Noise-against-Noise Decoders : Low Precision Iterative Decoders

Franklin Rafael Cochachin Henostroza

THÈSE DE DOCTORAT DE

L'UNIVERSITE BRETAGNE SUD
COMUE **U**NIVERSITÉ **B**RETAGNE **L**OIRE

Ecole Doctorale N°601
Mathèmatique et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Télécommunications*

Par

« Franklin Rafael COCHACHIN HENOSTROZA »

« Noise-Against-Noise Decoder » « Low Precision Iterative Decoders »

« Décodeur bruit contre bruit » « Décodeurs itératifs de faible précision »

Thèse présentée et soutenue à Lorient , le 02/05/2019
Unité de recherche : Lab-STICC
Thèse N° : 527

Rapporteurs avant soutenance :

Professeur Charly Poulliat, ENSEEIHT
Professeur Norbert Wehn, University of Kaiserslautern

Composition du jury :

Président :	Professeur Christophe Jego, INP/ENSEIRB-MATMECA
Examinateurs :	Professeur Christophe Jego, INP/ENSEIRB-MATMECA
	Docteur Savin Valentin, Ingénieur-Chercheur, CEA Leti
	Docteur Elsa Dupraz, Maitre de Conférences, IMT ATLANTIQUE
Dir. de thèse :	Professeur Emmanuel Boutillon, Université de Bretagne Sud, Lab-STICC
Co-dir. de thèse :	
	Professeur David Declercq, Chief Technical Officer, Codelucida Inc.
	Docteur Lounis Kessal, Maitre de conférences HdR, Université de Cergy-Pontoise, ENSEA
Invité(s) :	
	Docteur Fakhreddine Ghaffari, Maitre de conférences, ENSEA
	Docteur David Gnaëdig, CTO, Turboconcept Brest

To my dear parents.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisors, Prof. David Declercq, Prof. Emmanuel Boutillon, and Dr. Lounis Kessal for their support and guidance during all my work. I am honored that all of you have been my supervisors.

During my PhD study, I had the opportunity to visit the Ecole polytechnique fédérale de Lausanne (EPFL) of Lausanne, Switzerland, and meet the Prof. Andreas Burg and Reza Ghanaatian. I would like to thank them for their support in the ASIC implementation part of my work.

I would like to thank my friend Joel Ortiz Sosa for his help and discussions about ASIC synthesis. I thank also Titouan Gendron who helped me obtain the emulation results on FPGA.

I gladly acknowledge the Agence nationale de la recherche (ANR) that funded my research through ANR project NAND n° ANR-15-CE25-0006-01 . Part of this work uses resources funded by the Région Bretagne and the ANR through the CPER Sophie.

I would like to thank Prof. Norbert Wehn and Prof. Charly Poulliat for agreeing to be reviewers of my thesis. I extend my thanks to Prof. Christophe Jego for serving as the president of the PhD committee, Dr. Valentin Savin, and Dr. Elsa Dupraz for being the examiners. The comments of the committee have helped me improve my thesis and serve as a guide in my future career.

I would like to extend my gratitude to the colleagues in ETIS, ENSEA for their friendship, funs and encouragements especially Khoa Le Trung and Walid Hariri. All

my gratitude is also going to the administrate assistant of the ETIS laboratory, Annick Bertinoti, the administrate assistant of the Lab-STICC laboratory, Virginie Guillet, and the administrative staff of the doctoral school in Cergy-Pontoise and Lorient, were always very helpful in administrative issues.

I would like to express my sincere gratitude to all my teachers of the school Fe y Alegría n° 19, Huaraz, Peru. I would also like to warmly thank all of my professors at the National University of Engineering (Universidad Nacional de Ingeniería (UNI)), Lima, Peru. In the same way, I would like to extend my gratitude to all the engineers of the INICTEL-UNI, Lima, Peru, especially Eng. Daniel Díaz Ataucuri and Eng. Juan Alvarez Montoya.

Finally, my gratitude goes to my parents, Venancio Victor Cochachin Cursino and Teofila Cirila Henostroza Jamanca, who have been a perpetual source of courage and encouragement. They taught me the ethical values of life and always fight for what you want in life. I dedicate this work to them for their great love and for all their support that made it possible. My profound gratitude to my brother and my sisters, Clinton, Omaira, and Sandra.

Abstract

In this thesis, we are interested in improving the performance of the highly quantized Low-Density Parity-Check (LDPC) code decoder (3 or 4 bits of precision input). The first proposed decoder, named Noise-Against-Noise Min-Sum (NAN-MS) decoder, incorporates a certain amount of random perturbation due to deliberate noise injection into the decoding process. The other decoder, named Sign-Preserving Min-Sum (SP-MS) decoder, always preserve the sign of the messages and it uses all the possible combinations that can be generated for a given precision. We further show that a SP-MS decoder quantizing its inputs in 3 or 4 bits can reduce the precision of internal messages respectively to 2 or 3 bits without affecting the threshold of convergence of the code when the degree of the variable nodes is greater than 4. The NAN-MS decoder and the SP-MS decoder present a SNR gain of up to 0.43 dB in the waterfall region of the performance curve. On the other hand, we proposed a modification of an existing post-processing algorithm which makes it possible to reduce the residual error rate (region called "error floor") for the decoders with only 2 bits of precision for the exchanged messages. Applied to the IEEE 10 Gigabit ETHERNET code, the SP-MS algorithm combined with the post-processing algorithm reduces the error level below a frame error rate of 10^{-10}. We implemented this decoder in 28 nm ASIC technology with a completely parallel architecture. The resulting area is 1.76 mm^2, and the decoding rate of 319.34 Gbit/s for a signal-to-noise ratio of 5.5 dB, giving a hardware efficiency of 181.44 Gbit/s/mm^2.

Table of contents

List of figures

List of tables

Nomenclature

Notation

$\mathcal{A}_{app}$: Alphabet of the A Posteriory Probabilities

$\mathcal{A}_C$: Message alphabet of classical decoders

$\mathcal{A}_L$: The decoder input alphabet.

$\mathcal{A}_S$: Message alphabet of Sign-Preserving decoders

$\mathcal{A}_U$: Alphabet of the unsaturated variable-to-check message

c : Any Check Node

$\lceil a \rceil$: The ceiling function

d_c : Check node degree.

δ : Density Evolution threshold.

d_v : Variable node degree.

$\lfloor a \rfloor$: The floor function.

H : A parity check matrix of size $M \times N$ with $M < N$.

$\hat{\boldsymbol{x}}$: Decoder output of length N $\hat{\boldsymbol{x}} = (\hat{x}_1, ..., \hat{x}_N)$ which is an estimation of $\boldsymbol{x}$.

ℓ : Iteration number

$\lambda_i, \tilde{\lambda}_i$: The proportion of edges connected to variable-nodes of degree i, the proportion of variable-nodes of degree i.

$\lambda(x), \tilde{\lambda}(x)$: The edge-wise VN degree distribution, the node-wise VN degree distribution.

M : Number of check nodes: $M = N - K$.

$\max(a, b)$: Maximum of a and b.

$m_{c\to v}^{(\ell)}$: The message sent from CN c to VN v in the ℓ^{th} iteration

$\min(a, b)$: Minimum of a and b.

$m_{v\to c}^{(\ell)}$: The message sent from VN v to CN c in the ℓ^{th} iteration

$m_{v_n\to c_m}^{(\ell+1),U}$: The unsaturated variable-to-check message in the $(\ell+1)^{th}$ iteration

N : Number of variable nodes.

$\tilde{\delta}$: Noisy Density Evolution threshold.

Υ : A noise model

Ψ_c : Update function at a CN c of degree d_c

$\Theta^{(\ell)}$: The PMF of noiseless c-to-v messages in the ℓ^{th} iteration

$\tilde{\Theta}^{(\ell)}$: The PMF of noisy c-to-v messages in the ℓ^{th} iteration.

$\tilde{\Omega}^{(\ell)}$: The PMF of noisy v-to-c messages in the ℓ^{th} iteration.

$\Omega^{(\ell)}$: The PMF of noiseless v-to-c messages in the ℓ^{th} iteration

$\Omega^{(0)}$: The initial PMF of messages sent at $\ell = 0$

Ψ_v : Update function at a VN v of degree d_v.

q : The number of precision bits of messages.

q_{ch} : The number of precision bits of the decoder input.

R : Code rate.

$\rho_j, \tilde{\rho}_j$: The proportion of edges connected to check-nodes of degree j, the proportion of check-nodes of degree j.

$\rho(x), \tilde{\rho}(x)$: The edge-wise CN degree distribution, the node-wise CN degree distribution

$\boldsymbol{s}$: Encoder input of length K $\boldsymbol{s} = (s_1, ..., s_K)$.

$\text{sign}(a)$: Sign of a.

v : Any Variable Node

$\mathcal{V}(c)$: The set of neighbors of a CN c.

$\mathcal{V}(v)$: The set of neighbors of a VN v.

$\boldsymbol{w}$: Codeword mapped by *e.g.* a binary phase-shift keying modulation of length N $\boldsymbol{w} = (w_1, ..., w_N)$ which is sent through the noisy channel.

$\boldsymbol{x}$: Encoder output of length N $\boldsymbol{x} = (x_1, ..., x_N)$ which a codeword.

$\boldsymbol{y}$: Channel output of length N $\boldsymbol{y} = (y_1, ..., y_N)$.

Acronyms / Abbreviations

APP : A Posteriori Probability

ASIC : Application Specific Integrated Circuit

AWGN : Additive White Gaussian Noise

BER : Bit Error Rate

BI-AWGN : Binary-Input Additive White Gaussian Noise

BPSK : Binary Phase-Shift Keying

BP : Belief-Propagation

BSC : Binary Symmetric Channel

CNU : Check Node Update

CN : Check Node

DE : Density Evolution

FER : Frame Error Rate

FPGA : Field Programmable Gate Array

LDPC : Low Density Parity Check

LLR : Log-Likelihood Ratio

ML : Maximum Likelihood

MP : Message Passing

MS : Min-Sum

NAN : Noise-Against-Noise

NA : Noise-Aided

NMS : Normalized Min-Sum

OMS : Offset Min-Sum

PDF : Probability Density Function

PMF : Probability Mass Function

QC : Quasi-Cyclic

SNR : Signal-to-Noise Ratio

SP : Sign-Preserving

VNU : Variable Node Update

VN : Variable Node

Chapter 1

Introduction

It is a long way from the formal definition of channel capacity by Shannon in the 50^{th} and the modern coding theory. Nowadays, capacity (or almost capacity) achieving codes, like Polar code, Turbo code or LDPC code are available. Nevertheless, the rise of decoding throughput (up to the Tbit/s) due to emerging applications like optical fiber, free space laser communications, high speed memory access, push the decoding throughput at the edge of the deep submicron technology. The problem is no more the design of a code with good performance but to design of a hardware architecture that has contradictory requirement: low area print, low power dissipation and still, very good decoding performance.

This PhD explores solution to this problem with several a priori defined in the frame of the french funded project "Noise Against Noise Decoder" [1]: the use of LDPC code, the use of low input precision (3 or 4 bits for the channel quantization only) and finally, the help of some randomness in the decoding process to improve the decoding performance.

The main contributions of the thesis are the definition of improved decoders for quantized input channel with only 3 or 4 bits of precision, and a post-processing

[1]Funded by ANR, grant n° ANR-15-CE25-0006-01

algorithm for low precision iterative decoders. One of the proposed decoders, named Noise-Against-Noise Min-Sum (NAN-MS) decoder, incorporates a certain amount of random perturbation due to deliberate noise injection. The other of the proposed decoders, named Sign-Preserving Min-Sum (SP-MS) decoder, always preserve the sign of the messages and it uses all the possible combinations that can be generated for a given precision. Also, the SP-MS decoder can reduce the precision of its messages by one bit maintaining the same error correcting performance. The NAN-MS decoder and the SP-MS decoder present a SNR gain of up to 0.43 dB in the waterfall region of the performance curve. On the other hand, the proposed post-processing algorithm is very efficient and easily adaptable in low precision decoders. For the IEEE ETHERNET code, the post-processing algorithm implemented in a very low precision SP-MS decoder helps to lower the error floor below a FER of 10^{-10}.

The results of this work lead to two conference papers, one submitted patent, and one submitted journal paper.

[1] F. Cochachin, E. Boutillon and D. Declercq, "Optimization of Sign-Preserving Noise-Aided Min-Sum Decoders with Density Evolution," 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC), Hong Kong, Hong Kong, 2018, pp. 1-5.

[2] F. Cochachin, D. Declercq, E. Boutillon and L. Kessal, "Density evolution thresholds for noise-against-noise min-sum decoders," 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, 2017, pp. 1-7.

[3] F. Cochachin, E. Boutillon "ITERATIVE DECODER FOR DECODING A CODE COMPOSED OF AT LEAST TWO CONSTRAINT NODES", 03 december 2018, application number: EP18306599.4

[4] F. Cochachin, E. Boutillon, D. Declercq, "Sign-Preserving Min-Sum Decoders", Submitted to IEEE Transactions on Communications, February 2019.

The rest of the thesis is divided in 5 additional chapters.

Chapter II gives the background on regular and irregular LDPC codes, message passing decoding algorithms for floating point messages and for finite precision messages, and the asymptotic performance determination using density evolution for finite precision messages.

Chapter III presents the first contribution of the thesis: the elaboration of a new message passing decoding algorithm named Noise-Against-Noise Min-Sum (NAN-MS) decoder. In order to introduce randomness in the NAN-MS decoders, a noise injection method is proposed. The asymptotic behavior of the NAN-MS is investigated using a noisy density evolution (DE). Using the decoding thresholds obtained with the noisy DE, the best location of the noise injection is studied. On the other hand, in order to make a low cost implementation of the NAN-MS decoder, two finite precision decoders are defined, the first one called Modified Offset Min-Sum (M-OMS) decoder, and the second one is a version of M-OMS that uses an optimized offset vector $\boldsymbol{\lambda_v}$ and it is called OMS-$\boldsymbol{\lambda_v}$ decoder. The M-OMS and the OMS-$\boldsymbol{\lambda_v}$ decoders have an equivalent hardware complexity to the OMS decoder. It is shown by density evolution and by finite length simulations that: *(i)* the best location for the noise injection is in the variable nodes, and *(ii)* the NAN-MS, the M-OMS, and OMS-$\boldsymbol{\lambda_v}$ significantly outperform the decoders presented in the state of the art. A SNR gain of up to 0.4 dB is obtained in the waterfall region for low precision messages represented on 3 or 4 bits.

Chapter IV presents the second contribution of the thesis: the definition of a new finite precision iterative decoder for low-density parity-check codes named Sign-Preserving Min-Sum (SP-MS) decoder. The particularity of the SP-MS decoder is that variable-to-check messages are never set to 0 and always carry a sign information.

In order to optimize the SP-MS decoder performance, the noise injection method proposed in Chapter III is adapted to always preserve the sign of the messages and to cover the case of 2-bit precision message. The SP-MS decoder and its optimization are investigated in the asymptotic limit of the code length with the noisy DE. The study performed with DE and validated with finite length simulations presents that: *(i)* the SP-MS significantly improves the decoding performance compared to classical Offset Min-Sum when messages are quantized with only 2, 3 or 4 bits of precision, *(ii)* the SP-MS can reduce their precision messages by one bit maintaining the same error-correcting performance, thereby reducing the hardware complexity of the implementation, and *(iii)* the SP-MS decoder shows a SNR gain of up to 0.33 dB for regular codes and 0.43 dB for irregular codes, compared to the OMS.

Chapter V presents the third contribution of the thesis: a post-processing algorithm for low precision decoders. The first part of this chapter presents a fully parallel architecture to implement the MS, the OMS, and SP-MS. The proposed architecture is easily adapted to the implementation of regular and irregular LDPC codes. The synthesis results are presented for this architecture and reveal that the SP-MS decoder consumes less area than the OMS decoder, and that a saving of at least 25% of area used by the SP-MS decoder can be achieved when the precision of the messages is reduced. The second part of this chapter presents the post-processing algorithm for the IEEE 802.3 ETHERNET LDPC code. A fully parallel architecture is proposed to implement the SP-MS decoder and the proposed algorithm. Emulation results on FPGA exhibit that the error floor is lowered below a frame error rate (FER) level of 10^{-10}, observing only a slight error floor for FER bellow 10^{-10}. Implementation results on a 28 nm FD-SOI technology are presented showing a hardware efficiency of 181.44 Gbit/s/mm^2.

Chapter 2

Background

In a communication system, a transmitter sends information through a noisy channel to one or more receivers. The channel adds random noise and corrupts the information. The receiver has the purpose to retrieve the information with the least possible loss. In order to protect the information against the channel noise, the transmitter adds redundancy to the information such that the receiver can detect and correct the errors. Such process is called error correcting coding and decoding.

In the coding process, an error-correcting code converts a sequence of K information bits into a longer sequence of N bits using a coding function which defines how to build the $N-K$ redundancy bits. Examples of coding processes include convolutional codes, block turbo-codes, LDPC codes, algebraic codes, etc.

In convolutional codes, the coding function uses individually each bit of the sequence of K bits to build the redundancy bits, through a discrete linear filter. The classical algorithms used for the decoding are the BCJR algorithm and the Viterbi algorithm.

In block codes the encoding is made by block of bits, and in this case the sequence of K bits is used altogether to build the redundancy bits, through a binary generator matrix.

Nowadays, the most popular error-correcting codes are LDPC codes (or Gallager codes) [1–3], and the turbo-codes, because they have a high performance and practical decoding algorithms. LDPC codes, first introduced by Gallager in 1963, are widely used in communications standards like DVB-S2 [4], DVB-S2X [5], IEEE 802.3an [6], WIMAX (IEEE 802.16) [7], etc, and storage applications [8, 9] because they provide an exceptional error correction capability. Turbo codes are also used for communications standards like LTE, DVB-RCS, WIMAX, etc.

LDPC codes can be efficiently decoded by Message-Passing (MP) algorithms that use a Tanner graph [10] representation of the LDPC codes. One of the best MP algorithms is the Sum-Product algorithm also called Belief-Propagation (BP) algorithm [11]. The BP decoder has excellent decoding performance in the waterfall region but at a cost of a high computational complexity. It is worth noting that a well-designed LDPC code decoded by the BP decoder can approach the Shannon limit asymptotically [12, 13].

In the literature there are many BP-based decoders that are simplified versions of the BP decoder [14, 15]. The Min-Sum (MS) and offset corrected Min-Sum (OMS) decoders [14, 16], derived from the BP decoder, reduce the computational complexity, but also have a slight performance degradation, compared to BP, especially when they are implemented in finite precision. The effect of quantization on the messages of MP decoders has been extensively studied these past twenty years. In [17], the authors show that for the BP decoder, at least 6 precision bits should be used for the Binary-Input Additive White Gaussian Noise (BI-AWGN) channel. The effects of uniform quantization of the BP decoder over the Binary Symmetric Channel (BSC) is studied in [18]. As for the MS and OMS decoders, the effects of clipping and quantization for the BI-AWGN channel are studied in [16, 19]. The Self-Corrected Min-Sum (SC-MS) decoder [20], derived from the quantized MS decoder, sets to zero any message at

variable-nodes if the sign of the message changes between two consecutive iterations. From these works, it can be concluded that although the MS-based decoders are less sensitive than BP decoder to quantization effects, there is a non-negligible performance loss when the number of precision bits becomes too small.

It is worth mentioning that the quantized MS-based decoders mimic the BP decoder, but in the literature there are quantized decoders do not mimic the BP decoder like Finite Alphabet Iterative decoders (FAIDs) [21, 22], Non-surjective Finite Alphabet Iterative Decoders (NS-FAIDs) [23], Bit-Flipping decoders [24, 25], etc.

This work will only study LDPC codes and their associated quantized decoders that mimic the BP decoder.

2.1 Generalities on LDPC codes

An LDPC code is a linear block code defined by a sparse parity-check matrix $H = [h_{mn}]$ of M rows by N columns, with $M < N$. The following matrix is an example of a binary parity-check matrix

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

The usual graphical representation of an LDPC code is made by a Tanner graph which is a bipartite graph G composed of two types of nodes, the variable nodes (VNs) $v_n, n = 1, ..., N$ and the check nodes (CNs) $c_m, m = 1, ..., M$. A VN in the Tanner graph corresponds to a column of H and a CN corresponds to a row of H, with an edge

connecting CN c_m to VN v_n exists if and only if $h_{mn} \neq 0$, *i.e.* only CNS are connected to VNs and vice versa, other types of connection are not allowed.

In figure 2.1, we show a bipartite graph which is related to the sparse parity-check matrix presented in (2.1). This graph also represents a graph for a $(3, 4)$-regular LDPC code. Note that a VN is always represented with a circle, while a CN is always represented with a square.

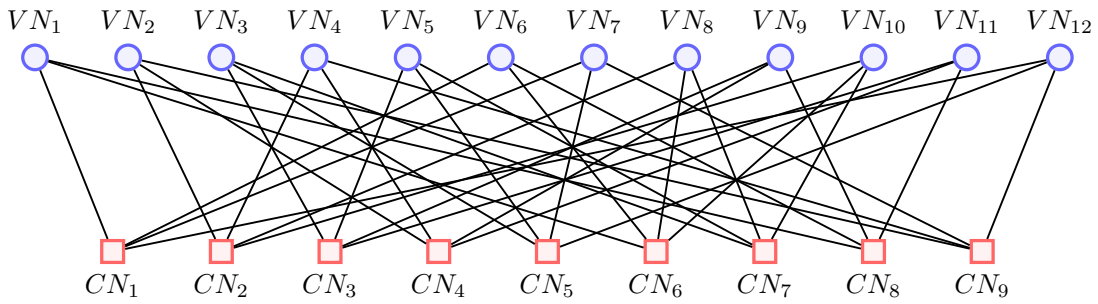


Fig. 2.1 A Tanner graph for a $(d_v = 3, d_c = 4)$-regular LDPC code of length $N = 12$. There are 12 VNs, 9 CNs, and 36 edges.

LDPC codes are classified according to their structural properties as regular or irregular LDPC codes. Additionally, the LDPC codes can also be classified as non-structured LDPC codes and structured LDPC codes. Non-structured LDPC codes does not exhibit a specific structure, while in structured LDPC codes, H is generated with algebraic equations, or constrained by specific topological properties. Usually, those constraints are introduced in order to help the decoder to have a low cost hardware implementation. One could further categorize the structured LDPC codes in three types: *(i)* quasi-cyclic LDPC codes, *(ii)* convolutional LDPC codes, and *(iii)* algebraic constructions of LDPC codes.

Let us assume that v is any VN and c is any CN. Let us also denote by $\mathcal{V}(v)$ the set of neighbors of a VN v, and denote $\mathcal{V}(c)$ the set of neighbors of a CN c. The degree of a node is the number of its neighbors in G.

2.1.1 Ensemble of Regular LDPC codes

A code is said to have a regular column-weight $d_v = |\mathcal{V}(v)|$ if all VNs v have the same degree d_v. Similarly, if all CNs c have the same degree $d_c = |\mathcal{V}(c)|$, a code is said to have a regular row-weight d_c. A (d_v, d_c)-regular LDPC code has all its VNs with the same degree d_v and all its CNs of a fixed degree d_c. In the corresponding sparse parity-check matrix H, d_v is the amount of nonzero elements per column, and d_c is the amount of nonzero elements per row. For a regular LDPC code, the density of H, denoted by d_H, is equal to $d_H = d_v/M = d_c/N$. For example, the matrix in (2.1) has a density d_H equal to $d_H = 1/3$. We can note that, for a fixed d_c, when N increases to infinity, d_H converges to zero.

Let us denote by E the number of edges in the Tanner graph, or equivalently E is the amount of non-zero elements in H. For a regular LDPC code we have $E = d_v \times N = d_c \times M$. The code rate R can be calculated as $R = K/N \geq \frac{N-M}{N}$ [26], and if the rows of H are linearly independent, we can write $R = 1 - (d_v/d_c)$, which is usually defined as the *design rate* [27].

Now, we introduce the concept of the ensemble of (d_v, d_c)-regular LDPC codes in order to do the theoretical analysis: *An ensemble or family $\mathcal{C}^N(d_v, d_c)$ of LDPC codes is composed of all the Tanner graphs with N VNs and regular degrees d_v and d_c.*

2.1.2 Ensemble of Irregular LDPC codes

In case of irregular LDPC codes [28], the nodes can have different connection degrees, defining an irregularity distribution, which is usually characterized by the two polynomials $\lambda(x) = \sum_{i=2}^{d_{v,max}} \lambda_i x^{i-1}$, and $\rho(x) = \sum_{j=2}^{d_{c,max}} \rho_j x^{j-1}$. The parameters λ_i (respectively ρ_j) indicate the fraction of edges connected to degree i VNs (respectively degree j CNs) [28, 29]. For regular codes, the polynomials reduce to monomials, $\lambda(x) = x^{d_v-1}$ and

$\rho(x) = x^{d_c - 1}$. Irregular LDPC codes can also be characterized by two other polynomials $\tilde{\lambda}(x) = \sum_{i=2}^{d_{v,max}} \tilde{\lambda}_i x^{i-1}$, and $\tilde{\rho}(x) = \sum_{j=2}^{d_{c,max}} \tilde{\rho}_j x^{j-1}$, where $\tilde{\lambda}_i \in [0,1]$ (respectively $\tilde{\rho}_j \in [0,1]$) denotes the number of VNs of degree i (respectively CNs of degree j).

As for the case of regular LDPC codes, E denotes the number of edges in the Tanner graph. Considering H, the amount of non-zero elements in columns of degree i is calculated as $E \times \lambda_i$ or $i \times \tilde{\lambda}_i \times N$. Similarly, the amount of non-zero elements in rows of degree j is given by $E \times \rho_j$ or $j \times \tilde{\rho}_j \times M$. Hence, it is easy to obtain $E = (i \times \tilde{\lambda}_i \times N)/\lambda_i = (j \times \tilde{\rho}_j \times M)/\rho_j$. The relationship between λ_i and $\tilde{\lambda}_i$ (respectively ρ_j and $\tilde{\rho}_j$) is giving by $\tilde{\lambda}(x) = \sum_{i=2}^{d_{v,max}} \frac{\lambda_i E}{iN} x^{i-1}$ (respectively $\tilde{\rho}(x) = \sum_{j=2}^{d_{c,max}} \frac{\rho_j E}{jM} x^{j-1}$).

The design rate $R(\lambda, \rho)$

For irregular LDPC codes, the design rate can be computed as $R(\lambda,\rho) = \frac{N-M}{N}$ [28]. We can calculate the number of VNs as $E\int_0^1 \lambda(x)dx$, similarly, the number of CNs is equal to $E\int_0^1 \rho(x)dx$. Using $\lambda(x)$ we have $\int_0^1 \lambda(x)dx = \int_0^1 \sum_{i=2}^{d_{v,max}} \lambda_i x^{i-1} = \sum_{i=2}^{d_{v,max}} \frac{\lambda_i}{i}$, and using $\rho(x)$ we get $\int_0^1 \rho(x)dx = \int_0^1 \sum_{j=2}^{d_{c,max}} \rho_j x^{j-1} = \sum_{j=2}^{d_{c,max}} \frac{\rho_j}{j}$. Therefore $R(\lambda,\rho)$ can also be computed as $R(\lambda,\rho) = 1 - (\int_0^1 \rho(x)dx)/(\int_0^1 \lambda(x)dx)$ or equivalently $R(\lambda,\rho) = 1 - (\sum_{j=2}^{d_{c,max}} \frac{\rho_j}{j})/(\sum_{i=2}^{d_{v,max}} \frac{\lambda_i}{i})$.

In figure 2.2, we show a Tanner graph for an irregular LDPC code. For example, for this graph we have $\lambda(x) = \frac{2}{3}x + \frac{1}{3}x^2$, $\rho(x) = \frac{1}{2}x^2 + \frac{2}{9}x^3 + \frac{5}{18}x^4$, $\tilde{\lambda}(x) = \frac{3}{4}x + \frac{1}{4}x^2$, $\tilde{\rho}(x) = \frac{3}{5}x^2 + \frac{1}{5}x^3 + \frac{1}{5}x^4$, and $R(\lambda,\rho) = \frac{3}{8}$.

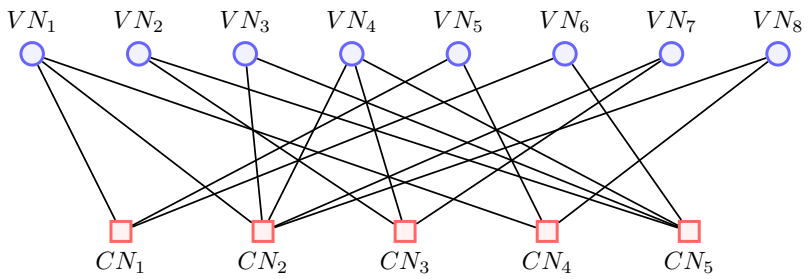


Fig. 2.2 A Tanner graph for an irregular LDPC code of length 8. There are 9 VNs, 5 CNs, and 18 edges.

Similar to the case of the (d_v, d_c)-regular LDPC ensemble, *an ensemble* $\mathcal{C}^N(\lambda(x), \rho(x))$ *of irregular LDPC codes is composed of all the Tanner graphs of length* N *and associated to the degree distribution pair* $(\lambda(x), \rho(x))$.

2.2 Binary LDPC Decoders

In this Section, we first introduce the notations and terminologies related to binary LDPC decoders we use throughout this work. Then, we briefly review a number of important message-passing decoders.

2.2.1 Definitions

In Fig. 2.3, we depict a simple communication system. We assume that the source produces a vector $\boldsymbol{s} = (s_1, ..., s_K)$. The encoder adds redundancy to $\boldsymbol{s}$ in order to obtain an encoded vector $\boldsymbol{x} = (x_1, ..., x_N)$, which is a codeword, and which is mapped *e.g.* by the binary phase-shift keying (BPSK) modulation, to obtain $\boldsymbol{w} = (w_1, ..., w_N)$. After $\boldsymbol{w}$ is sent through a noisy channel. Based on the channel output $\boldsymbol{y} = (y_1, ..., y_N)$, the decoder produces the vector $\hat{\boldsymbol{x}} = (\hat{x}_1, ..., \hat{x}_N)$ which is an estimation of $\boldsymbol{x}$. To check if $\hat{\boldsymbol{x}}$ is a valid codeword, we verify that the syndrome vector is all-zero, *i.e.* $H\hat{\boldsymbol{x}}^T = 0$.

