms

NB-LDPC, Layered Schedule, Check node processing, High Speed, high rate, VLSI design

I. Introduction

Low-Density Parity-Check (LDPC) codes have been adopted by numerous communication standards such as DVB-S2 [1], IEEE 802.16e [2] and IEEE 802.11n [3], among others. Good error rate performance, low complexity decoders and high-rate decoding are some of the advantages of implementing LDPC codes over other error correction schemes.

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

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

Binary LDPC codes suffer from error correction degradation for short/medium codeword lengths. On the other hand, an effect called error floor appears with high signal-to-noise ratios (SNR). This effect limits the error correction performance, so some additional processing is required to avoid it. Non-Binary LDPC (NB-LDPC) codes, defined over Galois Fields GF($q = 2^p$) with $p > 1$, were first investigated by Davey and MacKay [4] as an extension of binary LDPC codes, where $p = 1$. These codes emerge as an alternative to their binary counterparts to overcome the weaknesses shown by binary LDPC codes. Additionally, they improve the burst error correction capability, especially with high order Galois fields, and offer the possibility to be used in conjunction with high-order modulation schemes (16QAM, 64QAM, 256QAM), reducing the complexity in both the encoder and the decoder [5, 6]. Unfortunately, NB-LDPC codes have some drawbacks: i) high complexity of their check-node (CN); ii) large amount of area spent on storage elements (RAM memories and registers); and iii) routing congestion that limits the overall decoding throughput. From their appearance till now, many efforts have been put into mitigating these problems.

The first algorithm proposed to decode NB-LDPC codes was the Q-ary Sum-of-Product Algorithm (QSPA) [4], which was developed as a generalization of the Sum-of-Product Algorithm (SPA) for binary LDPC codes. Further improvements such as FFT-SPA[7], log-SPA and max-log-SPA[8], were proposed to reduce the complexity of the CN processing equations without introducing any performance loss. More recently, a trellis based implementation for QPSA (T-Max-log-QSPA) [9] was proposed, offering a solution that increases the throughput with respect to previous solutions based on QPSA. Its main drawback is that the required area is prohibitive for real applications in communications and storage systems. Extended Min-Sum (EMS) [10] and Min-Max [11] algorithms were presented as approximations of the QSPA [4], so that they reduce considerably the CN complexity, which only requires additions and/or comparisons. Additionally, EMS and Min-Max algorithms utilize forward-backward (FB) metrics to derive the CN output messages. These metrics involve serial computations which limit the throughput of the derived hardware architectures [11, 12].

Trellis Extended Min-Sum (T-EMS) algorithm was proposed [13, 14] with the aim of enabling parallel processing of the messages in the CN. The input messages are organized in a trellis structure, while the output messages are generated in parallel by means of an extra column included in the trellis. Trellis Min-Max (T-MM) algorithm in [15] adapts the idea of T-EMS to Min-Max algorithm. One Minimum Only TMM (OMO-TMM) [16] is an approximation of

T-MM that reduces the complexity of the CN by obtaining only one minimum and estimating the second one. All these algorithms [13–16] exchange $q \times d_c$ reliability values between CN and VN processors. This amount of exchanged messages is large enough to cause wiring congestion and this limits the maximum throughput, especially for high-rate NB-LDPC codes and high order Galois fields. Additionally, in decoder architectures with layered schedule, the CN output messages are stored to be used in the next iteration. So, the required memory, which is the main part of the area in NB-LDPC decoder architectures [15–17], is too high.

Other proposals from literature [9, 18–22] exchange a minor number of messages between CN and VN and vice versa. This fact, reduces the wiring congestion and the required memory resources, but implies the use of some kind of algorithm to generate the non-exhanged messages. Moreover, these approaches introduce a non-negligible performance loss that depends on the Galois Field order and the size of the reduced set.

In this paper we propose the modified T-MM algorithm (mT-MM) that reduces the number of check-to-variable messages taking advantage of the replicated information in the output messages from the CN in T-MM algorithm. The original idea comes from [23], where we proposed a method to compress the messages between CN and VN for NB-LDPC message-passing decoders. As the messages are not modified, this method does not introduce any performance loss. In [24] we particularise the proposal in [23] to T-MM algorithm, and we detail a hardware architecture for the CN processor and for a decoder with layered schedule. In this paper we extend the work in [24], and present a modification of the T-MM algorithm that allow us to reduce even more the number of exchanged messages. The CN output messages are split in two arrays: one that compresses the extrinsic information and another which represents the intrinsic one. Based on statistical analysis we found that reducing the size of the intrinsic information from q to only two elements introduces a negligible performance loss for high-rate LDPC codes over GF(16) and GF(32) and a performance loss smaller than 0.07dB for high-rate NB-LDPC codes over GF(64), compared to T-MM algorithm [15]. Additionally, we present a high-throughput architecture for the entire decoder (with layered scheduled), which includes the mT-MM algorithm in the CN processor, and compare our implementation results for 90nm CMOS technology with other state-of-the-art decoder architectures.

The rest of the paper is organized as follows: Section II includes the basis of NB-LDPC codes and T-MM algorithm. The proposed modified Trellis Min-Max algorithm (mT-MM) is presented in Section III. Section IV includes the hardware implementation of the mT-MM algorithm and its inclusion in a full decoder. Comparison with other proposals from literature are also devised. Finally, conclusions are presented in Section V.

II. Trellis Min-Max decoding algorithm

NB-LDPC codes are linear block codes defined by a sparse parity-check matrix $\mathbf{H}$ with M rows and N columns, where each non-zero element $h_{m,n}$ belongs to a Galois field GF$(q = 2^p)$. A bipartite graph is commonly used to represent in a graphical way NB-LDPC codes. In this graph, the nodes called variable nodes (VN) represent the N columns of $\mathbf{H}$ and the nodes called check nodes represent the M rows of $\mathbf{H}$. For the sake of simplicity, in this paper we consider regular NB-LDPC codes where the number of VN (CN) connected to a CN (VN) is constant and equal to d_c (d_v). Despite this, the approach presented in this paper is perfectly applicable to irregular NB-LDPC codes including the appropriate control signals to avoid possible memory access conflicts. In the same way, $\mathcal{N}(m)$ $(\mathcal{M}(n))$ denote the set of VN (CN) connected to a CN m (VN n), therefore, the cardinality of the set corresponds to d_c (d_v). $Q_{mn}(a)$ and $R_{mn}(a)$ denote the exchanged messages from VN to CN and from CN to VN for each symbol $a \in$ GF(q), respectively.

Let $\mathbf{c} = c_1, c_2, \cdots, c_N$ be the transmitted codeword over a binary input AWGN channel and $\mathbf{y} = y_1, y_2, \cdots, y_N$ the received symbol sequence, with $\mathbf{y} = \mathbf{c} + \mathbf{e}$, being $\mathbf{e}$ the error vector introduced by the noisy communication channel. $L_n(a)$ corresponds to the *a priori* information from the communication channel obtained by means of the log-likelihood ratio (LLR) as $L_n(a) = \log[P(c_n = z_n|y_n)/P(c_n = a|y_n)]$. All the LLR values are non-negative, and the hard-decision symbol z_n is the GF symbol associated to the highest reliability. $Q_n(a)$ is the *a posteriori* information which is updated as the message passing decoding algorithm progresses.

The CN operations solve the parity check equations, based on the messages from the VN $(Q_{mn}(a))$, and updates the reliability values for each GF symbol a. In this paper we propose an algorithm for the CN to do these tasks (described in Section III), which is based on Trellis Min-Max (T-MM) algorithm [15]. T-MM algorithm offers a good trade-off between coding gain and decoding complexity compared to other proposals from literature.

The basic steps to implement the CN processor of the T-MM algorithm [15] are presented in Algorithm 1. Step 1 involves normal-to-delta domain transformation using the input messages and the hard decision symbols. This transformation ensures that the reliabilities corresponding to the hard-decision symbols z_n are related to the GF symbols $\alpha^{-\infty}$, simplifying the rest of the steps in T-MM algorithm. Step 2 obtains the syndrome β by adding all hard-decision symbols. Step 3 calculates the first and second most reliable messages (minimum values), $m1(a)$ and $m2(a)$, by means of the function ψ, which also extracts the position of $m1(a)$, $m1_{col}(a)$. Step 4 computes the extra column of the trellis, $\Delta Q(a)$, which collects the reliability of the most reliable path for each GF symbol a.

