Lacruz Jucht, JO.; García Herrero, FM.; Declercq, D.; Valls Coquillat, J. (2015). Simplified trellis min-max decoder architecture for nonbinary low-density parity-check codes. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 23(9):1783-1792. doi:10.1109/TVLSI.2014.2344113.

# Simplified Trellis Min-Max Decoder Architecture for Non-Binary Low-Density Parity-Check Codes

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

**Abstract**

Non-binary 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, specially for the check node processing, which limits the maximum throughput achievable. Although a great effort has been expended to overcome this disadvantage, the decoders presented in literature are still away from high speed implementations for high order fields. In this paper a simplified Trellis Min-Max (TMM) algorithm is proposed, where the check node messages are computed in a parallel way using only the most reliable information. The proposed check node algorithm is implemented using an horizontal layered schedule. The complete decoder architecture has been implemented in a 90 nm CMOS process for the (837,726) NB-LDPC code over GF(32), achieving a throughput of 660 Mbps at 9 iterations based on post layout results. This decoder increases hardware efficiency in 110% compared to the existing solutions for the same code.

**Index Terms**

NB-LDPC, T-Min-Max, Layered Decoder, Message Passing Algorithm

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

F. García, 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, jvalls@eln.upv.es).

D. Declercq is with the ETIS Laboratory, ENSEA/Univ. Cergy-Pontoise/CNRS-UMR-8051, 6, Avenue du Ponceau, F-95000, Cergy-Pontoise, France (e-mail: david.declercq@ensea.fr).

## I. INTRODUCTION

Non-binary low-density parity-check (NB-LDPC) codes have become an interesting alternative to its binary counterparts for codes with moderate length. The main drawback of NB-LDPC codes is that the complexity of the decoder limits the maximum throughput that can be achieved in hardware implementations.

NB-LDPC are lineal block codes characterized by a sparse parity check matrix $\mathbf{H}$ with $M$ rows and $N$ columns. Each non-zero 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 (CN) and the ones that represent the columns in $\mathbf{H}$, called variable nodes (VN). Decoding algorithms for NB-LDPC codes use iterative message exchange between check nodes and variable nodes and vice versa to extract the most reliable codeword from the noisy received sequence.

Different decoding algorithms have been proposed since NB-LDPC codes were discovered. First, the Q-ary Sum Product algorithm (QSPA) was proposed in [2] as an extension of belief propagation (BP) algorithm for binary LDPC codes. Unfortunately, its complexity was too high to be suitable for hardware implementations. Several approaches such as FFT-SPA [3], log-SPA and max-log-SPA [4], were proposed to overcome the limitations of QSPA. These solutions reduce the complexity of the check node processing equations without introducing any performance loss. In [5] an approximation of QSPA, called Extended Min-Sum (EMS), is proposed, where the complexity of the check node is reduced considerably involving only comparisons and additions. In [6], Min-Max algorithm was presented. This algorithm applies comparisons to compute the maximum reliability values instead of additions, unlike EMS algorithm. This improvement prevents the growth of the data length of the decoder without introducing any performance loss with respect to EMS algorithm.

On the other hand, EMS and Min-Max algorithms still suffer from a bottleneck on check node caused by the use of forward - backward metrics for the extraction of check to variable messages. In [7] the Trellis Extended Min-Sum (T-EMS) is introduced. This algorithm computes the combination of the most reliable messages avoiding the use of forward-backward metrics and 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 hardware designers to derive higher throughput architectures. The main drawback of the approach presented in [8] is that requires a lot of area, reducing the overall efficiency of the decoder.

To further improve the T-EMS efficiency, we propose in this paper a simplification of this algorithm, building the extra column of the trellis and generating the output messages of the check node 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 a efficient way improving both area and latency of the decoder. On the other hand, we look for the maximum value instead of adding the reliabilities for the generation of the extra column values. For this reason we name the algorithm Trellis Min-Max (TMM).

The simplified check node algorithm is implemented using an 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 over high order fields, the (837,726) NB-LDPC code over GF(32) has been chosen. To the best knowledge of the authors, the proposed architecture achieves 110% higher efficiency (Mbps / Million Gates) than the most efficient decoder proposed in literature [9], for the same code. Moreover, the proposed design has lower latency and higher throughput than any proposed NB-LDPC decoder for high order fields.

The rest of the paper is organized as follows: in Section II we introduce the T-EMS decoding algorithm. The proposed algorithm and check node architecture is presented in Section III. Section IV describes the overall layered decoder architectures and the synthesis and post layout results of the proposed design. Section V includes comparisons with others proposed decoders. Conclusions are outlined in Section VI.