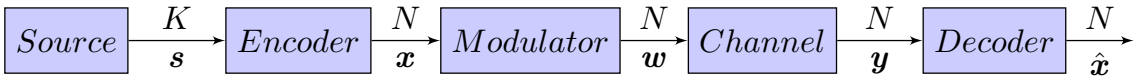


Fig. 2.3 A simple communication system

We denote the channel output alphabet by $\mathcal{A}_y$. For binary LDPC decoders we have $\boldsymbol{y} \in \mathcal{A}_y^N$, $\boldsymbol{s} \in \{0,1\}^K$, $\boldsymbol{x} \in \{0,1\}^N$, and $\hat{\boldsymbol{x}} \in \{0,1\}^N$. The channel output alphabet depends on the channel model, we consider in this work two models of binary memoryless channels: the first one, the Binary Symmetric Channel (BSC); and the second one, the Binary-Input Additive White Gaussian Noise (BI-AWGN) channel.

Binary Symmetric Channel

In the BSC, a bit transmitted $x_n \in \{0,1\}$ is flipped to y_n with probability ϵ, referred to as the *error probability* or *crossover probability* of the channel, hence $y_n \in \mathcal{A}_y = \{0,1\}$. The BSC satisfies the following symmetry condition

$$p(y_n = 1 \mid x_n = 0) = p(y_n = 0 \mid x_n = 1), \tag{2.2}$$

where $p(y \mid x)$ is the channel transition probability.

Binary-Input Additive White Gaussian Noise channel

The BI-AWGN channel is modeled by $y_n = (1 - 2x_n) + z_n$, where $1 - 2x_n \in \{+1, -1\}$, and z_n is a sequence of independent and identically distributed (i.i.d.) random variables with probability density function given by the normal (or Gaussian) distribution, hence $y_n \in \mathcal{A}_y = \mathbb{R}$.

$$p_{BIAWGN}(z_n) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z_n)^2/2\sigma^2}, \tag{2.3}$$

where σ^2 is the noise variance.

Similarly to the BSC, the BI-AWGN channel satisfies the following symmetry condition

$$p(y_n = \psi \mid x_n = 0) = p(y_n = -\psi \mid x_n = 1), \psi \in \mathcal{A}_y, \tag{2.4}$$

where $p(y \mid x)$ defines the likelihood distribution.

Log-Likelihood Ratio

A log-likelihood ratio (LLR) form for the bit x_n with probability $p(x_n)$ is defined by

$$LLR(x_n) = \log\left(\frac{p(x_n = 0)}{p(x_n = 1)}\right), \tag{2.5}$$

where $p(x_n = 0) + p(x_n = 1) = 1$. If $p(x_n = 0) > p(x_n = 1)$ then $LLR(x_n)$ is positive; if the inequality is reversed, then $LLR(x_n)$ is negative. Therefore, the sign of $LLR(x_n)$ indicates the value of the bit x_n ($x_n = (1-\text{sign}(LLR(x_n)))/2$), and the magnitude $|LLR(x_n)|$ of $LLR(x_n)$ give us a measure of its reliability. We consider that the notation log denotes the logarithm with base e in the rest of the text.

The channel output can be also expressed in a LLR form as follows

$$LLR(y_n) = \log\left(\frac{\Pr(y_n \mid x_n = 0)}{\Pr(y_n \mid x_n = 1)}\right). \tag{2.6}$$

In the case of the BSC, we get $LLR(y_n) = (1-2y_n)\log((1-\epsilon)/\epsilon)$ where $y_n \in \mathcal{A}_y = \{0,1\}$, and for the BI-AWGN channel with noise variance σ^2, we have $LLR(y_n) = 2y_n/\sigma^2$ where $y_n \in \mathcal{A}_y = \mathbb{R}$.

2.2.2 Message-Passing Decoders

Message-Passing (MP) decoders are iterative decoders that use a Tanner graph to pass messages along the edges. In MP decoders, a VN v_n (respectively a CN c_m) sends its message to its neighbors $\mathcal{V}(v_n)$ (respectively $\mathcal{V}(c_m)$). In each iteration, the VN update (VNU) and the CN update (CNU) compute outgoing messages from all incoming messages.

Here we present the notations used to describe different MP decoders. We consider that the message alphabet is $\mathcal{A}_C$, while the decoder input alphabet is denoted by $\mathcal{A}_L$. Unless otherwise stated, the decoder input alphabet will be the one of the messages, *i.e* $\mathcal{A}_L = \mathcal{A}_C$. Let us denote by $\ell \in \mathbb{N}$ the iteration number. Let us also denote by $m_{v\to c}^{(\ell)} \in \mathcal{A}_C$ the message sent from VN v to CN c in the ℓ^{th} iteration, denote by $m_{c\to v}^{(\ell)} \in \mathcal{A}_C$ the message sent from CN c to VN v in the ℓ^{th} iteration, and denote by $\boldsymbol{\gamma}^{(\boldsymbol{\ell})} = (\gamma_1^{(\ell)}, ..., \gamma_N^{(\ell)})$ the *a posteriori* probability (APP), where $\gamma_n^{(\ell)}$ is associated to the VN v_n, for $n = 1, 2, ..., N$. Let us also denote by $m_{v\to c}^{(\ell),U}$ the unsaturated variable-to-check

message in the ℓ^{th} iteration. Fig. 2.4 depicts a Tanner graph fragment, showing the flow of messages in MP decoders. From this figure we have $\mathcal{V}(v_n) = \{c_m, c_1, c_2, c_3\}$ and $\mathcal{V}(c_m) = \{v_n, v_1, v_2, v_3\}$.

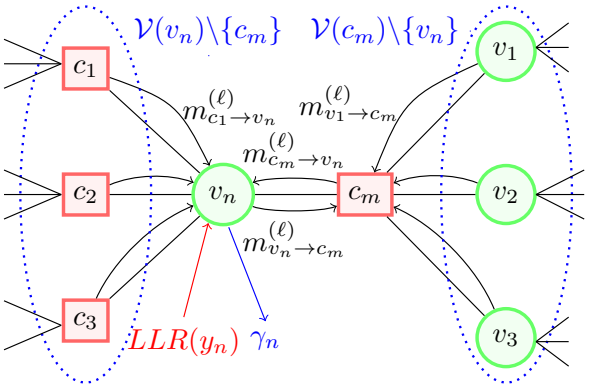


Fig. 2.4 A Tanner graph fragment

In this work, the message will be described in the log-likelihood ratio domain. The sign of a message m represents the hard-decision value of the VN it is connected to, and magnitude $|m|$ of m represents its reliability. As a consequence, the message alphabet $\mathcal{A}_C$ has to be symmetric around 0, for example $\mathcal{A}_C = \mathbb{R}$ for a continuous alphabet and $\mathcal{A}_C = \{-N_q, ..., -1, 0, +1, ..., +N_q\}$ for a discrete alphabet with $N_q \in \mathbb{N}$. The LLR from the channel observation $LLR(y_n) \in \mathcal{A}_L$ is usually referred to as the *intrinsic* message for the VN v_n. The exchanged messages in the decoder are usually referred to as *extrinsic* messages. For a successful decoding, the measure of the reliability of *extrinsic* messages becomes more and more reliable at each new iteration.

We now define update functions for the VNU and the CNU. Let $\Psi_v : \mathcal{A}_L \times \mathcal{A}_C^{(d_v-1)} \to \mathcal{A}_C$ denote the function used for the update at a VN v of degree d_v. Let $\Psi_c : \mathcal{A}_C^{(d_c-1)} \to \mathcal{A}_C$ denote the function used for the update at a CN c of degree d_c. The functions Ψ_v and Ψ_c satisfy the symmetry conditions defined in [27]. Let us denote by $\mathcal{A}_{app}$ the alphabet of APPs. Also, let $\Psi_a : \mathcal{A}_L \times \mathcal{A}_C^{(d_v)} \to \mathcal{A}_{app}$ denote the function used for the APP update at a VN v of degree d_v. With those notations, we describe the main steps of MP decoders:

1. [**Initialization**] the LLR $LLR(y_n)$ is computed for each VN v_n. Then, variable-to-check messages $m^{(\ell)}_{v_n \to c_m}$ are initialized by $LLR(y_n)$ at the 0^{th} iteration.

2. [**Iteration Loop**] Each decoding iteration consists of the following steps:

   (a) [**CNU**] a CN computes the outgoing message based on all the incoming messages except the one received from the outgoing message.
   $m^{(\ell)}_{c_m \to v_n} = \Psi_c \left(\left\{ m^{(\ell)}_{v \to c_m} \right\}_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \right).$

   (b) [**VNU**] a VN computes the outgoing message from the channel observation and from all the incoming messages except the one which receives the outgoing message.
   $m^{(\ell+1)}_{v_n \to c_m} = \Psi_v \left(LLR(y_n), \left\{ m^{(\ell)}_{c \to v_n} \right\}_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \right).$

   (c) [**APP-update**] (*a posteriori* probability update) the APP is computed from the channel observation and from all the incoming messages.
   $\gamma^{(\ell)}_n = \Psi_a \left(LLR(y_n), \left\{ m^{(\ell)}_{c \to v_n} \right\}_{c \in \mathcal{V}(v_n)} \right)$

   (d) [**Hard decision**] makes an estimation of the bits transmitted from the APP.
   $\hat{x}_n = (1 - \text{sign}(\gamma^{(\ell)}_n))/2$

   (e) [**Syndrome check**] verifies that the syndrome vector is all-zero in order to check if $\hat{\boldsymbol{x}}$ is a valid codeword.
   $H\hat{\boldsymbol{x}}^T = 0.$

The decoding stops when either $[\hat{x}_n]_{n=1,\ldots,N}$ is a codeword or a maximum number of iteration L_{max} is reached.

In binary LDPC decoders, the MP decoding can be performed either *(i)* using one bit of precision to represent messages (hard-decision MP decoders) or *(ii)* using more than one bit of precision to represent messages (soft-decision MP decoders). We present in the next sections the most common examples of such decoders.

Hard-decision MP decoders

Hard-decision MP decoders use just one bit of precision to represent the messages propagated in the Tanner graph, *i.e.* $m_{v\to c}^{(\ell)}$ and $m_{c\to v}^{(\ell)} \in \{+1,-1\}$ (which is equivalent to $\{0,1\}$). Hard-decision decoders are more interesting for the BSC than for the AWGN channel. We restrict in this section the presentation of the decoders to the BSC.

1) Gallager-B decoder: The Gallager-B decoder is a MP decoder [30, 31] with binary alphabet $\mathcal{A}_C = \{+1,-1\}$. At the initialization step, the LLR from the BSC is equal to $LLR(y_n) = 1-2y_n \in \mathcal{A}_C$, $y_n \in \mathcal{A}_y = \{0,1\}$, and $m_{v_n\to c_m}^{(0)} = LLR(y_n)$ at iteration $\ell = 0$. In each decoding iteration, the CNU computes the check-to-variable messages $m_{c_m\to v_n}^{(\ell)}$ as the parity (in ± 1 format) of the incoming messages $m_{v\to c_m}^{(\ell)}$, where $v \in \mathcal{V}(c_m)\backslash\{v_n\}$:

$$m_{c_m\to v_n}^{(\ell)} = \Psi_c\left(\left\{m_{v\to c_m}^{(\ell)}\right\}_{v\in\mathcal{V}(c_m)\backslash\{v_n\}}\right) = \prod_{v\in\mathcal{V}(c_m)\backslash\{v_n\}} m_{v\to c_m}^{(\ell)} \tag{2.7}$$

The VNU computes the variable-to-check messages $m_{v_n\to c_m}^{(\ell+1)}$ by comparing the sum of $LLR(y_n)$ and the incoming messages $m_{c\to v_n}^{(\ell)}$ to a pre-determined threshold t:

$$m_{v_n\to c_m}^{(\ell+1)} = \Psi_v\left(LLR(y_n), \left\{m_{c\to v_n}^{(\ell)}\right\}_{c\in\mathcal{V}(v_n)\backslash\{c_m\}}\right) = \begin{cases} LLR(y_n), & \text{if } \left|m_{v_n\to c_m}^{(\ell+1),U}\right| < t \\ \operatorname{sign}\left(m_{v_n\to c_m}^{(\ell+1),U}\right), & \text{otherwise.} \end{cases} \tag{2.8}$$

where $m_{v_n\to c_m}^{(\ell+1),U} = LLR(y_n) + \sum_{c\in\mathcal{V}(v_n)\backslash\{c_m\}} m_{c\to v_n}^{(\ell)}$.

The APP $\gamma_n^{(\ell)}$ can be calculated as $\gamma_n^{(\ell)} = LLR(y_n) + \sum_{c\in\mathcal{V}(v_n)} m_{c\to v_n}^{(\ell)}$. Hence, the hard decision $\hat{x}_n$ can be computed as $\hat{x}_n = (1-\operatorname{sign}(\gamma_n^{(\ell)}))/2$ if $\gamma_n^{(\ell)} \neq 0$, otherwise $\hat{x}_n = (1-\operatorname{sign}(LLR(y_n)))/2$. The decoder stops if either $[\hat{x}_n]_{n=1,\ldots,N}$ is a codeword or a maximum number of iteration is reached.

To improve the performance of the Gallager-B decoder, the threshold t can be optimized, could take different values at each iteration, and may vary from a VNU to another. However, the value of t is most of the time considered as constant. Two examples of Gallager-B decoders are the Gallager-A decoder and the Majority-Voting decoder, and we can refer to [32] for more details.

2) Gallager-A decoder: The Gallager-A decoder can be seen as a particular case of the Gallager-B decoder. In the Gallager-A decoder, the threshold t is equal to $d_v - 2$, where d_v is the degree of the VN v.

3) Majority-Voting decoder: The Majority-Voting decoder can be obtained from the Gallager-B decoder. In this case the threshold t is equal to 1. Hence, the update rule at a VN is given by

$$m_{v_n \to c_m}^{(\ell+1)} = \Psi_v \left(LLR(y_n), \left\{ m_{c \to v_n}^{(\ell)} \right\}_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \right) = \begin{cases} LLR(y_n), & \text{if } m_{v_n \to c_m}^{(\ell+1),U} = 0 \\ \text{sign}\left(m_{v_n \to c_m}^{(\ell+1),U} \right), & \text{otherwise.} \end{cases}$$

4) Gallager-B decoder with extended alphabet: The message alphabet for the Gallager-B decoder with extended alphabet is $\mathcal{A}_C = \{-1, 0, 1\}$. The value 0 is added to the alphabet to deal with the ties in the VNU of the Gallager-B and propagate 0 instead of the likelihood in such case. This decoder is also named *erasure decoder*. Strictly speaking this decoder is not a hard-decision decoder, and a least two bits of precision must be used in the hardware implementation. The update rule at a VN is given by

$$m_{v_n \to c_m}^{(\ell+1)} = \Psi_v \left(LLR(y_n), \left\{ m_{c \to v_n}^{(\ell)} \right\}_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \right) = \text{sign}\left(m_{v_n \to c_m}^{(\ell+1),U} \right).$$

Some authors have proposed to improve the performance of the erasure decoder by weighting differently the channel value and the extrinsic contribution. Refer to [32, 33] for more details.

Soft-decision MP decoders

Soft-decision MP decoders use more than one bit of precision to represent messages in the Tanner graph. These decoders propagate more information than hard-decision decoders, and they usually work in the LLR domain using real valued message, *i.e.* $m_{v\to c}^{(\ell)}$ and $m_{c\to v}^{(\ell)} \in \mathbb{R}$. The performance of soft-decision MP decoders is much better than the performance of the hard-decision MP decoders, but their hardware implementation is more complex. The sum-product algorithm, the Min-Sum (MS) algorithm are some examples of algorithms used for soft-decision MP decoders.

1) Belief Propagation decoder: One important kind of message-passing decoding algorithms is the sum-product algorithm [11], also called Belief-Propagation (BP) algorithm. This algorithm is commonly used in different applications like artificial intelligence, information theory, etc.

For the BP decoder presented in this section, we consider that the message alphabet is continuous, *i.e.* $\mathcal{A}_C = \mathbb{R}$. Also, we have obtained that the LLR is equal to $LLR(y_n) = (1-2y_n)\log\left((1-\epsilon)/\epsilon\right)$ for the BSC and $LLR(y_n) = 2y_n/\sigma^2$ for the BI-AWGN channel. The update rule at a VN is given by

$$m_{v_n\to c_m}^{(\ell+1)} = \Psi_v\left(LLR(y_n), \left\{m_{c\to v_n}^{(\ell)}\right\}_{c\in\mathcal{V}(v_n)\setminus\{c_m\}}\right) = LLR(y_n) + \sum_{c\in\mathcal{V}(v_n)\setminus\{c_m\}} m_{c\to v_n}^{(\ell)}. \tag{2.9}$$

And the update rule at a CN is given by

$$m_{c_m \to v_n}^{(\ell)} = \Psi_c \left(\left\{ m_{v \to c_m}^{(\ell)} \right\}_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \right) = \log \left(\frac{1 + \prod\limits_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \tanh \frac{m_{v \to c_m}^{(\ell)}}{2}}{1 - \prod\limits_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \tanh \frac{m_{v \to c_m}^{(\ell)}}{2}} \right). \quad (2.10)$$

Considering the function

$$\Phi(x) = \log \left(\tanh \frac{x}{2} \right) = \log \left(\frac{1 + e^{-x}}{1 - e^{-x}} \right), \forall x > 0,$$

equation (2.10) can be rewritten as

$$m_{c_m \to v_n}^{(\ell)} = \left(\prod_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \operatorname{sign} \left(m_{v \to c_m}^{(\ell)} \right) \right) . \Phi \left(\sum_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \Phi \left(\left| m_{v \to c_m}^{(\ell)} \right| \right) \right) \quad (2.11)$$

The APP update at a VN v_n of the BP decoder is given by

$$\gamma_n^{(\ell)} = \Psi_a \left(LLR(y_n), \left\{ m_{c \to v_n}^{(\ell)} \right\}_{c \in \mathcal{V}(v_n)} \right) = LLR(y_n) + \sum_{c \in \mathcal{V}(v_n)} m_{c \to v_n}^{(\ell)}. \quad (2.12)$$

From the APP, $\hat{x}_n$ can be computed as $\hat{x}_n = (1 - \operatorname{sign}(\gamma_n^{(\ell)}))/2$, for $n = 1, ..., N$.

The BP decoder is quite tedious to implement because of the function Φ used to compute check-to-variable messages.

2) Min-Sum decoder: The Min-Sum (MS) decoder [14, 16] is derived from the BP decoder. The MS decoder reduces the computational complexity at the CNU and is less sensitive than the BP decoder to message quantization effects.

In the MS decoder, the update rule at a VN is the same as the BP decoder, equation (2.9). To obtain the update rule at a CN, the following relation is used

$$\Phi\left(\Phi(a)+\Phi(b)\right) \leq \min(a,b), \quad \forall a>0 \text{ and } b>0. \tag{2.13}$$

Using this relation, one could replace the CNU of the BP (2.11) by

$$m^{(\ell)}_{c_m \to v_n} = \left(\prod_{v \in \mathcal{V}(c_m)\setminus\{v_n\}} \operatorname{sign}\left(m^{(\ell)}_{v\to c_m}\right) \right) . \left(\min_{v \in \mathcal{V}(c_m)\setminus\{v_n\}} \left(\left| m^{(\ell)}_{v\to c_m} \right| \right) \right). \tag{2.14}$$

The APP update of the MS decoder is also the same as the BP decoder.

3) Min-Sum-based decoders: There are several Min-Sum-based decoders proposed in the literature which have been proposed to improve the performance of the MS decoder. We introduce briefly two of them.

(i) The Normalized-Min-Sum (NMS) decoder [14, 16], in this decoder a factor $\gamma \in]0,1[$ is used to weight the messages at the output of the CN update. Since the relation (2.13) leads to a systematic overestimation of the amplitude of the CNU output message, it makes sense to shrink it with $\gamma \in]0,1[$. The factor γ could be fixed to a constant value, or vary according to the check-node degree. This factor can be optimized by Monte-Carlo simulation, or using Density Evolution analysis. For the NMS decoder, the VNU and APP update are defined in (2.9) and (2.12), respectively, while the CNU defined in (2.14) is replaced by

$$m^{(\ell)}_{c_m \to v_n} = \gamma . \left(\prod_{v \in \mathcal{V}(c_m)\setminus\{v_n\}} \operatorname{sign}\left(m^{(\ell)}_{v\to c_m}\right) \right) . \left(\min_{v \in \mathcal{V}(c_m)\setminus\{v_n\}} \left(\left| m^{(\ell)}_{v\to c_m} \right| \right) \right). \tag{2.15}$$

(ii) The Offset-Min-Sum (OMS) decoder [14, 16], a variant of the MS decoder is proposed by using an offset value $\lambda > 0$ to diminish the message amplitude at the output

of the CNU. As in the NMS decoder, the offset λ has the objective of compensating the over-estimation of the MS outputs. The offset λ can be a constant value, or vary according to the check-node degree, and can be optimized by Monte-Carlo simulation, or using Density Evolution analysis. The VNU and APP update of the OMS decoder are the same as the BP decoder, and the CNU defined in (2.14) is replaced by

$$m_{c_m \to v_n}^{(\ell)} = \left(\prod_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \operatorname{sign}\left(m_{v \to c_m}^{(\ell)} \right) \right) . \max \left\{ \left(\min_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \left(\left| m_{v \to c_m}^{(\ell)} \right| \right) \right) - \lambda, 0 \right\}. \tag{2.16}$$

2.3 Quantized Min-Sum-Based Decoders and Density Evolution

This section is dedicated to the presentation of the main theoretical tool that is used to analyze the performance of LDPC ensembles and decoders, called Density Evolution (DE). The concept of DE is to track the evolution of the probability mass function of the messages during the iterations of LDPC decoders. Although DE has been introduced as a theoretical approach, it turns out to be a very efficient tool to predict the performance of LDPC decoders in the waterfall region. This is especially true when the DE can follow the exact density of the messages (under the independence assumption), which is the case of quantized decoders, when the message alphabet is small enough.

From now on, and throughout the rest of this work, unless otherwise stated, we assume that the message is finite, and composed of $2N_q+1$ states, with $N_q = 2^{(q-1)} - 1$. Hence, $\mathcal{A}_C = \{-N_q, -(N_q-1), ..., -1, 0, +1, ..., +(N_q-1), +N_q\}$, *i.e.* the messages are quantized on q bits of precision.

2.3.1 Channel Value Quantization

Quantization for the Binary Symmetric Channel

According to Section 2.2.2, for the BSC we have $LLR(y_n) = (1-2y_n)\log((1-\epsilon)/\epsilon) \in \mathbb{R}$, with $y_n \in \mathcal{A}_y = \{0,1\}$. For a given BSC probability ϵ, the decoder input alphabet is composed of two values in $\mathcal{A}_L = \{+|LLR(y_n)|, -|LLR(y_n)|\}$. Fig. 2.5 shows the LLR $LLR(y_n)$ as a function of the cross-over probability ϵ, we can see that $LLR(y_n)$ is a real number.

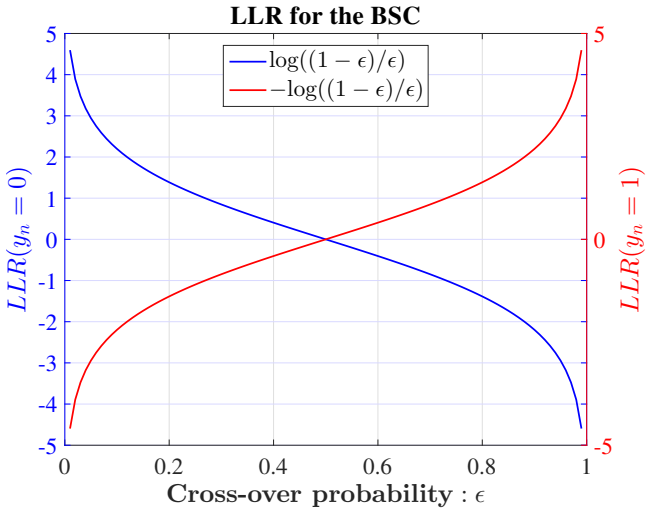


Fig. 2.5 $LLR(y_n)$ for the BSC.

As we deal with quantized decoders with message alphabet $\mathcal{A}_C$, we can consider without loss of generality that $\mathcal{A}_L \subseteq \mathcal{A}_C$. In the sequel, we denote by C the *channel value* which is a positive integer, and consider that $|LLR(y_n)|$ is mapped to C. As a result, the decoder input alphabet becomes $\mathcal{A}_L = \{+C, -C\}$, with $C \in \{+1, +2, ..., N_q\}$. In other words $y_n = 0$ is mapped to $+C$ and $y_n = 1$ is mapped to $-C$. The value of C can be seen as an extra degree of freedom in the decoder definition. For example, a MS decoder with $C = 1$ and a MS decoder with $C = 2$ are interpreted as two different decoders. The value of C can be optimized in order to improve the decoder performance.

Quantization for the Binary-Input Additive White Gaussian Noise channel

According to Section 2.2.2, for the BI-AWGN channel we have $LLR(y_n) = 2y_n/\sigma^2$, with $y_n \in \mathcal{A}_y = \mathbb{R}$. In the initialization step, variable-to-check messages are initialized by integer numbers in quantized decoders, hence, $LLR(y_n)$ has to be quantized on q bits of precision considering that $\mathcal{A}_L = \mathcal{A}_C$.

Let us denote the quantizer by $\mathcal{Q} : \mathbb{R} \to \mathcal{A}_L$ for MS-based decoders, defined as

$$\mathcal{Q}(a) = \mathcal{S}\left(\lfloor \alpha \times a + 0.5 \rfloor, N_q\right), \tag{2.17}$$

where $\lfloor . \rfloor$ depicts the floor function and $\mathcal{S}(b, N_q)$ is the saturation function clipping the value of b in the interval $[-N_q, N_q]$, *i.e.* $\mathcal{S}(b, N_q) = \min(\max(b, -N_q), +N_q)$. The parameter α is called *channel gain factor* and is used to enlarge or decrease the amplitude of LLRs at the decoder input. Similar to C for the BSC, the value of α can be seen as an extra degree of freedom in the quantized decoder definition that can be analyzed and optimized for quantized decoders on the BI-AWGN channel.

With those notations, we define the quantized version of the intrinsic LLR that initialize the MS-based decoders by the vector $\boldsymbol{I} = (I_1, ..., I_N) \in \mathcal{A}_L^N$, with

$$I_n = \mathcal{Q}(LLR(y_n)) \quad \forall n = 1, \ldots, N. \tag{2.18}$$

Fig. 2.6 shows the distribution of the quantized LLR I_n using $q = 3$ bits of precision. We can note that among 8 levels of a 3 bits representation, only 7 are used.

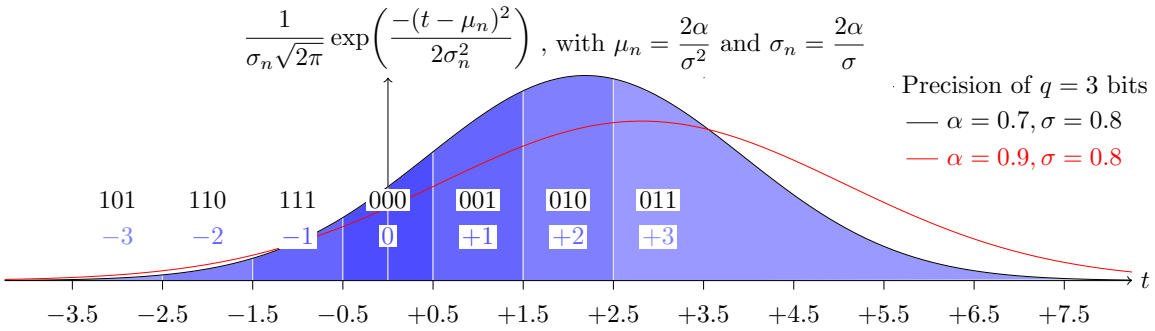


Fig. 2.6 $LLR(y_n)$ and I_n of quantized decoders for the BI-AWGN channel and precision $q = 3$.

2.3.2 Quantized Min-Sum-Based Decoders

Notations

We keep the notations presented in Section 2.1 and Section 2.2.2, *i.e.* $\ell \in \mathbb{N}$ denotes the iteration number, $m_{v\to c}^{(\ell)} \in \mathcal{A}_C$ denotes the message sent from VN v to CN c in the ℓ^{th} iteration, $m_{c\to v}^{(\ell)} \in \mathcal{A}_C$ denotes the message sent from CN c to VN v in the ℓ^{th} iteration, $m_{v\to c}^{(\ell),U}$ denotes the unsaturated variable-to-check message in the ℓ^{th} iteration. Also, $\mathcal{V}(v_n)$ is the set of neighbors of a VN v_n in a Tanner graph, and $\mathcal{V}(c_m)$ is the set of neighbors of a CN c_m in a Tanner graph.

Following the definitions of the VNU and the CNU presented in Section 2.2.2, in this section we present the discrete update functions for quantized Min-Sum-Based decoders. $\Psi_v : \mathcal{A}_L \times \mathcal{A}_C^{(d_v-1)} \to \mathcal{A}_C$ denotes the discrete function used for the update at a VN v of degree d_v, and $\Psi_c : \mathcal{A}_C^{(d_c-1)} \to \mathcal{A}_C$ denotes the discrete function used for the update at a CN c of degree d_c. $\Psi_v : \mathcal{A}_L \to \mathcal{A}_C$ can be written in the 0th iteration. The functions Ψ_v and Ψ_c also satisfy the symmetry conditions presented in [27]. Also, $\Psi_a : \mathcal{A}_L \times \mathcal{A}_C^{(d_v)} \to \mathcal{A}_{app}$ denotes the function used for the APP update at a VN v of degree d_v.

We also consider that $\boldsymbol{\gamma^{(\ell)}} = (\gamma_1^{(\ell)}, \gamma_2^{(\ell)}, ..., \gamma_N^{(\ell)})$ denote the *a posteriori* probability, where $\gamma_n^{(\ell)} \in \mathcal{A}_{app}$ is associated to the VN v_n. For quantized Min-Sum-Based decoders,

the alphabet of APPs is given by $\mathcal{A}_{app} = \{-N_q \times d_v - C, ..., -1, 0, +1, ..., +N_q \times d_v + C\}$ for the BSC, and by $\mathcal{A}_{app} = \{-N_q \times (d_v + 1), ..., -1, 0, +1, ..., +N_q \times (d_v + 1)\}$ for the BI-AWGN channel. Also, the likelihoods of the vector $\boldsymbol{I} = (I_1, ..., I_N) \in \mathcal{A}_L^N$ are computed as explained in the previous sections, *i.e.* $I_n = \pm C$ for the BSC, and $I_n = \mathcal{Q}(LLR(y_n))$ for the BI-AWGN channel.

