A 630 Mbps Non-Binary LDPC Decoder for FPGA

J. O. Lacruz
Electrical Engineering Department
Universidad de Los Andes. Mérida, Venezuela
E-mail: jlacruz@ula.ve

F. García-Herrero, M. J. Canet, J. Valls and A. Pérez-Pascual
Instituto de Telecomunicaciones y Aplicaciones Multimedia
Universitat Politècnica de València. Gandia, Spain
E-mail: fragarh2@epsg.upv.es, {macasu, jvalls,asperez}@eln.upv.es

***Abstract*—A high-speed non-binary LDPC decoder based on Trellis Min-Max algorithm with layered schedule is presented. The proposed approach compresses the check-node output messages into a reduced set, decreasing the number of messages sent to the variable node. Additionally, the memory resources from the layered architecture are reduced. The proposed decoder was implemented for the (2304,2048) NB-LDPC code over GF(16) on a Virtex-7 FPGA and in a 90 nm CMOS process. Our implementation outperforms state-of-the-art NB-LDPC decoder implementations for both technologies, achieving a throughput of 630 and 965 Mbps, respectively.**

I. Introduction

Non-Binary Low-Density Parity-Check (NB-LDPC) codes emerge as alternative to their binary counterparts in scenarios where short/medium codeword length codes and better performance at high signal-to-noise ratios (SNR) are required. Additionally, they improve burst error correction capability, especially with high order Galois fields. On the other hand, the main drawbacks of NB-LDPC codes are: i) the high complexity of their check-node (CN); ii) the large amount of area spend on storage (RAM memories and registers); and iii) the routing congestion that limits the overall decoding throughput.

NB-LDPC codes were first investigated by Davey and MacKay [1], as an extension of binary LDPC codes. Since then, great efforts have been made to reduce the complexity of the original Q-ary Sum-of-Product Algorithm (QSPA) [1]. Extended Min-Sum (EMS) [2] and Min-Max [3] algorithms were proposed as approximations of the QSPA [1], reducing considerably the CN complexity. However, EMS and Min-Max algorithms are unable to reach high throughput because of the use of forward-backward (FB) metrics on the CN processor.

Recently, Trellis EMS (T-EMS) algorithm [4] [5] was proposed. It enables the parallel processing of messages at the CN and increases the throughput in comparison with decoders that use FB metrics. The main disadvantage of T-EMS algorithm is that the CN complexity is still high due to the parallel processing and, thus, it leads to a large area decoder. Simplified Trellis Min-Max (T-MM) algorithm [6] was proposed with the aim of reducing the CN complexity of T-EMS algorithm without compromising the decoding performance. Despite the advantages of T-MM compared with its predecessors, the area required is still high due to thelarge amount of storage elements, specially when layered schedule is applied.

In this paper we propose a NB-LDPC decoder architecture for T-MM algorithm which requires many less memory elements than the conventional implementation of this algorithm. The main idea is to minimize the messages exchanged between CN and VN processors. Thus, we remove any redundant information and only keep the minimum set of values required to reconstruct all the messages at the VN processor. The proposed decoder architecture is implemented on a Virtex-7 FPGA for a (2304,2048) NB-LDPC code over GF(16) [7]. It needs 83% less memory resources in comparison with a conventional implementation of T-MM algorithm [6] without introducing any performance loss. The throughput achieved is 630 Mbps, outperforming state-of-the-art NB-LDPC decoders implemented on FPGA devices [8] [9] [10].

The rest of the paper is organized as follows: Section II reviews the basis of T-MM algorithm, in Section III the check node and the top-level decoder architecture are derived and implementation results for FPGA and ASIC are presented. Finally, conclusions are outlined in Section IV.

II. Basis on NB-LDPC codes and T-MM decoding algorithm

NB-LDPC codes are linear block codes defined by a sparse parity-check matrix **H** with M rows and N columns, where each non-zero element $h_{m,n}$ belongs to Galois field $GF(q = 2^p)$. We consider regular NB-LDPC codes with constant row weight d_c and column weight d_v. Each row (column) of **H** is associated to a check node CN (variable node VN). $Q_{m,n}(a)$ and $R_{m,n}(a)$ denote the exchanged messages from VN to CN and from CN to VN for each symbol $a \in GF(q)$, respectively. $\mathcal{N}(m)$ and $\mathcal{M}(n)$ denote the sets of non-zero elements per row and column in **H**, respectively.