## II. Trellis Extended Min-Sum Algorithm

Let us define the parity check matrix $\mathbf{H}$ with $M$ rows and $N$ columns. Each non-zero 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 $d_v$. Let $\mathcal{N}(m)$ ($\mathcal{M}(n)$) be the set of variable nodes (check nodes) connected to a check node (variable node) $m$ ($n$). Let $Q_{m,n}(a)$ and $R_{m,n}(a)$ be the messages from variable node to check node and from check node to variable node for each symbol $a \in GF(q)$ respectively. $L_n(a)$ denotes the channel information and $Q_n(a)$ the *a posteriori* information.

Let $\mathbf{c} = c_1, c_2, \cdots, c_N$ and $\mathbf{y} = y_1, y_2, \cdots, y_N$ be the transmitted codeword and received symbol sequence respectively, with $\mathbf{y} = \mathbf{c} + \mathbf{e}$ and $\mathbf{e}$ is the error vector introduced by the communication channel. 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 non-negative and that the smaller the value, the more reliable the message.

Algorithm 1 includes the T-EMS check node algorithm where the first step consists in the delta domain transformation of input messages. This transformation ensures that the most reliable messages are always in the first row 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 calculus of check node's syndrome $\beta$ using the most reliable symbol $z_n$ for each check node incoming message. For the syndrome calculation, all nonzero elements of $\mathbf{H}$ are taken as $\alpha^0 = 1$ thanks to the pre-processing of the incoming messages outside of the node, as will be explained in later 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 that includes 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], [10]. These combinations usually take the name of paths. Implementation of the step 3 requires the reordering of the delta messages in a trellis fashion 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 values with the higher reliability. This means that combinations of min1-min1, min1-min2, min2-min1, 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 2 times from the most reliable path. min1 and min2 represent the first and second most reliable messages respectively *i.e.* minimum values. For the same path, no more than one message from the same column of the trellis is considered. When higher order fields and larger check node degree $d_c$ are considered, implementation of step 3 requires larger computation in parallel, increasing the total complexity of the decoder.

**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** $j = 1 \rightarrow d_c$ **do**

1 $\quad \Delta Q_{m,n_j}(\eta_j = a + z_{n_j}) = Q_{m,n_j}(a)$

**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** $j = 1 \rightarrow d_c$ **do**

4 $\quad \Delta R_{m,n_j}(a + \eta'_j(a)) = \min(\Delta R_{m,n_j}(a + \eta'_j(a)), \Delta Q(a) - \Delta Q_{m,n_j}(\eta'_j(a)))$

5 $\quad R_{m,n_j}(a + \beta + z_{n_j}) = \lambda \cdot \Delta R_{m,n_j}(a), a \in GF(q)$

**end**

**Output**: $\mathbf{R_{m,n}}$

Output messages in delta domain $\Delta R_{m,n_j}(a)$ are generated subtracting the reliability values 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 converges to the same point, the minimum value is considered, because it contains the highest reliability. The use of an extra column in the trellis allows us to compute the output messages in parallel, which reduces the data dependency between the $d_c$ elements involved in the check node and hence improves the overall throughput of the decoder.

Last step of T-EMS algorithm involves the inverse transformation from delta to "normal domain" using the hard decision symbols $z_n$ and the syndrome value $\beta$. Before the inverse transformation, a scaling factor $\lambda$ can be applied to outgoing check node messages to improve the performance of the decoding algorithm.

In the next section, we introduce several simplifications to Algorithm 1, reducing the complexity of the check node and improving both latency and area of the proposed decoder.

## III. Simplified Trellis Min-Max Algorithm

T-EMS algorithm introduces a novel approach that allows the computation of the check node messages in parallel by means of using an extra column in the trellis. This 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 of the check node processing. Taking into account this drawback, we propose a simplified algorithm which reduces considerably the processing load of the check node messages (it avoids the use of subtractions and minimum finder) without introducing any performance loss.

### *A. Algorithm Description*

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

The simplification introduced in last paragraph takes advantage from the fact that only configurations with the most reliable message from each row are taken into account. This reduces the possible paths by a factor of four with respect to taking configurations with the two most reliable messages for each row of the trellis. Due to that, only combinations of min1-min1 messages must be analyzed leaving out combinations of min1-min2, min2-min1 and min2-min2.

