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 and 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 for practical applications, especially for the check node processing, which limits the maximum achievable throughput. Although a great effort has been made in the recent literature to overcome this disadvantage, the proposed decoders are still 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 overall decoder architecture has been implemented in a 90 nm CMOS process for a (N=837,K=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 terms of throughput-gates ratio in 110% compared to the existing recent solutions for the same code.

Index Terms—NB-LDPC, T-Min-Max, Layered Decoder, Message Passing Algorithm.

I. INTRODUCTION

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

NB-LDPC are lineal block codes characterized by a sparse parity check matrix H with M rows and N columns. Each non-zero element \( h_{m,n} \) of H belongs to the Galois field \( GF(q=2^p) \). In this paper we only consider regular NB-LDPC codes with constant row weight \( d_r \) and column weight \( d_c \). 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 H, called variable nodes (VN). Decoding algorithms for NB-LDPC codes use iterative message exchange between CNs and VNs and vice-versa to estimate the most reliable codeword from the noisy received sequence.

Different decoding algorithms have been proposed since the Q-ary Sum Product algorithm (QSPA) [2]. The complexity of QSPA is too large to be suitable for hardware implementations, and several approaches such as FFT-SPA [3], log-SPA and max-log-SPA [4], were proposed to overcome its limitations. 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), has been proposed, where the complexity of the check node is reduced considerably involving only comparisons and additions. Later, the Min-Max algorithm was proposed [6], which uses comparisons to compute the maximum reliability values instead of additions, unlike the EMS algorithm. This new solution helps preventing the growth of the data length of the decoder without introducing any performance loss with respect to the EMS algorithm.

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

To further improve the T-EMS efficiency, we propose in this paper a simplification of this algorithm, by building the extra column of the trellis and generating the output messages of the 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 an efficient way improving both area and latency of the decoder. Additionally, for each configuration path, we define the path reliability using the maximum value instead of aggregating the symbol reliabilities. For this reason we named our 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 on codes with high order Galois fields, a \((N=837,K=726)\) NB-LDPC code over \(GF(32)\) has been selected. This code has been used in many preceding papers and serves then as a...
benchmark for comparing different implementations of NB-LDPC decoders. To the best of our knowledge, the proposed architecture based on TMM achieves 110% higher efficiency (Mbps/Million Gates) than the most efficient decoder proposed in literature [8], [9], for the same code. Moreover, the proposed design has lower latency and higher throughput than any proposed NB-LDPC decoder.

The rest of the paper is organized as follows: in Section II we recall the principles of the T-EMS algorithm. The proposed TMM architecture and its check node architecture are presented in Section III. Section IV describes the overall layered decoder architecture and the synthesis and post layout results. Finally, Section V includes comparisons with others proposed decoders and conclusions are outlined in Section VI.

II. TRELLIS EXTENDED MIN-SUM ALGORITHM

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

Let \( N(m) \) (\( M(n) \)) 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 respectively. For a symbol value \( a \in GF(q) \), \( Q_{m,n}(a) \) represents the \( a \)-th entry in vector \( Q_{m,n} \) and measures the extrinsic reliability of symbol \( a \) being equal to \( a \), seen from the check-node \( m \). Accordingly, \( L_{m}(a) \) denotes the channel information for symbol \( n \) and \( Q_{n}(a) \) its \textit{a posteriori} information.

Let \( c = c_1, c_2, \ldots, c_N \) and \( y = y_1, y_2, \ldots, y_N \) be the transmitted codeword and received noisy symbol sequence respectively. The log-likelihood ratio (LLR) for each received symbol is obtained as \( L_n(a) = \log \left[ P(c_n = z_n \mid y_n) / P(c_n = a \mid y_n) \right] \) 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.