Trellis Min-Max (T-MM) algorithm [6] calculates the output CN reliabilities by organizing the $\Delta Q_{m,n}(a)$ messages in a trellis and including an extra column $\Delta Q(a)$ which enables the parallel processing in the CN processor. $\Delta Q_{m,n}(a)$ is the delta domain information defined as $\Delta Q_{m,n}(a + z_n) = Q_{m,n}(a)$, where $z_n \; \forall \; n \in \mathcal{N}(m)$ are the tentative hard-decision symbols.

In order to represent the trellis in a CN, the reliability information is organized in a matrix with the GF symbols in its rows and the $n \in \mathcal{N}(m)$ in its columns. Therefore, once the delta domain is applied, the most reliable symbols are located in the first row of the trellis, which is the hard-decision path. T-MM requires the computation of the two most reliable messages per row in $\Delta Q_{m,n}(a)$. The most reliable values, $m1(a)$, are used to compute the extra column values using (1).

$$\Delta Q(a) = \min_{a' \in conf^*(1,2)} \left\{ \max\left(m1(a')\right) \right\} \tag{1}$$

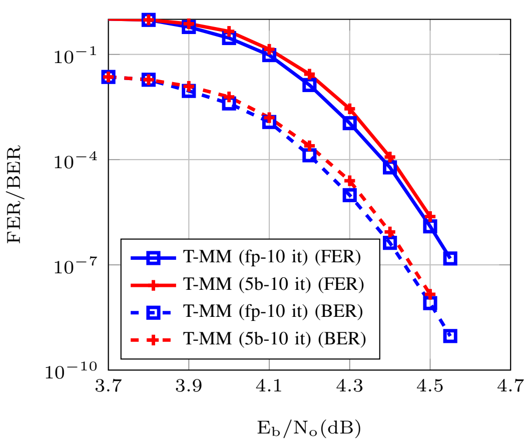

Fig. 1. FER-BER performance of T-MM algorithm for the (2304,2048) NB-LDPC code over GF(16), with AWGN channel and BPSK modulation.

$conf^*(1,2)$ [6] includes all possible sets of at most two symbols a' among the $m1(a)$ set, which deviate at most twice from the hard-decision path in the trellis. The set of symbols a' must satisfy the parity check equation for each symbol $a \in GF(q)$.

Each $\Delta Q(a)$ value from (1) is obtained from the set $m1(a')$, searching one or two symbols (a_1^* and a_2^*) that ensure the highest reliability (minimum value) for $\Delta Q(a)$. If only one symbol is selected, then the corresponding $\Delta Q(a)$ value is said to be a one-deviation path, otherwise, is considered to be a two-deviation path. For each $a \in$ GF(q), the set of one or two symbols that ensures the highest reliability is denoted as a^* to distinguish it from the rest of a' possible sets. $\Delta Q(a)$ value is chosen from the maximum between the $m1(a_1^*)$ and $m1(a_2^*)$ values.

CN output messages $\Delta R_{m,n}(a)$ are obtained using $\Delta Q(a)$, $m1(a)$, $m1(a)$ column index ($m1_{col}(a)$) and the second most reliable values $m2(a)$. The following algorithm is applied:

```
for $j = 1 \rightarrow d_c$ do
    if $m1_{col}(a_1^*) \neq j$ or $m1_{col}(a_2^*) \neq j$ then
        $\Delta R_{m,n_j}(a) = \Delta Q(a)$
    else if $m1_{col}(a_1^*) = m1_{col}(a_2^*)$ then
        $\Delta R_{m,n_j}(a) = m2(a)$
    else
        $\Delta R_{m,n_j}(a) = m1(a)$
end for
```

Finally, conversion to the normal domain is performed as follows: $R_{m,n}(a + \beta + z_n) = \lambda \cdot \Delta R_{m,n}(a)$, where β is the CN's syndrome and λ is a scaling value that improves the error-correction performance of the algorithm.

The frame error rate (FER) and bit error rate (BER) performance for the T-MM algorithm are presented in Fig. 1, where the (2304,2048) NB-LDPC code over GF(16) is used. The fixed-point model, which quantizes the input messages with 5 bits ($w = 5b$), introduces a performance loss of less than 0.05dB compared to the floating-point (fp) model. This code will be considered in the rest of the paper to show the implementation results of the proposed approach.

III. Proposed Decoder Architecture

T-MM algorithm [6] and the preceding T-EMS approach [4] [5] require the exchange of $q \times d_c$ messages from CN to VN. This amount of messages causes wiring congestion in the derived decoder architectures and, at the same time, increases the memory requirements of the decoder, especially when high-order fields ($q > 8$) and high-rate NB-LDPC codes are considered in the design. In this paper, we propose a new architecture for T-MM algorithm, which takes advantage of the redundancy in the output messages from the CN, to reduce the size of the messages exchanged with VN processors.