Update Rules

Two quantized decoders are considered, a quantized Min-Sum decoder and a quantized Offset Min-Sum decoder with offset value $\lambda \in \{+1, ..., +(N_q - 2)\}$. The update rule at a CN is the same for the MS and the OMS decoders, and is given by

$$m_{c_m \to v_n}^{(\ell)} = \left(\prod_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \operatorname{sign}\left(m_{v \to c_m}^{(\ell)} \right) \right) . \max \left\{ \left(\min_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \left(\left| m_{v \to c_m}^{(\ell)} \right| \right) \right) - \lambda, 0 \right\}. \tag{2.19}$$

As the CNU, we can use the same expression for both MS and OMS decoders, since the OMS decoder with offset value $\lambda = 0$ becomes the MS decoder.The VNU expression is given by

$$m_{v_n \to c_m}^{(\ell+1)} = \Psi_v \left(I_n, \left\{ m_{c \to v_n}^{(\ell)} \right\}_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \right) = \mathcal{S} \left(m_{v_n \to c_m}^{(\ell+1),U}, N_q \right). \tag{2.20}$$

where $m_{v_n \to c_m}^{(\ell+1),U} = I_n + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} m_{c \to v_n}^{(\ell)}$.

The *a posteriori* probability update at a VN is given by

$$\gamma_n^{(\ell)} = \Psi_a \left(I_n, \left\{ m_{c \to v_n}^{(\ell)} \right\}_{c \in \mathcal{V}(v_n)} \right) = I_n + \sum_{c \in \mathcal{V}(v_n)} m_{c \to v_n}^{(\ell)}. \tag{2.21}$$

And the decision on the estimated bit $\hat{x}_n$ is taken according to the sign of $\gamma_n^{(\ell)}$.

As we can see, there is not much difference with the decoders presented in Section 2.2.2, except that the messages belong to a discrete alphabet, and then the update rules require a saturation function $\mathcal{S}$.

2.3.3 Density Evolution for Quantized Min-Sum-Based Decoders

Density evolution (DE) is a tool which describes the asymptotic behavior of an iterative MP decoder as a dynamic system, and follows the probability mass function (PMF) of the messages in the Tanner graph along the iterations. With DE, one can predict if an ensemble of LDPC codes, parametrized by its degree distribution, decoded with a given MP decoder, converges to zero error probability in the limit of infinite block length. This property gives rise to the definition of a *density evolution threshold* [27, 34, 35].

The DE threshold δ is expressed as a crossover probability ($\delta = \epsilon$) for the BSC or as a standard deviation ($\delta = \sigma$) for the BI-AWGN channel, with the objective of separating two regions of decoder convergence. The first region composed of values smaller than δ corresponds to the region where the DE converges to the zero error probability fixed point. The second region composed of values greater than δ corresponds to the region where the DE does not converge. In this later case, the DE converges to a fixed point which does not represent the zero error probability. Then the DE threshold can be considered as a point of discontinuity between these two regions.

The value of the DE threshold δ is then an indicator of whether the LDPC codes ensemble decoded with a MP decoder is good or not. In particular, the DE threshold can be used to compare different systems and decide which ones are the best in terms of error correction. The most common utilization of DE in the literature is to compare different LDPC codes ensembles using the BP decoder [32]. It is used for example to optimize the degree distributions of irregular LDPC codes, or to design protograph

LDPC ensembles with good thresholds. Another less common utilization of DE is to fix the LDPC ensemble (to the same parameters (d_v, d_c) for example), and compare the thresholds of different decoders. This can be used for example to optimize the offset value in OMS decoders, or in the case of this work to analyze and optimize the injected noise within the noisy MP decoders.

Density Evolution Recursion

In this section, we describe how to implement DE using the discrete update functions defined in (2.19) and (2.20).

Let $\Theta^{(\ell)}(k)$, $k \in \mathcal{A}_C$, denote the PMF of check-to-variable messages in the ℓ^{th} iteration. Similarly, let $\Omega^{(\ell)}(k)$, $k \in \mathcal{A}_C$, denote the PMF of variable-to-check messages in the ℓ^{th} iteration. Also, let $\Omega^{(0)}(k)$, $k \in \mathcal{A}_L$, be the initial PMF of messages sent at $\ell = 0$.

The assumption of infinite block length is useful such that the PMF evolution does not depend on the iteration number. The infinite block length allows to consider that the incoming messages to a CNU or a VNU are *independent*, which is a necessary condition to ensure that the function $\Omega^{(\ell+1)} = function\left(\Theta^{(\ell)}\right)$ is the same for all iterations ℓ. In DE, we need also to consider that the all-zero codeword is sent over the channel.

1) Initialization: DE is initialized with the PMF of the channel as follows.

For the BSC with crossover probability ϵ:

$$\Omega^{(0)}(k) = \begin{cases} 1-\epsilon, & \text{if } k = C \\ \epsilon, & \text{if } k = -C \\ 0, & \text{otherwise.} \end{cases} \tag{2.22}$$

For the BI-AWGN channel with noise variance σ^2:

$$\Omega^{(0)}(k) = \begin{cases} F(k+0.5) & \text{if} \quad k = -N_q \\ F(k+0.5) - F((k-1)+0.5) & \text{if} \quad -N_q < k < +N_q \\ 1 - F((k-1)+0.5) & \text{if} \quad k = +N_q \end{cases} \tag{2.23}$$

where $F(k)$ is given by [34, 36, 37]:

$$F(k) = \frac{1}{\sqrt{2\pi}\sigma_n} \int_{-\infty}^{k} e^{-(t-\mu_n)^2/2\sigma_n^2} dt, \tag{2.24}$$

with $\sigma_n = (2/\sigma) \times \alpha$ and $\mu_n = (2/\sigma^2) \times \alpha$.

2) DE update for CNU: To compute the PMF of the output of a check-node of degree d_c, we can decompose the check-node into *elementary check-nodes*. An elementary check-node has only three edges, and its output PMF is computed with only two incoming messages which have the same PMF, because of the independence assumption. The PMF update for an elementary CN is expressed as

$$\Theta^{(\ell)}(k) = \sum_{(i,j):\Psi_c(i,j)=k} \Omega^{(\ell)}(i)\Omega^{(\ell)}(j), \quad \forall k \in \mathcal{A}_C \tag{2.25}$$

This equation is used $d_c - 2$ times in order to compute the PMF of the output of a check-node of degree d_c.

We can note that for a check-node of degree $d_c > 3$, the computational complexity of the direct PMF update would be $(\mathcal{A}_C)^{d_c-1}$, while its implementation using elementary CN updates is $(d_c-2)(\mathcal{A}_C)^2$. This represents a huge complexity reduction, and a large values of d_c is not a limitation to the practical computation of DE.

3) DE update for VNU: We compute the PMF of the output of a variable-node of degree d_v using the following relations.

For the BSC:

$$\Omega^{(\ell+1)}(k) = \sum_{(-C,i_1,\ldots,i_{d_v-1}):\Psi_v(-C,i_1,\ldots,i_{d_v-1})=k} \Omega^{(0)}(-C)\Theta^{(\ell)}(i_1)\ldots\Theta^{(\ell)}(i_{d_v-1}) + \sum_{(+C,i_1,\ldots,i_{d_v-1}):\Psi_v(+C,i_1,\ldots,i_{d_v-1})=k} \Omega^{(0)}(+C)\Theta^{(\ell)}(i_1)\ldots\Theta^{(\ell)}(i_{d_v-1}), \forall k \in \mathcal{A}_C \tag{2.26}$$

For the BI-AWGN channel:

$$\Omega^{(\ell+1)}(k) = \sum_{(t,i_1,\ldots,i_{d_v-1}):\Psi_v(t,i_1,\ldots,i_{d_v-1})=k} \Omega^{(0)}(t)\Theta^{(\ell)}(i_1)\ldots\Theta^{(\ell)}(i_{d_v-1}), \forall k \in \mathcal{A}_C \tag{2.27}$$

For the VNU, we cannot rely on the decomposition in elementary VNU, because Ψ_v cannot be factorized into a sequence of elementary operation. The complexity of DE implementation grows then rapidly with increasing d_v, and becomes a bottleneck of the DE analysis, especially for irregular LDPC codes. A solution that is usually proposed in the litterature is then to make use of the *Gaussian approximation* of DE, also known as the EXIT charts analysis of MP decoders [32]. We will not address the Gaussian approximation of DE in this work.

4) DE recursion: By combining equations (2.25) and (2.26) for the BSC, or (2.25) and (2.27) for the BI-AWGN channel, one gets the so called DE recursion, which expresses the evolution of the check-to-variable messages PMF from one iteration to another. This recursion is then computed iteratively to obtain $\Omega^{(+\infty)}$ or $\Omega^{(L_{max})}$ with L_{max} sufficiently large in a practical implementation. It can be shown that when the iteration number grows to infinity in case of unquantized channel, the PMF should converge to a dirac mass at $+\infty$ to characterize a zero error probability [38]. For the case of a quantized MP decoder, the convergence is translated to a dirac mass at the saturation

value $+N_q$. In other words, if the PMF converges to

$$\Omega^{(L_{max})}(k) = \begin{cases} 0, & \text{if } k \in \{-N_q, ..., -1, 0, 1, ..., +N_q - 1\}, \\ 1, & \text{if } k = +N_q, \end{cases} \tag{2.28}$$

then successful decoding is declared.

2.3.4 Asymptotic Bit Error Probability

The asymptotic bit error probability can be deduced from the PMF of the APPs, which is obtained from the DE equations. Let $\Gamma^{(\ell)}(k)$, $k \in \mathcal{A}_{app}$, denote the PMF of the APP at the end of the ℓ^{th} iteration. To compute $\Gamma^{(\ell)}$ for the BSC we use

$$\begin{aligned} \Gamma^{(\ell)}(k) = & \sum_{(-C,i_1,...,i_{d_v}):\Psi_a(-C,i_1,...,i_{d_v})=k} \Omega^{(0)}(-C)\Theta^{(\ell)}(i_1)...\Theta^{(\ell)}(i_{d_v}) + \\ & \sum_{(+C,i_1,...,i_{d_v}):\Psi_a(+C,i_1,...,i_{d_v})=k} \Omega^{(0)}(+C)\Theta^{(\ell)}(i_1)...\Theta^{(\ell)}(i_{d_v}), \forall k \in \mathcal{A}_{app} \end{aligned}$$

and for the BI-AWGN channel we use

$$\Gamma^{(\ell)}(k) = \sum_{(t,i_1,...,i_{d_v}):\Psi_a(t,i_1,...,i_{d_v})=k} \Omega^{(0)}(t)\Theta^{(\ell)}(i_1)...\Theta^{(\ell)}(i_{d_v}), \forall k \in \mathcal{A}_{app}$$

Let $p_e^{(\ell)}$ denote the bit error probability at iteration ℓ. Assuming the transmission of all-zero codeword, we have

$$p_e^{(\ell)} = \begin{cases} \frac{1}{2}\Gamma^{(\ell)}(0) + \sum_{i=-N_q \times d_v - C}^{-1} \Gamma^{(\ell)}(i), & \text{for the BSC,} \\ \frac{1}{2}\Gamma^{(\ell)}(0) + \sum_{i=-N_q \times (d_v+1)}^{-1} \Gamma^{(\ell)}(i), & \text{for the BI-AWGN channel.} \end{cases} \tag{2.29}$$

The evolution of $p_e^{(\ell)}$ with the iterations characterizes whether the MP decoder converges or diverges in the asymptotic limit of the codeword length. When the number of iterations ℓ goes to infinity, we obtain the asymptotic error probability $p_e^{(+\infty)}$, and when $p_e^{(+\infty)} = 0$, the decoder converges to a zero error probability and successful decoding is declared. Note that this condition is weaker than condition (2.28) presented in the previous section, but because of the properties of the functions Ψ_v and Ψ_c, both conditions are equivalent if $L_{max} = +\infty$. This work uses the condition on the bit error probability to define the DE threshold.

2.3.5 Density Evolution threshold

The DE threshold δ is defined as the point of discontinuity between these two regions: *(i)* the region of channel noise $(\epsilon, \sigma) > \delta$ in which the DE recursion does not converge to a zero error probability, and *(ii)* the region of channel noise $(\epsilon, \sigma) < \delta$ for which the DE recursion converges to a zero error probability (see (2.29)) in less than L_{max} iterations of the DE recursion. The most efficient way to compute the DE threshold is to perform a dichotomic search and stop when the bisection search interval size is lower than some precision, *e.g.* $prec = 10^{-10}$. The procedure is described in the algorithm 1.

A slight modification of the DE estimation procedure is to set a small target bit error probability η instead of targeting a zero error probability, to declare that the DE recursion converged. If η is small enough, *e.g.* $\eta = 10^{-6}$, it does not change the threshold estimation for noiseless decoders. Having $\eta > 0$ is however necessary for noisy decoders, as will be explained in the next Chapter.

In the rest of this work, we consider that the notation δ denotes the DE threshold for both the BSC or the BI-AWGN channel, and the interpretation will depend on the context. We describe the main steps to compute the DE threshold for the BSC channel in the algorithm 1, the DE threshold for the BI-AWGN channel can be easily deduced.

Algorithm 1 Computation of the DE threshold

1. [**Initialization**]
   Initialize interval limits $[\delta_1, \delta_2]$ with $\delta_1 < \delta_2$, such that DE succeeds for $\delta = \delta_1$ and fails for $\delta = \delta_2$. Further define $\delta_m = (\delta_1 + \delta_2)/2$.

2. [**While** $|\delta_2 - \delta_1| > prec$]

   (a) [**Perform DE**]

      i. [**Initialize DE**]
      DE is initialized with the equation (2.22) and $\epsilon = \delta_m$.

      ii. [**Iteration Loop**]

         A. [**Compute PMF**]
         Apply recursively the sequence of two equations (2.25) and (2.26) for L_{max} iterations.

         B. [**Break Iteration**]
         The iteration loop breaks when either the $p_e^{(\ell)} \leq \eta$ or L_{max} is reached.

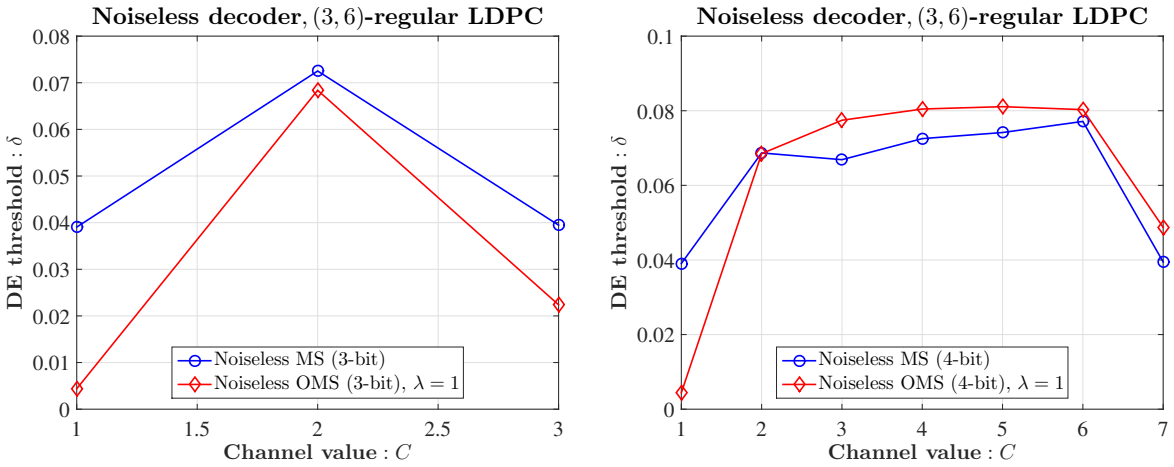
   (b) [**DE succeeds**]
   If $p_e^{(\ell)} \leq \eta$, the DE has converged and we update $\delta_1 = \delta_m$, $\delta_2 = \delta_2$ and $\delta_m = (\delta_1 + \delta_2)/2$.

   (c) [**DE fails**]
   If $p_e^{(L_{max})} > \eta$, the DE has not converged and we update $\delta_1 = \delta_1$, $\delta_2 = \delta_m$ and $\delta_m = (\delta_1 + \delta_2)/2$.

   (d) [**Tolerance**]
   Compute the size of the interval $|\delta_2 - \delta_1|$ and stops the procedure if it is smaller than the threshold tolerance (*e.g.* 10^{-10}).

3. [**Threshold**]
   $\delta = \delta_m$ is the DE threshold.

We show on Fig 2.7 two examples of the use of DE thresholds to compare different decoding algorithms. We plot the DE threshold of Min-Sum-Based decoders on the BSC, as a function of the channel value C. As can be seen the optimization of the channel value yields important gains for all algorithms. We can note also that the use of an offset is beneficial for the 4-bit MS while it is not for the 3-bit MS.



(a) Precision $q = 3$ (b) Precision $q = 4$

Fig. 2.7 DE thresholds of MS and OMS decoders using the BSC, as a function of the channel value C.

This description of the computation of the DE thresholds concludes the presentation of state of the art tools and main concepts used in the next chapter. Next chapter introduces a noisy version of the Min-Sum decoder.

Chapter 3

Noise-Against-Noise Min-Sum Decoders

In recent years, the noise has been used to study the transient hardware noise of a faulty hardware [39–44]. However, these works do not use the noise to improve the error correction performance.

Recently, the study of noisy bit-flipping (BF) decoders, the Noisy Gradient Descent BF (NGDBF) [45] and Probabilistic Gradient Descent BF (PGDBF) [46, 25], have shown that the noise could help to achieve better performance than noiseless versions of the same algorithms. The noise in these probabilistic decoders helps the decoder to escape from a local minima, and therefore improve the error correction performance.

In [47], the authors show that the introduction of randomness in the Belief Propagation (BP) algorithm can improve the performance in the error floor and the waterfall region. A noisy version of the Min-Sum decoder over the BSC and the BI-AWGN channel is studied in [35, 48, 40].

In this work, we investigate a modification of MS decoders, with the goal of improving the error correction performance when the number of precision bits is as low as 3 or 4, which could be the case when very low implementation complexity is

required by the application. Our modification, named Noise-against-Noise Min-Sum (NAN-MS) decoders, consists in injecting a deliberate random perturbation during the decoding process, to modify the dynamics of the decoder, and help to correct channel error events that would not be corrected by a deterministic implementation.

We investigate the asymptotic behavior of our NAN-MS decoders using noisy density evolution (DE) techniques, as introduced in [43, 35] for the study of fault-tolerance of BP and Min-Sum decoders. Using the decoding thresholds obtained with the noisy DE, we propose to study the best localization for the noise injection and to optimize the noise model parameters. Our study shows that in the case of very low precision, the NAN-MS can surpass the error correction performance of the MS decoders, for various regular and irregular LDPC code families. This conclusion is corroborated by finite length Monte Carlo simulations.

Another aspect investigated in this work is the implementation of the NAN-MS decoders avoiding the use of Random Generators (RGs). This is because the implementation of RGs in a hardware realization is very expensive [49, 50]. From the main conclusions obtained of the NAN-MS decoders, two implementations are proposed that avoid using RGs maintaining almost the same error correction performance.

The outline of this chapter is as follows. The first section briefly discusses the noiseless OMS-based decoders. The second section discusses the probabilistic error model used to inject noise to quantized decoders. In the third section, we present three methods to inject noise in order to implement NAN-MS decoders. In the fourth section, we present the noisy density evolution equations of the three methods to inject noise, and we explain how to optimize the model parameters with density evolution. In the fifth and sixth sections, we present the results for the optimization of the noise models for regular and irregular LDPC codes, respectively. The seventh section shown finite length performance validation of the gains obtained with the proposed NAN-MS

decoders. In the eighth section, we explain how to implement NAN-MS decoders, and we define a deterministic decoder called Modified Offset Min-Sum (M-OMS). In the ninth section, we present another alternative to implement the NAN-MS decoders. We propose to use the M-OMS decoder and an optimized offset vector. And the tenth section concludes this chapter.

3.1 Noiseless Offset Min-Sum-Based Decoders

In this section, we will redefine the update rules of the quantized OMS presented in Chapter 2. The main reason to refine the VNU and CNU of the OMS decoder is because such decoder is sub-optimal. In order to see why the decoder is sub-optimal, let us take the example of an OMS decoder quantized on $q = 3$ bits of precision with offset value $\lambda = 1$, see (2.19) and (2.20). We can note the offset applied in CNUs only gives us the possibility to use five values ($\{-2, -1, 0, +1, +2\}$) instead of the seven values of $\mathcal{A}_C$, in other words, the outgoing messages $m^{(\ell)}_{c_m \to v_n}$ from the CNU belong to the set $\mathcal{A}_C \backslash \{-N_q = -3, +N_q = +3\} = \{-2, -1, 0, +1, +2\}$. It can be clearly noted that all combinations that can be obtained from $q = 3$ bits are not used (the same analysis can be done for precision $q = 4$).

In order to allow c-to-v messages to use all the values of $\mathcal{A}_C$, the updates rules of the quantized OMS decoder need to be changed. Analyzing the update rule of a CN, the CNU can be written in two equivalent ways:

1) Classical method: The offset is applied after the calculation of the minimum value of all incoming messages.

$$\left| m^{(\ell)}_{c_m \to v_n} \right| = \max \left\{ \min_{v \in \mathcal{V}(c_m) \backslash \{v_n\}} \left| m^{(\ell)}_{v \to c_m} \right| - \lambda, 0 \right\}.$$

2) Equivalent method: The offset is applied to each message and then the minimum value is computed.

$$\left|m_{c_m\to v_n}^{(\ell)}\right| = \min_{v\in\mathcal{V}(c_m)\setminus\{v_n\}} \left(\max\left(\left|m_{v\to c_m}^{(\ell)}\right| - \lambda, 0\right)\right).$$

Therefore, we can move the offset from CNs to VNs and take to integrate the offset inside the VNU so that v-to-c messages take all values of $\mathcal{A}_C$, and thus, c-to-v messages take also all values of $\mathcal{A}_C$. Hence the CNU of OMS-based decoders can be rewritten as

$$m_{c_m\to v_n}^{(\ell)} = \Psi_c\left(\left\{m_{v\to c_m}^{(\ell)}\right\}_{v\in\mathcal{V}(c_m)\setminus\{v_n\}}\right) = \left(\prod_{v\in\mathcal{V}(c_m)\setminus\{v_n\}} \operatorname{sign}\left(m_{v\to c_m}^{(\ell)}\right)\right)\left(\min_{v\in\mathcal{V}(c_m)\setminus\{v_n\}}\left(\left|m_{v\to c_m}^{(\ell)}\right|\right)\right). \tag{3.1}$$

Let us denote by λ_v the offset value applied at the VNs. Moving the offset from CNs to VNs, the VNU of OMS-based decoders can be given by

$$m_{v_n\to c_m}^{(\ell+1)} = \Psi_v\left(I_n, \left\{m_{c\to v_n}^{(\ell)}\right\}_{c\in\mathcal{V}(v_n)\setminus\{c_m\}}\right) = \Lambda\left(m_{v_n\to c_m}^{(\ell+1),U}\right), \tag{3.2}$$

where the function $\Lambda(.)$ is defined by $\Lambda(a) = \operatorname{sign}(a)\times\mathcal{S}\left(\max(|a|-\lambda_v, 0), N_q\right)$, and the unsaturated v-to-c message $m_{v_n\to c_m}^{(\ell+1),U}$ is computed as

$$m_{v_n\to c_m}^{(\ell+1),U} = I_n + \sum_{c\in\mathcal{V}(v_n)\setminus\{c_m\}} m_{c\to v_n}^{(\ell)}.$$

We can observe that the offset λ_v is subtracted before saturation to allow $-N_q$ and N_q values in the variable-to-check message. The special case of $\lambda_v = 0$ corresponds to the MS decoder. The alphabet of the unsaturated v-to-c message $m_{v_n\to c_m}^{(\ell+1),U}$, denoted $\mathcal{A}_U$, is defined as $\mathcal{A}_U = \{-N_q\times d_v, ..., -1, 0, +1, ..., +N_q\times d_v\}$. Moving the offset from CNs to VNs does not alter the calculation of the APP $\gamma_n^{(\ell)}$ given in (2.21).

It is worth noting that in the literature the offset is often applied at the CN. Applying the offset at the VN or at the CN is equivalent only when the saturation function is not used since $\min_{i=1,\dots,n}(|x_i| - \lambda) = \min_{i=1,\dots,n}(|x_i|) - \lambda$.

From now on, when we refer to the MS and OMS decoders, we will refer to the decoders defined by the equations (3.1), (3.2), and (2.21).

3.2 Noise Model for Noise-against-Noise Min-Sum Decoders

3.2.1 Probabilistic Error Model for Noise-against-Noise Min-Sum Decoders

In order to define a NAN-MS decoder, we first express the constraints on the noise models, and then we present a noise model which will be used to perturb noiseless decoders. Let us denote by $\tilde{\mathcal{A}}_C$ the noisy message alphabet. Noisy messages $\tilde{m}^{(\ell)}_{v\to c} \in \tilde{\mathcal{A}}_C$ and $\tilde{m}^{(\ell)}_{c\to v} \in \tilde{\mathcal{A}}_C$ are obtained after corrupting the noiseless messages $m^{(\ell)}_{v\to c}$ and $m^{(\ell)}_{c\to v}$, respectively, with noise. To simplify the notations in this section, we use the notation m to denote any c-to-v message $m^{(\ell)}_{c\to v}$ at iteration ℓ or any v-to-c message $m^{(\ell)}_{v\to c}$, and let s_β be the sign of β.

In order to be able to perform DE analysis of NAN-MS decoders [51], the considered noise models need to be memoryless, *i.e.* the noise models are independent on the data streams processed by the decoders. Also, the considered noise models have to satisfy a symmetry condition defined as follows

$$\Pr(\tilde{m} = \beta_2 | m = \beta_1) = \Pr(\tilde{m} = -\beta_2 | m = -\beta_1), \quad \forall (\beta_1, \beta_2) \in \mathcal{A}_C \times \tilde{\mathcal{A}}_C.$$

When memoryless and symmetric noise models are used to perturb noiseless decoders, the noisy-VNU and noisy-CNU are symmetric, allowing to use the all-zero codeword assumption and the independence assumption necessary in DE [27].

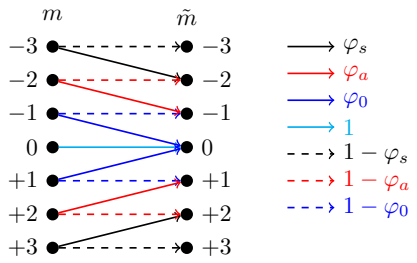
Now, let us denote by $\Upsilon : \mathcal{A}_N \to \tilde{\mathcal{A}}_N$ the symmetric function which transforms a noiseless message $m^{(\ell)} \in \mathcal{A}_C$ into a noisy message $\tilde{m}^{(\ell)} \in \tilde{\mathcal{A}}_C$. Unless otherwise stated, the alphabet of the noise model will be the one of the messages, *i.e.* $\mathcal{A}_N = \tilde{\mathcal{A}}_N = \mathcal{A}_C$. The noise model Υ is defined by the conditional probability density function (PDF) $\Pr(\tilde{m}|m)$ wich is denoted by $p_\Upsilon(m, \tilde{m})$. We propose in this work to analyze a noise model, defined in the following way:

$$\tilde{m} = \Upsilon(m = \beta) = \begin{cases} \beta, & \text{with prob. } 1-\varphi_s, \quad \beta \in \{\pm N_q\} \\ s_\beta.(|\beta|-1), & \text{with prob. } \varphi_s, \quad \beta \in \{\pm N_q\} \\ \beta, & \text{with prob. } 1-\varphi_a, \quad \beta \notin \{\pm N_q, \pm 1, 0\} \\ s_\beta.(|\beta|-1), & \text{with prob. } \varphi_a, \quad \beta \notin \{\pm N_q, \pm 1, 0\} \\ \beta, & \text{with prob. } 1-\varphi_0, \quad \beta \in \{\pm 1\} \\ s_\beta.(|\beta|-1), & \text{with prob. } \varphi_0, \quad \beta \in \{\pm 1\}. \end{cases} \tag{3.3}$$

This noise model is parametrized by three different transition probabilities, denoted $\boldsymbol{\varphi} = (\varphi_s, \varphi_a, \varphi_0)$. φ_s denotes the conditional PDF $p_\Upsilon(m = N_q, \tilde{m} = N_q - 1)$, φ_a denotes the conditional PDF $p_\Upsilon(m = \beta, \tilde{m} = \beta - 1)$, $\forall \beta \in \{+2, ..., +(N_q - 1)\}$, and finally φ_0 denotes the conditional PDF $p_\Upsilon(m = +1, \tilde{m} = 0)$.

The rationale behind this model is to implement a probabilistic offset. Let us discuss the case of $\varphi_s = \varphi_a = \varphi_0$. The MS and the OMS with $\lambda_v = 1$ can be obtained as special cases of the noise model Υ, *i.e.* $\boldsymbol{\varphi} = (0, 0, 0)$ for the MS and $\boldsymbol{\varphi} = (1, 1, 1)$ for the OMS. For other values of the transition probabilities, the offset is only applied from time to time during the decoding iterations, leading to a probabilistic weighted combination of a deterministic MS and a deterministic OMS.

We have also introduced two other probabilities φ_s and φ_0, in order to study the effect of the noise on the extreme values of the message alphabet. A special case occurs when $m = 1$ because it will be changed to $\tilde{m} = 0$ or it will keep its value $\tilde{m} = 1$. Since the value $\tilde{m} = 0$ propagates no information about the bit value, φ_0 has to be analyzed differently than φ_a. Additionally, in quantized decoders, all values which are greater than N_q at the VNU are saturated to N_q, according to (3.2). As a result, many more configurations of the VNU states lead to an output message $m = N_q$ compared to other values of m, and φ_s should also be analyzed differently than φ_a. As an example, Υ is depicted in Fig. 3.1 for $(q = 3, N_q = 3)$.



$\tilde{m}$ / m	-3	-2	-1	0	$+1$	$+2$	$+3$
-3	$1-\varphi_s$	0	0	0	0	0	0
-2	φ_s	$1-\varphi_a$	0	0	0	0	0
-1	0	φ_a	$1-\varphi_0$	0	0	0	0
0	0	0	φ_0	1	φ_0	0	0
$+1$	0	0	0	0	$1-\varphi_0$	φ_a	0
$+2$	0	0	0	0	0	$1-\varphi_a$	φ_s
$+3$	0	0	0	0	0	0	$1-\varphi_s$

Fig. 3.1 LUT representation and the mapping used for the noise model Υ.

