Reduced-complexity Non-Binary LDPC decoder for high-order Galois fields based on Trellis Min-Max algorithm

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

Member, IEEE

Abstract

Non-binary LDPC codes outperform its binary counterparts in different scenarios. However, they require a considerable increase in complexity, especially in the check-node processor, for high-order Galois fields higher than GF(16). To overcome this drawback, we propose an approximation for the Trellis Min-Max algorithm which allows us to reduce the number of exchanged messages between check node and variable node compared to previous proposals from literature. On the other hand, we reduce the complexity in the check-node processor, keeping the parallel computation of messages. We implemented a layered scheduled decoder, based on this algorithm, in a 90nm CMOS technology for the (837,723) NB-LDPC code over GF(32) and the (1536,1344) over GF(64), achieving an area saving of 16% and 36% for the check-node and 10% and 12% for the whole decoder, respectively. The throughput is 1.07 Gbps and 1.26 Gbps, which outperforms the state-of-the-art of high-rate decoders with high GF order from literature.

Index Terms

NB-LDPC, T-MM, Message Compression, Layered Schedule, Check Node Processing, High Speed, High Rate, VLSI Design

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

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

I. Introduction

Non-binary Low-Density Parity-Check (NB-LDPC) codes are a promising kind of linear block codes defined over Galois Fields $GF(q = 2^p)$ with $p > 1$. NB-LDPC codes have numerous advantages over its binary counterparts, including better error correction performance for short/medium codeword length, higher burst error correction capability and improved performance in the error-floor region.

The main disadvantage of NB-LDPC codes is the high complexity of the decoding algorithms and derived hardware architectures, which limit their application in real scenarios where high throughput and reduced silicon area are important requirements.

Davey and MacKay [1] rediscovered LDPC codes defined over Galois Fields GF($q = 2^p$) with $p > 1$ with the introduction of the Q-ary Sum-of-Product Algorithm (QSPA) as an extension of the binary LDPC decoding based on belief propagation. Since then, several advances have been made to reduce the complexity of the decoders.

Improvements based on QSPA, such as Fast Fourier Transform SPA (FFT-SPA) [2], log-SPA and max-log-SPA [3], reduce the computational load of the parity-check equations without introducing any performance loss. The recently proposed Trellis Max-Log-QPSA [4] algorithm improves considerably both area and decoding throughput compared to previous solutions based on QPSA, making use of a path construction scheme to generate the output message in the check-node (CN) processor. These solutions offer the highest coding gain for high-rate NB-LDPC codes, but at the same time, include costly processing that limits their application in real communication and storage systems.

Extended Min-Sum (EMS) [5] and Min-Max [6] algorithms were proposed with the aim of reducing the complexity offered by solutions based on QPSA. In these algorithms, the CN equations are simplified by making approximations to involve only additions and comparisons in their parity-check equations. Since both algorithms make use of forward-backward (FB) metrics in the CN processor, the maximum throughput is bounded due to serial computations. The number of exchanged messages between CN and Variable Node (VN) for both algorithms is $n_m \times d_c$, where n_m is a fraction of q total reliabilities, being $n_m \ll q$ and d_c the CN degree. Therefore, the number of messages between nodes is lower than previous solutions from literature.

To avoid the use of FB metrics, Trellis Extended Min-Sum (T-EMS) algorithm [7, 8] was proposed. The input messages are organized in a trellis, including an extra column on it, to enable the generation of CN output messages in parallel. On the other hand, Trellis Min-Max (T-MM) algorithm [9] improves both algorithm and architecture compared to T-EMS from [7, 8]. One Minimum Only TMM (OMO-TMM) [10] is an approximation of T-MM that reduces the complexity of the CN by obtaining only one minimum and estimating the second one. Both, T-EMS and T-MM, do not introduce any performance loss compared to EMS and Min-Max algorithms, respectively. Moreover, the derived hardware architectures improve in area and speed with respect to other proposals from literature based on algorithms from [5, 6]. The main drawbacks of T-EMS, T-MM and OMO-TMM are: i) the high number of exchanged messages between CN and VN ($q \times d_c$ reliabilities), which impacts in the wiring congestion, limiting the maximum throughput achievable; ii) the high amount of storage elements required in the hardware implementations of these algorithms, which supposes the major part of the decoder's area.

To overcome the drawbacks of T-EMS and T-MM, the proposal in [11] introduces a technique of message compression that reduces the wiring congestion between CN and VN and the storage elements used in the derived architectures. The messages at the output of the CN are reduced to four elementary sets which include the intrinsic and extrinsic information, the path coordinates and the hard-decision symbols. The information exchanged between processors is reduced from $q \times d_c$ reliabilities to $4 \times (q-1) + d_c$ messages without introducing any performance loss. A step further was taken in [12], where the mT-MM algorithm was proposed. This algorithm reduces the cardinality of the intrinsic information to only two elements, and the rest $q - 2$ values are approximated by a constant value. The information exchanged between processors is reduced to $3 \times (q - 1) + d_c$ messages but at the cost of some performance loss.

In this paper we take as starting point the solution from [11] to propose a novel algorithm which reduces the messages that include the intrinsic information and the path coordinates from $(q-1)$ values to only L messages each one, being $L < n_m \ll q$. This improvement allows us to pass from the number of messages exchanged in [11] to only $(q - 1) + 3 \times L + d_c$, saving area in the decoder thanks to the reduction of the memory requirements. This reduction of messages introduces a performance loss in the coding gain that can be controlled by means of the parameter L. In a second step, we introduce a novel method to generate the L most reliable values of the intrinsic set, reducing considerably the CN complexity compared to previous solutions from literature [8, 9, 11]. The low size of this set allows us to propose a simplified network that calculates the L most reliable values for the intrinsic information. These values are sent to the

VN. The proposed network greatly reduces the area required by the extra column processor from [8, 9, 11], which is the bottleneck of the implemented CN processors. Our proposal allows the design of high-rate NB-LDPC decoders over GF(32) and GF(64) without prohibitive areas. For the code (1536,1344) NB-LDPC code over GF(64) the area saving in the CN is about 36% and 15% considering the overall decoder compared to solutions from [11], with a performance loss of 0.1dB . In terms of throughput, the increase is about 17.5% compared to the design from [11]. For the (837,726) NB-LDPC code over GF(32) the area saving in the CN is about 16% and 10% for the overall decoder, introducing a performance loss of 0.08dB and a gaining in throughput of 10% compared to [11]. In both cases, we implemented a layered scheduled decoder because the aim of the paper is to obtain high-throughput decoders for codes with large Galois Field. For other efficient decoders not focused in high throughput we refer to [13].

The rest of the paper is organized as follows: Section II includes the basis on NB-LDPC codes and T-MM algorithm implemented using compressed messages. Section III includes the proposed approximation to reduce the CN output messages for T-MM algorithm and describes a novel way to obtain the most reliable intrinsic information without analyzing the entire trellis. Section IV includes the hardware implementation for the proposed check node architecture. The implementation of a layered scheduled decoder and comparison with other proposals from literature are devised in Section V. Finally, conclusions are presented in Section VI.

II. T-MM DECODING ALGORITHM WITH COMPRESSED MESSAGES

A sparse parity-check matrix $\mathbf{H}$ defines a NB-LDPC code, where each non-zero element $h_{m,n}$ belongs to a Galois field $GF(q = 2^p)$. Another common way to characterize NB-LDPC codes is by means of a Tanner graph [14], where two kinds of nodes are differentiated representing all N columns (variable nodes, VN) and M rows (check nodes, CN) of $\mathbf{H}$. $\mathcal{N}(m)$ denotes the set of VNs connected to a CN m and $\mathcal{M}(n)$ denotes the set of CNs connected to a VN n, therefore, the cardinality of the sets corresponds to d_c and d_v, respectively.