We present in Algorithm 1 the T-EMS check node decoding unit where the first step consists in the delta domain transformation of input messages which are denoted by \( \Delta Q_{m,n}(\eta_j) \), being \( \eta_j = a + z_n \) the delta domain index. This transformation ensures that the most reliable messages are always in the first index of \( \Delta Q_{m,n}(\eta_j) \) and the rest of the symbols are reordered and considered as deviations of the most reliable one, according to step 1. Step 2 involves the computation of check node syndrome \( \beta \) using the most reliable symbol \( z_n \) for each check node incoming message. For the syndrome computation, all nonzero elements of \( H \) are taken as \( \delta^0 = 1 \) thanks to the pre-processing of the incoming messages outside of the node, as will be explained in later sections.

### Algorithm 1: T-EMS Algorithm

**Input:** \( Q_{m,n}, z_n = \arg \min_{a \in GF(q)} Q_{m,n}(a) \ \forall \ n \in N(m) \)

**Output:** \( R_{m,n} \)

1. \( \Delta Q_{m,n}(\eta_j = a + z_n) = Q_{m,n}(a) \)
2. \( \beta = \sum_{j=1}^d z_n \in GF(q) \)
3. \( \Delta Q(a) = \min_{\eta'_j(a) \in \text{conf}(m,n)} \sum_{j=1}^d \Delta Q_{m,n}(\eta'_j(a)), a \in GF(q) \)
4. \( R_{m,n}(a + \beta + z_n) = \lambda \cdot \Delta R_{m,n}(a), a \in GF(q) \)

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

Hereinafter, we only consider the case when \( n_r = 2 \) and \( n_c = 2 \) for T-EMS algorithm. In this case combinations with the two most reliable symbols are analyzed to build the extra column reliability. This means that combinations of min1-min1, min1-min2, min2-min1, 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. In addition, for the same path, no more than one reliability from the same stage of the trellis is considered [8].

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

Last step of T-EMS algorithm involves the inverse transformation from the delta domain to the normal domain, using the
In the T-EMS, the output message calculation (Step 4 of Algorithm 1) involves \( q \times d_c \) subtractions and also \( q \times d_c \) minimum finders (min finders) which becomes the bottleneck of the check node processing. Taking into account this drawback, we propose a simplified algorithm which reduces considerably the complexity of the check node unit and improving both latency and area of the global decoder architecture.

### III. SIMPLIFIED TRELLIS MIN-MAX ALGORITHM

In the T-EMS, the output message calculation (Step 4 of Algorithm 1) involves \( q \times d_c \) subtractions and also \( q \times d_c \) minimum finders (min finders) which becomes the bottleneck of the check node processing. Taking into account this drawback, we propose a simplified algorithm which reduces considerably the complexity of the check node unit and improving both latency and area of the global decoder architecture.

#### A. Algorithm Description

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

The simplification mentioned in the last paragraph takes advantage from the fact that only configurations with the most reliable message from each trellis index 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 trellis index. As a consequence, only combinations of \( \text{min1-min1} \) messages are considered leaving out combinations of \( \text{min1-min2}, \text{min2-min1} \) and \( \text{min2-min2} \).

Although this simplification results in a large reduction of the number of considered paths to build the output messages, we did not see any performance loss on the NB-LDPC codes we used for the simulations.

As explained in Section II, the extra column \( \Delta Q(a) \) contains the paths formed by the most reliable combination of 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 \( a \neq 0 \). Although this distance between configurations is theoretically computed using sums of the local reliability in the trellis, it has been proposed in the Min-Max algorithm [6] to use the maximum value of the considered path as an alternative measure of distance. We also make use of this idea in our algorithm design, and propose the use of the maximum operator instead of the addition to compute the extra column \( \Delta Q(a) \) reliability. Making use of the maximum value to measure distances prevents the data length growth associated to the summation, introducing an important area reduction due to the parallel processing of the trellis algorithm. In (1) the modifications made on step 3 of Algorithm 1 are presented, converting the T-EMS on T-Min-Max algorithm (TMM).

\[
\Delta Q(a) = \min_{\eta_j(a) \in \text{conf}(n_j,n_c)} \left\{ \max_{j=1-d_c} \left( \Delta Q_{mn}(\eta_j(a)) \right) \right\}, a \in GF(q) \tag{1}
\]

The complete description of the proposed TMM algorithm is presented in Algorithm 2, where step 3 of Algorithm 1 has been split into two basic tasks. In step 3 of Algorithm 2 function \( \psi \) 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 (\( m1_{\text{col}}(a) \)), so it can take values from one to \( d_c \).