Algorithm 1: T-MM Algorithm [15]

```
  **Input**: $\mathbf{Q_{mn}}$ , $z_n = \arg\min_{a\in \text{GF}(q)} Q_{mn}(a)\ \forall\ n \in \mathcal{N}(m)$
  **for** $j = 1 \to d_c$ **do**
1     $\Delta Q_{mn_j}(\eta_j = a + z_{n_j}) = Q_{mn_j}(a)$
  **end**
2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in \text{GF}(q)$
3 $[m1(a), m1_{col}(a), m2(a)] = \psi\{\Delta Q_{mn_i}(a)\Big|_{i=1}^{d_c}\}$
4 $\Delta Q(a) = \min_{\eta'_k(a)\in conf^*(1,2)} \left\{ \max_{k=1,2} (m1(\eta'_k(a))) \right\}$
  **for** $j = 1 \to d_c$ **do**
5     **if** $m1(\eta'_1(a)) \neq \Delta Q_{mn_j}(a)$ **and** $m1(\eta'_2(a)) \neq \Delta Q_{mn_j}(a)$ **then**
          $\Delta R_{mn_j}(a) = \Delta Q(a)$
      **else if** $\eta'_1(a) = \eta'_2(a)$ **then**
          $\Delta R_{mn_j}(a) = m2(a)$
      **else**
          $\Delta R_{mn_j}(a) = m1(a)$
      **end**
6     $R_{mn_j}(a + \beta + z_{n_j}) = \lambda \cdot \Delta R_{mn_j}(a), a \in \text{GF}(q)$
  **end**
  **Output**: $\mathbf{R_{mn}}$
```

$conf^*(n_r, n_c)$ [15] is the configuration set which selects the possible paths conformed by the n_r symbols with higher reliability value. From all the possible paths, the ones that deviate at most n_c times from the hard-decision are selected. From this reduced set of possible paths, the one chosen for the corresponding $\Delta Q(a)$ value is the one that ensures the highest reliability (minimum value). In this paper we consider the case where $n_r = 1$ and $n_c = 2$. So, only the most reliable messages are considered (First minimum set $m1(a)$) and only one and two deviations paths are taken into account.

Finally, the CN output messages are generated in two steps. First (Step 5 in Algorithm 1), each row of $\Delta R_{m,n}(a)$ is filled with the corresponding $\Delta Q(a)$ reliability, except for the columns that correspond to the stage in the trellis where deviations from the hard-decision path are made. In cases where only one deviation is made, the empty column is filled with the reliability of the second most reliable symbol $m2(a)$. In those cases where two deviations are made, empty columns are filled with the $m1(a)$ reliability. Second (Step 6), conversion from delta to normal domain is required for the CN output messages, where β is used to correct the tentative hard-decision symbols. Additionally, a scaling factor λ is used to improve the performance and convergence rate of T-MM algorithm.

III. Modified Trellis Min-Max Algorithm

This section is organised as follows: in Section III-A we reformulate T-MM algorithm to introduce some variables required to explain how replicated information is reduced and that are used in the definition of the proposed modified T-MM algorithm. Section III-B extends the explanation of the algorithm proposed in [24] taking as a reference the algorithm reformulated in Section III-A and also includes an analogy with binary LDPC decoders. Finally, Section III-C defines the new algorithm (modified T-MM, mT-MM), which is based on an statistical analysis, and gives FER performance results for high-rate NB-LDPC codes over GF(16), GF(32) and GF(64).

A. Reformulation of Trellis Min-Max Algorithm

In this section we reformulate the Trellis Min-Max Algorithm (Algorithm 1) as a first step to define our proposal. As can be seen in Algorithm 2, steps 4 and 5 are the ones reformulated. The function ψ' in Step 4 obtains which path in the trellis was used to obtain $\Delta Q(a)$, that is, the most reliable path. Considering that a maximum of two deviations is evaluated, the function returns the two GF symbols that define this path, $\eta_1^*(a)$ and $\eta_2^*(a)$. If the path used to obtain $\Delta Q(a)$ has only one deviation from the hard-decision path, the function ψ' equals $\eta_2^*(a)$ to $\eta_1^*(a)$. On the other hand, Step 5 calculates $\Delta R_{mn}(a)$, which is equalled to $\Delta Q(a)$, the first minimum of $\Delta Q_{mn}(m1(a))$ or its second minimum ($m2(a)$), depending on the deviation information ($\eta_1^*(a)$ and $\eta_2^*(a)$). For a symbol a, if the most reliable path does not deviate at

Algorithm 2: Reformulated Trellis Min-Max Algorithm

```
Input: $\mathbf{Q_{mn}}$
    $z_n = \arg\min_{a \in \text{GF}(q)} Q_{mn}(a) \; \forall \; n \in \mathcal{N}(m)$
1 $\Delta Q_{mn}(a + z_n) = Q_{mn}(a)$
2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in \text{GF}(q)$
3 $[m1(a), m1_{col}(a), m2(a)] = \psi\{\Delta Q_{mn_i}(a)\Big|_{i=1}^{d_c}\}$
4 $\Delta Q(a) = \min_{\eta'_k(a) \in \; conf^*(1,2)} \big\{ \max\left(m1(\eta'_k(a))\right) \big\}$
  $[\eta^*_1(a), \eta^*_2(a)] = \psi'(\min_{\eta'_k(a) \in \; conf^*(1,2)} \big\{ \max\left(m1(\eta'_k(a))\right) \big\})$
  for $j = 1 \to d_c$ do
5     if $m1_{col}(\eta^*_1(a)) \neq j$ and $m1_{col}(\eta^*_2(a)) \neq j$ then
          $\Delta R_{m,n_j}(a) = \Delta Q(a)$
      else if $m1_{col}(\eta^*_1(a)) = m1_{col}(\eta^*_2(a))$ then
          $\Delta R_{mn_j}(a) = m2(a)$
      else
          $\Delta R_{mn_j}(a) = m1(a)$
      end
6     $R_{mn_j}(a + \beta + z_{n_j}) = \lambda \cdot \Delta R_{mn_j}(a), a \in \text{GF}(q)$
  end
  Output: $\mathbf{R_{mn}}$
```

column j ($m1_{col}(\eta^*_1(a)) \neq j$ **and** $m1_{col}(\eta^*_2(a)) \neq j$) the extra column information $\Delta Q(a)$ is assigned to the output $\Delta R_{mn}(a)$. On the other hand, two different updates can be performed at the columns where deviations from the most reliable path are made: (i) if this path has only one deviation, the second minimum $m2(a)$ is assigned to $\Delta R_{m,n}(a)$; (ii) if this path has two deviations, $m1(a)$ is assigned to the output.

Fig. 1 includes an example of trellis with GF(4) and $d_c = 5$. It shows the CN input messages before ($Q_{mn}(a)$) and after ($\Delta Q_{mn}(a)$) delta domain transformation. The hard-decision symbols are $\mathbf{z} = \{\alpha^1, \alpha^0, 0, \alpha^0, 0\}$. After the normal-to-delta domain transformation, the reliabilities $\Delta Q_{mn}(a)$ in the first row of the trellis are equal to 0. The minimum value per row (per GF symbol a) of $\Delta Q_{mn}(a)$ is enclosed by a dotted box, so, the most reliable path for a GF symbol a must include only these boxes. In Fig. 1 the most reliable path for the symbol α^2 is shown in red color ($\alpha^2 = \alpha^0 + \alpha^1$) . This path is most reliable (reliability equal to the maximum between 5 and 10) than the path that makes only one deviation (reliability equal to 17), so $\Delta Q(\alpha^2) = 10$. In a similar way, the most reliable paths for the symbols α^0 and α^1 are built, but in these cases, with only one deviation from the hard decision path ($\Delta Q(\alpha^0) = 5$ and $\Delta Q(\alpha^1) = 10$). For symbol α^2, the new variables defined in Algorithm 2 are $\eta_1^*(\alpha^2) = \alpha^0$ and $\eta_2^*(\alpha^2) = \alpha^1$. Thus, $m1_{col}(\eta_1^*(\alpha^2)) = 1$, $m1_{col}(\eta_2^*(\alpha^2)) = 2$, $\Delta R_{mn_1}(\alpha^2) = \Delta R_{mn_2}(\alpha^2) = m1(\alpha^2) = 17$ and $\Delta R_{mn_3}(\alpha^2) = \Delta R_{mn_4}(\alpha^2) = \Delta R_{mn_5}(\alpha^2) = \Delta Q(\alpha^2) = 10$.

Normal Domain

	$Q_{m,1}(a)$	$Q_{m,2}(a)$	$Q_{m,3}(a)$	$Q_{m,4}(a)$	$Q_{m,5}(a)$
$a = 0$	10	5	0	40	0
$a = \alpha^0$	30	0	35	0	10
$a = \alpha^1$	0	22	15	17	50
$a = \alpha^2$	25	25	40	30	40

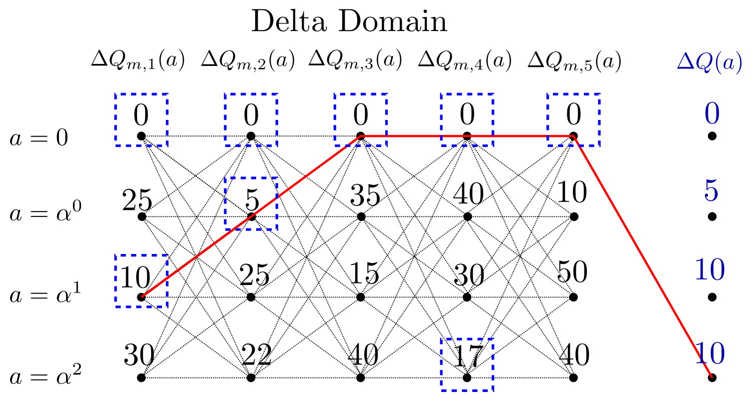

Fig. 1. Example of CN input messages in normal domain (upper size). Messages in delta domain and organized in trellis way including the extra column $\Delta Q(a)$ (bottom size). Example for GF(4) and $d_c = 5$.