Let's consider a message $\mathbf{m} \in GF(q)^K$ which is coded to $\mathbf{c} = \mathbf{m} \times \mathbf{G}$, where $\mathbf{G}$ is the generator matrix that satisfies $\mathbf{G} \cdot \mathbf{H}^{\mathbf{T}} = \mathbf{0}$, being $\mathbf{0}$ the zero matrix of size $K \times M$. Using Binary Phase Shift Keying (BPSK) signalling, the codeword $\mathbf{c}$ is transmitted over a binary input Additive White Gaussian Noise (AWGN) channel. The received sequence is $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $\mathbf{e}$ is the error vector introduced by the noisy communication channel.

Algorithm 1: Layered schedule

```
Inicialization:
    $Q_n^{(0)}(a) = L_n(a),\ R_{m,n}^{(0)}(a) = 0,\ t = 1$
Main Loop:
    while $t \leq Iter$ do
        for $l = 1$ to $M$ do
1               $Q_{m,n}(a) = Q_n^{(t-1)}(h_{m,n}a) - R_{m,n}^{(t-1)}(a)$
2               $R_{m,n}^{(t)}(a) = \phi\left(Q_{m,n}(a)\right)$
3               $Q_n^{(t)}(h_{m,n}^{-1}a) = R_{m,n}^{(t)}(a) + Q_{m,n}(a)$
        end
4       $\tilde{c}_n = \arg\min\left(Q_n^{(t)}(a)\right)$
5       if $\tilde{\mathbf{c}} \times \mathbf{H^T} = 0$ then break
        else $t = t + 1$
    end
Output: $\tilde{\mathbf{c}} = [\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_N]$
```

NB-LDPC codes are decoded applying iterative algorithms where messages that represent reliability values are passed from VN to CN and vice versa. Basically, two types of scheduling are used: i) Flooding, where first all CN are processed and then all VN are updated based on the CN output messages and the channel information; ii) Layered, where one CN is processed and then all connected VN are updated, so, the process is repeated until all CN are processed. In this paper we consider layered schedule since it offers a better trade-off between complexity and decoding speed and for its higher convergence compared to the flooding schedule [15]. Algorithm 1 includes the basic steps involved in the layered schedule of NB-LDPC decoding.

The initialization step requires to extract the *a priori* information from the communication channel to compute the log-likelihood ratio (LLR). This is obtained by means of $L_n(a) = \log[P(c_n = z_n|y_n)/P(c_n = a|y_n)]$. Additionally, a normalization is made to ensure that all the LLR values are non-negative, $L'_n(a) = |L_n(a) - L_n(z_n)|$, being z_n the hard-decision symbols associated to the highest reliability. LLRs are loaded in the VN which is represented by the set $Q_n(a)$. This set corresponds to the *a posteriori* information which is updated as the decoding algorithm progresses, as can be seen in Step 3 of Algorithm 1.

Messages from VN to CN are denoted as $Q_{m,n}(a)$ and are calculated using the VN information $Q_n(a)$ and the CN to VN messages $R_{m,n}(a)$ (Step 1, Algorithm 1). CN output messages $R_{m,n}(a)$ are calculated using function ϕ. This function varies depending on the algorithm applied for the decoding. If the tentative codeword $\tilde{\mathbf{c}}$, calculated in Step 4, satisfies the parity-check equation, then the decoding process stops outputting $\tilde{\mathbf{c}}$ as a valid codeword, or else the process is repeated until the maximum number of iterations ($Iter$) is reached.

Trellis Min-Max (T-MM) algorithm [9] was proposed as a new implementation of Min-Max from [6] that allows the parallel processing of messages in the CN and reduces the complexity. Applying a message compression technique [16], the basic steps of T-MM in the CN and the number of exchanged messages are further reduced without introducing any performance loss compared to the original T-MM algorithm.

In the compressed version of the T-MM algorithm, instead of sending $q \times d_c$ $R_{m,n}(a)$ messages to the VN processor, the information in the CN is organized in four elementary sets called $I(a)$, $E(a)$, $P(a)$ and z_n^*.

$I(a)$ is the set related to the intrinsic information sent to the VN processor. This set is calculated applying (1) to the most reliable CN input messages in delta domain [7], $m1(\eta(a))$.

$$I(a) = \min_{\eta(a) \in \; conf^*(1,2)} \left\{ \max\left(m1(\eta(a))\right) \right\} \tag{1}$$

$conf^*(n_r, n_c)$ [9] is the configuration set which selects the possible paths conformed by the n_r symbols with higher reliability value. From all the possible paths, the configuration set only selects the ones that deviate at most n_c times from the hard-decision path[1]. From this reduced set of possible paths, the one selected from the corresponding $I(a)$ value is the one that ensures the highest reliability (minimum value). In this paper we consider the case where $n_r = 1$ and $n_c = 2$. So, only the most reliable messages are considered (first minimum set $m1(a)$) and only one and two deviation paths are taken into account.

The set $E(a)$ is related to the extrinsic information. It is composed of $m1(a)$ or $m2(a)$ (second minimum set) messages depending on the number of deviations of the path used to form $I(a)$

[1]The hard-decision path is the one formed only by messages corresponding to the symbol $\alpha^{-\infty}$, which in delta domain corresponds to the reliability of z_n

according to (2).

$$E(a) = \begin{cases} m2(a) & \text{if} \quad I(a) \rightarrow \text{one deviation} \\ m1(a) & \text{otherwise} \end{cases} \tag{2}$$

The set $P(a)$, with $n_c \times (q-1)$ values, is used to keep track of the column where deviations take place, when the values of the set $I(a)$ are generated. This information is used in two situations: first, to select the proper values for the set $E(a)$ depending on the deviation information when the set $I(a)$ is computed; second, it is used at the VN to generate the $q \times d_c$ reliabilities as will be seen next.

Finally, the hard-decision symbols defined as $z_n = \arg\min_{a \in GF(q)} Q_{mn}(a)$ and the syndrome $\beta = \sum_1^{d_c} z_n$ are used to generate the hard-decision symbols $z_n^* = z_n + \beta$ required for delta-to-normal domain transformation.

At the VN processor a decompression of messages is made to generate the $R_{m,n}(a)$ values used to obtain the *a posteriori* information $Q_n(a)$ [9]. The decompression operations are made following (3).

$$R_{m,n}(a + z_n^*) = \begin{cases} I(a) & \text{if} \quad P(a,1) \neq n \ \textbf{and} \ P(a,2) \neq n \\ E(a) & \text{otherwise} \end{cases} \tag{3}$$

III. T-MM ALGORITHM WITH REDUCED SET OF MESSAGES

In this Section we introduce a novel method to reduce the number of messages exchanged between CN and VN compared to the proposal from [11]. First, we define the reduced set of compressed messages that are sent from CN to VN and an approximation to obtain the rest of values in the VN. Second, the performance of the method is analyzed. Third, a technique to generate the most reliable values of the set $I(a)$ without building a complete trellis structure is presented.

A. Reduction of the CN-to-VN messages

The sets $I(a)$ and $P(a)$ are required to generate the messages $R_{m,n}(a)$ at the VN processor, as can be seen in (3). Reducing the cardinality of $I(a)$, the one of $P(a)$ is also reduced.

Our proposal is to keep the L most reliable values of $I(a)$ and the corresponding ones of $P(a)$ and $E(a)$, being $L < (q-1)$. Consider the set $I^*(a') = \{I^*(a'_1),\ I^*(a'_2),\ \dots\ , I^*(a'_L)\}$, as the L most reliable values from the set $I(a)$ and $a' = \{a'_1, a'_2, \dots, a'_L\}$ are their corresponding GF symbols. On the other hand, consider the sets $E^*(a') = E(a) \; \forall \; a \in a'$ and $P^*(a') = P(a) \; \forall \; a \in a'$.

Defining the complementary set $a'' \in a \setminus a'$, we propose to set $E^*(a'') = m1(a'')$. So, the cardinality of the set $E^*(a)$ is kept in $q-1$. Table I includes the number of bits of each one of the sets exchanged from CN to VN processors compared to the proposal from [11], where w is the number of bits used to quantize the reliabilities.