The noise model Υ could then be used to determine whether a NAN-MS can outperform a noiseless decoder.

3.3 Noise-against-Noise Min-Sum Decoders

The first issue that we would like to analyze is the localization of the noise injection inside the decoder flow. Indeed, an iterative decoder is composed of several sequential steps, and the results could be different depending on when the noise is injected. We decided to restrict the possibilities of noise injection to the main steps of the algorithm, that are the VNU and the CNU steps. In this section, we present different versions

of the NAN-MS decoders, depending on the choice of the decoding step in which we inject the noise.

The noise can be injected in three different ways: *(i)* during the VNU processing, *(ii)* after the VNU processing, or *(iii)* after the CNU processing. We decide in this work to analyze the NAN-MS decoders with noise injection only in one of these steps, although one could easily extend the analysis by injecting noise during more than one step of the decoding iteration.

3.3.1 Noise Outside the Variable Node Update

The first choice is obtained by perturbing the outgoing message from the VNU, after the processing of (3.2) is completed. The update rule for the corresponding noisy-VNU is given by

$$\tilde{m}_{v_n \to c_m}^{(\ell+1)} = \Upsilon\left(\mathcal{S}\left(I_n + \sum_{c \in \mathcal{V}(v_n)\setminus\{c_m\}} m_{c \to v_n}^{(\ell)}\right)\right) \tag{3.4}$$

Note that the noise perturbs saturated v-to-c messages, which belong to $\mathcal{A}_C$, and the noise model of (3.3) can be used directly. We define this method to inject noise as *Noise Outside the VNU* (NOV). Fig. 3.2 depicts the simplest case of VNU, CNU, and the NOV model.

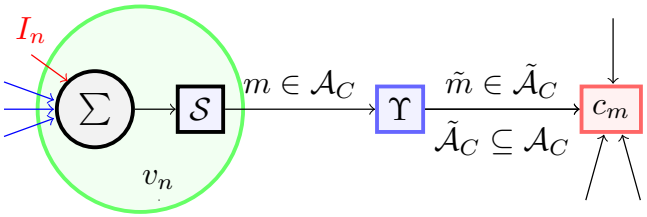


Fig. 3.2 Noise injected after the VNU processing.

3.3.2 Noise Inside the Variable Node Update

The second method to inject noise is implemented perturbing unsaturated v-to-c messages, before the use of function $\mathcal{S}$ in (3.2). In this case, the update rule for the noisy-VNU is given by

$$\tilde{m}_{v_n \to c_m}^{(\ell+1)} = \mathcal{S}\left(\Upsilon\left(I_n + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} m_{c \to v_n}^{(\ell)}\right)\right) \tag{3.5}$$

We label this method to inject noise as *Noise Inside the VNU* (NIV). Since Υ perturbs unsaturated values, the alphabet of the noise model $\mathcal{A}_N$ is larger than $\mathcal{A}_C$, and equal to $\mathcal{A}_N = \tilde{\mathcal{A}}_N = \{-(2^{(q-1)}-1) \times d_v, \ldots, -1, 0, +1, \ldots, +(2^{(q-1)}-1) \times d_v\}$, *i.e.* $\mathcal{A}_N = \tilde{\mathcal{A}}_N = \mathcal{A}_U$. It follows that there is no advantage in differentiating the transition probabilities φ_s and φ_a, as the largest values of $\mathcal{A}_N$ will be saturated with function $\mathcal{S}$. Throughout the work, we will then assume that $\varphi_s = \varphi_a$ for the NIV model.

The example of the NIV model with the simplest case of VNU and CNU is depicted on Fig. 3.3.

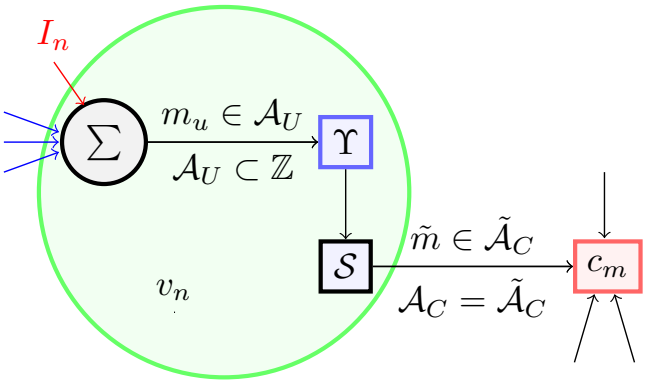


Fig. 3.3 Noise injected during the VNU processing.

3.3.3 Noise Outside the Check Node Update

The third method to inject noise perturbs the outgoing c-to-v messages from the CNU, after the processing is completed. Note that because during the CNU, the min operations leave the messages in the same alphabet $\mathcal{A}_C$, there is no need to consider two cases, like for the VNU. The update rule for a noisy-CNU is given by

$$\tilde{m}^{(\ell)}_{c_m \to v_n} = \Upsilon\left(m^{(\ell)}_{c_m \to v_n}\right) \tag{3.6}$$

where $m^{(\ell)}_{c_m \to v_n}$ is obtained with (3.1). We label this method as *Noise Outside the CNU* (NOC). Fig. 3.4 shows the simplest case of VNU, CNU, and the NOC model.

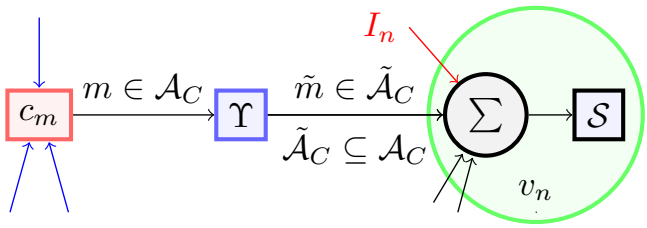


Fig. 3.4 Noise injected after the CNU processing.

The APP update of NAN-MS decoders depends on the model used to inject noise, *i.e.* NIV, NOV, or NOC. For the NOC model, the noisy c-to-v messages are used to compute the APP, hence, the APP update is given by $\gamma^{(\ell)}_n = I_n + \sum_{c \in \mathcal{V}(v_n)} \tilde{m}^{(\ell)}_{c \to v_n}$. In the case of NIV and NOV model, c-to-v messages are not corrupted, then the APP update is computed as $\gamma^{(\ell)}_n = I_n + \sum_{c \in \mathcal{V}(v_n)} m^{(\ell)}_{c \to v_n}$.

3.4 Noisy Density Evolution for NAN-MS decoders

We are interested in obtaining and comparing the DE thresholds of NAN-MS decoders with the DE thresholds of noiseless decoders.

Noisy DE recursions can be deduced from noiseless DE recursions and the noise model description of (3.3). To deduce the noisy DE equations, let $\tilde{\Theta}^{(\ell)}(k)$, $k \in \mathcal{A}_C$, denote the PMF of noisy check-to-variable messages in the ℓ^{th} iteration. Similarly, let $\tilde{\Omega}^{(\ell)}(k)$, $k \in \mathcal{A}_C$, denote the PMF of noisy variable-to-check messages in the ℓ^{th} iteration. We consider that the all-zero codeword is sent over the BI-AWGN channel.

3.4.1 Noisy Density Evolution for the NOV model

Initialization

like the case of noiseless quantized Min-Sum-Based decoders, the initialization of the noisy DE for NAN-MS decoders is given by equation (2.23).

Noiseless DE update for CNU

The input of a CNU is the PMF of the noisy messages going out of a noisy VNU, *i.e.* $\tilde{\Omega}^{(\ell)}$. For a CN of degree d_c, $\Theta_{d_c}^{(\ell)}$ is given by

$$\Theta_{d_c}^{(\ell)}(k) = \sum_{(i_1,\ldots,i_{d_c-1}):\Psi_c(i_1,\ldots,i_{d_c-1})=k} \tilde{\Omega}^{(\ell)}(i_1)\ldots\tilde{\Omega}^{(\ell)}(i_{d_c-1}),\ \forall k \in \mathcal{A}_C. \tag{3.7}$$

Considering the different connection degrees of CNs, we have

$$\Theta^{(\ell)}(k) = \sum_{d_c=2}^{d_{c,max}} \rho_{d_c} \times \Theta_{d_c}^{(\ell)}(k), \quad \forall k \in \mathcal{A}_C \tag{3.8}$$

Similar to the noiseless DE, we can decompose a CNU of NAN-MS decoders into elementary CNUs to compute (3.7).

Noisy DE update for VNU

Similarly to the calculation made at the VN for the noiseless DE, we compute the PMF of the output of a VN of degree d_v, *i.e.* $\Omega_{d_v}^{(\ell+1)}$. The input of noisy DE for the VNU are PMF computed with (3.8).

$$\Omega_{d_v}^{(\ell+1)}(k) = \sum_{(t,i_1,\ldots,i_{d_v-1}):\Psi_v(t,i_1,\ldots,i_{d_v-1})=k} \Omega^{(0)}(t)\Theta^{(\ell)}(i_1)\ldots\Theta^{(\ell)}(i_{d_v-1}),\ \forall k \in \mathcal{A}_C. \quad (3.9)$$

Then the noise effect is added at the output of the VNU.

$$\tilde{\Omega}_{d_v}^{(\ell+1)}(k) = \sum_{i\in\mathcal{A}_C} \Omega_{d_v}^{(\ell+1)}(i) \times p_{\Upsilon}(i,k),\ \forall k \in \mathcal{A}_C, \quad (3.10)$$

where p_{Υ} is the transition probability of the VN noise Υ.

Then the effect of the different connection degrees of VNs is considered using the following relation

$$\tilde{\Omega}^{(\ell+1)}(k) = \sum_{d_v=2}^{d_{v,max}} \lambda_{d_v} \times \tilde{\Omega}_{d_v}^{(\ell+1)}(k), \quad \forall k \in \mathcal{A}_C \quad (3.11)$$

Fig. 3.5 depicts a simple Tanner graph with a single CN, a single VN, and the PMF of noiseless and noisy messages.

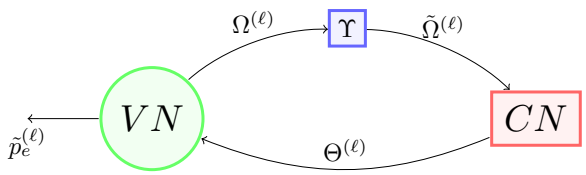


Fig. 3.5 Concept of Noisy DE calculation for the NOV model

3.4.2 Noisy Density Evolution for the NIV model

Initialization

The initialization of DE does not change for the NIV model, hence, (2.23) is used.

Noiseless DE update for CNU

Similarly to the calculation made at the CNU for the NOV model, we compute $\Theta^{(\ell)}(k)$, $\forall k \in \mathcal{A}_C$ with 3.7 and (3.8).

Noisy DE update for VNU

For the NIV model, we know that Υ perturbs unsaturated values. For this reason, we first compute the PMF of unsaturated v-to-c messages of a VN of degree d_v, *i.e.* $\Omega_{d_v}^{(\ell+1),U}$, with the following equation

$$\Omega_{d_v}^{(\ell+1),U}(k) = \sum_{(t,i_1,\ldots,i_{d_v-1}):\Psi_v(t,i_1,\ldots,i_{d_v-1})=k} \Omega^{(0)}(t)\Theta^{(\ell)}(i_1)\ldots\Theta^{(\ell)}(i_{d_v-1}),\ \forall k \in \mathcal{A}_U. \tag{3.12}$$

Second, the noise effect is added to the PMF of unsaturated v-to-c messages to obtain the corrupted PMF

$$\tilde{\Omega}_{d_v}^{(\ell+1)}(k) = \sum_{i\in\mathcal{A}_U} \Omega_{d_v}^{(\ell+1),U}(i) \times p_\Upsilon(i,k),\ \forall k \in \mathcal{A}_U, \tag{3.13}$$

Finally, the saturation effect on the corrupted PMF is given by

$$\tilde{\Omega}^{(\ell+1)}(k) = \begin{cases} \sum\limits_{i=-N_q\times d_v}^{-N_q} \tilde{\Omega}^{(\ell+1)}(i), & k = -N_q \\ \tilde{\Omega}^{(\ell+1)}(k), & \forall k \in \mathcal{A}_C \backslash \{\pm N_q\} \\ \sum\limits_{i=+N_q}^{+N_q\times d_v} \tilde{\Omega}^{(\ell+1)}(i), & k = +N_q \end{cases} \tag{3.14}$$

Similar to the NOV model, the effect of the different connection degrees of VNs is computed with (3.11).

3.4.3 Noisy Density Evolution for the NOC model

Initialization

Similar to the NOV model and the NIV model, the initialization of DE does not change, therefore, equation (2.23) is used to initialize the noisy DE.

Noisy DE update for CNU

The input of a noisy CNU is the PMF of the noiseless messages going out of a noiseless VNU, *i.e.* $\Omega^{(\ell)}$. The following relation is used the PMF of a CN of degree d_c

$$\Theta_{d_c}^{(\ell)}(k) = \sum_{(i_1,\ldots,i_{d_c-1}):\Psi_c(i_1,\ldots,i_{d_c-1})=k} \Omega^{(\ell)}(i_1)\ldots\Omega^{(\ell)}(i_{d_c-1}),\ \forall k \in \mathcal{A}_C. \tag{3.15}$$

To make into account the effect of noise on the PMF of the output messages of a CN, the following relation is used

$$\tilde{\Theta}_{d_c}^{(\ell)}(k) = \sum_{i\in\mathcal{A}_C} \Theta_{d_c}^{(\ell)}(i) \times p_\Upsilon(i,k), \quad \forall k \in \mathcal{A}_C, \tag{3.16}$$

where p_Υ is the transition probability of the CN noise (which can be identical to the VN probability).

To make into account the different connection degrees of CNs, we use

$$\tilde{\Theta}^{(\ell)}(k) = \sum_{d_c=2}^{d_{c,max}} \rho_{d_c} \times \tilde{\Theta}_{d_c}^{(\ell)}(k), \quad \forall k \in \mathcal{A}_C \tag{3.17}$$

Similar to the noiseless DE, we can use the decomposition of CNUs into elementary CNUs.

Noiseless DE update for VNU

Similarly to the calculation made at the VN for the noiseless DE, we compute the PMF of the output of a VN of degree d_v, *i.e.* $\Omega_{d_v}^{(\ell+1)}$. The input of noisy DE for the VNU are corrupted PMF computed with (3.19).

$$\Omega_{d_v}^{(\ell+1)}(k) = \sum_{(t,i_1,\ldots,i_{d_v-1}):\Psi_v(t,i_1,\ldots,i_{d_v-1})=k} \Omega^{(0)}(t)\tilde{\Theta}^{(\ell)}(i_1)\ldots\tilde{\Theta}^{(\ell)}(i_{d_v-1}),\ \forall k \in \mathcal{A}_C. \quad (3.18)$$

Then the effect of the different connection degrees of VNs is considered using the following relation

$$\Omega^{(\ell+1)}(k) = \sum_{d_v=2}^{d_{v,max}} \lambda_{d_v} \times \Omega_{d_v}^{(\ell+1)}(k), \quad \forall k \in \mathcal{A}_C \quad (3.19)$$

3.4.4 Noisy DE recursion

Once the noisy DE update for VNUs and CNUs are defined, the rest of the DE principle is unchanged compared to the noiseless DE of Chapter 2.

3.4.5 Bit Error Probability

Let $\tilde{p}_e^{(\ell)}$ denote the bit error probability at iteration ℓ for NAN-MS decoders, $\tilde{p}_e^{(\ell)}$ is computed from the PMF of all incoming messages to a VN in the ℓ^{th} iteration, and it is defined as

$$\tilde{p}_e^{(\ell)} = \frac{1}{2}\tilde{\Gamma}^{(\ell)}(0) + \sum_{i=-N_q\times(d_v+1)}^{-1} \tilde{\Gamma}^{(\ell)}(i), \quad (3.20)$$

where $\tilde{\Gamma}^{(\ell)}(k)$, $k \in \mathcal{A}_{app}$, denotes the PMF of the APP at the end of the ℓ^{th} iteration for NAN-MS decoders. For the three models presented in this work, *i.e.* NIV, NOV,

and NOC, $\tilde{\Gamma}^{(\ell)}$ is computed as follows

$$\tilde{\Gamma}^{(\ell)}(k) = \sum_{d_v=2}^{d_{v,max}} \lambda_{d_v} \times \tilde{\Gamma}_{d_v}^{(\ell)}(k), \quad \forall k \in \mathcal{A}_{app}$$

In the case of the NOV model and NIV model, $\tilde{\Gamma}_{d_v}^{(\ell)}$ is given by

$$\tilde{\Gamma}_{d_v}^{(\ell)} = \sum_{(t,i_1,...,i_{d_v}):\Psi_a(t,i_1,...,i_{d_v})=k} \Omega^{(0)}(t)\Theta^{(\ell)}(i_1)...\Theta^{(\ell)}(i_{d_v}), \quad \forall k \in \mathcal{A}_{app}$$

whereas in the case of the NOC model, we have

$$\tilde{\Gamma}_{d_v}^{(\ell)} = \sum_{(t,i_1,...,i_{d_v}):\Psi_a(t,i_1,...,i_{d_v})=k} \Omega^{(0)}(t)\tilde{\Theta}^{(\ell)}(i_1)...\tilde{\Theta}^{(\ell)}(i_{d_v}), \quad \forall k \in \mathcal{A}_{app}$$

In the asymptotic limit of the code-length, $\tilde{p}_e^{(\ell)}$ is the bit error probability of a noisy decoder at the ℓ^{th} iteration.

Contrary to the noiseless case $\tilde{p}_e^{(+\infty)}$ is not necessarily equal to zero when the noisy DE converges and corrects the channel noise. It depends mainly on the chosen error model and the computing units to which it is applied.

A lower bound on the asymptotic error probability can be computed. In [35], the authors have proposed two error models for a quantized noisy MS decoder, and confirmed that $\tilde{p}_e^{(+\infty)}$ is bounded away from zero.

3.4.6 Noisy Density Evolution threshold

The noisy-DE threshold is denoted $\tilde{\delta}$. It is a function of the code family, parametrized by its degree distribution $(\lambda(x), \rho(x))$, of the class of NAN-MS decoder as introduced in the previous section, of the number of precision bits q, of the value of the channel gain factor α, and the values of the transition probabilities of the noise model $(\varphi_s, \varphi_a, \varphi_0)$.

Similar to the noiseless case, the dichotomic search is used to compute noisy-DE thresholds. The DE estimation procedure is performed choosing a target residual error probability η, and declaring convergence of the noisy DE recursion when $\tilde{p}_e^{(+\infty)}$ is less than or equal to η, *e.g.* 10^{-6}. The algorithm 2 describes the procedure to compute $\tilde{\delta}$ for the NOV model.

Algorithm 2 Computation of the noisy DE threshold

1. [**Initialization**]
   Initialize interval limits $[\tilde{\delta}_1, \tilde{\delta}_2]$ with $\tilde{\delta}_1 < \tilde{\delta}_2$, such that DE succeeds for $\tilde{\delta} = \tilde{\delta}_1$ and fails for $\tilde{\delta} = \tilde{\delta}_2$. Further define $\tilde{\delta}_m = (\tilde{\delta}_1 + \tilde{\delta}_2)/2$.

2. [**While** $|\tilde{\delta}_2 - \tilde{\delta}_1| > prec$]

   (a) [**Perform noisy DE**]

      i. [**Initialize noisy DE**]
      Noisy DE is initialized with the equation (2.23) and $\sigma = \tilde{\delta}_m$

      ii. [**Iteration Loop**]

         A. [**Compute PMF**]
         Apply recursively the sequence of five equations (3.7), (3.8), (3.9), (3.10), and (3.11) for L_{max} iterations.

         B. [**Break Iteration**]
         The iteration loop breaks when either the $p_e^{(\ell)} \leq \eta$ or L_{max} is reached.

   (b) [**Noisy DE succeeds**]
   If $p_e^{(\ell)} \leq \eta$, the noisy DE has converged and we update $\tilde{\delta}_1 = \tilde{\delta}_m$, $\tilde{\delta}_2 = \tilde{\delta}_2$ and $\tilde{\delta}_m = (\tilde{\delta}_1 + \tilde{\delta}_2)/2$.

   (c) [**Noisy DE fails**]
   If $p_e^{(L_{max})} > \eta$, the noisy DE has not converged and we update $\tilde{\delta}_1 = \tilde{\delta}_1$, $\tilde{\delta}_2 = \tilde{\delta}_m$ and $\tilde{\delta}_m = (\tilde{\delta}_1 + \tilde{\delta}_2)/2$.

   (d) [**Tolerance**]
   Compute the size of the interval $|\tilde{\delta}_2 - \tilde{\delta}_1|$ and stops the procedure if it is smaller than the threshold tolerance (*e.g.* 10^{-10}).

3. [**Threshold**]
   $\tilde{\delta} = \tilde{\delta}_m$ is the noisy-DE threshold.

We can use $\tilde{\delta}$ to jointly optimize the noise model parameters and the channel gain factor, for a fixed NAN-MS decoder, a fixed precision, and a fixed degree distribution:

$$(\varphi_s^*, \varphi_a^*, \varphi_0^*, \alpha^*) = \arg \max_{(\varphi_s, \varphi_a, \varphi_0, \alpha)} \left\{ \tilde{\delta}\left(\lambda(x), \rho(x), q, \alpha, (\varphi_s, \varphi_a, \varphi_0)\right) \right\} \tag{3.21}$$

The optimization of the transition probabilities of the model Υ, and the channel gain factor is made using a *greedy* algorithm which computes a local maximum DE threshold. For noiseless decoders, the optimum channel gain factor α^* is computed performing a grid-search.

3.5 Asymptotic Analysis of NAN-MS Decoders for Regular LDPC codes

In this section, we consider the ensemble of (d_v, d_c)-regular LDPC codes with variable-node degree $d_v \in \{3, 4\}$ and code rate $R \in \{1/2, 3/4\}$, and quantized decoders with $q \in \{3, 4, 5\}$.

3.5.1 Optimization of the Channel Gain Factor of Noiseless Decoders

The DE thresholds of the noiseless MS and OMS decoders are given in Table 3.1. For the noiseless decoders, the optimization (3.21) is reduced to the optimum channel gain factor α^*. It can be seen that the OMS is almost always superior to the MS for the considered cases, which was expected. The only exception is for regular $(d_v, d_c) = (3, 6)$ and $(d_v, d_c) = (3, 12)$ LDPC codes with low precision messages $q = 3$, for which the MS threshold is better than the OMS's. We can also note that the largest difference is obtained for $d_v = 4$ LDPC codes and quantization $q \in \{4, 5\}$.

Table 3.1 DE thresholds of noiseless MS decoders and noiseless OMS decoders with offset value $\lambda_v = 1$.

(d_v, d_c)-regular LDPC code, BI-AWGN channel									
		$(d_v = 3, d_c = 6)$		$(d_v = 3, d_c = 12)$		$(d_v = 4, d_c = 8)$		$(d_v = 4, d_c = 16)$	
q	λ_v	α^*	δ_{db}	α^*	δ_{db}	α^*	δ_{db}	α^*	δ_{db}
3 bits	0	0.9375	**1.7888**	0.625	**2.7316**	0.8125	2.7360	0.6875	3.1550
	1	1.0625	2.2039	0.9375	3.1343	1.25	**2.3219**	0.9375	**3.0632**
4 bits	0	2.0	1.6437	1.25	2.5646	1.625	2.5389	1.375	2.9441
	1	1.875	**1.3481**	1.5	**2.4484**	1.75	**1.7509**	1.5	**2.5292**
5 bits	0	4.0	1.6132	2.5	2.5268	3.25	2.4948	2.75	2.8991
	1	2.625	**1.2154**	2.25	**2.3040**	2.0	**1.7061**	1.875	**2.4606**

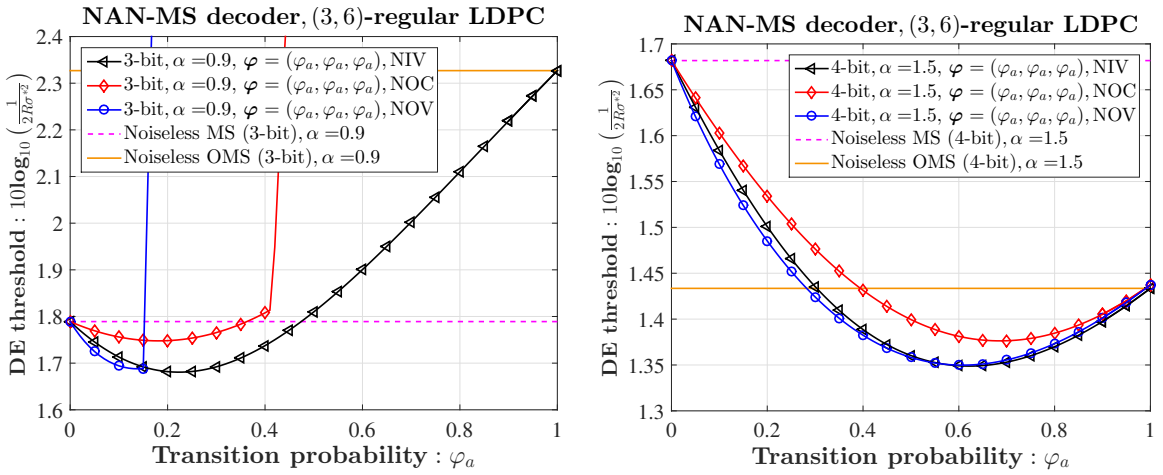
3.5.2 Localization of the Noise Injection in NAN-MS Decoders

In this section, we analyze and study the localization of the noise in NAN-MS decoders without performing the optimization of the channel gain factor, *i.e.* α. We restrict our analysis for the ensemble of (d_v, d_c)-regular LDPC codes with $d_v \in \{3, 4\}$, $R = 1/2$, and precision $q \in \{3, 4\}$, setting $\alpha = 0.9$ for $q = 3$-bit NAN-MS decoders and $\alpha = 1.5$ for $q = 4$-bit NAN-MS decoders.

We consider a special case of the noise model Υ doing the transition probabilities φ_s and φ_0 equal to φ_a, hence, we have $\boldsymbol{\varphi} = (\varphi_a, \varphi_a, \varphi_a)$. The MS decoder is obtained with $\boldsymbol{\varphi} = (0, 0, 0)$ for the NOV, NIV, and NOC model, for this reason DE thresholds of NAN-MS decoders are equal to the DE threshold of the MS decoder as is shown in Fig. 3.6 and Fig. 3.7. The OMS decoder with offset $\lambda_v = 1$, which is implemented with equations (3.1) and (3.2), is also implemented using the NIV model with $\boldsymbol{\varphi} = (1, 1, 1)$. With the NIV model the v-to-c and c-to-v messages belong to the message alphabet $\mathcal{A}_C$. The NOC model with $\boldsymbol{\varphi} = (1, 1, 1)$ implements the OMS decoder with offset $\lambda = 1$ described in Chapter 2 (equations (2.19) and (2.20)), *i.e.* the offset is applied at the CNUs obtaining c-to-v messages belong to the message alphabet $\mathcal{A}_C \backslash \{-N_q, +N_q\}$. In

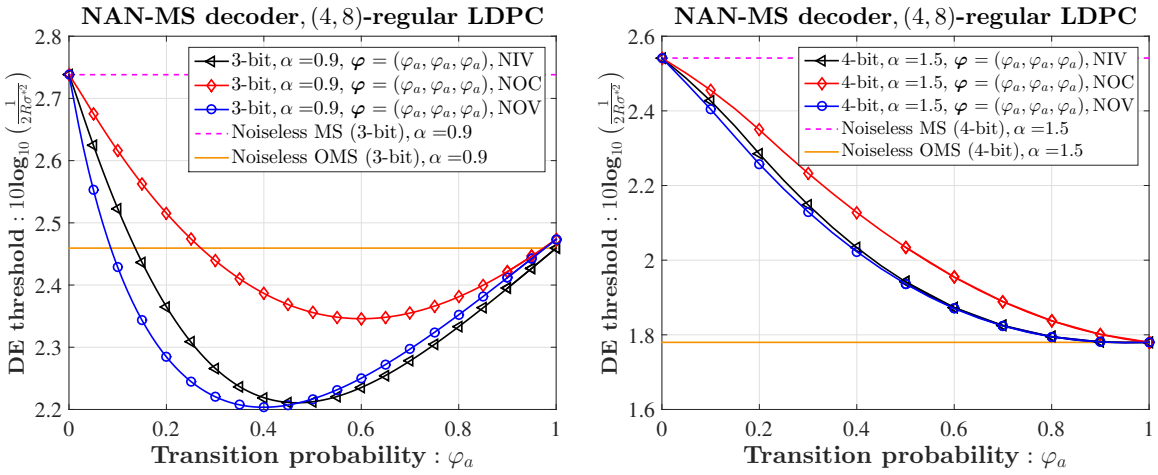
the case of the NOV model with $\boldsymbol{\varphi} = (1,1,1)$, the v-to-c messages also belong to the message alphabet $\mathcal{A}_C \backslash \{-N_q, +N_q\}$ because the offset $\lambda = 1$ applied at the VNUs. For this reason DE thresholds obtained with the NOV and NOC models are different to the DE threshold of the OMS decoder (implemented with (3.1) and (3.2)). The largest difference of the models is obtained for the case of the regular $d_v = 3$ LDPC codes and precision $q = 3$. Fig. 3.6 and Fig. 3.7 show these results.

An important conclusion derived for $\boldsymbol{\varphi} = (\varphi_a, \varphi_a, \varphi_a)$ is that the best location for the noise injection is inside the VNU, *i.e.* the NIV model, and the worst location is outside the CNU, *i.e.* the NOC model. The NOV model exhibits noisy DE thresholds close to noisy DE thresholds of the NIV model, especially for $q = 4$ bits of precision.



(a) Precision $q = 3$ (b) Precision $q = 4$

Fig. 3.6 Noisy DE thresholds $\tilde{\delta}_{db}$ of NAN-MS decoders as a function of φ_a for regular $d_v = 3$ LDPC codes.



(a) Precision $q = 3$ (b) Precision $q = 4$

Fig. 3.7 Noisy DE thresholds $\tilde{\delta}_{db}$ of NAN-MS decoders as a function of φ_a for regular $d_v = 4$ LDPC codes.

3.5.3 Joint Optimization of the Noise Model Parameters and the Channel Gain Factor of NAN-MS Decoders

In Table 3.2, we indicate the noisy DE thresholds obtained with (3.21), and for the different noise injection locations, described in Section 3.3. Several conclusions can be derived from this analysis.

