UNIVERSITAT POLITÈCNICA DE VALÈNCIA

VLSI algorithms and architectures for non-binary-LDPC decoding

September, 2016

Author: Jesús Omar Lacruz Jucht

Thesis supervisor: Javier Valls Coquillat
Thesis supervisor: Francisco García Herrero

Abstract

This thesis studies the design of low-complexity soft-decision Non-Binary Low-Density Parity-Check (NB-LDPC) decoding algorithms and their corresponding hardware architectures suitable for decoding high-rate codes at high throughput (hundreds of Mbps and Gbps).

In the first part of the thesis the main aspects concerning to the NB-LDPC codes are analyzed, including a study of the main bottlenecks of conventional soft-decision decoding algorithms (Q-ary Sum of Products (QSPA), Extended Min-Sum (EMS), Min-Max and Trellis-Extended Min-Sum (T-EMS)) and their corresponding hardware architectures. Despite the limitations of T-EMS algorithm (high complexity in the Check Node (CN) processor, wiring congestion due to the high number of exchanged messages between processors and the inability to implement decoders over high-order Galois fields due to the high decoder complexity), it was selected as starting point for this thesis due to its capability to reach high-throughput.

Taking into account the identified limitations of the T-EMS algorithm, the second part of the thesis includes six papers with the results of the research made in order to mitigate the T-EMS disadvantages, offering solutions that reduce the area, the latency and increase the throughput compared to previous proposals from literature without sacrificing coding gain. Specifically, five low-complexity decoding algorithms are proposed, which introduce simplifications in different parts of the decoding process. Besides, five complete decoder architectures are designed and implemented on a 90nm Complementary Metal-Oxide-Semiconductor (CMOS) technology. The results show an achievement in throughput higher than 1Gbps and an area less than 10 $\mathbf{mm}^2$. The increase in throughput is 120% and the reduction in area is 53% compared to previous implementations of T-EMS, for the (837,726) NB-LDPC code over GF(32). The proposed decoders reduce the CN area, latency, wiring between CN and Variable Node (VN) processor and the number of storage elements required in the decoder. Considering that these proposals improve both area and speed, the efficiency parameter (Mbps / Million

NAND gates) is increased in almost five times compared to other proposals from literature.

The improvements in terms of area allow us to implement NB-LDPC decoders over high-order fields which had not been possible until now due to the high-complexity of decoders previously proposed in literature. Therefore, we present the first post-place and route report for high-rate codes over high-order fields higher than Galois Field (GF)(32). For example, for the (1536,1344) NB-LDPC code over GF(64) the throughput is 1259Mbps occupying an area of 28.90 $\mathbf{mm}^2$. On the other hand, a decoder architecture is implemented on a Field Programmable Gate Array (FPGA) device achieving 630 Mbps for the high-rate (2304,2048) NB-LDPC code over GF(16). To the best knowledge of the author, these results constitute the highest ones presented in literature for similar codes and implemented on the same technologies.

September, 2016

Jesús Omar Lacruz
jlacruz@ula.ve

Resumen

En esta tesis se aborda el estudio del diseño de algoritmos de baja complejidad para la decodificación de códigos de comprobación de paridad de baja densidad no binarios (NB-LDPC) y sus correspondientes arquitecturas apropiadas para decodificar códigos de alta tasa a altas velocidades (cientos de Mbps y Gbps).

En la primera parte de la tesis los principales aspectos concernientes a los códigos NB-LDPC son analizados, incluyendo un estudio de los principales cuellos de botella presentes en los algoritmos de decodificación convencionales basados en decisión blanda (QSPA, EMS, Min-Max y T-EMS) y sus correspondientes arquitecturas hardware. A pesar de las limitaciones del algoritmo T-EMS (alta complejidad en el procesador del nodo de chequeo de paridad (CN), congestión en el rutado debido al intercambio de mensajes entre procesadores y la incapacidad de implementar decodificadores para campos de Galois de orden elevado debido a la elevada complejidad), éste fue seleccionado como punto de partida para esta tesis debido a su capacidad para alcanzar altas velocidades.

Tomando en cuenta las limitaciones identificadas en el algoritmo T-EMS, la segunda parte de la tesis incluye seis artículos con los resultados de la investigación realizada con la finalidad de mitigar las desventajas del algoritmo T-EMS, ofreciendo soluciones que reducen el área, la latencia e incrementando la velocidad comparado con propuestas previas de la literatura sin sacrificar la ganancia de codificación. Especificamente, cinco algoritmos de decodificación de baja complejidad han sido propuestos, introduciendo simplificaciones en diferentes partes del proceso de decodificación. Además, arquitecturas completas de decodificadores han sido diseñadas e implementadas en una tecnologia CMOS de 90nm consiguiéndose una velocidad mayor a 1Gbps con un área menor a 10 $\mathbf{mm}^2$, aumentando la velocidad en 120% y reduciendo el área en 53% comparado con previas implementaciones del algoritmo T-EMS para el código (837,726) implementado sobre campo de Galois GF(32). Las arquitecturas propuestas reducen el área del CN, latencia, número de mensajes intercambiados entre el nodo de comprobación de paridad (CN) y el nodo variable (VN) y el número de elementos de almacenamiento en el decodificador. Considerando que estas propuestas mejoran tanto el área como

la velocidad, el parámetro de eficiencia (Mbps / Millones de puertas NAND) se ha incrementado en casi cinco veces comparado con otras propuestas de la literatura.

Las mejoras en términos de área nos ha permitido implementar decodificadores NB-LDPC sobre campos de Galois de orden elevado, lo cual no habia sido posible hasta ahora debido a la alta complejidad de los decodificadores anteriormente propuestos en la literatura. Por lo tanto, en esta tesis se presentan los primeros resultados incluyendo el emplazamiento y rutado para códigos de alta tasa sobre campos finitos de orden mayor a GF(32). Por ejemplo, para el código (1536,1344) sobre GF(64) la velocidad es 1259 Mbps ocupando un área de 28.90 $\mathbf{mm}^2$. Por otro lado, una arquitectura de decodificador ha sido implementada en un dispositivo FPGA consiguiendo 660 Mbps de velocidad para el código de alta tasa (2304,2048) sobre GF(16). Estos resultados constituyen, según el mejor conocimiento del autor, los mayores presentados en la literatura para códigos similares implementados para las mismas tecnologías.

September, 2016

Jesús Omar Lacruz
jlacruz@ula.ve

Resum

En esta tesi s'aborda l'estudi del disseny d'algoritmes de baixa complexitat per a la descodificació de codis de comprovació de paritat de baixa densitat no binaris (NB-LDPC), i les seues corresponents arquitectures per a descodificar codis d'alta taxa a altes velocitats (centenars de Mbps i Gbps).

En la primera part de la tesi els principals aspectes concernent als codis NB-LDPC són analitzats, incloent un estudi dels principals colls de botella presents en els algoritmes de descodificació convencionals basats en decisió blana (QSPA, EMS, Min-Max i T-EMS) i les seues corresponents arquitectures. A pesar de les limitacions de l'algoritme T-EMS (alta complexitat en el processador del node de revisió de paritat (CN), congestió en el rutat a causa de l'intercanvi de missatges entre processadors i la incapacitat d'implementar descodificadors per a camps de Galois d'orde elevat a causa de l'elevada complexitat), este va ser seleccionat com a punt de partida per a esta tesi degut a la seua capacitat per a aconseguir altes velocitats.

Tenint en compte les limitacions identificades en l'algoritme T-EMS, la segona part de la tesi inclou sis articles amb els resultats de la investigació realitzada amb la finalitat de mitigar els desavantatges de l'algoritme T-EMS, oferint solucions que redueixen l'àrea, la latència i incrementant la velocitat comparat amb propostes prèvies de la literatura sense sacrificar el guany de codificació. Específicament, s'han proposat cinc algoritmes de descodificació de baixa complexitat, introduint simplificacions en diferents parts del procés de descodificació. A més, s'han dissenyat arquitectures completes de descodificadors i s'han implementat en una tecnologia CMOS de 90nm aconseguint-se una velocitat major a 1Gbps amb una àrea menor a 10 mm^2, augmentant la velocitat en 120% i reduint l'àrea en 53% comparat amb prèvies implementacions de l'algoritme T-EMS per al codi (837,726) implementat sobre camp de Galois GF(32). Les arquitectures proposades redueixen l'àrea del CN, la latència, el nombre de missatges intercanviats entre el node de comprovació de paritat (CN) i el node variable (VN) i el nombre d'elements d'emmagatzemament en el descodificador. Considerant que estes propostes milloren tant l'àrea com la velocitat, el paràmetre d'eficiència (Mbps / Milions de

portes NAND) s'ha incrementat en quasi cinc vegades comparat amb altres propostes de la literatura.

Les millores en termes d'àrea ens ha permès implementar descodificadors NB-LDPC sobre camps de Galois d'orde elevat, la qual cosa no havia sigut possible fins ara a causa de l'alta complexitat dels descodificadors anteriorment proposats en la literatura. Per tant, nosaltres presentem els primers reports després de l'emplaçament i rutat per a codis d'alta taxa sobre camps finits d'orde major a GF(32). Per exemple, per al codi (1536,1344) sobre GF(64) la velocitat és 1259 Mbps ocupant una àrea de 28.90 mm^2. D'altra banda, una arquitectura de descodificador ha sigut implementada en un dispositiu FPGA aconseguint 660 Mbps de velocitat per al codi d'alta taxa (2304,2048) sobre GF(16). Estos resultats constitueixen, per al millor coneixement de l'autor, els millors presentats en la literatura per a codis semblants implementats per a les mateixes tecnologies.

September, 2016

Jesús Omar Lacruz
jlacruz@ula.ve

Acknowledgments

Firstly, i want to thank my advisor Dr. Javier Valls for let me embark in this PhD project, give me the confidence and support during these years, even when it seems really hard to finish it.

Besides, my deeply gratitude to Dr. Francisco Garcia for his brilliant ideas that help me to develop the works presented in this thesis. More than an advisor he and Javier are good friends.

Secondly, I have to thank all the staff of GISED, specially to Dr. Maria Jose Canet for her invaluable help improving the quality of the papers and this manuscript. Moreover, to my lab partners Julian, Ferran and Joan Marc for the good moments we share all these years.

Thanks to Universidad de Los Andes in Venezuela, to give me the chance to improve my quality as university professor, giving me the financial support during the major part of the PhD program.

Finally i would like to thank my parents for their encouragement words in the distance. To you, Ori, you came into my life at the right time to give me your love and support to finish this project.

Contents

List of Figures

List of Tables

Acronyms

ASIC Application Specific Integrated Circuit.

AWGN Additive White Gaussian Noise.

BER Bit Error Rate.

BPSK Binary Phase Shift Keying.

CMOS Complementary Metal-Oxide-Semiconductor.

CN Check Node.

CNBMP Compressed Non-Binary Message Passing.

EMS Extended Min-Sum.

FER Frame Error Rate.

fp floating point.

FPGA Field Programmable Gate Array.

GF Galois Field.

GSL GNU Scientific Library.

HD hard-decision.

LDPC Low-Density Parity-Check.

LLR Log-Likelihood Ratio.

m-TMM modified Trellis Min-Max.

NB-LDPC Non-Binary Low-Density Parity-Check.

NCG Net Coding Gain.

OMO One - Minimum - Only.

PRNG Pseudo-Random Number Generator.

QC Quasi-cyclic.

QSPA Q-ary Sum of Products.

RC-TMM Reduced Complexity Trellis Min-Max.

SD Soft-Decision.

SNR Signal-to-Noise Ratio.

T-EMS Trellis-Extended Min-Sum.

T-MM Trellis Min-Max.

UUT Unit Under Test.

VHDL Very High Speed Integrated Circuit Hardware Description Language.

VLSI Very Low Scale Integration.

VN Variable Node.

Preface

Communication and storage systems demand for high-speed information exchange between transmitter and receiver nodes. At these high-speed information rates, channel impairments become more harmful as well as the inherent non-linearities of the electronic components, which reduce the reliability of the received information. To overcome this situation and provide more reliable communications, efficient channel coding techniques are required [1]. In the last two decades, Low-Density Parity-Check (LDPC) codes [2] have been successfully included in numerous standards such as DVB-S2 [3], IEEE 802.16e [4] and IEEE 802.11n [5], among others. The main reasons for their success are that their performance are close to the channel capability for long codewords and that they can be decoded with low-complexity algorithms based on belief propagation, which make them suitable for practical applications where high-speed and low area are important constraints.

LDPC codes defined over a GF GF($q = 2^p$), with $p > 1$, were analyzed in [6] as an extension of binary LDPC codes. These codes perform better than their binary counterparts for codes with low and medium codeword length. Additionally, they improve the burst error-correction capability and work in conjunction with high-order modulation schemes (16QAM, 64QAM, 256QAM) [7, 8]. Nowadays, NB-LDPC codes have been considered strong candidates to be used in the 100Gbps optical transport system [9, 10]. These systems require Soft-Decision (SD) decoding strategies that provide 10dB of Net Coding Gain (NCG) at a BER $= 10^{-15}$, with a maximum overhead of 20% and code rate higher than 0.8.

NB-LDPC codes are strong candidates to be used in other important applications such as flash memory devices [11]. The NB-LDPC coding scheme is suitable for this application due to the constant increase in capacity, the use of multi-level NAND cells and the requirement of high-rate codes.

Besides the previous applications, NB-LDPC codes have been studied to be used in space communications [12, 13], where burst error correction capability and high coding gain are important issues.

Despite the multiple advantages of NB-LDPC codes over another coding schemes, they can not still be practically implemented because the state-of-the-art hardware implementations are far from offering high-throughput and high-rate decoders with reasonably low area.

This work is focused on reducing the complexity of SD decoding algorithms suitable for high-rate NB-LDPC codes, without sacrificing the coding gain of conventional algorithms such as EMS [14] and Min-Max [15]. The high-speed architectures derived from this low-complexity algorithms are also part of this job.

Objectives

The starting point of this thesis is the results obtained during the realization of my Master Degree thesis, where a complete decoder architecture based on the T-EMS algorithm was implemented. In that work, the main bottlenecks for the implementation of T-EMS were identified. The first one is the high complexity of the CN processor, which increases the latency and limits the throughput, especially for high-order Galois fields. The second bottleneck is the high density of wires that exchange information between CN and VN processors, due to the transportation of the full set of q messages. This congestion increases the decoder area and also reduces the achieved throughput.

The main objective of this thesis is to develop low-complexity algorithms and architectures for Very Low Scale Integration (VLSI) implementation of high-speed SD NB-LDPC decoders suitable for high-rate codes over high-order Galois fields. The emphasis is placed on the reduction of both bottlenecks, previously mentioned, that limit the performance of current NB-LDPC decoder architectures. Therefore, the specific objectives of this thesis are:

- Objective 1: To propose low-complexity decoding algorithms and their corresponding decoder architectures to simplify and reduce the complexity of the CN processor, with the goal of reducing the area and increase the throughput compared to the existing approaches from literature without compromising the coding gain.
- Objective 2: To develop solutions to reduce the number of messages exchanged between processors in NB-LDPC decoders with the aim of reducing the wiring congestion that usually appear on decoder implementations.
- Objective 3: To formulate a EMS- based algorithm that allows the implementation of high-performance decoders for Galois fields larger than 32.

Methodology

The methodology followed to meet the objectives of this thesis includes the next steps:

- Bibliography review of the state-of-the-art decoding algorithms. Analysis of the most efficient decoder architectures in terms of area and throughput, and detection improvements.

- Development of a software communication system model using C/C++ language that includes the components in Fig. 1. The method used to test the algorithms is through Montecarlo simulation, where the non-binary message is constructed using uniform random number generation. Next, this message is encoded (using the generator matrix) and modulated in Binary Phase Shift Keying (BPSK). For each modulated message the noise pattern is changed using Additive White Gaussian Noise (AWGN) generators[1]. The received signal is demodulated and the soft information is used to obtain the Log-Likelihood Ratio (LLR) which is the input of the decoding algorithm under test. The decoder receives the LLR values and then, applies the iterative message exchange to estimate the codeword. Finally, the estimated codeword is compared to the transmitted one to check if the decoding is successful or there is a decoding failure.

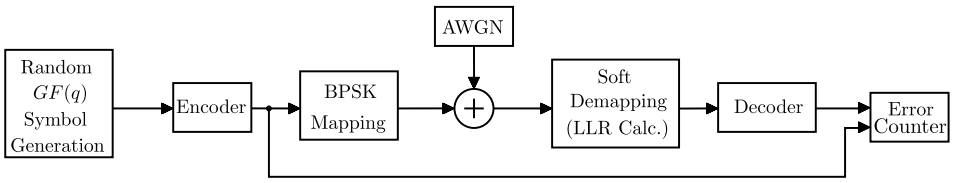


Figure 1: Software simulation model

- Analysis of the Frame Error Rate (FER)/BER performance of the proposed algorithms, and comparison to the state-of-the-art proposals from literature. Measurement of the possible performance loss introduced by our proposal and adjustment in the parameters of the algorithm in order to minimize these losses.

- Analysis of the iterative decoding algorithm with finite precision, after the validation of its floating-point model. To this end, a finite precision model is included in the software simulator and then Montecarlo simulations are performed (Fig. 1). The objective is to find the best fixed-point parameters

[1]The AWGN model included in GNU Scientific Library (GSL) is used to make all the simulations required to validate each algorithm at low Bit Error Rate (BER) values. The standadard C/C++ Pseudo-Random Number Generator (PRNG) can not be employed because its short period length would affect the validation process

to establish a balance between performance loss, compared to the floating-point model, and required area.

- Design of the decoder architecture applying the results of the finite precision analysis and taking into account the next hardware constraints: area, latency and decoding throughput.

- Early estimation of the gate count of the design and comparison to other proposals from literature. Modifications in the architecture, in the quantization scheme or in the algorithm are made when the area of the design or the estimated latency do not accomplish requirements.

- Description of the hardware architecture using Very High Speed Integrated Circuit Hardware Description Language (VHDL) and functional verification before implementing the design. The verification is performed following the steps listed below:

    1. Generation of the input and output vectors using the software finite precision model.

    2. Description of a testbench in VHDL to validate the proposed design at functional level using Modelsim simulator. The Unit Under Test (UUT) will receive the input test vectors, process them and then it will generate the output values which are compared to the output test vectors (obtained using the software finite precision model).

- Implementation of the design for FPGA and Application Specific Integrated Circuit (ASIC) technologies considering the particular characteristics of each one of them. For FPGA implementation Xilinx tools are used and Cadence tools are employed for ASIC implementation. Once implementation is completed, the area results are compared to the estimations made during the design process and to other proposals from literature. After implementation, new simulations are made to ensure that the design fits with the software simulation model. If any constraint is not meet or there is some malfunction, the previous steps are applied again to redesign the decoder. Therefore, the design process is iterative.

Contributions

The contributions made during the realization of this research are summarized in the following list:

1. Five NB-LDPC decoding algorithms were proposed, two of them (Trellis Min-Max (T-MM) and Compressed Non-Binary Message Passing (CNBMP)) do not introduce any performance loss compared to previous proposals from

literature. The remaining three algorithms introduce a negligible performance loss less than 0.1dB for the high-rate codes under test.

2. Five decoding architectures were coded in VHDL and implemented in a 90nm CMOS technology and, moreover, post place and route results were presented. Besides, the architectures designed were implemented for three different high-rate codes with GF(16), GF(32) and GF(64). For the three codes the achieved throughput was higher than 1Gbps, being the highest compared to other proposals from literature for codes with similar characteristics to the best knowledge of the author.

3. One decoding architecture based on the CNBMP approach was implemented in a FGPA device for the (2304,2048) NB-LDPC code with GF(16) achieving 630 Mbps, the highest throughput for FPGA based decoder architectures compared to other proposals from literature.

4. The first post-place and route report for a high-rate decoding architecture with GF(64) was presented. The decoding algorithm was the Reduced Complexity Trellis Min-Max (RC-TMM) which introduces simplifications over the T-MM algorithm that saves a considerably amount of area for codes over high-order fields.

5. All the proposed decoding architectures applied parallel processing of messages, thus low latency is achieved in all of them (less than 4 μs).

6. Since the proposals presented in this manuscript improve both area and throughput, the efficiency parameter (Mbps / Million NAND gates) was greatly increased in almost six times compared to the T-EMS approach previous to this thesis.

The results got during the realization of this thesis have been published in five international journals. Furthermore, an international conference paper have been presented. Chapter 2 to 7 include the publications derived from this thesis.

- International journals:

    1. **Chapter 2**: Jesús O. Lacruz, Francisco Garcia-Herrero, David Declercq, Javier Valls, "Simplified Trellis Min-Max Decoder Architecture for Non-Binary Low-Density Parity-Check Codes". IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 23, No. 9, pp. 1783-1792, Sept. 2015.

    2. **Chapter 3**: Jesús O. Lacruz, Francisco García-Herrero, David Declercq, Javier Valls, "One Minimum Only Trellis Decoder for Non-Binary Low-Density Parity-Check Codes". IEEE Transaction on Cir-

cuits and Systems-I: Regular Papers, Vol. 62, No. 1, pp. 177-184, Jan. 2015.

3. **Chapter 4**: Jesús O. Lacruz, Francisco Garcia-Herrero, Javier 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.

4. **Chapter 6**: Jesús O. Lacruz, Francisco Garcia-Herrero, María José Canet, Javier Valls, "High-performance NB-LDPC decoder with reduction of message exchange". IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 24, No. 5, pp. 1950-1961, May 2016.

5. **Chapter 7**: Jesús O. Lacruz, Francisco Garcia-Herrero, María José Canet, Javier Valls, "Reduced-complexity Non-Binary LDPC decoder for high-order Galois fields based on Trellis Min-Max algorithm". IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 24, No. 8, pp. 2643-2653, Aug. 2016.

- International Conferences:

1. **Chapter 5**: Jesús O. Lacruz, Francisco Garcia-Herrero, María José Canet, Javier Valls, Asunción Perez-Pascual, "A 630 Mbps Non-Binary LDPC Decoder for FPGA". IEEE International Symposium on Circuits & Systems (ISCAS) 2015, pp. 1989-1992. Lisbon-Portugal. May, 2015

Thesis Structure

This thesis is presented as a compilation of publications. Chapter 1 includes the basis of NB-LDPC codes, schedules, hardware architectures and soft-decision decoding algorithms. Chapters from 2 to 7 include the compilation of publications product of this thesis. The articles have been organized in chronological order just as the research was made. The structure of these chapters is the one commonly used for papers, ie, state-of-the-art and related works, presentation of the proposed approach, BER/FER performance of the algorithm, proposed hardware implementation, results and comparison to related works and, finally, the conclusions.

To ease the identification of the improvements made in each paper, we include in the corresponding chapter a figure as the one in Fig. 2, where the decoder characteristics that were improved in each paper are highlighted. Therefore, if the algorithm complexity was reduced, the "Check Node" is highlighted. If the latency was decreased, the discontinuous lines crossing the "Check Node", that represent the pipeline of the architecture are highlighted. In case of the reduction

of the messages exchanged, the wires connecting the corresponding blocks are emphasized. Similar treatment was done for the memory, area and throughput improvements. In cases where the proposal is suitable for high-order Galois fields, "GF(q) ↑↑" is highlighted.

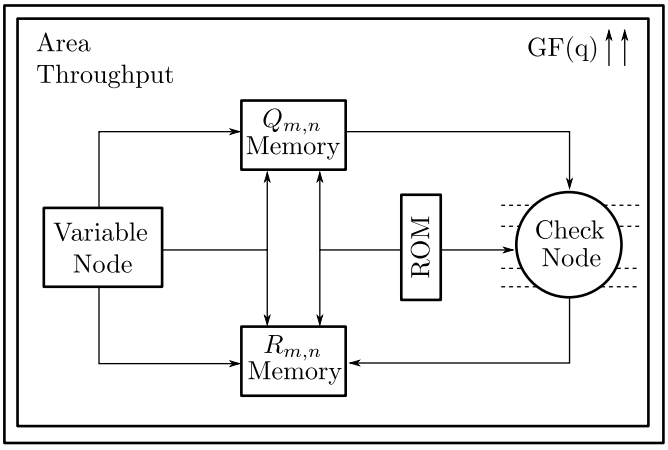


Figure 2: Key points of the improvements presented in each chapter of the manuscript.

The starting point of the research made in this thesis is the T-EMS algorithm from [16]. T-EMS offers the possibility of the parallel processing of messages in the decoder, which allows the design of high-speed decoders with higher throughput compared to previous proposals from literature. The first implementation of this algorithm, also made by this author [17], achieved a throughput of 484Mbps in a 90nm CMOS process after synthesis results for the (837,726) NB-LDPC code over GF(32). This previous work allowed the identification of bottlenecks in both algorithm and hardware architecture of T-EMS, and set the bases of this thesis.

Chapter 2 introduces simplifications over the T-EMS algorithm and decoder architecture. Specifically, the way to derive the check-node output messages in T-EMS was simplified. On the other hand, the reliability of the output messages was computed with min-max operators, and this reduces the width of the decoder data-path. This simplification of T-EMS was named Trellis Min-Max algorithm (T-MM). Additionally, a complete decoder architecture for the (837,726) NB-LDPC code over GF(32) was proposed. It achieves a throughput 660Mbps and an area of 14.75 mm^2.

Chapter 3 includes further simplifications over the CN processor compared to the proposal presented in Chapter two. Estimators for the second-most reliable value for the check-node input messages were presented. This estimators avoids the use of two-minimum finders in the check-node processors of NB-LDPC decoders. A complete hardware architecture was also presented, which achieves 818 Mbps of

throughput with an area of 16.10 mm^2. The increase in throughput is about 24% compared to the results achieved in Chapter 2.

Chapter 4 presents a method to reduce the quantity of messages exchanged between CN and VN processors. The number of storage elements in the decoder was also reduced. The performance of T-MM or T-EMS algorithms was kept since there is no information loss. The achieved throughput was 981 Mbps requiring an area of 10.6 mm^2. The increase in throughput is 20% and the reduction in area is about 34% compared to the results from Chapter 3.

The work from Chapter 5 includes implementation details for the proposal from Chapter 4, describing the CN and VN architectures thoroughly. Furthermore, the design was particularized and optimized for FPGA using the (2304,2048) NB-LDPC code over GF(16). It achieved 630 Mbps of throughput, the highest for FPGA implementation compared to previous proposals.

An algorithm which reduces the number of messages exchanged between processors in NB-LDPC decoders was proposed in Chapter 6. The results from Chapter 4 and 5 were used to derive this new algorithm. The reduction of messages causes a performance loss that is negligible compared to T-MM algorithm (Chapter 2). The achieved throughput was 1.08 Gbps and the area 8.97 mm^2, outperforming the results of the paper from Chapter 4 in 10% and 19% for throughput and area, respectively.

Finally, a new method to reduce the number of messages exchanged between processors in NB-LDPC decoders is included in Chapter 7. This reduction of messages allows a great decrease in the complexity of the CN processor compared to the architecture from Chapter 2, whose application was limited for codes with field orders higher than GF(32) due to its complexity. The implementation of the design for the (837,726) NB-LDPC code over GF(32) achieved a throughput (1.07 Gbps) and area (9.8mm^2) are very similar to the results from Chapter 6. Additionally, we included the first post place & route results for a high-rate code over GF(64). The achieved throughput was 1.26 Gbps with an area of 28.9mm^2 for the high-rate (1536,1344) NB-LDPC code over GF(64).

To analyze the contributions of this thesis, Chapter 8 includes a discussion of the main results achieved during this research. First, each proposal is analyzed individually; secondly, the impact of each proposal in the objectives of the thesis is studied. Next, a comparison between the starting-point [17] and the final results achieved during this thesis is made. The improvement in throughput is about 120%, whereas the reduction in area (measured in terms of NAND gates) is about 53%. Furthermore, this chapter includes the conclusions derived from the thesis and the future research lines.

Chapter 1

State of the art of non-binary low-density parity-check codes

This chapter summarizes the background necessary to understand the following chapters and the state of art of the Non-Binary Low-Density Parity-Check (NB-LDPC) decoding algorithms and architectures. It includes 1^{st}) the basis of the NB-LDPC codes and their iterative decoding mechanism; 2^{nd}) the nomenclature to formulate the decoding algorithms; 3^{rd}) the schedules used in the decoding process; 4^{th}) the different architectural options suitable to implement the decoders; and finally, 5^{th}) the algorithms and architectures to compute the check node, with the emphasis placed in the Trellis Extended Min-Sum algorithm, which is the starting point of this thesis.

1.1 LDPC codes and decoding process

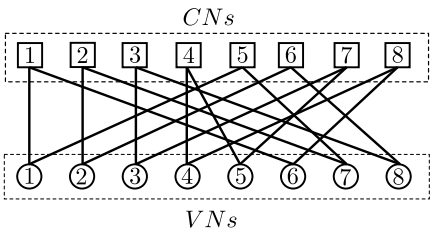
LDPC codes are linear block codes characterized by a parity-check matrix $\mathbf{H}$. The term low-density means that $\mathbf{H}$ is sparse, with just a few non-zero elements. If the elements in $\mathbf{H}$ belong to the set $\{0, 1\}$, the code is said to be binary. LDPC codes were first investigated by Gallager [18] during his doctoral studies. His work was ignored due to the complexity of the decoding process and due to technology limitations in those years, until MacKay and Neal rediscovered them in 1995 [19]. LDPC codes can also been characterized by a bipartite graph called Tanner graph [20], where two kind of nodes are distinguished: the ones called VN, related to each column of $\mathbf{H}$, and the others called CN, which correspond to each row of $\mathbf{H}$. LDPC codes are decoded iteratively by means of message-passing algorithms between VN and CN and vice versa. Fig. 1.1 shows an example of a parity-check matrix (a), and its corresponding Tanner graph (b). The decoding process is the

following: i) bit reliability values are passed from VN to CN, where parity-check equations are solved; ii) messages based on the incoming reliability values are sent back to the VNs, where each node is updated. The process is repeated iteratively until all parity-check equations are satisfied or a maximum number of iterations ($iter$) is reached.

LDPC decoding algorithms usually include an early stopping criterion that consists on verifying if all parity-check equations are fulfilled with the tentative codeword $\tilde{\mathbf{c}}$. If that condition is reached, $\tilde{\mathbf{c}}$ is a valid codeword and the decoder stops, otherwise a new iteration starts.

$$\mathbf{H} = \left[\begin{array}{cccc:cccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}\right]$$

(a)



(b)

Figure 1.1: (a) Example of a LDPC code parity-check matrix. (b) Tanner graph representation for the matrix in (a)

LDPC codes can be regular or irregular. The first ones contain the same number of non-zero elements in all the columns of $\mathbf{H}$ (d_c) and the same number of non-zero elements in all the rows (d_v). In the second ones the columns and/or rows do not necessarily have the same number of non-zero elements. In this thesis, without loss of generality and for the sake of simplicity, we only consider regular LDPC codes. The CN degree (d_c) refers to the number of non-zero elements per row in $\mathbf{H}$. In the same way, the VN degree (d_v) corresponds to the number of non-zero elements per column in $\mathbf{H}$. In the example of Fig. 1.1 $d_c = d_v = 2$.

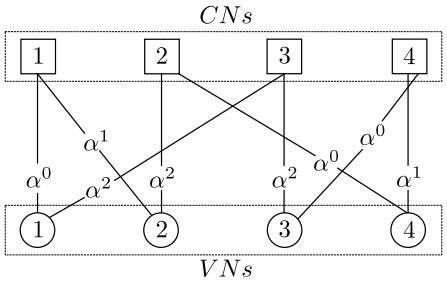
NB-LDPC codes were first studied by Davey and MacKay [6], emerging as a natural extension of the binary ones. Similar to the binary case, NB-LDPC codes are characterized by a parity-check matrix whose non-zero elements, $h_{m,n}$, belong to a Galois field GF($q = 2^p$), with $p > 1$, being $p = 1$ the binary case, as in the example of Fig. 1.1.

Fig. 1.2.a includes an example of a non-binary parity-check matrix with the non-zero elements from GF(2^2). In the non-binary case, the Tanner graph is modified to include the non-zero $\mathbf{H}$ elements in the edge that connects the nodes. In the literature, some authors [14, 16, 21] include intermediate nodes between CN and VN in the Tanner graph, denominated permutation nodes, which include the corresponding non-zero $\mathbf{H}$ values. These nodes perform the multiplication of the

symbols coming from the VN and the corresponding non-zero **H** value. The symbols from CN to VN are multiplied by the corresponding non-zero **H** element.

$$\mathbf{H} = \begin{vmatrix} \alpha^0 & \alpha^1 & 0 & 0 \\ 0 & \alpha^2 & 0 & \alpha^0 \\ \alpha^2 & 0 & \alpha^2 & 0 \\ 0 & 0 & \alpha^0 & \alpha^1 \end{vmatrix}$$

(a)



(b)

Figure 1.2: (a) Example of a NB-LDPC code parity-check matrix. (b) Tanner graph corresponding to the matrix in (a)

1.2 Nomenclature

Let us define $\mathbf{m} = [m_1, m_2, \ldots, m_K]$ as the message to be coded, where each element $m_i \in \mathrm{GF}(q)$ and $K = N - GFrank(\mathbf{H})$[1]. The codeword $\mathbf{c} = [c_1, c_2, \ldots, c_N]$ is obtained by means of $\mathbf{c} = \mathbf{m} \cdot \mathbf{G}$, where $\mathbf{G}$, with dimensions $K \times N$, is the code generator matrix which must accomplish $\mathbf{G} \cdot \mathbf{H}^T = 0$. $\mathbf{y} = [y_1, y_2, \ldots, y_N]$ is the sequence at the receiver side, being $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $\mathbf{e}$ is the GF representation of the noise introduced by the communication channel, assumed as AWGN. The received sequence is used to calculate the LLR values, which serve as input for most of the NB-LDPC decoding algorithms. LLRs are obtained by means of $L_n(a) = \log[P(c_n = z_n|y_n)/P(c_n = a|y_n)]$, where usually a normalization is made to ensure that all the values are non-negative, $L'_n(a) = |L_n(a) - L_n(z_n)|$, being z_n the hard-decision symbol associated to the highest reliability. So, for each symbol n in the received sequence $\mathbf{y}$, the LLR forms the set $\mathbf{L_n} = [L_n(0), L_n(1), \ldots, L_n(q-1)]$. Fig. 1.3 includes a simplified block diagram for the system model including the nomenclature used and the cardinality of each set.

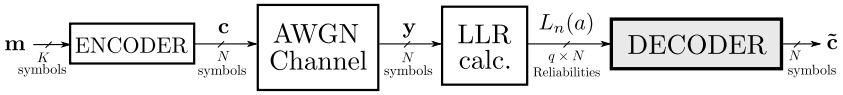


Figure 1.3: Simplified block diagram for the system model

[1] $GFrank(\cdot)$ denotes the number of linearly independent equations of a matrix, using GF arithmetic

This thesis is focused on soft-decision decoders where the reliability values from the received sequence $\mathbf{y}$, used to calculate the LLR, are updated in an iterative way based on their own reliability and the ones from their neighbors. In a general case, each VN exchanges with its connected CNs all the LLR values updated during a decoding iteration. This set of messages is denoted as $Q_{m,n}(a)\ \forall\ a\ \in\ \mathrm{GF}(q)$ and $n\ \in\ \mathcal{M}(n)$, where $\mathcal{M}(n)$ refers to the set of CNs connected to a VN n. In a similar way, after the CN processing, the CNs send the updated reliability values $R_{m,n}(a)\ \forall\ a\ \in\ \mathrm{GF}(q)$ and $m\ \in\ \mathcal{N}(m)$ to their connected VNs. $\mathcal{N}(m)$ refers to the set of VNs connected to a CN m. Based on the incoming set of updated reliability values $R_{m,n}(a)$ and the channel LLR values, each VN calculates the *a posteriori* information, which is used to obtain the tentative decoded codeword $\tilde{\mathbf{c}}$. Fig. 1.4 includes a scheme of the message exchanged between nodes in a Tanner graph.

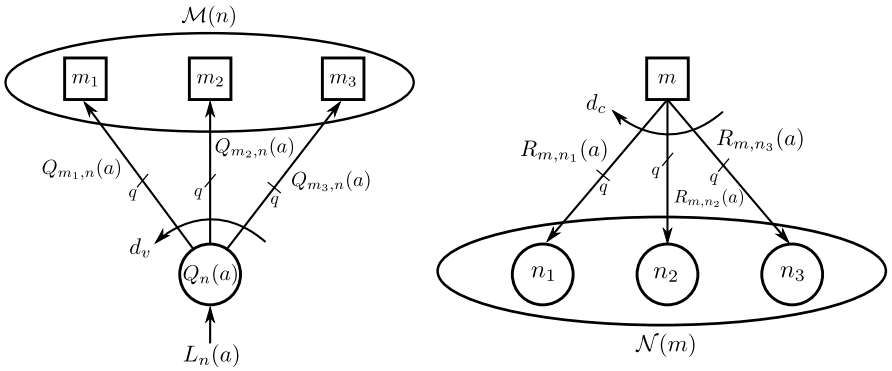


Figure 1.4: Detail of a Tanner graph used to show the nomenclature of the exchanged messages between nodes in NB-LDPC decoding algorithms.

1.3 Decoding schedules

There are two main methods to update the nodes during the iterations of the decoding process of an LDPC code: flooding and layered schedules. Flooding schedule uses the so-called two-phase node update, which consists on: a) first, all CN equations are solved by calculating the output reliabilities $R_{m,n}(a)$; b) then, the $Q_n(a)$ values of all VNs are computed solving for all VNs. The process is repeated until all parity-check equations are solved or a maximum number of iterations is reached. Algorithm 1 includes the general steps when flooding schedule is used to decode NB-LDPC codes. As explained before, the set $Q_{m,n}(a)$ is initialized with the channel information at the beginning of the decoding process. Function ϕ updates the $R_{m,n}(a)$ values, using the set $Q_{m,n}(a)$. The CN output messages and the channel information are employed to derive an updated version of the set

Algorithm 1: Flooding schedule

Inicialization:

$Q_{mn}^{(0)}(a) = L_n(a)$, $t = 1$

Main Loop:

while $t \leq iter$ **do**

CN Update:

1 $R_{mn}^{(t)}(a) = \phi\left(Q_{mn}^{(t-1)}(a) \in \mathcal{N}(m)\right)$

VN Update:

2 $Q_{mn}^{(t)}(a) = L_n(a) + \sum_{m' \in \mathcal{M}(n) \setminus m} R_{m'n}^{(t)}(a)$

Tentative Decoding:

3 $Q_n^{(t)}(a) = L_n(a) + \sum_{m' \in \mathcal{M}(n)} R_{m'n}^{(t)}(a)$

4 $\tilde{c}_n = \arg\min\left(Q_n^{(t)}(a)\right) \quad \forall\ a\ \in\ \mathrm{GF}(q)$

5 **if** $\tilde{\mathbf{c}} \times \mathbf{H^T} = 0$ **then** break

6 $t = t + 1$

end

Output: $\tilde{\mathbf{c}} = [\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_N]$

$Q_{m,n}(a)$ to be used in the next iteration. At the end of each decoding iteration, the total VN information $Q_n(a)$ is calculated to obtain the tentative codeword $\tilde{\mathbf{c}}$.