TABLE I
NUMBER OF BITS REQUIRED TO BE EXCHANGED FROM CN TO VN PROCESSOR

	Number of bits	
	T-MM [11]	Proposed
$I(a)/I^*(a)$	$(q-1) \times w$	$L \times w$
$E(a)/E^*(a)$	$(q-1) \times w$	$(q-1) \times w$
z_n^*	$d_c \times p$	$d_c \times p$
$P(a)/P^*(a)$	$2 \times (q-1) \times \lceil \log d_c \rceil$	$2 \times L \times \lceil \log d_c \rceil$

As an example consider the (837,726) NB-LDPC code over GF(32) ($d_c = 27, d_v = 4$) and the (1536,1344) NB-LDPC code over GF(64) ($d_c = 24, d_v = 3$) built using the methods from [17]. For the first code the number of bits at the CN output is 817 bits using $w = 6$ bits with the method from [11], while for our proposal the number of bits is only 385, so there is a reduction of 53%. For the second code, the method from [11] outputs 1530 bits, while our proposal only exchanges 586 bits, which corresponds to a reduction of 62% in the number of bits. The L value was set to four in these examples.

Since the cardinality of the sets $I^*(a)$ and $P^*(a)$ has been reduced compared to $I(a)$ and $P(a)$, respectively, it is no longer possible to generate the messages $R_{m,n}(a)$ using (3) at the VN.

For the symbols a' is possible to construct $L \times d_c$ values for $R^*_{m,n}(a')$ using (4). It is easy to see that $R^*_{m,n}(a') = R_{m,n}(a) \; \forall \; a \; \in \; a'$.

$$R^*_{m,n}(a' + z^*_n) = \begin{cases} I^*(a') & \textbf{if } P^*(a',1) \neq n \\ & \quad \textbf{and } P^*(a',2) \neq n \\ E^*(a') & \textbf{otherwise} \end{cases} \tag{4}$$

For the complementary set of symbols a'', it is necessary to propose a function that approximates the reliabilities of the messages $R_{m,n}(a'')$ due to the cardinality reduction of the sets $I(a)$ and $P(a)$. Therefore, we introduce a novel way to obtain the messages $R_{m,n}$ corresponding to the symbols a''. This uses an approximation function based on an offset and a scaled version of the set $E^*(a'')$ as expressed in (5).

$$R^*_{m,n}(a'' + z^*_n) = \gamma_1 \times E^*(a'') + \gamma_2 \times I^*(a'_L) \tag{5}$$

Even considering that the scaling factors γ_1 and γ_2 are constant values, the offset $I^*(a'_L)$ and the set $m1(a'')$ depend on the specific CN input messages at each iteration. This fact introduces a self-adjusted term for the approximated values of $R_{m,n}(a'')$.

B. Performance Analysis

To show the behaviour of the set $R^*_{m,n}(a)$ compared to $R_{m,n}(a)$ in an implementation of T-MM algorithm [9], we computed histograms for the sets $R^*_{m,n}(a)$ and $R_{m,n}(a)$. We tested several NB-LDPC codes over different Galois field and degree distribution, for various Eb/No values and taking 10^6 repetitions for each configuration. We achieved similar results in all cases.

In Fig. 1 we present the results for the (837,726) NB-LDPC code over GF(32) [17]. Eb/No was set to 4.3dB, $\gamma_1 = \gamma_2 = 0.5$ in (5) and $L = 4$ for this example. In this figure, the x-axis includes the arranged reliabilities for the sets $R^*_{m,n}(a)$ and $R_{m,n}(a)$, where index 0 corresponds to the symbol with the highest reliability and indexes 1 to 4 are related to the reliabilities filled with (4) considering $L = 4$. As can be seen, for indexes 1 to 4 the values for the set $R^*_{m,n}(a')$ are equal to the ones for the set $R_{m,n}(a)$ for the same indexes, since (4) corresponds to (3) for $a \in a'$. For indexes 5 to 31, we observe that the approximation introduced in (5) underestimates the mean values of $R^*_{m,n}(a'')$ compared to the ones from $R_{m,n}(a)$. Even so, the tendency of the reliability values is similar. The γ_1 and γ_2 values were adjusted by means of Bit Error Rate (BER) simulations, considering hardware-friendly values for the sake of simplicity of hardware implementations.

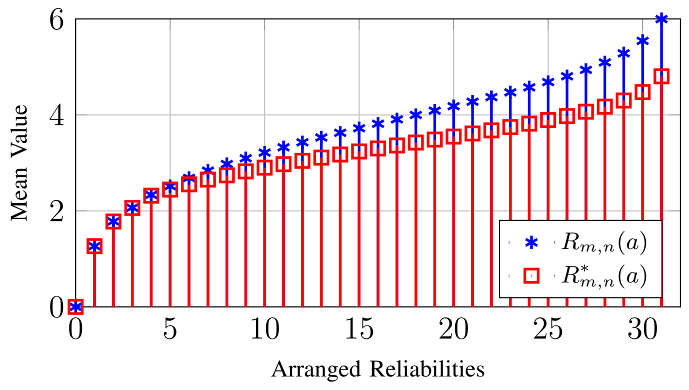


Fig. 1. Mean values for each reliability in the set $\Delta Q(a)$. The values were arranged in the x axis. The code under test is the (837,726) NB-LDPC code over GF(32).

In order to test our proposal and reduce the number of exchanged messages between CN and VN, we performed BER simulations to compare it to the conventional T-MM algorithm. The code under test was the (837,726) NB-LDPC code over GF(32) [17]. It can be seen in Fig. 2 that an increment of the parameter L (more exchanged messages from CN to VN) is translated into a BER performance closer to the conventional implementation of T-MM algorithm. It is observed an improvement of almost 0.2dB in the coding gain increasing L from $L = 2$ to $L = 4$, 0.05dB from $L = 4$ to $L = 6$ and almost negligible when passing from $L = 6$ to $L = 8$. We also include in Fig. 2 the BER performance for $L = 4$ and 8 decoding iterations for the quantized model (6 bits) to ease comparisons with other proposals in Section V.

The same analysis was made for the (1536,1344) NB-LDPC code over GF(64) varying the L parameter. The BER performance is presented in Fig. 3. It can be seen that the performance losses are greater than the ones from Fig. 2 for small L values, comparing both to the conventional T-MM algorithm [9]. This is due to the percentage of reliabilities approximated using (5), which is 87.5% for the GF(32) code and 93.75% for the GF(64) code, considering $L = 4$ for both cases. It will be seen in Section IV that the performance loss of 0.1dB for $L = 4$ introduced with our approach is compensated with an important reduction in the complexity of the check node.

C. Generation of the set $I^*(a')$

In Section III-A a method to reduce the number of messages sent from CN to VN was presented. It was shown that modifying the parameter L, the performance loss compared to

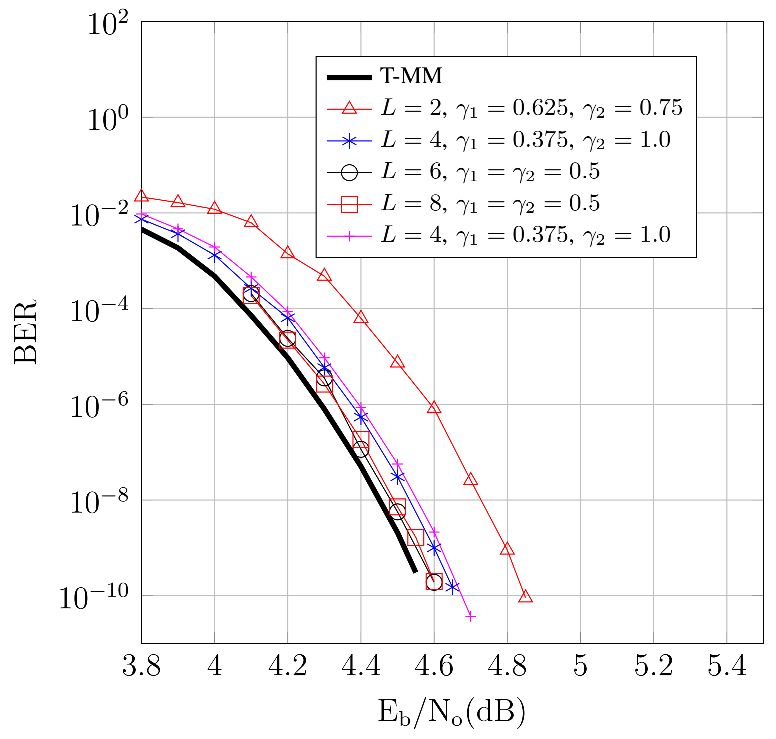


Fig. 2. Bit Error Rate performance for our proposal varying the L parameter compared to T-MM algorithm. The code under test is the (837,726) NB-LDPC code over GF(32). 15 decoding iterations and floating point model are considered in all cases except for the last curve where 8 iteration and 6 bits are employed.