1. When comparing the noisy thresholds for the three methods to inject noise (NOV, NIV, and NOC), one can see that the best location for the noise injection is inside the variable node update, *i.e.* the NIV model of (3.5). When the noise is injected at the VNU, but after the quantization, *i.e.* the NOV model of (3.4), the noisy thresholds are also very good, and close to the NIV model. However, the injection of the noise after the CNU gives always the worse threshold. This gives us information about the best way to inject noise in NAN-MS decoders.

2. A second conclusion that can be drawn is that the DE thresholds of the NAN-MS decoders are always better than the DE thresholds for the noiseless decoders.

When comparing the best thresholds indicated in bold, the gains for NAN-MS decoders are quite important for the case of low precision $q = 3$, around 0.22 dB for $(d_v \in \{3,4\}, R = 1/2)$, 0.13 dB for $(d_v = 3, R = 3/4)$, and 0.25 dB for $(d_v = 4, R = 3/4)$, while the gains are smaller for the precision $q = 4$, 0.06 dB for $(d_v = 3, R = 1/2)$, 0.07 dB for $(d_v = 3, R = 3/4)$, 0.01 dB for $(d_v = 4, R = 1/2)$, and 0.02 dB for $(d_v = 4, R = 3/4)$. In the case of the largest precision $q = 5$ the gains are much smaller, 0.01 dB for $(d_v = 3, R \in \{1/2, 3/4\})$, 0.00006 dB for $(d_v = 4, R = 1/2)$, and 0.001 dB for $(d_v = 4, R = 3/4)$. From this analysis, we can conclude that the noise injection for NAN-MS decoders is more and more beneficial as the decoders are implemented in low precision.

3. A third interesting remark comes from the interpretation of the optimum noise parameters $\boldsymbol{\varphi}^*$ that we obtained through the DE analysis. For almost all cases, we have $\varphi_0^* = 0$, which means that the decoder should never apply an offset to the message when the amplitude of the message is equal to 1. This makes sense as transforming $m = \pm 1$ into $\tilde{m} = 0$ would erase the bit value and remove the sign information contained in the message. An exception appears for regular $d_v = 4$ LDPC codes and precision $q \in \{4,5\}$, as an example, we draw the evolution of the NAN-MS threshold versus the value of φ_0 for $(q = 4, d_v = 4, R = 1/2)$ on Fig. 3.8, while keeping the other parameters constant, one can see that for all models, there is an optimum value of the threshold around $\varphi_0^* = 0.5$, and each of them is beating the DE threshold of the OMS. The NIV and NOV models are very much better than the NOC model for all cases.

4. Finally, some of the obtained NAN-MS decoders are not really probabilistic since the values of the transition probabilities are very close to 0 or 1. This is the case for $(q = 5, d_v = 3, R = 1/2)$, $(q = 5, d_v = 3, R = 3/4)$, and $(q = 3, d_v = 4, R = 1/2)$.

All models are close to $(\varphi_s^*, \varphi_a^*, \varphi_0^*) = (1, 1, 0)$, which correspond to a deterministic OMS decoder for which the offset is not applied when $m = \pm 1$.

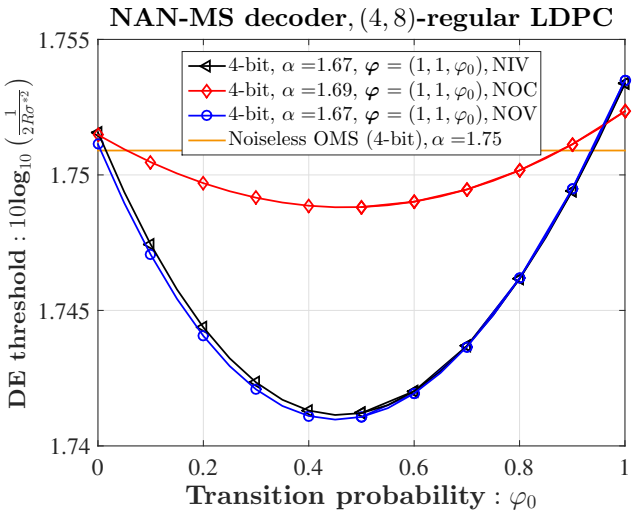


Fig. 3.8 Noisy DE thresholds $\tilde{\delta}_{db}$ of NAN-MS decoders as a function of φ_0 for regular $d_v = 4$ LDPC codes

3.6 Asymptotic Analysis of NAN-MS Decoders for Irregular LDPC codes

In the previous section we have seen that the optimum noise parameters $(\varphi_s^*, \varphi_a^*, \varphi_0^*)$ and the respective gains of NAN-MS decoders depend on the VN degree. For LDPC codes with irregular VN distribution, we propose therefore to extend our approach by considering a noise injection model Υ with different values of the transition probabilities for the different connection degrees.

We denote by $\Upsilon^{(2)} : \boldsymbol{\varphi}^{(2)} = \left(\varphi_s^{(2)}, \varphi_a^{(2)}, \varphi_0^{(2)}\right)$ the model which injects noise at VNs of degree $d_v = 2$. Both the NIV and NOV methods can be considered. Similarly, let $\Upsilon^{(3)} : \boldsymbol{\varphi}^{(3)} = \left(\varphi_s^{(3)}, \varphi_a^{(3)}, \varphi_0^{(3)}\right)$ denote the noise model for the VNs of degree $d_v = 3$.

Table 3.2 Noisy DE thresholds of NAN-MS decoders

NAN-MS decoders, (d_v, d_c)-regular LDPC code, BI-AWGN channel							
(d_v, d_c)	q	Method	α^*	φ_s^*	φ_a^*	φ_0^*	$\tilde{\delta}_{db}$
$(3,6)$	3 bits	Υ-NIV	0.82	0.475	0.475	0.000	**1.5711**
		Υ-NOV	0.77	0.338	0.313	0.000	1.5995
		Υ-NOC	0.77	0.850	0.363	0.000	1.6306
	4 bits	Υ-NIV	1.72	0.887	0.887	0.000	**1.2877**
		Υ-NOV	1.7	0.000	0.912	0.000	1.2917
		Υ-NOC	1.75	0.000	1.000	0.000	1.2931
	5 bits	Υ-NIV	2.49	1.000	1.000	0.000	**1.2045**
		Υ-NOV	2.49	1.000	1.000	0.000	**1.2045**
		Υ-NOC	2.49	1.000	1.000	0.000	**1.2045**
$(3,12)$	3 bits	Υ-NIV	0.60	0.413	0.413	0.000	**2.5989**
		Υ-NOV	0.60	0.900	0.338	0.000	**2.5989**
		Υ-NOC	0.61	1.000	0.425	0.000	2.6267
	4 bits	Υ-NIV	1.26	0.750	0.750	0.000	**2.3777**
		Υ-NOV	1.25	0.800	0.737	0.000	**2.3777**
		Υ-NOC	1.33	1.000	0.962	0.000	2.3873
	5 bits	Υ-NIV	2.15	1.000	1.000	0.000	**2.2938**
		Υ-NOV	2.15	0.800	1.000	0.000	**2.2938**
		Υ-NOC	2.15	1.000	1.000	0.000	**2.2938**
$(4,8)$	3 bits	Υ-NIV	0.93	0.950	0.950	0.000	**2.1056**
		Υ-NOV	0.92	1.000	0.912	0.000	2.1105
		Υ-NOC	0.92	1.000	1.000	0.000	2.1119
	4 bits	Υ-NIV	1.67	1.000	1.000	0.463	**1.7411**
		Υ-NOV	1.67	1.000	1.000	0.450	**1.7411**
		Υ-NOC	1.69	1.000	1.000	0.475	1.7488
	5 bits	Υ-NIV	1.95	1.000	1.000	0.825	**1.7055**
		Υ-NOV	1.95	0.800	1.000	0.825	**1.7055**
		Υ-NOC	1.98	1.000	1.000	1.000	1.7060
$(4,16)$	3 bits	Υ-NIV	0.72	0.900	0.900	0.000	**2.8121**
		Υ-NOV	0.72	0.700	0.887	0.000	**2.8121**
		Υ-NOC	0.71	1.000	1.000	0.000	2.8132
	4 bits	Υ-NIV	1.36	1.000	1.000	0.213	**2.5077**
		Υ-NOV	1.36	0.700	1.000	0.213	**2.5077**
		Υ-NOC	1.36	1.000	1.000	0.000	2.5101
	5 bits	Υ-NIV	1.81	1.000	1.000	0.662	**2.4594**
		Υ-NOV	1.81	0.500	1.000	0.662	**2.4594**
		Υ-NOC	1.86	1.000	1.000	1.000	2.4606

Finally, we decide to use the same model for all other VNs with degrees $d_v \geq 4$, denoted $\Upsilon^{(\geq 4)} : \boldsymbol{\varphi}^{(\geq 4)} = \left(\varphi_s^{(\geq 4)}, \varphi_a^{(\geq 4)}, \varphi_0^{(\geq 4)}\right)$.

The optimization of the transition probabilities for an irregular LDPC code with distribution $(\lambda(x), \rho(x))$ is still performed by the maximization of the noisy DE thresholds:

$$\left(\boldsymbol{\varphi}^{(2)*}, \boldsymbol{\varphi}^{(3)*}, \boldsymbol{\varphi}^{(\geq 4)*}, \alpha^*\right) = \arg \max_{(\boldsymbol{\varphi}^{(2)}, \boldsymbol{\varphi}^{(3)}, \boldsymbol{\varphi}^{(\geq 4)}, \alpha)} \left\{\tilde{\delta}\left(\lambda(x), \rho(x), q, \alpha, \boldsymbol{\varphi}^{(2)}, \boldsymbol{\varphi}^{(3)}, \boldsymbol{\varphi}^{(\geq 4)}\right)\right\} \tag{3.22}$$

For our analysis, we consider the ensemble of irregular LDPC codes which follow the distribution of the rate $R \in \{1/2, 3/4\}$, length $N = 2304$ code described in the WIMAX standard [7]. The degree distribution for the rate 1/2 code is $\lambda(x) = (22/76)x + (24/76)x^2 + (30/76)x^5$ and $\rho(x) = (48/76)x^5 + (28/76)x^6$, while for the rate 3/4 B code is $\lambda(x) = (10/88)x + (36/88)x^2 + (42/88)x^5$ and $\rho(x) = (28/88)x^{13} + (60/88)x^{14}$. For these distributions, we indicate in Table 3.3 the DE thresholds of the noiseless MS decoder and the noiseless OMS decoder.

Table 3.3 DE thresholds of noiseless MS decoders and noiseless OMS decoders with offset value $\lambda_v = 1$ for the WIMAX degree distribution

Irregular LDPC code, BI-AWGN channel					
		$R = 1/2$		$R = 3/4$	
q	λ_v	α^*	δ_{db}	α^*	δ_{db}
3 bits	0	0.44	**1.8310**	0.66	**2.8236**
	1	0.40	5.2283	0.50	3.4406
4 bits	0	1.07	**1.3941**	1.29	2.6150
	1	0.80	2.8140	1.42	**2.2416**
5 bits	0	2.30	1.3013	2.50	2.5654
	1	1.55	**1.1828**	1.97	**2.1637**

Noisy DE thresholds are summarized in Tables 3.4 and 3.5, where we indicate the optimum values of α and of the noise parameters for the different degrees. Those results confirm the conclusions of the regular LDPC codes analysis: *(i)* the NIV model is again

the best one, *(ii)* the optimum value for φ_0^* is 0 or it is close to 0 for $d_v = 3$ VNs and *(iii)* some of the optimized models are not probabilistic since the optimized values of the transition probabilities are very close to 0 or 1.

Table 3.4 DE thresholds of NAN-MS decoders and the rate 1/2 code.

NAN-MS decoders, Irregular LDPC code, BI-AWGN channel							
q	Method	α^*	d_v	φ_s^*	φ_a^*	φ_0^*	$\tilde{\delta}_{db}$
3 bits	Υ-NIV	0.42	2	0.000	0.000	0.000	**1.6236**
			3	0.350	0.350	0.000	
			≥ 4	1.000	1.000	0.225	
	Υ-NOV	0.42	2	0.000	0.013	0.000	1.6302
			3	0.000	0.063	0.000	
			≥ 4	0.000	1.000	0.300	
	Υ-NOC	0.44	-	0.000	0.000	0.000	1.8310
4 bits	Υ-NIV	1.0	2	0.000	0.000	0.000	**0.9995**
			3	1.000	1.000	0.000	
			≥ 4	1.000	1.000	0.000	
	Υ-NOV	0.98	2	0.000	0.000	0.000	1.0077
			3	0.000	0.988	0.000	
			≥ 4	0.000	1.000	0.000	
	Υ-NOC	0.83	-	0.000	0.375	0.000	1.2178
5 bits	Υ-NIV	1.84	2	0.600	0.600	0.000	**0.8233**
			3	1.000	1.000	0.000	
			≥ 4	1.000	1.000	0.000	
	Υ-NOV	1.85	2	0.000	0.613	0.000	**0.8233**
			3	0.000	1.000	0.000	
			≥ 4	0.000	1.000	0.000	
	Υ-NOC	1.66	-	0.000	0.800	0.000	0.8679

Another conclusion can be driven from these tables. From the DE analysis we can conclude that the noise should not be injected on degree $d_v = 2$ VNs for the case of low precision $q = 3$, since we obtain always $(\varphi_s^{(2)}, \varphi_a^{(2)}, \varphi_0^{(2)}) \simeq (0, 0, 0)$. While in the case of the two considered precision $q = 4$ and $q = 5$, the noise should be injected on

Table 3.5 DE thresholds of NAN-MS decoders and the rate 3/4 B code.

NAN-MS decoders, Irregular LDPC code, BI-AWGN channel							
q	Method	α^*	d_v	φ_s^*	φ_a^*	φ_0^*	$\tilde{\delta}_{db}$
3 bits	Υ-NIV	0.67	2	0.038	0.038	0.000	**2.5323**
			3	0.713	0.713	0.000	
			≥ 4	0.925	0.925	0.025	
	Υ-NOV	0.63	2	0.025	0.000	0.000	2.5930
			3	0.000	0.025	0.000	
			≥ 4	0.013	1.000	0.238	
	Υ-NOC	0.60	-	0.025	0.788	0.000	2.6172
4 bits	Υ-NIV	1.39	2	1.000	1.000	0.838	**2.2138**
			3	1.000	1.000	0.000	
			≥ 4	1.000	1.000	0.000	
	Υ-NOV	1.23	2	0.000	0.513	0.000	2.2284
			3	0.000	1.000	0.138	
			≥ 4	0.838	1.000	0.000	
	Υ-NOC	1.16	-	1.000	1.000	0.000	2.2341
5 bits	Υ-NIV	1.89	2	1.000	1.000	0.000	**2.1611**
			3	1.000	1.000	0.288	
			≥ 4	1.000	1.000	1.000	
	Υ-NOV	1.89	2	0.400	1.000	0.000	**2.1611**
			3	0.500	1.000	0.288	
			≥ 4	0.500	1.000	1.000	
	Υ-NOC	1.97	-	1.000	1.000	1.000	2.1637

degree $d_v = 2$ VNs for almost all cases. These observations, combined with the fact that the optimum values for $\boldsymbol{\varphi}^{(\geq 4)*}$ are almost always 0 or 1, lead to the conclusion that injecting noise in NAN-MS decoders for irregular LDPC codes is especially important for the degree $d_v = 3$ VNs (inject the noise on degree $d_v = 2$ VNs will be depend on the degree distribution and the precision used)

Finally, the NAN-MS decoders obtain higher gains using irregular codes. The gain of the rate 1/2 code is 0.2074 dB for the lower precision $q = 3$, 0.3946 dB for the precision $q = 4$, and 0.3595 dB for the largest precision $q = 5$. In the case of the rate 3/4 B code, the gains for the three considered precision $q = 3$, $q = 4$ and $q = 5$ are smaller than the rate 1/2 code, a gain of 0.2913 dB for $q = 3$, a gain of 0.0278 dB for $q = 4$, and a gain of 0.0026 dB for $q = 5$.

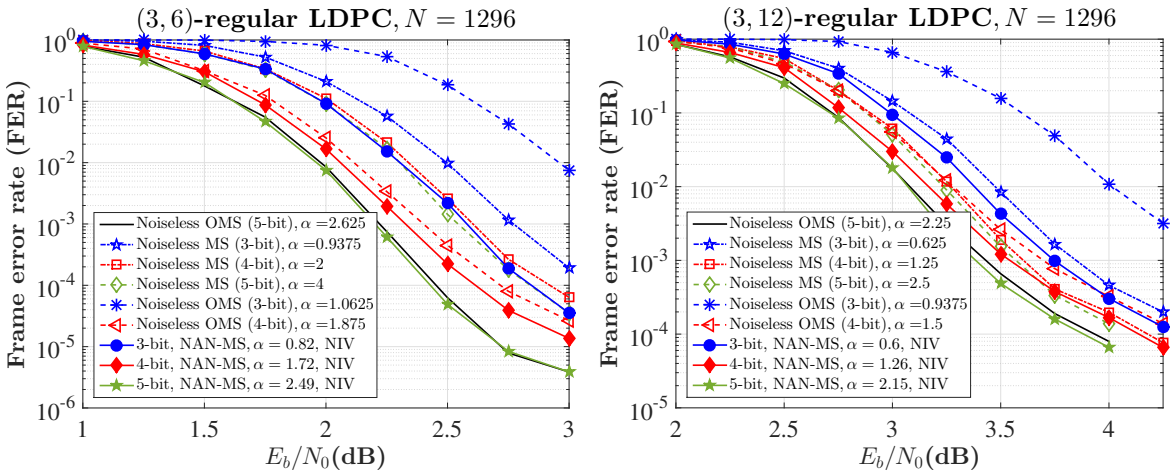
3.7 Finite Length Performance of NAN-MS Decoders

In this chapter we present the frame error rate (FER) performance for noiseless and NAN-MS decoders. In order to corroborate the asymptotic results obtained in the previous chapter, we analyze the quantized decoder performance over the BI-AWGN channel. For this purpose, we designed a $(3, 6)$-regular QC-LDPC code and a $(4, 8)$-regular QC-LDPC code (respectively a $(3, 12)$-regular QC-LDPC code and a $(4, 16)$-regular QC-LDPC code) with length $N = 1296$ and circulant size $L = 54$ (respectively $L = 27$), using the PEG algorithm from [52]. The considered decoders are the ones with the best DE thresholds, indicated in bold in Tables 3.1 and 3.2. The OMS decoder performance for $q = 5$ and $\lambda_v = 1$ is considered as benchmark.

Fig. 3.9 shows the FER performance comparisons between the noiseless MS, noiseless OMS, and best NAN-MS decoders, for the three considered precisions $q = 3$, $q = 4$,

and $q = 5$, and for the regular $d_v = 3$ QC-LDPC code. A maximum of 100 iterations has been set for all decoders. Fig. 3.10 draws the same curves for the regular $d_v = 4$ QC-LDPC code. A first conclusion is that the finite length FER performances are in accordance with the gains predicted by the DE analysis.

For low precision messages $q = 3$ and for regular $d_v = 3$ LDPC codes, the MS decoder is better than the OMS decoder. This result is surprising, but it can be justified by the fact that the variable degree is small ($d_v = 3$), while an offset of $\lambda_v = 1$ represents a significant amount of the total available dynamic when 3 bits is used ($N_q = 3$). In those conditions, the OMS does not help extrinsic messages to be more reliable at each new decoding iteration.



(a) $(d_v = 3, d_c = 6)$ (b) $(d_v = 3, d_c = 12)$

Fig. 3.9 FER performance of NAN-MS decoders for the $d_v = 3$ regular LDPC codes.

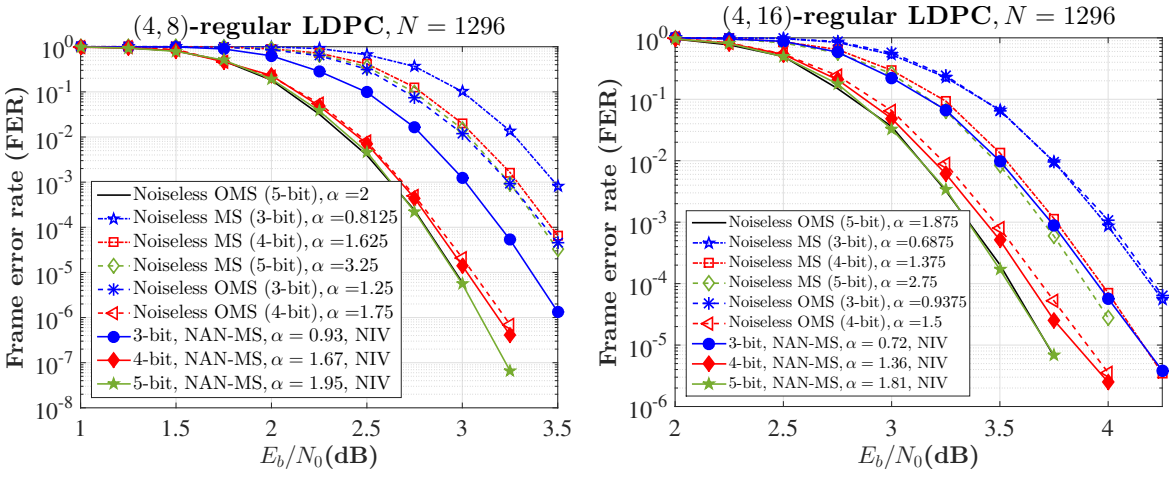
We observe in the waterfall (*i.e.* at FER $= 10^{-2}$) an SNR gain for the NAN-MS decoders which corresponds to the threshold differences: around 0.2 dB for ($q = 3$, $d_v = 3$, $R = 1/2$), and 0.05 dB for ($q = 4$, $d_v = 3$, $R = 1/2$). The others SNR gains are provided in Table 3.6.

Additional simulation results are provided in Fig. 3.11 which shows the FER performance comparisons between NAN-MS decoders implemented with the NIV,

Table 3.6 DE gains and SNR gains of NAN-MS decoders.

(d_v, d_c)-regular LDPC code, BI-AWGN channel								
	$(d_v=3, d_c=6)$		$(d_v=3, d_c=12)$		$(d_v=4, d_c=8)$		$(d_v=4, d_c=16)$	
q	DE gain	SNR gain	DE gain	SNR gain	DE gain	SNR gain	DE gain	SNR gain
3 bits	**0.2177**	**0.2**	**0.1327**	**0.1**	**0.2163**	**0.2**	**0.2511**	**0.25**
4 bits	**0.0604**	**0.05**	**0.0707**	**0.08**	**0.0098**	**0.000**	**0.0215**	**0.02**
5 bits	**0.0109**	**0.000**	**0.0102**	**0.000**	**0.0006**	**0.000**	**0.0012**	**0.000**

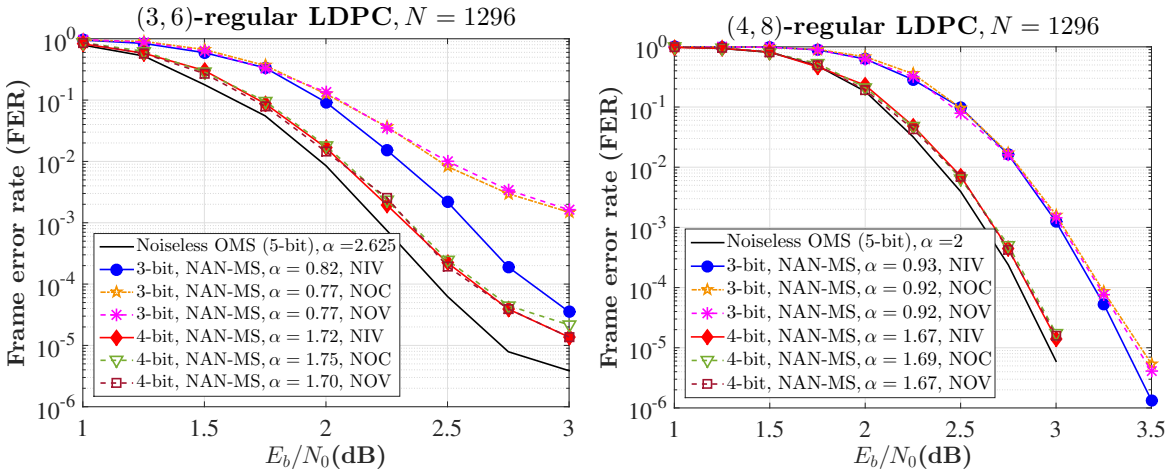
NOV, and NOC models. For the regular $d_v = 3$ QC-LDPC code and precision $q = 3$, it is evident that the NIV model clearly outperforms the NOV and NOC models (which exhibit poor performance) in the error-floor region. In the case of the regular $d_v = 4$ QC-LDPC code and $q = 3$, the NIV model achieves slightly better FER performance in the error-floor region compared to the NOV and NOC models. For the two regular QC-LDPC codes and $q = 4$ bits of precision, the three models exhibit almost the same FER performance.



(a) $(d_v = 4, d_c = 8)$ (b) $(d_v = 4, d_c = 16)$

Fig. 3.10 FER performance of NAN-MS decoders for the $d_v = 4$ regular LDPC codes.

Simulation results for the WIMAX rate 1/2 LDPC code and the WIMAX rate 3/4 B LDPC code are provided on Fig. 3.12. Again, the SNR gains in the waterfall correspond to what was predicted with the DE analysis, with a 0.4 dB gain for $(q = 4, R = 1/2)$, a 0.2 dB gain for $(q = 3, R = 1/2)$, a 0.32 dB gain for $(q = 4, R = 3/4)$, and a 0.24 dB



(a) $(d_v = 3, d_c = 6)$ (b) $(d_v = 4, d_c = 8)$

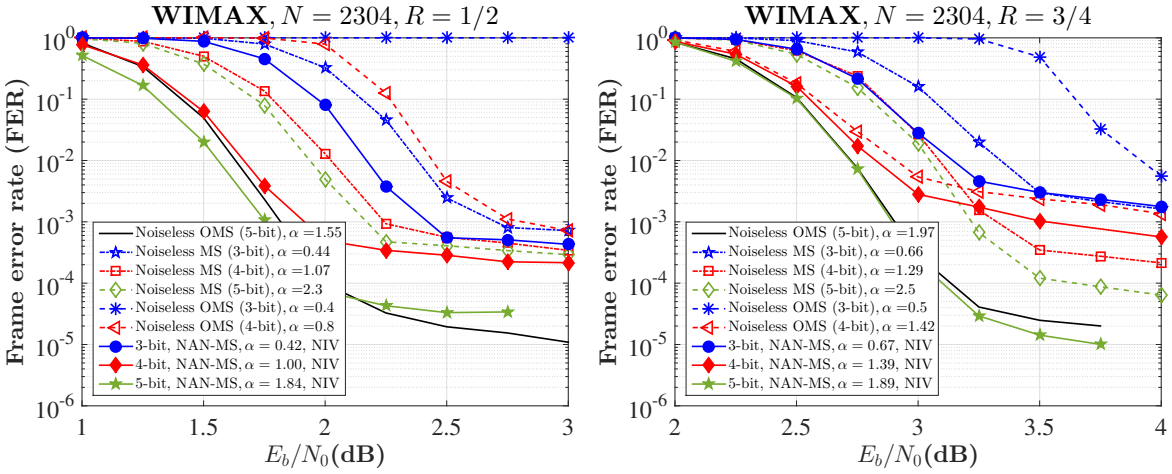
Fig. 3.11 FER performance of NAN-MS decoders implemented with the NIV, NOV, and NOC model.

gain for $(q = 3, R = 3/4)$. For the largest precision $q = 5$, the gain for $R = 1/2$, which is 0.08 dB, is much smaller than was predicted with the DE analysis, while for $R = 3/4$ the same performance is achieved.

The NIV model is the best model because after the noise injection, the noisy v-to-c messages and c-to-v messages always belong to the alphabet $\mathcal{A}_C$. In the case of the NOC model, the noise perturbs the saturated c-to-v messages obtaining in some iterations the noisy c-to-v messages belonging to the alphabet $\mathcal{A}_C \backslash \{-N_q, +N_q\}$, and in other iterations the noisy messages belonging to the alphabet $\mathcal{A}_C$. Similarly, in the NOV model, the noise perturbs the saturated v-to-c messages obtaining in some iterations the noisy v-to-c messages belonging to $\mathcal{A}_C \backslash \{-N_q, +N_q\}$ and in other iterations the noisy messages belonging to $\mathcal{A}_C$. For the NOC and NOV models, not using the saturation values in some decoding iterations causes the degradation of the decoder performance, especially for low precision messages.

As a last remark, we can also see that the injection of errors with the proposed models Υ do not seem to have an influence in the error floor of the decoders, since all

the curves have similar slopes in the low FER region. This means that the proposed models do not correct the dominant error events due to trapping sets.



(a) $R = 1/2$ (b) $R = 3/4$

Fig. 3.12 FER performance of NAN-MS decoders for the WIMAX LDPC code.

3.8 Implementation of NAN-MS Decoders as Deterministic Decoders

This section is dedicated to a simple implementation of NAN-MS decoders as deterministic decoders. From the analysis made in Section 3.5 and 3.6, an important conclusion obtained was that the offset should not be applied to messages whose magnitude is 1 ($m = \pm 1$), *i.e.* $(\varphi_s^*, \varphi_a^*, \varphi_0^*)$ should be equal to $(1,1,0)$, but for some cases analyzed φ_0^* is not close or equal to 0. In order to make a trade-off between FER performance and cost of implementation, we set $(\varphi_s^*, \varphi_a^*, \varphi_0^*) = (1,1,0)$ for all cases. Considering the NIV model with $(\varphi_s^*, \varphi_a^*, \varphi_0^*) = (1,1,0)$, we define the Modified Offset Min-Sum (M-OMS) decoder. For this decoder, the update rule at a CNU is given by equation (3.1), and

the update rule at a VNU is written as

$$m_{v_n \to c_m}^{(\ell+1)} = \Psi_v \left(I_n, \left\{ m_{c \to v_n}^{(\ell)} \right\}_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \right) = \Lambda \left(m_{v_n \to c_m}^{(\ell+1),U} \right), \tag{3.23}$$

where the function $\Lambda(.)$ is redefined by

$$\Lambda(a) = \begin{cases} \mathcal{S}(a+1, N_q), & \text{if } a < -1 \\ a, & \text{if } a \in \{-1, 0, +1\} \\ \mathcal{S}(a-1, N_q), & \text{if } a > +1 \end{cases}$$

We can note that the M-OMS decoders have similar cost of implementation than the OMS decoders because both use the same function Ψ_c to perform the update rule at CNUs, and also because the update rule at VNUs is almost the same except for the case where the unsaturated v-to-c message is -1, 0, or $+1$, *i.e.* $m_{v_n \to c_m}^{(\ell+1),U} \in \{-1, 0, +1\}$.