Layered schedule was originally proposed in the context of binary LDPC codes [22]. Despite this, the concepts are totally applicable for the non-binary case. The method consists on updating the nodes by steps, in a serial fashion. There are two ways to perform layered schedule: vertical (column) schedule and horizontal (row) schedule. In column layered schedule, one VN is processed at a time, then its connected CNs are updated. The steps are repeated until all VNs are processed. The horizontal schedule is the dual process since one CN is considered and then all its connected VNs are updated. Both ways of layered schedule are equivalent and achieve similar correction capability. In this thesis we only consider horizontal layered schedule since it fits better with the algorithms that are proposed. In this manuscript the terms layered schedule and horizontal layered schedule are used indistinctly.

The non-binary Tanner graph from Fig. 1.2.b is simplified in Fig. 1.5, omitting the non-zero coefficients of $\mathbf{H}$ in $\mathrm{GF}(2^p)$, to show an example of how the horizontal scheduling performs the decoding process. First CN $\{1\}$ is considered (Fig. 1.5.a), then VNs $\{1, 2\}$ are updated (Fig. 1.5.b). Secondly, CN $\{2\}$ is processed (Fig.

1.5.c) and then VNs $\{2, 4\}$ are updated (Fig. 1.5.d). The process is repeated until all CNs are processed.

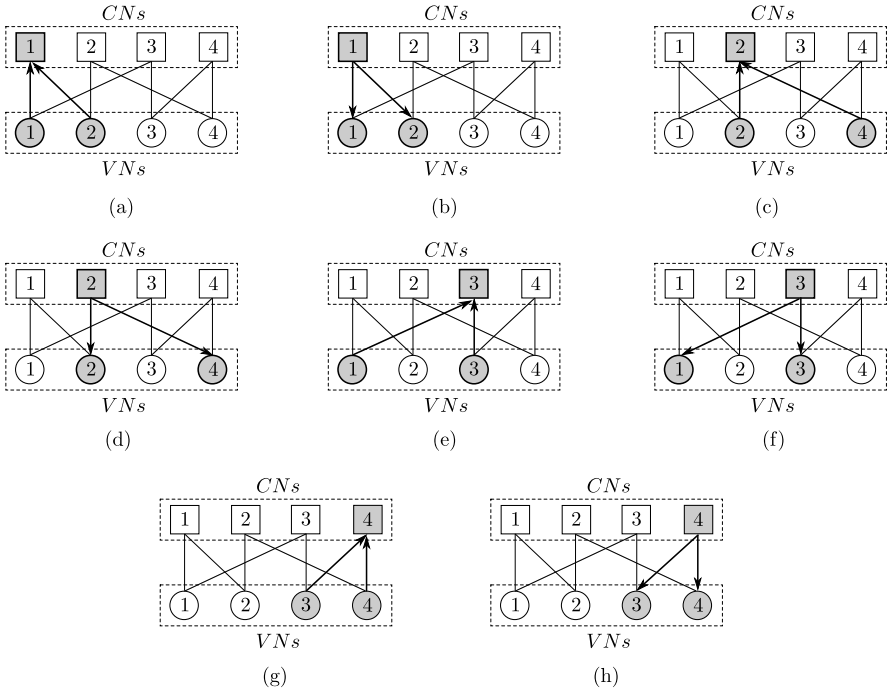


Figure 1.5: Example of horizontal layered schedule message exchange

Algorithm 2 includes the basic steps to implement an horizontal layered schedule decoder. Similar to the flooding schedule from Algorithm 1, function ϕ represents the specific CN processing algorithm, which uses as input the set $Q_{m,n}(a)$ incoming from its associated VNs. In order to avoid the use of GF-multipliers and, thus, to simplify the CN processing, the set of messages are permuted before the CN processing and inverse-permuted after it. The permutations are done using the non-zero $\mathbf{H}$ elements $h_{m,n}$.

One important advantage of the layered schedule is that the decoding algorithms require less iterations to converge compared to the flooding schedule [22]. The reason for the faster convergence is that each message in $R_{m,n}(a)$ is calculated using updated information from its connected VNs.

Algorithm 2: Horizontal layered schedule

Inicialization:

$Q_n^{(0)}(a) = L_n(a)$, $t = 1$

Main Loop:

while $t \leq Iter$ **do**

for $l = 1$ **to** M **do**

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

2 $R_{mn}^{(t)}(a) = \phi\left(Q_{mn}(a)\right)$

3 $Q_n^{(t)}(h_{mn}^{-1}a) = R_{mn}^{(t)}(a) + Q_{mn}(a)$

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

6 $t = t + 1$

end

Output: $\tilde{\mathbf{c}} = [\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_N]$

1.4 Decoding architectures

Depending on how many rows of the parity-check matrix are processed at a time, the decoding architectures can be classified into three basic groups: fully parallel, serial and partial parallel.

A fully parallel decoder processes all the rows of **H** at a time. This kind of decoders includes one CN processor per row and one VN processor per column. This kind of implementation is also called "two-phase decoder", since in one phase all the CN processors work and in the next one all the VN processors compute the incoming messages.

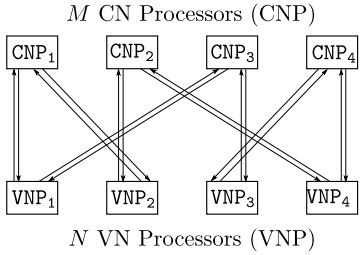


Figure 1.6: Example of a fully-parallel decoder for the code of Fig. 1.2

Fully parallel architectures are associated with flooding schedule since it allows the processing of all rows in **H** at the same time. These architectures can achieve high decoding speeds for short codes at the expense of larger area and they usually exhibit wiring congestion due to the high density of connections between processors, and this limits the maximum frequency. The throughput of fully parallel decoders is obtained using Eq. (1.1). f_{clk} corresponds to the operating clock frequency of the decoder, $iter$ the number of iterations for the algorithm, seg_{CN} and seg_{VN} correspond to the latency in the CN processor and VN processor respectively. In most cases, the CN involves the major computational load, therefore, the number of clock cycles per phase is limited by the CN.

$$Throughput_{fp} = \frac{N \times p \times f_{clk}}{iter \times (seg_{CN} + seg_{VN})} \tag{1.1}$$

The throughput can be almost duplicated by means of processing two codewords at a time. While the CNs process one codeword, the VNs perform computations over the other one.

To the best author's knowledge, just few works in literature [23, 24] have presented parallel NB-LDPC decoder implementations for short codes. For high-rate codes with medium codeword length, parallel implementations are still non-viable, due to the large area consumed and the routing congestion (that dramatically reduces the throughput). Hence, these architectures are no suitable for practical applications.

Serial based decoder architectures process one row of **H** at a time. They use a unique CN processor and interconnection networks for the message exchanges. Therefore, the required area is the lowest at the expense of penalizing the achievable throughput of the decoder. These decoders can be used with flooding or layered schedule, but are commonly used with the latter due to the advantages of this schedule seen on Section 1.3. An example of a serial decoder architecture is presented on Fig. 1.7, where it can be seen that d_v VN processors are required to process one row of **H**. Besides, it requires memories to store the CN-to-VN and VN-to-CN messages and interconnection networks to route the messages.

The throughput of the serial architecture can be defined by means of Eq. (1.2), where X represents the number of times that the pipeline stages of the decoder must be emptied per iteration to avoid memory conflicts.

$$Throughput_{serial} = \frac{N \times p \times f_{clk}}{it \times (M + (seg_{CN} + seg_{VN}) \times X)} \tag{1.2}$$

Due to the high complexity of the CN processor, most of the published implementations of NB-LDPC decoders use this kind of architecture with the aim of finding a balance between area and throughput.

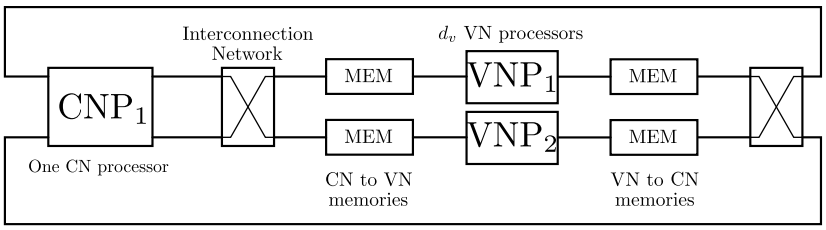


Figure 1.7: Example of a serial decoder for the code of Fig. 1.2. It requires $d_v = 2$ VN processors.

Finally, partial parallel decoders include more than one CN processor to increase the throughput compared to serial implementations without requiring the area of the fully parallel ones. The number of rows processed in parallel is selected taking into account the requirements of area and, on the other hand, the memory conflicts.

An example of a partial parallel decoder structure is shown on Fig. 1.8. The example corresponds to the Tanner graph from Fig. 1.2.b, implementing two CN processors. For the graph from Fig. 1.2.b, three VN processors are required to process the messages coming from the two CN processors. In general, the number of VNPs will depend on the particular **H** matrix that defines the LDPC code.

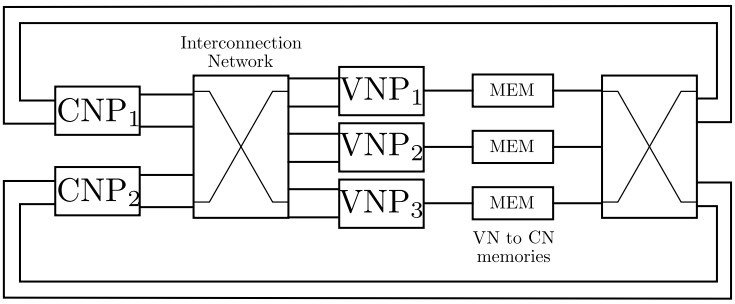


Figure 1.8: Example of a partial parallel decoder

When layered schedule is applied, the number of rows that can be processed in parallel can be maximized using parity-check matrices composed by circulant sub-matrices. This is only possible if the submatrices are of degree one ($d_c = 1$ and $d_v = 1$). An example of this kind of matrices is presented in Fig. 1.1.a, where **H** is formed by four circulant sub-matrices with individual dimensions 4×4.

This kind of Quasi-cyclic (QC) NB-LDPC codes allow us to process in parallel a maximum of QC check nodes, without losing the advantages of applying layered

schedule and avoiding memory access conflicts, being QC the size of each circulant sub-matrix.

1.5 NB-LDPC decoding algorithms and architectures

In this thesis we only consider soft-decision decoding algorithms since they offer the highest coding gain. However, there are hard-decision (HD) solutions recently proposed in literature that have reduced the existing gap between HD and SD decoding algorithms for high-rate codes [25, 26].

Generally, SD decoding algorithms process the CN input messages, $Q_{m,n}(a)$, using a general function as the one from Eq. 1.3, where Θ and ρ represent generic functions that depend on the particular algorithm used to decode the NB-LDPC code. The common functions are product, sum, min and max. Depending on which combination of functions is used in the decoding algorithm, the complexity in the decoder increases or decreases.

$$R_{m,n}(a) = \Theta\{\ \rho[\ Q_{m,n}(a)\ ]\ \} \tag{1.3}$$

On the other hand, there are two ways to solve the CN equations: serial or parallel. Serial computation of the CN equations is usually done by means of a recursive algorithm called forward-backward. Parallel computation in the CN is implemented by processing the CN input messages using a trellis structure. Both techniques to compute the CN output messages will be explained in detail later in this chapter.

When MacKay and Davey studied NB-LDPC codes, they introduced a generalization of the belief propagation decoding algorithm for LDPC codes based on high-order fields. They called it QSPA and it was defined originally in the probability domain [6].

In this algorithm, messages passing from VN to CN, $Q_{m,n}(a)$, correspond to the conditional probability that the symbol z_n is equal to the symbol $a \in \mathrm{GF}(q)$, given the information obtained from the set of CNs $\mathcal{M}(n) \setminus m$. Messages are calculated using (1.4), where $f_{m,n}$ is chosen such that $Q_{m,n}(a_1) + Q_{m,n}(a_2) + \cdots + Q_{m,n}(a_q) = 1$ and $P_n(a)$ corresponds to the prior probabilities of the nth received symbol z_n for each $a_1, a_2, \ldots, a_q$ GF(q) symbol[27].

$$Q_{m,n}(a) = f_{m,n} P_n(a) \prod_{m' \in \mathcal{M}(n) \setminus m} R_{m',n}(a) \tag{1.4}$$

On the other hand, messages from CN to VN are obtained using (1.5).

$$R_{m,n}(a) = \sum_{z_n=a} P(s_n = 0|\mathbf{z}, z_n = a) \cdot \prod_{n' \in \mathcal{N}(m)\backslash n} Q_{m,n'}(a) \tag{1.5}$$

As can be seen on Eq. 1.5, the generic function Θ is implemented as an addition and the ρ function involves a product over all the combinations of the CN input messages $Q_{m,n}(a)$, to calculate the sets of output messages $R_{m,n}(a)$.

In order to reduce the complexity of the original QSPA, new versions of the algorithm, in the frequency [28] and logarithmic domains [29], were proposed simplifying the CN calculations without introducing any performance loss compared to the algorithm from [6]. Even with these simplifications, hardware implementations were limited to short codes with Galois field orders lower than GF(16).

A recent proposal from literature is the one from [30], which introduces a trellis-based log-QPSA algorithm. This algorithm makes use of the configuration sets introduced in [14] to reduce the number of exchanged messages between VN and CN processor to $n_m \ll q$ and, at the same time, to reduce the number of operations in the CN processor. The performance loss, compared to QSPA, is controlled by means of the n_m parameter.

As an extension of the Min-Sum algorithm defined for binary LDPC decoders, the EMS decoding algorithm was proposed in [14]. It requires only additions and comparisons in the CN processing. Based on a reduced set of $n_m \ll q$ messages, this algorithm allows hardware implementations over high-order Galois fields due to the reduction in the complexity of the CN [31]. In this case, compared to the generic CN function (1.3), Θ is replaced by the maximum (max) operator and ρ is implemented by means of additions. The resultant function to obtain the CN output messages is included in (1.6), where $conf(n_m, n_c)$ corresponds to the configuration set [14] defined as the set of n_m symbols with higher reliability that differs in at most n_c symbols from the zero-order configuration, meeting the parity-check equation[2].

$$R_{m,n}(a) = \max_{a' \in conf(n_m,n_c)} \sum_{n' \in \mathcal{N}(m)\backslash n} Q_{m,n'}(a') \tag{1.6}$$

The max operator is used when the highest reliability is associated with the highest magnitude, otherwise, the min operator is selected. EMS is a sub-optimal algorithm that introduces a performance loss compared to QSPA. One of the main reasons for the degradation in the performance is due to an overestimation of the

[2]Zero-order configuration corresponds to the combination of symbols with the highest reliability that meet the parity-check equation

messages $R_{m,n}(a)$ [14]. An approach to reduce the performance loss is the use of a scaling factor or offset in the messages at the VN.

There is a proposal in the literature that further reduces the complexity of the EMS algorithm from [14]. The authors called it Simplified Min-Sum Algorithm (SMSA) [32] as their proposal reduces the complexity during the CN processing by means of the selection of a smaller set of incoming messages with higher reliability, which reduces the number of possible configuration sets involved in the computation of the CN output messages. They also propose an architecture that considerably increases the throughput compared to previous proposals from literature. Even so, the proposal from [32] is still far from being suitable for high-speed applications due to its inherent serial processing at the CN, which limits the maximum achievable decoding speed.

More recently, in [24] a fully parallel chip implementation based on the truncated EMS algorithm was proposed. This work truncates the least significant reliability values, keeping only the n_m most reliable ones ($n_m \ll q$). They implement a decoder for a short code over GF(64) achieving more than 1Gbps of throughput. Extrapolating their results for longer codes with higher rate, it would exhibit prohibitive chip area due to the fully-parallel implementation strategy. On the other hand, the throughput is limited due to the use of forward-backward metrics for the CN processing.

With the aim of eliminating the data-path growing in the CN output compared to the EMS algorithm, in [15] min-max algorithm is presented. The author proposes the implementation of the ρ function from (1.3) using the max operator instead of the sum from Eq. (1.6). Since no addition is made during the CN processing, there is no increase in the data-path width, in contrast with the use of Eq. (1.6). The CN update equation is modified as shown in Eq. (1.7), where it is assumed that the $Q_{m,n}(a)$ reliabilities are normalized with respect to the hard-decision symbol, therefore, Θ function is replaced by the minimum (min) operator. Eq. (1.7) also considers the q full set of reliability values. Furthermore, in [15] the author introduces a method to reduce the number of possible combinations to be analyzed, reducing the complexity of Eq. (1.7).

$$R_{m,n}(a) = \min_{a_{n'} | a_n + \sum_{n' \in \mathcal{N}(m) \setminus n} a'_n = 0} \max_{n' \in \mathcal{N}(m) \setminus n} Q_{m,n'}(a'_n) \tag{1.7}$$

The performance loss of min-max algorithm compared to QSPA is slightly higher than the one offered by EMS with the proper scaling or offset value.

Since the min-max algorithm was proposed [15] numerous works have been presented in literature with the aim of reduce its complexity that would allow the implementation of high-speed decoders. Two of these works are the ones from [33, 34]. The one in [33] introduces a modified shuffle (vertical layered) scheduled

decoder that reduces the complexity of the CN processor. The work from [34] presents a CN processor based on treating the CN messages in a trellis fashion. The complexity of the entire decoder was reduced compared to previous proposals from literature. The performance loss compared to the original min-max can be controlled depending of the number of messages taken into account in the CN processor. On the other hand, the throughput is limited since no parallel computations are made to process the trellis in the CN.

More recently, an approach based on the min-max algorithm has been presented in [23]. The authors proposed a method to reduce considerably the number of messages exchanged between CN and VN and vice versa. When it is required, the dismissed messages are approximated using a linear interpolation method. They implemented a fully-parallel decoder for short codes over high-order fields. Similar to the work from [24], the decoder may suffer from wiring congestion when uses codewords with medium to long lengths and/or high-rate codes with large check-node degree.

Most of the proposals previously mentioned use forward-backward metrics to derive the CN output messages in the decoder, so next, we include a short explanation of the algorithm. The forward-backward approach requires two matrices, $\mathbf{F}$ and $\mathbf{B}$, to store the intermediate results which are combined in a new matrix $\mathbf{M}$ in a final step.

Consider the example from Fig. 1.9, where for the sake of simplify the explanation, the non-zero $\mathbf{H}$ are taken as $\alpha^0 = 1$. The CN input messages are shown in Fig. 1.9.a being $q = 4$ and $d_c = 3$. The $\mathbf{F}$ matrix is filled serially column by column, where the first one (starting from the leftmost one) is a copy of the first column of the CN input messages ($Q_{m,1}$). The second column is filled with the combination of $Q_{m,2}$ and the first column of $\mathbf{F}$ ($\mathbf{F_1}$). The combination is computed following the rules of the particular algorithm implemented in the decoder. For the case of this example we use min-max. As an example, equations (1.8) and (1.9) show how the elements $\mathbf{F_1}(0)$ and $\mathbf{F_1}(\alpha^0)$ are calculated. The rest of elements in $\mathbf{F}$ are calculated in a similar way. It is important to remark that calculating the elements of the column i, requires that the elements of the column $i-1$ have to be already computed. Therefore, it is a recursive process that is slower when the CN degree gets higher.

$$\mathbf{F_2}(0) = \min \left\{ \begin{array}{l} \max\{\mathbf{F_1}(0), Q_{m,2}(0)\} = 3 \\ \max\{\mathbf{F_1}(\alpha^0), Q_{m,2}(\alpha^0)\} = 10 \\ \max\{\mathbf{F_1}(\alpha^1), Q_{m,2}(\alpha^1)\} = 12 \\ \max\{\mathbf{F_1}(\alpha^2), Q_{m,2}(\alpha^2)\} = 6 \end{array} \right\} = 3 \tag{1.8}$$

$Q_{m,n}(a) =$

	$Q_{m,1}$	$Q_{m,2}$	$Q_{m,3}$
0	1	3	14
α^0	0	10	12
α^1	12	7	8
α^2	6	4	5

(a)

$\mathbf{F} =$

	$\mathbf{F_1}$	$\mathbf{F_2}$	$\mathbf{F_3}$
0	1	3	5
α^0	0	3	5
α^1	12	4	5
α^2	6	4	5

(b)

$\mathbf{B} =$

	$\mathbf{B_1}$	$\mathbf{B_2}$	$\mathbf{B_3}$
0	5	5	14
α^0	5	7	12
α^1	5	8	8
α^2	5	5	5

(c)

$\mathbf{M} =$

	$\mathbf{M_1}$	$\mathbf{M_2}$	$\mathbf{M_3}$
0	5	6	3
α^0	7	8	3
α^1	8	5	4
α^2	5	5	4

(d)

Figure 1.9: Forward Backward example

$$\mathbf{F_2}(\alpha^0) = \min \left\{ \begin{array}{l} \max\{\mathbf{F_1}(0), Q_{m,2}(\alpha^0)\} = 10 \\ \max\{\mathbf{F_1}(\alpha^0), Q_{m,2}(0)\} = 3 \\ \max\{\mathbf{F_1}(\alpha^1), Q_{m,2}(\alpha^2)\} = 12 \\ \max\{\mathbf{F_1}(\alpha^2), Q_{m,2}(\alpha^1)\} = 7 \end{array} \right\} = 3 \tag{1.9}$$

In a similar way, the matrix $\mathbf{B}$ is processed. In this case, the last column ($\mathbf{B_3}$ in the example of Fig. 1.9.c) is filled with the column $Q_{m,3}$. The rest of columns are computed in a way similar to the $\mathbf{F}$ matrix but filling the columns from left to right. For example, the element $\mathbf{B_2}(\alpha^1)$ is calculated as shown in Eq. (1.10).

$$\mathbf{B_2}(\alpha^1) = \min \left\{ \begin{array}{l} \max\{\mathbf{B_3}(0), Q_{m,2}(\alpha^0)\} = 14 \\ \max\{\mathbf{B_3}(\alpha^0), Q_{m,2}(0)\} = 12 \\ \max\{\mathbf{B_3}(\alpha^1), Q_{m,2}(\alpha^2)\} = 8 \\ \max\{\mathbf{B_3}(\alpha^2), Q_{m,2}(\alpha^1)\} = 10 \end{array} \right\} = 8 \tag{1.10}$$

Finally, to process the elements of the matrix $\mathbf{M}$, matrices $\mathbf{F}$ and $\mathbf{B}$ are required. In this case, the rule of combination is that to fill elements in the column i, the $i-1$ and $i+1$ column of $\mathbf{F}$ and $\mathbf{B}$, respectively, are required. For the first and

last column of $\mathbf{M}$, $\mathbf{M_1} = \mathbf{B_2}$ and $\mathbf{M_{d_c}} = \mathbf{F_{d_c-1}}$. For example, for the case in Fig. 1.9.d, the element $\mathbf{M_2}(\alpha^2)$ is calculated as shown in Eq. (1.11).

$$\mathbf{M_2}(\alpha^2) = \min \left\{ \begin{array}{l} \max\{\mathbf{F_1}(0), \mathbf{B_3}(\alpha^2)\} = 5 \\ \max\{\mathbf{F_1}(\alpha^0), \mathbf{B_3}(\alpha^1)\} = 8 \\ \max\{\mathbf{F_1}(\alpha^1), \mathbf{B_3}(\alpha^0)\} = 12 \\ \max\{\mathbf{F_1}(\alpha^2), \mathbf{B_3}(0)\} = 14 \end{array} \right\} = 5 \tag{1.11}$$

Due to the data dependency, a total of $3 \times d_c - 4$ clock cycles are required to calculate the three matrices. This number of clock cycles can be reduced to $d_c - 1$ if the calculation of $\mathbf{M}$ starts immediately when the required values from $\mathbf{F}$ and $\mathbf{B}$ are available. This reduction in the latency comes with an increase in the hardware area. To further reduce the latency other approaches to derive the CN output messages must be employed.

1.5.1 Trellis Extended Min-Sum Algorithm

Trellis Extended Min-Sum (T-EMS) [35, 16] was presented as a new computation method for the EMS algorithm, treating the messages in the CN processor in a trellis structure. This fact allows the parallel processing of messages, which is the main drawback of other proposals from literature. The basic steps for T-EMS are presented in Algorithm 3.

Algorithm 3: T-EMS Algorithm

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

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

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

end

2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in GF(q)$

3 $\Delta Q(a) = \min_{\eta'_j(a) \in conf(n_r, n_c)} \sum_{j=1}^{d_c} \Delta Q_{m,n_j}(\eta'_j(a)), a \in GF(q)$

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

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

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

end

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

As a difference with the proposal from [14], which processes a reduced set of $n_m < q$ messages, T-EMS algorithm works with the full set of q reliabilities. On the other hand, the algorithm assumes that the reliabilities are normalized in such a way that the most reliable symbol z_n corresponds to a value of zero

(highest reliability), being all reliabilities non-negative values. Previous to the computation of the CN output messages, the algorithm requires a transformation of the $Q_{m,n}(a)$ messages to the so-called "delta domain" (Step 1), where the set $Q_{m,n}(a)$ is reordered in a way that the most reliable symbol corresponds to the $\alpha^{-\infty}$ symbol once is transformed.

Step 2 involves the syndrome (β) calculation. It is obtained adding, in a GF domain, all tentative hard-decision symbols z_n.

Messages in delta domain can be interpreted as organized in a trellis, where the rows are the GF symbols sorted by their exponent and each column (each stage of the trellis) corresponds to one of the d_c CN input messages, as shown in Fig 1.10.a. To enable the parallel computation of CN output messages, an extra column is added to the trellis and it is filled with the reliability of the syndrome, obtained using Step 3 of Algorithm 3. It can be seen in Step 3 that the addition of the reliability values $\Delta Q_{m,n}(a)$ is made for the set of symbols $\eta'_j(a)$ which is selected to be within the configuration set $conf(n_r, n_c)$. This is done to reduce the possible combination of symbols to only the ones that ensure the highest possible reliability for the $\Delta Q(a)$ value.

The configuration set $conf(n_r, n_c)$ selects the set of at most n_c most reliable symbols per row of the trellis. From this reduced set of $n_c \times (q-1)$ symbols it selects the combination of them that produces at most n_r deviations from the hard-decision path, fulfilling the parity-check equation for each symbol a.

From all the possible combinations of symbols deviating at most n_r times from the hard-decision path, the selected one ensures the highest reliability i.e. the one with the minimum numerical value.

$\mathbf{Q_{m,n}} =$

	$Q_{m,1}$	$Q_{m,2}$	$Q_{m,3}$	$Q_{m,4}$
$\alpha^{-\infty}$	53	8	68	36
α^0	4	**0**	91	41
α^1	16	1	86	3
α^2	**0**	36	13	**0**
α^3	30	10	26	61
α^4	92	58	64	31
α^5	11	5	**0**	35
α^6	2	72	92	15

$\mathbf{\Delta Q_{m,n}} =$

	$\Delta Q_{m,1}$	$\Delta Q_{m,2}$	$\Delta Q_{m,3}$	$\Delta Q_{m,4}$	ΔQ
$\alpha^{-\infty}$	**0**	**0**	**0**	**0**	**0**
α^0	2	8	64	15	2
α^1	92	10	92	31	3
α^2	53	72	26	36	13
α^3	11	1	13	35	1
α^4	16	5	91	3	3
α^5	30	58	68	61	5
α^6	4	36	86	41	4

Figure 1.10: Example of the set $Q_{m,n}(a)$ and $\Delta Q_{m,n}(a)$ including the extra column $\Delta Q(a)$. $q = 8$ and $d_c = 4$ in this example.

In Fig. 1.10 an example of the set of CN input messages $Q_{m,n}(a)$ and the ones after the transformation to the delta domain is presented. The example considers GF(8) and $d_c = 4$. The hard-decision symbols are $\mathbf{z} = [\alpha^2, \alpha^0, \alpha^5, \alpha^2]$ and the

syndrome is $\beta = \alpha^2 + \alpha^0 + \alpha^5 + \alpha^2 = \alpha^4$. In the delta-domain set $\Delta Q_{m,n}(a)$ we include the extra column $\Delta Q(a)$. The values of $\Delta Q(a)$ are calculated using Step 3 of Algorithm 3 considering $n_r = 2$ and $n_c = 1$. As an example, next we explain the procedure used to calculate the values of $\Delta Q(\alpha^0)$ and $\Delta Q(\alpha^1)$.

First of all, since $n_c = 1$, only the most reliable symbol per row of the trellis is considered to build possible paths by making deviations from the hard-decision one. These values are marked on the set $\Delta Q_{m,n}(a)$ in Fig. 1.10.b rounding them using dashed squares. To select the most reliable value for $\Delta Q(\alpha^0)$, three possible paths are considered. They are shown in Fig. 1.11.a: i) the one-deviation path passing through the coordinates $(\alpha^0, 0)$; $(0, 1)$; $(0, 2)$ and $(0, 3)$ with reliability equal to 2+0+0+0=2. ii) a two-deviation path passing through $(\alpha^6, 0)$; $(0, 1)$; $(\alpha^2, 2)$ and $(0, 3)$ with reliability 4+0+26+0 = 30. iii) another two-deviation path passing through $(\alpha^5, 0)$; $(0, 1)$; $(0, 2)$ and $(\alpha^4, 3)$ with reliability 30+0+0+3=33. The two-deviation path passing through the symbols α^1 and α^3 is neglected since it would deviate twice at the same column of the trellis (it is not shown in Fig. 1.11.a). To select the most reliable value for $\Delta Q(\alpha^0)$, the one with the minimum value (highest reliability) is chosen. Therefore, the value 2 obtained by the one-deviation path is selected.

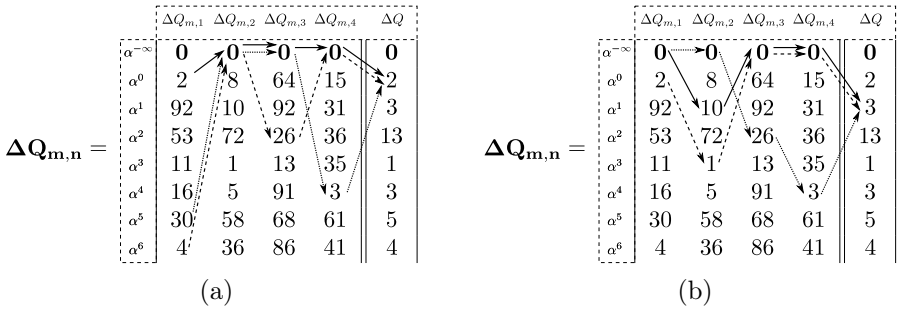


Figure 1.11: (a) Example of paths taken into account to compute the $\Delta Q(\alpha^0)$ reliability. (b) Example for the $\Delta Q(\alpha^1)$ reliability.

For the $\Delta Q(\alpha^1)$ value a similar procedure is applied (Fig. 1.11.b). In this case, the possible paths are: i) the one-deviation path passing through the coordinates $(0, 0)$; $(\alpha^1, 1)$; $(0, 2)$ and $(0, 3)$ with reliability equal to 0+10+0+0=10. ii) a two-deviation path passing through $(\alpha^0, 0)$; $(\alpha^3, 1)$; $(0, 2)$ and $(0, 3)$ with reliability 2+1+0+0 = 3. iii) another two-deviation path passing through $(0, 0)$; $(0, 1)$; $(\alpha^2, 2)$ and $(\alpha^4, 3)$ with reliability 0+0+26+3=29. The one with the highest reliability is selected, so, $\Delta Q(\alpha^1) = 1$ passing through the coordinates $(\alpha^0, 0)$; $(\alpha^3, 1)$; $(0, 2)$ and $(0, 3)$. The two-deviation path passing through the symbols α^5 and α^6 is not taken into account since it deviates more than once at the same column of the trellis. The rest of values for the extra column $\Delta Q(a)$ are obtained in a similar way to the ones explained in the example.

Output messages in delta-domain are calculated using Step 4 of Algorithm 3. According to [16], the set $\Delta R_{m,n}(a)$ is initialized with the lowest reliability (maximum value of the quantization scheme) and the set $\eta'_j(a)$ corresponds to the rows (symbols) which conform the most reliable path for each symbol a. Therefore, the positions where no deviations were made are directly filled with the corresponding $\Delta Q(a)$ reliability. After computing all j and a values, there are some positions in the $\Delta R_{m,n}(a)$ set that are not filled using Step 4. These positions correspond to the ones where deviations are made. Two different cases can be distinguished : i) in the one-deviation cases, the second minimum of the corresponding row can be used to fill the empty space; ii) when two o more deviations are made in a row of the trellis, the empty spaces are filled either with the first minimum or the second one depending if the first minimum is in the corresponding column [16].

Step 5 performs inverse transformation from delta to normal domain over the set $\Delta R_{m,n}(a)$. It must be noted that the syndrome β is used adding it to the hard-decision symbols. On the other hand, an scaling value λ is commonly used to improve the FER performance.

Analyzing the implementation results from [17], where a T-EMS based decoder is presented, we conclude that the parallel trellis-based processing in the CN is the best solution for high-rate codes over high-order Galois fields. This is the main reason to use it as starting point to develop the solutions presented in this manuscript.

1.6 Frame Error Rate (FER) Performance

Fig. 1.12 includes the FER performance of the algorithms shortly discussed in this part of the manuscript. The code under test is the quasi-cyclic (837,726) NB-LDPC code over GF(32) constructed using the methods described in [36]. BPSK modulation and AWGN channel are assumed. The maximum number of iterations for all algorithms is 15.

QSPA algorithm offers the highest coding gain for the high-rate code under test at the expenses of the highest complexity for the CN processor, as explained before. EMS and T-EMS introduce 0.05dB of performance loss compared to QSPA at FER $= 10^{-4}$. Min-Max algorithm introduces 0.05dB of performance loss compared to EMS and T-EMS algorithms (almost 0.1dB compared to QSPA).

In terms of hardware complexity, a direct implementation of the QSPA algorithm involves products which increase considerably the area and introduce extra latency making it non practical for applications that demand high-speed and reasonable area. On the other hand, with EMS and Min-Max the products are replaced by adders and comparisons, respectively. These operations have a smaller cost in

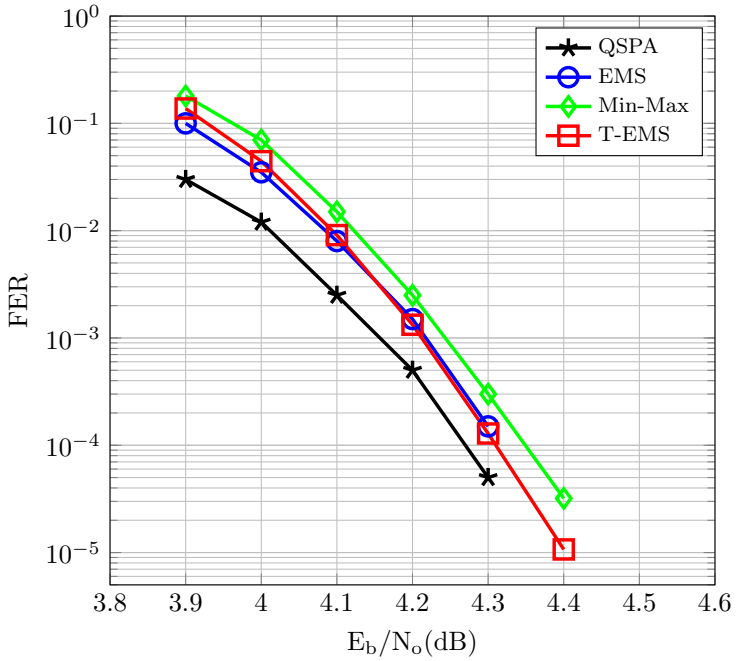


Figure 1.12: FER performance for QSPA, EMS, Min-Max and T-EMS decoding algorithms for the (837,726) NB-LDPC code over GF(32)

terms of hardware resources which compensate the performance loss compared to the QPSA.

With T-EMS the latency in the CN is reduced due to the parallel computation of messages in the CN, avoiding the use of the forward-backward algorithm which limits the maximum achievable throughput, as explained before. The gain in throughput is higher than the growth in area, compared to conventional implementations of EMS. Additionally, the coding gain is maintained (Fig. 1.12). Thus, the potential of this algorithm for high-speed applications is clear.

Since one of the objectives in this manuscript is the design of hardware architectures to implement the proposed decoding algorithms, it is important to establish the way how the finite precision analysis is carried out for the T-EMS algorithm, since it is the starting point for the proposals included in this manuscript.

First of all, we consider the case where the LLR values are quantized but the messages inside the decoder have full precision. This analysis is useful to understand how the quantized input affects the FER performance of the decoding algorithm. Fig. 1.13 includes the FER analysis for the high-rate (837,726) NB-LDPC code

over GF(32). The format $r.s$ means that for a data-path of r bits, s bits represent the fractional part.

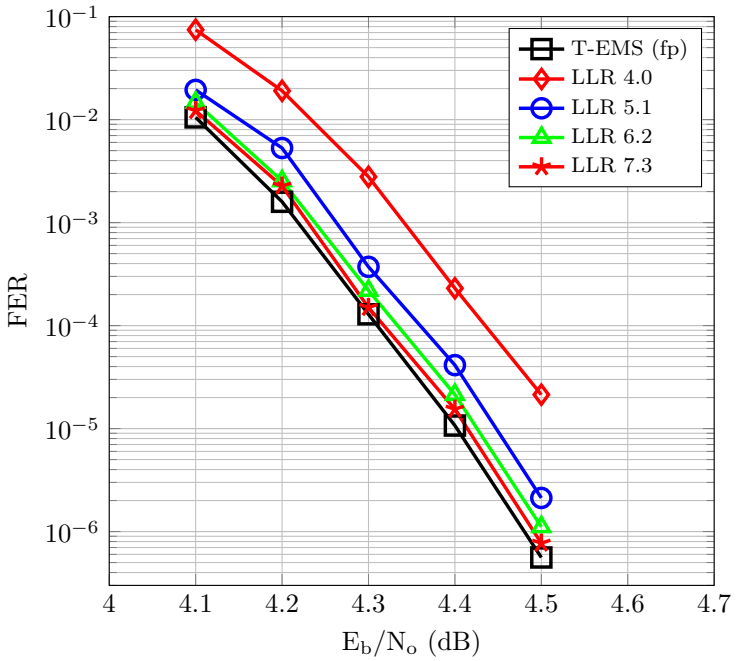


Figure 1.13: FER performance for T-EMS algorithm quantifying the LLR values. The test code is the (837,726) NB-LDPC code over GF(32).