T-MM algorithm [9] can be tuned. On the other hand, a method to approximate the discarded messages of the set $I(a)$ was introduced using (5). The maximum performance loss is set to 0.1dB, so we fix $L = 4$ in the rest of the paper. In this way, the performance loss is 0.08dB for the (837,726) NB-LDPC code over GF(32) and 0.1dB for the (1536,1344) NB-LDPC code over GF(64).

From the analysis made in Section III-A, it is easy to see that even reducing considerably the number of exchanged messages from CN to VN, the CN has to calculate the entire set $I(a)$ using (1) and the set $P(a)$ before selecting the L most reliable values from them. In this paper we propose a method to obtain the L most reliable values without using (1) nor introducing any approximation. Our method takes advantage of the $\min$-$\max$ operator involved in (1). The $\min$-$\max$ operator is used to obtain the reliability value among the reliabilities selected by the configuration set, for each symbol a. Examining how the $\min$-$\max$ operator behaves to obtain the L most reliable symbols, it is possible to extract some rules to avoid the implementation

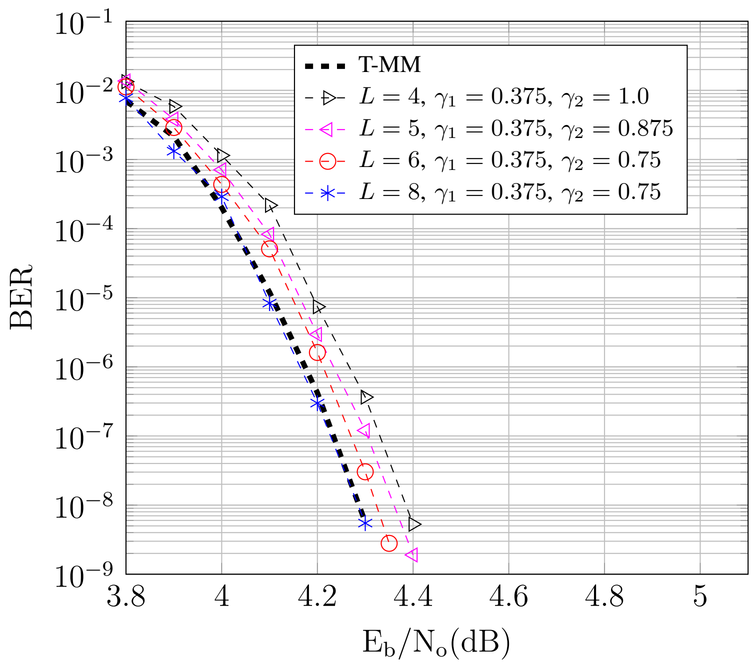
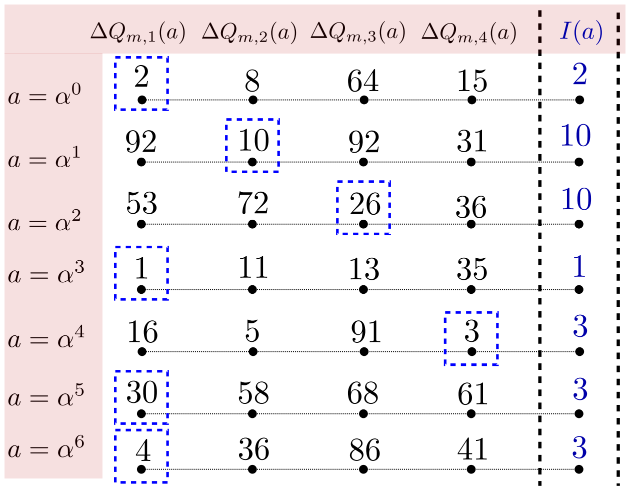


Fig. 3. Bit Error Rate performance for our proposal with different values of L compared to T-MM algorithm. The test code is the (1536,1344) NB-LDPC code over GF(64). 15 decoding iterations and floating point model are considered for both algorithms

of a complete trellis structure. In Fig. 4 an example for the set $\Delta Q_{m,n}(a)$ (GF(8), $d_c = 4$) is presented, where the most reliable messages per row are marked with a dashed square. The rightmost column includes the set $I(a)$ formed by combination of the $m1(a)$ values following (1). This example will be used to explain the method to obtain the set $I^*(a')$, composed of the $L = 4$ most reliable values of the set $I(a)$.

First, consider the absolute minimum, $m1_1$, among all the $m1(a)$ reliabilities, in the example from Fig. 4 $m1_1 = 1$. $m1_1$ will appear on the set $I(a)$ only in one-deviation paths, because in the two-deviation cases, $m1_1$ will be discarded by the $\max$ operator when all the possible paths for each symbol a are analysed. On the other hand, there is only one "one-deviation" path for each symbol a, so, in the example of Fig. 4, for the symbol α^3, the one-deviation path corresponds to $m1_1$. In fact, this path is the most reliable among all the possible ones for α^3. Then, instead of analysing all the possible paths to obtain the most reliable value of the set $I(a)$ ($I^*(a'_1)$), we only have to assign the value $I^*(a'_1) = m1_1$ and retain the value of the corresponding symbol $a'_1 = a_{m1_1} = \alpha^3$.

A similar analysis can be done to find the second most reliable value of the set $I(a)$. This



	$\Delta Q_{m,1}(a)$	$\Delta Q_{m,2}(a)$	$\Delta Q_{m,3}(a)$	$\Delta Q_{m,4}(a)$	$I(a)$
$a=\alpha^0$	$\boxed{2}$	8	64	15	2
$a=\alpha^1$	92	$\boxed{10}$	92	31	10
$a=\alpha^2$	53	72	$\boxed{26}$	36	10
$a=\alpha^3$	$\boxed{1}$	11	13	35	1
$a=\alpha^4$	16	5	91	$\boxed{3}$	3
$a=\alpha^5$	$\boxed{30}$	58	68	61	3
$a=\alpha^6$	$\boxed{4}$	36	86	41	3

Fig. 4. Example of the sets $\Delta Q_{m,n}(a)$ and $I(a)$ for GF(8) and $d_c = 4$

value can be obtained assigning the second minimum of $m1(a)$ ($m1_2 = 2$ in the example from Fig. 4), so, $I^*(a'_2) = m1_2$ and $a'_2 = a_{m1_2} = \alpha^0$. Note that there can be a two-deviation path that gives the same reliability value as $m1_2$, this is the combination of $m1_1$ and $m1_2$ if they belong to different columns ($m1_{1_{\text{col}}} \neq m1_{2_{\text{col}}}$). In this case, the reliability of this two-deviation path corresponds to $m1_2$ due to the $\max$ operation involved in (1).

The selection of the third most reliable value of $I(a)$ ($I^*(a'_3)$) requires a comparison between multiple candidates, which includes the one-deviation path formed with $m1_3$ and the two-deviation path made with the combination of $m1_1$ and $m1_2$. The two-deviation path will be selected for $I^*(a'_3)$ ($I^*(a'_3) = m1_2$ and $a'_3 = a_{m1_1} + a_{m1_2}$) unless $m1_1$ and $m1_2$ belong to the same column of $\Delta Q_{m,n}(a)$. In that case, the reliability selected is $m1_3$ ($I^*(a'_3) = m1_3$ and $a'_3 = a_{m1_3}$). In the example from Fig. 4, since $m1_1 = 1$ and $m1_2 = 2$ belong to the same column of the trellis ($n = 1$), $m1_2$ can not be used for $I^*(a'_3)$. Instead of this, the selected reliability is $I^*(a'_3) = m1_3 = 3$ and $a'_3 = a_{m1_3} = \alpha^4$.