Step 4 of Algorithm 2 involves the processing of the trellis extra column reliability using information related to the most reliable symbol \( m1(a) \). The configuration set \( \text{conf}(n_j,n_c) \) 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 \( \text{conf}^+(n_j,n_c) \), where the difference is that \( \eta_j(a) \) only retains information from the \( n_c \) columns where deviations from the zero-order configuration are made instead of keeping information from all nodes. This simplified storage
of configuration sets implies that \( k \) can take values from one to \( n_c \), and lead to a significant area gain. In the rest of paper, we will keep the same constraints for the decoder design, and restrict to the case in which only configurations with the most reliable message for each symbols \( a \) (\( n_c = 1 \)) and a maximum of two deviations (\( n_c = 2 \)) are considered.

Additionally, in the TMM algorithm, when \( \Delta Q(a) \) is formed by only one deviation, the corresponding \( \eta_1(a) \) and \( \eta_2(a) \) will have the same values. This situation contributes to simplifications in the hardware implementation of Algorithm 2 as we will see in next sections.

Step 5 of Algorithm 2 presents a simplified way to obtain delta domain output messages using a simple assignment of \( \Delta Q(a) \), \( m1(a) \) or \( m2(a) \) depending only on the deviation information from \( \eta_1(a) \). 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 \( \Delta R_{n,j}(a) \). 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 describes the transformation of delta domain messages to the normal domain, including a scaling factor \( \lambda \) to improve the performance of the decoder in the waterfall region. The scaling factor value \( \lambda \) is selected among the possible hardware friendly values that do not increase the area of the decoder.

\subsection*{B. Frame Error Rate Performance}

For testing the performance of our simplified TMM algorithm, simulations were conducted for a \((N=837,K=726)\) NB-LDPC code over \( GF(2^5) \) where \( H \) is generated using the methods in [12], with \( d_c = 27 \) and \( d_r = 4 \), and using transmission over BPSK modulation and AWGN channel. We compare the TMM to the QSPA [2], and the recently published Relaxed Min-Max (RMM) [9] and T-EMS algorithms [13]. On Fig. 1, we show the frame error rate (FER) simulation results for a layered scheduling.

The TMM algorithm in floating point (fp) simulation was made to be compared to T-EMS and QSPA 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 T-EMS algorithm the optimum \( \lambda \) was selected and 15 iterations (it) for the iterative decoding were used. For T-MM approach, \( \lambda \) value was set to 0.5, since this value doesn’t requires extra hardware for implementation purposes. Despite this, using the optimum \( \lambda \) value the performance of T-EMS of T-MM approach are very similar. In Fig. 1 we can see that the TMM has a negligible performance loss of 0.05dB compared to the T-EMS, despite the proposed simplifications.

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

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

\section*{IV. Check Node Architecture}

In this section, the design of the check node unit 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 the normal domain to the delta domain. This delta domain transformation is made using a permutation network similar to the one proposed in [14]. This network requires \( q \cdot \log_2(q) \) multiplexors of two inputs to perform the delta domain transformation of each input vector message \( Q_{m,n} \). 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 syndrome \( \beta \) is calculated adding all \( d_c \) tentative hard decision symbols. This is performed by means of a \( GF \) adder in a tree structure fashion.

The 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 [15]. Only \( q-1 \) cells are required to implement all \( \Psi \) functions, because in delta domain messages the most reliable symbols remains on \( \eta_j = 0 \) in the delta domain and their magnitudes are equal to zero. Each \( \Psi \) function requires \( d_c \) inputs because of the processing based on the trellis reordering of the delta messages. The approach followed to implement the \( \Psi \) function is the tree structure proposed on [15] since it provides a good tradeoff between area and latency.