The selection of the quantization model is a compromise between area (memory resources) and performance. The 4-bit LLRs introduce a performance loss of 0.15dB in the FER compared to a floating point (fp) LLR representation. This loss is unacceptable if we take into account that it will increase when the entire decoder data-path is quantized. On the other hand, with 6.2 and 7.3 formats, the performance loss is almost negligible, but at a expense of more memory resources. A good choice is the format 5.1, which introduces a performance loss of only 0.06dB compared to the fp implementation.

The next step in the finite precision analysis is the study of the number of bits required for the data-path taking into account the data growth in the decoder. For the T-EMS algorithm, Step 3 is the only one where the message values are increased. As can be seen in the following chapters, considering $n_r = 2$ implies that only two reliability values are added, therefore, increasing only one bit is enough for the CN implementation.

In the VN processor the messages from the associated CNs and the LLRs are accumulated in each decoding iteration. There are some factors that make that the number of required bits in the data-path does not increase a lot compared to the LLR values.

- CN output messages are usually scaled by a $\lambda < 1$ value.
- Messages $R_{mn}(a)$ are formed considering only the most reliable CN input messages, which are the ones with the lower magnitudes.

In Fig. 1.14 we show the results of the finite precision analysis for T-EMS algorithm considering a format of 5.1 for the LLR values. If we do not consider any increase in the data-path width, the FER performance is greatly degraded inducing an early error-floor. On the other hand, increasing only one bit in the data-path compared to the quantized LLR, a performance loss of 0.07dB is introduced compared to the fp version of the algorithm.

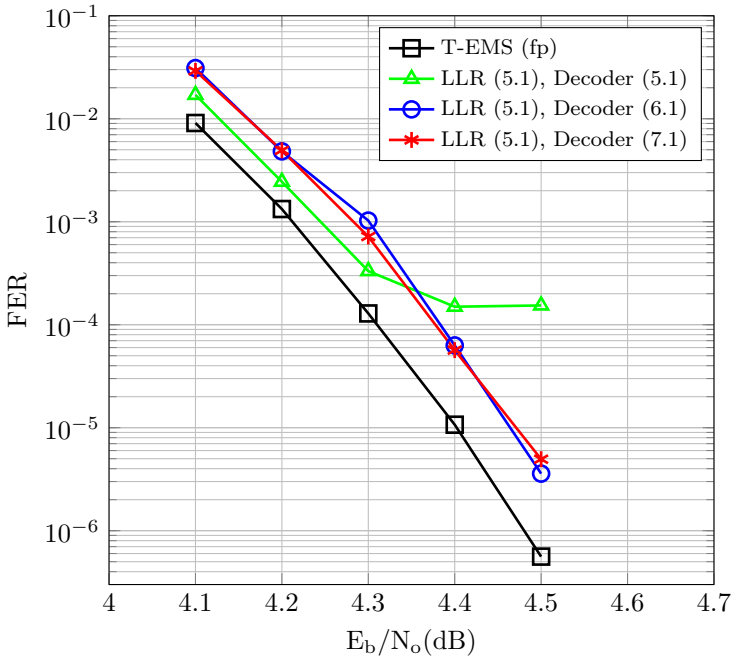


Figure 1.14: Fixed point analysis for the T-EMS algorithm. The test code is the (837,726) NB-LDPC code over GF(32). Layered schedule and 15 decoding iterations are used for all cases.

1.7 Conclusions of the state of the art

In this chapter the basis of LDPC codes are presented, including the nomenclature and the advantages of the non-binary LDPC codes compared to the binary ones. The decoding schedules have been analyzed concluding that the layered one is a good choice to accomplish the objectives proposed in this manuscript thanks to the higher convergence speed compared to the flooding schedule and, on the other hand, its inherent good coupling with quasi-cyclic parity-check matrices.

The most common soft-decision decoding algorithms were briefly introduced (QSPA, Min-Sum, min-max), including the CN update equations and main drawbacks of each one of them. We also included a more detailed description of the Trellis Min-Sum algorithm, since it was selected as the starting point to develop the contributions of this thesis due to its inherent parallel processing of messages which can potentially increase the decoding speed compared to other proposals from literature.

Making use of parallel computation of messages (trellis) it is possible to achieve high decoding speeds at the expense of an increase in the area of the CN processor. Therefore, after analyzing the decoding architectures (full parallel, partial parallel and serial) we conclude that it is inviable to implement more than one CN processor working in parallel for high-rate and medium to large code lengths, due to the increase in area and to avoid possible wiring congestion due to the exchange of messages between processors.

Finally, a FER performance analysis for the high-rate code under test was included to establish comparisons between the algorithms described in this chapter. Besides, we include a summarized finite-precision analysis for the T-EMS algorithm.

Chapter 2

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

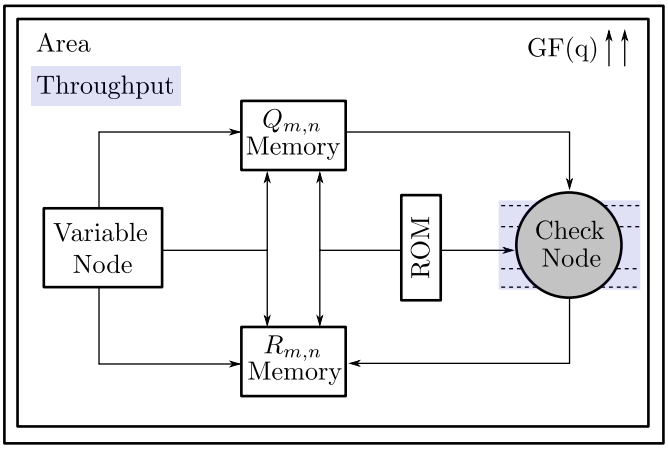


Figure 2.1: Key points of the improvements presented in this chapter.

In this chapter, the paper "Simplified Trellis Min-Max Decoder Architecture for Non-Binary Low-Density Parity-Check Codes" is presented. This research takes the T-EMS algorithm from [16] as starting point to develop a new simplified algorithm, where the complexity is decreased improving the latency in the CN processor and also the maximum achieveable throughput, compared to previous proposals from literature.

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 compared to the existing recent solutions for the same code.

Index terms: Layered decoder, message passing algorithm, nonbinary low-density parity-check (NB-LDPC), trellis - min-max (TMM).

2.1 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 $\mathbf{H}$ with M rows and N columns. Each non-zero element $h_{m,n}$ of $\mathbf{H}$ belongs to the Galois field $GF(q = 2^p)$. In this paper we only consider regular NB-LDPC codes with constant row weight d_c and column weight d_v. NB-LDPC codes can also be characterized by a bipartite graph called Tanner Graph[20], where two types of nodes can be differentiated, the ones representing the rows of the parity check matrix called check nodes (CN) and the ones that represent the columns in $\mathbf{H}$, called variable nodes (VN). Decoding algorithms for NB-LDPC codes use iterative message exchange between 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) [6]. The complexity of QSPA is too large to be suitable for hardware implementations, and several approaches such as FFT-SPA[28], log-SPA

and max-log-SPA [29], were proposed to overcome its limitations. These solutions reduce the complexity of the check node processing equations without introducing any performance loss. In [14] 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 [15], 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 [35] 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 [35] was improved in [16] 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 [16] 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 a 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 over 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 efficients decoder proposed in literature [34, 16], 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 algorithm 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.

2.2 Trellis Extended Min-Sum Algorithm

A NB-LDPC code is defined by its parity check matrix $\mathbf{H}$ with M rows and N columns. Each non-zero element $h_{m,n}$ of $\mathbf{H}$ 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 [14, 37]. An NB-LDPC code can be either regular, that is with constant row weight d_c and column weight d_v, 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 $\mathcal{N}(m)$ ($\mathcal{M}(n)$) be the set of variable nodes (check nodes) connected to a check node (variable node) m (n). Let $Q_{m,n}(a)$ and $R_{m,n}(a)$ be the messages from variable node to check node and from check node to variable node 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 n being equal to a, seen from the check-node m. Accordingly, $L_n(a)$ denotes the channel information for symbol n and $Q_n(a)$ its *a posteriori* information.

Let $\mathbf{c} = c_1, c_2, \cdots, c_N$ and $\mathbf{y} = y_1, y_2, \cdots, y_N$ be the transmitted codeword and received noisy symbol sequence respectively. The log-likelihood ratio (LLR) for each received symbol is obtained as $L_n(a) = \log[P(c_n = z_n|y_n)/P(c_n = a|y_n)]$ where z_n is the symbol associated to the highest reliability. The previous definition ensures that all messages $L_n(a)$ are non-negative and that the smaller the value, the more reliable the message.

We present in Algorithm 4 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 $\eta_j = a + z_{n_j}$ 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 β using the most reliable symbol z_n for each check node incoming message. For the syndrome computation, all nonzero elements of $\mathbf{H}$ are taken as $\alpha^0 = 1$ thanks to the pre-processing of the incoming messages outside of the node, as will be explained in later sections.

Step 3 makes use of the configuration sets originally proposed in [14] with the aim of building the output messages by just using the most reliable information. $conf(n_r, n_c)$ is defined as the configuration set composed of the most reliable paths

Algorithm 4: T-EMS Algorithm

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

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

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

end

2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in GF(q)$

3 $\Delta Q(a) = \min_{\eta'_j(a) \in conf(n_r,n_c)} \sum_{j=1}^{d_c} \Delta Q_{m,n_j}(\eta'_j(a)), a \in GF(q)$

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

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

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

end

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

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 [14, 38]. 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 η_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 $conf(n_r, n_c)$ 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 [16].

Output messages in delta domain $\Delta R_{m,n_j}(a)$ are generated subtracting the reliability of configurations to the information collected in the extra column of trellis $\Delta Q(a)$ (step 4 of Algorithm 4). 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 hard decision symbols z_n and the syndrome value β. Before the inverse transformation, a scaling factor λ can be applied to outgoing check node messages to improve the performance of the decoding algorithm.

Although very appealing in terms of throughput, the implementation of step 3 requires many computations in parallel, which increases the overall complexity of the decoder. This is especially true when high order fields and large check node degree d_c are considered. In the next section, we introduce several simplifications to Algorithm 4, greatly reducing the complexity of the check node unit and improving both latency and area of the global decoder architecture.

2.3 Simplified Trellis Min-Max Algorithm

In the T-EMS, the output message calculation (Step 4 of Algorithm 4) involves $q \times d_c$ subtractions and also $q \times d_c$ minimum finders (min finders) which becomes the bottleneck of the check node processing. Taking into account this drawback, we propose a simplified algorithm which reduces considerably the processing load of the check node messages (it avoids the use of subtractions and minimum finder) without inducing any performance loss.

2.3.1 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 min1-min1 messages are considered leaving out combinations of min1-min2, min2-min1 and 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 2.2, the extra column $\Delta Q(a)$ contains the paths formed by the most reliable combination of symbols (Step 3 of Algorithm 4). 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 [15] 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 (2.1) the modifications made on step 3 of Algorithm 4 are presented, converting the T-EMS on T-Min-Max algorithm (TMM).

$$\Delta Q(a) = \min_{\eta'_j(a) \in conf(n_r, n_c)} \left\{ \max_{j=1 \to d_c} \left(\Delta Q_{m,n_j}(\eta'_j(a)) \right) \right\}, a \in GF(q) \tag{2.1}$$

The complete description of the proposed TMM algorithm is presented in Algorithm 5, where step 3 of Algorithm 4 has been split into two basic tasks. In step 3 of Algorithm 5 function ψ 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 ψ function also extracts the position of the most reliable message ($m1_{col}(a)$), so it can take values from one to d_c. Step 4 of Algorithm 5 involves the processing of the trellis extra column reliability using information related to the most reliable symbol $m1(a)$. The configuration set $conf(n_r, n_c)$ from Algorithm 4 [16] includes the set of symbols $\eta'_j(a)$ which contains information about all nodes through which pass the configuration. In this approach we redefined the configuration set to $conf^*(n_r, n_c)$, where the difference is that $\eta'_k(a)$ only retains information from the n_c columns where deviations from the zero-order configuration are made instead of keeping information from all nodes. This 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 ($n_r = 1$) 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 5 as we will see in next sections.

Algorithm 5: Simplified TMM Algorithm

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

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

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

end

2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in GF(q)$

3 $[m1(a), m1_{col}(a), m2(a)] = \psi\{\Delta Q_{m,n_i}(a)\Big|_{i=1}^{d_c}\}$

4 $\Delta Q(a) = \min_{\eta'_k(a) \in conf^*(1,2)} \left\{ \max_{k=1,2} \left(m1(\eta'_k(a)) \right) \right\}$

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

5 $\quad$ **if** $\eta'_1(a) \neq j$ **or** $\eta'_2(a) \neq j$ **then**
$\quad\quad \Delta R_{m,n_j}(a) = \Delta Q(a)$
$\quad$ **else if** $\eta'_1(a) = \eta'_2(a)$ **then**
$\quad\quad \Delta R_{m,n_j}(a) = m2(a)$
$\quad$ **else**
$\quad\quad \Delta R_{m,n_j}(a) = m1(a)$
$\quad$ **end**

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

end

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

Step 5 of Algorithm 5 presents a simplified way to obtain delta domain output messages using a simple assignation of $\Delta Q(a)$, $m1(a)$ or $m2(a)$ depending only on the deviation information from $\eta'_k(a)$. If no deviation for the most reliable path is made on column j for symbol a, then extra column information $\Delta Q(a)$ is directly assigned to the corresponding output message $\Delta R_{m,n_j}(a)$. On the other hand, if any deviation is made for column j and the corresponding path is build with only one deviation, then the second most reliable message for symbol a ($m2(a)$) is assigned to the corresponding output message. In the case of paths formed by more than one deviation, $m1(a)$ is assigned to the output message.

Step 6 of Algorithm 5 depicts the transformation of delta domain messages to the normal domain, including a scaling factor λ to improve the performance of the decoder in the waterfall region. The scaling factor value λ is selected among the possible hardware friendly values that do not increase the area of the decoder.

2.3.2 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 $\mathbf{H}$ is generated using the methods in [36], with $d_c = 27$ and $d_v = 4$, and using transmission over BPSK modulation and AWGN channel. We compare the TMM to the QSPA [6], and the recently published Relaxed Min-Max (RMM) [34] and T-EMS algorithms [17]. On Fig. 2.2, 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 QPSA performance. The configuration set parameters are $n_r = 1$ and $n_c = 2$ for the TMM algorithm although for T-EMS algorithm $n_r = 2$, $n_c = 2$ were used. For T-EMS algorithm the optimum λ was selected and 15 iterations (it) for the iterative decoding were used. For T-MM approach, λ value was set to 0.5, since this value don't requires extra hardware for implementation purposes. Despite this, using the optimum λ value the performance of T-EMS of T-MM approach are very similar. In Fig. 2.2 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 [6], 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 performace loss with respect to fp implementation with the same number of iterations.

The proposed quantized approach with 9 iterations was compared with RMM [34]. For RMM algorithm 5 bits are used for the datapath and the number of iterations are set to 15. In Fig. 2.2 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 [34], as we will see in the next sections.

2.4 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. 2.3, 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

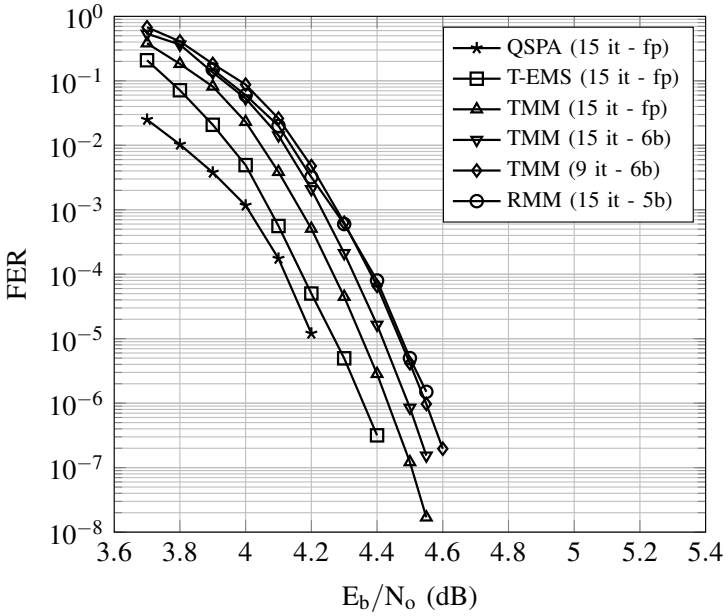


Figure 2.2: FER of (837,726) NB-LDPC over $GF(32)$ under AWGN channel. Layered schedule is used for all algorithms. $\lambda = 0.375$ for T-EMS algorithm and $\lambda = 0.5$ for TMM algorithm.

a permutation network similar to the one proposed in [39]. This network requires $q \cdot \log_2(q)$ multiplexors of two inputs to perform the delta domain transformation of each input vector message $\mathbf{Q_{m,n}}$. Therefore, the check node requires d_c permutation networks where multiplexors are addressed by tentative hard decision symbols z_n. The same structure is used for inverse transformation to normal domain applied to output messages $\Delta R_{m,n}(a)$, where instead of addressing multiplexors using tentative hard decisions symbols, $z_n + \beta$ sum is applied. The check node syndrome β 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 ψ, 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 [40]. Only $q-1$ cells are required to implement all ψ 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 ψ function requires d_c inputs because of the processing based on the trellis reordering of the delta messages. The approach followed to implement the

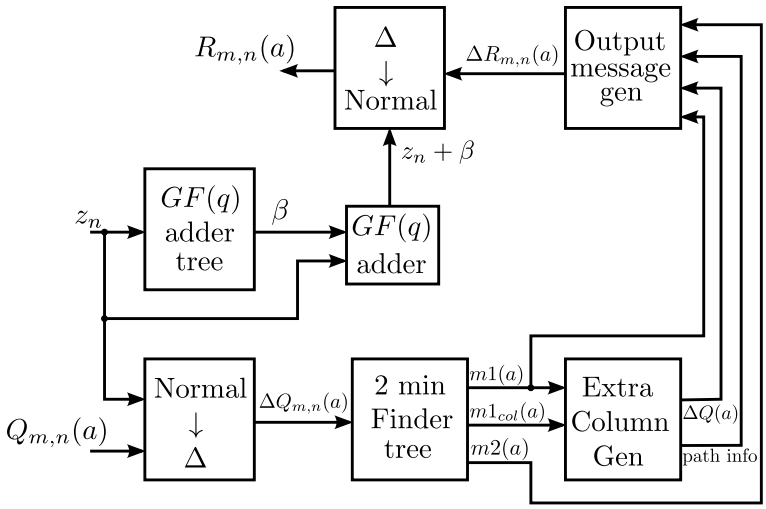


Figure 2.3: Proposed top level check node structure.

ψ function is the tree structure proposed on [40] 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 2.3. The architecture designed for building the extra column is presented in Fig. 2.4. 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. 2.4, 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. 2.4, 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 α^0 on Fig. 2.4) is passed to the block in charge of finding the most reliable path for the corresponding symbol a ("1 mind find" block of Fig. 2.4).

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

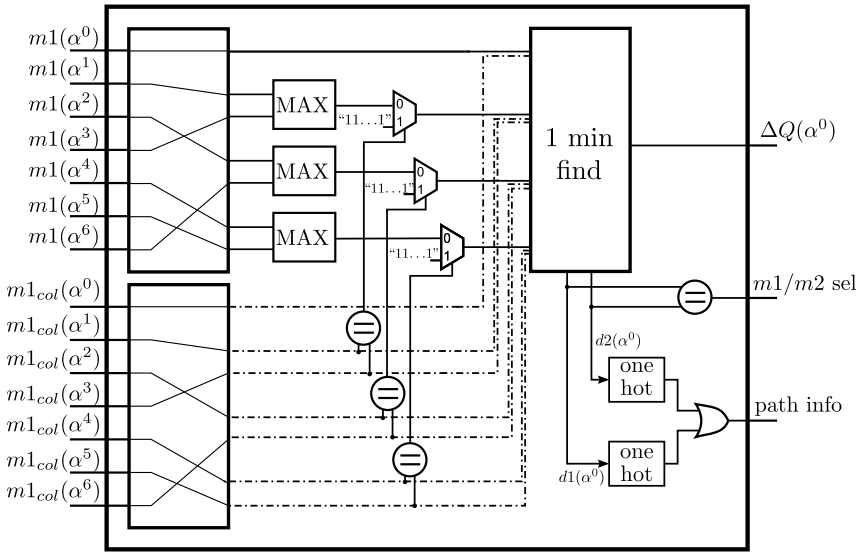


Figure 2.4: Architecture for extra column extraction. Example for generation of message $\Delta Q(\alpha^0)$ over $GF(8)$.

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 5) 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 [17].

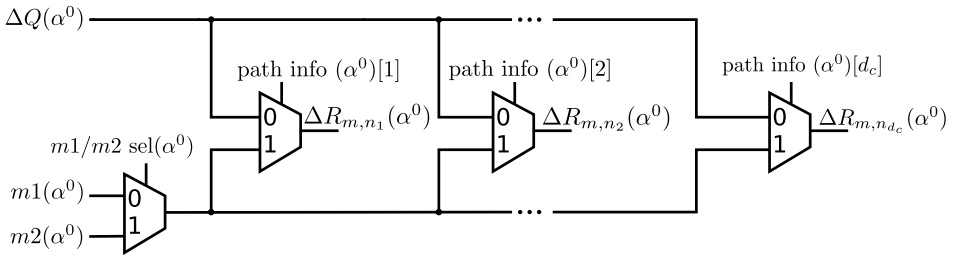


Figure 2.5: Output message generation in delta domain. Example for symbol α^0

In [17] $\Delta R_{m,n}(a)$ 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 2.2. 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. 2.5. 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. 2.5 implements step 5 of Algorithm 5, where each $\Delta R_{m,n}(a)$ 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 (λ) applied to the output messages can be incorporated in the input messages to the multiplexors shown in Fig. 2.5. 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. 2.2), 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.

Algorithm 6: Layered schedule for proposed decoder

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

Inicialization:

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

Main Loop:

while $t \leq MaxIter$ **do**

for $m = 1$ **to** M **do**

1 $Q'^{(t)}_{mn}(a) = Q_n^{(t)}(h_{mn}a) - R_{mn}^{(t-1)}(a)$

2 $Q'^{(t)}_{mn} = \min \left\{Q'^{(t)}_{mn}(a)\right\} \quad \forall\ a \in GF(q)$

3 $Q_{mn}^{(t)}(a) = Q'^{(t)}_{mn}(a) - Q'^{(t)}_{mn}$

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

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

end

6 $t = t + 1$

end

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

2.5 Architecture for the Complete Decoder

In this section the top level design of the NB-LDPC decoder, which includes the CN proposed in Section 2.3, is presented. The proposed decoder has been designed for QC-NB-LDPC codes over $GF(q)$ constructed using the methods included in [36], where $\mathbf{H}$ is formed by $(q-1) \times (q-1)$ circulant sub-matrices that can be composed of zero elements or cyclic shifted identity matrix with non-zero elements from $GF(q)$.

2.5.1 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 6 the layered schedule for the proposed decoder is presented, where the check node processor corresponds to the simplified TMM (Algorithm 5). 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 3 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 [39] 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 $\mathbf{H}$ element h_{mn} to obtain $Q_n(h_{mn}a)$. The permuted VN messages and the last iteration check node outgoing messages $R_{mn}^{(t-1)}(a)$ are processed to obtain $Q'_{mn}(a)$ which 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.

Step 5 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 3 of Algorithm 6. In this step, inverse permutation of $Q_n(a)$ messages have to be done before processing of a new row of $\mathbf{H}$. The decoding process stops when the maximum number of iterations is reached, then the output codeword $\tilde{\mathbf{c}}$ is formed by the most reliable symbols associated to the VN messages $Q_n(a)$.

2.5.2 Decoder Architecture

The block diagram of the complete architecture for the proposed decoder is shown in Fig. 2.6, 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 [36], 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 $\mathbf{H}$.

Blocks $\mathbf{P}$ and $\mathbf{P^{-1}}$ in Fig. 2.6 perform direct and inverse permutation of messages $Q_n(a)$ as can be seen in steps 1 and 3 of Algorithm 6, respectively. The permutations are implemented using multiplexor networks as the ones presented in Fig. 2.7 for the block $\mathbf{P}$ over $GF(8)$. For the block $\mathbf{P^{-1}}$ the only differences are the connections between multiplexors. Each network requires $(q-1) \cdot \log_2(q)$ multiplexors of w bits. For the entire decoder $2 \cdot d_c$ networks are required to implement the blocks $\mathbf{P}$ and $\mathbf{P^{-1}}$.

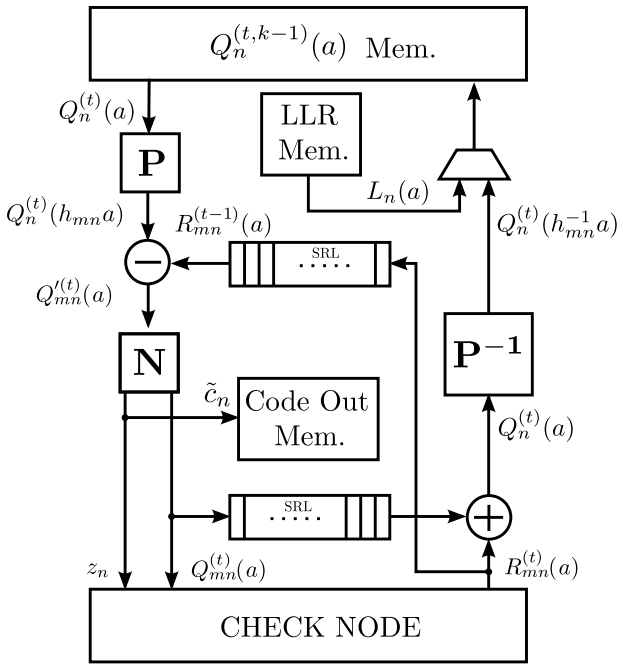


Figure 2.6: Top level decoder architecture based on the horizontal layered schedule

Block **N** in Fig. 2.6 implements the normalization included in steps 2 and 3 of Algorithm 6. This block includes a one minimum finder which searches the most reliable value to derive $Q'_{mn}(a)$ as explained before. In our approach, we take advantage of the one minimum finder to obtain the most reliable symbol of each $Q'_{mn}(a)$ message using the position to recover the minimum. The recovered symbol is used as input for the check node (z_n). On the other hand, the same symbol corresponds to the estimated hard decision symbol $\tilde{c}_n$ at the end of the decoding process.

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.

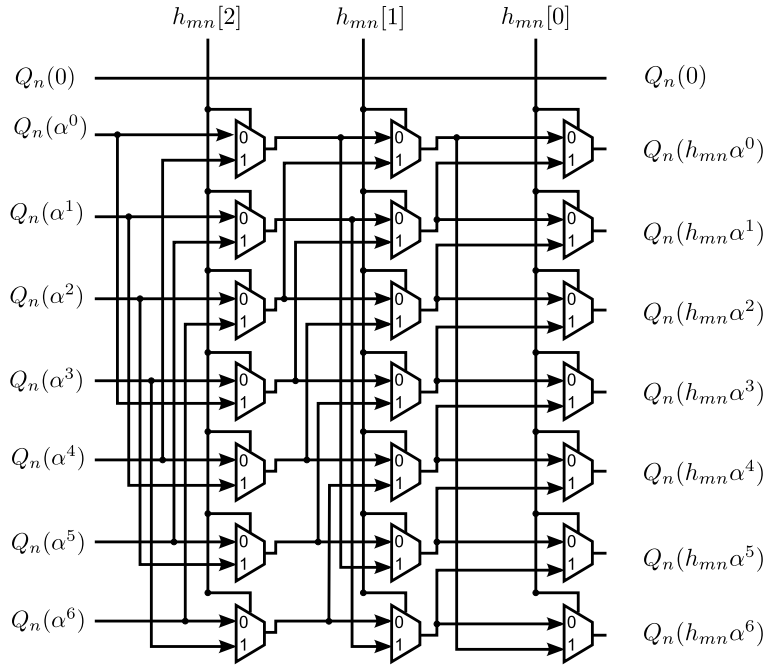


Figure 2.7: Permutation network implemented for $GF(8)$

LLR of the received sequence is initially stored in "LLR Mem." memories (Fig. 2.6) and then extracted to be loaded on VN memories ($Q_n(a)$) when the decoding process starts. d_c memories are required with $q-1$ positions and $q \cdot w$ bits each of them. To store the output codeword ($\tilde{c}_n$), d_c memories are also included, each of them with $q-1$ positions of p bits each one. On addition to these memories, parity check matrix nonzero coefficients h_{mn} need to be stored. Due to the structure of $\mathbf{H}$, only the coefficients of the first row of each circulant sub-matrix need to be saved. For doing this d_c small memories with d_v elements of p bits are added.

2.5.3 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. 2.6. This last process requires that the permutation block $\mathbf{P}$ 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 $\mathbf{H}$, 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 $\mathbf{H}$ can be processed and seg additional clock cycles are required. The process continues until all the M rows of $\mathbf{H}$ are processed. In total, one decoding iteration spends $(M/(q-1)) * ((q-1) + seg) = M + seg \cdot d_v$ 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 (2.2) where $q-1$ clock cycles are added for loading the channel information for a new decoding process and the output codeword is estimated.

$$Throughput = \frac{f_{clk}[\text{MHz}] \cdot N \cdot p}{MaxIter \cdot (M + d_v \cdot seg) + (q-1)} \left[\frac{\text{Mb}}{\text{s}}\right] \tag{2.2}$$

2.5.4 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 $GF(32)$, with parity check matrix $\mathbf{H}$ and parameters $d_c = 27$ and $d_v = 4$. The check node processor is based on simplified TMM, thus the CN design is entirely combinational logic and has an equivalent area of 135K NAND gates, using $w = 6$ bits for the datapath. Additionally, 10 pipeline stages have been used in the decoder to increase the maximum frequency of the decoder requiring 31K registers. The registering points have been selected to balance the critical path, to this ends the segmentation points are: 1) at the output of the P block (Fig.2.6); 2) inside the N block (Fig.2.6) that normalizes the incoming messages at the check node; 3) at the input of the check node processor (Fig.2.6); 4) at the output of the normal domain to delta domain converter (Fig. 2.3); 5) in the third stage of the two minimum finder (Fig.2.3); 6) at the output of the two minimum finder (Fig.2.3); 7) in the second stage of the one minimum finders (Fig.2.4); 8) at the output of the extra column reliability values calculation (Fig.2.4); 9) at the output of the check node processor (Fig.2.6); 10) at the output of the P^{-1} block (Fig.2.6).

Outside the check node, the permutation networks $\mathbf{P}$ and $\mathbf{P^{-1}}$ need 76K NAND gates, and the normalization blocks ($\mathbf{N}$) uses 58.2K NAND gates. The logic resources of the decoder implementation are summarized in Table 2.1. 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^oC and 1.2V.

Table 2.1: Complexity analysis for the proposed decoder. For the (N=837,K=726) NB-LDPC code over $GF(32)$

	Logic Gates (NAND)	Memory bits
Check Node	222K	31K
Permutations ($\mathbf{P}$ and $\mathbf{P^{-1}}$)	76K	-
Normalization ($\mathbf{N}$)	58.2K	-
Add/Sub	46.7K	-
$Q_n(a)$	-	160.7K
$R_{mn}^{(t-1)}(a)$	-	535.7K
LLR mem	-	133.9K
Code Out mem	-	4.1K
$Q_{mn}(a)$	-	51.8K
Total	**402.9K**	**917.2K**

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

Since one iteration of the decoding algorithm takes $M + d_v \cdot seg$ clock cycles and considering 10 pipeline stages, 164 clock cycles per iteration are needed. On the other hand, to achieve the same performance as the approach proposed in [34], as shown in Fig. 2.2, 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 (2.2), which is very much higher than all existing solutions from the SoA, as shown in the next section, except from the T-EMS [16, 17].

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.

2.6 Comparisons With Other NB-LDPC Decoders

The proposed decoder has been compared to the most efficient NB-LDPC decoder designs. Table 2.2 summarizes the results of different architectures found in literature. Results included in Table 2.2 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 [32, 34, 30]. However, the number of equivalent NAND gates that we obtained based on the layout area after the place and route report

Table 2.2: Comparison of the proposed NB-LDPC layered decoder with other works from literature. For the NB-LDPC code (837,726) over $GF(32)$

Algorithm	Simplify-MS [32]	Trellis Max-log QSPA [30]	Min-Max [33]	RMM [34]	T-EMS [17]	Simplified TMM [This Proposal]
Report	Synthesis	Post-layout	Synthesis	Synthesis	Synthesis	Post-layout
Technology	180 nm	90 nm	130 nm	180 nm	90 nm	90 nm
Quantization (w)	5 bits	7 bits	5 bits	5 bits	6 bits	6 bits
NAND Count	1.29M	8.51M	2.1M	871K	2.75M	1.78M
f_{clk} (MHz)	200	250	500	200	250	238
Iterations	15	5	15	15	12	9
Latency (clock cycles)	12995	4460	28215	12675	2160	1507
Throughput (Mbps)	64	223	64	66	484	660
Throughput (Mbps) 90 nm	149	223	107	154	484	660
Efficiency 90 nm (Mbps/M-gates)	115.5	26.2	50.9	176.8	176	371.3
Area (mm^2)	-	46.18	-	-	-	16.12

($16125908\mu m^2$) 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 $R_{mn}^{(t-1)}(a)$ (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 2.2 are implemented under different CMOS technologies, we scale the technology to show results over a 90 nm CMOS process using first order approximations [41] 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 2.2 are 1.66 and 2.33 for 130nm/90nm and 180nm/90nm scaling respectively. Note that, we compare different algorithms

[1] The equivalent area of a NAND gate is $3.136\mu m^2$.

under the same performance, so each one has a different number of iterations. Efficiency is calculated dividing the normalized throughput (for 90 nm technology) over area ratio (Millions of NAND gates).

Considering that only the approach presented in [30] includes post layout results, only comparisons with [30] can be made for the total decoder area given in mm^2. The TMM outperforms by a factor of four compared to the one presented in [30]. On the other hand, our approach has three times higher throughput and fourteen times more efficiency than the decoder in [30].

Comparing our decoder to the approach presented in [32], 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 [33], 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 [34]. Although the TMM requires twice the area occupied by the decoder presented in [34], 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 [34], which makes it suitable for high speed implementations.

Finally, a Trellis EMS decoder was presented [17], 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 [17] in area (54% less), latency (43% less), throughput (36% higher) and efficiency (more than two times higher), under the same FER performance.

2.7 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.

Chapter 3

One Minimum Only Trellis Decoder for Non-Binary Low-Density Parity-Check Codes

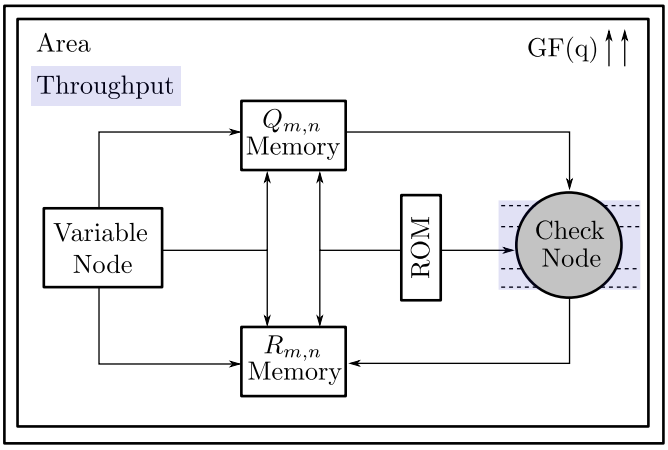


Figure 3.1: Key points of the improvements presented in this chapter.

In this chapter, the paper "One Minimum Only Trellis Decoder for Non-Binary Low-Density Parity-Check Codes" is presented. This research takes as starting point the T-MM algorithm and decoder architecture from the previous chapter and the T-EMS approach in its simplified version using the results from [42]. The research was focused in the reduction of complexity in the CN processor by means of a novel approach to avoid the use of two-minimum finders to compute the

two most reliable messages. Since these blocks are characterized by high-latency networks avoiding them improve both latency and achieved throughput reducing the CN complexity. Fig. 3.1 summarizes the key points that were improved in the derived paper.

Abstract: A one minimum only decoder for Trellis-EMS (OMO T-EMS) and for Trellis-Min-max (OMO T-MM) is proposed in this paper. In this novel approach, we avoid computing the second minimum in messages of the check node processor, and propose efficient estimators to infer the second minimum value. By doing so, we greatly reduce the complexity and at the same time improve latency and throughput of the derived architectures compared to the existing implementations of EMS and Min-max decoders. This solution has been applied to various NB-LDPC codes constructed over different Galois fields and with different degree distributions showing in all cases negligible performance loss compared to the ideal EMS and Min-max algorithms. In addition, two complete decoders for OMO T-EMS and OMO T-MM were implemented for the (837,726) NB-LDPC code over GF(32) for comparison proposals. A 90 nm CMOS process was applied, achieving a throughput of 711 Mbps and 818 Mbps respectively at a clock frequency of 250 MHz, with an area of 19.02mm^2 and 16.10mm^2 after place and route. To the best knowledge of the authors, the proposed decoders have higher throughput and area-time efficiency than any other solution for high-rate NB-LDPC codes with high Galois field order.

Index terms: NB-LDPC, OMO T-EMS, OMO T-MM, Check node processing, low-latency, VLSI design

3.1 Introduction

Since the first non-binary low-density parity-check (NB-LDPC) decoder architecture was proposed for the Q-ary Sum-of-Product algorithm (QSPA) [6], hardware designers have been working to derive solutions that allow the use of NB-LDPC codes in a wide range of communication and storage systems. Good error correction, high throughput and small area remain the challenge of any NB-LDPC decoder designer.