B. Reduction of replicated information in check-to-variable exchanged messages

For a better understanding of our proposal, an analogy with binary LDPC decoders is established. In [25] a decoder architecture for binary LDPCs is proposed. In this architecture the messages are compressed in a similar way to which we propose here for the non-binary case. Instead of sending an individual message to each neighbour VN, a CN sends the same message to all its connected VNs, which includes the first minimum, the second minimum, the position of the first minimum and the sign (that depends on the syndrome value). In this way, the routing congestion is reduced.

In the non-binary case, the T-MM algorithm behaves in a similar way. Step 5 in Algorithm 2 generates the CN output messages in delta domain. For each GF symbol a:

$$\Delta R_{mn}(a) = \Delta Q(a)\ \forall\ n \in \mathcal{N}(m) \setminus \{m1_{col}(\eta_1^*(a)), m1_{col}(\eta_2^*(a))\} \tag{1}$$

In (1), $m1_{col}(\eta_1^*(a))$ and $m1_{col}(\eta_2^*(a))$ are the positions of the symbols that ensure the highest reliability of $\Delta Q(a)$ (Step 4 of Algorithm 2). Let us consider the case where the highest reliability path in $\Delta Q(a)$ was built performing only one deviation from the hard-decision path. In this particular case, the exclusion set is reduced to only one position and $\eta_1^*(a) = \eta_2^*(a)$. Therefore, $\Delta Q(a)$ is equal to $m2(a)$ and (1) can be rewritten as (2), which corresponds to the generalization of the CN output message of binary Min-Sum LDPC decoders to the non-binary ones.

$$\Delta R_{mn}(a) = \begin{cases} m1(a)\ \forall\ n \in \mathcal{N}(m) \setminus \{m1_{col}(a)\} \\ \\ m2(a)\ \forall\ n \in \{m1_{col}(a)\} \end{cases} \tag{2}$$

Although (2) is a particular case of (1) in T-MM algorithm, it is useful to remark that $\Delta Q(a)$ plays the role of the intrinsic information from the binary Min-Sum. For the extrinsic messages we define a set $E(a)$ (3) which includes the $m1(a)$ or $m2(a)$ reliabilities depending on the number of deviations (1 or 2) from the hard-decision path.

$$E(a) = \begin{cases} m2(a) & \text{if}\ \ m1_{col}(\eta_1^*(a)) = m1_{col}(\eta_2^*(a)) \\ m1(a) & \text{otherwise} \end{cases} \tag{3}$$

Exchanging the sets $E(a)$ and $\Delta Q(a)$ instead of $R_{mn}(a)$ from the CN to the VN, the cardinality of the messages is reduced from $q \times d_c$ to $2 \times (q-1)$.

Following with the analogy with binary Min-Sum-based LDPC decoders, the extrinsic information related to the syndrome (sign values) is also sent to the VN processor. In the non-binary case, these extrinsic syndromes are obtained as $z_n^* = z_n + \beta \quad \forall\ n \in \mathcal{N}(m)$. This increments the amount of information sent to the VN in d_c p-bits values.

Finally, to reconstruct the $q \times d_c$ messages at the VN processor it is necessary to send the positions where deviations were made to obtain the $\Delta Q(a)$ values. These positions are included in a set $P(a)$ which contains $2 \times (q-1)$ $\lceil \log d_c \rceil$-bits elements.

In terms of bits, the total among of information exchanged from CN to VN is $2 \times (q-1) \times (w + \lceil \log d_c \rceil) + d_c \times p$ bits, where w is the number of bits used to represent the reliability of messages in the decoder. This information has been detailed in Table I for each set exchanged from CN to VN. In this way, the information sent is only reorganized (not modified), so we do not have any performance loss with respect to T-MM.

TABLE I
NUMBER OF BITS EXCHANGED FROM CN TO VN PROCESSOR AFTER REDUCTION OF THE REPLICATED INFORMATION

Set	Number of bits
$\Delta Q(a)$	$(q-1) \times w$
$E(a)$	$(q-1) \times w$
z_n^*	$d_c \times p$
$P(a)$	$2 \times (q-1) \times \lceil \log d_c \rceil$
Total	$2 \times (q-1) \times (w + \lceil \log d_c \rceil) + d_c \times p$

C. Modified Trellis Min-Max algorithm

In this section we propose a new definition of the CN output messages (based on Section III-B) that allow us to reduce the exchanged messages from CN to VN even more. This new definition keeps only a minimum amount of values from $\Delta Q(a)$ (the most reliable ones) and obtains the rest using an approximation function.

First, a statistical analysis for the set $\Delta Q(a), a \in \text{GF}(q)$ was done in order to find its mean value. Using a software model for a NB-LDPC decoder based on T-MM algorithm, we obtained $\Delta Q(a)$ for all the M rows of $\mathbf{H}$ when decoding 10^6 noisy sequences (for $E_b/N_0 = 4.3dB$). Then, we ordered each set $\Delta Q(a)$ from lower to higher value and obtained the mean value of the ordered sets $\Delta Q(a)$ ($\overline{\Delta Q}(a)$). The results of the analysis are presented in Fig. 2. We used the (837,726) NB-LDPC code over GF(32), with degree distribution $d_c = 27, d_v = 4$ built using the methods presented in [26]. Besides, we replicated the analysis for NB-LDPC codes with different degree distributions and Galois Field orders and we obtained the same conclusions. As can be seen in Fig. 2, there is a big increase of mean value from one $\overline{\Delta Q}(a)$ index to the next for the first indexes, however, this increase is lower for the rest of indexes. Based on this observation, we propose to keep only the first minimum, ΔQ_{m1}, and the second minimum,

ΔQ_{m2}, from the set $\Delta Q(a)\ \forall\ a\ \in\ \mathrm{GF}(q) \setminus \alpha^{-\infty}$. Storing only a limited set of values from $\Delta Q(a)$, the exchanged information between CN and VN is reduced.

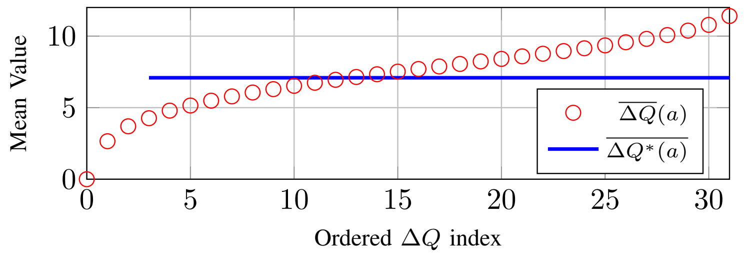

Fig. 2. Mean values for each reliability in the ordered set $\Delta Q(a)$. The code used is the (837,726) NB-LDPC code over GF(32).

At the VN processor, we propose to approximate the rest $q-3$ $\Delta Q(a)$ values using (4), where a_{m1} and a_{m2} are the GF symbols corresponding to ΔQ_{m1} and ΔQ_{m2}, respectively. As the second most reliable value, ΔQ_{m2}, is updated at each iteration, the distance between it and the values $\Delta Q(a)$ approximated using (4) is kept. So, it is expected that the fixed scaling factor γ is greater than one, to ensure that the reliabilities of the approximated values of $\Delta Q(a)$ will not be lower than ΔQ_{m2}.

$$\Delta Q(a) = \gamma \times \Delta Q_{m2} \quad \forall\ a\ \in\ \mathrm{GF}(q)\ \setminus \{\alpha^{-\infty}, a_{m1}, a_{m2}\} \tag{4}$$

The value of the scaling factor γ from (4) is obtained as follows. First, we calculate the mean value of the entire set $\Delta Q^*(a) = \Delta Q(a)\ \forall\ a\ \in\ \mathrm{GF}(q)\ \setminus \{\alpha^{-\infty}, a_{m1}, a_{m2}\}$, named as $\overline{\Delta Q^*(a)}$ in Fig. 2. Then, we obtain the initial value of γ dividing $\overline{\Delta Q^*(a)}$ by the mean value of ΔQ_{m2} ($\overline{\Delta Q_{m2}}$). In Fig. 2, $\overline{\Delta Q^*(a)} = 7.097$ and ΔQ_{m2} is $\overline{\Delta Q_{m2}} = 3.697$, thus the initial value for γ is 1.9198. Finally, we adjust the initial value chosen for γ by means of frame error-rate (FER) simulations, optimized for $E_b/N_0 = 4.3dB$.

Taking into account the modifications presented above and the definitions made in III-B, Algorithm 3 describes the modified Trellis Min-Max (m-TMM) decoding algorithm. Function ψ'' is a modified version of the ψ function from Algorithm 1 which also extracts the position (GF symbol) of the second minimum.