As explained in Section II, the extra column $\Delta Q(a)$ contains the paths formed by the most reliable combination of symbols (Step 3 of Algorithm 1). On the other hand, messages $\Delta Q_{mn}(a) \; \forall \; a \neq 0$ 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 $a \neq 0$. Moreover, in [6] the use of the maximum value as a measure of distance is used in the context of Min-Max algorithm. Applying this idea, we propose the use of the maximum operator instead of the addition to compute the extra column $\Delta Q(a)$ values. 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 T-Min-Max algorithm (TMM).

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

**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 $\quad \Delta Q_{m,n_j}(\eta_j = a + z_{n_j}) = Q_{m,n_j}(a)$

**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 $\quad$ **if** $\eta'_1(a) \neq j$ **or** $\eta'_2(a) \neq j$ **then**
$\quad\quad \Delta R_{m,n_j}(a) = \Delta Q(a)$
$\quad$ **else if** $\eta'_1(a) = \eta'_2(a)$ **then**
$\quad\quad \Delta R_{m,n_j}(a) = m2(a)$
$\quad$ **else**
$\quad\quad \Delta R_{m,n_j}(a) = m1(a)$
$\quad$ **end**

6 $\quad R_{m,n_j}(a + \beta + z_{n_j}) = \lambda \cdot \Delta R_{m,n_j}(a), a \in GF(q)$

**end**

**Output**: $\mathbf{R_{m,n}}$

Must be pointed out that the proposed method for building $\Delta Q(a)$ can be also applied in the T-EMS algorithm [8].

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

Step 4 of Algorithm 2 involves the processing of the trellis extra column values using information related to the most reliable symbol $m1(a)$. The configuration set $conf(n_r, n_c)$ 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 new definition of configuration sets implies that $k$ can take values from one to $n_c$. In the rest of paper we focus on 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.

In the proposed 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 next 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 includes the "normal domain" output message generation where the scaling factor $\lambda$ is included to improve the performance of the proposed approach. The scaling factor value $\lambda$ 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 the simplified algorithm, simulations were made for (837,726) NB-LDPC code over $GF(2^5)$ where $\mathbf{H}$ is generated using the methods in [11], with $d_c = 27$ and $d_v = 4$, and using transmission over BPSK modulation and AWGN channel. We compare the proposed approach to QSPA [2], Relaxed Min-Max (RMM) [9] and T-EMS algorithms [12]. Fig. 1 shows the frame error rate (FER) simulation results for layered schedule.

A TMM algorithm floating point (fp) simulation was done to be compared to 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 $n_r = 2$, $n_c = 2$ were used. For both algorithms $\lambda = 0.5$ and 15 iterations (it) for the iterative decoding were used. In Fig. 1 we can see that the proposed approach has a coding gain of 0.05dB with respect to T-EMS approach even considering that the T-EMS approach analyze configurations with two most reliable messages instead of use only the most reliable information as in TMM algorithm.

For QSPA algorithm [2], the $n_m$ parameter is set to 32 and the number of iterations is equal to 15. Using these values for the QSPA algorithm, it can be seen that TMM algorithm has 0.2 dB of performance loss, which can be assumed considering the complexity involved in the check node implementation of any version of QSPA algorithm (FFT-SPA, log-QSPA or max-log-SPA).

The quantized version of TMM algorithm was also simulated where 6 bits (6b) has been used for the datapath of the decoder. The cases with 9 and 15 iterations were considered for the iterative decoding. The case with 15 iterations has 0.05 dB of performace loss with respect to fp implementation with the same number of iterations.

The proposed quantized approach with 9 iterations was compared with RMM [9]. For RRM 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 equal with the parameters described before. Despite this, the TMM approach requires less iterations than the method from [9] to achieve the same performance which will improve throughput and latency, as we will see in next sections.

## IV. Check Node Architecture

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

The first step in the check node processing requires transformation from "normal" to delta domain. This delta domain transformation is made using a permutation network similar to the one proposed in [13]. 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 check node 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 check node's syndrome $\beta$ is calculated adding all