A. *Check-node architecture*

CN output ($\Delta R_{m,n}(a)$) contains d_c messages per GF(q) symbol. Among them, there are $d_c - 1$ or $d_c - 2$ messages equal to $\Delta Q(a)$, depending if the number of deviations made from the hard-decision path in the extra column calculation (1) is one or two, respectively. The rest of messages are equal to $m_2(a)$ or $m_1(a)$, respectively.

In this paper we propose an architecture that exchanges only the minimum amount of information from CN to VN. Let's define $E(a)$ as the set of messages corresponding to the extrinsic information sent to the VN processor. The $E(a)$ values compress the $m1(a)$ and $m2(a)$ list of values in a unique set, using the following rule: if a $\Delta Q(a)$ value is obtained from one deviation, $E(a) = m2(a)$, otherwise, $E(a) = m1(a)$.

On the other hand, the reliability values from $\Delta Q(a)$ are the most repeated in the exchanged messages from CN to VN. This set represents the intrinsic information obtained during the CN processing. We will refer to $\Delta Q(a)$ as $I(a)$ in the rest of the paper. However, $I(a)$ and $E(a)$ do not include all the necessary information to generate all $q \times d_c$ CN outputs messages in the T-MM algorithm [6]. In addition to $I(a)$ and $E(a)$, the index of the minimums involved in the calculation of the set $I(a)$ is also required to know the positions where $E(a)$ values must be used at the VN processor instead of $I(a)$. Taking into account that at most two symbols are used to derive each one of the $I(a)$ values, the deviation information is split into two sets $P1(a)$ and $P2(a)$, where each value of the sets requires $\lceil \log d_c \rceil$ bits to define the position of the minimum involved to derive $I(a)$. Finally, the updated hard-decision symbols (z_n^*) are required to construct the output CN messages ($R_{m,n}(a)$). z_n^* is obtained by adding the hard-decision symbols z_n to the syndrome value β as $z_n^* = z_n + \beta \quad \forall\ n \in \mathcal{N}(m)$. These symbols are used in the delta-to-normal domain conversion at the VN processor.

Table I summarizes the minimum information to be exchanged to the VN processor in terms of bits, and also the numerical results for the (2304,2048) NB-LDPC code over GF(16), where **H** is constructed following the methods in [7] [11]. In this code, $d_c = 36$, $d_v = 4$ and the number of bits for the quantized messages are $w = 5$.

On the other hand, the T-MM based decoder architecture in [6] exchanges 2880 bits ($q \times d_c \times w$) from CN to VN for the target code and $w = 5$. So, our proposal reduces the total wiring connections in 83% compared to [6]. It is important to remark that this reduction in the amount of information exchanged

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

	Number of bits		
	Generic	(2304,2048) NB-LDPC code, GF(16)	
		Proposed	Conventional
$I(a)$	$(q-1)\times w$	75 bits	-
$E(a)$	$(q-1)\times w$	75 bits	-
z_n^*	$d_c \times p$	144 bits	-
$P1(a)$	$(q-1)\times\lceil\log d_c\rceil$	96 bits	-
$P2(a)$	$(q-1)\times\lceil\log d_c\rceil$	96 bits	-
$R_{m,n}(a)$	$q\times d_c\times w$	-	2880 bits
Total		**486 bits**	**2880 bits**

between CN and VN does not introduce any performance loss compared to T-MM algorithm [6] since no approximations are made in the message processing, just compression of the information is performed.

The top-level CN architecture is presented in Fig. 2. Parallel processing is adopted to handle the input messages $Q_{m,n}(a)$ and the tentative hard decision symbols z_n through all the CN blocks.

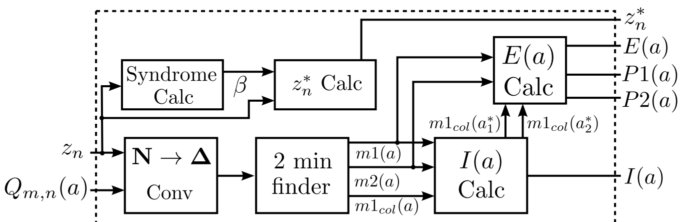

Fig. 2. Check-node top-level architecture

B. Top-level decoder architecture

The CN architecture presented in Section III-A is included in a decoder with horizontal layered schedule. Layered improves the convergence of the T-MM algorithm and, at the same time, the area of the entire decoder is considerably lower than the required by a fully parallel one.