Extra column reliability \( \Delta Q(a) \) 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, 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 in charge 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 arising from the same trellis stage are discarded, since only deviations from different stage 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 trellis stage of both reliabilities is the same, then the maximum value for the quantization scheme is assigned to the corresponding “1 min finder” input.

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

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

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

The proposed simplified structure for the output message generation is 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 \( \Delta R_{\text{m},n}(a) \) message takes its value from \( m1(a), m2(a) \) or \( \Delta Q(a) \) depending on the control signals obtained during extra column generation. The scaling factor \( (\lambda) \) 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.

In the following section, the TMM check node unit is integrated with the rest of the blocks, that depicts the entire decoder architecture based on a non-binary layered scheduling.

V. ARCHITECTURE FOR THE COMPLETE DECODER

In this section the top level design of the NB-LDPC decoder, which includes the CN proposed in Section III, is presented. The proposed decoder has been designed for QC-NB-LDPC codes over \( GF(q) \) constructed using the methods included in [12], where \( H \) is formed by \( (q - 1) \times (q - 1) \) circulant 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).

The decoding process starts loading the channel information \( L_n(a) \) on the variable nodes memories \( Q_n(a) \) 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 [14] for the flooding schedule. In this paper we address this idea for the horizontal layered schedule. To do this, in step 1 messages \( Q_n(a) \) are permuted depending on the corresponding nonzero \( H \) element \( h_{mn} \) to obtain \( Q_n(h_{mn}a) \). The permuted VN messages and the last iteration check node outgoing messages \( R_{\text{m}}^{(t-1)}(a) \) are processed to obtain \( \hat{Q}_{\text{m}}(a) \) which correspond to the outgoing 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 datapath.

### Algorithm 3: Layered schedule for proposed decoder

**Input:** \( L_n(a) = \log \frac{P(\text{c}=a|x_n)}{P(\text{c}=\bar{a}|x_n)} \)

**Initialization:**

\[
Q_n^{(1)}(a) = L_n(a), \quad R_{mn}^{(0)}(a) = 0, \quad t = 1
\]

**Main Loop:**

\[
\text{while } t \leq \text{MaxIter} \text{ do}
\]

\[
\begin{align*}
1 & \quad R_{mn}^{(t)}(a) = Q_n^{(t)}(h_{mn}a) - R_{mn}^{(t-1)}(a) \\
2 & \quad Q_n^{(t)}(a) = \min \{ Q_n^{(t)}(a) \} \quad \forall \ a \in GF(q) \\
3 & \quad Q_n^{(t)}(a) = Q_n^{(t)}(a) - Q_n^{(t)}(a) \\
4 & \quad R_{mn}^{(t)}(a) = \text{TMM} \{ Q_n^{(t)}(a) \in \mathcal{N}(m) \} \\
5 & \quad Q_n^{(t+1)}(h_{mn}a) = R_{mn}^{(t)}(a) + Q_n^{(t)}(a) \\
6 & \quad t = t + 1
\end{align*}
\]

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

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 update on step 5 of Algorithm 3. In this step, inverse permutation of \( Q_n(a) \) messages have to be done before processing of a new row of \( H \). The decoding process stops when the maximum number of iterations is reached, then the output codeword \( \hat{c} \) 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 decoder addresses the case of quasi-cyclic NB-LDPC codes build from [12], the \( Q_n(a) \) messages can be grouped on sets with \( q - 1 \) messages. In total, assuming \( w \) quantification bits for the messages, \( d_c \) memories with \( q - 1 \) positions of \( q \cdot w \) bits are required for \( Q_n(a) \). Only one message is read and one is written in the same clock cycle from each memory during the processing of one row of \( H \).