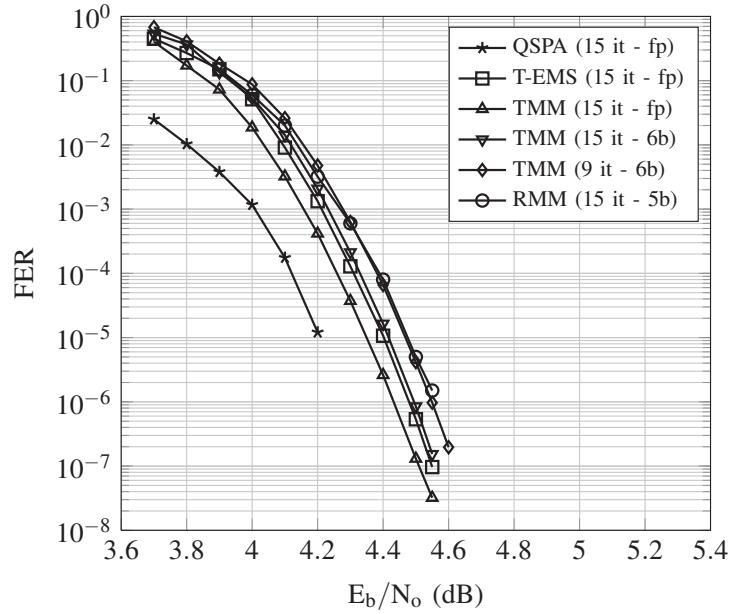


Fig. 1. FER of (837,726) NB-LDPC over $GF(32)$ under AWGN channel. Layered schedule is used for all algorithms. $\lambda = 0.5$ for both T-EMS and TMM algorithms

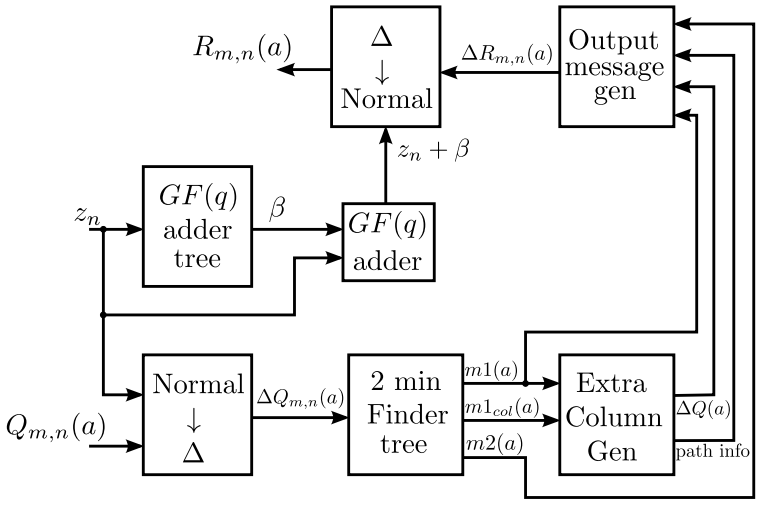


Fig. 2. Proposed top level check node structure.

$d_c$ tentative hard decision symbols. This is performed by means of a $GF$ adder in a tree structure fashion.

Next step of the check node processing involves the implementation of the function $\psi$, which extracts the two most reliable messages for each symbol $a \in GF(q)$. This function is implemented

using a 2-min finder tree structure where also the position of the first minimum is extracted [14]. Only $q-1$ cells are required to implement all $\psi$ functions, because in delta domain messages from the most reliable symbols remain on the first row and their magnitudes are equal to zero. Each $\psi$ function requires $d_c$ inputs because of the row-wise processing of the delta messages. The approach followed to implement the $\psi$ function is the tree structure proposed on [14] since it provides a good compromise between area and latency.

Extra column elements $\Delta Q(a)$ are generated using configurations composed by the most reliable message of each symbol $a \in GF(q)$ as explained in Section III. The architecture designed for building the extra column is presented in Fig. 3. As an example, $\Delta Q(\alpha^0)$ is obtained for $GF(8)$. The entire cell is similar for all $GF(q)$ symbols except for the reordering networks in the left side of Fig. 3, which are particularized for each $GF(q)$ symbol. Since a maximum of two deviations have been considered in the check node implementation addressed in this paper, then 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 $a \in GF(q)$, 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 $\alpha^0$ on Fig. 3) is passed to the block responsible of finding the most reliable path for the corresponding symbol $a$ ("1 mind 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 values arising from the same column are discarded, since only deviations from different columns are taken into account. The method used for discarding invalid reliabilities is through comparing the origin of the most reliable messages for a symbol $a$. If the source column 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 value for the symbol $a$ of the extra column and the two more outputs that correspond to the column numbers 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 one, 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 columns where deviations were