Extended Min-Sum (EMS) [14] and Min-Max (MM) [15] algorithms were proposed, with the aim of reducing the complexity involved in the check node processor, which is the bottleneck of QSPA algorithm. Although the decoding process is simplified by means of using forward-backward for the extraction of check-to-variable messages, these metrics penalize the maximum throughput achievable when they are implemented in hardware.

To avoid the use of forward-backward, in [35] the Trellis Extended Min-Sum (T-EMS) was proposed. With T-EMS, the degree of parallelism is increased using only combinations of the most reliable Galois field (GF) symbols to compute the check-to-variable messages. The decoder presented in [35] was outperformed in [16] where an extra column is added to the original trellis with the purpose of generating the check-to-variable messages in a parallel way. The main drawback of the approach presented in [16] is that requires a lot of area and pipeline stages, reducing the overall efficiency of the decoder. In [17] the hardware implementation of a T-EMS decoder is described, reaching the highest throughput found in literature. Previous trellis-based proposals, such as the ones from [43], [44] and [45], applied partial-parallel decoding as a way to obtain the output messages in the check node processor.

In [34] a decoder architecture named Relaxed Min-Max (RMM) is proposed. RMM makes an approximation for the second minimum calculation and hence, generates the check-to-variable messages with less complexity. The main drawbacks for this approach are: i) the check node output messages are derived serially, reducing the overall throughput of the decoder and increasing latency; and ii) the proposed approach suffers of an early degradation in the error floor region, due to the way of deriving the second minimum.

In this paper, we introduce a novel second minimum approximation based on the statistical analysis of the check node messages named as One Minimum Only (OMO) decoder. The motivation to perform this approximation is that the two-minimum finder duplicates the critical path and increases the complexity of the check node processor. In addition to the second minimum estimator proposed in [34], we analyze two other estimators: one based on a slight modification of the one-minimum finder, and a last one which linearly combines the first two estimators, and showed the best performance in simulations. The proposed OMO decoder can be applied to both T-EMS and Trellis Min-max obtaining OMO T-EMS and OMO T-MM decoders respectively. By avoiding the use of two-minimum finders [40] in the check node, we were able to reduce both area and latency of the check node update without introducing any performance loss compared to the original EMS or Min-max algorithms.

The OMO T-EMS and OMO T-MM check node architectures have been implemented and included in a layered scheduling decoder. A 90nm CMOS process has been employed and a (837,726) NB-LDPC code over GF(32) has been chosen to show the efficiency of our approach for high order fields and high rate codes. The OMO T-EMS and OMO T-MM decoders achieve 100% and 159% higher efficiency (Mbps / Million Gates) compared to the most efficient decoder found in literature [34] respectively, with about 30% less latency and 40% higher throughput than the solution from [17] depending on whether EMS or Min-max version is implemented.

The rest of the paper is organized as follows: in Section 3.2 we introduce the nomenclature and the main concepts of T-EMS algorithm. The proposed approach for the second minimum estimation of T-EMS algorithm, OMO T-EMS, is presented in Section 3.3, including and analysis of performance for different NB-LDPC codes and showing that can be extended to Min-max algorithm without loss of generality. Section 3.4 includes the hardware implementation of the proposed check node and the overall decoder. Moreover, synthesis and post place and route results of the design and comparisons with other architectures are also included. Finally, conclusions are outlined in Section 6.5.

3.2 Trellis - Extended Min-Sum algorithm

NB-LDPC codes are characterized by a sparse parity check matrix $\mathbf{H}$ 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. Decoding algorithms for NB-LDPC codes use iterative message exchange between two types of nodes called check nodes (CN) (M rows of $\mathbf{H}$) and variable nodes (VN) (N columns of $\mathbf{H}$).

Let $\mathcal{N}(m)$ ($\mathcal{M}(n)$) be the set of VN (CN) connected to a CN (VN) m (n). Let $Q_{m,n}(a)$ and $R_{m,n}(a)$ be the edge messages from VN to CN and from CN to VN for each symbol $a \in GF(q)$ respectively. $L_n(a)$ denotes the channel information and $Q_n(a)$ the *a posteriori* information.

Let $\mathbf{c} = c_1, c_2, \cdots, c_N$ and $\mathbf{y} = y_1, y_2, \cdots, y_N$ be the transmitted codeword and received symbol sequence respectively, with $\mathbf{y} = \mathbf{c} + \mathbf{e}$ and $\mathbf{e}$ is the error vector introduced by the communication channel. The log-likelihood ratio (LLR) for each received symbol is obtained as $L_n(a) = \log[P(c_n = z_n|y_n)/P(c_n = a|y_n)]$ ensuring that all values are non-negative where z_n is the symbol associated to the highest reliability.

Trellis Extended Min-Sum (T-EMS) algorithm [35] presents a way of implementing the original EMS algorithm [14], avoiding the use of the forward-backward metrics and increasing the degree of parallelism of the CN. Algorithm 7 includes the original T-EMS CN algorithm, where the first and fifth steps perform the transformation of incoming messages ($\mathbf{Q_{m,n}}$) from "normal" to delta domain (N $\rightarrow \Delta$) and from delta domain to normal domain ($\Delta \rightarrow$ N) for the CN outgoing messages ($\mathbf{R_{m,n}}$) respectively. For the N $\rightarrow \Delta$ transformation, syndrome β of the CN must be obtained (Step 2 of Algorithm 7) using the incoming tentative hard decision z_n for each CN message.

The extra column ($\Delta Q(a)$) calculation is derived on step 3 using the configuration sets originally proposed in [14], with the aim of building the output messages using only the most reliable information. The configuration set $conf(n_r, n_c)$ is defined

Algorithm 7: T-EMS Algorithm

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

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

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

end

2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in GF(q)$

3 $\Delta Q(a) = \min_{\eta'_j(a) \in conf(n_r,n_c)} \sum_{j=1}^{d_c} \Delta Q_{m,n_j}(\eta'_j(a)), a \in GF(q)$

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

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

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

end

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

as the set of at most n_r symbols that satisfy the parity equation, deviating at most n_c times from the combination (path) of symbols with the highest reliability. Considering only the case when $n_r = 1$ and $n_c = 2$, the extra column $\Delta Q(a)$ is built with the combination of the most reliable messages for each GF(q) symbol i.e. with the minimum value message, $min_1(a)$.

Once the $\Delta Q(a)$ values are derived, the outgoing CN messages in delta domain $\mathbf{\Delta R_{m,n}}$, are generated in Algorithm 7 using step 4 which provides all the values for extrinsic CN outgoing messages. For the intrinsic values, $min_1(a)$ and $min_2(a)$ are used as it is explained in detail in [16].

It is important to remark that the $min_1(a)$ values are used for both, $\Delta Q(a)$ and $\Delta R_{m,n}(a)$ generation while $min_2(a)$ is only used to compute $\Delta R_{m,n}(a)$ (in the case of $n_r = 1$ and $n_c = 2$). Additionally, the extraction of the position of the first minimum is also required ($min_{1_{pos}}(a)$), since this information is used to derive the path for each extra column value in the trellis. However, the two minimum values must be processed using a two-minimum finder before the extra column calculation. This two-minimum finder increases the critical path for $min_1(a)$ due to the $min_2(a)$ extraction.

In next section we propose a novel approach to approximate the second minimum, which at the same time that reduces the critical path to get the first minimum, achieves an accurate estimation of the second one without degrading the performance of the original T-EMS and Min-max algorithms.

3.3 One Minimum Only Trellis Decoder

As shown in Section 3.2, the two-minimum finder represents an important part of the CN architecture. On the other hand, the hardware architectures to implement the minimum finder processor [40] introduce the same delay for both $min_1(a)$ and $min_2(a)$, which is not optimal for EMS and Min-max algorithms.

This observation is our principal motivation for creating a novel check node architecture which approximates the computation of the second minimum, reducing the delay for the first one and hence improving the latency and the throughput of the decoder as it can be seen in next sections. Our proposed approach has been tested on multiple NB-LDPC codes with different GF(q) and degree distribution, showing in all cases a negligible performance loss compared with the T-EMS and Min-max algorithms. In order to simplify the description of our proposal, we will focus on T-EMS, however, this method can be directly derived to Min-max algorithm without any loss of generality. However, we will provide performance and implementation results of this new solution for both EMS and Min-max decoders.

In the rest of the section, different estimators of the second minimum values are considered, and a statistical analysis of their distribution compared to the true second minimum is made. The analysis is done for the (837,726) NB-LDPC code over GF(32), where $\mathbf{H}$ is generated using the methods proposed in [36], with circulant sub-matrices of size $(q-1)\times(q-1)$. However, other codes with different GF and degree distribution have been tested obtaining the same behavior.

3.3.1 Estimators for the second minimum value

A first natural solution for the estimation of min_2 is to make use of a scaled version of the first minimum min_1, described in equation (3.1):

$$min_2''(a) = min_1(a) \times \gamma_p \tag{3.1}$$

This approximation has been already proposed in [34]. However, by just applying equation (3.1) the value of the minimum is usually underestimated if we apply a γ_p value that mimics as much as possible the behavior of EMS or Min-max in the waterfall region [1]. As it can be seen in Fig.3.2, where we draw the distributions of the true $min_2(a)$ and their proposed estimators, the value of $min_2''(a)$ is on average smaller than the real $min_2(a)$, which leads to an important performance degradation in the error floor region.

[1] γ_p is selected as the mean value of the ratio between $min_2(a)$ and $min_1(a)$. $\gamma_p = min_2(a)/min_1(a)$

A second possible estimator makes advantage of a re-use of the hardware architecture. Using a radix-2 one-minimum finder is possible to determine an early estimation for the second minimum. In Fig. 3.4, a one-minimum tree finder is presented. In the figure, we include an extra multiplexor in the last stage, that allows extracting the looser term, denoted $min_2'''(a)$. By doing so and just using an extra multiplexor, this term can be used as an early estimator of the second minimum, which represents an upper-bound on the true minimum value. If the true $min_2(a)$ value is located in the other half part of the tree that $min_1(a)$ ($d_c/2$ branches of the minimum tree finder not connected to $min_1(a)$), then we obtain $min_2'''(a) = min_2(a)$. In the other cases, $min_2'''(a) > min_2(a)$. Hence, the resultant value corresponds to an provable upper bound on the true $min_2(a)$. A systematic over-estimation of the second minimum value could lead also to performance degradation of the complete decoder, and we propose to combine $min_2''(a)$ and $min_2'''(a)$ in order to get an estimator with a better statistical behavior.

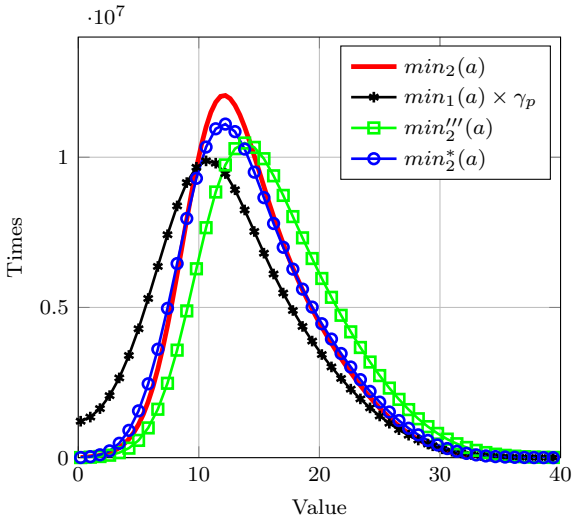


Figure 3.2: Histograms for the different estimators of $min_2(a)$. The γ_p value was set to 1.125.

As we will demonstrate with a statistical analysis in the next section, both $min_2''(a)$ and $min_2'''(a)$ are biased estimators, one over-estimating the true second minimum, and the other one under-estimating the true second minimum. We therefore propose in this paper to combine those two estimators, by using a linear combination of the two preceding estimators, in the following way:

$$min_2^*(a) = \frac{min_2''(a) + min_2'''(a)}{2} = \frac{min_1(a) \times \gamma_p + min_2'''(a)}{2} \tag{3.2}$$

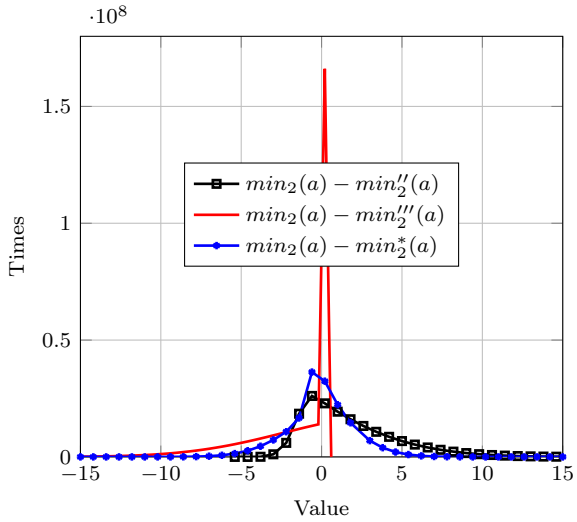


Figure 3.3: Histograms showing the error distribution of different estimators of $min_2(a)$. The γ_p value was set to 1.125.

Compared to the real $min_2(a)$ values, $min_2^*(a)$ presents a similar behavior in the histogram shown in Fig. 1 which implies that the proposed estimation has similar statistical behavior than the exact $min_2(a)$ values.

The operations involved to implement (3.2) can be performed after $min_1(a)$ and $min_2'''(a)$ values are obtained (using the hardware structure in Fig. 3.4). Therefore, the second minimum estimation can be made at the same time that $\Delta Q(a)$ values are obtained, to finally calculate check-to-variable output messages.

In the next section, we analyze the statistical behavior of each of the three proposed estimators.

3.3.2 Statistical analysis of the different estimators

In Fig. 3.3, we plot the distributions of the difference between the proposed estimators, $\widehat{min}_2(a)$ being defined following one of the equations (3.1)-(3.2), and the true minimum, *i.e.* $p\left(min_2(a) - \widehat{min}_2(a)\right)$. We performed this analysis by computing for each iteration and for different E_b/N_0 values the difference between the real second minimum at the check node and each one of the estimators. The information for this analysis is computed based on all the M check nodes of the parity check matrix.

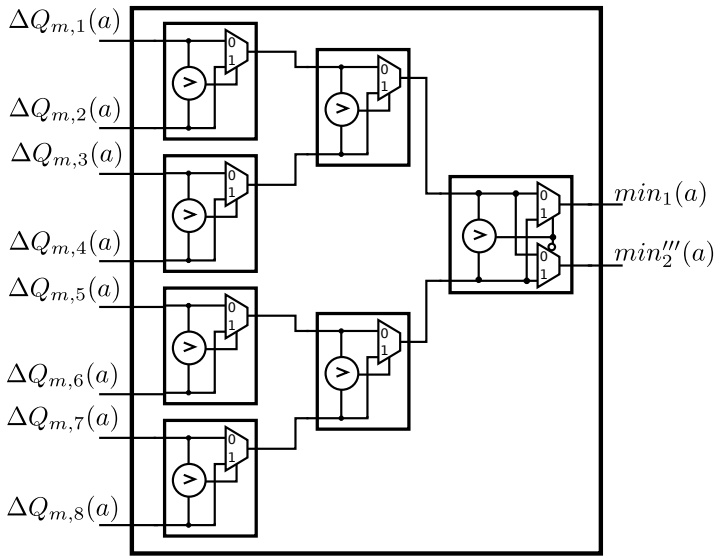


Figure 3.4: Second minimum estimation based on a radix-2 one-minimum finder. Example for an eight inputs tree.

From the shape of the distributions, we can see that $\widehat{min}_2(a) = min_2''(a)$ seems to be biased and skewed to the positive values of the difference, which means that not only $min_2''(a)$ under-estimates the true minimum, but also that the difference is not symmetric around its bias. Of course, as it was expected for $\widehat{min}_2(a) = min_2'''(a)$, which represents a upper bound on the true second minimum, we get the opposite behavior, as the distribution of $min_2(a) - min_2'''(a)$ is left biased and skewed. In order to better measure the performance of each estimator, we have computed the first four cumulants of the distributions $p\left(min_2(a) - \widehat{min}_2(a)\right)$, and reported their values in Table 3.1 after 1 and 15 decoding iterations. The first cumulant of the distribution is the *mean*, and measures the bias of the estimator, a value of zero indicating that the estimator is unbiased. The second cumulant is the *square-root of the variance*, which indicates the spread of the estimator around the mean value. The third cumulant is the *skewness*, and is a measure of the symmetry of the distributions. A zero skewness indicates that an estimator does not favor positive or negative difference with the true value $min_2(a)$. Finally, the fourth cumulant, called the *kurtosis* is a measure of the flatness of the tails of the distribution. A low value of the kurtosis indicates that very large outliers values of the difference with the true minimum do not appear with high probability. The kurtosis value for a Gaussian distribution is equal to 3.

Table 3.1: Statistical properties of the different $\widehat{min_2}(a)$ estimators after $I = 1$ and $I = 15$ decoding iterations.

	$p\left(min_2 - min_2''\right)$	$p\left(min_2 - min_2'''\right)$	$p\left(min_2 - min_2^*\right)$
mean ($I = 1/I = 15$)	1.70 / 2.09	-1.74 / -1.67	-0.07/0.16
σ ($I = 1/I = 15$)	2.94 / 2.65	2.83 / 2.74	2.04 / 1.92
Skewness ($I = 1/I = 15$)	1.17 / 1.09	-0.11 / -0.25	-0.0011 / -0.0025
Kurtosis ($I = 1/I = 15$)	4.51 / 4.18	5.82 / 5.71	4.05 / 3.99

As we can see from those tables, $min_2''(a)$ typically tends to under-estimates the true $min_2(a)$ value, since both the mean and the skewness are positive, while $min_2'''(a)$ over-estimates the true $min_2(a)$ value, since both the mean and the skewness are negative (which was expected as $min_2'''(a)$ is actually an upper bound of $min_2(a)$). As we can see, the third estimator that we propose, namely $min_2^*(a)$, is a better estimation than the other 2, with respect to all statistics. First it is practically unbiased at the first iteration, although a slight positive bias seems to appear at iteration 15. The skewness is almost zero, which tells us that, on average, the decoder will under-estimate or over-estimate the min_2 with the same frequency. Finally, both the variance and the kurtosis of $p\left(min_2(a) - min_2^*(a)\right)$ are the minimum among the three estimators, which indicates that values very different than the true minimum will appear less often with $min_2^*(a)$ than with the other two estimators. With respect to those indicators, $min_2^*(a)$ is a better estimator of min_2 than $min_2''(a)$ or $min_2'''(a)$. We will confirm in the next section that $min_2^*(a)$ also provides the maximum gain in error correction performance for the overall LDPC decoder.

3.3.3 Frame Error Rate Performance

To prove the correct behavior of the proposed OMO T-EMS and OMO T-MM algorithms, we performed Frame Error Rate (FER) simulations for NB-LDPC codes with different degree distributions and Galois field values, from GF(4) to GF(32), assuming transmission over Binary Phase Shift Keying (BPSK) modulation and Additive White Gaussian Noise (AWGN) channel. In this subsection we only include the performance for two different codes, the (837,726) NB-LDPC code over GF(32) with $d_c = 27$ and $d_v = 4$ and the (2212,1896) NB-LDPC code over GF(4) with $d_c = 28$ and $d_v = 4$. For the rest of the codes we obtained similar results. We compare the proposed OMO T-EMS and OMO T-MM approaches to T-EMS [17] and Relaxed Min-Max (RMM) algorithms [34].

Fig. 3.5 shows the frame error rate (FER) simulation results of the (837,726) NB-LDPC code. For this code, the proposed OMO T-EMS algorithm in its floating point version (fp) achieves the same performance as T-EMS algorithm without any performance loss. Both algorithms use 15 iterations (it) and a scaling factor $\lambda = 0.5$. In addition, the OMO T-MM algorithm has a coding gain of 0.12dB compared to the RMM from [34]. Comparing the quantized version of OMO T-EMS algorithm to RMM algorithm, OMO T-EMS algorithm with 7 bits (7b) for the datapath and 9 iterations achieves the same performance as RMM [34] with 5 bits for the datapath and 15 iterations, so the proposed approach requires less iterations than the method from [34] to achieve the same performance. For the quantized version of the OMO T-MM the performance with 6 bits (6b) and 8 iterations achieves the same than the RMM decoder.

On Fig. 3.6, we have plotted the performance of the T-EMS decoder with 15 iterations, and for all the approximations of the second minimum discussed in this paper. The curve labeled T-EMS uses the exact computed value of min_2. As we can see, the fact that min_2'' and min_2''' do not estimate accurately the second minimum has indeed an impact on the overall decoder performance, and especially in the error floor region, where a strong early flattening appears for both approximations (especially for min_2''). On the other hand, our proposed approximation min_2^* has absolutely no performance loss compared to the T-EMS with the exact minimum computation, both in the waterfall and the error floor regime. It results that the complexity gains provided by the OMO-T-EMS comes at no performance loss, at least for the codes that we simulated.

In Fig. 3.7, simulations for the (2212,1896) NB-LDPC show a negligible performance loss of 0.03dB for a FER $= 10^{-7}$ comparing T-EMS to OMO T-EMS. The same happens when we compare Min-max algorithm to OMO T-MM, just a negligible difference of 0.04dB is introduced by the approximation. The γ_p values in all simulations are adjusted using the mean of the ratio min_2/min_1 as we said before.

3.4 OMO T-EMS and OMO T-MM Hardware Architectures

In this section the hardware architectures for the proposed OMO T-EMS and OMO T-MM are introduced. Since the main contribution of this paper focuses in the CN processing, first we detail the implementation results for the OMO T-EMS and OMO T-MM CN architectures comparing them to other existing solutions. Finally, we present the results for the complete decoders with horizontal layered schedule.

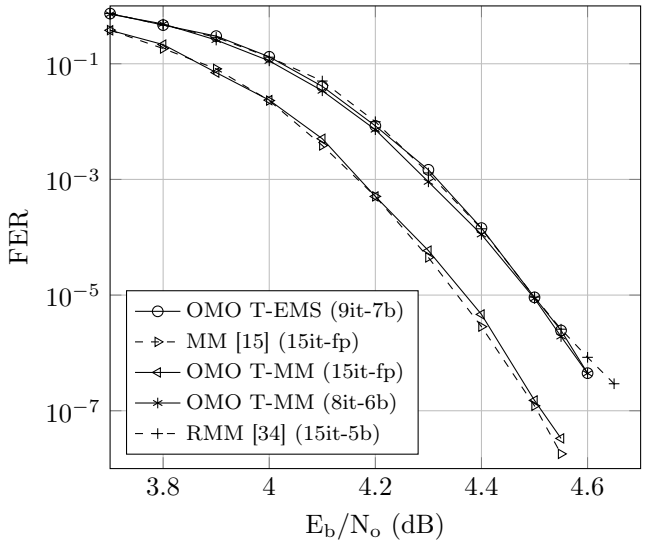


Figure 3.5: FER performance for the (837,726) NB-LDPC code over GF(32) with AWGN channel. Layered schedule is applied to all algorithms. $\lambda = 0.5$ for TEMS and OMO T-EMS algorithms. $\gamma_p = 1.125$ for OMO T-EMS algorithm. $\gamma_p = 1.5$ for OMO T-MM algorithm.

3.4.1 Check Node Architecture

In Fig. 3.8.a the original T-EMS hardware structure [17] is included, while the proposed OMO T-EMS CN structure is presented in Fig. 3.8.b. It can be observed that the main advantage of our approach is to avoid the use of two-minimum finders and apply one-minimum finders, reducing the total complexity and the delay for the $min_1(a)$ values, introducing the novel second-minimum estimation. To do this approximation, the block labeled "$min_2^*(a)$ *Processor*" is responsible to implement the Eq. (3.2). The $min_2^*(a)$ Processor does not introduce any additional delay since the processing is made at the same time that the $\Delta Q(a)$ values are computed. Moreover, it is important to remark that the OMO decoding technique can be directly implemented for a Min-max decoder obtaining the same advantages.

For both OMO T-EMS and OMO T-MM algorithms, the row-wise search of the most reliable messages $min_1(a)$ implies that the one-minimum finder must have d_c inputs and includes an extra multiplexor in the last stage to extract the $min_2'''(a)$ values as shown in Fig. 3.4. For the CN $q-1$ one-minimum finders are required, each one formed by $(d_c - 1)$ w-bit comparators and $(d_c \times w)$ 2-input multiplexors, where w is the number of bits for the datapath. On the other hand, compared with the two-minimum finders [40], the critical path is reduced by half due to the

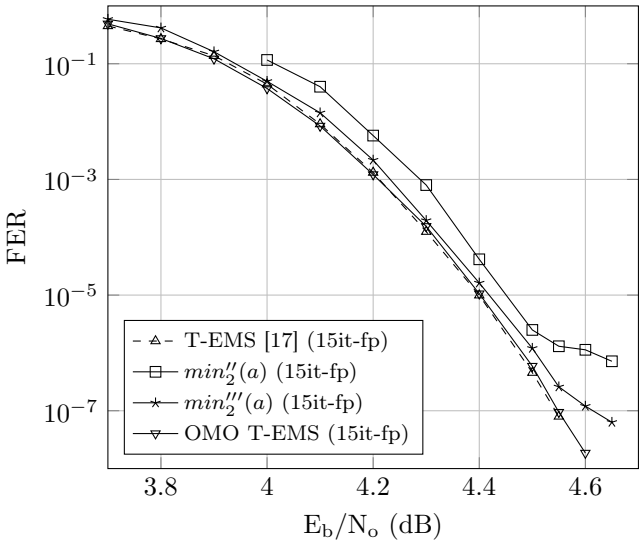


Figure 3.6: FER performance for the (837,726) NB-LDPC code over GF(32) with AWGN channel for the estimators of the second minimum value. $\gamma_p = 1.125$ for OMO T-EMS algorithm.

reduction of the hardware spent on the second minimum computation, which will impact greatly on the obtained throughput.

To make a fair comparison with the two-minimum finder used in conventional designs, we must add extra resources to implement (3.2), which reduce to $2\times(q-1)$ w-bit adders for the one-minimum finders. This value is calculated considering that the implementation of (3.1) and (3.2) need only two additional adders.

On the other hand, a conventional two-minimum finder [40] requires $2 \times d_c$ w-bit comparators and $3 \times d_c \times w$ 2-input multiplexors. Considering the same number of equivalent gates for an adder and a comparator (w bits both of them), the two-minimum finder has three times more multiplexors and two times more comparators than the one minimum finder plus the second minimum estimation implementing (3.2).

For the $N \rightarrow \Delta$ and $\Delta \rightarrow N$ transformation, the approach used is similar to the one proposed in [39], requiring $2 \times q \times p \times d_c \times w$ 2-input multiplexors to perform both transformations. The check node's syndrome β is calculated adding all d_c tentative hard decision symbols z_n by means of a GF(q) adder in a tree structure fashion needing $p \times (dc - 1)$ XOR gates.

The extra column values are generated using a configuration processor similar to the one proposed in [17] using $(q - 1) \times (q/2 - 1)$ w-bit adders or comparators to

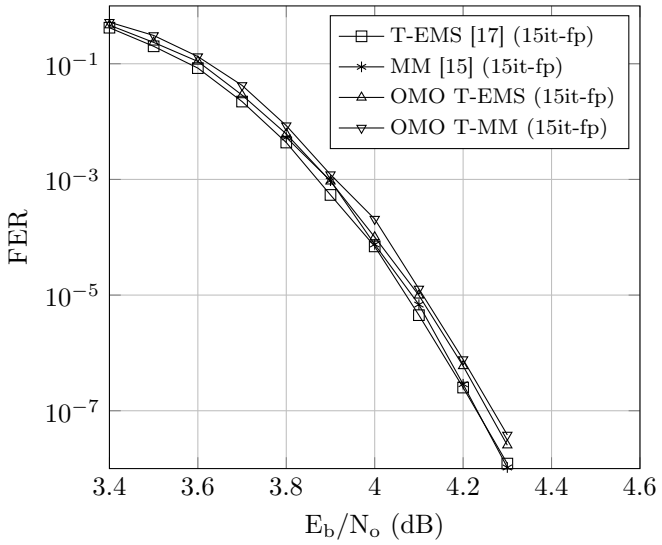


Figure 3.7: FER performance for the (2212,1896) NB-LDPC code over $GF(4)$ with AWGN channel. Layered schedule is applied to all algorithms. $\lambda = 0.5$ for T-EMS and OMO T-EMS algorithms. $\gamma_p = 2.5$ for OMO T-EMS algorithm. $\lambda = 0.75$ for MM and OMO T-MM algorithms. $\gamma_p = 1.125$ for OMO T-MM algorithm.

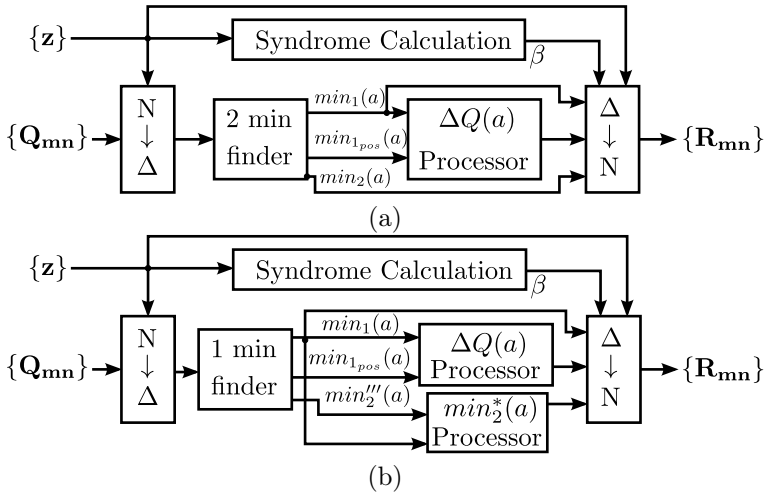


Figure 3.8: Check node top architecture for T-EMS algorithm (a). Proposed OMO T-EMS/ OMO T-MM check node architecture (b).

Table 3.2: CN complexity comparisons. For the (837,726) NB-LDPC code over GF(32)

Architecture	Datapath	Logic Gates (NAND)	Memory (bits)
Relaxed Min-Max[34]	5 bits	152594	52080
T-EMS ($n_r = 2$, $n_c = 2$) [17]	6 bits	304260	-
OMO T-EMS ($n_r = 1$, $n_c = 2$)	7 bits	190780	-
OMO T-MM ($n_r = 1$, $n_c = 2$)	6 bits	165700	-

generate the tentative extra column values depending on whether EMS or Min-max check node is implemented. To select the most reliable value, $q-1$ one minimum finders are required, each one formed by $(q/2-1)$ w-bit comparators and $(q/2 \times w)$ 2-input multiplexors. To compute the path info, $(q-1)(2 \times \lceil \log d_c \rceil + (q/2-1) \times w)$ 2-input multiplexors, $(q-1)(\lceil \log d_c \rceil)$ XOR gates and $(q-1)(\lceil \log d_c \rceil - 1)$ OR gates are implemented.

The resources required for the CN implementation of OMO T-EMS and OMO T-MM are summarized in Table 3.2 and compared with the approaches from [34] and [17]. VHDL was used for the description of the hardware and the total gate account was derived after synthesis using Cadence RTL Compiler. The hardware implementation was performed for the (837,726) NB-LDPC code over $GF(32)$, with $d_c = 27$ and $d_v = 4$.

As it can be observed, although the CN in [34] has less NAND gates than our proposals, their CN requires to store intermediate messages due to the serial processing, increasing the gate account of the CN to 230714 NAND gates (considering that storing one bit of RAM memory is equivalent in terms of area to 1.5 NAND gates [32, 34, 30]). Hence, our proposals requires at most 18% less logical resources than the CN presented in [34], even considering that we use two extra bits.

For T-EMS decoder presented in [17] we did not provide separate results for the CN architecture, however we obtained these results considering the main differences with our new proposal. The CN from [17] needs about four times more hardware than our OMO approaches for the extra column values computation due to the use of the first and second minimum for the extraction of the extra column values. As we can see in Table 3.2 OMO T-EMS and OMO T-MM require 37% less logical resources than [17].

3.4.2 Complete decoder architecture

The proposed CN architecture has been included in an horizontal layered schedule decoder (with one CN cell (Fig. 3.8) and d_c VN units. Each VN processor includes dual-port memories that store the LLR values ($Q_n(a)$) and avoid adding extra latency. On the other hand, due to the layered schedule, a shift register is required to store the "last iteration" CN output information ($R_{m,n}(a)$).

Since, only one CN cell is implemented in the decoder, M clock cycles are required to complete one decoding iteration. This value is increased due to the pipeline stages (k) introduced in the decoder ($k \times d_v$ clock cycles are added) with the aim of achieving the desired clock frequency (f_{clk}). As after processing one entire circulant sub-matrix the pipeline registers must be empty before processing a new one, reducing the logical path of the decoder has a great impact in the maximum throughput achieved by the decoder (Eq. (3.3)). Finally, $q-1$ additional clock cycles are required to load the LLR values and output the estimated codeword of the decoder.

$$Throughput = \frac{f_{clk}[\text{MHz}] \times N \times p}{it \times (M + d_v \times k) + (q-1)} \left[\frac{\text{Mb}}{\text{s}}\right] \tag{3.3}$$

With OMO T-EMS we reduce the critical path of the CN, so we only require 8 pipeline stages to achieve a clock frequency of $f_{clk} = 250MHz$ after place and route with Cadence SOC encounter tools and employing a 90 nm CMOS library in which the area of a NAND gate is $3.13\mu m^2$. The total area of the decoder is 19.02 mm^2 with a core occupation of 60% and a gate account of $(19.02 \times 0.6)/3.13 = 3.6$ Million of NAND gates.

For OMO T-MM the number of pipeline stages is 8 and the maximum clock frequency is $f_{clk} = 250MHz$. The total area is 16.10 mm^2 with a core occupation of 70% and a gate account of $(16.1 \times 0.7)/3.13 = 3.6$ Million NAND gates.

It is important to remark that the library used to implement both OMO T-EMS and OMO T-MM do not include optimized RAM memories, so each bit of RAM is implemented as a register, and hence, the area for the memories is about three times larger. Due to this, the total number of equivalent NAND gates is overestimated compared to the results found in literature that always assume optimized memories. For this reason we include in Table 3.3, for comparison purposes, the equivalent number of NAND gates assuming that each bit of RAM is implemented with and area of 1.5 NAND gates.

To achieve the same performance as in [34] and [17], our OMO T-EMS approach requires only 9 decoding iterations, as can be seen in Fig. 3.5, therefore the total latency of the decoder is 1435 clock cycles, which corresponds to a throughput of 729 Mbps (3.3). For the OMO T-MM 1279 clock cycles are required to get the

same FER performance as RMM or T-EMS, so a maximum throughput of 818 Mbps is reached.

OMO T-EMS and OMO T-MM decoders have been compared to the most efficient NB-LDPC decoder designs to the best knowledge of the authors. The results of the comparisons have been included in Table 3.3, where we have scaled the results in [34] to include all throughput results over 90 nm CMOS process [41].

The throughput of both OMO T-EMS and OMO T-MM decoders is higher than any decoder proposed in literature for high order fields and high rate NB-LDPC codes (see Table 3.3), because of the improvements made at the check node processor.

Our approaches require in the worst case (OMO T-EMS) less than half area than [30] and achieve at least 3.2 times higher throughput, so our most complex solution is 13 times more efficient. [2]

On the other hand, the decoder presented in [34] has been considered since it was the most efficient one until now and it uses (3.1) as a method to approximate the second minimum, which gives benefits in terms of area but introduces early performance degradation (Fig. 3.5). Despite this, OMO T-EMS has 8.8 times less latency than [34] achieving 4.75 times higher throughput with a decoder 49.7% more efficient in terms of area over throughput (for a 90 nm CMOS process). On the other hand, OMO TMM has 61.2% higher efficiency than [34] with 9.9 times less latency and 5.3 times higher throughput.

Finally, our OMO T-EMS approach has been compared against the T-EMS decoder presented in [17]. Making use of the novel approach for the second minimum estimation, the latency is reduced on 33% with respect to [17] with an increment in throughput of 50%. The area was also reduced in 25%, so the efficiency is 50% higher.

Is important to remark that the proposed approach is focused on high-rate NB-LDPC codes. However, efficient NB-LDPC decoders suitable for lower rate codes have been proposed in the literature [23, 46]. These architectures make a parallel processing of messages.

[2] Note that [30] is the only proposal that also provides post place and route results.

Table 3.3: Comparison of the proposed NB-LDPC layered decoders to other works from literature. For the (837,726) NB-LDPC code over GF(32)

Algorithm	T-Max-log QSPA [30]	RMM [34]	T-EMS [17]	OMO T-EMS / OMO T-MM
Report	Post-layout	Synthesis	Synthesis	Post-layout
Technology	90 nm	180 nm	90 nm	90 nm
Quantization (w)	7 bits	5 bits	6 bits	7 bits / 6 bits
Gate Count (NAND)	8.51M	871K	2.75M	2.07M / 1.79M
f_{clk} (MHz)	250	200	250	250
Iterations	5	15	12	8
Latency (clock cycles)	4460	12675	2160	1435 / 1279
Throughput (Mbps)	223	66	484	729 / 818
Throughput (Mbps) 90 nm	223	154	484	729 / 818
Efficiency 90 nm (Mbps/M-gates)	26.2	176.8	176	352 / 456
Area (mm^2)	46.18	-	-	19.02 / 16.10

3.5 Conclusions

In this paper a new method to estimate the second minimum value in message of the check node processor of NB-LDPC decoders is proposed. This solution avoids the use of two-minimum finders, greatly reducing the check node complexity. The simplifications applied to the T-EMS and T-MM algorithms reduce latency and area with respect to the original proposal, without introducing any significant performance loss. The proposed check node architecture was included in a complete decoder with layered schedule achieving 729 Mbps of throughput after place and route on a 90nm CMOS process for OMO T-EMS and 818 Mbps for OMO T-MM. The designed decoder nearly doubles the efficiency of the best solutions found in literature for high order fields and high rate codes.

Chapter 4

Reduction of complexity for Non-binary LDPC decoders with compressed messages

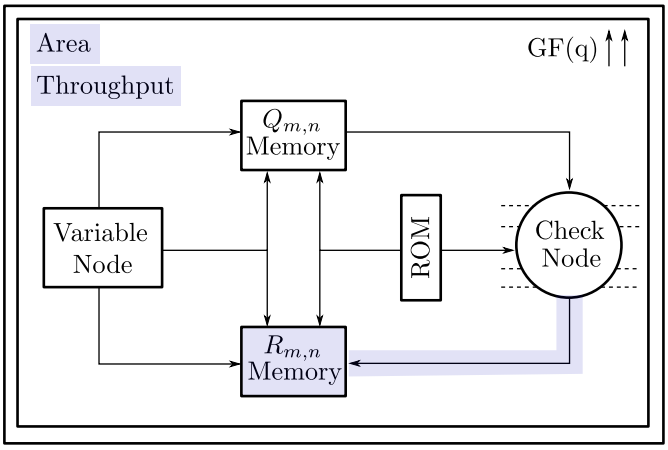


Figure 4.1: Key points of the improvements presented in this chapter.