We include in Table II the number of bits exchanged between CN and VN for our proposal and for other works from literature. The rightmost column includes numerical results for the (837,726) NB-LDPC code over GF(32) [26] with degree distribution ($d_c = 27, d_v = 4$). We

```
Algorithm 3: Modified Trellis Min-Max Algorithm
  Input: $\mathbf{Q_{mn}}$
      $z_n = \arg\min_{a \in \text{GF}(q)} Q_{mn}(a) \ \forall \ n \in \mathcal{N}(m)$
1 $\Delta Q_{mn}(a + z_n) = Q_{mn}(a)$
2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in \text{GF}(q)$
3 $[m1(a), m1_{col}(a), m2(a)] = \psi\{\Delta Q_{m,n_i}(a)\Big|_{i=1}^{d_c}\}$
4 $\Delta Q(a) = \min_{\eta'_k(a) \in \ conf^*(1,2)} \big\{ \max\left(m1(\eta'_k(a))\right) \big\}$
  $[\eta^*_1(a), \eta^*_2(a)] = \psi'(\min_{\eta'_k(a) \in \ conf^*(1,2)} \big\{ \max\left(m1(\eta'_k(a))\right) \big\})$
5 $[\Delta Q_{m1}, a_{m1}, \Delta Q_{m2}, a_{m2}] = \psi''\{\Delta Q(a)\Big|_{a=\alpha^0}^{\alpha^{q-2}}\}$
6 $E(a) = \begin{cases} m2(a) & \text{if} \quad m1_{col}(\eta^*_1(a)) = m1_{col}(\eta^*_2(a)) \\ m1(a) & \text{otherwise} \end{cases}$
7 $z^*_n = z_n + \beta \quad \forall \ n \ \in \ \mathcal{N}(m)$
  Output: $\begin{cases} \Delta Q_{m1}, a_{m1}, \Delta Q_{m2}, a_{m2} \\ E(a) \\ z^*_n \\ P(a) = \{ \ m1_{col}(\eta^*_1(a)), m1_{col}(\eta^*_2(a)) \ \} \end{cases}$
```

consider the same number of quantization bits for all proposals ($w = 6$ bits) and we set $n_m = 16$ and $n_v = 5$ according to [19, 21, 22, 27] as they propose for their codes. The work from [15] exchanges a full set of messages which turns into a higher number of bits at the CN output. As can be seen, proposals from [19, 22, 27] eliminate the q-dependence, exchanging only a fraction $n_m < q$ of the reliabilities. This proposals maintain a strong dependence on the CN degree d_c, which penalizes for high-rate NB-LDPC codes. The work from [21] reduces even more the fraction of output messages at the CN compared to previous proposals from literature, being $n_v < n_m < q$. This reduction is offered at the cost of some error-correction degradation and the need of including real-multipliers at the VN for the message approximation. The work from [23] exchanges a fixed number of sets and the size of each set depends of q and d_c without introducing

TABLE II

COMPARISON BETWEEN MULTIPLE PROPOSALS FROM LITERATURE TO REDUCE THE NUMBER OF MESSAGES EXCHANGED FROM CN TO VN

Proposal	Number of bits	($q = 2^p = 32$, $d_c = 27$, $w = 6$, $n_m = 16$, $n_v = 5$)
[15]	$q \times d_c \times w$	5184 bits
[19, 22, 27]	$n_m \times d_c \times w$	2592 bits
[21]	$n_v \times d_c \times w$	810 bits
[23]	$2 \times (q-1) \times (w + \lceil \log d_c \rceil) + d_c \times p$	817 bits
This work	$2 \times (q-1) \times \lceil \log d_c \rceil + (q+1) \times w + (d_c+2) \times p$	653 bits

any performance loss compared to [15]. Finally, we propose in this work a cardinality reduction of the set $\Delta Q(a)$ to only two elements, reducing the total among of bits exchanged to the VN compared to the others proposals from Table II as can be seen in the example from its rightmost column for the high-rate NB-LDPC code over GF(32).

In terms of complexity, the CN processor of Algorithm 3 has less computational load than the one of 2 because it does not compute $q \times d_c$ output messages. So, the number of wires between CN and VN is also reduced.

Fig. 3 and Fig. 4 compare the amount of bits exchanged from CN to VN in a conventional implementation of T-MM with our proposal, varying the field order (p) or the CN degree (d_c), respectively. In all cases, the proposed approach outperforms conventional T-MM in terms of exchanged bits. The differences are considerably higher when the field order and/or the check node degree is increased. This has a great impact on the area of a decoder that uses a layered schedule, as will be seen in Section IV.

Fig. 5 shows the FER performance of the proposed modified Trellis Min-Max (mT-MM) decoding algorithm (floating-point and fixed-point versions (6 bits)) for the (837,726) NB-LDPC code over GF(32), with 15 iterations and $\gamma = 2.0$ (approximated to be hardware-friendly value). It also includes the performance of the floating-point T-MM algorithm [15] with 15 iterations for performance comparison purposes. As can be seen, our proposed algorithm introduces a negligible performance loss of 0.01dB with respect to T-MM (floating-point versions).

Fig. 5 also shows the FER performance of other algorithms from the literature (SMSA [12],

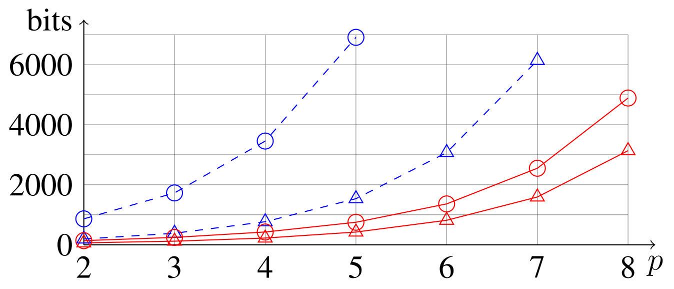

Fig. 3. Number of bits exchanged from CN to VN varying the GF order. Dashed lines corresponds to T-MM and solid lines to mT-MM. Circle mark corresponds to $d_c = 36$ and Triangle mark to $d_c = 8$. $w = 6$.

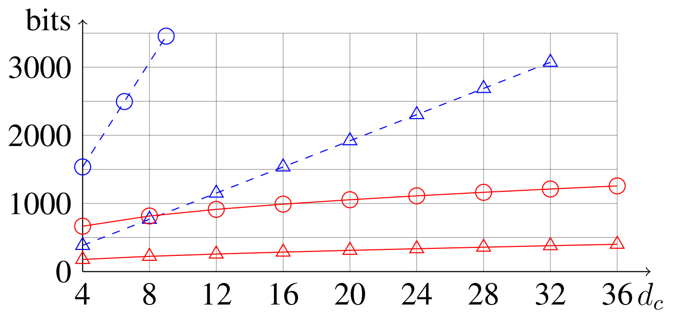

Fig. 4. Number of bits exchanged from CN to VN varying the CN degree. Dashed lines corresponds to T-MM and solid lines to mT-MM. Circle mark corresponds to $q = 64$ and Triangle mark to $q = 16$. $w = 6$.

T-Max-log-QSPA [9], RMM [17] and OMO-TMM [16]) that will be used in Section IV-C to compare their implementation results under the same performance. The number of iterations of each algorithm is adjusted to obtain a performance similar to [17] with 15 iterations, that is, a FER approximately equal to 10^{-6} for $E_b/N_0 = 4.55dB$.

Fig. 6 and Fig. 7 show FER performance results for the (2304,2048) NB-LDPC code over GF(16) ($d_c = 36, d_v = 4$) and the (1536,1344) NB-LDPC code over GF(64) ($d_c = 24, d_v = 3$), respectively. Both NB-LDPC codes are constructed based on the methods from [26]. The algorithms analysed are T-MM and mT-MM. The results show that mT-MM has a performance loss of 0.05dB for the code in Fig. 6 and 0.07dB for the code in Fig. 7 with respect to T-MM. Thus, the proposed mT-MM algorithm achieves good FER performance results for several GF orders and different degree distributions.