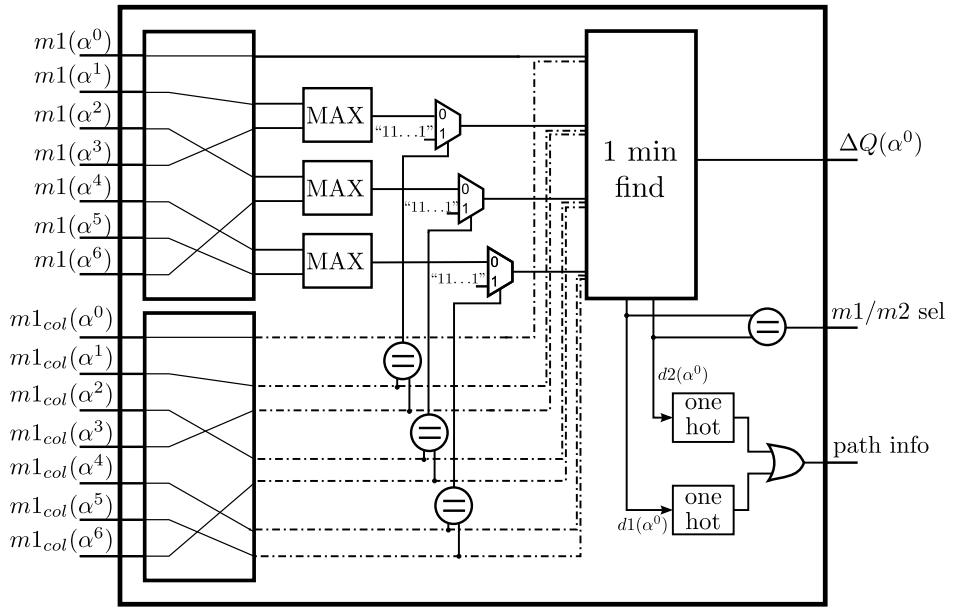


Fig. 3. Architecture for extra column extraction. Example for generation of message $\Delta Q(\alpha^0)$ over $GF(8)$.

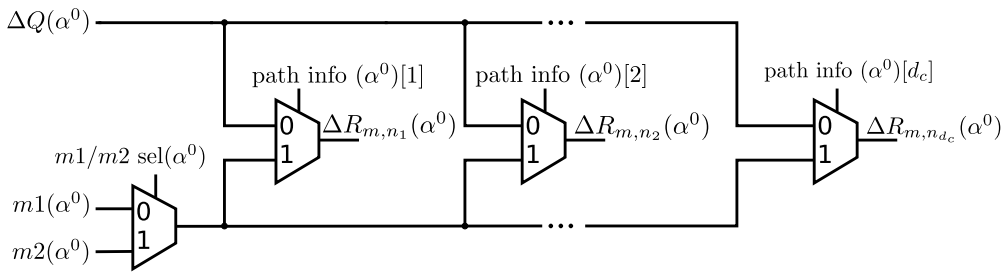


Fig. 4. Output message generation in delta domain. Example for symbol $\alpha^0$

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 values have been obtained, the output values in delta domain must be generated. The process for building the output messages $\Delta R_{m,n}(a)$ have been greatly simplified with respect to the approach presented in [12].

In [12] $\Delta R_{m,n}(a)$ generation is made subtracting from the extra column $\Delta Q(a)$ the contributions of symbols in which deviations were taken. On the other hand, when more than one configuration converge to the same point the minimum value is considered as explained in Section II. As can

be seen, the output message generation requires minimum finders and subtractions that increase hardware requirements limiting the maximum throughput.

The proposed simplified structure for the output message generation is presented 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 presented 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 value of control signals obtained during extra column generation. The scaling factor ($\lambda$) 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 on Fig. 1), these multiplexors can reduce in one bit the datapath. This has an important impact in the area saving since parallel processing has been adopted in this design. It is also important to remark that the decoder proposed in this paper is focused in high order Galois fields.

In the following section, the check node unit presented is integrated with the rest of the blocks that conform the entire decoder architecture based on non-binary layered schedule.

## 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 [11], where $\mathbf{H}$ is formed by $(q-1)\times(q-1)$ circulant sub-matrices that can be composed of zero elements or cyclic shifted identity matrix with non-zero 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 to the flooding schedule. In Algorithm 3 the layered schedule for the proposed decoder is presented, where the check node processor corresponds to the simplified TMM (Algorithm 2).