For $I^*(a'_4)$, we consider the candidates listed in Table II with the priority given in its leftmost column. The conditions to select a reliability are listed in the rightmost column of Table II. Basically, the conditions ensure that a value will not be selected if another one with higher reliability has been used for a symbol $a'_i \ \forall\ i \ \in 1, 2, \ldots, L$ and, on the other hand, for the two-deviation cases, no more than one deviation is made on each stage of the trellis [7, 9].

Following with the example in Fig. 4 and the priority and conditions listed in Table II for the possible candidates for $I^*(a'_4)$, the highest priority candidate (OD, $m1_3$) must be discarded

TABLE II

POSSIBLE CANDIDATES FOR THE $I^*(a'_4)$ RELIABILITY

Priority	Involved Reliabilities	$I(a'_4)$	One (OD) / Two (TD) deviation path	Condition to be selected
1º	$m1_3$	$m1_3$	OD	Not been used for $I^*(a'_3)$ and $a_{m1_1} + a_{m1_2} \neq a_{m1_3}$
2º	$m1_3$, $m1_1$	$m1_3$	TD	$a_{m1_3} + a_{m1_1} \neq a_{m1_2}$ and $m1_{3_{\text{col}}} \neq m1_{1_{\text{col}}}$
3º	$m1_3$, $m1_2$	$m1_3$	TD	$a_{m1_3} + a_{m1_2} \neq a_{m1_1}$ and $m1_{3_{\text{col}}} \neq m1_{2_{\text{col}}}$
4º	$m1_4$	$m1_4$	OD	-

since it was used for the $I^*(a'_3)$ reliability. Next, we select the one with the second priority since it meets the conditions from Table II. Thus, $I^*(a'_4) = m1_3$ and the corresponding symbol $a'_4 = a_{m1_1} + a_{m1_3} = \alpha^6$.

The conditions derived to obtain the L most reliable values of the set $I(a)$ can be mapped directly in a hardware structure, avoiding a complete analysis of the trellis. The CN architecture is presented in next section.

The proposed CN decoding algorithm is summarized in Algorithm 2. Step 1 corresponds to the delta-domain transformation [18] of the CN input messages, $Q_{m,n}(a)$, using the tentative hard-decision symbols z_n. The syndrome β is calculated adding, in the GF domain, all z_n symbols (Step 2). Step 3 finds the two-minimum among the d_c input messages in delta-domain for each symbol a. The position of the first minimum, $m1_{col}(a)$, is also retained. A L-min finder for the set $m1(a)$ is included in Step 4. Function ψ selects the L values for the set I^*, as detailed in this section. Step 6 includes the conditions to select the values of the set $E(a)$, as explained in Section II.

IV. CHECK NODE ARCHITECTURE

In this section we present the architecture for the CN processor based on the proposed method. It includes a network to calculate, in an efficient way, the $L = 4$ most reliable messages of the set $I(a)$, using the conditions explained in Section III-C.

Algorithm 2: Proposed check-node decoding algorithm

```
Input: $\mathbf{Q_{mn}}$
       $z_n = \arg\min_{a \in \text{GF}(q)} Q_{mn}(a)\ \forall\ n \in \mathcal{N}(m)$
1 $\Delta Q_{mn}(a + z_n) = Q_{mn}(a)$
2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in \text{GF}(q)$
3 $[m1(a), m1_{col}(a), m2(a)] = 2\text{-min}\{\Delta Q_{m,n_i}(a)\Big|_{i=1}^{d_c}\}$
4 $[m1^*, m1^*_{col}, a'] = L\text{-min}\{m1(a)\}$
5 $\left[I^*, I^*_{path}, I^*_{sym}\right] = \psi\{m1^*, m1^*_{col}, a'\}$
6 $E(a) = \begin{cases} m2(a) & \text{if } \ I(a) \rightarrow \text{one deviation} \\ m1(a) & \text{otherwise} \end{cases}$
Output: $\begin{cases} I^*, I^*_{path}, I^*_{sym} \\ E(a) \\ z^*_n = z_n + \beta \quad \forall\ n\ \in\ \mathcal{N}(m) \end{cases}$
```

The top-level block diagram for the proposed CN is detailed in Fig. 5. The CN input messages are $\mathbf{Q_{m,n}}$, which come from the VN processor, and the tentative hard decision symbols $\mathbf{z}$. Both input messages are used to compute the Normal-to-Delta domain transformation ($\mathbf{N} \rightarrow \boldsymbol{\Delta}$ block in Fig. 5). d_c transformation networks are needed in the CN, each one requires $q \times \log(q)$ w-bit MUX following the approach proposed in [19], where w is the number of bits for the data-path.

$\mathbf{z}$ is also used to obtain the syndrome β adding all d_c tentative hard-decision symbols. This operation requires $w \times (d_c - 1)$ XOR gates. β is used to generate the new hard-decision symbols $\mathbf{z}^*$, which are sent to the VN to generate the $R^*_{m,n}$ messages using (4). $\mathbf{z}^*$ symbols are generated using GF(q) adders which require $d_c \times w$ XOR gates to implement them.

Two-minimum finders obtain the two most reliable messages for each GF(q) symbol over the delta-domain values ($\Delta Q_{m,n}$). The search of $\alpha^{-\infty}$ is excluded, since it corresponds to the hard-decision symbols, with the highest reliability (zero-value). So, in the CN processor there are $q - 1$ two-minimum finders where the position of the first minimum values is also extracted to obtain the set $I^*(a')$, as explained in Section III-C. Implementation is done by means of