Table III summarises the parameters needed to adjust the initial value for the scaling value γ ($\overline{\Delta Q^*(a)}$ and $\overline{\Delta Q_{m2}}$), as well as the hardware-friendly value of γ (γ_{HF}) chosen to generate the

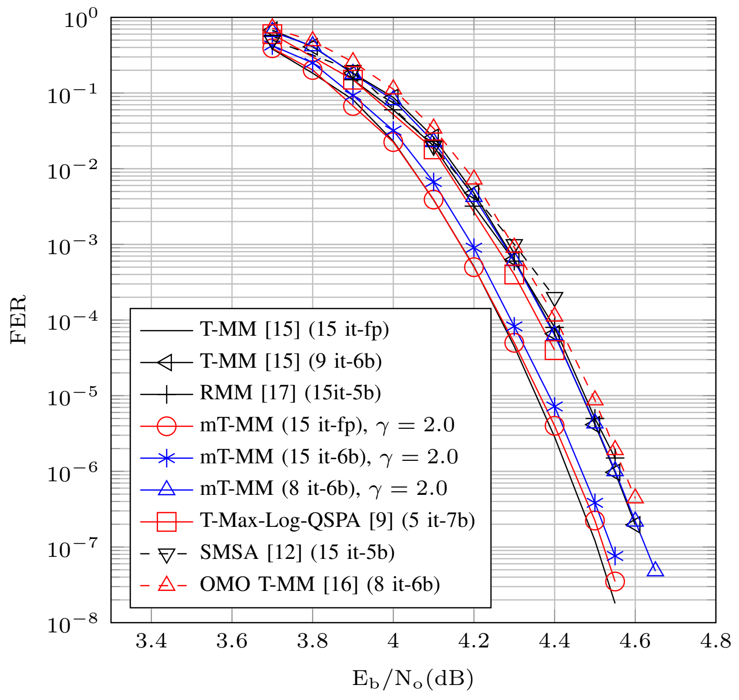

Fig. 5. Frame-error-rate simulation for the (837,726) NB-LDPC code over GF(32), BPSK modulated and assuming AWGN channel

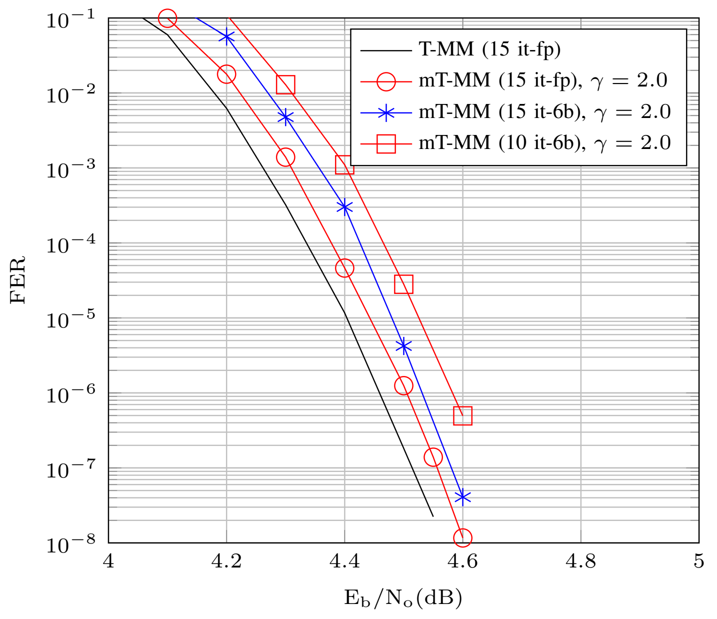

Fig. 6. Frame-error-rate simulation for the (2304,2048) NB-LDPC code over GF(16), BPSK modulated and assuming AWGN channel

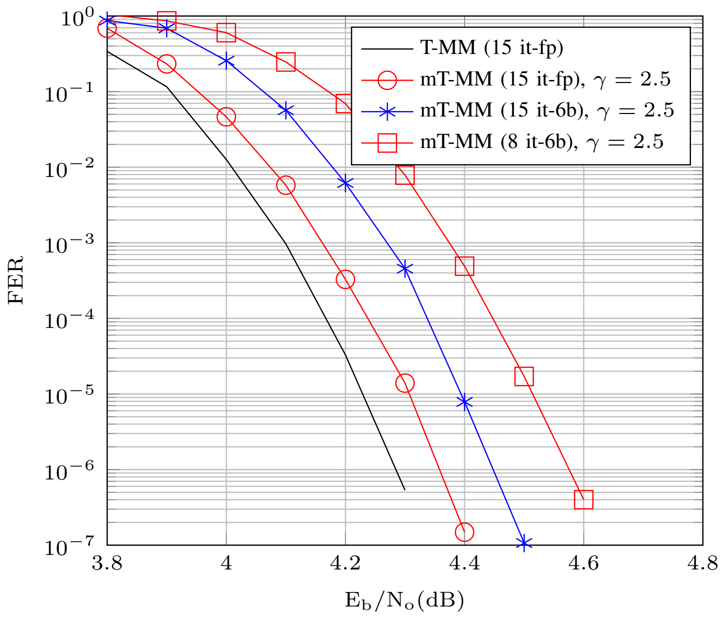

Fig. 7. Frame-error-rate simulation for the (1536,1344) NB-LDPC code over GF(64), BPSK modulated and assuming AWGN channel

FER curves from Fig. 5, Fig. 6 and Fig. 7.

TABLE III

EXPERIMENTAL RESULTS TO SELECT THE APPROPRIATE SCALING VALUE γ, OPTIMIZED FOR $E_b/N_0 = 4.3dB$

NB-LDPC code	$\overline{\Delta Q^*(a)}$	$\overline{\Delta Q_{m2}}$	γ	γ_{HF}
(2304,2048) GF(16)	6.259	3.0612	2.0446	2
(837,726) GF(32)	7.097	3.697	1.9198	2
(1536,1344) GF(64)	8.392	3.084	2.7211	2.5

IV. NB-LDPC DECODER IMPLEMENTATION

In this section we describe the architecture designed to implement the proposed mT-MM Algorithm (Section III-C). Additionally, we include the top level design of a NB-LDPC decoder which uses a layered schedule. The proposed decoder is designed for quasi-cyclic NB-LDPC codes over GF(q) constructed applying the methods in [26], where **H** is formed by QC x QC circulant sub-matrices. These sub-matrices can be composed of zero elements or a cyclic shifted identity matrix with non-zero elements from GF(q). In this way, the number of rows and columns in **H** is $M = QC \times d_v$ and $N = QC \times d_c$, respectively.

A. CN architecture for mT-MM algorithm

Parallel processing is adopted in the CN processor, so its latency is kept low and this increases the overall throughput, as will be seen in next section. The main characteristic of the proposed mT-MM Algorithm is to move part of the complexity of the CN processor to the VN processor. In this way, the number of exchanged messages between them and also the storage resources of the decoder are reduced. Therefore, the CN architecture presented in this section requires less functional blocks than a conventional implementation of T-MM algorithm [15].

Next, the hardware required to perform Algorithm 3 is detailed. Fig. 9 shows the block diagram for the top-level CN architecture, where each block corresponds to a step in the mT-MM algorithm.

Step 1, that is, Normal-to-Delta domain transformation is made by means of d_c permutation networks which follow the structure introduced in [28]. Each one requires $q \times \log(q)$ w-bit MUXES. CN syndrome (Step 2) is obtained using GF-adders in a tree structure ($(d_c - 1) \times p$ XOR gates).

Function ψ (Step 3) is implemented using a tree-based two minimum finder [29], modified to also extract the position of the first minimum [14]. In total $q - 1$ two-minimum finders with d_c inputs are required. Each one is implemented with $2 \times d_c$ w-bit comparators and $3 \times d_c$ w-bit MUXES.

The extra column values, $\Delta Q(a)$, and the corresponding path information (Step 4) are generated using only the most reliable values $m1(a)$ and their corresponding positions $m1_{col}(a)$. A maximum of two deviations from the hard decision path is considered, so, the most reliable path for each value in the set $\Delta Q(a)$ is chosen among a maximum of $q/2$ possible paths (for example, the possible paths for the GF symbol α^0 and GF(8) are α^0, α^1 α^3, α^2 α^6, α^4 α^5). Since the possible paths are different for each value of $\Delta Q(a)$ (for each GF symbol), a custom wired network is required for each one of the $q - 1$ processors used to generate all $\Delta Q(a)$ values. As an example, the processor for the GF symbol α^0 and GF(8) is presented in Fig. 8. The SAT block from Fig. 8 excludes paths deviating more than once in the same stage of trellis. That is, when it detects more than one $m1(a)$ in the same path coming from the same column of the trellis, it assigns the maximum value (minimum reliability) to the one-minimum finder input.

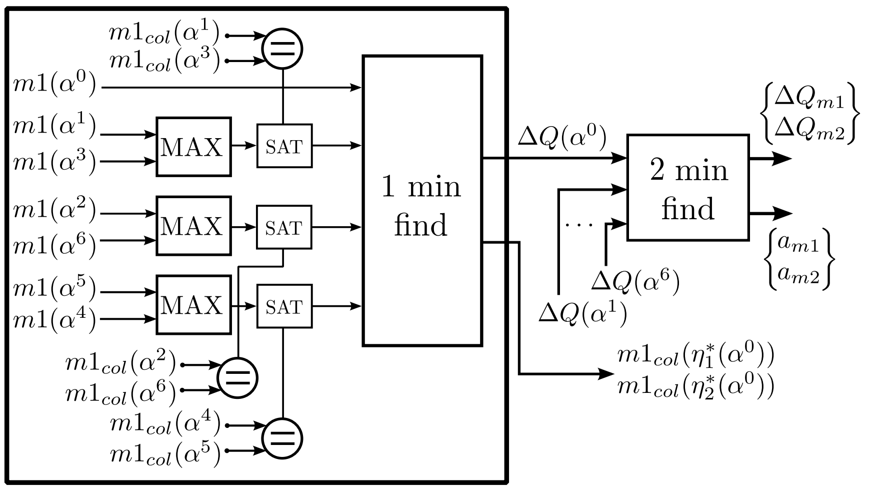

Fig. 8. Extra-Column processor. Example for GF(8) and symbol α^0

Step 5 is implemented as a single two minimum finder with $q-1$ inputs as shown in Fig. 8. It selects the first and second minimum values of the set $\Delta Q(a)$ (ΔQ_{m1} and ΔQ_{m2}, respectively) and their position (GF symbol), a_{m1} and a_{m2}. It receives as inputs the outputs of the $q-1$ extra-column processors to extract the two most reliable (minimum) values.