**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 $Q_{mn}^{\prime\,(t)}(a) = Q_n^{(t)}(h_{mn}a) - R_{mn}^{(t-1)}(a)$

2 $Q_{mn}^{\prime\,(t)} = \min\ \left\{Q_{mn}^{\prime\,(t)}(a)\right\} \quad \forall\ a \in GF(q)$

3 $Q_{mn}^{(t)}(a) = Q_{mn}^{\prime\,(t)}(a) - Q_{mn}^{\prime\,(t)}$

4 $R_{mn}^{(t)}(a) = \text{TMM}\ \left\{Q_{mn}^{(t)}(a) \in \mathcal{N}(m)\right\}$

5 $Q_n^{(t+1)}(h_{mn}^{-1}a) = R_{mn}^{(t)}(a) + Q_{mn}^{(t)}(a)$

**end**

6 $t = t + 1$

**end**

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

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

Implementation of the simplified TMM, in the same way as T-EMS, requires processing the check node incoming and outgoing messages avoiding $GF$ multipliers inside the check node processor, similar to the one proposed in [13] for the flooding schedule. In this paper we address this idea for the horizontal layered schedule. To do this, in step 1 messages $Qn(a)$ are permuted depending on the corresponding nonzero $\mathbf{H}$ element $h_{mn}$ to obtain $Q_n(h_{mn}a)$. The permuted VN messages and the last iteration check node outgoing messages $R_{mn}^{(t-1)}(a)$ are processed to obtain $Q'_{mn}(a)$ which corresponds to the outgoings VN messages.

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's datapath.

Step 4 involves the check node processor where simplified TMM has been used. Check node outgoing and incoming messages $R_{mn}(a)$ and $Q_{mn}(a)$ are used for the $Q_n(a)$ message actualization 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 $\mathbf{H}$. 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 check node is presented. Since the design of the decoder has been addressed for NB-LDPC codes constructed with method proposed in [11], the $Q_n(a)$ messages can be grouped on sets with $q-1$ messages each one. 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 is written in the same clock cycle from each memory during the processing of one row of $\mathbf{H}$.

Blocks $\mathbf{P}$ 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 $\mathbf{P}$ 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 $\mathbf{P}$ and $\mathbf{P^{-1}}$.

Block $\mathbf{N}$ 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 check node ($z_n$). On the other hand, the same symbol corresponds to the estimated hard decision symbol $\tilde{c}_n$ at the end of the decoding process.

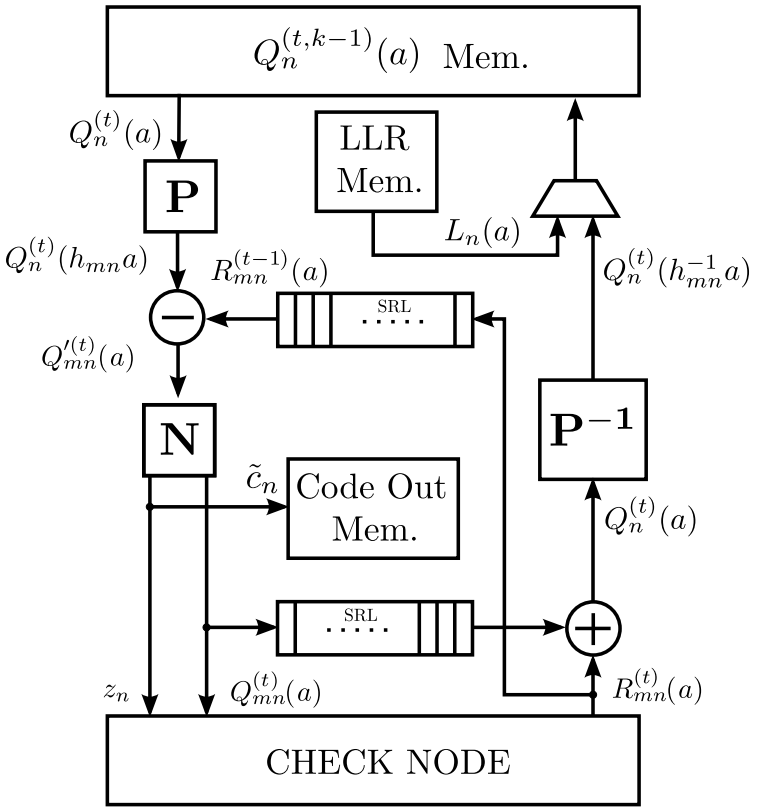


Fig. 5. Top level decoder architecture based on the horizontal layered schedule

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

The incoming check node messages $Q_{mn}(a)$ also require passing through a SRL for synchronizing them with $R_{mn}(a)$ messages (to add them correctly due to segmentation 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 $q \cdot w$ bits each of them. 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. On addition to these memories, parity check matrix nonzero coefficients $h_{mn}$ need to be stored. Due to the structure of $\mathbf{H}$, only the coefficients of the first row of each circulant sub-matrix need to be saved. For doing this $d_c$ small memories with $d_v$ elements of $p$ bits are added.