3.8.1 Density Evolution thresholds

In this section, we present DE thresholds for the M-OMS decoders considering the four (d_v, d_c)-regular QC-LDPC codes presented in Section 3.5, and the two WIMAX LDPC codes studied in Section 3.6.

Density Evolution Thresholds for Regular LDPC Codes

The optimization (3.21) is used to find the optimum channel gain factor α^* for regular LDPC codes. In Table 3.7, we show the DE thresholds of M-OMS decoders. When comparing the best thresholds indicated in bold in Table 3.1 and Table 3.7, we can observe: *(i)* the DE thresholds of the M-OMS decoders are always better than the DE thresholds of the MS and OMS decoders for $d_v = 3$, *(ii)* for almost all cases when $q \in \{3, 4\}$ and $d_v = 4$, the M-OMS decoders give us always better DE thresholds

than OMS decoders, an exception appears for $(q=4, d_v=4, R=1/2)$, and *(iii)* for $(q=5, d_v=4)$ the DE thresholds of the OMS decoders are always better than the DE thresholds of the M-OMS decoders. From this analysis, we can conclude that the M-OMS decoders are better decoders than the MS and OMS in low precision $q \in \{3,4\}$.

On the other hand, when comparing the best thresholds indicated in bold in Table 3.2 and Table 3.7, we observe that the NAN-MS decoders (using the NIV model) give us always better DE thresholds than M-OMS decoders, hence, one can conclude that NAN-MS decoders are better decoders than the M-OMS decoders especially for $d_v = 3$ and precision $q = 3$.

Table 3.7 DE thresholds of M-OMS decoders.

(d_v, d_c)-regular LDPC code, BI-AWGN channel								
	$(d_v=3, d_c=6)$		$(d_v=3, d_c=12)$		$(d_v=4, d_c=8)$		$(d_v=4, d_c=16)$	
q	α^*	δ_{db}	α^*	δ_{db}	α^*	δ_{db}	α^*	δ_{db}
3 bits	0.78	**1.7012**	0.53	**2.6715**	0.93	**2.1061**	0.71	**2.8131**
4 bits	1.78	**1.2906**	1.35	**2.3869**	1.64	**1.7514**	1.35	**2.5101**
5 bits	2.49	**1.2045**	2.15	**2.2938**	1.88	**1.7213**	1.77	**2.4657**

Density Evolution Thresholds for Irregular LDPC Codes

In the case of WIMAX LDPC codes and considering the NIV model, we set $\boldsymbol{\varphi}^{(\geq 4)} = \boldsymbol{\varphi}^{(3)} = (1,1,0)$, and we decide to analyze two cases for $\boldsymbol{\varphi}^{(2)}$: *(i)* the first one setting $\boldsymbol{\varphi}^{(2)} = (0,0,0)$ and *(ii)* the second one setting $\boldsymbol{\varphi}^{(2)} = (1,1,0)$.

The optimum channel gain factor α^* is computed using the optimization (3.22). Table 3.8 and Table 3.9 summarize the DE thresholds of the M-OMS decoders. From the results obtained, we can note: *(i)* $\boldsymbol{\varphi}^{(2)}$ has to be equal to $(0,0,0)$ for $R = 1/2$, *(ii)* for $R = 3/4$, $\boldsymbol{\varphi}^{(2)}$ has to be equal to $(0,0,0)$ for low precision $q = 3$, while for $R = 3/4$ and precision $q \in \{4,5\}$, $\boldsymbol{\varphi}^{(2)}$ has to be equal to $(1,1,0)$, and *(iii)* the DE thresholds of the M-OMS decoders are always better than the DE thresholds of the MS and OMS

decoders. Comparing the best thresholds indicated in bold showed in Tables 3.4, 3.5, 3.8, and 3.9, one can note that NAN-MS decoders achieve slightly better DE thresholds than M-OMS decoders.

We can conclude that in low precision $q = 3$, $\boldsymbol{\varphi}^{(2)}$ has to be equal to $(0, 0, 0)$ in order to implement better decoders than MS and OMS decoders. Hence, the function $\Lambda(.)$ of (3.23) can be redefined by

$$\Lambda(a) = \begin{cases} \mathcal{S}(a, N_q), & \text{if } d_v = 2 \\ \mathcal{S}(a+1, N_q), & \text{if } d_v \geq 3, \text{and if } a < -1 \\ a, & \text{if } d_v \geq 3, \text{and if } a \in \{-1, 0, +1\} \\ \mathcal{S}(a-1, N_q), & \text{if } d_v \geq 3, \text{and if } a > +1 \end{cases}$$

Table 3.8 DE thresholds of M-OMS decoders and the rate 1/2 code.

Irregular LDPC code, BI-AWGN channel					
q	α^*	$\varphi_s^{(2)}$	$\varphi_a^{(2)}$	$\varphi_0^{(2)}$	$\tilde{\delta}_{db}$
3 bits	0.4100	0.0	0.0	0.0	**1.6688**
	0.3600	1.0	1.0	0.0	4.4922
4 bits	1.0000	0.0	0.0	0.0	**0.9995**
	0.7800	1.0	1.0	0.0	2.5934
5 bits	1.8400	0.0	0.0	0.0	**0.8767**
	1.5300	1.0	1.0	0.0	1.1266

3.8.2 Finite Length Performance

In this section, we present the FER performance of noiseless MS and OMS, NAN-MS, and M-OMS decoders. In order to corroborate the asymptotic results obtained in the previous section, we analyze the quantized decoder performance over the BI-AWGN channel. For this purpose, we use the four (d_v, d_c)-regular QC-LDPC codes presented

Table 3.9 DE thresholds of M-OMS decoders and the rate 3/4 B code.

Irregular LDPC code, BI-AWGN channel					
q	α^*	$\varphi_s^{(2)}$	$\varphi_a^{(2)}$	$\varphi_0^{(2)}$	$\tilde{\delta}_{db}$
3 bits	0.6600	0.0	0.0	0.0	**2.5324**
	0.4500	1.0	1.0	0.0	2.7331
4 bits	1.3100	0.0	0.0	0.0	2.2338
	1.3800	1.0	1.0	0.0	**2.2148**
5 bits	1.7200	0.0	0.0	0.0	2.1937
	1.8600	1.0	1.0	0.0	**2.1647**

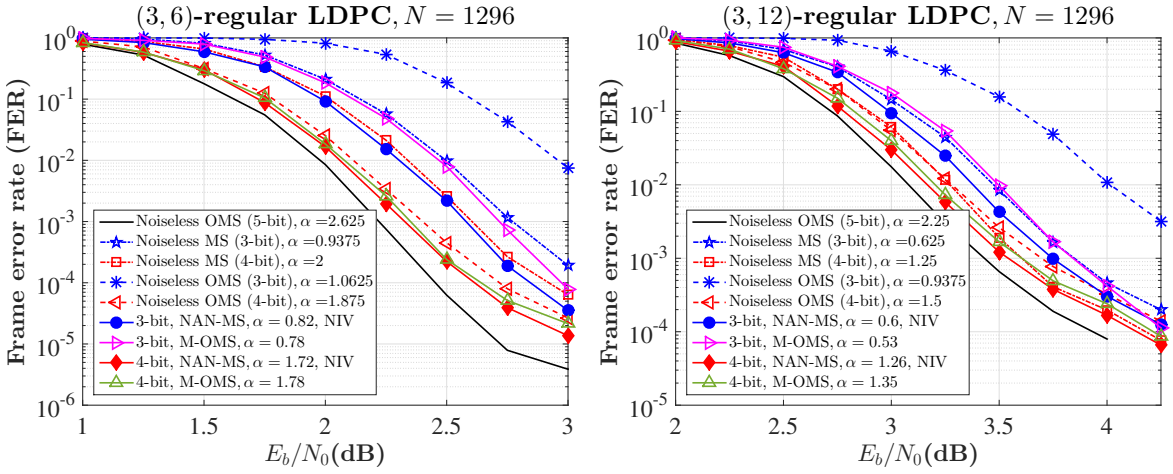
in Chapter 3.5. The considered decoders are the ones with the best DE thresholds, indicated in bold in Tables 3.1, 3.2, and 3.7. The noiseless 5-bit OMS decoder with offset $\lambda_v = 1$ is also shown as benchmark.

Fig. 3.13 and Fig. 3.14 show the FER performance comparisons between the 5-bit OMS decoder, the noiseless quantized decoders, the NAN-MS decoders which use the NIV model, and the M-OMS decoders, as a function of E_b/N_0 over the BI-AWGN channel.

FER performance results for the regular $d_v = 3$ QC-LPDC codes are drawn in Fig. 3.13. We observe that the 3-bit M-OMS decoders have better FER performance than 3-bit OMS decoders, but 3-bit NAN-MS decoders achieve better FER performance than 3-bit M-OMS decoders, as it was predicted by the DE analysis. In the case of $q = 4$, the FER performance of NAN-MS decoders achieve slightly better FER performance results than the M-OMS decoders, as predicted by the DE threshold analysis.

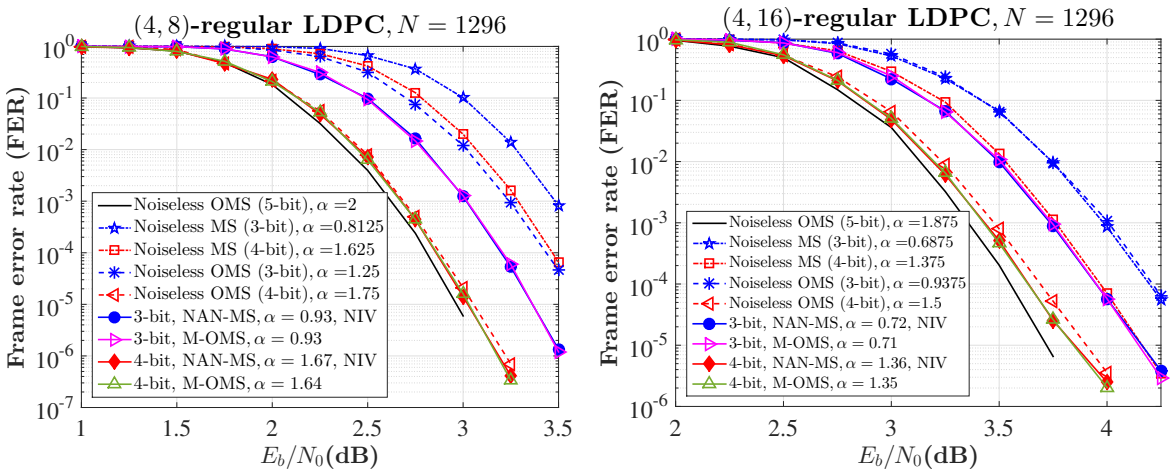
Simulation results for the regular $d_v = 4$ QC-LPDC codes are provided in Fig. 3.14a. For the two considered precisions $q = 3$ and $q = 4$, we can see that both the M-OMS decoders and NAN-MS decoders have almost the same performance in the waterfall region and in the error-floor region. We can also note that the 4-bit NAN-MS

decoders and the 4-bit M-OMS decoders achieve slightly better FER performance results than the 4-bit noiseless OMS decoders. It is also evident that both the 3-bit NAN-MS decoders and the 3-bit M-OMS decoders are capable of outperforming the 3-bit noiseless OMS decoders in both the waterfall region (as was predicted by the DE analysis) and the error-floor region.



(a) $R = 1/2$ (b) $R = 3/4$

Fig. 3.13 FER performance of M-OMS and NAN-MS decoders for the regular $d_v = 3$ regular LDPC codes.



(a) $R = 1/2$ (b) $R = 3/4$

Fig. 3.14 FER performance of M-OMS and NAN-MS decoders for the regular $d_v = 3$ regular LDPC codes.

3.9 Implementation of NAN-MS Decoders Using an Offset Vector

This section presents an alternative way to implement NAN-MS decoders using an *offset vector*. From the analysis made in Section 3.5 and 3.6, the noise is injected into the NAN-MS decoders at each decoding iteration. We can relax the way of injecting noise to the decoder considering the transition probabilities equal to 0 or 1 at each decoding iteration, in other words, an offset equal to 0 or 1 can be applied to the decoder at each decoding iteration.

Let $\boldsymbol{\lambda_v}$ denote the *offset vector* defined as $\boldsymbol{\lambda_v} = \left(\lambda_v^{(1)}, \lambda_v^{(2)}, \ldots, \lambda_v^{(L_{max})}\right)$, where $\lambda_v^{(\ell)} \in \{0,1\}$, and L_{max} is the maximum number of iterations. Let also denote OMS-$\boldsymbol{\lambda_v}$ the Offset Min-Sum decoder applying the offset value $\lambda_v^{(\ell)}$ at iteration ℓ, the update rule at a CNU is given by equation (3.1), and the update rule at a VNU is written as

$$m_{v_n \to c_m}^{(\ell+1)} = \Psi_v \left(I_n, \left\{ m_{c \to v_n}^{(\ell)} \right\}_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \right) = \Lambda \left(m_{v_n \to c_m}^{(\ell+1),U} \right), \tag{3.24}$$

where the function $\Lambda(.)$ is redefined by

$$\Lambda(a) = \begin{cases} \mathcal{S}(a + \lambda_v^{(\ell)}, N_q), & \text{if } a < -1 \\ a, & \text{if } a \in \{-1, 0, +1\} \\ \mathcal{S}(a - \lambda_v^{(\ell)}, N_q), & \text{if } a > +1 \end{cases}$$

We can implement $2^{L_{max}}$ different decoders because $\lambda_v^{(\ell)}$ can only be 0 or 1. We obtain the M-OMS decoder of Section 3.8 setting $\boldsymbol{\lambda_v} = (1, 1, \ldots, 1)$. Also, the special case of $\boldsymbol{\lambda_v} = (0, 0, \ldots, 0)$ corresponds to the classical MS decoder.

The OMS-$\boldsymbol{\lambda_v}$ decoders have similar cost of implementation than the M-OMS decoders because both use the same function Ψ_c to perform the update rule at CNUs,

and also because the update rule at VNUs is almost the same, except for the offset vector. In order to save the offset vector $\boldsymbol{\lambda_v}$, we need a very small memory, specifically a memory that stores L_{max} bits of information.

3.9.1 Density Evolution thresholds

The OMS-$\boldsymbol{\lambda_v}$ decoder is analyzed with DE relaxing one of the DE constraints, specifically, the number of iterations used to compute the DE threshold will change from a very large value ($L_{max} = 1000$) to a small value ($L_{max} = 20$). In other words, the behavior of the OMS-$\boldsymbol{\lambda_v}$ decoder is analyzed using only 20 iterations.

We consider the ensemble of (d_v, d_c)-regular LDPC codes with $d_v \in \{3, 4\}$ and $R = 1/2$ presented in Section 3.5, and the two WIMAX LDPC codes studied in Section 3.6.

Density Evolution Thresholds for Regular LDPC Codes

The DE equations of the NIV model are used to compute the DE thresholds considering $\lambda_v^{(\ell)} = \varphi_s = \varphi_a$ and $\varphi_0 = 0$. We use the DE threshold δ to jointly optimize the offset vector for a fixed number of iterations, and a fixed precision:

$$\left(\alpha^*, \lambda_v^{(1)*}, \lambda_v^{(2)*}, ..., \lambda_v^{(L_{max})*}\right) = \arg \max_{\left(\alpha, \lambda_v^{(1)}, \lambda_v^{(2)}, ..., \lambda_v^{(L_{max})}\right)} \left\{ \tilde{\delta}\left(d_v, d_c, q, \alpha, \left(\lambda_v^{(1)}, \lambda_v^{(2)}, ..., \lambda_v^{(L_{max})}\right)\right)\right\}. \tag{3.25}$$

Table 3.10 shows the DE thresholds and the optimum channel gain factor α^* for the MS and the OMS using the optimization (3.21) with $L_{max} = 20$. The DE thresholds of the M-OMS decoders are shown in Table 3.11 using the optimization (3.25).

In Fig. 3.15 and Fig. 3.16, we show the DE thresholds of 2^{20} OMS-$\boldsymbol{\lambda_v}$ decoders for the regular LDPC codes with $d_v = 3$ and $d_v = 4$. The results are obtained considering a fixed channel gain factor α, and varying the offset vector from $(\lambda_v^{(1)} = 0, ..., \lambda_v^{(20)} = 0)$

Table 3.10 DE thresholds of noiseless MS decoders and noiseless OMS decoders with offset value $\lambda_v = 1$ using $L_{max} = 20$.

(d_v, d_c)-regular LDPC code, BI-AWGN channel					
(d_v, d_c)	q	λ_v	α^*	δ	δ_{db}
$(3,6)$	3 bits	0	0.75	0.7870784477	**2.0796**
		1	1.00	0.7314404398	2.7164
	4 bits	0	1.51	0.8028499898	1.9073
		1	1.76	0.8174758297	**1.7505**
$(4,8)$	3 bits	0	0.75	0.7277239691	2.7607
		1	1.15	0.7433918740	**2.5756**
	4 bits	0	1.49	0.7449151475	2.5579
		1	1.70	0.7969916843	**1.9709**

to $(\lambda_v^{(1)} = 1, ..., \lambda_v^{(20)} = 1)$. We denote the first decoder if $(\lambda_v^{(20)} = 0, ..., \lambda_v^{(2)} = 0, \lambda_v^{(1)} = 1)$, the second decoder if $(\lambda_v^{(20)} = 0, ..., \lambda_v^{(3)} = 0, \lambda_v^{(2)} = 1, \lambda_v^{(1)} = 0)$, and so on. From the results, we can see that there are many offset vectors that help the OMS-$\boldsymbol{\lambda_v}$ decoder beat the MS and OMS. In Table 3.12, we show the best DE thresholds of OMS-$\boldsymbol{\lambda_v}$ decoders, we also show the DE gains obtained comparing the best thresholds indicated in bold in Table 3.10 and the thresholds of OMS-$\boldsymbol{\lambda_v}$ decoders.

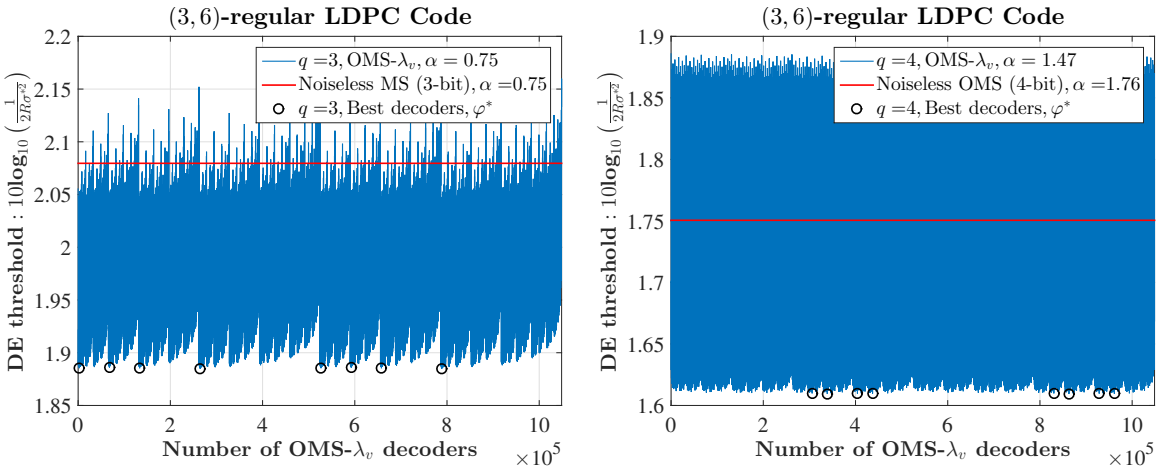
When comparing the best thresholds indicated in bold in Table 3.10 and Table 3.12, we can observe: *(i)* the DE thresholds of the OMS-$\boldsymbol{\lambda_v}$ decoders are always better than the DE thresholds of the MS and OMS decoders, *(ii)* the offset equal to 1 is applied mainly during the first iterations, while in the rest of iterations the offset equal to 1 is applied from time to time, *(iii)* for the regular $d_v = 3$ LDPC code, about 50% of the applied offset is equal to 1 for both precisions $q = 3$ and $q = 4$, *(iv)* for the regular $d_v = 4$ LDPC code, about 25% of the applied offset is equal to 1 precisions $q = 3$, while for the largest precision $q = 4$, almost always the offset equal to 1 is applied, and *(v)* the DE gains for low precision is quite important, the largest gain obtained is around 0.2773 dB. While the DE gains are smaller for the largest precision $q = 4$.

From this analysis, we can conclude that the OMS-$\boldsymbol{\lambda_v}$ decoders are better decoders than the MS and OMS when the same precision is used.

On the other hand, when comparing the best thresholds indicated in bold in Table 3.11 and Table 3.12, we observe that the OMS-$\boldsymbol{\lambda_v}$ decoders give us always better DE thresholds than the M-OMS decoders, hence, one can conclude that the OMS-$\boldsymbol{\lambda_v}$ decoders are better decoders to implement the NAN-MS decoders.

Table 3.11 DE thresholds of the M-OMS decoders using $L_{max} = 20$.

(d_v, d_c)-regular LDPC code, BI-AWGN channel				
(d_v, d_c)	q	α	δ_{db}	$\lambda_v^{(20)}\lambda_v^{(19)}...\lambda_v^{(2)}\lambda_v^{(1)}$
$(3,6)$	3 bits	0.78	**2.1352**	11111111111111111111
	4 bits	1.78	**1.6612**	11111111111111111111
$(4,8)$	3 bits	0.93	**2.3196**	11111111111111111111
	4 bits	1.64	**1.9483**	11111111111111111111



(a) Precision $q = 3$ (b) Precision $q = 4$

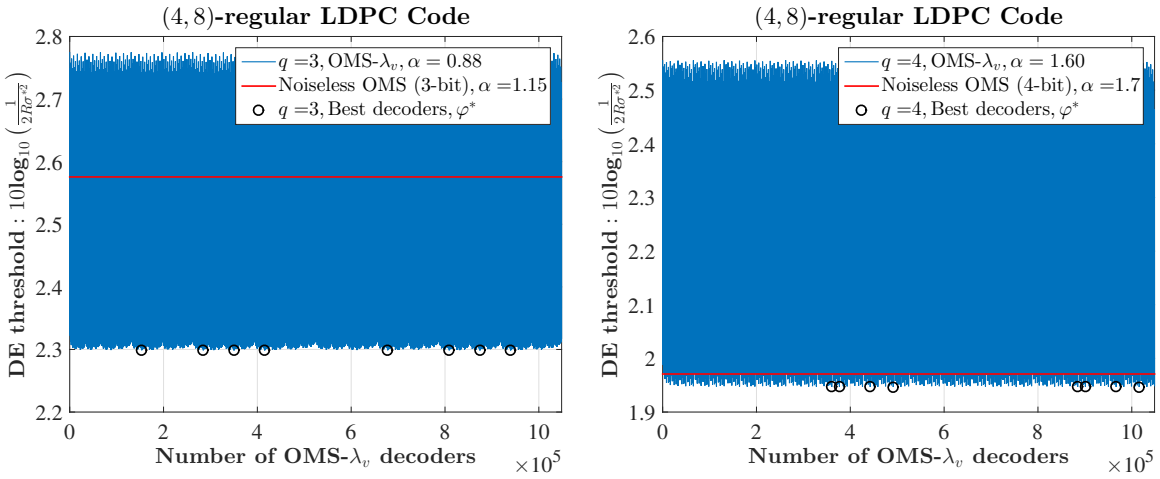
Fig. 3.15 DE thresholds δ_{db} of 2^{20} OMS-$\boldsymbol{\lambda_v}$ decoders for the regular $d_v = 3$ LDPC code.

Density Evolution Thresholds for Irregular LDPC Codes

Similar to Section 3.6, for LDPC codes with irregular VN distribution, we consider an offset vector for the different connection degree. Hence, $\Upsilon^{(2)} : \boldsymbol{\lambda_v}^{(2)} =$

Table 3.12 DE thresholds of the OMS-$\boldsymbol{\lambda_v}$ decoders using $L_{max} = 20$.

(d_v, d_c)-regular LDPC code, BI-AWGN channel					
(d_v, d_c)	q	α^*	δ_{db}	$\lambda_v^{(20)*}\lambda_v^{(19)*}...\lambda_v^{(2)*}\lambda_v^{(1)*}$	DE gain
$(3,6)$	3 bits	0.75	1.8853	00000000100101010111	0.1943
			1.8858	00010000100101010111	0.1938
			1.8853	00100000100101010111	0.1943
			1.8848	**01000000100101010111**	**0.1948**
			1.8853	10000000100101010111	0.1943
			1.8858	10010000100101010111	0.1938
			1.8853	10100000100101010111	0.1943
			1.8848	**11000000100101010111**	**0.1948**
	4 bits	1.47	1.6098	01001010101011011111	0.1407
			1.6094	**01010010101011011111**	**0.1411**
			1.6097	01100010101011011111	0.1408
			1.6096	01101010101011011111	0.1409
			1.6098	11001010101011011111	0.1407
			1.6094	**11010010101011011111**	**0.1411**
			1.6097	11100010101011011111	0.1408
			1.6096	11101010101011011111	0.1409
$(4,8)$	3 bits	0.88	2.2986	00100101011111111111	0.2770
			2.2985	01000101011111111111	0.2771
			2.2983	**01010101011111111111**	**0.2773**
			2.2984	01100101011111111111	0.2772
			2.2986	10100101011111111111	0.2770
			2.2985	11000101011111111111	0.2771
			2.2983	**11010101011111111111**	**0.2773**
			2.2984	11100101011111111111	0.2772
	4 bits	1.60	1.9468	01010111111111111111	0.0241
			1.9470	01011011111111111111	0.0239
			1.9469	01101011111111111111	0.0240
			1.9467	**01110111111111111111**	**0.0242**
			1.9468	11010111111111111111	0.0241
			1.9470	11011011111111111111	0.0239
			1.9469	11101011111111111111	0.0240
			1.9467	**11110111111111111111**	**0.0242**



(a) Precision $q = 3$ (b) Precision $q = 4$

Fig. 3.16 DE thresholds δ_{db} of 2^{20} OMS-$\boldsymbol{\lambda_v}$ decoders for the regular $d_v = 4$ LDPC code.

$\left(\lambda_v^{(2,1)}, \lambda_v^{(2,2)}, ..., \lambda_v^{(2,L_{max})}\right)$ is the offset vector at VNs of degree $d_v = 2$, $\Upsilon^{(3)} : \boldsymbol{\lambda}_v{}^{(3)} = \left(\lambda_v^{(3,1)}, \lambda_v^{(3,2)}, ..., \lambda_v^{(3,L_{max})}\right)$ is the offset vector for the VNs of degree $d_v = 3$, and finally, for all other VNs with degrees $d_v \geq 4$, $\Upsilon^{(\geq 4)} : \boldsymbol{\lambda}_v{}^{(\geq 4)} = \left(\lambda_v^{(\geq 4,1)}, \lambda_v^{(\geq 4,2)}, ..., \lambda_v^{(\geq 4,L_{max})}\right)$ is the offset vector.

The optimization of the offset vector for an irregular LDPC code with distribution $(\lambda(x), \rho(x))$ is still performed by the maximization of the DE thresholds:

$$\left(\boldsymbol{\lambda}_v{}^{(2)*}, \boldsymbol{\lambda}_v{}^{(3)*}, \boldsymbol{\lambda}_v{}^{(\geq 4)*}, \alpha^*\right) = \arg \max_{(\boldsymbol{\lambda}_v{}^{(2)}, \boldsymbol{\lambda}_v{}^{(3)}, \boldsymbol{\lambda}_v{}^{(\geq 4)}, \alpha)} \left\{\tilde{\delta}\left(\lambda(x), \rho(x), q, \alpha, \boldsymbol{\lambda}_v{}^{(2)}, \boldsymbol{\lambda}_v{}^{(3)}, \boldsymbol{\lambda}_v{}^{(\geq 4)}\right)\right\}. \tag{3.26}$$

For two WIMAX LDPC codes studied in Section 3.6, we indicate in Table 3.13 the DE thresholds of the noiseless MS and noiseless OMS decoders using $L_{max} = 20$.

Table 3.14 summarizes the DE thresholds of the OMS-$\boldsymbol{\lambda_v}$ decoders using $L_{max} = 20$. From the results obtained, we can note: *(i)* the offset vector $\boldsymbol{\lambda}_v{}^{(2)}$ has to be all-zero for low precision $q = 3$ and $R \in \{1/2, 3/4\}$, *(ii)* for the degree $d_v > 2$ VNs, $\boldsymbol{\lambda}_v{}^{(3)*}$ and $\boldsymbol{\lambda}_v{}^{(\geq 4)*}$ show that the offset equal to 1 is applied mainly during the first iterations, while in the last iterations the last iterations the offset equal to 1 is not applied. The

Table 3.13 DE thresholds of noiseless MS decoders and noiseless OMS decoders with offset value $\lambda_v = 1$ for the WIMAX degree distribution using $L_{max} = 20$.

Irregular LDPC code, BI-AWGN channel					
R	q	λ_v	α^*	δ	δ_{db}
1/2	3 bits	0	0.53	0.7871453339	**2.0789**
		1	0.54	0.6148141689	4.2251
	4 bits	0	1.05	0.8124462941	**1.8041**
		1	1.17	0.7676937113	2.2962
3/4	3 bits	0	0.58	0.5830334538	**2.9252**
		1	0.71	0.5568327660	3.3246
	4 bits	0	1.12	0.5969529723	2.7203
		1	1.39	0.6109303460	**2.5193**

same happens for the largest precision $q = 4$ and $\boldsymbol{\lambda}_v{}^{(2)}$, *(iii)* the offset vector $\boldsymbol{\lambda}_v{}^{(3)}$ (resp. $\boldsymbol{\lambda}_v{}^{(2)}$) is almost all-zero for $(R = 1/2, q = 3)$ (resp. $(R = 1/2, q = 4)$).