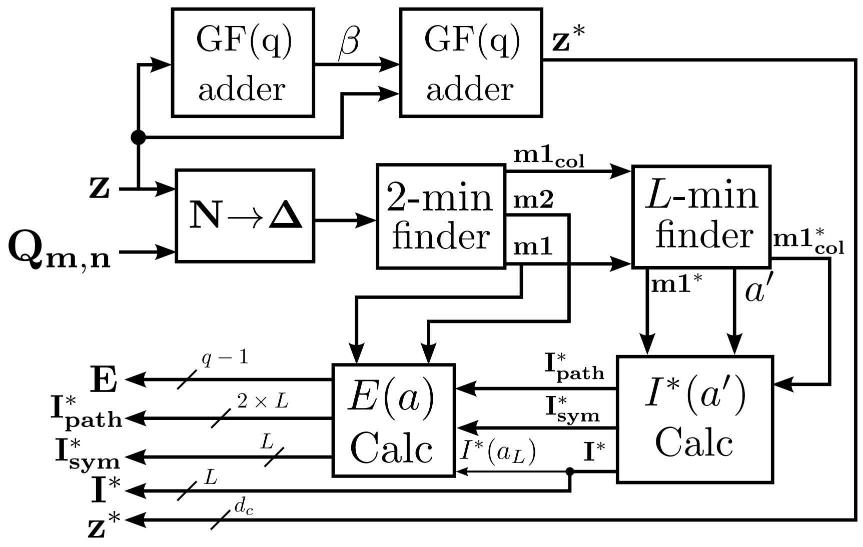


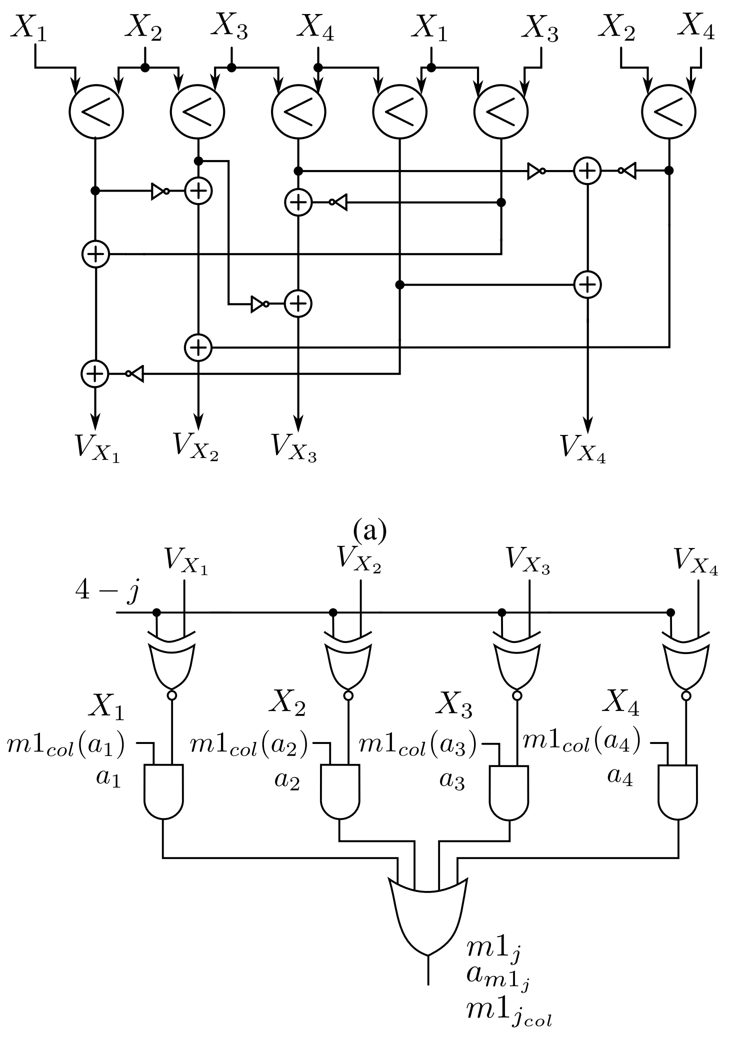
Fig. 5. Proposed check-node block diagram

tree-based two-minimum finders, following the approach from [20]. Each finder has d_c inputs, implemented with $2 \times d_c$ w-bit comparators and $3 \times d_c$ w-bit MUXES.

A L-min finder is used to obtain the L most reliable values of the set $m1(a)$, $m1(a')$ ($m1^*$ in Fig. 5), outputted from the 2-min finder. We propose to use a parallel sorting approach for the implementation with the aim of improving speed at the CN processor. The proposed architecture is presented in Fig. 6, where an example for four inputs is included. It is based on a two stage circuit: first (Fig. 6.a), we compute comparisons between all the combination of input pairs (X_i, X_j) $\forall$ $i \neq j$ and, then, we add the output of the comparators for each one of the inputs. The main idea is to count the number of times that an input X_i is lower than the other $N-1$ inputs, being V_{X_i} the number of times and N the number of inputs of the network. The greater the V_{X_i} value, the lower X_i is. So, the second stage (Fig. 6) is responsible to find the value V_{X_i} corresponding to the minimum that we are looking for. For example, the $m1_1$ value corresponds to the one with $V_{X_i} = N-1$, since it is lower than the rest of inputs. So, $m1_2$ corresponds to $V_{X_i} = N-2$ and so on for the rest of $m1_j$ values which corresponds to $V_{X_i} = N-j$.

The proposed CN architecture requires a structure as the one in Fig. 6.a operating with $q-1$ inputs. Since we particularize the CN for the case where $L = 4$, we require four selection networks from Fig. 6.b, one for each $m1_j$ value.

The implementation of the structure from Fig. 6.a requires $(q-1)$ $\times (\frac{q-2}{2})$ w-bit comparators.



(b)

Fig. 6. (a) First stage of the proposed L-min finder. (b) Circuit to extract the j-th minimum value. Example for four inputs.

The number of adders is summarized in Table III for different field orders.

Four structures as the one in Fig. 6.b, considering $L = 4$, need $4 \times (q-1) \times p$ XNOR gates, $4 \times (q-1) \times (w + 2 \times p + \lceil \log d_c \rceil)$ AND gates, assuming that the symbols a' and columns $m1_{col}(a')$ from the L most reliable $m1(a)$ values must be retained to be used in the calculation of the $I^*(a')$, as can be seen in the block diagram from Fig. 5. Finally, $4 \times q \times (w + p + \lceil \log d_c \rceil)$ OR gates complete the logic elements required in the implementation of the circuit.

The solution from Fig. 6 to the L-min finder offers a high-speed structure that does not compromise the latency of the overall CN processor.

TABLE III

ADDERS REQUIRED TO IMPLEMENT THE CIRCUIT FROM FIG 6.A

Field size (q)							# ADD	bits
256	128	64	32	16	8	4	$q \times q/2$	1 bit
							$q \times q/4$	2 bit
						▨	$q \times q/8$	3 bit
					▨	▨	$q \times q/16$	4 bit
				▨	▨	▨	$q \times q/32$	5 bit
			▨	▨	▨	▨	$q \times q/64$	6 bit
		▨	▨	▨	▨	▨	$q \times q/128$	7 bit
	▨	▨	▨	▨	▨	▨	$q \times q/256$	8 bit

The set $I^*(a')$ is generated using the circuit presented in Fig. 7 which is a direct implementation of the method explained in Section III-C. It uses the outputs of the L-min finder as inputs to obtain the sets $I^*(a')$, $I^*(a')_{path}$ and $I^*(a')_{sym}$.

As can be seen in Fig. 7, the generation of the set $I^*(a')$ requires few hardware resources which can be easily summarized in $15 \times p + 3 \times w + 17 \times \lceil \log d_c \rceil + 6$ equivalent NAND gates. For the (837,726) NB-LDPC code over GF(32) this corresponds to 184 NAND gates and 216 NAND gates for the (1536,1344) NB-LDPC code over GF(64). The increase of the field order does not increment significantly the number of required gates compared to the structure that generates the extra column $\Delta Q(a)$ in the proposal from [9], which is unsuitable for fields higher than GF(32).

The reliabilities of the set $E^*(a)$ are generated using the circuit from Fig. 8. The portion of the circuit rounded by dashed lines is repeated for each GF symbol. The generation of the set $E^*(a)$ requires $(q-1) \times (23 \times w + 6 \times p) + 3 \times \lceil \log d_c \rceil$ equivalent NAND gates. To compare our proposed CN architecture with a conventional implementation of T-MM algorithm [11], we synthesized the design using Cadence Register Transfer Level (RTL) compiler for the (837,726) NB-LDPC code over GF(32) and the (1536,1344) NB-LDPC code over GF(64). It can be seen in Table IV that the area saving is almost doubled for the GF(64) NB-LDPC code compared to

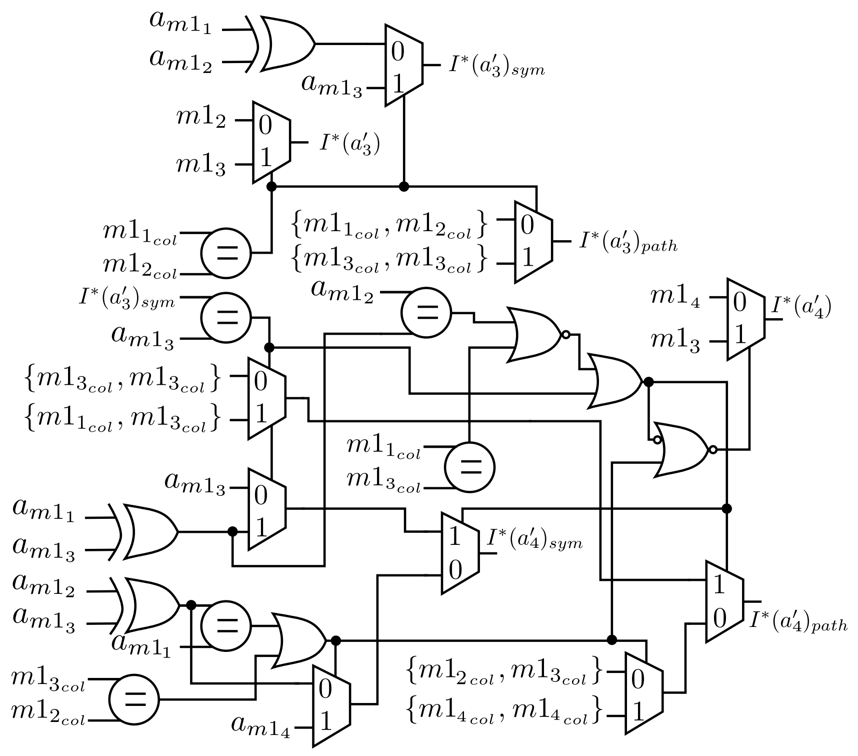


Fig. 7. Circuit to generate the set $I^*(a')$

the GF(32) case. This is due to the reduction of complexity in the $I^*(a)$ generation that is the bottleneck in the CN implementation from [11].