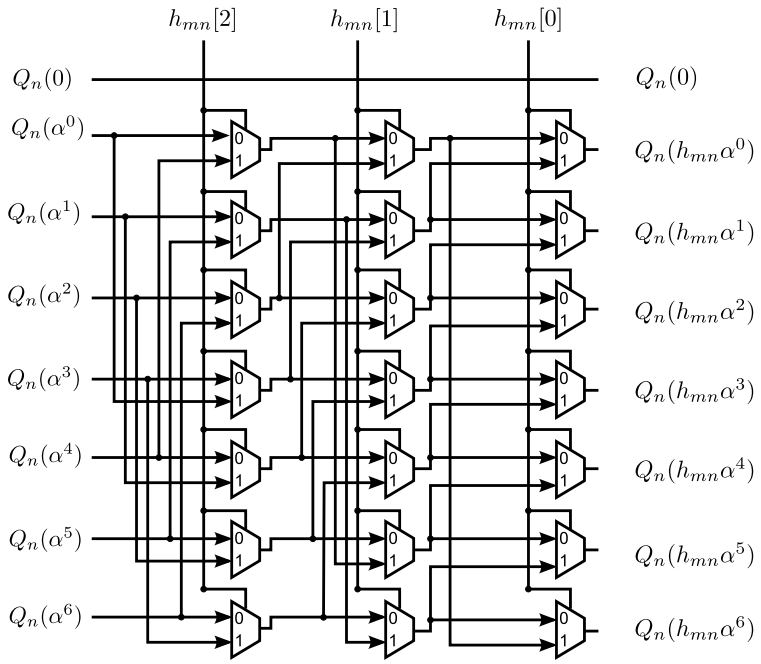


Fig. 6. Permutation network implemented for $GF(8)$

### C. Decoder Timing

The decoding process starts loading the channel information on $Q_n(a)$ memories, this process consumes $q-1$ clock cycles. At the same time $\tilde{c}_n$ is taken out of $Q_n(a)$ memories and stored on "Code Out Mem.", as can be seen in Fig. 5. This last process requires that the permutation block **P** and the subtractor don't 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 $M + seg \cdot d_v$ clock cycles. 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.

TABLE I

COMPLEXITY ANALYSIS FOR THE PROPOSED DECODER. FOR THE (837,726) NB-LDPC CODE OVER $GF(32)$

| | Logic Gates (NAND) | Memory bits |
|---|---|---|
| Check Node | 222K | 31K |
| Permutations ($\mathbf{P}$ and $\mathbf{P^{-1}}$) | 76K | - |
| Normalization ($\mathbf{N}$) | 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** |

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

### *D. Decoder Complexity and Implementation Results*

As it has been explained before, the decoder was implemented for the (837,726) NB-LDPC code over $GF(32)$, with parity check matrix $\mathbf{H}$ and parameters $d_c = 27$ and $d_v = 4$. The check node processor is based on simplified TMM, thus the CN design is entirely combinational logic and has an equivalent area of 135K 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 31K registers.

Outside the check node, the permutation netwotks $\mathbf{P}$ and $\mathbf{P^{-1}}$ need 76K NAND gates, and the normalization blocks ($\mathbf{N}$) uses 58.2K NAND gates. The logic resources of the decoder implementation are summarized in Table I. VHDL was used for the description of the hardware. Cadence RTL Compiler was used for the synthesis and SOC encounter for place and route of the design employing 90 nm CMOS standard cells.

After routing the design, the maximum frequency achieved is 238 MHz and the total area of the decoder is 16.12 $mm^2$ with a core occupation of 70%.

Since one iteration of the decoding algorithm takes $M + d_v \cdot seg$ clock cycles and considering 10 pipeline stages, 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, 9 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 Mbps using (2), highest than any other decoder proposed in literature for high order fields, to the best knowledge of the authors.

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 to the most efficient NB-LDPC decoder designs. Table II summarizes the results of different architectures found in literature, where the total gate count (NAND) is derived assuming that storing one memory bit requires the area of 1.5 NAND gates [9], [15], [16].