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

Block \( N \) in Fig. 5 implements the normalization included in steps 2 and 3 of Algorithm 3. This block includes a one minimum finder which searches the most reliable value to
and \( q \) messages. The decoder requires \( d \) shift registers (SRL) that synchronize with the permuted VN R check node messages at the end of the decoding process.

c input for the check node (to recover the minimum. The recovered symbol is used as reliable symbol of each Q advantage of the one minimum finder to obtain the most 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.

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 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 pipeline stages used in the decoder). For this purpose \( d_c \) SRL are required.

LLR of the received sequence is initially stored in “LLR Mem.” memories (Fig. 5) and then extracted to be loaded on VN memories \( (Q_n(a)) \) when the decoding process starts. \( d_c \) memories are required with \( q-1 \) positions and \( q \cdot w \) 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 \( H \), only the coefficients of the first row of each circulant sub-matrix need to be saved. For doing this \( d_c \) small memories with \( d_c \) elements of \( p \) bits are added.

C. Decoder Timing

The decoding process starts loading the channel information on \( Q_n(a) \) memories, this process consumes \( q-1 \) clock cycles. 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 do not modify the \( Q_n(a) \) messages. Control signals are included to this end.

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

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

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

D. Decoder Complexity and Implementation Results

As it has been explained before, the decoder was implemented for a \((N=837, K=726)\) NB-LDPC code over \( G(F(8)) \), with parity check matrix \( H \) 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. The registering points have been selected to balance the critical path, to this ends the segmentation points are listed below:

1) At the output of the P block (Fig.5).
### Table I

**COMPLEXITY ANALYSIS FOR THE PROPOSED DECODER. FOR THE**

(N=837, K=726) **NB-LDPC CODE OVER GF(32)**

<table>
<thead>
<tr>
<th>Logic Gates (NAND)</th>
<th>Memory bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Node</td>
<td>222K</td>
</tr>
<tr>
<td>Permutations (P and P^−1)</td>
<td>76K</td>
</tr>
<tr>
<td>Normalization (N)</td>
<td>58.2K</td>
</tr>
<tr>
<td>Add/Sub</td>
<td>46.7K</td>
</tr>
<tr>
<td>Q(a)</td>
<td>-</td>
</tr>
<tr>
<td>R''(n−1) a)</td>
<td>-</td>
</tr>
<tr>
<td>LLR mem</td>
<td>-</td>
</tr>
<tr>
<td>Code Out mem</td>
<td>-</td>
</tr>
<tr>
<td>Qmn(a)</td>
<td>-</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>402.9K</strong></td>
</tr>
</tbody>
</table>

2) Inside the N block (Fig.5) that normalizes the incoming messages at the check node.

3) At the input of the check node processor (Fig.5).

4) At the output of the normal domain to delta domain converter (Fig. 2).

5) In the third stage of the two minimum finder (Fig.2).

6) At the output of the two minimum finder (Fig.2).

7) In the second stage of the one minimum finders (Fig.3).

8) At the output of the extra column reliability values calculation (Fig.3).

9) At the output of the check node processor (Fig.5).

10) At the output of the P^−1 block (Fig.5).

Outside the check node, the permutation networks P and P^−1 need 76K NAND gates, and the normalization blocks (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 a 90nm CMOS process of nine layers with standard cells and operating conditions of 25°C and 1.2V.

After routing, the maximum frequency achieved is 238 MHz and the total area of the decoder is 16.12 mm² with a core occupation of 70%.

Since one iteration of the decoding algorithm takes \( M + d_v \cdot d_e \cdot d_{seg} \) 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), which is very much higher than all existing solutions from the SoA, as shown in the next section, except from the T-EMS [8], [13].

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

### VI. Comparisons With Other NB-LDPC Decoders

The proposed decoder has been compared to the most efficient NB-LDPC decoder designs. Table II summarizes the results of different architectures found in literature. Results included in Table II are computed considering that all the memory bits were implemented as RAM memories, in which one bit of memory is equivalent to 1.5 NAND gates [9], [16], [17]. However, the number of equivalent NAND gates that we obtained based on the layout area after the place and route report (16125908 \( \mu m^2 \)) and the core occupation (70%), using the libraries of our CMOS process, is \((0.7 \times 16125908 \mu m^2)/3.136\mu m^2 = 3.599M \) NAND gates \(^1\). This is due to the fact that the memory bits for \( R''(n−1) a) \) (535.7K), the check node (31K) and the \( Q_{mn}(a) \) (51.8K) were implemented as registers which have an equivalent area of 4.5 NAND gates instead of 1.5 NAND gates, because we do not have fully customized memories for these sizes.