In this chapter, the paper "Reduction of complexity for Non-binary LDPC decoders with compressed messages" is presented. This research is focused in the reduction of the number of messages exchanged between CN and VN processors. In addition, the memory used to store the information from the last iteration was greatly reduced compared to previous proposals from literature. The improvements shown in this paper allow a great area reduction and an increase in the

achieved throughput. Fig. 4.1 summarizes the key points that were improved in the derived paper.

Abstract: In this paper a method to compress the messages between the check nodes and the variable nodes is proposed. This method is named as compressed non-binary message-passing (CNBMP). The CNBMP reduces the number of messages exchanged between one check node and the connected variable nodes from $d_c \times q$ to $5 \times q$, and its application has a high impact in the performance of the decoder: the storage and routing area is reduced and the throughput is increased. Unlike other methods, the CNBMP does not introduce any approximation or modification in the information and the processed operations are exactly the same as the original decoders, hence, no performance degradation is introduced. To demonstrate its advantages, an architecture applying this CNBMP to the Trellis Min-max algorithm was derived showing that most of the storage resources were also reduced from $d_c \times q$ to $5 \times q$. This architecture was implemented for a (837,726) NB-LDPC code using a 90nm CMOS technology reaching a throughput of 981Mbps with an area of 10.67mm^2, which is 3.9 more efficient than the best solution found in literature.

Index terms: LDPC codes, decoding, non-binary, hardware implementation, high-throughput

4.1 Introduction

The two main bottlenecks of non-binary low-density parity-check (NB-LDPC) decoder architectures are the storage resources and the maximum throughput. Regardless their significant benefits, such as a better behaviour in the error floor region and a more robust correction for burst errors, NB-LDPC codes cannot compete with their binary counterparts in terms of complexity or throughput/area efficiency.

Several alternatives to the original Q-ary Sum-of-Product algorithm (QSPA) [6] were proposed during this last decade in order to keep the best correction performance possible and reduce complexity. The most remarkable ones are the Extended Min-Sum (EMS) [14] and Min-Max (MM) [15] algorithms, which reduced the complexity of the check node processor and the storage resources. However, a parallel implementation of these algorithms was prohibitive in terms of wiring between check node and variable node processors and arithmetic resources. For this reason all the architectures derived from these two algorithms applied the forward-backward metrics, which consist in a serial computation of the check node information. All the decoders based on the forward-backward suffer from a very

large number of clock cycles per iteration, limiting the maximum throughput to a few Mbps [39].

In order to increase the degree of parallelism keeping the same error correction, a new version of the EMS algorithm named as Trellis-EMS (T-EMS) was proposed in [16]. This method allowed hardware designers to implement a fully parallel check node in a layered architecture [17]. This implementation did not sacrifice efficiency in terms of throughput/area compared to other serial implementations based on trellis [34] and increased throughput more than three times. Further improvements were introduced with the Trellis Min-max (TMM) in [42]. Despite this, the decoder from [42] required $14.7mm^2$ of area with a 90nm CMOS process and reached a throughput of 660Mbps, which is far from the results of modern binary LDPC decoders for the same technology ($9.6mm^2$, 45.42Gbps) [47]. While the binary architectures just exchange a number of messages equal to the degree of the check node (d_c) between check node and variable node, non-binary decoders require q times more wires/connections; and the same happens for the memories and registers, which are about the 80% of the decoder's area.

In this brief a method to reduce the number of messages exchanged in non-binary decoders between the check node and the variable node is introduced. This method does not vary the computation of the decoding algorithm nor reduces the information transferred between nodes, so it does not introduce any performance degradation. This proposal compresses the information transmitted in the message passing reducing the size of the messages from $d_c \times q$ to $5 \times q$. This has a great impact in both area and throughput specially for high rate codes. As an example, an implementation for the same code as in [42] and [34] achieves 981Mbps of throughput with an area of $10.6mm^2$ for a 90nm CMOS process.

The rest of the paper has four sections. Section II includes a summary of the NB-LDPC message-passing of the decoding algorithms. Section III describes the proposal of this work. Section IV shows the impact of the new message-passing in a hardware implementation and compares the results to other existing architectures. Section V outlines the conclusions.

4.2 Non-binary LDPC message passing

Let $\mathbf{H}$ be the $M \times N$ parity check matrix with coefficients $h_{i,j} \in GF(q)$ that defines an (N,K) NB-LDPC code. $\mathcal{N}(m)$ and $\mathcal{M}(n)$ are described as the sets that consist of all the non-zero elements of a row m (check node) and a column n (variable node) respectively. The size of the sets $\mathcal{N}(m)$ and $\mathcal{M}(n)$ are the degree of check node (d_c) and the degree of variable node (d_v). The d_c and d_v degrees represent the number of messages that each check node and variable node receive respectively. The set of messages from check node to variable node are denoted as $\mathbf{R}$ and the

set of messages from variable node to check node are $\mathbf{Q}$. Each of these messages consists of q elements, due to the fact of performing operations over GF(q). The method to compute each of these sets depends on the decoding algorithm applied. The algorithms that provide a better performance with lower complexity are T-EMS and T-MM, which have a different processing at the check node but share the same operations at the variable node. To a better understanding of the message-passing between check node and variable node, a short explanation of the basics operations performed in the check node is included next, for more details about the different decoding processes we refer to [16] and [42].

In addition, to perform a parallel processing of the check node we will assume delta domain [16], [17] messages as inputs and outputs at the check node.

Let $\triangle\mathbf{Q}$ be the set of d_c messages from the variable node in delta domain defined as:

$$\triangle\mathbf{Q} = \{\triangle\mathbf{Q}_{m,n}\} \ , \ n \in \mathcal{N}(m) \ , \ m \in \mathcal{M} \tag{4.1}$$

Each element $\triangle\mathbf{Q}_{m,n}$ includes the likelihood of being the symbol $\alpha^x \in GF(q)$, $x = \{-\infty, 0, 1, \ldots, q-2\}$:

$$\triangle\mathbf{Q}_{m,n} = \{\mathcal{Q}_{m,n}(\alpha^{-\infty}), \mathcal{Q}_{m,n}(\alpha^0), \ldots, \mathcal{Q}_{m,n}(\alpha^{q-2})\} \tag{4.2}$$

The output messages of the check node in the delta domain are also of length d_c:

$$\triangle\mathbf{R} = \{\triangle\mathbf{R}_{m,n}\} \ , \ n \in \mathcal{N}(m) \ , \ m \in \mathcal{M} \tag{4.3}$$

The likelihood of each symbol to accomplish the parity check equation of the check node is defined as:

$$\triangle\mathbf{R}_{m,n} = \{\triangle\mathcal{R}_{m,n}(\alpha^{-\infty}), \triangle\mathcal{R}_{m,n}(\alpha^0), \ldots, \triangle\mathcal{R}_{m,n}(\alpha^{q-2})\} \tag{4.4}$$

To compute the reliability of each one of the q symbols in a single message, the check node update equations consider the combinations of the most reliable input messages. If only the two most reliable messages per symbol are considered the update rules for the check node follow the next conditions:

i) If the input likelihood of the symbol α^x for the edge $\{m,n\}$ is not the most reliable for α^x nor is considered to compute other α^y output message, $\triangle\mathcal{R}_{m,n}(\alpha^x)$ is equal to the most reliable value $\mathcal{Q}_{m,n_0}(\alpha^x)$:

$$\begin{aligned}\triangle\mathcal{R}_{m,n}(\alpha^x) &= \{min(\mathcal{Q}_{m,n_0}(\alpha^x), \mathcal{Q}_{m,n_0}(\alpha^y) + \mathcal{Q}_{m,n_0}(\alpha^z))\},\\ &\alpha^y + \alpha^z = \alpha^x, \forall\alpha^y, \alpha^z \in GF(q)\\ &\qquad \leftrightarrow [\mathcal{Q}_{m,n}(\alpha^x) \neq \mathcal{Q}_{m,n_0}(\alpha^x)] \bigwedge [\mathcal{Q}_{m,n_0}(\alpha^x) + \mathcal{Q}_{m,n_0}(\alpha^z) \neq \triangle\mathcal{R}_{m,n}(\alpha^y)],\\ &\alpha^x + \alpha^z = \alpha^y, \forall\alpha^y, \alpha^z \in GF(q)\end{aligned} \tag{4.5}$$

Being $\mathcal{Q}_{m,n_0}(\alpha^x)$ and $\mathcal{Q}_{m,n_1}(\alpha^x)$:

$$\mathcal{Q}_{m,n_0}(\alpha^x) \leq \mathcal{Q}_{m,n_1}(\alpha^x) \leq \mathcal{Q}_{m,n}(\alpha^x)\ ,\ \forall n \in \mathcal{N}(m) \setminus \{n_0, n_1\} \tag{4.6}$$

ii) If the input likelihood of the symbol α^x for the edge $\{m,n\}$ is the most reliable for α^x, $\triangle\mathcal{R}_{m,n}(\alpha^x)$ takes the value of the second more reliable message:

$$\triangle\mathcal{R}_{m,n}(\alpha^x) = \{\mathcal{Q}_{m,n_1}(\alpha^x)\} \leftrightarrow [\mathcal{Q}_{m,n}(\alpha^x) = \mathcal{Q}_{m,n_0}(\alpha^x)] \tag{4.7}$$

iii) If the input likelihood of the symbol α^x for the edge $\{m,n\}$ is involved in the output reliability of α^y, $\triangle\mathcal{R}_{m,n}(\alpha^x)$ takes the value of the most reliable message $\mathcal{Q}_{m,n_0}(\alpha^x)$:

$$\begin{aligned}\triangle\mathcal{R}_{m,n}(\alpha^x) &= \{\mathcal{Q}_{m,n_0}(\alpha^x)\}\\ &\leftrightarrow [\mathcal{Q}_{m,n_0}(\alpha^x) + \mathcal{Q}_{m,n_0}(\alpha^z)\\ &= \triangle\mathcal{R}_{m,n}(\alpha^y)], \alpha^x + \alpha^z = \alpha^y, \forall\alpha^y, \alpha^z \in GF(q)\end{aligned} \tag{4.8}$$

To reduce the number of operations at the check node and share results a set that includes common computation was proposed in [16], and defined as:

$$\mathbf{P}_m = \{\mathbf{P}_m(\alpha^{-\infty}), \mathbf{P}_m(\alpha^0), \ldots, \mathbf{P}_m(\alpha^{q-2})\}\ ,\ m \in \mathcal{M} \tag{4.9}$$

Where each element from the set $\mathbf{P}_m$ includes the two most reliable input values from α^x:

$$\mathbf{P}_m(\alpha^x) = \{\mathcal{P}_{m_0}(\alpha^x) = \mathcal{Q}_{m,n_0}(\alpha^x), \mathcal{P}_{m_1}(\alpha^x) = \mathcal{Q}_{m,n_1}(\alpha^x)\} \tag{4.10}$$

Based on the set $\mathbf{P}_m$ an extra set is computed in [16]. This set includes the values from $\triangle\mathcal{R}_{m,n}(\alpha^x)$ in equation (4.5). The set is defined as follows:

$$\mathbf{E}_m = \{\mathcal{E}_m(\alpha^{-\infty}), \mathcal{E}_m(\alpha^0), \ldots, \mathcal{E}_m(\alpha^{q-2})\}\ ,\ m \in \mathcal{M} \tag{4.11}$$

$$\begin{aligned} &\mathcal{E}_m(\alpha^x) = \{min(\mathcal{Q}_{m,n_0}(\alpha^x), \mathcal{Q}_{m,n_0}(\alpha^y) + \mathcal{Q}_{m,n_0}(\alpha^z))\} \\ &(\alpha^y + \alpha^z = \alpha^x \in GF(q)) \bigwedge (\mathcal{Q}_{m,n_0}(\alpha^y) + \mathcal{Q}_{m,n_0}(\alpha^z) < \mathcal{Q}_{m,n_0}(\alpha^a) + \mathcal{Q}_{m,n_0}(\alpha^b)), \\ &\alpha^a + \alpha^b = \alpha^x, \quad \forall \quad \alpha^a, \alpha^b \in GF(q) \backslash \{\alpha^y, \alpha^z\} \end{aligned} \tag{4.12}$$

Regardless the definition of the extra set the output messages of the check node are $\triangle\mathbf{R}_{m,n}$, which is a set of size $q \times d_c$.

4.3 Compressed Non-Binary Message-Passing (CNBMP)

With the aim of reducing the size of the sets that conform the messages shared between check node and variable node we propose a new ordering of the information. With these new sets the number of information exchanged between check node and variable node is reduced considerably and the set $\triangle\mathbf{R}_{m,n}$ is easily derived at the variable node. We name this method Compressed Non-Binary Message-Passing (CNBMP).

First we define the set $\mathbf{C}_m$ as follows:

$$\mathbf{C}_m = \{\mathbf{C}_m(\alpha^{-\infty}), \mathbf{C}_m(\alpha^0), \ldots, \mathbf{C}_m(\alpha^{q-2})\}\ ,\ m \in \mathcal{M} \tag{4.13}$$

$$\mathbf{C}_m(\alpha^x) = \{\mathbf{N}_{x'}(m)\} \tag{4.14}$$

Each $\mathbf{N}_{x'}(m)$ element contains the index n of the edge $\{m, n\}$ for the symbol α^x in which $\triangle\mathbf{R}_{m,n}$ is not updated following equation (4.5):

$$\begin{aligned} \mathbf{N}_{x'}(m) = \{n_0\} \leftrightarrow [&(\alpha^x \in GF(q)) \bigwedge (\mathcal{Q}_{m,n_0}(\alpha^x) \\ &= \mathbf{E}_m(\alpha^x))] \bigvee [(\alpha^x + \alpha^z = \alpha^y, \forall \alpha^y, \alpha^z \in GF(q)) \\ &\bigwedge (\mathcal{Q}_{m,n_0}(\alpha^x) + \mathcal{Q}_{m,n_0}(\alpha^z) = \mathbf{E}_m(\alpha^y))] \end{aligned} \tag{4.15}$$

Considering that the sets $\mathbf{E}_m$ and $\mathbf{P}_m$ are computed the message $\triangle\mathbf{R}_{m,n}$ can be recovered at the variable node following the next equations:

$$\triangle\mathcal{R}_{m,n}(\alpha^x) = \mathcal{E}_m(\alpha^x)\ ,\ n \in \mathcal{N}(m)\backslash\mathbf{N}_{x'}(m) \tag{4.16}$$

$$\triangle\mathcal{R}_{m,n}(\alpha^x) = \mathcal{P}_{m_1}(\alpha^x) \leftrightarrow \mathcal{P}_{m_0}(\alpha^x) = \mathcal{E}_m(\alpha^x)\ ,\ n \in \mathbf{N}_{x'}(m) \tag{4.17}$$

$$\triangle\mathcal{R}_{m,n}(\alpha^x) = \mathcal{P}_{m_0}(\alpha^x) \leftrightarrow \mathcal{P}_{m_0}(\alpha^x) \neq \mathcal{E}_m(\alpha^x)\ ,\ n \in \mathbf{N}_{x'}(m) \tag{4.18}$$

It is important to remark that: i) whether CNBMP is applied or not the sets $\mathbf{P}_m$ and $\mathbf{E}_m$ are computed because of computational efficiency [16], so we are not adding any extra operation; and ii) it can be demonstrated that the value of the messages $\triangle\mathbf{R}_{m,n}$ are exactly the same applying equations (4.5) to (4.8) or (4.16) to (4.18), so in terms of error correction performance we can claim that CNBMP is equivalent to the original T-EMS or T-MM algorithms as it does not include any approximation.

Note that applying the CNBMP the output information of the check node is conformed by the set $\mathbf{E}_m$ that contains q elements and the sets $\mathbf{C}_m$ and $\mathbf{P}_m$ that contain $2 \times q$ elements each one. So in total the cardinality of the output information is $5 \times q$, unlike previous proposals found in literature.

To sum up, the check node with the CNBMP does not compute equations (4.5) to (4.8), but equations (4.16) to (4.18). In addition, the message passing consists of the sets $\mathbf{C}_m$, $\mathbf{P}_m$ and $\mathbf{E}_m$, not of $\triangle\mathbf{R}_{m,n}$, which is of size $d_c \times q$, as shown in Fig.4.2.

4.4 Hardware impact of CNBMP

The first improvement for the hardware architectures of NB-LDPC decoders is the reduction of the wiring. According to the implementation reports, the maximum frequency of the decoder is not limited by the depth of the logic gates, but for the length of the wiring and the routing congestion. So, if we apply CNBMP, the wires between both check node and variable node processors will be reduced and hence, routing congestion will be mitigated. The reduction is $\lambda = (d_c \times q \times Q_b)/(3 \times q \times Q_b + 2 \times q \times \lceil log_2(d_c))\rceil)$ (Fig.4.3), assuming that the messages at the check node are quantized with Q_b bits and that the set $\mathbf{C}_m$ requires $\lceil log_2(d_c)\rceil$ bits to represent the indexes n. As it is shown next with this reduction of the routing there is an improvement in the maximum frequency.

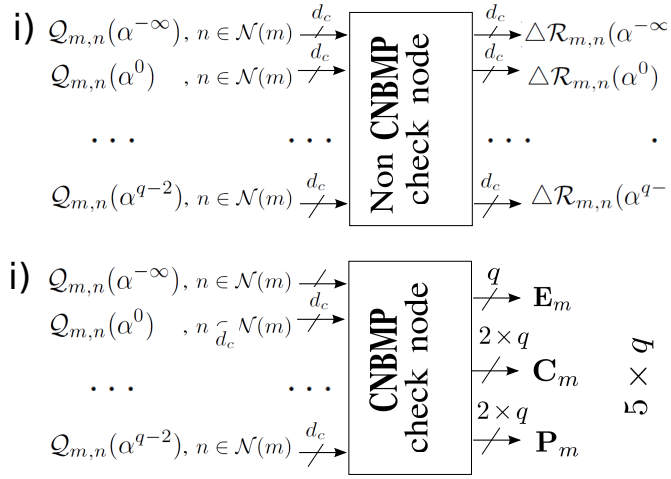


Figure 4.2: i) Check node without CNBMP ii) Check node with CNBMP

Table 4.1: Comparison of the proposed NB-LDPC layered decoder with other works from literature

Algorithm	MS [32]	T-QSPA [30]	MM [33]	MM [34]	T-EMS [17]	T-MM [42]	T-MM CNBMP
Report (nm)	Syn. (180)	Layout (90)	Syn. (130)	Syn. (180)	Syn. (90)	Layout (90)	Syn/Layout (90)
Quantization (Q_b)	5 bits	7 bits	5 bits	5 bits	7 bits	6 bits	6 bits
Gate Count (NAND)	1.29M	8.51M	2.1M	871K	2.75M	3.28M	0.9M / 1.25M
f_{clk} (MHz)	200	250	500	200	250	238	333 / 300
Throughput (Mbps)	64	223	64	66	484	660	1089 / 981
Throughput (Mbps) 90 nm	149	223	107	154	484	660	1089 / 981
Efficiency 90 nm (Mbps / M-gates)	115.5	26.2	50.9	176.8	176	201	1210 / 784.8
Area (mm^2)	-	46.18	-	-	19	14.75	10.4 / 10.6

The second improvement is in terms of storage resources. To perform the layered schedule the decoder requires the storage, in registers or memories, of the information from the check node in the previous iteration, in order to compute the

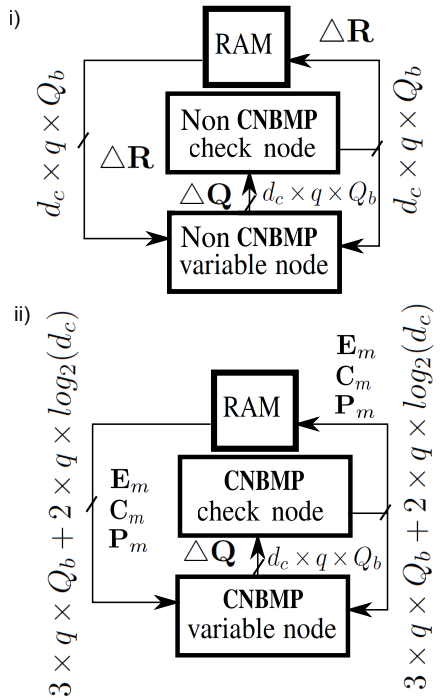


Figure 4.3: i) Layered architecture of a NB-LDPC decoder without CNBMP. RAM memory from this architecture has M addresses of size $d_c \times q \times Q_b$ ii) Layered architecture of a NB-LDPC decoder with CNBMP. RAM memory from this architecture has M addresses of size $3 \times q \times Q_b + 2 \times q \times log_2(d_c)$

extrinsic information. Therefore, M addresses of depth equal to the size of the output messages from the check node are required. As it is previously explained, the number of the output messages without CNBMP is $d_c \times q \times Q_b$ and the number with CNBMP is equal to $3 \times q \times Q_b + 2 \times q \times \lceil log_2(d_c) \rceil$, so the reduction in storage resources is also λ (Fig.4.3). Note that applying CNBMP will be specially advantageous for high rate codes, where d_c is very large. However, even with low and medium rate codes there will be significant improvements, as far as the only requirement to get some complexity reduction is that $d_c > 5$. To de-compress the messages at the variable node comparators and multiplexors implement the conditions from equations (4.16) to (4.18) to select whether $\mathcal{E}_m(\alpha^x)$ or $\mathcal{P}_{m_0}(\alpha^x)$ and $\mathcal{P}_{m_1}(\alpha^x)$ is applied to update $\triangle\mathcal{R}_{m,n}(\alpha^x)$.

In Table 4.1 we include the hardware results of the best architectures for NB-LDPC decoding and the results of our layered T-MM decoder with CNBMP. The code under test is for all the decoders the (N=837,K=726) NB-LDPC code over $GF(32)$,

with $d_c = 27$ and $d_v = 4$ [36]. 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^oC and 1.2V. Compared with a conventional implementation of T-MM algorithm, CNBMP decoder improves the requirements of area due to the reduction of storage resources in the check-node, in a layered schedule. On the other hand, the clock frequency is increased owing to the reduction of the wiring congestion and the core area in general. Additionally, we eliminate some pipeline stages in the decoder thanks to the reduction in the complexity of the check-node processor and hence the critical path is also reduced. These facts contribute to increment the overall throughput of the decoder.

If we compare this work to the most efficient architectures found in literature [34] and [42], we can see that the maximum frequency is increased in 50% and 26% respectively due to the reduction of the routing congestion. On the other hand, area is about 43% larger than the decoder from [34] and 3 times smaller than the one in [42]. After applying the CNBMP the area of storage resources (RAM memories and registers) is reduce from 80% (2.2×10^6 NAND gates) of the total area in [**?**] to 50% (0.62×10^6 NAND gates). About the throughput, the CNBMP proposal is 1.48 times faster than the T-MM decoder in [42] and 14.8 times faster than the Min-max from [34]. In terms of efficiency Throughput/Area the decoder with CNBMP is 3.9 times more efficient than [34] and [42]. For the gate count, we consider the equivalence of one bit of RAM equals to 1.5 NAND gates and one register equals to 4.5 NAND gates.

Finally, if we compare CNBMP to the binary LDPC decoder from [47], which has a gate count of 3.4 millions of equivalent NAND gates and a throughput of 45.42Gbps for a code with a similar rate and half codeword length in terms of bits ((2048, 1723) LDPC code), CNBMP has 2.72 times less gates and reaches 17.46 times less throughput[1]. So, in terms of Throughput/Area efficiency, our non-binary decoder is 6.32 times less efficient than the binary one. Even not reaching the efficiency of a binary decoder, with CNBMP we reduce the difference to less than q, which is a good step forward compared to solutions like the one in [8] that has $2 \times q$ times lower efficiency..

4.5 Conclusions

In this paper a new message-passing definition is proposed for NB-LDPC decoders. This method reduces the number of the messages exchanged between check node and variable node, simplifying the routing of the derived hardware architectures and saving a big percentage of storage resources. Moreover, the new message passing does not modify the processing of the information at the decoder, keeping the same error correction performance as the original message-passing.

Chapter 5

A 630 Mbps Non-Binary LDPC Decoder for FPGA

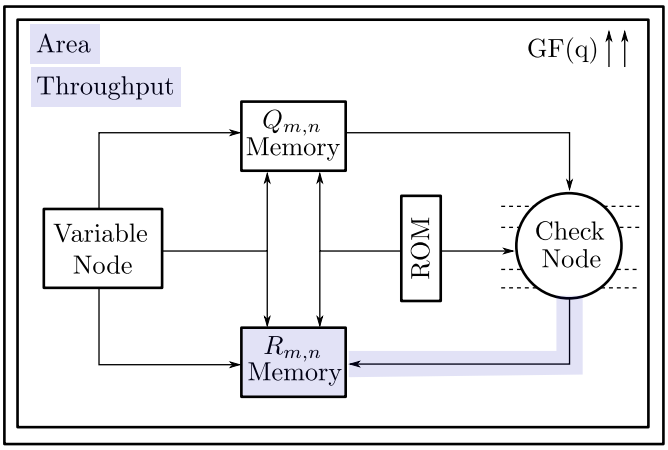


Figure 5.1: Key points of the improvements presented in this chapter.

In this chapter, the paper "A 630 Mbps Non-Binary LDPC Decoder for FPGA" is included. This paper shows implementation details for the approach presented in Chapter IV. The base algorithm is the T-MM from [42] and the design is particularized for a FPGA device. The results show an important reduction in the area and an increase in throughput compared to other proposals implemented in FPGA devices. Fig. 5.1 summarizes the key points that were improved in the derived paper.

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.

5.1 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 [6], 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) [6]. Extended Min-Sum (EMS) [14] and Min-Max [15] algorithms were proposed as approximations of the QSPA [6], 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 [16] [17] 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 [42] 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) [10]. It needs 83% less memory resources in comparison with a conventional implementation of T-MM algorithm [42] without introducing any performance loss. The throughput achieved is 630 Mbps, outperforming state-of-the-art NB-LDPC decoders implemented on FPGA devices [44][48][49].

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.

5.2 Basis on NB-LDPC codes and T-MM decoding algorithm

NB-LDPC codes are linear block codes defined by a sparse parity-check matrix $\mathbf{H}$ with M rows and N columns, where each non-zero element $h_{m,n}$ belongs to 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 $\mathbf{H}$ 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 $\mathbf{H}$, respectively.

Trellis Min-Max (T-MM) algorithm [42] 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 (5.1).

$$\Delta Q(a) = \min_{a' \in conf^*(1,2)} \left\{ \max \left(m1(a') \right) \right\} \tag{5.1}$$

$conf^*(1,2)$ [42] 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 (5.1) 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. 5.2, 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.

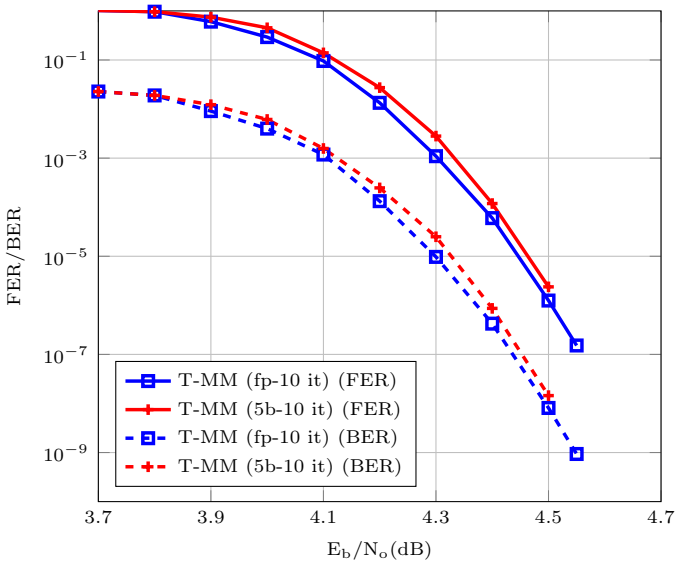


Figure 5.2: FER-BER performance of T-MM algorithm for the (2304,2048) NB-LDPC code over GF(16), with AWGN channel and BPSK modulation.

5.3 Proposed Decoder Architecture

T-MM algorithm [42] and the preceding T-EMS approach [16][17] 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.

5.3.1 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 (5.1) 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 [42]. 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 5.1 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 [10] [36]. In this code, $d_c = 36$, $d_v = 4$ and the number of bits for the quantized messages are $w = 5$.

Table 5.1: 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**

On the other hand, the T-MM based decoder architecture in [42] 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 [42]. It is important to

remark that this reduction in the amount of information exchanged between CN and VN does not introduce any performance loss compared to T-MM algorithm [42] 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. 5.3. 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.

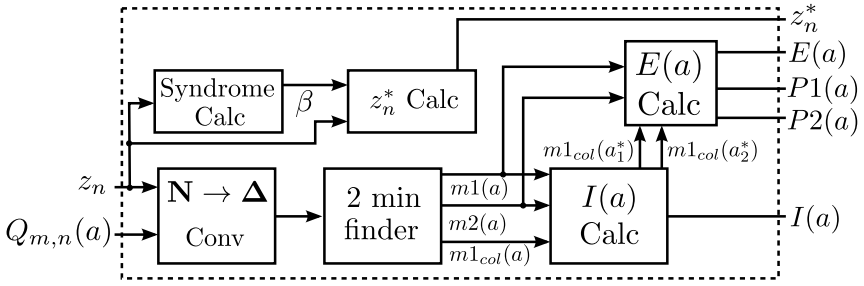


Figure 5.3: Check-node top-level architecture

5.3.2 Top-level decoder architecture

The CN architecture presented in Section 5.3.1 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 5.1 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. 5.4, where an example for GF(8) is used. In total, d_c decompression networks are required to generate all $q \times d_c$ values.

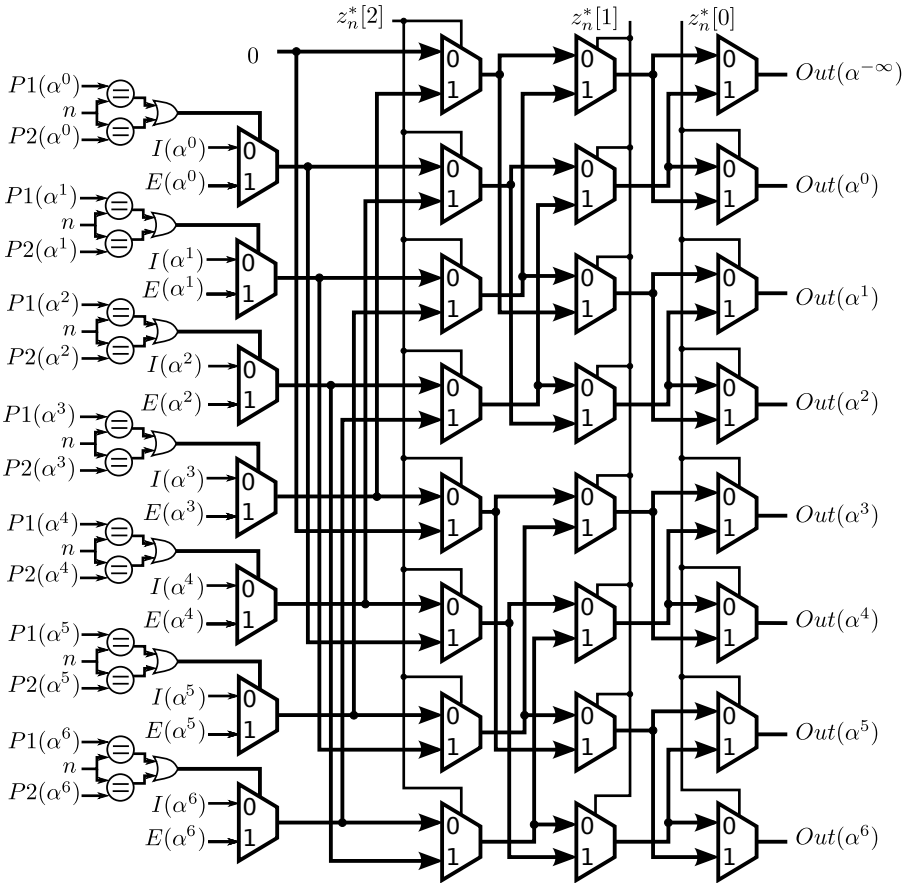


Figure 5.4: Decompression Network for CN output messages. Example for GF(8).

The top-level decoder architecture is presented in Fig. 5.5, where the Check Node Processor is the one from Fig. 5.3 and the blocks labeled as **DN** are the decompres-

sion network from Fig. 5.4. The blocks labeled as $\mathbf{P}$ 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 $\mathbf{H}$. 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

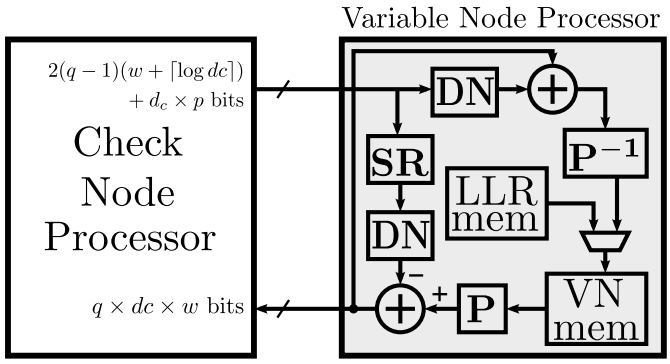


Figure 5.5: Top-level proposed decoder architecture

mem" block is the memory required to store the 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 $\mathbf{H}$ [10][36]. In the case of the target code, the size of the circulant sub-matrices is $QC = 64$, which allows us to implement 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. 5.5 was implemented on a Virtex-7 FPGA device for the target code and the results are presented in the last column of Table 5.2. Under the best knowledge of the authors, Table 5.2 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 [44]. In the case of the decoders proposed in [48] and [49], 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 5.2.

Table 5.2: Comparison of the proposed NB-LDPC decoder with other works implemented in FPGA devices from literature

Algorithm	Min-Max [44]	IHRB [48]	M-GBFDA [49]	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

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 [42] is, under the best knowledge of the authors, the most efficient decoder implementation for ASIC reported in the literature. We include in Table 5.3 the ASIC implementation results for the decoder from [42] 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 5.3 that our proposal outperforms the decoder from [42] 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 5.3: ASIC implementation of the proposed NB-LDPC decoder for the (2304,2048) NB-LDPC code over GF(16)

	[42]	[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

5.4 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.

Chapter 6

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

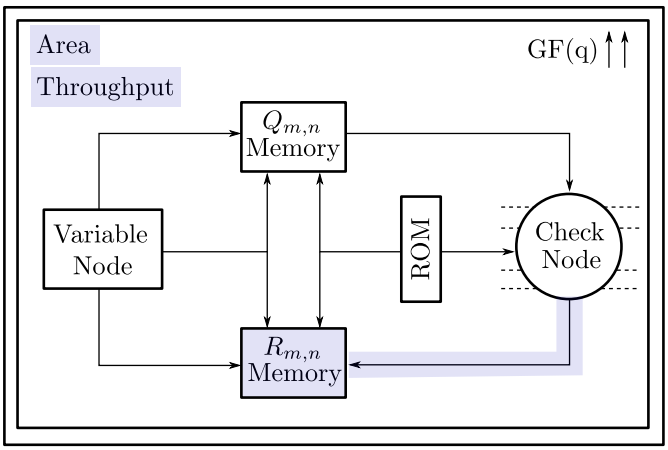


Figure 6.1: Key points of the improvements presented in this chapter.

In this chapter, the paper "High-performance NB-LDPC decoder with reduction of message exchange" is included. This research uses the results from the paper presented in Chapter 4 as starting point to develop a modified version of the T-MM algorithm (m-TMM) that further reduces the number of exchanged messages, compared to the results presented in Chapter 4. This reduction of messages lower

the number of storage elements in the decoder. Fig. 6.1 summarizes the key points that were improved in the derived paper.

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

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

6.1 Introduction

LDPC codes have been adopted by numerous communication standards such as DVB-S2 [3], IEEE 802.16e [4] and IEEE 802.11n [5], among others. Good error rate performance, low complexity decoders and high-rate decoding are some of the advantages of implementing LDPC codes over other error correction schemes.

Binary LDPC codes suffer from error correction degradation for short/medium codeword lengths. On the other hand, an effect called error floor appears with high Signal-to-Noise Ratio (SNR). This effect limits the error correction performance, so some additional processing is required to avoid it. Non-Binary LDPC (NB-LDPC) codes, defined over Galois Fields GF($q = 2^p$) with $p > 1$, were first investigated by Davey and MacKay [6] as an extension of binary LDPC codes, where $p = 1$. These codes emerge as an alternative to their binary counterparts to overcome the weaknesses shown by binary LDPC codes. Additionally, they improve the burst error correction capability, especially with high order Galois fields, and offer the possibility to be used in conjunction with high-order modulation schemes (16QAM, 64QAM, 256QAM), reducing the complexity in both the encoder and the decoder [7, 8]. Unfortunately, NB-LDPC codes have some drawbacks: i) high complexity of their check-node (CN); ii) large amount of area spent on storage elements (RAM memories and registers); and iii) routing congestion that limits the overall decoding throughput. From their appearance till now, many efforts have been put into mitigating these problems.

The first algorithm proposed to decode NB-LDPC codes was the Q-ary Sum-of-Product Algorithm (QSPA) [6], which was developed as a generalization of the Sum-of-Product Algorithm (SPA) for binary LDPC codes. Further improvements such as FFT-SPA[28], log-SPA and max-log-SPA[29], were proposed to reduce the complexity of the CN processing equations without introducing any performance loss. More recently, a trellis based implementation for QPSA (T-Max-log-QSPA) [30] was proposed, offering a solution that increases the throughput with respect to previous solutions based on QPSA. Its main drawback is that the required area is prohibitive for real applications in communications and storage systems. Extended Min-Sum (EMS) [14] and Min-Max [15] algorithms were presented as approximations of the QSPA [6], so that they reduce considerably the CN complexity, which only requires additions and/or comparisons. Additionally, EMS and Min-Max algorithms utilize forward-backward (FB) metrics to derive the CN output messages. These metrics involve serial computations which limit the throughput of the derived hardware architectures [15, 32].