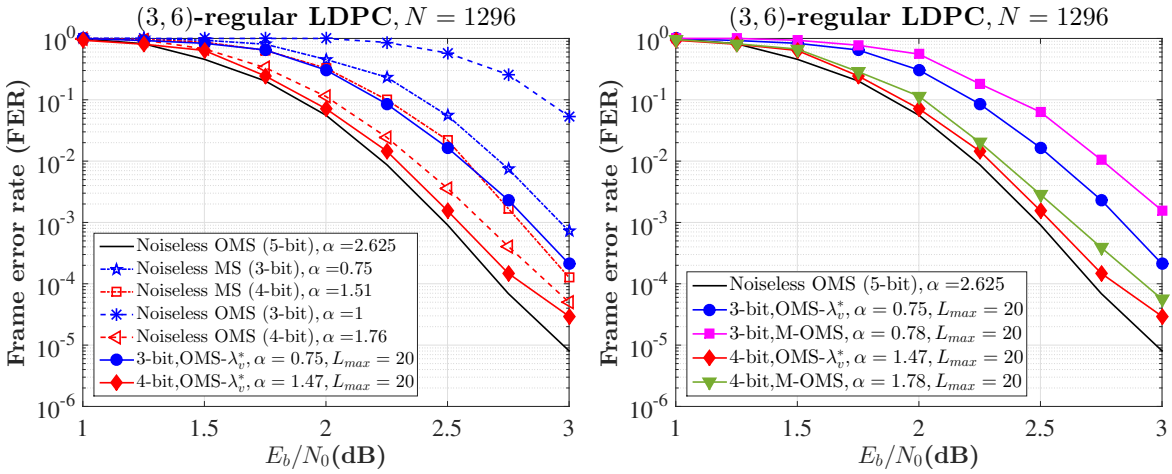
Comparing the best thresholds indicated in bold showed in Tables 3.14 and Table 3.13, we can note that the OMS-$\boldsymbol{\lambda_v}$ decoders achieve better DE thresholds than the MS and OMS decoders.

Table 3.14 DE thresholds of the OMS-$\boldsymbol{\lambda_v}$ decoders for the WIMAX degree distribution using $L_{max} = 20$.

Irregular LDPC code, BI-AWGN channel					
R	q	Υ	α^*	$\lambda_v^{(20)*}\lambda_v^{(19)*}...\lambda_v^{(2)*}\lambda_v^{(1)*}$	δ_{db}
1/2	3 bits	$\boldsymbol{\lambda}_v{}^{(2)*}$	0.52	00000000000000000000	**2.0173**
		$\boldsymbol{\lambda}_v{}^{(3)*}$		00000000000000001011	
		$\boldsymbol{\lambda}_v{}^{(\geq 4)*}$		00000000000111111111	
	4 bits	$\boldsymbol{\lambda}_v{}^{(2)*}$	1.03	00000000000000001001	**1.6718**
		$\boldsymbol{\lambda}_v{}^{(3)*}$		00000000010010101011	
		$\boldsymbol{\lambda}_v{}^{(\geq 4)*}$		00000000011011101111	
3/4	3 bits	$\boldsymbol{\lambda}_v{}^{(2)*}$	0.63	00000000000000000000	**2.7511**
		$\boldsymbol{\lambda}_v{}^{(3)*}$		00000000000010111111	
		$\boldsymbol{\lambda}_v{}^{(\geq 4)*}$		00000011111111111111	
	4 bits	$\boldsymbol{\lambda}_v{}^{(2)*}$	1.25	00000011110111111111	**2.4595**
		$\boldsymbol{\lambda}_v{}^{(3)*}$		00000000011111111111	
		$\boldsymbol{\lambda}_v{}^{(\geq 4)*}$		00001011010111111111	

3.9.2 Finite Length Performance

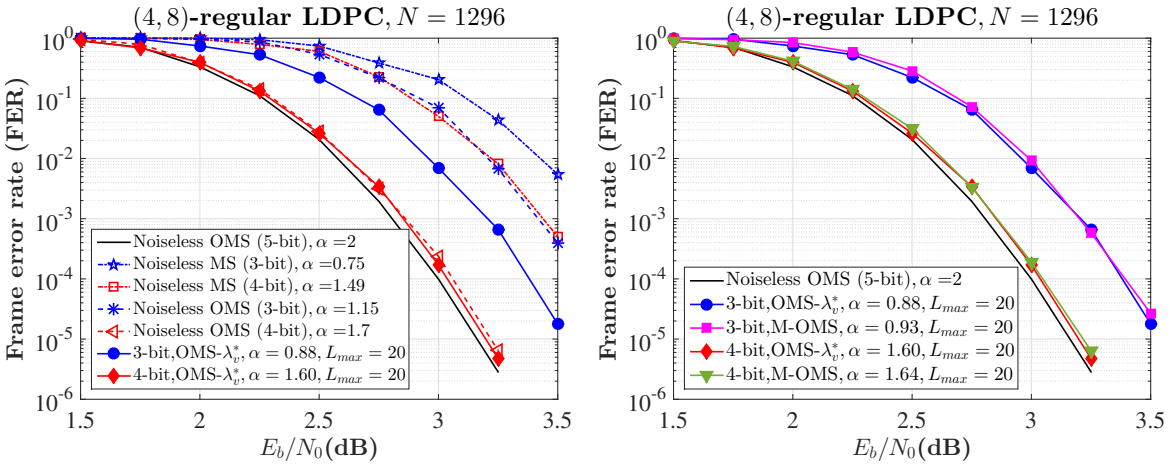
In this section, we present the FER performance of noiseless MS and OMS, M-OMS, and OMS-$\boldsymbol{\lambda_v}$ decoders. We analyze the OMS-$\boldsymbol{\lambda_v}$ decoder performance over the BI-AWGN channel using a maximum of 20 iterations. We use the $(d_v = 3, d_c = 6)$-regular QC-LDPC code, and the $(d_v = 4, d_c = 8)$-regular QC-LDPC code presented in Chapter 3.5. The considered decoders are the ones with the best DE thresholds, indicated in bold in Tables 3.10, 3.11, and 3.12. The noiseless 5-bit OMS decoder with offset $\lambda_v = 1$ is also shown as benchmark.



(a) OMS-$\boldsymbol{\lambda_v}$ decoders (b) OMS-$\boldsymbol{\lambda_v}$ and M-OMS decoders

Fig. 3.17 FER performance of OMS-$\boldsymbol{\lambda_v}$ and M-OMS decoders for the regular $d_v = 3$ LDPC code.

Fig. 3.17 and Fig. 3.18 show the FER performance comparisons between the 5-bit OMS decoder, the noiseless quantized decoders, the M-OMS decoders, and the OMS-$\boldsymbol{\lambda_v}$ decoders, as a function of E_b/N_0 over the BI-AWGN channel. From the results, we conclude that the finite length FER performances are in accordance with the DE gains. We observe that the 3-bit OMS-$\boldsymbol{\lambda_v}$ decoders have better FER performance than 3-bit MS and 3-bit OMS decoders. For the regular $d_v = 3$ LDPC code, the FER performance of the 4-bit OMS-$\boldsymbol{\lambda_v}$ decoder is better than the FER performance of the 4-bit MS

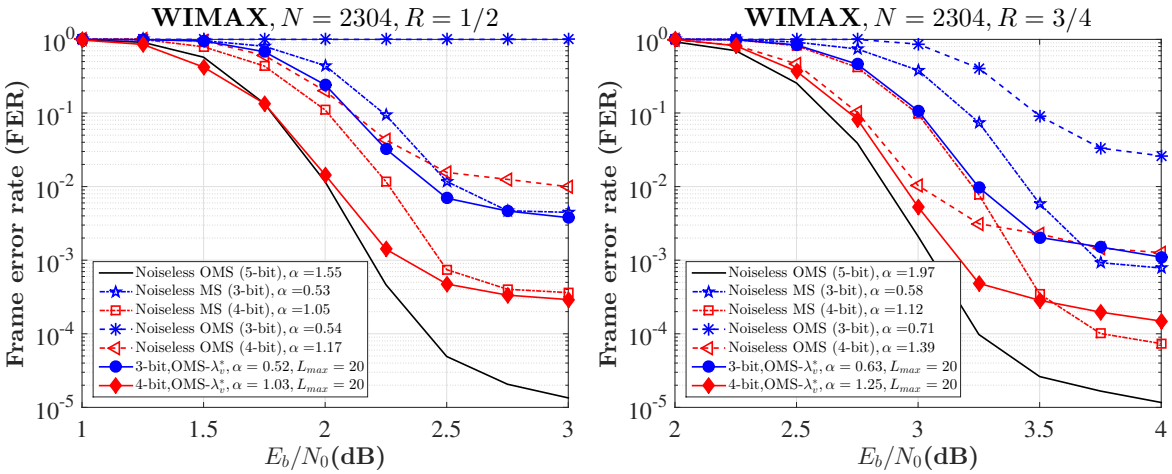


(a) OMS-$\boldsymbol{\lambda_v}$ decoders (b) OMS-$\boldsymbol{\lambda_v}$ and M-OMS decoders

Fig. 3.18 FER performance of OMS-$\boldsymbol{\lambda_v}$ and M-OMS decoders for the regular $d_v = 4$ LDPC code.

and the 4-bit OMS decoder. In the case of the regular $d_v = 4$ LDPC code, the 4-bit OMS-$\boldsymbol{\lambda_v}$ decoder and the 4-bit OMS have the same FER performance.

We can also note that the OMS-$\boldsymbol{\lambda_v}$ decoders have better FER performance than the M-OMS decoders for $d_v = 3$. On the other hand, for $d_v = 4$, the M-OMS decoders and OMS-$\boldsymbol{\lambda_v}$ decoders have almost the same performance.



(a) $R = 1/2$ (b) $R = 3/4$

Fig. 3.19 FER performance of OMS-$\boldsymbol{\lambda_v}$ decoders for the WIMAX LDPC code.

Simulation results for the WIMAX LDPC code with $R \in \{1/2, 3/4\}$ are provided in Fig. 3.19. For the two considered precisions $q = 3$ and $q = 4$, we can see that the OMS-$\boldsymbol{\lambda_v}$ decoders are better decoders than the MS and OMS decoders.

3.10 Conclusions

In this Chapter, we have first proposed a model to introduce randomness in highly quantized message passing decoder. Thanks to density evolution, we obtained the optimal parameters of "noise injection" in the asymptotic regime. The result shows that, for highly quantized decoder (3 or 4 bit of input quantization), the proposed Noise-Against-Noise Min-Sum decoder can improve significantly the decoding performance in the waterfall region (up to 0.4 dB). The finite-length Monte Carlo simulations for several code rates and degrees of variable node have corroborated the DE analysis. Then, the we have constraint the proposed NAN-MS decoder to become a deterministic decoder to simplify its hardware implementation. The obtained Modified Offset-Min-Sum decoder has performance closed to the NAN-MS and outperform classical MS and OMS when input message are quantized on 3 or 4 bits of precision. Moreover, the M-OMS decoder has equivalent complexity than the OMS decoder. Finally, in case where the decoder unrolled the decoding iterations, DE tool helps up to find optimal parameters of the decoder where each iteration uses the same type of Variable Node Unit. The obtained decoder is called OMS-$\boldsymbol{\lambda_v}$ where $\boldsymbol{\lambda_v}$ is a binary vector of the size of the maximum number of iterations.

Chapter 4

Sign-Preserving Min-Sum Decoders

This work proposes a new finite precision iterative decoder for low-density parity-check (LDPC) codes and for low complexity hardware implementation. The proposed decoder, named Sign-Preserving Min-Sum (SP-MS), significantly improves the decoding performance compared to classical Offset Min-Sum (OMS) decoder when messages are quantized on $q = 2$, 3 or 4 bits of precision. The particularity of the SP-MS decoder is that all messages can take profit of the full dynamic given by q bits of precision and, corollary, that the 0 value is never used. In other words, the SP-MS decoder forbids the 0 value in its message alphabet during the iterative decoding, *i.e.* a message cannot be erased. In order to achieve a high-throughput with low complexity hardware implementation, we investigate the SP-MS decoder defining a message alphabet constructed from 2, 3, or 4 bits of precision. In our research we also define a decoder input alphabet to quantize the Log-Likelihood Ratios (LLRs) values constructed from 3 or 4 bits of precision.

In order to optimize the SP-MS decoder performance, the optimization methodology using injection of noise during the iterative process (see Chapter 3) is also applied in the context of SP-MS decoders. The SP-MS decoder and its optimization are investigated in the asymptotic limit of the code length using a noisy version of density

evolution (DE). We use the optimization method proposed in the previous chapter. After optimization we can obtain two kinds of decoders: *(i)* a probabilistic decoder that keeps some randomness named Sign-Preserving Noise-Aided Min-Sum (SP-NA-MS) decoder, or *(ii)* a deterministic decoder that combines MS and OMS behaviors, named Sign-Preserving Min-Sum (SP-MS) decoder.

The outline of this chapter is as follows. The first section briefly analyzes the classical OMS-based decoders. In the second section, we present a quantification method used in SP-MS decoders. In the third section, we introduce the SP-MS decoders and we explain how to preserve the sign of exchanged messages of SP-MS decoders. In the fourth section, we explain how to optimize the SP-MS decoders. In the fifth section, we present the density evolution equations of SP-MS decoders. The sixth section shows how to compute the the asymptotic bit error probability. The seventh section explains how to compute the DE threshold. In the eighth section, we present the results of the asymptotic analysis of SP-MS decoders for regular and irregular LDPC codes. The ninth section shows finite length performance validation of the gains obtained with the proposed SP-MS decoders. In the tenth section, we present the convergence performance analysis of the SP-MS decoders, and the eleventh section concludes this chapter.

4.1 classical OMS-based Decoders

The message alphabet $\mathcal{A}_C = \{-N_q, .., -1, 0, +1, ..., +N_q\}$ of classical quantized decoders only uses $2^q - 1$ levels of a total of 2^q levels that can be used for a precision of q bits. For example, using $q = 3$ bits of precision, only 7 levels are used for $\mathcal{A}_C$.

In Chapter 3, the offset of the OMS decoders was moved from CNs to VNs allowing the c-to-v messages and v-to-c messages to use all values of $\mathcal{A}_C$. But the transfer of

the offset does not allow the decoder to use all the combinations that can be obtained for q bits of precision.

Analyzing the VNU of the classical OMS-based decoders, see 3.2, we can see that the value of the v-to-c message can be zero. In that case, the erased message, *i.e.* $m_{v_n \to c_m}^{(\ell+1)} = 0$, does not carry any information and does not participate in the convergence of the decoder. In this work, we propose a new type of decoder that always preserves the sign of the messages, with a modified VNU using a *sign preserving factor*, so that VNU never generates null or erased messages.

4.2 Quantization used for SP-MS Decoders

Using the sign-and-magnitude representation one can obtain a decoder input alphabet which is symmetric around zero and which is composed of $2^{q_{ch}}$ states. Hence the decoder input alphabet for SP-MS decoders is redefined as $\mathcal{A}_L = \{-N_{ch}, ..., -1, -0, +0, +1, ..., +N_{ch}\}$. Similarly, the message alphabet for SP-MS decoders denoted by $\mathcal{A}_S$ is defined as $\mathcal{A}_S = \{-N_q, ..., -1, -0, +0, +1, ..., +N_q\}$, consists of 2^q states. The sign of a message $m \in \mathcal{A}_S$ indicates the estimated bit value associated with the VN to or from which m is being passed while the magnitude $|m|$ of m represents its reliability. In this work, it is assumed that $2 \leq q \leq q_{ch}$. The alphabets $\mathcal{A}_L$ and $\mathcal{A}_S$ can be easily implemented in hardware because each value of $\mathcal{A}_L$ and $\mathcal{A}_S$ has a natural (sign, magnitude) binary representation. An example of the binary representation of $\mathcal{A}_C$ and $\mathcal{A}_S$ for $q = 3$ is shown in Table 4.1, one can see that -0 is represented by 100_2, $+0$ is represented by 000_2, etc. The distribution of the quantized LLR I_n using $q = 3$ bits of precision for the SP-MS decoder is depicted in Fig. 4.1.

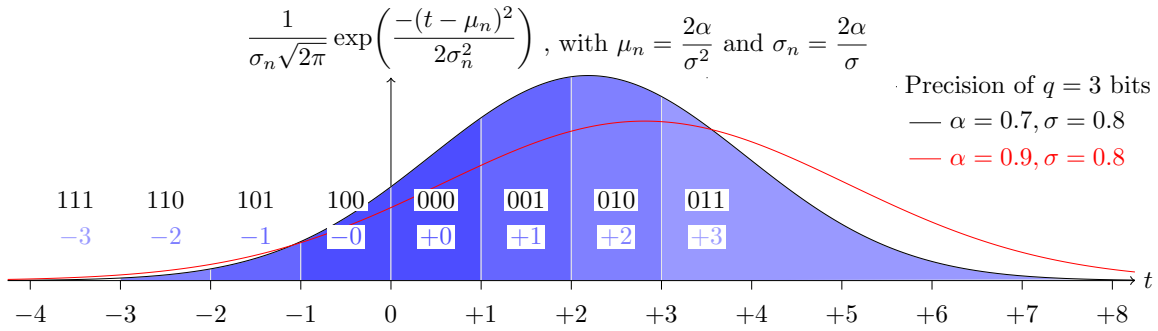


Fig. 4.1 $LLR(y_n)$ and I_n of the SP-MS decoder for the BI-AWGN channel and precision $q = 3$.

Table 4.1 Binary representation of the quantized values.

Classical Decoder		Sign Preserving Decoder		
$m \in \mathcal{A}_C$	$q = 3$ bits	$m \in \mathcal{A}_S$	$q = 3$ bits	$(\text{sign}(m), \lvert m \rvert)$
−3	101	−3	111	(−1,3)
−2	110	−2	110	(−1,2)
−1	111	−1	101	(−1,1)
–	100	−0	100	(−1,0)
0	000	+0	000	(+1,0)
+1	001	+1	001	(+1,1)
+2	010	+2	010	(+1,2)
+3	011	+3	011	(+1,3)

In order to obtain the quantized version of the LLRs belonging to $\mathcal{A}_L$, the quantization process defined in (2.17) is replaced by

$$\mathcal{Q}^*(a) = (\text{sign}(a), \mathcal{S}(\lceil \alpha \times |a| \rceil - 1, N_{ch})), \tag{4.1}$$

where $\lceil . \rceil$ depicts the ceiling function.

Then, I_n is thus defined as $I_n = \mathcal{Q}^*(LLR(y_n)) \in \mathcal{A}_L$ for $n = 1, \ldots, N$. In the initialization stage of the SP-MS decoders, *i.e.* at $\ell = 0$, the variable-to-check messages $m^{(\ell)}_{v_n \to c_m}$ are computed as $m^{(0)}_{v_n \to c_m} = \mathcal{S}(I_n, N_q)$ where $v_n \in \mathcal{V}(c_m)$, for $n = 1, ..., N$.

Let us define the update rules for Sign-Preserving Min-Sum decoders.

4.3 Sign-Preserving Min-Sum Decoders

The discrete functions Ψ_v, Ψ_c, and Ψ_a of SP-MS decoders are redefined by $\Psi_v : \mathcal{A}_L \times \mathcal{A}_S^{(d_v-1)} \to \mathcal{A}_S$, $\Psi_c : \mathcal{A}_S^{(d_c-1)} \to \mathcal{A}_S$, and $\Psi_a : \mathcal{A}_L \times \mathcal{A}_S^{(d_v)} \to \mathcal{A}_{app}$, respectively.

One can note from (3.1) that the CNU by construction determines the sign of each outgoing message, thus the CNU generates outgoing messages that always belong to $\mathcal{A}_S$, therefore, the CNU remains identical.

In the case of the VNU, (3.2) should be modified to ensure that the outgoing message will always belong to $\mathcal{A}_S$. To preserve always the sign of the messages, let us denote by $\mu_{v_n \to c_m}^{(\ell)}$ the *sign-preserving factor* of the message $m_{v_n \to c_m}^{(\ell+1)}$, defined as

$$\mu_{v_n \to c_m}^{(\ell)} = \xi \times \operatorname{sign}(I_n) + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \operatorname{sign}\left(m_{c \to v_n}^{(\ell)}\right), \tag{4.2}$$

where the values of ξ depends on the value of the column-weight d_v of a VN v_n, thus we have

$$\xi = \begin{cases} 0, & \text{if} \quad d_v = 2, \\ 1, & \text{if} \quad d_v > 2 \text{ and } (d_v \bmod 2) = 1, \\ 2, & \text{if} \quad d_v > 2 \text{ and } (d_v \bmod 2) = 0. \end{cases} \tag{4.3}$$

From (4.2), one can note that, by construction, $\mu_{v_n \to c_m}^{(\ell)}$ takes its values among $\{-1, +1\}$ for the special case of $d_v = 2$, $\{-d_v, -d_v+2, ..., -1, +1, ..., +d_v\}$ for d_v odd, and $\{-d_v-1, -d_v+1, \ldots, -1, +1, ..., +d_v+1\}$ for d_v even and $d_v > 2$. Thus, $\mu_{v_n \to c_m}^{(\ell)}$ is always an odd number.

Let us now redefine the unsaturated variable-to-check message $m_{v_n \to c_m}^{(\ell+1),U}$ as

$$m_{v_n \to c_m}^{(\ell+1),U} = \frac{\mu_{v_n \to c_m}^{(\ell)}}{2} + I_n + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} m_{c \to v_n}^{(\ell)}.$$

We can note that the fractional part of $(\mu_{v_n\to c_m}^{(\ell)})/2$ is 0.5. Hence, the alphabet of $m_{v_n\to c_m}^{(\ell+1),U}$ is given by $\mathcal{A}_U = \{-N_q \times (d_v - 1) - N_{ch} - (d_v - 1 + \xi)/2, ..., -1.5, -0.5, +0.5, +1.5, ..., +N_q \times (d_v - 1) + N_{ch} + (d_v - 1 + \xi)/2\}$.

Then, the update rule at a VNU of the SP-MS decoder with offset value λ_v is given by

$$m_{v_n\to c_m}^{(\ell+1)} = \Psi_v\left(I_n, \left\{m_{c\to v_n}^{(\ell)}\right\}_{c\in\mathcal{V}(v_n)\setminus\{c_m\}}\right) = \left(\text{sign}\left(m_{v_n\to c_m}^{(\ell+1),U}\right), \mathcal{S}\left(\max\left(\left\lfloor\left|m_{v_n\to c_m}^{(\ell+1),U}\right|\right\rfloor - \lambda_v, 0\right), N_q\right)\right). \tag{4.4}$$

The APP update at a VN v_n of the SP-MS decoder is defined as follows

$$\gamma_n^{(\ell)} = \Psi_a\left(I_n, \left\{m_{c\to v_n}^{(\ell)}\right\}_{c\in\mathcal{V}(v_n)}\right) = I_n + \tfrac{1}{2}\times\xi\times\text{sign}(I_n) + \sum_{c\in\mathcal{V}(v_n)}\left(m_{c\to v_n}^{(\ell)} + \frac{1}{2}\times\text{sign}\left(m_{c\to v_n}^{(\ell)}\right)\right). \tag{4.5}$$

The alphabet of APPs for SP-MS decoders is given by $\mathcal{A}_{app} = \{-N_q \times d_v - N_{ch} - (d_v + \xi)/2), ..., -1, 0, +1, ..., +N_q \times d_v + N_{ch} + (d_v + \xi)/2)\}$. From the APP, $\hat{x}_n$ can be computed as $\hat{x}_n = \text{sign}(I_n)$ if $\gamma_n^{(\ell)} = 0$, otherwise, $\hat{x}_n = \text{sign}(\gamma_n^{(\ell)})$ for $n = 1, ..., N$.

4.4 Optimization of Sign-Preserving Min-Sum Decoders

In order to optimize the Sign-Preserving Min-Sum decoders, we use the optimization method proposed in [51], *i.e.*, we introduce a certain level of randomness in the decoder during the optimization process. In this work, the optimization process is defined by injecting some randomness during the VNU processing, *i.e.* the noise perturbs unsaturated v-to-c messages $m_{v_n\to c_m}^{(\ell+1),U}$. Hence, the optimization process is given by

$$\tilde{m}_{v_n\to c_m}^{(\ell+1)} = \Upsilon\left(m_{v_n\to c_m}^{(\ell+1),U}\right), \tag{4.6}$$

where Υ is a noise model that also performs the saturation function.

After optimization we can obtain two kinds of decoders: (i) a decoder that keeps some randomness named Sign-Preserving Noise-Aided Min-Sum (SP-NA-MS) decoder, or (ii) a deterministic decoder named Sign-Preserving Min-Sum (SP-MS) decoder.

Let us now introduce the constraints on the noise models, and then let us present a noise model that we use to perturb the unsaturated v-to-c messages $m_{v_n \to c_m}^{(\ell+1),U}$.

4.4.1 Probabilistic Error Model to Optimize SP-MS Decoders

We assume that the noisy message alphabet is denoted by $\tilde{\mathcal{A}}_S$. The noisy message $\tilde{m}_{v_n \to c_m}^{(\ell+1)}$ is obtained after corrupting the noiseless message $m_{v_n \to c_m}^{(\ell+1),U}$ with noise. To simplify the notations in this section, we use m_u to denote any $m_{v_n \to c_m}^{(\ell+1),U}$ and $\tilde{m}$ to denote any $\tilde{m}_{v_n \to c_m}^{(\ell+1)}$.

In order to be able to perform DE analysis to optimize SP-MS decoders, the considered noise model need to be memoryless, *i.e.* the noise model has to be independent on data streams processed by the SP-MS decoders. Also, the considered noise model must satisfy the following condition of symmetry

$$\Pr(\tilde{m} = \psi_2 | m_u = \psi_1) = \Pr(\tilde{m} = -\psi_2 | m_u = -\psi_1), \forall \psi_1 \in \mathcal{A}_U \text{ and } \psi_2 \in \tilde{\mathcal{A}}_S.$$

When the noise model is memoryless and symmetric, it can be used to inject some randomness at the VNU during the optimization process of the SP-MS decoders, so the noisy-VNU is symmetric, allowing to use the all-zero codeword assumption and the independence assumption necessary in DE [27]. Since the addition of noise in VNUs is independent of the sign of the messages, we will suppose in the sequel without loss of generality that the messages m_u and $\tilde{m}$ are always positive.

Now, let us denote by $\Upsilon : \mathcal{A}_U \to \tilde{\mathcal{A}}_S$ the function which transforms $m_u \in \mathcal{A}_U$ into $\tilde{m} = \Upsilon(m_u) \in \tilde{\mathcal{A}}_S$ with the random process defined by the conditional probability density function (CPDF) $\Pr(\tilde{m} \mid m_u)$. Unless otherwise stated, the noisy message alphabet will be the one of the messages, *i.e.* $\mathcal{A}_S = \tilde{\mathcal{A}}_S$. We propose in this work to analyze a noise model Υ whose CPDF $\Pr(\tilde{m} \mid m_u)$, denoted by $p_\Upsilon(m_u, \tilde{m})$, is given by

$$p_\Upsilon(m_u, \tilde{m}) = \begin{cases} 1, & \text{if } \tilde{m} = +0, m_u = +0.5 \text{ or if } \tilde{m} = +N_q, m_u > +N_q + 0.5, \\ \varphi(m_u), & \text{if } \tilde{m} = \lfloor m_u \rfloor - 1, \forall m_u \in \{+1.5, +2.5, ..., +N_q + 0.5\}, \\ 1 - \varphi(m_u), & \text{if } \tilde{m} = \lfloor m_u \rfloor, \forall m_u \in \{+1.5, +2.5, ..., +N_q + 0.5\}, \\ 0, & \text{otherwise,} \end{cases} \tag{4.7}$$

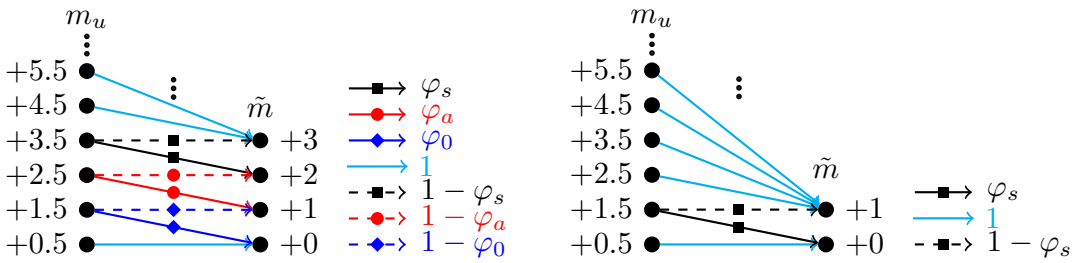
where $\varphi(m_u)$ is defined as

$$\varphi(m_u) = \begin{cases} \varphi_0, & \text{if } \quad m_u = +1.5, \\ \varphi_a, & \text{if } \quad m_u \in \{+2.5, +3.5, ..., +N_q - 0.5\}, \\ \varphi_s, & \text{if } \quad m_u = +N_q + 0.5, \end{cases} \tag{4.8}$$

The noise model analyzed is parametrized by three different transition probabilities $\boldsymbol{\varphi} = (\varphi_s, \varphi_a, \varphi_0)$. The choice of these three transition probabilities is a trade-off between complexity and the process of the border effects in the message alphabet $\mathcal{A}_S$.

The reasoning behind Υ is to implement a probabilistic offset with the purpose of always keeping the sign of the messages. Let us discuss the case of $\varphi_s = \varphi_a = \varphi_0$, the SP-MS decoder with offset value $\lambda_v = 1$ can be obtained as special case of Υ setting $\boldsymbol{\varphi} = (1, 1, 1)$, similarly, the SP-MS without offset, *i.e.* $\lambda_v = 0$, can be obtained setting $\boldsymbol{\varphi} = (0, 0, 0)$. Thanks to Υ, we can implement a probabilistic SP-MS decoder whose behaviour is a probabilistic weighted combination of a SP-MS decoder without offset and a SP-MS decoder with offset $\lambda_v = 1$. The effect of the noise on the extreme values of the message alphabet $\mathcal{A}_S$ is studied introducing two other probabilities φ_s

and φ_0. A special case occurs when $m_u = \pm 1.5$ because it will be mapped to $\tilde{m} = \pm 0$ with probability φ_0. Since $\tilde{m} = \pm 0$ propagates the sign but with a reliability of zero, therefore, φ_0 has to be analyzed differently than φ_a. Additionally, in finite precision messages, all values greater than N_q are saturated to N_q. As a result, many more configurations of the VNU states lead to an output message $\tilde{m} = N_q$ compared to other values of $\tilde{m}$, and φ_s should also be analyzed differently than φ_a. As an example, Υ is depicted in Fig. 4.2a for $(q = 3, N_q = 3)$.



(a) Precision $q = 3$ (b) Precision $q = 2$

Fig. 4.2 The mapping used for the noise model Υ.

It must be noted that the noise model Υ defined above is for the precision messages $q \geq 3$. In the case of very low precision messages $q = 2$, the noise model Υ is given only by one transition probability $\boldsymbol{\varphi} = (\varphi_s)$ because the saturation value is $N_q = 1$, hence $m_u = \pm 1.5$ will be mapped to $\tilde{m} = \pm 0$ with probability φ_s. For this special case, the message alphabet, which is only composed of four different values, is given by $\mathcal{A}_S = \{-1, -0, +0, +1\}$. Fig. 4.2b shows the noise model Υ for $(q = 2, N_q = 1)$.