The computation of the set $E(a)$ (Step 6) requires $(q-1)$ $\lceil \log(d_c) \rceil$-bit comparators and $(q-1)$ w-bit MUXES. With this hardware we distinguish paths with one ($E(a) = m2(a)$) and two deviations ($E(a) = m1(a)$) from the hard-decision path, for each GF(q) symbol.

Finally, the calculation of the extrinsic syndromes (Step 7), z_n^*, requires d_c XOR gates.

As can be seen in Fig. 9, some blocks do not depend on others, so they can be processed in parallel to the rest of blocks. This is the case of the CN syndrome calculation (Step 2), β, and the extrinsic syndromes calculation (Step 7), z_n^*. Additionally, the $E(a)$ calculation (Step 6) and the two-minimum finder (Step 5) can be processed at the same time. This reduces the total latency of the CN architecture.

As it will be explained in Section IV-B, the VN processor uses z_n^*, $E(a)$, $P(a)$, ΔQ_{m1}, ΔQ_{m2}, a_{m1} and a_{m2} to build R_{mn} in Algorithm 2. So, the total among of information exchanged from CN to VN is $(q-1) \times (w + 2 \times \lceil \log d_c \rceil) + d_c \times p + 2 \times p + 2 \times w$ bits, where w is the number of bits used to represent the reliability of messages in the decoder.

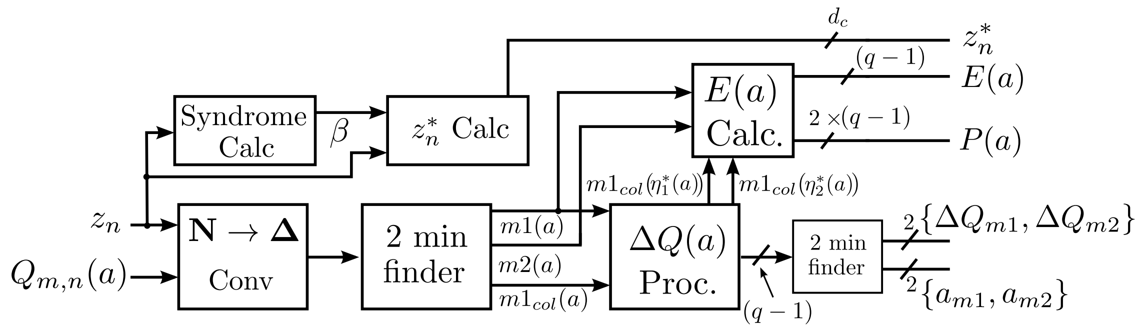

Fig. 9. Proposed check-node block diagram

B. Top-level decoder architecture

In this Section we explain how the CN architecture for the mT-MM algorithm from Section IV-A is included in a complete decoder with horizontal layered schedule. This schedule improves the convergence of the decoding algorithm in comparison with the flooding one. In this way, the number of iterations is reduced and hence the throughput is improved. On the other hand, the area of the resulting decoder is considerably lower than the one required by a fully parallel implementation.

In Algorithm 4 the layered schedule for the proposed decoder is presented, where mT-MM is the CN processor which implements Algorithm 3, and DN is the decompression network from Algorithm 5. The VN processor uses the DN blocks, which generate R_{mn} by using the information given by the mT-MM CN processor.

Algorithm 5 details the operations required to reconstruct R_{mn}, that is, the entire set of $q \times d_c$ messages that goes from CN to VN processors. The decompression network (DN) has as input the reduced set of messages coming from the CN.

The complete block diagram for the proposed decoder is presented in Fig. 10. As can be seen, there is only one check node processor and one VN processor, which processes one row of **H** per clock cycle. Layered schedule requires to store the CN output messages from one iteration to be used in the next one. This is done by means of a shift register with M stages (SR in Fig. 10). The implementation of a conventional CN processor with $q \times d_c$ output messages would require $q \times d_c \times w \times M$ registers. Our proposal only requires $M \times [(q-1) \times (w+2 \times \lceil \log d_c \rceil) + d_c \times p + 2 \times w]$ registers to store the messages from the last iteration. This reduces the storage elements following the behaviour presented in Fig. 3 and Fig. 4 when the field order or the CN degree is varied.

Algorithm 4: Layered Schedule for the Proposed Decoder

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

Inicialization:
    $Q_n^{(0)}(a) = L_n(a)$, $t = 1$, $\Delta Q_{m1} = 0$, $a_{m1} = 0$, $\Delta Q_{m2} = 0$, $a_{m2} = 0$, $E(a) = 0$, $z_n^* = 0$,
    $P(a) = 0$

Main Loop:
    while $t \leq MaxIter$ do
        for $l = 1$ to $M$ do
1           $R_{mn}^{(t-1)}(a) = \text{DN}\{\Delta Q_{m1}, a_{m1}, \Delta Q_{m2}, a_{m2}, E(a), z_n^*, P(a)\}$
2           $Q'_{mn}(a) = Q_n^{(t-1)}(h_{mn}a) - R_{mn}^{(t-1)}(a)$
3           $Q_{mn}(a) = Q'_{mn}(a) - \min \left\{Q'_{mn}(a)\right\}$
4           $z_n = \arg\min(Q'_{mn}(a))$
5           $\left\{\begin{array}{c} \Delta Q_{m1}, \Delta Q_{m2} \\ a_{m1}, a_{m2} \\ E(a), P(a), z_n^* \end{array}\right\} = \text{mT-MM} \left\{Q_{mn}(a),\ z_n\right\}$
6           $R_{mn}^{(t)}(a) = \text{DN}\{\Delta Q_{m1}, a_{m1}, \Delta Q_{m2}, a_{m2}, E(a), z_n^*, P(a)\}$
7           $Q_n^{(t)}(h_{mn}^{-1}a) = R_{mn}^{(t)}(a) + Q_{mn}(a)$
        end
8       $t = t + 1$
    end
Output: $\tilde{c}_n = \arg\min(Q_n(a))$
```

The blocks $\mathbf{P}$ and $\mathbf{P^{-1}}$ in Fig. 10 perform direct and inverse permutation of messages from VN to CN and vice versa, respectively. The permutation is done based on the $h_{m,n}$ non-zero values of **H**.

The "VN mem" block is the memory required to store the messages in the VN processor during the decoding process. The depth of the required memories fits with the size of the circulant sub-matrices (QC) which form **H** [26]. On the other hand, the block "LLR mem" stores the channel

Algorithm 5: Proposed Decompression Operations