Trellis Extended Min-Sum (T-EMS) algorithm was proposed [16, 17] with the aim of enabling parallel processing of the messages in the CN. The input messages are organized in a trellis structure, while the output messages are generated in parallel by means of an extra column included in the trellis. Trellis Min-Max (T-MM) algorithm in [42] adapts the idea of T-EMS to Min-Max algorithm. One Minimum Only TMM (OMO-TMM) [50] is an approximation of T-MM that reduces the complexity of the CN by obtaining only one minimum and estimating the second one. All these algorithms [16, 17, 42, 50] exchange $q \times d_c$ reliability values between CN and VN processors. This amount of exchanged messages is large enough to cause wiring congestion and this limits the maximum throughput, especially for high-rate NB-LDPC codes and high order Galois fields. Additionally, in decoder architectures with layered schedule, the CN output messages are stored to be used in the next iteration. So, the required memory, which is the main part of the area in NB-LDPC decoder architectures [34, 42, 50], is too high.

Other proposals from literature [51, 52, 31, 23, 30, 24] exchange a minor number of messages between CN and VN and vice versa. This fact, reduces the wiring congestion and the required memory resources, but implies the use of some kind of algorithm to generate the non-exchanged messages. Moreover, these approaches introduce a non-negligible performance loss that depends on the Galois Field order and the size of the reduced set.

In this paper we propose the modified T-MM algorithm (mT-MM) that reduces the number of check-to-variable messages taking advantage of the replicated information in the output messages from the CN in T-MM algorithm. The original idea comes from [53], where we proposed a method to compress the messages between CN and VN for NB-LDPC message-passing decoders. As the messages are not modified, this method does not introduce any performance loss. In [54] we particularize the proposal in [53] to T-MM algorithm, and we detail a hardware

architecture for the CN processor and for a decoder with layered schedule. In this paper we extend the work in [54], and present a modification of the T-MM algorithm that allow us to reduce even more the number of exchanged messages. The CN output messages are split in two arrays: one that compresses the extrinsic information and another which represents the intrinsic one. Based on statistical analysis we found that reducing the size of the intrinsic information from q to only two elements introduces a negligible performance loss for high-rate LDPC codes over GF(16) and GF(32) and a performance loss smaller than 0.07dB for high-rate NB-LDPC codes over GF(64), compared to T-MM algorithm [42]. Additionally, we present a high-throughput architecture for the entire decoder (with layered scheduled), which includes the mT-MM algorithm in the CN processor, and compare our implementation results for 90nm CMOS technology with other state-of-the-art decoder architectures.

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

6.2 Trellis Min-Max decoding algorithm

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

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

the GF symbol associated to the highest reliability. $Q_n(a)$ is the *a posteriori* information which is updated as the message passing decoding algorithm progresses.

The CN operations solve the parity check equations, based on the messages from the VN ($Q_{mn}(a)$), and updates the reliability values for each GF symbol a. In this paper we propose an algorithm for the CN to do these tasks (described in Section 6.3), which is based on Trellis Min-Max (T-MM) algorithm [42]. T-MM algorithm offers a good trade-off between coding gain and decoding complexity compared to other proposals from literature.

The basic steps to implement the CN processor of the T-MM algorithm [42] are presented in Algorithm 8. Step 1 involves normal-to-delta domain transformation using the input messages and the hard decision symbols. This transformation ensures that the reliabilities corresponding to the hard-decision symbols z_n are related to the GF symbols $\alpha^{-\infty}$, simplifying the rest of the steps in T-MM algorithm. Step 2 obtains the syndrome β by adding all hard-decision symbols. Step 3 calculates the first and second most reliable messages (minimum values), $m1(a)$ and $m2(a)$, by means of the function ψ, which also extracts the position of $m1(a)$, $m1_{col}(a)$. Step 4 computes the extra column of the trellis, $\Delta Q(a)$, which collects the reliability of the most reliable path for each GF symbol a.

$conf^*(n_r, n_c)$ [42] is the configuration set which selects the possible paths conformed by the n_r symbols with higher reliability value. From all the possible paths, the ones that deviate at most n_c times from the hard-decision are selected. From this reduced set of possible paths, the one chosen for the corresponding $\Delta Q(a)$ value is the one that ensures the highest reliability (minimum value). In this paper we consider the case where $n_r = 1$ and $n_c = 2$. So, only the most reliable messages are considered (First minimum set $m1(a)$) and only one and two deviations paths are taken into account.

Finally, the CN output messages are generated in two steps. First (Step 5 in Algorithm 8), each row of $\Delta R_{m,n}(a)$ is filled with the corresponding $\Delta Q(a)$ reliability, except for the columns that correspond to the stage in the trellis where deviations from the hard-decision path are made. In cases where only one deviation is made, the empty column is filled with the reliability of the second most reliable symbol $m2(a)$. In those cases where two deviations are made, empty columns are filled with the $m1(a)$ reliability. Second (Step 6), conversion from delta to normal domain is required for the CN output messages, where β is used to correct the tentative hard-decision symbols. Additionally, a scaling factor λ is used to improve the performance and convergence rate of T-MM algorithm.

Algorithm 8: T-MM Algorithm [42]

Input: $\mathbf{Q_{mn}}$, $z_n = \arg\min_{a \in \mathrm{GF}(q)} Q_{mn}(a) \ \forall \ n \in \mathcal{N}(m)$

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

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

end

2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in \mathrm{GF}(q)$

3 $[m1(a), m1_{col}(a), m2(a)] = \psi\{\Delta Q_{mn_i}(a)\Big|_{i=1}^{d_c}\}$

4 $\Delta Q(a) = \min_{\eta'_k(a) \in conf^*(1,2)} \left\{ \max_{k=1,2} \left(m1(\eta'_k(a)) \right) \right\}$

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

5 $\quad$ **if** $m1(\eta'_1(a)) \neq \Delta Q_{mn_j}(a)$ **and** $m1(\eta'_2(a)) \neq \Delta Q_{mn_j}(a)$ **then**
$\quad\quad \Delta R_{mn_j}(a) = \Delta Q(a)$
$\quad$ **else if** $\eta'_1(a) = \eta'_2(a)$ **then**
$\quad\quad \Delta R_{mn_j}(a) = m2(a)$
$\quad$ **else**
$\quad\quad \Delta R_{mn_j}(a) = m1(a)$
$\quad$ **end**

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

end

Output: $\mathbf{R_{mn}}$

6.3 Modified Trellis Min-Max Algorithm

This section is organized as follows: in Section 6.3.1 we reformulate T-MM algorithm to introduce some variables required to explain how replicated information is reduced and that are used in the definition of the proposed modified T-MM algorithm. Section 6.3.2 extends the explanation of the algorithm proposed in [54] taking as a reference the algorithm reformulated in Section 6.3.1 and also includes an analogy with binary LDPC decoders. Finally, Section 6.3.3 defines the new algorithm (modified T-MM, mT-MM), which is based on an statistical analysis, and gives FER performance results for high-rate NB-LDPC codes over GF(16), GF(32) and GF(64).

6.3.1 Reformulation of Trellis Min-Max Algorithm

In this section we reformulate the Trellis Min-Max Algorithm (Algorithm 8) as a first step to define our proposal. As can be seen in Algorithm 9, steps 4 and 5 are the ones reformulated. The function ψ' in Step 4 obtains which path in the trellis was used to obtain $\Delta Q(a)$, that is, the most reliable path. Considering that a maximum of two deviations is evaluated, the function returns the two GF symbols that define this path, $\eta_1^*(a)$ and $\eta_2^*(a)$. If the path used to obtain $\Delta Q(a)$ has only one deviation from the hard-decision path, the function ψ' equals $\eta_2^*(a)$ to $\eta_1^*(a)$. On the other hand, Step 5 calculates $\Delta R_{mn}(a)$, which is equaled to $\Delta Q(a)$, the first minimum of $\Delta Q_{mn}(m1(a))$ or its second minimum ($m2(a)$), depending on the deviation information ($\eta_1^*(a)$ and $\eta_2^*(a)$). For a symbol a, if the most reliable path does not deviate at column j ($m1_{col}(\eta_1^*(a)) \neq j$ **and** $m1_{col}(\eta_2^*(a)) \neq j$) the extra column information $\Delta Q(a)$ is assigned to the output $\Delta R_{mn}(a)$. On the other hand, two different updates can be performed at the columns where deviations from the most reliable path are made: (i) if this path has only one deviation, the second minimum $m2(a)$ is assigned to $\Delta R_{m,n}(a)$; (ii) if this path has two deviations, $m1(a)$ is assigned to the output.

Fig. 6.2 includes an example of trellis with GF(4) and $d_c = 5$. It shows the CN input messages before ($Q_{mn}(a)$) and after ($\Delta Q_{mn}(a)$) delta domain transformation. The hard-decision symbols are $\mathbf{z} = \{\alpha^1, \alpha^0, 0, \alpha^0, 0\}$. After the normal-to-delta domain transformation, the reliabilities $\Delta Q_{mn}(a)$ in the first row of the trellis are equal to 0. The minimum value per row (per GF symbol a) of $\Delta Q_{mn}(a)$ is enclosed by a dotted box, so, the most reliable path for a GF symbol a must include only these boxes. In Fig. 6.2 the most reliable path for the symbol α^2 is shown in red color ($\alpha^2 = \alpha^0 + \alpha^1$) . This path is most reliable (reliability equal to the maximum between 5 and 10) than the path that makes only one deviation (reliability equal to 17), so $\Delta Q(\alpha^2) = 10$. In a similar way, the most reliable paths for the symbols α^0 and α^1 are built, but in these cases, with only one deviation from the hard decision path ($\Delta Q(\alpha^0) = 5$ and $\Delta Q(\alpha^1) = 10$). For symbol α^2, the new variables defined in Algorithm 9 are $\eta_1^*(\alpha^2) = \alpha^0$ and $\eta_2^*(\alpha^2) = \alpha^1$. Thus, $m1_{col}(\eta_1^*(\alpha^2)) = 1$, $m1_{col}(\eta_2^*(\alpha^2)) = 2$, $\Delta R_{mn_1}(\alpha^2) = \Delta R_{mn_2}(\alpha^2) = m1(\alpha^2) = 17$ and $\Delta R_{mn_3}(\alpha^2) = \Delta R_{mn_4}(\alpha^2) = \Delta R_{mn_5}(\alpha^2) = \Delta Q(\alpha^2) = 10$.

6.3.2 Reduction of replicated information in check-to-variable exchanged messages

For a better understanding of our proposal, an analogy with binary LDPC decoders is established. In [55] a decoder architecture for binary LDPCs is proposed. In this architecture the messages are compressed in a similar way to which we propose here for the non-binary case. Instead of sending an individual message to each neighbor VN, a CN sends the same message to all its connected VNs, which includes the first minimum, the second minimum, the position of the first minimum and the

Algorithm 9: Reformulated Trellis Min-Max Algorithm

Input: $\mathbf{Q_{mn}}$

$z_n = \arg\min_{a \in \mathrm{GF}(q)} Q_{mn}(a) \ \forall \ n \in \mathcal{N}(m)$

1 $\Delta Q_{mn}(a + z_n) = Q_{mn}(a)$

2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in \mathrm{GF}(q)$

3 $[m1(a), m1_{col}(a), m2(a)] = \psi\{\Delta Q_{mn_i}(a)\Big|_{i=1}^{d_c}\}$

4 $\Delta Q(a) = \min_{\eta'_k(a) \in \ conf^*(1,2)} \left\{ \max\left(m1(\eta'_k(a))\right) \right\}$

$[\eta_1^*(a), \eta_2^*(a)] = \psi'(\min_{\eta'_k(a) \in \ conf^*(1,2)} \left\{ \max\left(m1(\eta'_k(a))\right) \right\})$

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

5 **if** $m1_{col}(\eta_1^*(a)) \neq j$ **and** $m1_{col}(\eta_2^*(a)) \neq j$ **then**
$\Delta R_{m,n_j}(a) = \Delta Q(a)$
else if $m1_{col}(\eta_1^*(a)) = m1_{col}(\eta_2^*(a))$ **then**
$\Delta R_{mn_j}(a) = m2(a)$
else
$\Delta R_{mn_j}(a) = m1(a)$
end

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

end

Output: $\mathbf{R_{mn}}$

sign (that depends on the syndrome value). In this way, the routing congestion is reduced.

In the non-binary case, the T-MM algorithm behaves in a similar way. Step 5 in Algorithm 9 generates the CN output messages in delta domain. For each GF symbol a:

$$\Delta R_{mn}(a) = \Delta Q(a) \ \forall \ n \in \mathcal{N}(m) \setminus \{m1_{col}(\eta_1^*(a)), m1_{col}(\eta_2^*(a))\} \tag{6.1}$$

In (6.1), $m1_{col}(\eta_1^*(a))$ and $m1_{col}(\eta_2^*(a))$ are the positions of the symbols that ensure the highest reliability of $\Delta Q(a)$ (Step 4 of Algorithm 9). Let us consider the case where the highest reliability path in $\Delta Q(a)$ was built performing only one deviation from the hard-decision path. In this particular case, the exclusion set is reduced to only one position and $\eta_1^*(a) = \eta_2^*(a)$. Therefore, $\Delta Q(a)$ is equal to $m2(a)$ and

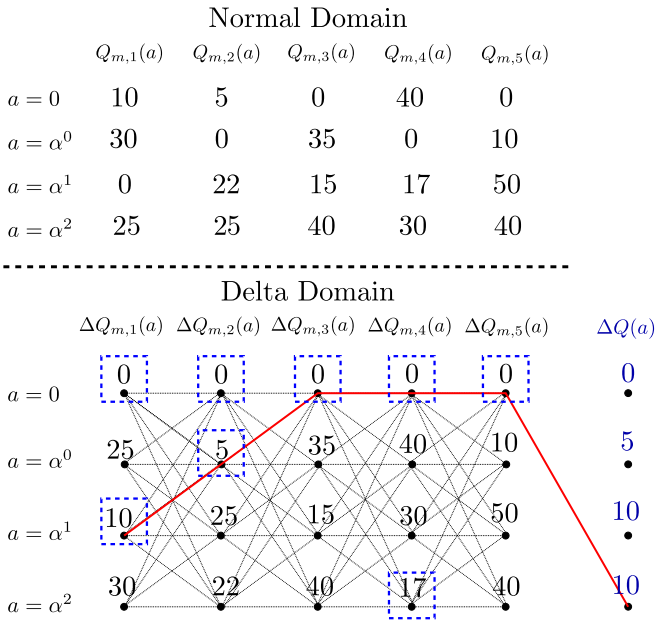


Figure 6.2: Example of CN input messages in normal domain (upper size). Messages in delta domain and organized in trellis way including the extra column $\Delta Q(a)$ (bottom size). Example for GF(4) and $d_c = 5$.

(6.1) can be rewritten as (6.2), which corresponds to the generalization of the CN output message of binary Min-Sum LDPC decoders to the non-binary ones.

$$\Delta R_{mn}(a) = \begin{cases} m1(a) \ \forall \ n \in \mathcal{N}(m) \setminus \{m1_{col}(a)\} \\ \\ m2(a) \ \forall \ n \in \{m1_{col}(a)\} \end{cases} \tag{6.2}$$

Although (6.2) is a particular case of (6.1) in T-MM algorithm, it is useful to remark that $\Delta Q(a)$ plays the role of the intrinsic information from the binary Min-Sum. For the extrinsic messages we define a set $E(a)$ (6.3) which includes the $m1(a)$ or $m2(a)$ reliabilities depending on the number of deviations (1 or 2) from the hard-decision path.

$$E(a) = \begin{cases} m2(a) & \text{if } \ m1_{col}(\eta_1^*(a)) = m1_{col}(\eta_2^*(a)) \\ m1(a) & \text{otherwise} \end{cases} \tag{6.3}$$

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

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

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

In terms of bits, the total among of information exchanged from CN to VN is $2\times(q-1)\times(w+\lceil \log d_c \rceil)+d_c\times p$ bits, where w is the number of bits used to represent the reliability of messages in the decoder. This information has been detailed in Table 6.1 for each set exchanged from CN to VN. In this way, the information sent is only reorganized (not modified), so we do not have any performance loss with respect to T-MM.

Table 6.1: Number of bits exchanged from CN to VN processor after reduction of the replicated information

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

6.3.3 Modified Trellis Min-Max algorithm

In this section we propose a new definition of the CN output messages (based on Section 6.3.2) that allow us to reduce the exchanged messages from CN to VN even more. This new definition keeps only a minimum amount of values from $\Delta Q(a)$ (the most reliable ones) and obtains the rest using an approximation function.

First, a statistical analysis for the set $\Delta Q(a), a \in \mathrm{GF}(q)$ was done in order to find its mean value. Using a software model for a NB-LDPC decoder based on T-MM algorithm, we obtained $\Delta Q(a)$ for all the M rows of $\mathbf{H}$ when decoding 10^6 noisy sequences (for $E_b/N_0 = 4.3dB$). Then, we ordered each set $\Delta Q(a)$ from lower to higher value and obtained the mean value of the ordered sets $\Delta Q(a)$ ($\overline{\Delta Q}(a)$). The results of the analysis are presented in Fig. 6.3. We used the (837,726) NB-LDPC code over GF(32), with degree distribution $d_c = 27, d_v = 4$ built using the methods presented in [36]. Besides, we replicated the analysis for NB-LDPC codes

with different degree distributions and Galois Field orders and we obtained the same conclusions. As can be seen in Fig. 6.3, there is a big increase of mean value from one $\overline{\Delta Q}(a)$ index to the next for the first indexes, however, this increase is lower for the rest of indexes. Based on this observation, we propose to keep only the first minimum, ΔQ_{m1}, and the second minimum, ΔQ_{m2}, from the set $\Delta Q(a) \; \forall \; a \; \in \; \mathrm{GF}(q) \setminus \alpha^{-\infty}$. Storing only a limited set of values from $\Delta Q(a)$, the exchanged information between CN and VN is reduced.

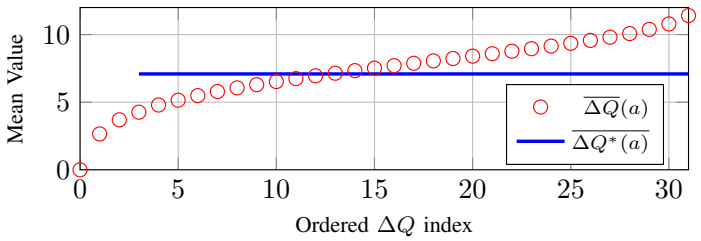


Figure 6.3: Mean values for each reliability in the ordered set $\Delta Q(a)$. The code used is the (837,726) NB-LDPC code over GF(32).

At the VN processor, we propose to approximate the rest $q-3$ $\Delta Q(a)$ values using (6.4), where a_{m1} and a_{m2} are the GF symbols corresponding to ΔQ_{m1} and ΔQ_{m2}, respectively. As the second most reliable value, ΔQ_{m2}, is updated at each iteration, the distance between it and the values $\Delta Q(a)$ approximated using (6.4) is kept. So, it is expected that the fixed scaling factor γ is greater than one, to ensure that the reliabilities of the approximated values of $\Delta Q(a)$ will not be lower than ΔQ_{m2}.

$$\Delta Q(a) = \gamma \times \Delta Q_{m2} \quad \forall \; a \; \in \; \mathrm{GF}(q) \; \setminus \{\alpha^{-\infty}, a_{m1}, a_{m2}\} \tag{6.4}$$

The value of the scaling factor γ from (6.4) is obtained as follows. First, we calculate the mean value of the entire set $\Delta Q^*(a) = \Delta Q(a) \; \forall \; a \; \in \; \mathrm{GF}(q) \; \setminus \; \{\alpha^{-\infty}, a_{m1}, a_{m2}\}$, named as $\overline{\Delta Q^*(a)}$ in Fig. 6.3. Then, we obtain the initial value of γ dividing $\overline{\Delta Q^*(a)}$ by the mean value of ΔQ_{m2} ($\overline{\Delta Q_{m2}}$). In Fig. 6.3, $\overline{\Delta Q^*(a)} = 7.097$ and ΔQ_{m2} is $\overline{\Delta Q_{m2}} = 3.697$, thus the initial value for γ is 1.9198. Finally, we adjust the initial value chosen for γ by means of frame error-rate (FER) simulations, optimized for $E_b/N_0 = 4.3dB$.

Taking into account the modifications presented above and the definitions made in 6.3.2, Algorithm 10 describes the modified Trellis Min-Max (m-TMM) decoding algorithm. Function $\psi^{''}$ is a modified version of the ψ function from Algorithm 8 which also extracts the position (GF symbol) of the second minimum.

We include in Table 6.2 the number of bits exchanged between CN and VN for our proposal and for other works from literature. The rightmost column includes

Algorithm 10: Modified Trellis Min-Max Algorithm

Input: $\mathbf{Q_{mn}}$

$z_n = \arg\min_{a \in \mathrm{GF}(q)} Q_{mn}(a) \ \forall \ n \in \mathcal{N}(m)$

1 $\Delta Q_{mn}(a + z_n) = Q_{mn}(a)$

2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in \mathrm{GF}(q)$

3 $[m1(a), m1_{col}(a), m2(a)] = \psi\{\Delta Q_{m,n_i}(a)\Big|_{i=1}^{d_c}\}$

4 $\Delta Q(a) = \min_{\eta'_k(a) \in \ conf^*(1,2)} \left\{ \max\left(m1(\eta'_k(a))\right) \right\}$

$[\eta_1^*(a), \eta_2^*(a)] = \psi'(\min_{\eta'_k(a) \in \ conf^*(1,2)} \left\{ \max\left(m1(\eta'_k(a))\right) \right\})$

5 $[\Delta Q_{m1}, a_{m1}, \Delta Q_{m2}, a_{m2}] = \psi''\{\Delta Q(a)\Big|_{a=\alpha^0}^{\alpha^{q-2}}\}$

6 $E(a) = \begin{cases} m2(a) & \text{if } \ m1_{col}(\eta_1^*(a)) = m1_{col}(\eta_2^*(a)) \\ m1(a) & \text{otherwise} \end{cases}$

7 $z_n^* = z_n + \beta \quad \forall \ n \ \in \ \mathcal{N}(m)$

Output: $\begin{cases} \Delta Q_{m1}, a_{m1}, \Delta Q_{m2}, a_{m2} \\ E(a) \\ z_n^* \\ P(a) = \{ \ m1_{col}(\eta_1^*(a)), m1_{col}(\eta_2^*(a)) \ \} \end{cases}$

numerical results for the (837,726) NB-LDPC code over GF(32) [36] with degree distribution ($d_c = 27, d_v = 4$). We consider the same number of quantization bits for all proposals ($w = 6$ bits) and we set $n_m = 16$ and $n_v = 5$ according to [52, 44, 24, 23] as they propose for their codes. The work from [42] exchanges a full set of messages which turns into a higher number of bits at the CN output. As can be seen, proposals from [52, 44, 24] eliminate the q-dependence, exchanging only a fraction $n_m < q$ of the reliabilities. This proposals maintain a strong dependence on the CN degree d_c, which penalizes for high-rate NB-LDPC codes. The work from [23] reduces even more the fraction of output messages at the CN compared to previous proposals from literature, being $n_v < n_m < q$. This reduction is offered at the cost of some error-correction degradation and the need of including real-multipliers at the VN for the message approximation. The work from [53] exchanges a fixed number of sets and the size of each set depends of q and d_c without introducing any performance loss compared to [42]. Finally, we propose in this work a cardinality reduction of the set $\Delta Q(a)$ to only two elements, reducing the total among of bits exchanged to the VN compared to the others proposals

Table 6.2: Comparison between multiple proposals from literature to reduce the number of messages exchanged from CN to VN

Proposal	Number of bits	($q = 2^p = 32$, $d_c = 27$, $w = 6$, $n_m = 16$, $n_v = 5$)
[42]	$q \times d_c \times w$	5184 bits
[52, 44, 24]	$n_m \times d_c \times w$	2592 bits
[23]	$n_v \times d_c \times w$	810 bits
[53]	$2 \times (q-1) \times (w + \lceil \log d_c \rceil) + d_c \times p$	817 bits
This work	$2 \times (q-1) \times \lceil \log d_c \rceil + (q+1) \times w + (d_c + 2) \times p$	653 bits

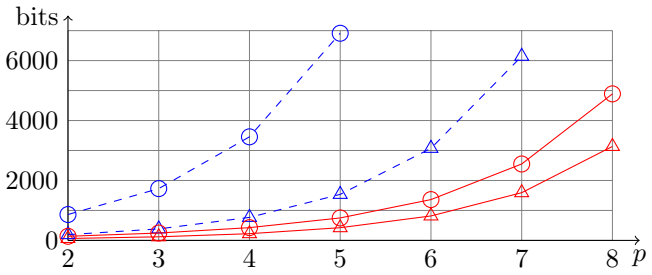


Figure 6.4: Number of bits exchanged from CN to VN varying the GF order. Dashed lines corresponds to T-MM and solid lines to mT-MM. Circle mark corresponds to $d_c = 36$ and Triangle mark to $d_c = 8$. $w = 6$.

from Table 6.2 as can be seen in the example from its rightmost column for the high-rate NB-LDPC code over GF(32).

In terms of complexity, the CN processor of Algorithm 10 has less computational load than the one of 9 because it does not compute $q \times d_c$ output messages. So, the number of wires between CN and VN is also reduced.

Fig. 6.4 and Fig. 6.5 compare the amount of bits exchanged from CN to VN in a conventional implementation of T-MM with our proposal, varying the field order (p) or the CN degree (d_c), respectively. In all cases, the proposed approach outperforms conventional T-MM in terms of exchanged bits. The differences are considerably higher when the field order and/or the check node degree is increased. This has a great impact on the area of a decoder that uses a layered schedule, as will be seen in Section 6.4.

Fig. 6.6 shows the FER performance of the proposed modified Trellis Min-Max (mT-MM) decoding algorithm (floating-point and fixed-point versions (6 bits)) for

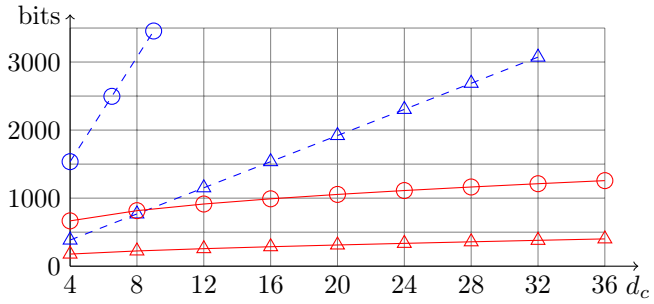


Figure 6.5: Number of bits exchanged from CN to VN varying the CN degree. Dashed lines corresponds to T-MM and solid lines to mT-MM. Circle mark corresponds to $q = 64$ and Triangle mark to $q = 16$. $w = 6$.

the (837,726) NB-LDPC code over GF(32), with 15 iterations and $\gamma = 2.0$ (approximated to be hardware-friendly value). It also includes the performance of the floating-point T-MM algorithm [42] with 15 iterations for performance comparison purposes. As can be seen, our proposed algorithm introduces a negligible performance loss of 0.01dB with respect to T-MM (floating-point versions).

Fig. 6.6 also shows the FER performance of other algorithms from the literature (SMSA [32], T-Max-log-QSPA [30], RMM [34] and OMO-TMM [50]) that will be used in Section 6.4.3 to compare their implementation results under the same performance. The number of iterations of each algorithm is adjusted to obtain a performance similar to [34] with 15 iterations, that is, a FER approximately equal to 10^{-6} for $E_b/N_0 = 4.55dB$.

Fig. 6.7 and Fig. 6.8 show FER performance results for the (2304,2048) NB-LDPC code over GF(16) ($d_c = 36, d_v = 4$) and the (1536,1344) NB-LDPC code over GF(64) ($d_c = 24, d_v = 3$), respectively. Both NB-LDPC codes are constructed based on the methods from [36]. The algorithms analysed are T-MM and mT-MM. The results show that mT-MM has a performance loss of 0.05dB for the code in Fig. 6.7 and 0.07dB for the code in Fig. 6.8 with respect to T-MM. Thus, the proposed mT-MM algorithm achieves good FER performance results for several GF orders and different degree distributions.

Table 6.3 summarizes the parameters needed to adjust the initial value for the scaling value γ ($\overline{\Delta Q^*(a)}$ and $\overline{\Delta Q_{m2}}$), as well as the hardware-friendly value of γ (γ_{HF}) chosen to generate the FER curves from Fig. 6.6, Fig. 6.7 and Fig. 6.8.

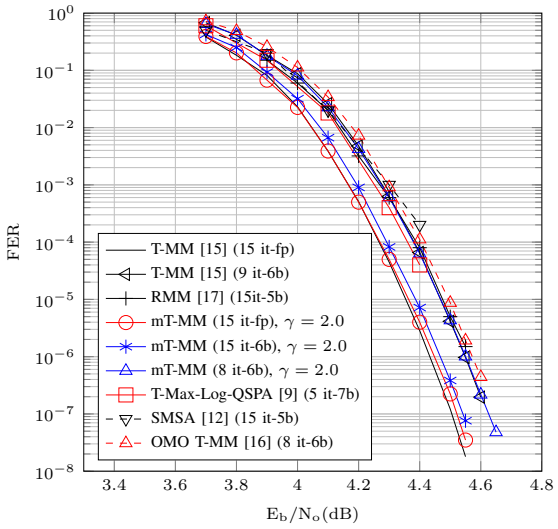


Figure 6.6: Frame-error-rate simulation for the (837,726) NB-LDPC code over GF(32), BPSK modulated and assuming AWGN channel

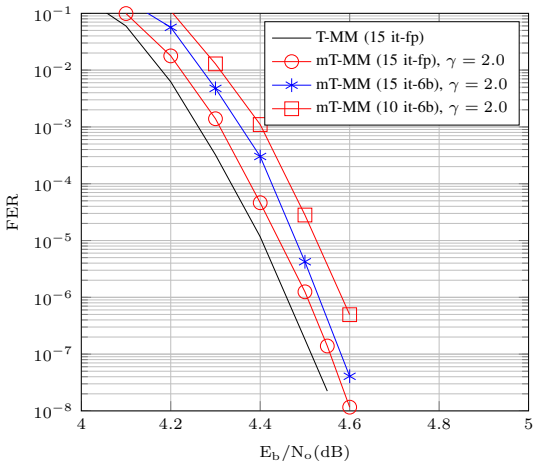


Figure 6.7: Frame-error-rate simulation for the (2304,2048) NB-LDPC code over GF(16), BPSK modulated and assuming AWGN channel

6.4 NB-LDPC Decoder Implementation

In this section we describe the architecture designed to implement the proposed mT-MM Algorithm (Section 6.3.3). Additionally, we include the top level design

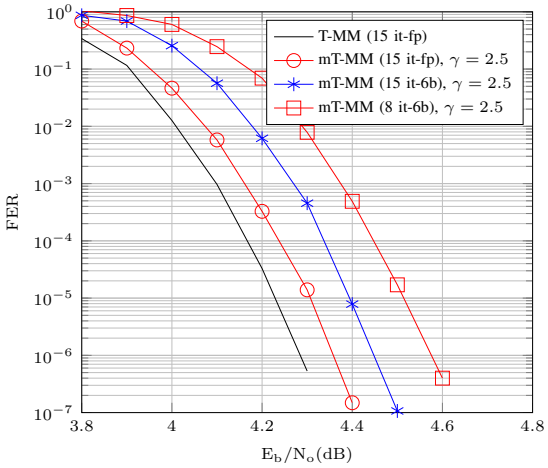


Figure 6.8: Frame-error-rate simulation for the (1536,1344) NB-LDPC code over GF(64), BPSK modulated and assuming AWGN channel

Table 6.3: Experimental results to select the appropriate scaling value γ, optimized for $E_b/N_0 = 4.3dB$

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

of a NB-LDPC decoder which uses a layered schedule. The proposed decoder is designed for quasi-cyclic NB-LDPC codes over GF(q) constructed applying the methods in [36], where **H** is formed by QC x QC circulant sub-matrices. These sub-matrices can be composed of zero elements or a cyclic shifted identity matrix with non-zero elements from GF(q). In this way, the number of rows and columns in **H** is $M = QC \times d_v$ and $N = QC \times d_c$, respectively.

6.4.1 CN architecture for mT-MM algorithm

Parallel processing is adopted in the CN processor, so its latency is kept low and this increases the overall throughput, as will be seen in next section. The main characteristic of the proposed mT-MM Algorithm is to move part of the complexity of the CN processor to the VN processor. In this way, the number of exchanged messages between them and also the storage resources of the decoder are reduced. Therefore, the CN architecture presented in this section requires less functional blocks than a conventional implementation of T-MM algorithm [42].

Next, the hardware required to perform Algorithm 10 is detailed. Fig. 6.10 shows the block diagram for the top-level CN architecture, where each block corresponds to a step in the mT-MM algorithm.

Step 1, that is, Normal-to-Delta domain transformation is made by means of d_c permutation networks which follow the structure introduced in [39]. Each one requires $q \times \log(q)$ w-bit MUXES. CN syndrome (Step 2) is obtained using GF-adders in a tree structure ($(d_c - 1) \times p$ XOR gates).

Function ψ (Step 3) is implemented using a tree-based two minimum finder [40], modified to also extract the position of the first minimum [17]. In total $q - 1$ two-minimum finders with d_c inputs are required. Each one is implemented with $2 \times d_c$ w-bit comparators and $3 \times d_c$ w-bit MUXES.

The extra column values, $\Delta Q(a)$, and the corresponding path information (Step 4) are generated using only the most reliable values $m1(a)$ and their corresponding positions $m1_{col}(a)$. A maximum of two deviations from the hard decision path is considered, so, the most reliable path for each value in the set $\Delta Q(a)$ is chosen among a maximum of $q/2$ possible paths (for example, the possible paths for the GF symbol α^0 and GF(8) are α^0, α^1 α^3, α^2 α^6, α^4 α^5). Since the possible paths are different for each value of $\Delta Q(a)$ (for each GF symbol), a custom wired network is required for each one of the $q - 1$ processors used to generate all $\Delta Q(a)$ values. As an example, the processor for the GF symbol α^0 and GF(8) is presented in Fig. 6.9. The SAT block from Fig. 6.9 excludes paths deviating more than once in the same stage of trellis. That is, when it detects more than one $m1(a)$ in the same path coming from the same column of the trellis, it assigns the maximum value (minimum reliability) to the one-minimum finder input.

Step 5 is implemented as a single two minimum finder with $q - 1$ inputs as shown in Fig. 6.9. It selects the first and second minimum values of the set $\Delta Q(a)$ (ΔQ_{m1} and ΔQ_{m2}, respectively) and their position (GF symbol), a_{m1} and a_{m2}. It receives as inputs the outputs of the $q - 1$ extra-column processors to extract the two most reliable (minimum) values.

The computation of the set $E(a)$ (Step 6) requires $(q-1)$ $\lceil \log(d_c) \rceil$-bit comparators and $(q - 1)$ w-bit MUXES. With this hardware we distinguish paths with one

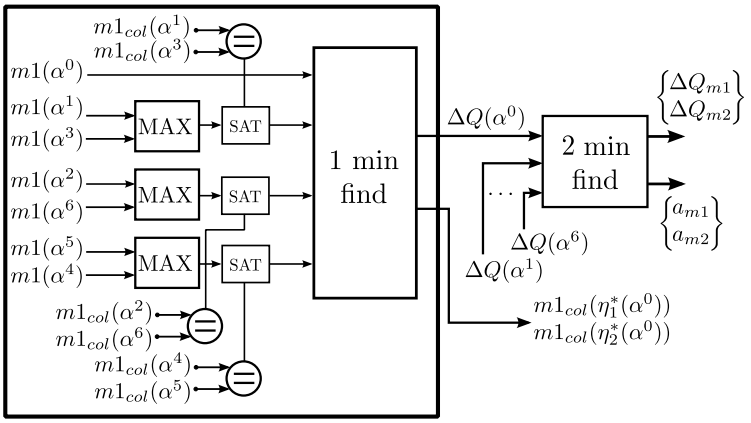


Figure 6.9: Extra-Column processor. Example for GF(8) and symbol α^0

$(E(a) = m2(a))$ and two deviations $(E(a) = m1(a))$ from the hard-decision path, for each GF(q) symbol.

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

As can be seen in Fig. 6.10, some blocks do not depend on others, so they can be processed in parallel to the rest of blocks. This is the case of the CN syndrome calculation (Step 2), β, and the extrinsic syndromes calculation (Step 7), z_n^*. Additionally, the $E(a)$ calculation (Step 6) and the two-minimum finder (Step 5) can be processed at the same time. This reduces the total latency of the CN architecture.

As it will be explained in Section 6.4.2, the VN processor uses z_n^*, $E(a)$, $P(a)$, ΔQ_{m1}, ΔQ_{m2}, a_{m1} and a_{m2} to build R_{mn} in Algorithm 9. So, the total among of information exchanged from CN to VN is $(q-1) \times (w + 2 \times \lceil \log d_c \rceil) + d_c \times p + 2 \times p + 2 \times w$ bits, where w is the number of bits used to represent the reliability of messages in the decoder.

6.4.2 Top-level decoder architecture

In this Section we explain how the CN architecture for the mT-MM algorithm from Section 6.4.1 is included in a complete decoder with horizontal layered schedule. This schedule improves the convergence of the decoding algorithm in comparison with the flooding one. In this way, the number of iterations is reduced and hence the throughput is improved. On the other hand, the area of the resulting decoder is considerably lower than the one required by a fully parallel implementation.

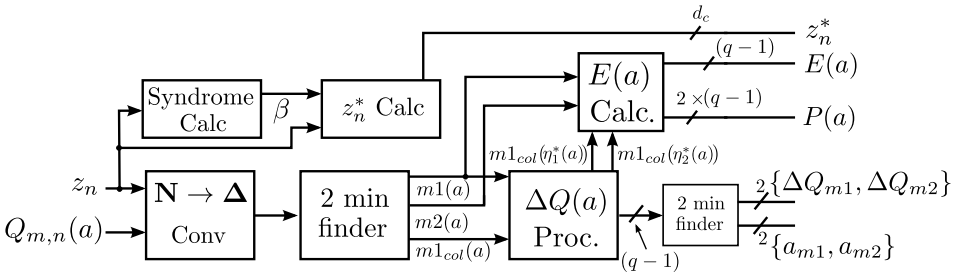


Figure 6.10: Proposed check-node block diagram

In Algorithm 11 the layered schedule for the proposed decoder is presented, where mT-MM is the CN processor which implements Algorithm 10, and DN is the decompression network from Algorithm 12. The VN processor uses the DN blocks, which generate R_{mn} by using the information given by the mT-MM CN processor.