Since the decoders that have been compared are implemented under different CMOS technologies, we scale technology to include results over a 90 nm CMOS process using first order approximations [18]. Note that, we compare different algorithms under the same performance, so each one has a different number of iterations. Efficiency is calculated dividing the normalized throughput (for 90 nm technology) over area ratio (Millions of NAND gates).

Considering that only the approach presented in [16] includes post layout results, only comparisons with [16] can be made for the total decoder area given in $mm^2$. The proposed decoder outperforms by a factor of four compared to the one presented in [16] the area occupied by the decoder. On the other hand, our approach has three times higher throughput and fourteen times more efficiency than the decoder in [16]. Note that in [16] a QPSA approximation is presented, despite this it involves a high complexity check node requiring only 5 iterations to achieve the same performance than our proposal.

TABLE II

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

| Algorithm | Simplify-MS [15] | Trellis Max-log QSPA [16] | Min-Max [17] | RMM [9] | T-EMS [12] | 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 | 8.51M | 2.1M | 871K | 2.75M | 1.78M |
| $f_{clk}$ (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 |
| Efficiency 90 nm (Mbps/M-gates) | 115.5 | 26.2 | 50.9 | 176.8 | 176 | 371.3 |
| Area ($mm^2$) | - | 46.18 | - | - | - | 16.12 |

Comparing the proposed decoder to the approach presented in [15], we can see that the first one has more than eight times less latency, more than six times higher throughput and three times higher efficiency, although our decoder requires 37% more logic elements (NAND gates).

Until now, the decoder presented in [17] was the most efficient decoder for a Min-Max implementation without introducing approximations in the algorithm, even so, our approach outperforms it in area (17% less NAND gates), latency (almost 20 times lower), throughput (almost 10 times higher) and efficiency (almost 7.5 times higher). In addition, the proposed algorithm do not introduce simplifications on the original Min-Max algorithm ensuring that performance is not compromised.

Although the proposed decoder requires little bit more area than twice the one occupied by

the decoder presented in [9], our approach is more than twice efficient in term of throughput-over-area ratio. In addition, the proposed decoder has eight times less latency than [9], which makes it suitable for high speed implementations.

Recently, a Trellis EMS decoder was presented [12], where the authors introduce a low latency decoder achieving 484 Mbps of throughput. Thanks to the simplifications presented in this paper, our proposed decoder outperforms [12] in area (54% less), latency (43% less), throughput (36% higher) and efficiency (more than two times higher), without introducing any performance loss compared with [12].

## VII. Conclusions

In this paper we present a simplified Trellis Min-Max algorithm which improves both area and latency with respect to the most efficient decoders included in literature for high order fields. The outgoing check node 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 to the T-EMS decoder. Using the layered schedule with the proposed check node algorithm reduces the required maximum number of iterations to achieve a desired performance. On the other hand, since the proposed approach does not make approximations on the reliability values used for derive the check node messages, the performance of the algorithm does not show any performance degradation.

## Acknowledgment

This work was supported in part by the Spanish Ministerio de Ciencia e Innovación under Grant TEC2011-27916 and in part by the Universitat Politècnica de València under Grant PAID-06-2012-SP20120625. The work of F. García-Herrero was supported by the Spanish Ministerio de Educación under Grant AP2010-5178.

## References

[1] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] E. Li, K. Gunnam, and D. Declercq, "Trellis based Extended Min-Sum for decoding nonbinary LDPC codes," in *8th International Symposium on Wireless Communication Systems (ISWCS)*, 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 Transactions on Communications*, vol. 61, no. 7, pp. 2600–2611, 2013.

[9] 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. PP, no. 99, pp. 1–1, 2012.

[10] E. Li, "Décodeurs Haute Performance et Faible Complexité pour les codes LDPC Binaires et Non-Binaires," Ph.D. dissertation, École Nationale Supèrieure de l'électronique et de ses Applications, à l'Université de Cergy-Pontoise, 2012.

[11] 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 - [transactions papers]," *IEEE Transactions on Communications*, vol. 57, no. 6, pp. 1652–1662, 2009.

[12] E. Li, D. Declercq, K. Gunnam, F. García-Herrero, J. Lacruz, and J. Valls, "Low Latency T-EMS Decoder for NB-LDPC Codes," in *Conference Record of the Forty Seventh Asilomar Conference on Signals, Systems and Computers (ASILOMAR), 2013*, 2013.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] J. Lin and Z. Yan, "Efficient Shuffled Decoder Architecture for Nonbinary Quasi-Cyclic LDPC Codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 9, pp. 1756–1761, 2013.

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