TABLE IV
SYNTHESIS RESULTS FOR THE PROPOSED CN ARCHITECTURE

	Equivalent NAND gates		Saving
	[11]	Proposed	
(837,726) NB-LDPC code over GF(32)	154806	133273	16.16 %
(1536,1344) NB-LDPC code over GF(64)	423144	309938	36.52%

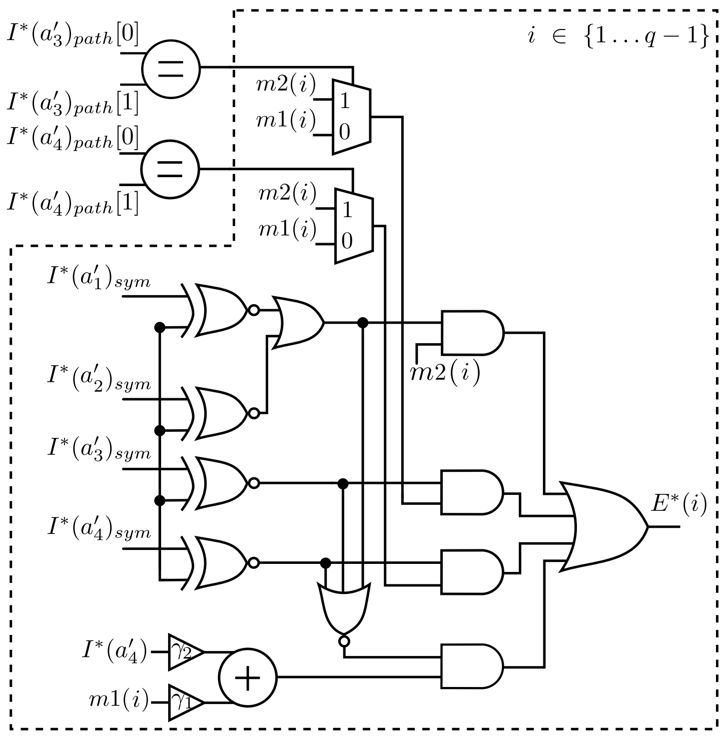


Fig. 8. Circuit to generate the set $E^*(a')$

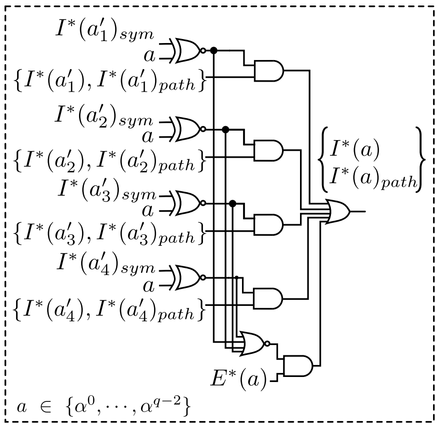
V. Top-level decoder architecture and complexity comparison

In this section we include the proposed CN architecture in a layered decoder with a similar structure to [16].

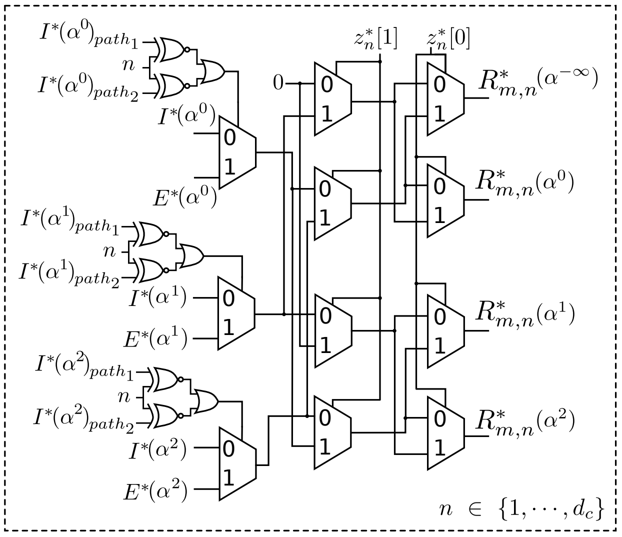
The decompression network generates the set $R^*_{m,n}(a)$ and implements (4) and (5) using the structures presented in Fig. 9. The circuit from Fig. 9.a generates a $(q-1)$-length set $I^*(a)$ from the reduced set $I^*(a')$. Once the set $I^*(a)$ is obtained, the circuit from Fig. 9.b is used to generate the set $R^*_{m,n}(a)$ performing the Normal-to-Delta domain transformation from the sets $I^*(a)$, $E^*(a)$ and the new hard-decision symbols z^*_n.

The decoder requires $2 \times (q-1)$ circuits as the one from the left-side in Fig. 9, each one uses $[27 \times \log d_c + 14 \times w + 6 \times p]$ equivalent NAND gates. On the other hand, it requires $2 \times d_c$ circuits as the one presented on the right-side of Fig. 9 using $q \times ((p+1) + 2 \times \log d_c + 1)$ equivalent NAND gates each one of them.

One of the main benefits of reducing the number of messages exchanged from CN to VN is that the number of registers required to store the CN output messages from one iteration to the


$I^*(a'_1)_{sym}$
a
$\{I^*(a'_1), I^*(a'_1)_{path}\}$
$I^*(a'_2)_{sym}$
$\{I^*(a'_2), I^*(a'_2)_{path}\}$
$I^*(a'_3)_{sym}$
$\{I^*(a'_3), I^*(a'_3)_{path}\}$
$I^*(a'_4)_{sym}$
$\{I^*(a'_4), I^*(a'_4)_{path}\}$
$\{I^*(a), I^*(a)_{path}\}$
$E^*(a)$
$a \in \{\alpha^0, \cdots, \alpha^{q-2}\}$

(a)



(b)

Fig. 9. Proposed decompression network circuits. (a) Circuit to generate the set $I^*(a)$. (b) Circuit to generate the set $R^*_{m,n}(a)$, an example with GF(4)

next one are greatly reduced compared to conventional implementations of T-MM algorithm [9], which store $M \times q \times d_c \times w$ information bits. Our proposal only requires $M \times [(q-1) \times w + 4 \times (w + 2 \times \log d_c + p) + d_c \times p]$ registers.

A. Decoder implementation results and comparisons

The complete decoder architecture based on the CN architecture explained in Section IV was implemented on a 90nm CMOS process with nine metal layers and operating conditions 1.2V and 25^oC. VHDL was used for the description of the hardware and Cadence tools were used for synthesis and implementation of the proposed approach. To show the efficiency of our proposal for high-rate NB-LDPC codes over high-order fields, we present results for the (1536,1344) NB-LDPC code over GF(64). In order to simplify comparisons with other proposals from literature, we include results for the (837,726) NB-LDPC code over GF(32). Both QC-codes have been constructed using the methods from [17]. The throughput is obtained as:

$$Throughput = \frac{f_{clk} \times N \times p}{iter \times (M + d_v \times seg) + qQC},$$

where qQC is the size of the circulant sub-matrices which conform $\mathbf{H}$ and seg corresponds to the pipeline stages used in the design. For the both codes we choose $seg = 16$ to achieve a balance between throughput and area.