Algorithm 12 details the operations required to reconstruct R_{mn}, that is, the entire set of $q \times d_c$ messages that goes from CN to VN processors. The decompression network (DN) has as input the reduced set of messages coming from the CN.

The complete block diagram for the proposed decoder is presented in Fig. 6.11. As can be seen, there is only one check node processor and one VN processor, which processes one row of $\mathbf{H}$ per clock cycle. Layered schedule requires to store the CN output messages from one iteration to be used in the next one. This is done by means of a shift register with M stages (SR in Fig. 6.11). The implementation of a conventional CN processor with $q \times d_c$ output messages would require $q \times d_c \times w \times M$ registers. Our proposal only requires $M \times [(q-1) \times (w + 2 \times \lceil \log d_c \rceil) + d_c \times p + 2 \times w]$ registers to store the messages from the last iteration. This reduces the storage elements following the behavior presented in Fig. 6.4 and Fig. 6.5 when the field order or the CN degree is varied.

The blocks $\mathbf{P}$ and $\mathbf{P^{-1}}$ in Fig. 6.11 perform direct and inverse permutation of messages from VN to CN and vice versa, respectively. The permutation is done based on the $h_{m,n}$ non-zero values of $\mathbf{H}$.

The "VN mem" block is the memory required to store the messages in the VN processor during the decoding process. The depth of the required memories fits with the size of the circulant sub-matrices (QC) which form $\mathbf{H}$ [36]. On the other hand, the block "LLR mem" stores the channel information. This information is loaded in "VN mem" at the beginning of each new decoding frame.

Fig. 6.12 shows the implementation of a decompression network (DN) for GF(4). A total of d_c decompression networks are required to generate all $q \times d_c$ R_{mn} values. Note that two decompression networks are included in the VN processor.

Algorithm 11: Layered Schedule for the Proposed Decoder

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

Inicialization:

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

Main Loop:

while $t \leq MaxIter$ **do**

for $l = 1$ **to** M **do**

1 $R_{mn}^{(t-1)}(a) = \mathrm{DN}\{\Delta Q_{m1}, a_{m1}, \Delta Q_{m2}, a_{m2}, E(a), z_n^*, P(a)\}$

2 $Q'_{mn}(a) = Q_n^{(t-1)}(h_{mn}a) - R_{mn}^{(t-1)}(a)$

3 $Q_{mn}(a) = Q'_{mn}(a) - \min\ \{Q'_{mn}(a)\}$

4 $z_n = \arg\min(Q'_{mn}(a))$

5 $\left\{ \begin{array}{c} \Delta Q_{m1}, \Delta Q_{m2} \\ a_{m1}, a_{m2} \\ E(a), P(a), z_n^* \end{array} \right\} = \text{mT-MM}\ \{Q_{mn}(a),\ z_n\}$

6 $R_{mn}^{(t)}(a) = \mathrm{DN}\{\Delta Q_{m1}, a_{m1}, \Delta Q_{m2}, a_{m2}, E(a), z_n^*, P(a)\}$

7 $Q_n^{(t)}(h_{mn}^{-1}a) = R_{mn}^{(t)}(a) + Q_{mn}(a)$

end

8 $t = t + 1$

end

Output: $\tilde{c}_n = \arg\min(Q_n(a))$

Algorithm 12: Proposed Decompression Operations

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

if $P1(a) \neq j$ **and** $P2_(a) \neq j$ **then**

if $a = a_{m1}$ **then**

$Out(a + z_j^*) = \Delta Q_{m1}$

else if $a = a_{m2}$ **then**

$Out(a + z_j^*) = \Delta Q_{m2}$

else

$Out(a + z_j^*) = \gamma \times \Delta Q_{m2}$

else

$Out(a + z_j^*) = E(a)$

$R_{mnj}(a + z_j^*) = \lambda \times Out(a + z_j^*)$

end for

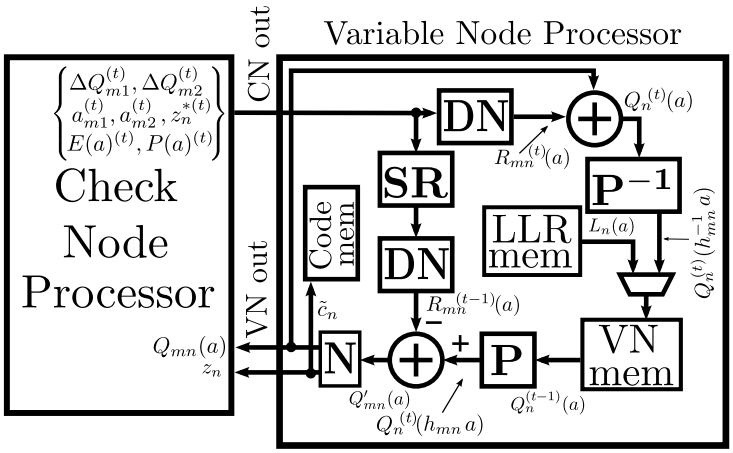


Figure 6.11: Top-level proposed decoder architecture

However, the area required in our proposal, which duplicates the logic required to implement DN, is much lower than the one of a conventional implementation of T-MM algorithm with layered schedule ([42], [50]).

To illustrate the decoder operation, in Fig. 6.13 a timing diagram is presented. It includes the input and output of the VN processor memory (VN MEM), the CN processor output (CN output) and the VN processor output (VN output). There are $d_v \times QC = M$ rows in $\mathbf{H}$ to be processed in each iteration, that is, M layers which require M clock cycles (one layer per clock cycle). On the other hand, we included *seg* pipeline stages in the CN to improve timing. After processing QC layers (the size of a circulant matrix), the pipeline must be emptied before processing the following QC layers, which requires *seg* clock cycles. So, block n in Fig. 6.13 includes the processing of layers from $QC \times (n-1)+1$ to $n \times QC$, plus *seg* clock cycles due to the pipeline.

The decoding process starts loading the channel information $L_n(a) = Q_n^{(0)}$ corresponding to the first QC rows on the VN memory ($QC \times d_c \times q$ reliabilities). Then, iteration 1 starts with the processing of block1: $Q_n^{(0)}$ is read from VN MEM and, at the same time, the VN processor starts; after *seg* clock cycles the CN processor obtains its outputs and $Q_n^{(1)}$ is saved in VN MEM. Then, this process is replicated for blocks from 2 to d_v. The same operations are repeated till the maximum number of decoding iterations ($MaxIter$) is reached. At this point, the tentative hard decoding starts to obtain the symbols $\tilde{c}_n$ and store them in the corresponding memory (Code mem in Fig. 6.11). Some control signals avoid that the permutation block $\mathbf{P}$ and the substractor in Fig. 6.11) modify $Q_n^{(MaxIter)}$ during

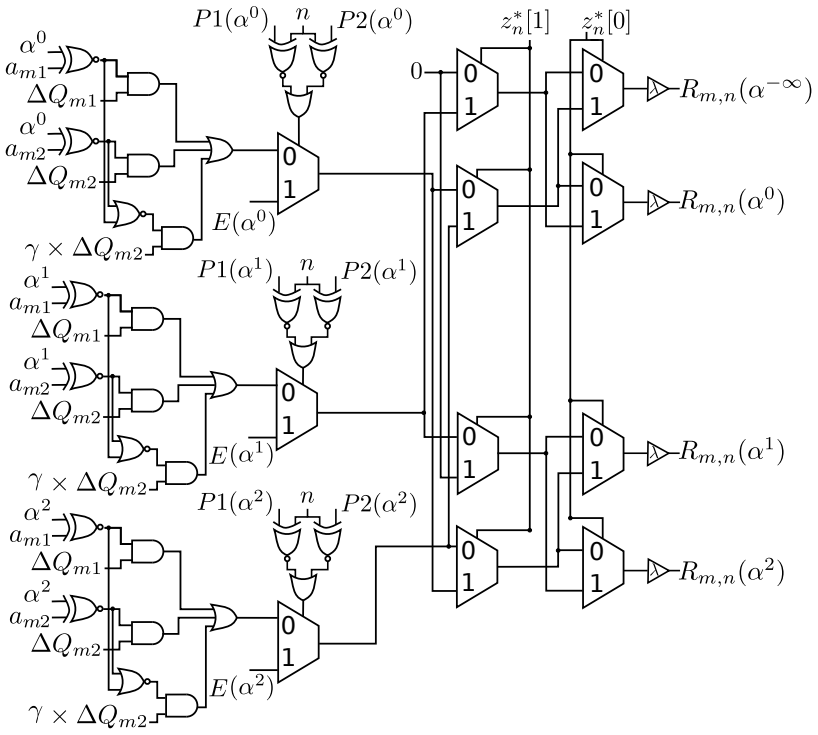


Figure 6.12: Proposed Decompression Network. Example for GF(4)

this process. Finally, the new $QC \times d_c \times q$ LLR values are stored in VN MEM while $\tilde{c}_n$ is obtained, and a new decoding process starts.

The throughput ($Thrput$) of the decoder can be obtained applying (6.5), where the $d_v \times (QC + seg) = M + seg \times d_v$ clock cycles required per iteration and the QC clock cycles required for initialization and output codeword estimation are included.

$$Thrput = \frac{f_{clk}[\text{MHz}] \cdot N \cdot p}{MaxIter \cdot (M + d_v \cdot seg) + QC} \left[\frac{\text{Mb}}{\text{s}}\right] \tag{6.5}$$

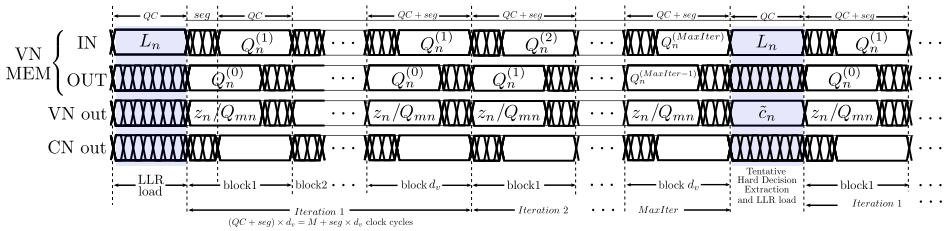


Figure 6.13: Decoder timing

6.4.3 Decoder implementation results and comparisons

The decoder architecture explained in Section 6.4.2 was implemented on a 90nm CMOS process with nine metal layers and operating conditions 1.2V and 25^oC. VHDL was used for hardware description and Cadence tools were used for synthesis and implementation.

Table 6.4 shows the implementation results for two high-rate NB-LDPC codes whose performance are analysed in Section 6.3.3: (2304,2048) NB-LDPC code over GF(16) ($d_c = 36, d_v = 4$) and (1536,1344) NB-LDPC code over GF(64) ($d_c = 24, d_v = 3$). Our purpose is to show the efficiency of our proposal over different GF(q) order and different degree distribution. The size of the circulant sub-matrices for both codes is $QC = 64$. The number of iterations in Table 6.4 is adjusted to reach a FER approximately equal to 10^{-6} for $E_b/N_0 = 4.55dB$ (see Fig. 6.7 and Fig. 6.8). Although both codes have equal number of bits per codeword (9216) and similar rate, an increase by four in the GF(q) order does not have the same impact on the number of gates of the decoder (the GF(64) NB-LDPC code has 2.85 times the number of gates of the one for the GF(16) NB-LDPC code). Additionally, the GF(64) NB-LDPC code has stronger burst error correction capability. On the other hand, our proposal reach a throughput over 1Gbps and 1.3Gbps for GF(16) and GF(64), respectively.

Table 6.5 compares the implementation of our proposal with other state-of-the-art proposals from literature for the (837,726) NB-LDPC code over GF(32). For each reference, the number of iterations is selected to achieve approximately the same performance (see Fig. 6.6) and all of them use layered schedule on their implemented decoders. For the proposals that do not use a CMOS 90nm process, the throughput showed in Table 6.5 is scaled to this technology using the equations in [41]. On the other hand, our place-and-routed results have a core occupation of 70%.

In terms of gate count, our proposal, which applies parallel processing in the CN, outperforms the other decoders from Table 6.5 except for [34]. The decoder from [34] requires 23% less gates than our approach thanks to the serial processing

Table 6.4: Implementation results for the proposed mT-MM algorithm. 90nm CMOS process

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

used in their design. This fact introduces an important reduction in the area but increases considerably the latency of the design, as can be seen in Table 6.5.

In terms of throughput, our proposal achieves the highest throughput among the solutions from literature listed in Table 6.5. This is due to the reduced set of exchanged messages between CN and VN, which reduces the wiring congestion. Our approach outperforms solutions from [42] and [50],which are the ones with higher throughput in Table 6.5, by 48% and 20 %, respectively.

Regarding efficiency, which is obtained as throughput divided by gate count, our proposal clearly outperforms the rest of decoders: its efficiency is 93.85% higher than the most efficient decoder in Table 6.5 [50].

The post-layout area required by the proposed decoder is smaller than any other solution from literature for similar CMOS technology and code parameters. The reduction in area is about 65% compared to [42], which was the solution with lower area until now.

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

To sum up, the proposed decoder based on the novel mT-MM algorithm offers important advantages compared to the state-of-the-art in both area and throughput. On the other hand, it is important to remark that the proposed mT-MM algorithm does not introduce significant performance loss for Galois field orders lower or equal to GF(32) and involves a non-negligible performance loss of about 0.07dB for GF(64), which is compensated with a great area saving and a throughput over 1.3Gbps, as can be seen in Table 6.4.

Table 6.5: Comparison of the proposed NB-LDPC layered decoder with other works from literature, for the NB-LDPC code (837,726) over GF(32)

Algorithm	SMSA [32]	T-Max-log-QSPA [30]	RMM [34]	T-MM [42]	OMO-TMM [50]	mT-MM [This Proposal]
Report	Synthesis	Post-layout	Synthesis	Post-layout	Post-layout	Post-layout
Technology	180 nm	90 nm	180 nm	90 nm	90 nm	90 nm
Quantization (w)	5 bits	7 bits	5 bits	6 bits	6 bits	6 bits
Gate Count (NAND)	1.29M	8.51M	871K	3.28M	1.79M	1.17M
f_{clk} (MHz)	200	250	200	238	250	345
Iterations	15	5	15	9	8	8
Latency (clock cycles)	12995	4460	12675	1507	1279	1343
Throughput (Mbps) 90 nm	149	223	154	660	818	1080
Efficiency 90 nm (Mbps/M-gates)	115.51	26.2	176.81	201.22	456.98	923.07
Area (mm^2)	-	46.18	-	14.75	16.10	8.97

6.5 Conclusions

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

Chapter 7

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

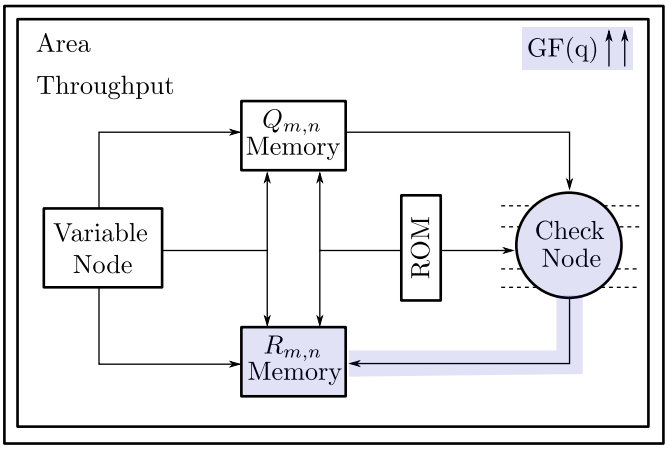


Figure 7.1: Key points of the improvements presented in this chapter.

In this chapter, the paper "Reduced-complexity Non-Binary LDPC decoder for high-order Galois fields based on Trellis Min-Max algorithm" is included. This work follows the line from the previous chapters, reducing the cardinality of the

exchanged messages between CN and VN processors. This reduction is made by means of simplifying the way how the extra column of the trellis in T-MM processor is built, keeping only the most reliable information that is outputted from the CN processor. The complexity of the CN is reduced, allowing us to implement a full decoder for high-rate GF(64) codes. Fig. 7.1 summarizes the key points that were improved in the derived paper.

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.

7.1 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 [6] 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) [28], log-SPA and max-log-SPA [29], reduce the computational load of the parity-check equations without introducing any performance loss. The recently proposed Trellis Max-Log-QPSA [30] 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) [14] and Min-Max [15] 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 [16, 17] 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 [42] improves both algorithm and architecture compared to T-EMS from [16, 17]. One Minimum Only TMM (OMO-TMM) [50] 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 [14, 15]. 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 [53] 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 [56], 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 [53] 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 [53] 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 [17, 42, 53]. 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 [17, 42, 53], 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 [53], with a performance loss of 0.1dB . In terms of throughput, the increase is about 17.5% compared to the design from [53]. 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 [53]. 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 [57].

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.

7.2 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 [20], 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 $\mathbf{G} \cdot \mathbf{H^T} = \mathbf{0}$, 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.

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 [22]. Algorithm 13 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 13.

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 13). 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.

Algorithm 13: 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: $\tilde{\mathbf{c}} = [\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_N]$

Trellis Min-Max (T-MM) algorithm [42] was proposed as a new implementation of Min-Max from [15] that allows the parallel processing of messages in the CN and reduces the complexity. Applying a message compression technique [54], 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 (7.1) to the most reliable CN input messages in delta domain [16], $m1(\eta(a))$.

$$I(a) = \min_{\eta(a) \in \ conf^*(1,2)} \left\{ \max\left(m1(\eta(a))\right) \right\} \tag{7.1}$$

$conf^*(n_r, n_c)$ [42] 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

[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

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 $I(a)$ according to (7.2).

$$E(a) = \begin{cases} m2(a) & \text{if } I(a) \rightarrow \text{one deviation} \\ m1(a) & \text{otherwise} \end{cases} \tag{7.2}$$

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 $Q_n(a)$ [42]. The decompression operations are made following (7.3).

$$R_{m,n}(a + z_n^*) = \begin{cases} I(a) & \text{if } P(a,1) \neq n \textbf{ and } P(a,2) \neq n \\ E(a) & \text{otherwise} \end{cases} \tag{7.3}$$

7.3 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 [53]. 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.

7.3.1 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 (7.3). 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 $I^*(a') = \{I^*(a'_1),\ I^*(a'_2),\ \ldots\ , I^*(a'_L)\}$, as the L most reliable values from the set $I(a)$ and $a' = \{a'_1, a'_2, \ldots, a'_L\}$ 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 7.1 includes the number of bits of each one of the sets exchanged from CN to VN processors compared to the proposal from [53], where w is the number of bits used to quantize the reliabilities.

Table 7.1: Number of bits required to be exchanged from CN to VN processor

	Number of bits	
	T-MM [53]	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 [36]. For the first code the number of bits at the CN output is 817 bits using $w = 6$ bits with the method from [53], 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 [53] 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 (7.3) at the VN.

For the symbols a' is possible to construct $L \times d_c$ values for $R^*_{m,n}(a')$ using (7.4). It is easy to see that $R^*_{m,n}(a') = R_{m,n}(a)\ \forall\ a\ \in\ a'$.

$$R^*_{m,n}(a' + z^*_n) = \begin{cases} I^*(a') & \textbf{if } P^*(a',1) \neq n \\ & \quad \textbf{and } P^*(a',2) \neq n \\ E^*(a') & \textbf{otherwise} \end{cases} \tag{7.4}$$

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 (7.5).

$$R^*_{m,n}(a'' + z^*_n) = \gamma_1 \times E^*(a'') + \gamma_2 \times I^*(a'_L) \tag{7.5}$$

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 $R_{m,n}(a'')$.

7.3.2 Performance Analysis

To show the behavior of the set $R^*_{m,n}(a)$ compared to $R_{m,n}(a)$ in an implementation of T-MM algorithm [42], 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. 7.2 we present the results for the (837,726) NB-LDPC code over GF(32) [36]. Eb/No was set to 4.3dB, $\gamma_1 = \gamma_2 = 0.5$ in (7.5) 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 (7.4) 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 (7.4) corresponds to (7.3) for $a\ \in\ a'$. For indexes 5 to 31, we observe that the approximation introduced in (7.5) 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.

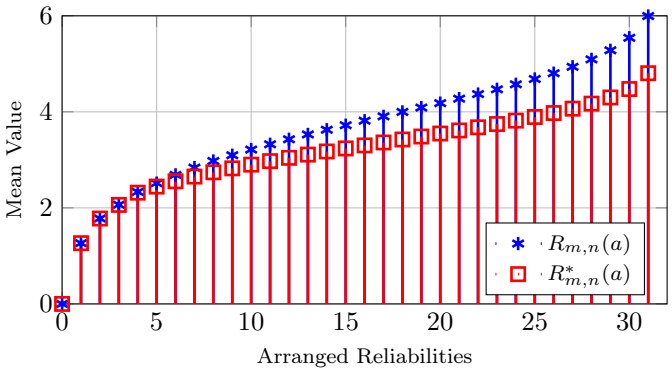


Figure 7.2: Mean values for each reliability in the set $\Delta Q(a)$. The values were arranged in the x axis. The code under test is the (837,726) NB-LDPC code over GF(32).

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) [36]. It can be seen in Fig. 7.3 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. 7.3 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. 7.4. It can be seen that the performance losses are greater than the ones from Fig. 7.3 for small L values, comparing both to the conventional T-MM algorithm [42]. This is due to the percentage of reliabilities approximated using (7.5), 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 7.4 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.

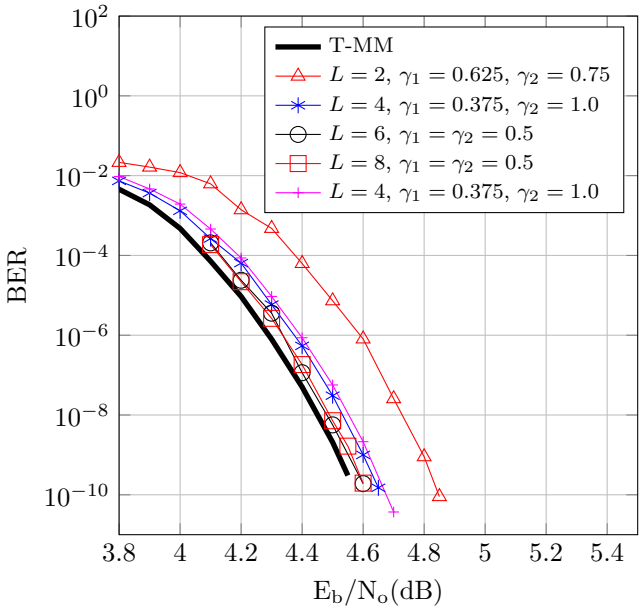


Figure 7.3: Bit Error Rate performance for our proposal varying the L parameter compared to T-MM algorithm. The code under test is the (837,726) NB-LDPC code over GF(32). 15 decoding iterations and floating point model are considered in all cases except for the last curve where 8 iteration and 6 bits are employed.

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

In Section 6.3.1 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 T-MM algorithm [42] can be tuned. On the other hand, a method to approximate the discarded messages of the set $I(a)$ was introduced using (7.5). 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 6.3.1, 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 (7.1) 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 (7.1) nor introducing any approximation. Our method takes advantage of the min-max operator involved in (7.1). 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

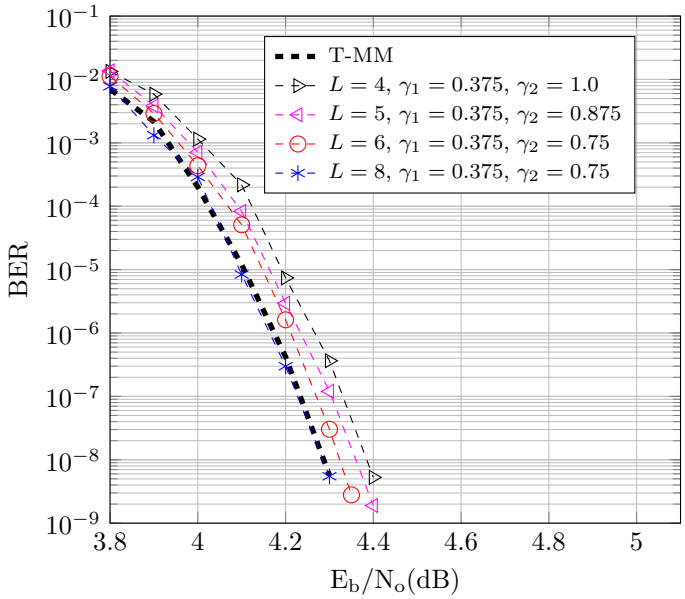


Figure 7.4: Bit Error Rate performance for our proposal with different values of L compared to T-MM algorithm. The test code is the (1536,1344) NB-LDPC code over GF(64). 15 decoding iterations and floating point model are considered for both algorithms

some rules to avoid the implementation of a complete trellis structure. In Fig. 7.5 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 (7.1). 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. 7.5 $m1_1 = 1$. $m1_1$ 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 analyzed. On the other hand, there is only one "one-deviation" path for each symbol a, so, in the example of Fig. 7.5, 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 analyzing 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 value can be obtained assigning the second minimum of $m1(a)$ ($m1_2 = 2$ in

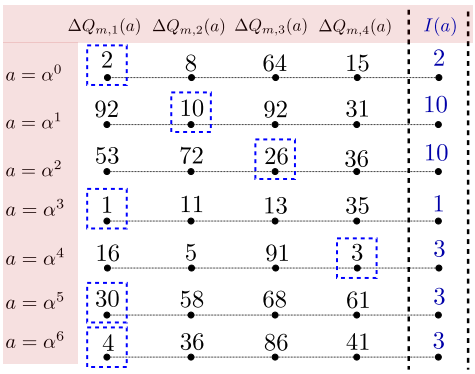


Figure 7.5: Example of the sets $\Delta Q_{m,n}(a)$ and $I(a)$ for GF(8) and $d_c = 4$

the example from Fig. 7.5), 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 (7.1).

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. 7.5, 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 7.2 with the priority given in its leftmost column. The conditions to select a reliability are listed in the rightmost column of Table 7.2. 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 [16, 42].

Following with the example in Fig. 7.5 and the priority and conditions listed in Table 7.2 for the possible candidates for $I^*(a'_4)$, the highest priority candidate (OD, $m1_3$) must be discarded 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 7.2. Thus, $I^*(a'_4) = m1_3$ and the corresponding symbol $a'_4 = a_{m1_1} + a_{m1_3} = \alpha^6$.

Table 7.2: Possible candidates for the $I^*(a'_4)$ reliability

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

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 14. Step 1 corresponds to the delta-domain transformation [35] 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.

7.4 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 7.3.3.

The top-level block diagram for the proposed CN is detailed in Fig. 7.6. 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 ($\mathbf{N} \rightarrow \mathbf{\Delta}$ block in Fig. 7.6). d_c transformation

Algorithm 14: Proposed check-node decoding algorithm

Input: $\mathbf{Q_{mn}}$

$z_n = \arg\min_{a \in \mathrm{GF}(q)} Q_{mn}(a) \ \forall \ n \in \mathcal{N}(m)$

1 $\Delta Q_{mn}(a + z_n) = Q_{mn}(a)$

2 $\beta = \sum_{j=1}^{d_c} z_{n_j} \in \mathrm{GF}(q)$

3 $[m1(a), m1_{col}(a), m2(a)] = \text{2-min}\{\Delta Q_{m,n_i}(a)\Big|_{i=1}^{d_c}\}$

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: $\begin{cases} I^*, I^*_{path}, I^*_{sym} \\ E(a) \\ z^*_n = z_n + \beta \quad \forall \ n \ \in \ \mathcal{N}(m) \end{cases}$

networks are needed in the CN, each one requires $q \times \log(q)$ w-bit MUX following the approach proposed in [39], 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 (7.4). $\mathbf{z}^*$ 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 7.3.3. Implementation is done by means of tree-based two-minimum finders, following the approach from [40]. 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. 7.6), 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. 7.7, where an example for four inputs is included. It is based on a two stage circuit: first

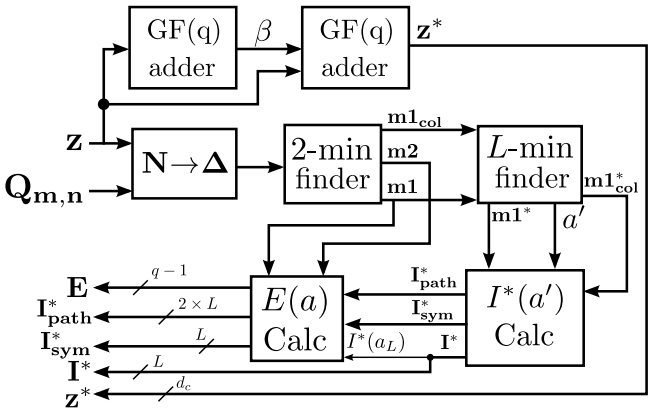


Figure 7.6: Proposed check-node block diagram

(Fig. 7.7.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. 7.7) 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. 7.7.a operating with $q-1$ inputs. Since we particularize the CN for the case where $L=4$, we require four selection networks from Fig. 7.7.b, one for each $m1_j$ value.

The implementation of the structure from Fig. 7.7.a requires $(q-1)\ \times(\frac{q-2}{2})$ w-bit comparators. The number of adders is summarized in Table 7.3 for different field orders.

Four structures as the one in Fig. 7.7.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. 7.6. 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. 7.7 to the L-min finder offers a high-speed structure that does not compromise the latency of the overall CN processor.

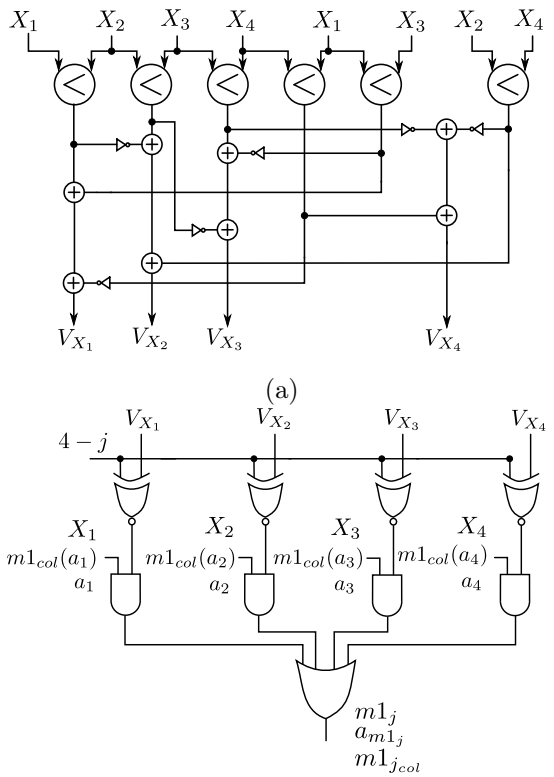


Figure 7.7: (a) First stage of the proposed L-min finder. (b) Circuit to extract the j-th minimum value. Example for four inputs.

The set $I^*(a')$ is generated using the circuit presented in Fig. 7.8 which is a direct implementation of the method explained in Section 7.3.3. 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. 7.8, 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 [42], which is unsuitable for fields higher than GF(32).

Table 7.3: Adders required to implement the circuit from Fig 7.7.a

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

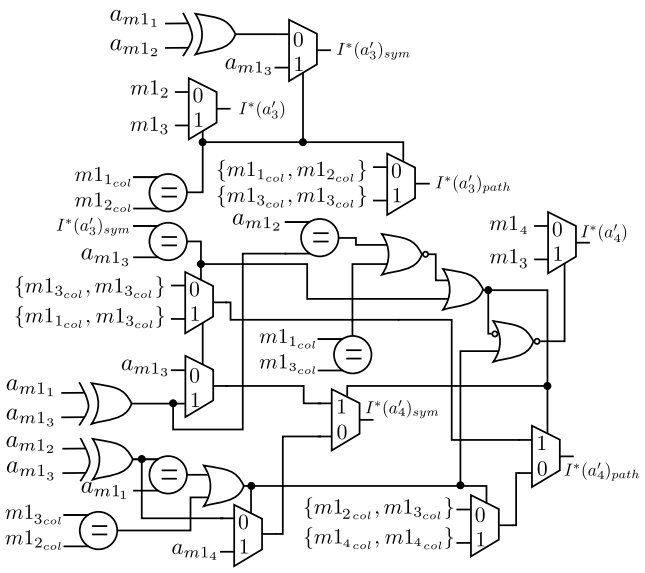


Figure 7.8: Circuit to generate the set $I^*(a')$

The reliabilities of the set $E^*(a)$ are generated using the circuit from Fig. 7.9. 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 [53], we synthesized the design using Cadence

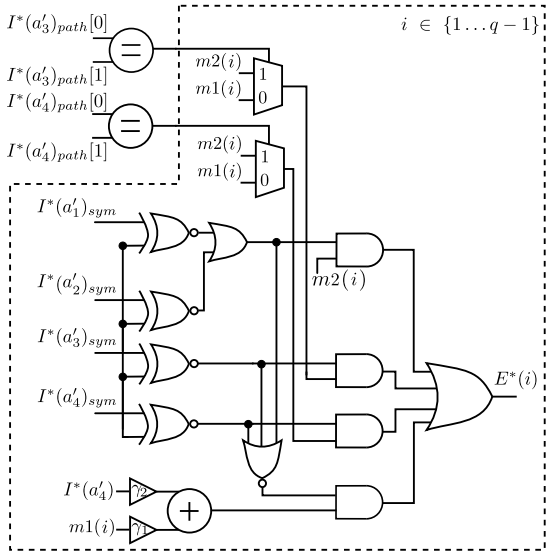


Figure 7.9: Circuit to generate the set $E^*(a')$

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 7.4 that the area saving is almost doubled for the GF(64) NB-LDPC code compared to 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 [53].

Table 7.4: Synthesis results for the proposed CN architecture

	Equivalent NAND gates		Saving
	[53]	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%

7.5 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 [54].

The decompression network generates the set $R^*_{m,n}(a)$ and implements (7.4) and (7.5) using the structures presented in Fig. 7.10. The circuit from Fig. 7.10.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. 7.10.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.

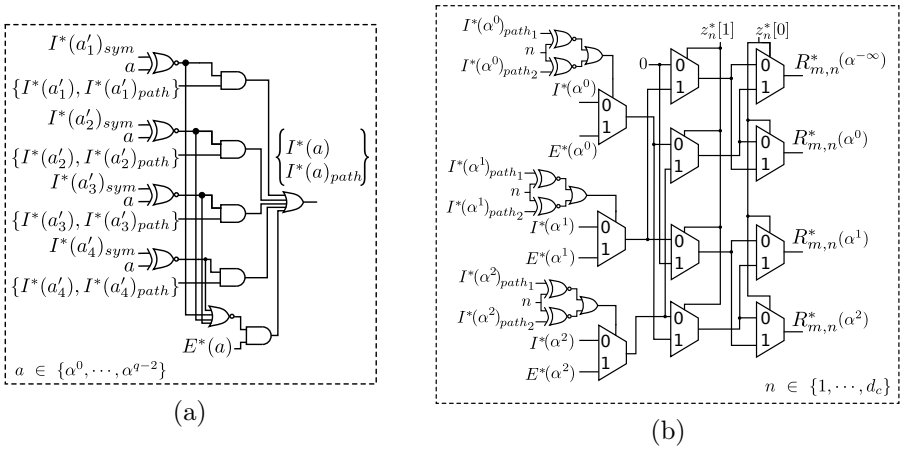


Figure 7.10: Proposed decompression network circuits. (a) Circuit to generate the set $I^*(a)$. (b) Circuit to generate the set $R^*_{m,n}(a)$, an example with GF(4)

The decoder requires $2 \times (q-1)$ circuits as the one from the left-side in Fig. 7.10, 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. 7.10 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 next one are greatly reduced compared to conventional implementations of T-MM algorithm [42], 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.

Table 7.5: Implementation results for the (1536,1344) NB-LDPC code over (GF(64) in a 90nm CMOS process.

	T-MM [42]	T-MM CNBMP [53]	OMO-TMM [50]	mT-MM [56]	[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

7.5.1 Decoder implementation results and comparisons

The complete decoder architecture based on the CN architecture explained in Section 7.4 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 [36]. 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.

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 [24]. 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 [42, 50, 53, 56] 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 7.5, 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 7.5, it can be seen that the highest reduction in the gate account is about 61% compared to the work from [42], and the lowest is 12% compared to the proposal from [53]. 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 7.5. 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 [42] in almost 2.4 times. Compared to the design from [53], our proposal outperforms it in 35%.

Table 7.6 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 7.6 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 [34] 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 [56], 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. 7.4.

In terms of efficiency, our approach is five times most efficient that the proposals from [34, 42] and almost 9 times higher than the decoder from [32]. Compared to the design from [56], our novel proposals offers 8.6% higher efficiency.

7.6 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.

Table 7.6: Comparison of the proposed NB-LDPC layered decoder with other works from literature, for the (837,726) NB-LDPC code with GF(32)

Algorithm	sMS [32]	Trellis Max-log QSPA [30]	RMM [34]	T-MM [42]	T-MM CNBMP [53]	OMO-TMM [50]	mT-MM [56]	[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 @ $E_b/N_o = 4.4$ 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

Chapter 8

Discussion and conclusions

In this chapter the results obtained during the realization of this thesis are discussed from two points of view: i) firstly, each one of the proposals developed in chapters two to seven is considered independently and their strengths and weaknesses are analyzed; ii) secondly, the impact of each proposal on the objectives of this thesis is evaluated. Furthermore, comparisons with T-EMS and the best approaches from literature are made to show the improvements introduced in this thesis. Finally, the main conclusions derived from the thesis are presented, based on the fulfillment of the objectives for this work. Moreover, future research lines to continue the works made during this thesis are included.

8.1 Summary of the main contributions

In this section the highlights of each proposal presented in the previous chapters are summarized and also their impact on the objectives proposed in this thesis from a qualitative way.

- **Chapter 2, Simplified Trellis Min-Max (T-MM) algorithm:** introduces a low-complexity version of the T-EMS algorithm from [16], where the computation of the extra-column of the Trellis and the generation of the CN output messages were greatly simplified, reducing the complexity of the CN processor in both, area and latency. The substitution of the addition operator with the maximum avoided the increase in the data-path width, which has a direct impact in the area of the processor. The proposed CN processor was included in a layered-scheduled architecture, which includes permutation networks to perform the $Q_{m,n}(a) \times h_{m,n}$ operation using GF(q) arithmetic, this improvement avoided the use of multipliers in the CN. The parallel processing of messages in the decoder and the reduction of complex-

ity in the CN allowed an increase in the achieved throughput compared to proposals using forward-backward metrics to derive the CN output messages.