Since the decoders of table II are implemented under different CMOS technologies, we scale the technology to show results over a 90 nm CMOS process using first order approximations [19] based on the ratio of the maximum achievable frequency for the different processes. To this end, the scaling factors used to deriving the comparisons presented in Table II are 1.66 and 2.33 for 130nm/90nm and 180nm/90nm scaling respectively. 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).

Comparing our decoder to the approach presented in [16], 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).

In [18], the authors have presented a very efficient implementation of the Min-Max decoder. Our solution 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). The Min-Max decoder has also been modified to improve its hardware efficiency in [9]. Although the TMM requires twice the area occupied by the decoder presented in [9], our approach is more than two times more efficient in term of throughput-over-area ratio. In addition, our TMM decoder shows eight times less latency than [9], which makes it suitable for high speed implementations.

Finally, a Trellis EMS decoder was presented [13], 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 [13] in area

\(^1\)The equivalent area of a NAND gate is 3.136\(\mu m^2\).
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)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>Synthesis</td>
<td>Post-layout</td>
<td>Synthesis</td>
<td>Synthesis</td>
<td>Post-layout</td>
<td>Post-layout</td>
</tr>
<tr>
<td>Technology</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>180 nm</td>
<td>90 nm</td>
<td>130 nm</td>
<td>180 nm</td>
<td>90 nm</td>
<td>90 nm</td>
</tr>
<tr>
<td>Quantization (w)</td>
<td>5 bits</td>
<td>7 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>6 bits</td>
<td>6 bits</td>
</tr>
<tr>
<td>Gate Count (NAND)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_clk (MHz)</td>
<td>1.29M</td>
<td>8.51M</td>
<td>2.1M</td>
<td>871K</td>
<td>2.75M</td>
<td>1.78M</td>
</tr>
<tr>
<td>Iterations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>5</td>
<td>13</td>
<td>13</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>Latency (clock cycles)</td>
<td>12995</td>
<td>4460</td>
<td>28215</td>
<td>12675</td>
<td>2160</td>
<td>1507</td>
</tr>
<tr>
<td>Throughput (Mpbs)</td>
<td>64</td>
<td>223</td>
<td>64</td>
<td>66</td>
<td>484</td>
<td>660</td>
</tr>
<tr>
<td>Throughput (Mpbs)</td>
<td>149</td>
<td>223</td>
<td>107</td>
<td>154</td>
<td>484</td>
<td>660</td>
</tr>
<tr>
<td>Efficiency 90 nm (Mpbs/M-gates)</td>
<td>115.5</td>
<td>26.2</td>
<td>50.9</td>
<td>176.8</td>
<td>176</td>
<td>371.3</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td></td>
<td>46.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(54% less), latency (43% less), throughput (36% higher) and efficiency (more than two times higher), under the same FER performance.

VII. CONCLUSIONS
In this paper we have presented 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. The improvements proposed in this paper on the hardware implementation of the T-EMS, including the replacement of the addition by a maximum operator (to derive the TMM algorithm) intends to keep the feature of a very high speed implementation, but with a maximum hardware complexity reduction.

ACKNOWLEDGMENT
The authors would like to thank Dr. Erbao Li and Dr. Kiran Gunnam for their discussions on this topic, which helped to improved the paper.

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.

This work has been partially funded by the Institut Universitaire de France.

The work of F. García-Herrero was supported by the Spanish Ministerio de Educación under Grant AP2010-5178. This work has been also partially funded by the Institute Universitaire de France.

REFERENCES