TABLE V

IMPLEMENTATION RESULTS FOR THE (1536,1344) NB-LDPC CODE OVER (GF(64) IN A 90NM CMOS PROCESS.

	T-MM [9]	T-MM CNBMP[11]	OMO-TMM [10]	mT-MM [12]	[This work] $L = 4$	[This work] $L = 5$
Report	Synthesis	Synthesis	Synthesis	Synthesis	Post-layout	Post-layout
Quantization (w)	6 bits	6 bits	6 bits	6 bits	6 bits	6 bits
Gate Count (NAND)	4.88M	3.34M	4.63M	4.05M	2.97M	2.99M
f_{clk} (MHz) Synthesis	250	300	250	300	351	351
f_{clk} (MHz) Post-layout	192	231	192	231	271	265
Iterations	8	8	8	8	8	8
Throughput (Mbps) Post-layout	874	1049	874	1049	1259	1231
Efficiency (Mbps/ Million NAND)	179	314	189	259	424	412
Area(mm^2)	-	-	-	-	28.90	29.09

To the best authors' knowledge, we present the first post-layout results for a high-rate NB-LDPC code over GF(64). As fas as the authors' knowledge, the best high-throughput decoder implementation for GF(64) is presented in [21]. It includes a chip implementation for a full-parallel decoder based on the (160,80) NB-LDPC code over GF(64) with degree distribution $(d_c = 4, d_v = 2)$ using a 65nm CMOS process. The reported gate count is 2.78M reaching a throughput of 1221Mbps (881Mbps for 90nm). A direct comparison is not possible because this is not a high-rate code (the rate is only 0.5) and our code has a rate of 0.875, furthermore, it is about 10 times shorter than the one we use (960 bits per codeword compared to 9216 bits in our code).

In order to compare our decoder with previous proposals implementing the same code, we synthesized the designs from [9–12] for the GF(64) code. We could not obtain post-layout results due to the high gate count of the designs.The results are summarized in Table V, where we also show the implementation results of our decoder for $L = 4$ and $L = 5$. The implementation for $L = 5$ was done by the extrapolation of the architecture for $L = 4$. Comparing the implementation for $L = 4$ and $L = 5$, the increment in area and the reduction of throughput are both about 1%. On the other hand, there is a coding gain of 0.02dB with this increment in L. Comparing our decoder for $L = 4$ with the others proposals in Table V, it can be seen that the highest reduction in the gate account is about 61% compared to the work from [9], and the lowest is 12% compared to the proposal from [11]. In order to make fair comparisons in terms of throughput, it is important to remark that the clock frequency (f_{clk}) usually reduces its value after placing and routing the design. For example, our proposal achieves f_{clk} = 351 MHz after synthesis and this value is lowered to 271 MHz after the place and route stage, which corresponds to a reduction of 23%. Thus, the post- synthesis throughput of the other works is reduced in the same percentage and showed in Table V. Considering these values, our work would outperform them between 30.6% and 16.6%, thanks to the reduction of complexity in the CN processor and the minimization of messages exchanged between CN and VN, which mitigates the routing congestion.

In terms of efficiency measured as the ratio between throughput (Mbps) and number (million) of equivalent NAND gates, our approach outperforms the one from [9] in almost 2.4 times. Compared to the design from [11], our proposal outperforms it in 35%.

Table VI compares the implementation results of the proposed decoder ($L = 4$) with other state-of-the-art proposals for the (837,726) NB-LDPC code over GF(32). The number of iterations in all the proposals listed in Table VI was adjusted to achieve similar performance at $\mathrm{E_b/N_o} = 4.4$ dB. As can be seen, our proposal outperforms most of the other approaches in both area and throughput. In terms of gate count, despite he fact that [23] requires 21% less gates, our work achieves a throughput which is almost seven times higher due to the parallel processing used in the CN. Compared to the proposal from [12], our approach has similar throughput and outperforms it almost 6% in area, thanks to the reduction of complexity in the CN with the hardware structures presented in Section. IV.

In terms of efficiency, our approach is five times most efficient that the proposals from [9, 23] and almost 9 times higher than the decoder from [22]. Compared to the design from [12], our novel proposals offers 8.6% higher efficiency.

TABLE VI

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

Algorithm	Simplify-MS [22]	Trellis Max-log QSPA [4]	RMM [23]	T-MM [9]	T-MM CNBMP [11]	OMO-TMM [10]	mT-MM [12]	[This Proposal]
Report	Synthesis	Post-layout	Synthesis	Post-layout	Post-layout	Post-layout	Post-layout	Post-layout
Technology	180 nm	90 nm	180 nm	90 nm	90 nm	90 nm	90 nm	90 nm
Quantization (w)	5 bits	7 bits	5 bits	6 bits	6 bits	6 bits	6 bits	6 bits
Gate Count (NAND)	1.29M	8.51M	871K	3.28M	1.25M	1.79M	1.13M	1.06M
f_{clk} (MHz)	200	250	200	238	300	250	345	393
Iterations	15	5	15	9	8	8	8	8
FER @ $\mathrm{E_b/N_o} = 4.4$ dB	2×10^{-4}	5×10^{-5}	9×10^{-5}	9×10^{-5}	1×10^{-4}	9×10^{-5}	1×10^{-4}	1×10^{-4}
Throughput (Mbps)	64	223	66	660	981	818	1080	1071
Throughput (Mbps) 90 nm	149	223	154	660	981	818	1080	1071
Efficiency (Mbps / Million NAND gates)	115.5	26.2	176.8	201.2	784.8	457	923	1010.4
Area (mm^2)	-	46.18	-	14.75	10.6	16.1	8.97	9.80

VI. CONCLUSIONS

In this paper we introduce an approximation for the T-MM algorithm to reduce the complexity of the CN architecture, which was the bottleneck in previous solutions from literature. This reduction allow us to offer post-layout results for high-rate NB-LDPC codes over GF(64) without prohibitive areas and higher throughput than the existing proposals, at the expense of some performance loss.

ACKNOWLEDGMENT

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

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[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, 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.

[8] 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.

[9] J. Lacruz, F. Garcia-Herrero, D. Declercq, and J. Valls, "Simplified Trellis Min-Max Decoder Architecture for Nonbinary Low-Density Parity-Check Codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 9, pp. 1783–1792, Sept 2015.

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

[11] J. Lacruz, F. Garcia-Herrero, and J. Valls, "Reduction of Complexity for Nonbinary LDPC Decoders With Compressed Messages," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 11, pp. 2676–2679, Nov 2015.

[12] J. Lacruz, F. Garcia-Herrero, M. Canet, and J. Valls, "High-Performance NB-LDPC Decoder With Reduction of Message Exchange," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–12, 2015.

[13] Y.-L. Ueng, C.-Y. Leong, C.-J. Yang, C.-C. Cheng, K.-H. Liao, and S.-W. Chen, "An Efficient Layered Decoding Architecture for Nonbinary QC-LDPC Codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 2, pp. 385–398, Feb 2012.

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

[15] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 976–996, 2003.

[16] J. Lacruz, F. Garcia-Herrero, M. Canet, J. Valls, and A. Perez-Pascual, "A 630 Mbps non-binary LDPC decoder for FPGA," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 1989–1992.

[17] B. Zhou, J. Kang, S. Song, S. Lin, K. Abdel-Ghaffar, and M. Xu, "Construction of non-binary quasi-cyclic LDPC codes by arrays and array dispersions," *IEEE Transactions on Communications*, vol. 57, no. 6, pp. 1652–1662, 2009.

[18] 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.

[19] 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.

[20] 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.

[21] Y. S. Park, Y. Tao, and Z. Zhang, "A Fully Parallel Nonbinary LDPC Decoder With Fine-Grained Dynamic Clock Gating," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 2, pp. 464–475, Feb 2015.

[22] 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.

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