Since the proposal requires the exchange of the full set of $q \times d_c$ reliability values between processors, the amount of memory resources required to implement the decoder is the same as the one from T-EMS [17]. On the other hand, T-MM has no performance loss compared to conventional implementations of Min-Max algorithm, moreover, the number of iterations to achieve the same performance is less than the ones required by the conventional implementations of Min-Max.

- The proposal included in **Chapter 3** includes two novel algorithms called **One - Minimum - Only (OMO) T-EMS (OMO-TEMS) and One - Minimum - Only T-MM (OMO-TMM)**. These algorithms simplify the computation of the two most reliable CN input messages used to calculate the extra column in trellis-based CN processors. The proposed algorithms avoided the implementation of two-minimum finders, which demand an important percentage of the CN area especially for high-rate rate codes (high CN degree d_c). The two-minimum finders were substituted by simple one-minimum finders plus an estimation of the second minimum using a combination of an scaled version of the first minimum and the discarded value in the final stage of each finder. This estimation follows a statistical behavior similar to the real second minimum, as shown in Chapter 3. Besides, the introduced performance loss is negligible compared to conventional implementations of T-MM and T-EMS, which takes a maximum of about 0.03 dB for all tested codes.

  Comparing the OMO-TMM to the T-MM proposal, the first one requires less area in the CN processor and also introduces a reduction in the latency which is translated into an increment in the overall throughput of the implemented decoder. Since this work is focused on the reduction of complexity in the CN, it needs to exchange of the full $q \times d_c$ reliability values and, thus the number of memory resources required for the decoder implementation is maintained.

- The **Compressed Non-Binary Message Passing (CNBMP)** approach introduced in **Chapters 4 and 5** corresponds to a modification at the architectural level of the simplified T-MM and T-EMS approaches. The modification includes the translation of part of the complexity of the CN to the VN processor which enables the exchange of only five elementary sets to the VN processor, which includes all the necessary information to reconstruct the entire set of messages. The reduction of the number of exchanged messages is from $q \times d_c \times w$ bits to only $2 \times (q-1) \times (w + d_c) + d_c \times p$ bits and, at the same time, the number of storage resources used to store the CN output messages from one iteration to the next one is decreased in the same quantity.

An important fact is that no performance loss is introduced compared to T-MM and T-EMS approaches, since the architectural modification was made without modifying the algorithm.

The reduction of the exchanged messages and the storage elements allows a reduction of the decoder area and an increase in throughput compared to T-MM and CNBMP approaches from previous chapters.

- The algorithm named **modified Trellis Min-Max (m-TMM)** described in **Chapter 6** follows the same line of work that the approach from chapters 4 and 5. Moreover, the reduction of the number of exchanged messages coming from CN to the VN processor is higher than the CNBMP work. m-TMM algorithm introduces a way to reduce the intrinsic set of messages to only two elements. This decrease in the cardinality of the set reduces the number of exchanged messages and the reduction in the requirements of storage elements allows an increment in the achieved throughput and a reduction in the decoder area. Another important fact is that with m-TMM we were able to obtain synthesis results for the high-rate (1536,1334) NB-LDPC code over GF(64).

  On the other hand, the performance loss compared to T-MM depends on the specific code and on the adjustment of the scaling parameter γ used to approximate the $q-3$ elements discarded in the CN processor.

- **Chapter 7** includes a **RC-TMM algorithm** which introduces a novel way to reduce even more the number of exchanged messages between CN and VN compared to the previous proposals from this manuscript. In this case not only do we reduce the cardinality of the intrinsic set, but also the number messages related to the information concerned to the deviation coordinates are decreased.

  The L most reliable messages in the CN processor are kept. Therefore, the higher the L value is, the lower the introduced performance loss is. It is important to remark that a low value of $L = 4$ is enough to keep the performance loss lower than 0.1dB for the codes under test.

  On the other hand, a new way to obtain the L most reliable values for the intrinsic set and their associated deviation coordinates is introduced. With this computation unit, an important part of the complexity in the CN is reduced, especially for codes over high-order Galois fields.

  The RC-TMM algorithm allows us to present the first post-place and route results for a high-rate NB-LDPC code over GF(64) achieving a throughput higher than 1Gbps with reasonable occupied area.

The main benefits for each work presented in this thesis are summarized in Table 8.1 in a qualitative fashion, where the acronym used to quickly identify each proposal is included.

8.2 Analysis of results

In this section the works presented in the previous chapters are quantified including how each one of them contributes to improve the key-points of NB-LDPC decoders listed in Table 8.1.

To perform the quantification of the results, three high-rate NB-LDPC codes with different degree distribution and over various Galois field orders were used. The codes are constructed using the methods introduced in [36]. The characteristics of the NB-LDPC codes are listed in table 8.2.

Despite this, most papers from literature use the (837,726) NB-LDPC code over GF(32) to perform comparisons with other works. Therefore, this was the code used in all papers from previous chapters to present the results of the implemented decoder architectures.

- First, the **CN area** measured with equivalent NAND gates is analyzed. The CN processor is the most complex block of NB-LDPC decoders. This is the main reason why some of the works presented in this manuscript are focused in the reduction of the complexity of CN, in both area and latency, without sacrificing coding gain compared to the reference algorithms (EMS, Min-Max, T-EMS).

  Fig. 8.1 includes the comparison of the CN area for the codes listed in Table 8.2. All the proposals included in this manuscript were compared to the original T-EMS algorithm and hardware architecture from [16, 17]. As can be seen, the improvements for the CN area are more visually notable passing from T-EMS to the T-MM approach, for all codes.

  Comparing each proposal to its left neighbour, it can be noted that the reduction in area is more notable for T-MM, OMO-TMM and RC-TMM approaches. This is due to the fact that these works focus on reducing the CN complexity, and in the case of RC-TMM approach the reduction in area is higher for high-order Galois fields.

- The next step consists on evaluate the **storage resources** required to implement each one of the decoders proposed in this thesis. Although the CN is the most complex part of NB-LDPC decoders, most of the area of layered-schedule based decoders corresponds to storage elements such as memories for the channel information and VN messages and, on the other hand, shift-registers to store the CN output messages from one iteration to be used in

Table 8.1: Main benefits of each proposal presented in this thesis

Proposal	**Acronym**	**CN area**	**Storage Resources**	**Decoder Area**	**Reduction Message Exchange**	**Latency**	**Throughput**	**High-Order GF**
Chapter 2: Simplified Trellis Min-Max Decoder Architecture for NB-LDPC Codes	T-MM	✓				✓	✓	
Chapter 3: One Minimum Only Trellis Decoder for NB-LDPC Codes	OMO-TMM	✓				✓	✓	
Chapter 4: Reduction of complexity for NB-LDPC decoders with compressed messages and Chapter 5: A 630 Mbps Non-Binary LDPC Decoder for FPGA	CNBMP		✓	✓	✓		✓	
Chapter 6: High-performance NB-LDPC decoder with reduction of message exchange	m-TMM		✓	✓	✓		✓	
Chapter 7: Reduced-complexity NB-LDPC decoder for high-order Galois fields based on T-MM algorithm	RC-TMM	✓	✓		✓	✓		✓

Table 8.2: NB-LDPC codes used to perform comparisons between the proposals included in the manuscript

NB-LDPC code	**Degree Distribution (d_c,d_v)**	**rate**
(2304,2048) over GF(16)	(36,4)	0.889
(837,726) over GF(32)	(26,4)	0.863
(1536,1344) over GF(64)	(24,3)	0.875

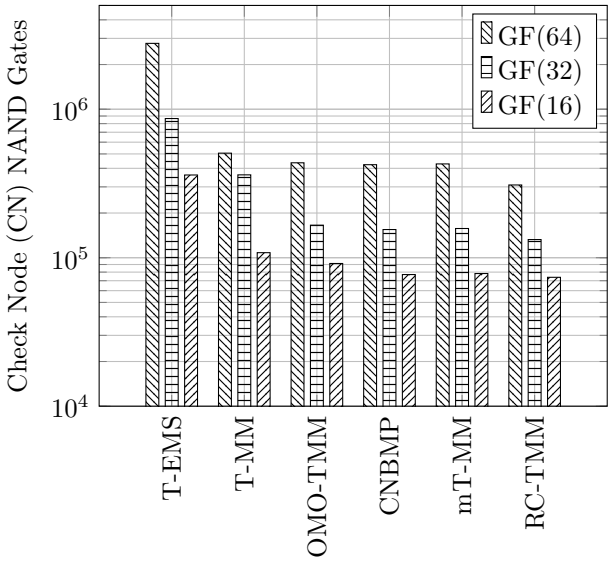


Figure 8.1: Graphical comparison of the CN area for all the proposals included in this thesis

the next-one. Then, as can be seen on Table 8.1, some of the proposals presented in this manuscript deal with the reduction of storage resources in NB-LDPC decoders. Fig. 8.2 presents a bar-graph for the three high-rate codes under test where the number of memory bits to implement each decoder are counted. The first group of bars includes results for the T-EMS based decoder for comparison purposes. As can be seen, since T-MM and OMO-TMM approaches deal with the CN complexity but work with the full set of $q \times d_c$ reliability values, the number of memory bits required is equal for all of them. On the other hand, the last three works (CNBMP, mT-MM and RC-TMM) introduce methods to reduce the number of exchanged messages between CN and VN processor, which directly impact on a reduction of the number of bits stored on shift-registers from one iteration to the next-one.

The reduction from the two initial works to the CNBMP approach is due to the fact that instead of exchanging $q \times d_c$ messages, CNBMP exchanges only four elementary sets of $q - 1$ elements each one, which reduces the strong dependency on the CN degree (d_c) and reduces the area for high-rate codes.

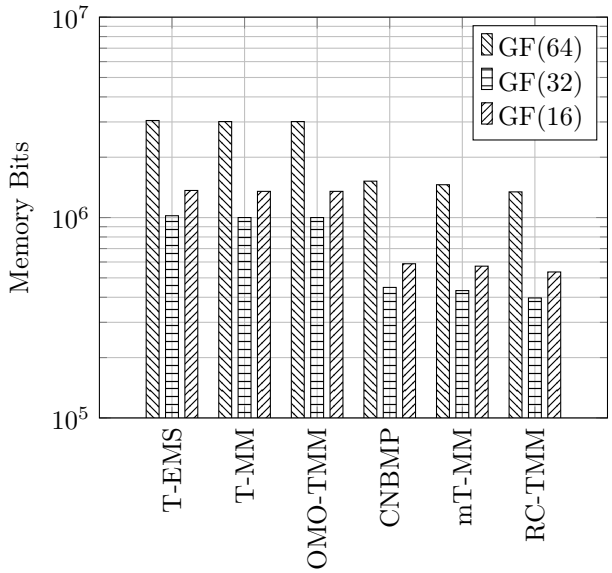


Figure 8.2: Graphical comparison of the memory bits required for all the proposals included in this thesis

Although the reduction in the memory bits cannot be easily seen on Fig. 8.2, it is almost 4% passing from CNBMP to m-TMM approach and 7% from m-TMM to RC-TMM work for the GF(16) code under test. For the GF(64) code the reduction is about 5% and 10%, respectively. This apparently low decreasing in the number of stored bits is due to the fact that the reduction is made on the four elementary sets introduced in the CNBMP paper.

Another important fact is that the number of memory bits is greater for the GF(16) code compared to the GF(32) one. This is due to the GF(16) code, which is more than twice longer (number of bits) than the GF(32) code and the channel information and VN memories store all the code bits multiplied by the number of field elements.

- Next we analyze the **decoder area**, measured in equivalent NAND gates, for each proposal implementing the three codes under test. In this case, the bar graph presented in Fig. 8.3 is the addition of graphs from Fig. 8.1 and 8.2 plus the equivalent NAND gates of the logic required to process the messages at the VN processor for each proposal.

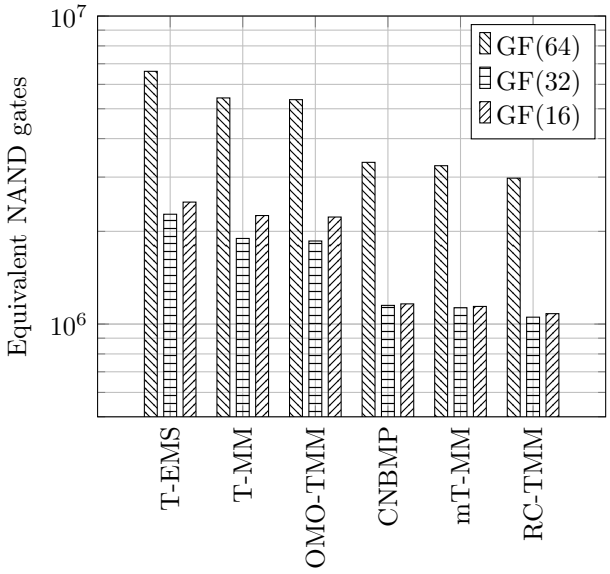


Figure 8.3: Graphical comparison of the decoder area for all the proposals included in this thesis

It can be seen in Fig. 8.3 how each proposal outperforms the previous one. The reduction in the decoder area is most notable for the three last works thanks to the decrease of memory resources which conforms the major part of the decoder area. Comparing the T-EMS proposal to the RC-TMM approach, the reduction in area is 56%, 53% and 55% for the GF(16), GF(32) and GF(64) codes, respectively. It is important to remark that the area occupied by the implementation of the RC-TMM algorithm for the GF(64) code is only 19% higher compared to the area occupied by the T-EMS proposal for the GF(16) code, having both codes the same bit-length.

- Next, the evolution in the achieved **throughput** is included, starting from T-EMS to the last work (RC-TMM). In this case, Fig. 8.4 only includes the results for the implemented decoders included in the papers from chapter 2 to 7. It can be noted the constant tendency of increasing the throughput for the three codes under test. In all cases, we achieved a throughput higher than 1 Gbps for all codes, for example, the GF(16) code passed from 957 Mbps for the CNBMP approach to 1047 Mbps for the m-TMM proposal. The GF(32) code passed from 484 Mbps with T-EMS to 1080 Mbps (m-TMM decoder).

  The GF(64) code exhibits a great increment in the achieved throughput from the m-TMM to the RC-TMM approach. This occurs due to the fact that the RC-TMM approach was focused in high-order Galois fields, where the

simplifications made to compute the most reliable values of the extra column in the CN processor become more important. For this code, the throughput increased from 874 Mbps (T-MM, OMO-TMM) to 1259 Mbps (RC-TMM).

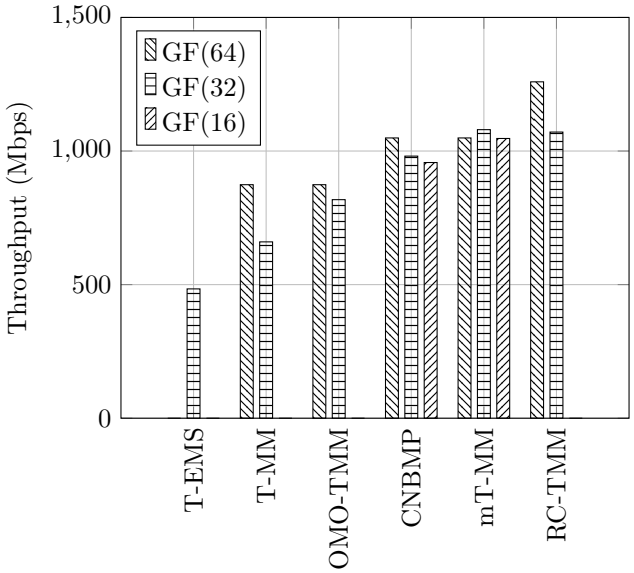


Figure 8.4: Graphical comparison of the achieved throughput for all the proposals included in this thesis

It is important to remark that all the results are taken from post place and route reports except the ones for the GF(64) code (T-MM, OMO-TMM, CNBMP and m-TMM), where the post place and route results were estimated based on a reduction of the clock frequency passing from synthesis to a routed design. To be fair, the same percentage of reduction of the RC-TMM approach was used, where the reports are actually from a placed and routed design. This situation was clearly explained in the paper from Chapter 7.

- Since all the works from this thesis use parallel processing of messages, the **latency** is kept low compared to other proposals from literature using forward-backward to derive the CN output messages. It is well-known that those approaches require long-latency networks to implement their decoding algorithms. Table 8.3 include the achieved latency for all the proposals from this thesis, compared to the one from T-EMS [17].

  Passing from T-EMS to the T-MM proposal, the latency was reduced in 30% thanks to the reduction of complexity in the computation of the CN output messages and the extra column compared to the T-EMS based architecture.

Table 8.3: Latency of all the proposals for the (837,726) NB-LDPC code over GF(32)

Proposal	**Latency (clock cycles)**
T-EMS	2160
T-MM	1507
OMO-TMM	1279
CNBMP	1280
m-TMM	1343
RC-TMM	1535

An additional 15% of reduction in the latency was achieved passing from T-MM to the OMO-TMM approach. The reduction is due to the shortening in the critical-path thanks to the use of single-minimum finders instead of two-minimum finders.

The latency of the decoders from m-TMM and RC-TMM approaches was increased compared to OMO-TMM and CNBMP proposals with the aim of increase the clock frequency and hence the achieved throughput. Despite this, if required, the latency can be reduced without modifying the decoder behaviour.

- The last parameter used to evaluate the results of this thesis is the **efficiency**, calculated as the ratio between the achieved throughput and the equivalent Million of NAND gates of the entire decoder (Mbps/Million NAND Gates). This parameter has been used in the literature as a quantity that measures how many Mbps are achieved per Million of NAND gates required to implement the decoder. For example, serial decoders require many less NAND gates that a parallel one, but the achieved throughput could be of the same magnitude order for both designs. This fact can be observed with the efficiency parameter where the serial design from the example will exhibit higher efficiency than the parallel one.

  Fig. 8.5 includes a bar graph for the efficiency parameter of all the proposals included in this manuscript. It can be seen that the efficiency of T-EMS approach for the GF(32) code was outperformed in all the approaches for the same code until reaching almost five times the efficiency with the RC-TMM proposal. For the GF(64) code the increment was almost three times passing from 161 with the T-MM based decoder to 422 with the RC-TMM approach. It is important to remark how the efficiency is doubled comparing the T-EMS decoder for the GF(32) code to the RC-TMM approach for the

GF(64) code, even taking into account that the GF(64) code is twice longer than the GF(32) one and the field size is doubled too.

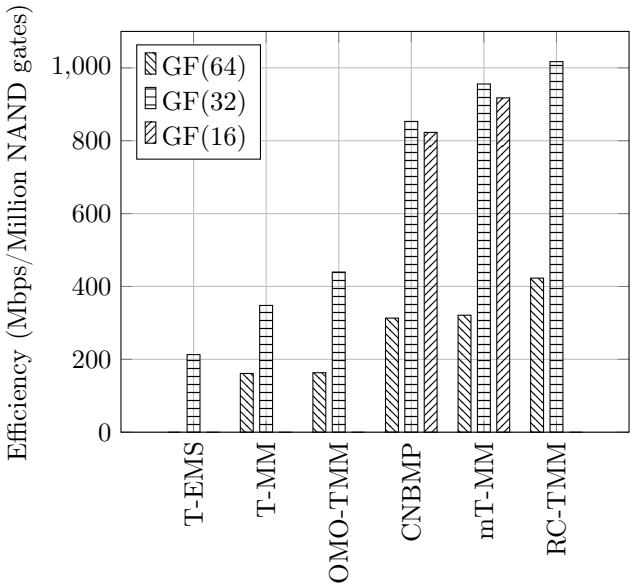


Figure 8.5: Graphical comparison of the efficiency parameter for all the proposals included in this thesis

For all the parameters analyzed in this section it has be shown that they were improved from one work to the next one for the three high-rate codes under test.

8.3 Comparison with other works from literature

In this section comparisons between the works included in this manuscript and other works from literature are made. Since the (837,726) NB-LDPC code over GF(32) is usually used by hardware designers to compare their proposals and most of the works in literature use this code to present their implementation results, we decided to use this code to show the comparisons in this section as was made in the papers from the previous chapters.

In order to simplify comparisons, only two of our works, CNBMP and RC-TMM are included. The first one (CNBMP) is selected since it represents an implementation of the T-MM algorithm without any performance loss, compared to the original algorithm. On the other hand, the RC-TMM approach is selected since it represents our state-of-the-art proposal with lower area and offering higher throughput for high-order fields, as was seen in the previous section. Besides, RC-

Table 8.4: Comparison of the works from this manuscript with other proposals from literature, for the (837,726) NB-LDPC code with GF(32)

Algorithm	sMS [32]	Trellis Max-log QSPA [30]	Min-Max [33]	RMM [34]	CNBMP [53]	RC-TMM [58]
Report	Synthesis	Post-layout	Synthesis	Synthesis	Post-layout	Post-layout
Technology	180 nm	90 nm	130 nm	180 nm	90 nm	90 nm
Quantization (w)	5 bits	7 bits	5 bits	5 bits	6 bits	6 bits
Gate Count (NAND)	1.29M	8.51M	2.1M	871K	1.25M	1.06M
f_{clk} (MHz)	200	250	500	200	300	393
Iterations	15	5	15	15	8	8
FER @ $E_b/N_o = 4.4$ dB	2×10^{-4}	5×10^{-5}	5×10^{-5}	9×10^{-5}	1×10^{-4}	1×10^{-4}
Throughput (Mbps)	64	223	64	66	981	1071
Throughput (Mbps) 90 nm	149	223	107	154	981	1071
Efficiency (Mbps / Million NAND gates)	115.5	26.2	50.9	176.8	784.8	1010.4
Area (mm^2)	-	46.18	-	-	10.6	9.80

TMM algorithm introduces a negligible performance loss for the GF(32) code and only 0.1 dB for the GF(64) one.

Table 8.4 includes the implementation results for the state-of-the-art works from literature implemented for the test code over GF(32). The works included are the ones with remarkable results implementing different soft-decision algorithms. As explained before, CNBMP and RC-TMM proposals are included in the right-most columns of Table 8.4 to perform the comparisons.

As can be seen, apart from our proposals, only the work from [30] includes post-layout results for their design, therefore, the throughput report for the rest of works listed in Table 8.4 ([32, 23, 34]) are overestimated compared to post-layout reports.

Since the works from Table 8.4 are implemented for different CMOS technologies, the technology was scaled to show results over a 90 nm CMOS process using first-order approximations [41] based on the transistor gate-length and its relationship with the transmission delay for the different processes. To this end, the scaling factors used to derive the comparisons shown in Table 8.4 are 1.66 and 2.33 for 130/90 nm and 180/90 nm scaling, respectively. Note that, different algorithms were compared under the same performance, so each one has a different number of iterations.

In terms of gate count, the RMM decoder [34] is the one that requires less NAND gates outperforming the RC-TMM proposal by 22%. This fact is due to the serial processing for the RMM approach in the CN which reduces considerably the required area. ON the other hand, the serial processing reduces considerably the throughput in almost 7 times compared to the RC-TMM proposal. Note that despite the sMS [32] and Min-Max [33] approaches use serial-based processors, they require more equivalent NAND gates than our CNBMP and RC-TMM proposals which apply parallel processing. The complexity of NB-LDPC decoders which applies parallel processing of messages was reduced until reaching the same order of magnitude of decoders which uses serial processing of messages. The work from [30] requires more than 8 times the number of gates compared to the RC-TMM approach. This fact is compensated with the high coding gain achieved thanks to the use of QSPA algorithm.

Compared to the proposals from other authors in terms of the normalized throughput (90 nm), it can be observed that the approach from [30] achieves the highest one thanks to the lower number of iterations required to achieve a desired coding gain. Despite this, our works outperforms its throughput results 4.39 times for the CNBMP approach and 4.80 times for the RC-TMM proposal. The other proposals [32, 33, 34] are limited because of the serial processing of messages which increases the latency reducing the achieved throughput.

Only the work from [30] includes a post-layout report. Their decoder occupies more than six times the area compared to our proposals with much lower throughput.

In general, our works outperforms the best proposals from literature for the same code taking into account different parameters of decoders. The general conclusions for all the works included in this thesis are devised in the next chapter.

8.4 Conclusions

The main soft-decision decoding algorithms for NB-LDPC codes were analyzed in Chapter 1, concluding that the Trellis Extended Min-Sum (T-EMS) algorithm has strong potential to achieve high decoding speeds thanks to the parallel processing of messages in the CN processor. This was the main reason to select T-EMS algorithm as the starting point to develop the contributions included in this thesis.

The main objective of this thesis was the development of low-complexity algorithms and architectures for VLSI implementation of high-speed SD NB-LDPC decoders suitable for high-rate codes over high-order Galois fields. Three specific objectives were established to achieve this main objective. Below, the conclusions related with each one of the specific objectives are exposed.

8.4.1 Objective 1: reduction of area and latency of Check Node (CN) processors

To accomplish this objective, three proposals were presented in this thesis. They dealt with the reduction of the CN processor area. The first one, T-MM, (Chapter 2) reduced the complexity on how the CN output messages are computed. On the other hand, the substitution of the addition operator by the maximum, avoided the data-path growing in the processor. All these features of the T-MM based CN processor reduced the CN area in 58% compared to the T-EMS based processor.

The second proposal, OMO-TMM, (Chapter 3) further introduced simplifications in the CN processor. OMO-TMM eliminated the use of two-minimum finders to search for the two most reliable messages per Galois field symbol. A novel method was presented, which introduces the use of single-minimum finders plus an estimator of the second most reliable messages. The reduction in area was bigger for high-rate codes where the CN degree (d_c) is higher. With OMO-TMM a reduction in area of 54% of the CN was achieved, compared to the previous T-MM approach. An important fact is that the performance loss was negligible when compared to T-MM, being about 0.02dB.

The third proposal that dealt with the reduction of the CN area is the RC-TMM (Chapter 7). Once the T-MM and OMO-TMM reduced the complexity in the computation of CN output information and the calculation of the two most reliable messages, respectively, the extra-column processor was identified as the most complex block in the CN, specially for high-order fields, being the complexity in the order of $\mathcal{O}(\frac{q^2}{2})$ in conventional implementations of T-EMS and T-MM approaches. Therefore, this work introduced a novel and simplified way to only compute the L most reliable elements of the extra-column. The complexity of the extra-column processor was reduced to be $\mathcal{O}(L)$ when L takes small values lower than 6. This proposal allows an area reduction of 15% compared to the OMO-TMM. As can

be seen, the three listed works outperform the previous one. Furthermore, the CN area reduction from the T-EMS version previous to this thesis to the RC-TMM is about 84%.

8.4.2 Objective 2: reduction of the number of messages exchanged between processors in NB-LDPC decoders

One of the identified bottlenecks for T-EMS was the high density of wires that exchange information between CN and VN processors, due to the transportation of the full set of $q \times d_c$ messages. The high number of wires connecting the CN and the VN processor causes wire congestion and reduces the achieved throughput due to the extra area required to route all the wires. Additionally, this wiring makes the processor to be placed away from each other, which increases the critical path. On the other hand, the number of messages passing from the CN to the VN is directly related to the quantity of memory resources required to store these messages from one iteration to the next-one in layered scheduled decoders. Therefore, a reduction in the number of exchanged messages bring two benefits: i) an increase in the clock frequency and the achieved throughput and, ii) a reduction in memory resources and hence the decoder area.

Three works included in this thesis dealt with the reduction of the number of exchanged messages between CN and VN processors: CNBMP (Chapter 4 and 5), m-TMM (Chapter 6) and RC-TMM (Chapter 7). The CNBMP approach redefines the CN output messages from the T-MM algorithm. They are organized in five elementary sets, these compressed messages are sent to the VN processor where the entire set of $q \times d_c$ messages is reconstructed. Since there is no information loss, no performance loss was introduced. The cardinality of the compressed sets is independent of the CN degree, being suitable for high-rate codes. Compared to the conventional implementation of T-EMS and T-MM decoders, the reduction in the number of exchanged messages was 83% for the GF(32) code under test. The same percentage of reduction was achieved in the amount of memory elements required to store the CN output messages from one iteration to the next one. The global reduction in area was about 40% and the increase in throughput was 32% for the same code.

The m-TMM decoder used the five elementary sets defined in the CNBMP approach (Chapters 4 and 5) as starting point to propose a reduction in the cardinality of one of the elementary sets from q to only two elements. The dismissed $q - 2$ less reliable messages were approximated in the VN to reconstruct the full set of $q \times d_c$ reliability values. The introduced performance loss due to the approximation of messages is less than 0.05dB for the codes under test. Compared to the CNBMP decoder, the reduction in the number of exchanged messages and hence in the storage resources is 23%. These achievements introduce an extra 10% in

the increase of throughput and 3% of area decreasing, compared to the CNBMP approach.

Following the same line of work, the RC-TMM approach introduced a reduction in the cardinality of three of the five elementary sets defined in chapters 4 and 5. Specifically, a method to keep only the L most reliable messages of three sets was introduced. Besides, a novel method to approximate the reliability values in the VN was presented. Furthermore, thanks to the reduction in size of the three sets in the CN, a simplified method to compute the L most reliable messages was presented. The performance loss due to the approximation of messages in the VN is less than 0.07dB for the GF(32) code and 0.1dB for the GF(64) code. On the other hand, the percentage of reduction in the number of exchanged messages was 38% compared to its previous proposal (m-TMM) and in addition, for the GF(32) code the area was decreased in 7%.

Compared to the T-EMS approach, in the RC-TMM proposal the number of exchanged messages from CN to the VN was reduced in 92% for the GF(32) code under test.

8.4.3 Objective 3: implementation of high-performance decoders for Galois fields larger than 32

All the progresses made with the proposals included in this manuscript in terms of area and throughput allowed us to present the first post-place and route reports for a high-rate code over high-order Galois fields. For example, the (1536,1344) NB-LDPC code over GF(64) achieved a throughput of 1259Mbps and required an area of 28.90 mm^2.

Keeping the coding gain near to the one of T-EMS algorithm was one of the constraints imposed in this thesis. Therefore, we take care that our proposals introduced less than 0.1 dB of performance loss for high-rate codes. For example, for the GF(32) code under test, the RC-TMM proposal, requiring only a seventh part of the CN area required by T-EMS, introduces 0.07 dB of performance loss compared to this algorithm, and 0.1 dB for the GF(64) code tested.

An important quality factor when designing hardware architectures to decode NB-LDPC codes is how efficiently they use the area in order to achieve high decoding speeds. The efficiency, measured as the relation between the throughput (Mbps) and the number of equivalent NAND gates required by the design (Million NAND gates), is commonly used in the literature by hardware designers to compare their proposals with others. All the decoder architectures included in this thesis apply parallel processing of messages, which allow us to reduce the area and hence the latency, which is translated into an increase in throughput. Specifically, the efficiency of the T-MM approach proposed in this thesis is 38% higher than the

previous T-EMS implementations. An additional 21% of increasing was achieved passing from T-MM to OMO-TMM implementation. Both proposals (T-MM and OMO-TMM) reduce the complexity of the CN and, at the same time, the latency was decreased, so both area and throughput were improved.

When compared the OMO-TMM to the CNBMP proposal, the increase in the efficiency was 48%. Despite the increment in throughput of the CNBMP decoder achieved with the reduction in the wiring between processors, the great increment in the efficiency is due to the reduction in area thanks to the decrease in the storage elements used in the decoder implementation.

The efficiency was increased by 11% when passing from CNBMP to m-TMM. In this case, the achieved increment is due to both improvements (area and throughput) since the m-TMM reduces the number of wires between processors and the number of storage elements of the decoder. Finally, 6% of improvement was achieved passing from the m-TMM to the RC-TMM approach. The reason of this increment was the reduction in the CN complexity, specifically the computation unit used to calculate the extra-column of the trellis.

Comparing the implementation of the original T-EMS algorithm to the RC-TMM, the efficiency is almost five times higher for the latter.

8.4.4 Final Comments

To sum up this section we summarize the achievements of this thesis:

i) the computational load of the main processor was simplified, which reduces the arithmetic complexity;

ii) the dependency of the check node processor with his degree (d_c) was reduced by half;

iii) a new definition of sets to allow the compression of the messages exchanged between processors was introduced, which reduces the wiring of the hardware implementations; and

iv) a new decoding algorithm that avoids the dependency with GF(q) of the NB-LDPC decoders, with negligible performance loss, was proposed.

All these contributions allow us to increase the efficiency of the derived architectures at least four times and even more important, the new decoders can be applied to high-rate codes of medium codeword length over Galois Fields larger than GF(32), which will improve the integration of NB-LDPC codes with modulations of higher order (to the best knowledge of the authors was not feasible to implement in hardware with the existing proposals in literature).

8.5 Future Research Lines

Although the considerable improvements obtained with the proposals included in this manuscript, NB-LDPC decoders are far from be integrated in some real systems with high demanding constraints. Architectures of tens of Gbps are required for high-speed communication systems and standards (such as the 100Gbps links), in conjunction with error-floor regions lower than a BER of 10^{-15} with a waterfall performance close to the Shannon's limit. In order to meet all these requirements without decreasing the code rate, a lot of research needs to be done to combine both VLSI and coding theory. Considering the state of the art at the moment of the publication of this thesis, we consider that the most promising research lines that can contribute to accomplish with all these objectives are the following ones:

- One of the main constraint for optical transport systems is to have a FER performance without error-floor until BER = 10^{-15} is a main constraint for optical transport systems. Therefore, we propose to study the proposed algorithms from this manuscript for BER values lower than 10^{-10} where the error-floor effect could appear degrading the performance in the decoder. When this effect occur, a solution could be the use of concatenated codes. Specifically, the use of a concatenated NB-LDPC + RS codes helps to correct the possible residual errors resulting from the use of a simple NB-LDPC decoding algorithm.

- Spatially-Coupled LDPC (SC-LDPC) codes are strong candidates for future optical transport systems due to their close performance to the Shannon channel capability and their potentially high-speed decoding using a slicing windows over the photograph of the code. It has been shown in the literature that SC-LDPC codes exhibit relatively high error-floor, limiting their application in optical systems where very low error-floor is required. Since NB-LDPC codes have better performance than its binary counterpart for high SNR, the use of the low-complexity algorithms presented in this thesis could be studied to decode SC-LDPC codes in its non-binary form.

- In order to improve the coding gain in optical systems joint coding and modulation schemes are used. This systems are called coded-modulation (CM) and are widely studied for wireless and optical systems. NB-LDPC codes could be used in CM schemes with the aim of reducing the complexity in the coding-modulation process since the Galois field symbols are directly mapped to modulated symbols. Since the proposed algorithms from this thesis exhibit low complexity for high-order fields and high-rate codes, they could be considered to be used in CM schemes.

Bibliography

[1] B. Pierce, "Five key steps to high-speed nand flash performance and reliability," in *Proc. of Flash Memory Summit, 2010*, Aug 2010.

[2] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar 1997.

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

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

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

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

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

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

[9] *100G Forward Error Correction White Paper, OIF.* OIF-FEC-100G-01.0, May 2010.

[10] 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.

[11] F. Sala, K. Schouhamer Immink, and L. Dolecek, "Error control schemes for modern flash memories: Solutions for flash deficiencies," *Consumer Electronics Magazine, IEEE*, vol. 4, no. 1, pp. 66–73, Jan 2015.

[12] G. Liva, E. Paolini, T. De-Cola, and M. Chiani, "Codes on high-order fields for the CCSDS next generation uplink," in *Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC), 2012 6th*, Sept 2012, pp. 44–48.

[13] L. Costantini, B. Matuz, G. Liva, E. Paolini, and M. Chiani, "Non-binary protograph low-density parity-check codes for space communications," *Int. J. Satell. Commun. Network*, vol. 30, pp. 43–51, 2012.

[14] 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.

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

[16] 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.

[17] 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.

[18] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[19] D. J. MacKay and R. Neal, "Good codes based on very sparse matrices," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, C. Boyd, Ed., vol. 1025. Springer Berlin Heidelberg, 1995, pp. 100–111.

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

[21] C. Chavet and P. Coussy, *Advanced Hardware Design for Error Correcting Codes*, Springer, Ed., 2014.

[22] 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.

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

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

[25] F. Garcia-Herrero, E. Li, D. Declercq, and J. Valls, "Multiple-Vote Symbol-Flipping Decoder for Nonbinary LDPC Codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 11, pp. 2256–2267, Nov 2014.

[26] F. Garcia-Herrero, D. Declercq, and J. Valls, "A symbol flipping decoder for NB-LDPC relying on multiple votes," in *8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2014*, Aug 2014, pp. 203–207.

[27] W. Ryan and S. Lin, *Channel Codes: Classical and Modern.* Cambridge University Press, 2009.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

[37] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular ($2,d_c$)-LDPC codes over GF(q) using their binary images," *IEEE Transactions on Communications*, vol. 56, no. 10, pp. 1626–1635, October 2008.

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

[39] 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.

[40] 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.

[41] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital integrated circuits-A design perspective*, 2nd ed. Prentice Hall, 2004.

[42] 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.

[43] X. Zhang and F. Cai, "Reduced-latency scheduling scheme for min-max nonbinary LDPC decoding," in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2010*, Dec 2010, pp. 414–417.

[44] ——, "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.

[45] M. Punekar and M. Flanagan, "Trellis-based check node processing for low-complexity nonbinary LP decoding," in *IEEE International Symposium on Information Theory Proceedings (ISIT), 2011*, July 2011, pp. 1653–1657.

[46] Y. S. Park, Y. Tao, and Z. Zhang, "A 1.15Gb/s fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013*, Feb 2013, pp. 422–423.

[47] C.-C. Cheng, J.-D. Yang, H.-C. Lee, C.-H. Yang, and Y.-L. Ueng, "A Fully Parallel LDPC Decoder Architecture Using Probabilistic Min-Sum Algorithm for High-Throughput Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 9, pp. 2738–2746, Sept 2014.

[48] 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.

[49] 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.

[50] 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.

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

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

[53] 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.

[54] 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.

[55] J. Sha, Z. Wang, M. Gao, and L. Li, "Multi-Gb/s LDPC Code Design and Implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 2, pp. 262–268, Feb 2009.

[56] 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.

[57] 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.

[58] J. O. Lacruz, F. Garcia-Herrero, M. J. Canet, and J. Valls, "Reduced-Complexity Nonbinary LDPC Decoder for High-Order Galois Fields Based on Trellis Min-Max Algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–11, 2016.