Besides the above commented benefits, another important advantage comes from the fact that the layered schedule requires to store the CN output messages from one iteration to be used in the next one. Since the proposed decoder implements only one CN processor, the messages from the last iteration can be stored using shift registers with M stages, which reduces the number or registers required. The implementation of a conventional CN processor with $q\times d_c$ output messages would require $q\times d_c\times w\times M$ registers (737280 for the target code). On the other hand, our proposal only requires $M\times[2(q-1)\times(w+\lceil\log d_c\rceil)+d_c\times p]$ registers (121344 for the same code) to store the messages from the last iteration. This means a reduction of 83% in the use of registers compared to a conventional implementation of T-MM algorithm.

The VN processor requires a decompression network to build the CN output messages. Thus, the messages listed in Table I are converted to the $q\times d_c$ messages needed to perform the operations in the VN processor. The following operations must be performed to generate all the messages:

```
for $j = 1 \rightarrow d_c$ do
  if $P1(a) \neq j$ or $P2_(a) \neq j$ then
    $Out(a + z_j^*) = I(a)$
  else
    $Out(a + z_j^*) = E(a)$
end for
```

The proposed network is detailed in Fig. 3, where an example for GF(8) is used. In total, d_c decompression networks are required to generate all $q\times d_c$ values.

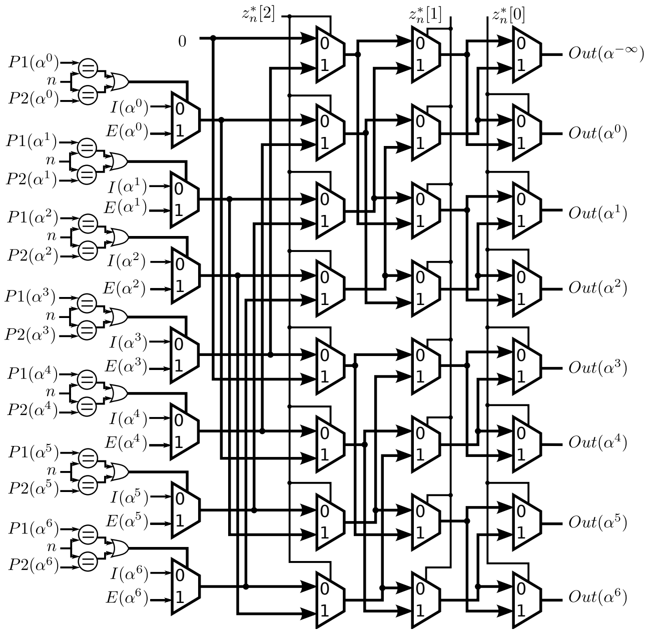

Fig. 3. Decompression Network for CN output messages. Example for GF(8).

The top-level decoder architecture is presented in Fig. 4, where the Check Node Processor is the one from Fig. 2 and the blocks labeled as **DN** are the decompression network from Fig. 3. The blocks labeled as **P** and $\mathbf{P^{-1}}$ are the permutation and inverse permutation networks responsible of rotating the messages according to the $h_{m,n}$ non-zero values of **H**. The **SR** block is the shift register that stores the CN output messages from one iteration to be used in the next one. The "VN mem" block is the memory required to store the

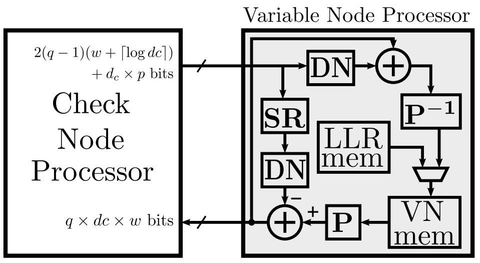

Fig. 4. Top-level proposed decoder architecture

processed messages during the decoding operation according to the layered schedule. The depth of the required memories fits with the size of the circulant sub-matrices which form **H** [7] [11]. In the case of the target code, the size of the circulant sub-matrices is $QC = 64$, which allows us to implement

TABLE II. COMPARISON OF THE PROPOSED NB-LDPC DECODER WITH OTHER WORKS IMPLEMENTED IN FPGA DEVICES FROM LITERATURE

Algorithm	Min-Max [8]	IHRB [9]	M-GBFDA [10]	T-MM [This Work]
Code	(744,653) GF(32)	(403,226) GF(32)	(837,723) GF(32)	(2304,2048) GF(16)
Code length (bits)	3720	2015	4185	9216
rate	0.88	0.56	0.86	0.89
Device	Virtex-2	Virtex-5	Virtex-6	Virtex-7
Slice LUT	47341	7841	29965	81644
Slice Registers	44659	529	21632	51995
BRAM	180	56	112	8
f_{clk} (MHz)	106	117.6	222	226
Iterations	15	25	20	10
Throughput (Mbps)	9.30	90.68	267	630.4

efficiently the memories in LUTs instead of BRAMs. The same conclusions are derived for the memories used to store the channel LLR values "LLR mem".