```
for $j = 1 \to d_c$ do
  if $P1(a) \neq j$ and $P2_(a) \neq j$ then
    if $a = a_{m1}$ then
      $Out(a + z_j^*) = \Delta Q_{m1}$
    else if $a = a_{m2}$ then
      $Out(a + z_j^*) = \Delta Q_{m2}$
    else
      $Out(a + z_j^*) = \gamma \times \Delta Q_{m2}$
  else
    $Out(a + z_j^*) = E(a)$
    $R_{mnj}(a + z_j^*) = \lambda \times Out(a + z_j^*)$
end for
```

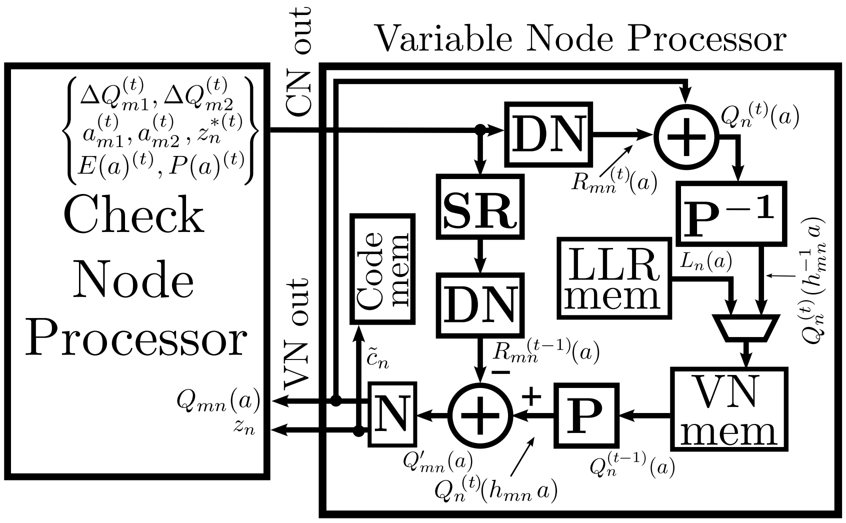

Fig. 10. Top-level proposed decoder architecture

information. This information is loaded in “VN mem” at the beginning of each new decoding frame.

Fig. 11 shows the implementation of a decompression network (DN) for GF(4). A total of d_c decompression networks are required to generate all $q \times d_c$ R_{mn} values. Note that two decompression networks are included in the VN processor. However, the area required in our proposal, which duplicates the logic required to implement DN, is much lower than the one of a conventional implementation of T-MM algorithm with layered schedule ([15], [16]).

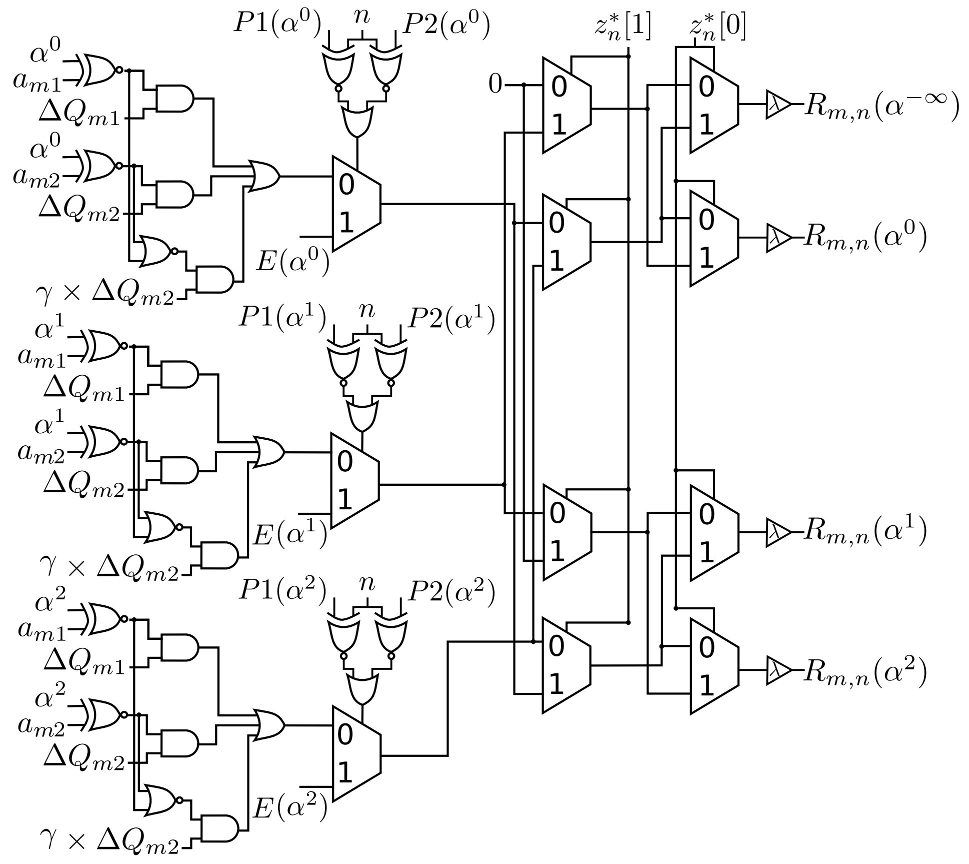

Fig. 11. Proposed Decompression Network. Example for GF(4)

To illustrate the decoder operation, in Fig. 12 a timing diagram is presented. It includes the input and output of the VN processor memory (VN MEM), the CN processor output (CN output) and the VN processor output (VN output). There are $d_v \times QC = M$ rows in $\mathbf{H}$ to be processed in each iteration, that is, M layers which require M clock cycles (one layer per clock cycle). On the other hand, we included seg pipeline stages in the CN to improve timing. After processing QC layers (the size of a circulant matrix), the pipeline must be emptied before processing the following QC layers, which requires seg clock cycles. So, block n in Fig. 12 includes the processing of layers from $QC \times (n-1)+1$ to $n \times QC$, plus seg clock cycles due to the pipeline.

The decoding process starts loading the channel information $L_n(a) = Q_n^{(0)}$ corresponding to the first QC rows on the VN memory ($QC \times d_c \times q$ reliabilities). Then, iteration 1 starts with the processing of block1: $Q_n^{(0)}$ is read from VN MEM and, at the same time, the VN processor starts; after seg clock cycles the CN processor obtains its outputs and $Q_n^{(1)}$ is saved in VN MEM. Then, this process is replicated for blocks from 2 to d_v. The same operations are repeated till the maximum number of decoding iterations ($MaxIter$) is reached. At this point, the tentative hard decoding starts to obtain the symbols $\tilde{c}_n$ and store them in the corresponding memory (Code mem in Fig. 10). Some control signals avoid that the permutation block **P** and the substractor in Fig. 10) modify $Q_n^{(MaxIter)}$ during this process. Finally, the new $QC \times d_c \times q$ LLR values are stored in VN MEM while $\tilde{c}_n$ is obtained, and a new decoding process starts.

The throughput ($Thrput$) of the decoder can be obtained applying (5), where the $d_v \times (QC + seg) = M + seg \times d_v$ clock cycles required per iteration and the QC clock cycles required for initialization and output codeword estimation are included.

$$Thrput = \frac{f_{clk}[\text{MHz}] \cdot N \cdot p}{MaxIter \cdot (M + d_v \cdot seg) + QC} \left[\frac{\text{Mb}}{\text{s}}\right] \quad (5)$$

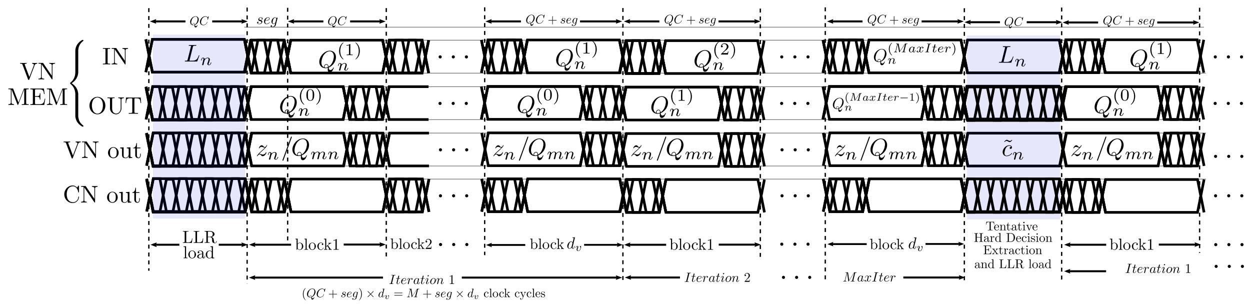

Fig. 12. Decoder timing

C. Decoder implementation results and comparisons

The decoder architecture explained in Section IV-B was implemented on a 90nm CMOS process with nine metal layers and operating conditions 1.2V and 25^oC. VHDL was used for hardware description and Cadence tools were used for synthesis and implementation.

Table IV shows the implementation results for two high-rate NB-LDPC codes whose performance are analysed in Section III-C: (2304,2048) NB-LDPC code over GF(16) ($d_c = 36, d_v = 4$) and (1536,1344) NB-LDPC code over GF(64) ($d_c = 24, d_v = 3$). Our purpose is to show the efficiency of our proposal over different GF(q) order and different degree distribution. The size of the circulant sub-matrices for both codes is $QC = 64$. The number of iterations in Table IV is adjusted to reach a FER approximately equal to 10^{-6} for $E_b/N_0 = 4.55dB$ (see Fig. 6 and Fig. 7). Although both codes have equal number of bits per codeword (9216) and similar rate, an increase by four in the GF(q) order does not have the same impact on the number of gates of the decoder (the GF(64) NB-LDPC code has 2.85 times the number of gates of the one for the GF(16) NB-LDPC code). Additionally, the GF(64) NB-LDPC code has stronger burst error correction capability. On the other hand, our proposal reach a throughput over 1Gbps and 1.3Gbps for GF(16) and GF(64), respectively.

TABLE IV

IMPLEMENTATION RESULTS FOR THE PROPOSED MT-MM ALGORITHM. 90NM CMOS PROCESS

NB-LDPC code	(2304,2048), GF(16)	(1536,1344), GF(64)
(d_c, d_v) / (rate)	(36,4) / (0.889)	(24,3) / (0.875)
Report	Post-layout	Synthesis
Quantization (w)	6 bits	6 bits
Gate Count (NAND)	1.42M	4.05M
f_{clk} (MHz)	380	300
Iterations	10	8
Throughput (Mbps)	1047	1345
Area (mm^2)	11.65	-

Table V compares the implementation of our proposal with other state-of-the-art proposals from literature for the (837,726) NB-LDPC code over GF(32). For each reference, the number of iterations is selected to achieve approximately the same performance (see Fig. 5) and all of them use layered schedule on their implemented decoders. For the proposals that do not use a CMOS 90nm process, the throughput showed in Table V is scaled to this technology using the equations in [30]. On the other hand, our place-and-routed results have a core occupation of 70%.

In terms of gate count, our proposal, which applies parallel processing in the CN, outperforms the other decoders from Table V except for [17]. The decoder from [17] requires 23% less gates than our approach thanks to the serial processing used in their design. This fact introduces an important reduction in the area but increases considerably the latency of the design, as can be

TABLE V

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

Algorithm	SMSA [12]	T-Max-log-QSPA [9]	RMM [17]	T-MM [15]	OMO-TMM [16]	mT-MM [This Proposal]
Report	Synthesis	Post-layout	Synthesis	Post-layout	Post-layout	Post-layout
Technology	180 nm	90 nm	180 nm	90 nm	90 nm	90 nm
Quantization (w)	5 bits	7 bits	5 bits	6 bits	6 bits	6 bits
Gate Count (NAND)	1.29M	8.51M	871K	3.28M	1.79M	1.17M
f_{clk} (MHz)	200	250	200	238	250	345
Iterations	15	5	15	9	8	8
Latency (clock cycles)	12995	4460	12675	1507	1279	1343
Throughput (Mbps) 90 nm	149	223	154	660	818	1080
Efficiency 90 nm (Mbps/M-gates)	115.51	26.2	176.81	201.22	456.98	923.07
Area (mm^2)	-	46.18	-	14.75	16.10	8.97

seen in Table V.

In terms of throughput, our proposal achieves the highest throughput among the solutions from literature listed in Table V. This is due to the reduced set of exchanged messages between CN and VN, which reduces the wiring congestion. Our approach outperforms solutions from [15] and [16],which are the ones with higher throughput in Table V, by 48% and 20 %, respectively.

Regarding efficiency, which is obtained as throughput divided by gate count, our proposal clearly outperforms the rest of decoders: its efficiency is 93.85% higher than the most efficient decoder in Table V [16].