The decoder architecture from Fig. 4 was implemented on a Virtex-7 FPGA device for the target code and the results are presented in the last column of Table II. Under the best knowledge of the authors, Table II includes the best decoder architectures found in the literature which report implementation results for FPGA devices. We include the ones that achieve higher throughput, though each one implements different algorithms and considers different NB-LDPC codes. Our decoder significantly outperforms, in terms of throughput, the best decoder found in the literature for Min-Max algorithm implemented in FPGA [8]. In the case of the decoders proposed in [9] and [10], the throughput achieved is in the order of hundred on Mbps for GF(32). On the other hand, these algorithms exhibit considerable performance loss (between 0.7 - 1.2 dB), when compared against T-MM algorithm and it is important to remark that the decoding performance of this kind of algorithms do not improve when the number of iterations is increased. In addition, the NB-LDPC code used in this paper has the highest rate and the longest code length when compared against the others proposals presented in Table II.

The proposed decoder achieves a throughput of 630 Mbps, the highest for a NB-LDPC decoder implemented in a FGPA device. This throughput is calculated as

$$Throughput = \frac{f_{clk}[\text{MHz}] \cdot N \cdot p}{It \cdot (M + d_v \cdot seg) + (QC)} \left[\frac{\text{Mb}}{\text{s}}\right],$$

where It represents the number of iterations and seg is the number of pipeline stages used in the decoder. In the proposed decoder, $seg = 17$, $d_v = 4$ and $It = 10$.

The decoder from [6] is, under the best knowledge of the authors, the most efficient decoder implementation for ASIC reported in the literature. We include in Table III the ASIC implementation results for the decoder from [6] and for our proposed approach for the GF(16) NB-LDPC code used through this paper. A 90 nm CMOS process with standard cells and operating conditions of 25^{o}C and 1.2 V were used. It can be seen from Table III that our proposal outperforms the decoder from [6] in gate count and area in 48% and 38%, respectively, while the throughput achived is similar in both cases. These results confirm that the proposed decoder is suitable for FPGA and ASIC implementations.

TABLE III. ASIC IMPLEMENTATION OF THE PROPOSED NB-LDPC DECODER FOR THE (2304,2048) NB-LDPC CODE OVER GF(16)

	[6]	[This work]
Report	Post-layout	Post-layout
Gate Count (NAND)	1882 K	975 K
f_{clk} (MHz)	330.8	333.3
Throughput (Mbps)	957.5	964.7
Area (mm^2)	16.84	10.49

IV. CONCLUSIONS

We propose a NB-LDPC decoder which outperforms state-of-the-art implementations on both FPGA and ASIC technologies. Our approach presents an implementation for T-MM algorithm that greatly reduces the number of connections between check node and variable node processors and the memory requirements in a layered schedule, compared with a conventional implementation of T-MM based decoders.

ACKNOWLEDGMENT

This work was supported in part by the Universidad Politécnica de Valencia under Grant PAID-06-2012-SP20120625, in part by the Spanish Ministerio de Ciencia e Innovación under Grant TEC2011-27916 and in part by the Generalitat Valenciana under Grant GV/2014/011.

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] 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.

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

[4] 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.

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

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

[7] C.-S. Choi, H. Lee, N. Kaneda, and Y.-K. Chen, "Concatenated non-binary LDPC and HD-FEC codes for 100Gb/s optical transport systems," in *IEEE International Symposium on Circuits and Systems (ISCAS), 2012*, 2012, pp. 1783–1786.

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

[9] X. Zhang, F. Cai, and S. Lin, "Low-Complexity Reliability-Based Message-Passing Decoder Architectures for Non-Binary LDPC Codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 11, pp. 1938–1950, Nov 2012.

[10] F. Garcia-Herrero, M. Canet, and J. Valls, "High-speed NB-LDPC decoder for wireless applications," in *International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), 2013*, Nov 2013, pp. 215–220.

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