The post-layout area required by the proposed decoder is smaller than any other solution from literature for similar CMOS technology and code parameters. The reduction in area is about 65% compared to [15], which was the solution with lower area until now.

To quantify the reduction in the wire length when mT-MM algorithm is applied, we compare the post-layout results of the decoder from [15] with the proposed approach where the same process is considered for both implementations. The total wire length is 75.4 cm for [15] and 58.2 cm for the proposed decoder which corresponds to a reduction of 23%.

To sum up, the proposed decoder based on the novel mT-MM algorithm offers important advantages compared to the state-of-the-art in both area and throughput. On the other hand, it is important to remark that the proposed mT-MM algorithm does not introduce significant performance loss for Galois field orders lower or equal to GF(32) and involves a non-negligible performance loss of about 0.07dB for GF(64), which is compensated with a great area saving and a throughput over 1.3Gbps, as can be seen in Table IV.

V. Conclusions

The modified Trellis Min-Max algorithm (mT-MM) is proposed in this paper. This algorithm reduces considerably the number of exchanged messages between check-node and variable-node processor in NB-LDPC decoders. In terms of performance, the proposed algorithm introduces a negligible performance loss compared to the original T-MM algorithm for high-rate codes over GF(16) and GF(32). Regarding implementation results, our approach has significant advantages in terms of area and speed compared to proposals that exchange the complete set of messages between check-node and variable-node processors, especially for codes with high order fields and high check-node degree. To show these advantages we implemented several layered decoders with the mT-MM algorithm for different fields and degree distributions, outperforming in all cases others proposals from literature in terms of area and throughput.

Acknowledgment

This work was supported in part by the Spanish Ministerio de Ciencia e Innovación under Grant TEC2011-27916, in part by the Generalitat Valenciana under Grant GV/2014/011 and in part by the Spanish Ministerio de Ciencia e Innovación under Grant TEC2012-38558-C02-02.

References

[1] Digital Video Broadcasting (DVB), "Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)," Ago 2009.

[2] *LDPC coding for OFDMA PHY. 802.16REVe Sponsor Ballot Recirculation Comment.* IEEE C802.16e-04/141r2, 2004.

[3] *Joint Proposal: High Throughput Extension to the 802.11 Standard: PHY. IEEE P802.11 Wireless LANs*. IEEE 802.11-05/1102r4, 2006.

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

[5] J. Fu, M. Arabaci, I. Djordjevic, Y. Zhang, L. Xu, and T. Wang, "First experimental demonstration of nonbinary LDPC-coded modulation suitable for high-speed optical communications," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, March 2011, pp. 1–3.

[6] M. Arabaci, I. Djordjevic, L. Xu, and T. Wang, "Nonbinary LDPC-Coded Modulation for High-Speed Optical Fiber Communication Without Bandwidth Expansion," *Photonics Journal, IEEE*, vol. 4, no. 3, pp. 728–734, June 2012.

[7] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *Proceedings 2003 IEEE Information Theory Workshop*, 2003, pp. 70–73.

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

[9] 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 Transactions on Signal Processing*, vol. 61, no. 11, pp. 2940–2951, 2013.

[10] 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, 2007.

[11] V. Savin, "Min-Max decoding for non binary LDPC codes," in *IEEE International Symposium on Information Theory*, 2008, pp. 960–964.

[12] X. Chen and C.-L. Wang, "High-Throughput Efficient Non-Binary LDPC Decoder Based on the Simplified Min-Sum Algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2784 –2794, nov. 2012.

[13] E. Li, D. Declercq, and K. Gunnam, "Trellis-Based Extended Min-Sum Algorithm for Non-Binary LDPC Codes and its Hardware Structure," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2600–2611, 2013.

[14] E. Li, F. Garcia-Herrero, D. Declercq, K. Gunnam, J. Lacruz, and J. Valls, "Low latency T-EMS decoder for non-binary LDPC codes," in *Signals, Systems and Computers, 2013 Asilomar Conference on*, Nov 2013, pp. 831–835.

[15] J. Lacruz, F. Garcia-Herrero, D. Declercq, and J. Valls, "Simplified Trellis Min-Max Decoder Architecture for Nonbinary Low-Density Parity-Check Codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–1, 2014.

[16] J. Lacruz, F. Garcia-Herrero, J. Valls, and D. Declercq, "One Minimum Only Trellis Decoder for Non-Binary Low-Density Parity-Check Codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 177–184, Jan 2015.

[17] F. Cai and X. Zhang, "Relaxed Min-Max Decoder Architectures for Nonbinary Low-Density Parity-Check Codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 11, pp. 2010–2023, Nov 2013.

[18] X. Zhang and F. Cai, "Efficient Partial-Parallel Decoder Architecture for Quasi-Cyclic Nonbinary LDPC Codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 2, pp. 402–414, Feb 2011.

[19] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Transactions on Communications*, vol. 58, no. 5, pp. 1365–1375, May 2010.

[20] A. Voicila, F. Verdier, D. Declercq, M. Fossorier, and P. Urard, "Architecture of a low-complexity non-binary LDPC decoder for high order fields," in *International Symposium on Communications and Information Technologies, 2007. ISCIT '07.*, Oct 2007, pp. 1201–1206.

[21] J. Lin and Z. Yan, "An Efficient Fully Parallel Decoder Architecture for Nonbinary LDPC Codes," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 12, pp. 2649–2660, Dec 2014.

[22] Y. Park, Y. Tao, and Z. Zhang, “A Fully Parallel Nonbinary LDPC Decoder With Fine-Grained Dynamic Clock Gating,” *IEEE Journal of Solid-State Circuits*, vol. PP, no. 99, pp. 1–12, 2014.

[23] J. Lacruz, F. Garcia-Herrero, and J. Valls, “Reduction of Complexity for Nonbinary LDPC Decoders With Compressed Messages,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–1, 2014.

[24] J. Lacruz, F. Garcia-Herrero, M. Canet, and J. Valls, “A 630 Mbps Non-Binary LDPC Decoder for FPGA,” in *IEEE International Symposium on Circuits and Systems (ISCAS), 2015*, 2015, pp. 1–1.

[25] J. Sha, Z. Wang, M. Gao, and L. Li, “Multi-gb/s ldpc code design and implementation,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 2, pp. 262–268, Feb 2009.

[26] 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 Transactions on Communications*, vol. 57, no. 6, pp. 1652–1662, 2009.

[27] X. Zhang and F. Cai, “Reduced-Complexity Decoder Architecture for Non-Binary LDPC Codes,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 7, pp. 1229–1238, July 2011.

[28] J. Lin, J. Sha, Z. Wang, and L. Li, “Efficient Decoder Design for Nonbinary Quasicyclic LDPC Codes,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 5, pp. 1071–1082, 2010.

[29] C.-L. Wey, M.-D. Shieh, and S.-Y. Lin, “Algorithms of Finding the First Two Minimum Values and Their Hardware Implementation,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 11, pp. 3430–3437, 2008.

[30] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital integrated circuits: a design perspective*. Pearson Education, 2003, 761 pp.