4.5 Density Evolution for Sign-Preserving Decoders

The goal of DE [27, 28, 35] is to recursively compute the probability mass function (PMF) of the exchanged messages in the Tanner graph along the iterations. DE allows

us to predict if an ensemble of LDPC codes, parametrized by its degree distribution, decoded with a given MP decoder, converges to zero error probability in the limit of infinite block length.

In order to derive the DE equations for Sign-Preserving decoders, let $\Theta^{(\ell)}(k)$, $k \in \mathcal{A}_S$, denote the PMF of noiseless c-to-v messages in the ℓ^{th} iteration. Similarly, let $\Omega^{(\ell)}(k)$, $k \in \mathcal{A}_S$, denote the PMF of noiseless v-to-c messages in the ℓ^{th} iteration. Also, let $\Omega^{(0)}(k)$, $k \in \mathcal{A}_L$, be the initial PMF of messages sent at $\ell = 0$. To deduce the noisy DE equations, let $\tilde{\Omega}^{(\ell)}(k)$, $k \in \mathcal{A}_S$, denote the PMF of noisy v-to-c messages in the ℓ^{th} iteration. We consider that the all-zero codeword is sent over the BI-AWGN channel.

4.5.1 Initialization

DE is initialized with the PMF of the BI-AWGN channel with noise variance σ^2 as follows

$$\Omega^{(0)}(k) = \begin{cases} F(k) & \text{if} \quad k = -N_{ch} \\ F(k) - F(k-1) & \text{if} \quad -N_{ch} < k \leq -1 \\ F(0) - F(-1) & \text{if} \quad k = -0 \\ F(1) - F(0) & \text{if} \quad k = +0 \\ F(k+1) - F(k) & \text{if} \quad +1 \leq k < +N_{ch} \\ 1 - F(k) & \text{if} \quad k = +N_{ch} \end{cases} \tag{4.9}$$

where $F(k)$ is given by [34, 36, 37]:

$$F(k) = \frac{1}{\sqrt{2\pi}\sigma_n} \int_{-\infty}^{k} e^{-(t-\mu_n)^2/2\sigma_n^2} dt, \tag{4.10}$$

with $\sigma_n = (2/\sigma) \times \alpha$ and $\mu_n = (2/\sigma^2) \times \alpha$.

4.5.2 DE update for CNU

The input of a CNU is the PMF of the noisy messages going out of a noisy VNU, *i.e.* $\tilde{\Omega}^{(\ell)}$. For a CN of degree d_c, $\Theta_{d_c}^{(\ell)}$ is given by

$$\Theta_{d_c}^{(\ell)}(k) = \sum_{(i_1,\ldots,i_{d_c-1}):\Psi_c(i_1,\ldots,i_{d_c-1})=k} \tilde{\Omega}^{(\ell)}(i_1)\ldots\tilde{\Omega}^{(\ell)}(i_{d_c-1}),\ \forall k \in \mathcal{A}_S. \tag{4.11}$$

Considering the different connection degrees of CNs of irregular LDPC codes, we have

$$\Theta^{(\ell)}(k) = \sum_{d_c=2}^{d_{c,max}} \rho_{d_c} \times \Theta_{d_c}^{(\ell)}(k), \quad \forall k \in \mathcal{A}_S \tag{4.12}$$

4.5.3 DE update for VNU

We know that Υ perturbs unsaturated values. For this reason, we first compute the PMF of unsaturated v-to-c messages of a VN of degree d_v, *i.e.* $\Omega_{d_v}^{(\ell+1),U}$, with the following equation

$$\Omega_{d_v}^{(\ell+1),U}(k) = \sum_{(t,i_1,\ldots,i_{d_v-1}):\Psi_v(t,i_1,\ldots,i_{d_v-1})=k} \Omega^{(0)}(t)\Theta^{(\ell)}(i_1)\ldots\Theta^{(\ell)}(i_{d_v-1}),\ \forall k \in \mathcal{A}_U. \tag{4.13}$$

And second, the noise effect is added to the PMF of unsaturated v-to-c messages to obtain the corrupted PMF

$$\tilde{\Omega}_{d_v}^{(\ell+1)}(k) = \sum_{i \in \mathcal{A}_U} \Omega_{d_v}^{(\ell+1),U}(i) \times p_\Upsilon(i,k),\ \forall k \in \mathcal{A}_S, \tag{4.14}$$

where p_Υ, expressed in (4.7), is the transition probability of the VN noise.

In this work we use only the transition probabilities of the noise model Υ defined in Section 4.4. Although of course other noise models can be used.

Then the effect of the different connection degrees of VNs is considered using the following relation

$$\tilde{\Omega}^{(\ell+1)}(k) = \sum_{d_v=2}^{d_{v,max}} \lambda_{d_v} \times \tilde{\Omega}_{d_v}^{(\ell+1)}(k), \quad \forall k \in \mathcal{A}_S \tag{4.15}$$

In the optimization process of SP-MS decoders, where the effect of noise injection is added at VNUs, the DE update of noisy-VNU is implemented with (4.13), (4.14), and (4.15). The DE update of VNU for SP-MS decoders without noise injection can be obtained setting $p_\Upsilon(i,k) = 1$ if $i = k$, otherwise $p_\Upsilon(i,k) = 0$, *i.e.* $\boldsymbol{\varphi} = (0,0,0)$ which corresponds to the offset $\lambda_v = 0$.

4.6 Asymptotic Bit Error Probability

The asymptotic bit error probability can be deduced from the PMF of the APPs, which is obtained from the DE equations. Let $p_e^{(\ell)}$ denote the bit error probability at iteration ℓ, which is computed from the PMF of all incoming messages to a VN in the ℓ^{th} iteration, and defined by

$$p_e^{(\ell)} = \frac{1}{2}\Gamma^{(\ell)}(0) + \sum_{i=-(N_q \times d_{v,max} + N_{ch} + (d_{v,max}+\xi)/2)}^{-1} \Gamma^{(\ell)}(i), \tag{4.16}$$

where $\Gamma^{(\ell)}(k)$, $k \in \mathcal{A}_{app}$, denotes the PMF of the APP at the end of the ℓ^{th} iteration for Sign-Preserving decoders. We can compute $\Gamma^{(\ell)}(k)$ as follows

$$\Gamma^{(\ell)}(k) = \sum_{d_v=2}^{d_{v,max}} \lambda_{d_v} \times \Gamma_{d_v}^{(\ell)}(k), \quad \forall k \in \mathcal{A}_{app}$$

where $\Gamma_{d_v}^{(\ell)}(k)$ is computed as

$$\Gamma_{d_v}^{(\ell)}(k) = \sum_{(t,i_1,\ldots,i_{d_v}):\Psi_a(t,i_1,\ldots,i_{d_v})=k} \Omega^{(0)}(t)\Theta^{(\ell)}(i_1)\ldots\Theta^{(\ell)}(i_{d_v}),\ \forall k \in \mathcal{A}_{app}.$$

The evolution of $p_e^{(\ell)}$ with the iterations characterizes whether the Sign Preserving decoder converges or diverges in the asymptotic limit of the codeword length. When the number of iterations ℓ goes to infinity, we obtain the asymptotic error probability $p_e^{(+\infty)}$.

For the SP-MS decoder which is a noiseless decoder, the decoder converges to zero error probability and successful decoding is declared, *i.e.* $p_e^{(+\infty)} = 0$. In the case of SP-NA-MS decoders, contrary to the noiseless case, $p_e^{(+\infty)}$ is not necessarily equal to zero when the noisy DE converges and corrects the channel noise. For noisy decoders, the value of $p_e^{(+\infty)}$ depends mainly on the chosen error model and the computing units to which it is applied [35].

4.7 Density Evolution threshold

The concepts and procedure for calculating the DE thresholds are given in Chapter 3 (Section 3.4.6). In this work, we use $\tilde{\delta}$ to jointly optimize the noise model parameters $(\varphi_s, \varphi_a, \varphi_0)$ and the channel gain factor α for a fixed precision (q_{ch}, q) and a fixed degree distribution $(\lambda(x), \rho(x))$ as follows

$$(\varphi_s^*, \varphi_a^*, \varphi_0^*, \alpha^*) = \arg \max_{(\varphi_s,\varphi_a,\varphi_0,\alpha)} \left\{ \tilde{\delta}\left(\lambda(x), \rho(x), q_{ch}, q, \alpha, (\varphi_s, \varphi_a, \varphi_0)\right) \right\}. \tag{4.17}$$

The optimization of the transition probabilities of the noise model Υ, and the channel gain factor α is made using a *greedy* algorithm which computes a local maximum DE

threshold. For noiseless decoders, the optimization (4.17) is reduced to the optimum channel gain factor α^* which is computed performing a grid-search.

4.8 Asymptotic Analysis of Sign-Preserving Min-Sum Decoders

This section presents the asymptotic analysis of SP-MS decoders with the values of ξ defined in (4.3). One can note that other values of ξ (different from those defined in (4.3)) give worse decoding performance.

4.8.1 Asymptotic Analysis of SP-MS Decoders for Regular LDPC codes

For all results presented in this section, we consider the ensemble of (d_v, d_c)-regular LDPC codes with various code rate R for $d_v \in \{3, 4, 5, 6\}$, and quantized decoders with precision $q_{ch} \in \{3, 4\}$ and $q \in \{2, 3, 4\}$.

We draw the evolution of SP-MS thresholds versus the check-node degree d_c on Fig. 4.3, while keeping the variable-node degree d_v constant. We also show on Fig. 4.4 the DE threshold of SP-MS decoders as a function of the variable-node degree d_v for $d_c \in \{12, 20, 32\}$. Several conclusions can be derived from Fig. 4.3 and Fig. 4.4.

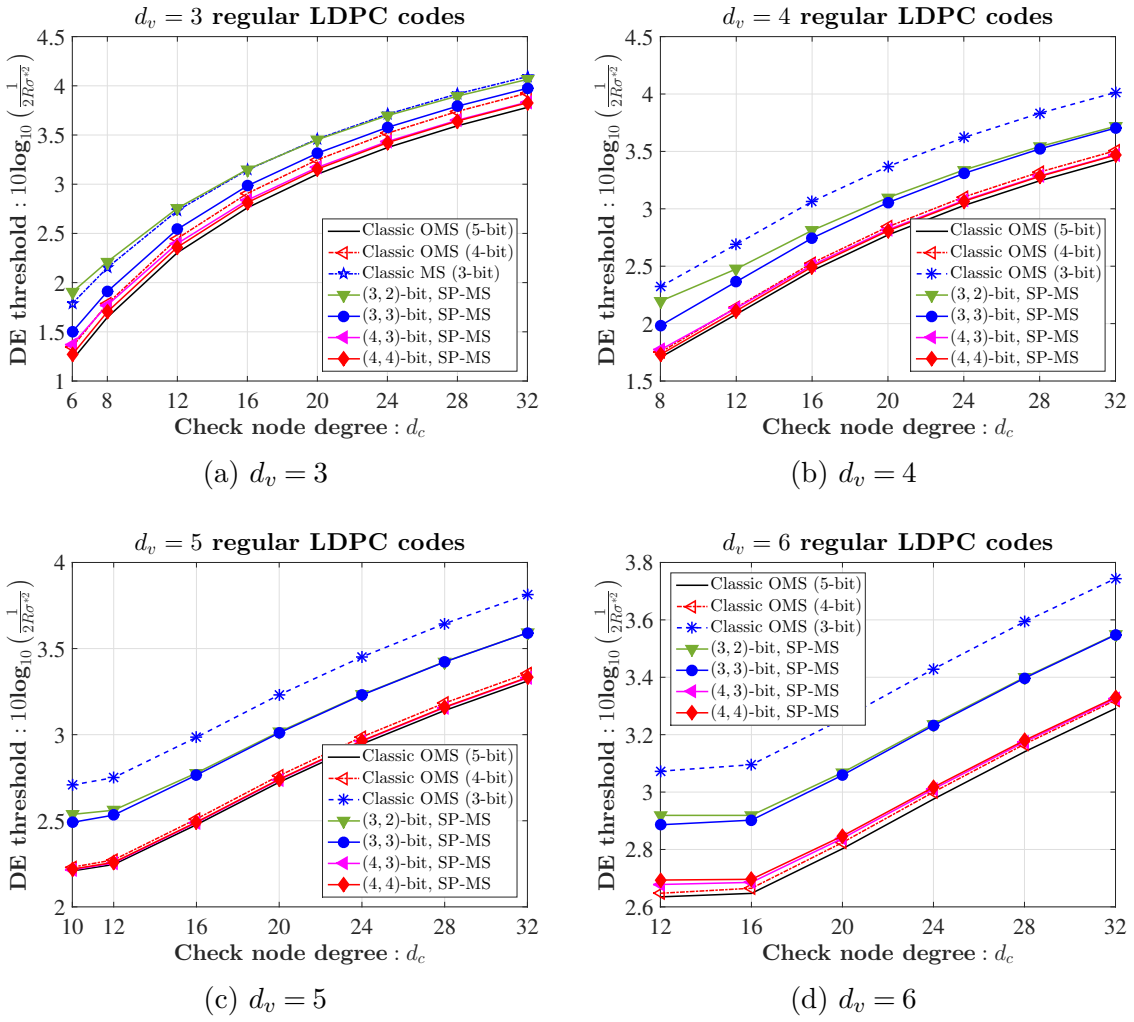
1. First, the DE thresholds of the SP-MS decoders are always better than the DE thresholds for noiseless classical decoders when the same precision is used. Also, the $(q_{ch} = 4, q = 4)$-bit SP-MS decoders can almost achieve the same performance as 5-bit OMS decoders because the DE thresholds of $(q_{ch} = 4, q = 4)$-bit SP-MS decoders are very close to the DE thresholds of the 5-bit OMS decoders.

2. Second, one can see that for regular LDPC codes with $d_v > 2$, the DE thresholds of $(q_{ch} = 4, q = 4)$-bit SP-MS decoders are almost equal or equal to the DE thresholds of $(q_{ch} = 4, q = 3)$-bit SP-MS decoders, except for the regular $d_v = 3$ LDPC codes with $d_c < 16$, in that case the DE thresholds of the $(q_{ch} = 4, q = 3)$-bit SP-MS decoders are worse than the $(q_{ch} = 4, q = 4)$-bit SP-MS decoders. We can also observe that the $(q_{ch} = 3, q = 3)$-bit SP-MS decoders have the same DE thresholds as $(q_{ch} = 3, q = 2)$-bit SP-MS decoders for the regular $d_v \geq 4$ LDPC codes, except for the regular $d_v = 4$ LDPC codes with $d_c < 24$ that exhibit worse DE thresholds, while in case of $d_v = 3$ LDPC codes, the $(q_{ch} = 3, q = 3)$-bit SP-MS decoders exhibit better DE thresholds than the $(q_{ch} = 3, q = 2)$-bit SP-MS decoders.

   From this analysis, we can conclude that the SP-MS decoders with precision q_{ch} can be implemented using very low precision $q = q_{ch} - 1$ for the *extrinsic* messages. This implies a considerable reduction in the complexity of the CNs and the VNs, as well as a reduction in the number of wires in an implementation, *e.g* an ASIC implementation.

3. Third, one can note that the difference between the DE thresholds of the $(q_{ch}, q = q_{ch})$-bit SP-MS decoders and $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoders decreases as d_c increases, this phenomenon is very evident when the precision $q_{ch} = 3$ is used.

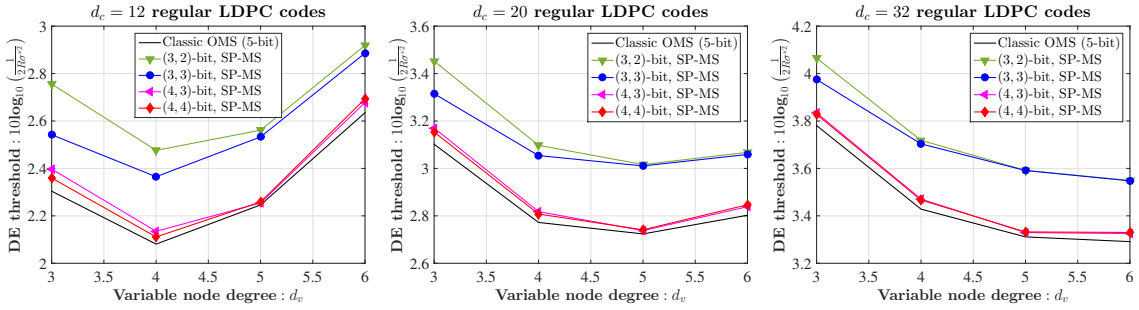
In Table 4.2 and 4.4, we present more detailed results for the ensemble of (d_v, d_c)-regular LDPC codes with code rate $R \in \{1/2, 3/4\}$ for $d_v \in \{3, 4, 5\}$, $R = 0.8413$ and $d_v = 6$ for the IEEE 802.3 ETHERNET code [6], $R = 0.94$ and $d_v = 4$ for Flash Memory [9]. The DE thresholds of the noiseless classical MS and OMS decoders are given in Table 4.2. It can be seen that the OMS is almost always superior to the MS for the considered cases, except for the regular $d_v = 3$ LDPC codes and the regular $(d_v = 4, d_c = 64)$ LDPC codes with low precision $q_{ch} = q = 3$.



(a) $d_v = 3$ (b) $d_v = 4$

(c) $d_v = 5$ (d) $d_v = 6$

Fig. 4.3 DE thresholds of optimized SP-MS decoders for (d_v, d_c)-regular LDPC codes.



(a) $d_c = 12$ (b) $d_c = 20$ (c) $d_c = 32$

Fig. 4.4 DE thresholds of optimized SP-MS decoders, as a function of the VN degree d_v.

Table 4.2 DE thresholds of classical MS and OMS decoders.

(d_v, d_c)-regular LDPC code, BI-AWGN channel									
		$(d_v = 3, d_c = 6)$		$(d_v = 4, d_c = 8)$		$(d_v = 5, d_c = 10)$		$(d_v = 6, d_c = 32)$	
$q_{ch} = q$	λ_v	α^*	δ_{db}	α^*	δ_{db}	α^*	δ_{db}	α^*	δ_{db}
3 bits	0	0.9375	**1.7888**	0.8125	2.7360	0.63	3.4117	0.455	4.0812
	1	1.0625	2.2039	1.25	**2.3219**	1.15	**2.7079**	0.84	**3.5928**
4 bits	0	2.0	1.6437	1.625	2.5389	1.25	3.1772	1.035	3.8154
	1	1.875	**1.3481**	1.75	**1.7509**	1.59	**2.2306**	1.28	**3.1685**
5 bits	0	4.0	1.6132	3.25	2.4948	2.30	3.1126	1.985	3.7506
	1	2.625	**1.2154**	2.0	**1.7061**	1.69	**2.2089**	1.45	**3.1400**
		$(d_v = 3, d_c = 12)$		$(d_v = 4, d_c = 16)$		$(d_v = 5, d_c = 20)$		$(d_v = 4, d_c = 64)$	
$q_{ch} = q$	λ_v	α^*	δ_{db}	α^*	δ_{db}	α^*	δ_{db}	α^*	δ_{db}
3 bits	0	0.625	**2.7316**	0.6875	3.1550	0.56	3.6449	0.50	**4.7599**
	1	0.9375	3.1343	0.9375	**3.0632**	0.92	**3.2312**	0.69	4.9036
4 bits	0	1.25	2.5646	1.375	2.9441	1.40	3.3917	1.06	4.5723
	1	1.5	**2.4484**	1.5	**2.5292**	1.39	**2.7620**	1.15	**4.4211**
5 bits	0	2.5	2.5268	2.75	2.8991	2.47	3.3373	2.20	4.5340
	1	2.25	**2.3040**	1.875	**2.4606**	1.61	**2.7238**	1.73	**4.3380**

In Table 4.4, we indicate the noisy and noiseless DE thresholds obtained with (4.17) for optimized Sign-Preserving decoders, we also show the DE gains obtained comparing the best thresholds indicated in bold in Table 4.2 and the noisy (resp. noiseless) thresholds of SP-NA-MS decoders (resp. SP-MS decoders) for different precision q_{ch} and q. Many more conclusions can be derived from this analysis.

4. Fourth, the DE thresholds of the SP-MS decoders are very close or equal to the DE thresholds of the SP-NA-MS decoders, specially for the regular $d_v > 3$ LDPC codes, since the values of the transition probabilities are close or equal to 0 or 1. This implies that the noise injected to the SP-MS decoders during the optimization process can be ignored in a hardware realization, thus avoiding the implementation of Random Generators (RGs) to perturb the v-to-c messages. Of course, in all cases the injection of noise cannot be ignored, this depends on the VN degree and the precision used.

The DE gains for the SP-MS decoders are quite important for low precision ($q_{ch} = 3, q = 3$), and very low precision ($q_{ch} = 3, q = 2$). The largest gain obtained is around 0.3399 dB for the regular ($d_v = 4, d_c = 8$) LDPC code and precision ($q_{ch} = 3, q = 3$). While the DE gains are smaller for the largest precision ($q_{ch} = 4, q = 4$) and ($q_{ch} = 4, q = 3$). We can observe a loss of around 0.0102 dB for the regular ($d_v = 6, d_c = 32$) LDPC code and precision ($q_{ch} = 4, q = 4$).

From this analysis, we can conclude that the preservation of the sign of messages is more and more beneficial as the decoders are implemented in low precision.

5. A Fifth remark comes from the interpretation of the optimum $\boldsymbol{\varphi}^*$ obtained through the DE analysis. For the case of precision messages $q > 2$, we have $\varphi_0^* = 0$ for regular $d_v = 3$ LDPC codes, this makes sense because $d_v = 3$ is small enough to transform $m_u = \pm 1.5$ into $\tilde{m} = \pm 0$, which gives to $\tilde{m}_{v_n \to c_m}^{(\ell)}$ a reliability of zero and which could not help to *extrinsic* messages become more and more reliable at each new decoding iteration. In the case of very low precision messages $q = 2$ and regular $2 < d_v < 5$ LDPC codes, we obtain φ_s^* close to 0 for SP-NA-MS decoders and $\varphi_s^* = 0$ for SP-MS decoders, this also makes sense because $q = 2$ is too small and transform $m_u = \pm 1.5$ into $\tilde{m} = \pm 0$ sets the maximum reliability (which is 1) to zero.

   For SP-MS decoders, when using the precision $q > 2$ and regular $d_v > 3$ LDPC codes, we have always $\varphi_0^* = 1$. In the case of very low precision $q = 2$ we obtain always $\varphi_s^* = 1$ for regular $d_v = 5$ and $d_v = 6$ LDPC codes. Hence, one can conclude that for ($d_v > 3, q > 2$), the transformation from $m_u = \pm 1.5$ to $\tilde{m} = \pm 0$, does not affect the decoding process, a similar conclusion is obtained for $d_v = 5$ and $d_v = 6$ using very low precision $q = 2$.

6. Finally, for SP-MS decoders and precision messages $q > 2$, the optimum noise parameters $\boldsymbol{\varphi}^*$ are equal to $(\varphi_s^*, \varphi_a^*, \varphi_0^*) = (1, 1, 0)$ for the regular $d_v = 3$ LDPC codes, while in the case of the regular $d_v > 3$ LDPC codes, $\boldsymbol{\varphi}^*$ are equal to $(\varphi_s^*, \varphi_a^*, \varphi_0^*) = (1, 1, 1)$ which correspond to a deterministic SP-MS decoder with offset $\lambda_v = 1$.

   For very low precision messages $q = 2$, the only transition probability $\boldsymbol{\varphi}^*$ is equal to $(\varphi_s^*) = (0)$ for the regular $d_v < 5$ LDPC codes which correspond to a SP-MS decoder without offset, while for the regular $d_v \geq 5$ LDPC codes, we obtain that $\boldsymbol{\varphi}^*$ is equal to $(\varphi_s^*) = (1)$ which correspond to a SP-MS decoder with offset $\lambda_v = 1$.

When comparing the DE thresholds of the SP-MS decoders (see Table 4.4) and the NAN-MS decoders (see Table 3.2 and Table 4.3), one can observe that the SP-MS decoders achieve better DE thresholds for almost all (d_v, d_c)-regular LDPC codes tested, the only exception appears for the regular $(d_v = 6, d_c = 32)$ LDPC code and $q = 4$. The largest gain obtained, when comparing the SP-MS thresholds and NAN-MS thresholds, is around 0.1803 dB for the regular $(d_v = 6, d_c = 32)$ LDPC code and $q = 3$. We can conclude that the SP-MS decoders are better decoders compared to the NAN-MS, the OMS, and the MS decoders.

Table 4.3 Noisy DE thresholds of NAN-MS decoders using the NIV model

NAN-MS decoders, (d_v, d_c)-regular LDPC code, BI-AWGN channel							
(d_v, d_c)	(q_{ch}, q)	Method	α^*	φ_s^*	φ_a^*	φ_0^*	$\tilde{\delta}_{db}$
$(4, 64)$	$(3, 3)$	Υ-NIV	0.57	0.988	0.988	0.000	**4.6171**
	$(4, 4)$		1.07	1.000	1.000	0.000	**4.3872**
$(5, 10)$	$(3, 3)$	Υ-NIV	0.94	1.000	1.000	0.338	**2.6417**
	$(4, 4)$		1.59	1.000	1.000	1.000	**2.2306**
$(5, 20)$	$(3, 3)$	Υ-NIV	0.74	0.987	0.987	0.225	**3.1400**
	$(4, 4)$		1.36	1.000	1.000	0.775	**2.7596**
$(6, 32)$	$(3, 3)$	Υ-NIV	0.69	1.000	1.000	0.425	**3.5766**
	$(4, 4)$		1.28	1.000	1.000	0.975	**3.1685**

Table 4.4 Noisy DE thresholds of SP-NA-MS decoders and DE thresholds of SP-MS decoders.

		SP-NA-MS decoders						SP-MS decoders			
(d_v, d_c)	(q_{ch}, q)	α^*	φ_s^*	φ_a^*	φ_0^*	$\tilde{\delta}_{db}$	DE gain	α^*	$(\varphi_s^*, \varphi_a^*, \varphi_0^*)$	δ_{db}	DE gain
$(3,6)$	$(3,2)$	0.48	0.050	–	–	**1.9033**	–	0.48	$(0,-,-)$	**1.9315**	–
	$(3,3)$	0.96	0.987	0.712	0.000	**1.4994**	**0.2894**	0.95	(1, 1, 0)	**1.5096**	**0.2792**
	$(4,3)$	1.18	0.912	0.625	0.000	**1.3726**	–	1.16	(1, 1, 0)	**1.3910**	–
	$(4,4)$	1.79	1.000	1.000	0.000	**1.2688**	**0.0793**	1.79	(1, 1, 0)	**1.2688**	**0.0793**
$(3,12)$	$(3,2)$	0.43	0.038	–	–	**2.7558**	–	0.43	$(0,-,-)$	**2.7696**	–
	$(3,3)$	0.72	1.000	0.725	0.000	**2.5421**	**0.1895**	0.71	(1, 1, 0)	**2.5468**	**0.1848**
	$(4,3)$	1.04	0.888	0.500	0.000	**2.3973**	–	1.01	(1, 1, 0)	**2.4115**	–
	$(4,4)$	1.34	1.000	0.938	0.000	**2.3596**	**0.0888**	1.36	(1, 1, 0)	**2.3600**	**0.0884**
$(4,8)$	$(3,2)$	0.67	0.350	–	–	**2.1952**	–	0.74	$(0,-,-)$	**2.2353**	–
	$(3,3)$	1.01	0.900	1.000	1.000	**1.9820**	**0.3399**	1.01	(1, 1, 1)	**1.9824**	**0.3395**
	$(4,3)$	1.44	1.000	1.000	1.000	**1.7720**	–	1.44	(1, 1, 1)	**1.7720**	–
	$(4,4)$	1.54	1.000	1.000	1.000	**1.7306**	**0.0203**	1.54	(1, 1, 1)	**1.7306**	**0.0203**
$(4,16)$	$(3,2)$	0.61	0.313	–	–	**2.8115**	–	0.66	$(0,-,-)$	**2.8411**	–
	$(3,3)$	0.75	1.000	0.962	0.712	**2.7448**	**0.3184**	0.78	(1, 1, 1)	**2.7459**	**0.3173**
	$(4,3)$	1.27	1.000	1.000	1.000	**2.5092**	–	1.27	(1, 1, 1)	**2.5092**	–
	$(4,4)$	1.30	1.000	1.000	1.000	**2.4941**	**0.0351**	1.30	(1, 1, 1)	**2.4941**	**0.0351**
$(4,64)$	$(3,2)$	0.53	0.300	–	–	**4.6034**	–	0.57	$(0,-,-)$	**4.6235**	–
	$(3,3)$	0.53	1.000	0.988	0.338	**4.6015**	**0.1584**	0.57	(1, 1, 1)	**4.6120**	**0.1479**
	$(4,3)$	1.05	1.000	1.000	1.000	**4.3791**	–	1.05	(1, 1, 1)	**4.3791**	–
	$(4,4)$	1.04	1.000	1.000	1.000	**4.3790**	**0.0421**	1.04	(1, 1, 1)	**4.3790**	**0.0421**
$(5,10)$	$(3,2)$	1.02	0.825	–	–	**2.5371**	–	1.05	$(1,-,-)$	**2.5445**	–
	$(3,3)$	1.12	1.000	1.000	0.987	**2.4908**	**0.2171**	1.12	(1, 1, 1)	**2.4908**	**0.2171**
	$(4,3)$	1.64	1.000	1.000	1.000	**2.2149**	–	1.64	(1, 1, 1)	**2.2149**	–
	$(4,4)$	1.57	1.000	1.000	1.000	**2.2196**	**0.0110**	1.57	(1, 1, 1)	**2.2196**	**0.0110**
$(5,20)$	$(3,2)$	0.86	0.800	–	–	**3.0164**	–	0.88	$(1,-,-)$	**3.0219**	–
	$(3,3)$	0.87	1.000	1.000	0.838	**3.0106**	**0.2206**	0.89	(1, 1, 1)	**3.0137**	**0.2175**
	$(4,3)$	1.42	1.000	1.000	1.000	**2.7379**	–	1.42	(1, 1, 1)	**2.7379**	–
	$(4,4)$	1.39	1.000	1.000	1.000	**2.7412**	**0.0208**	1.39	(1, 1, 1)	**2.7412**	**0.0208**
$(6,32)$	$(3,2)$	0.74	1.000	–	–	**3.3979**	–	0.74	$(1,-,-)$	**3.3979**	–
	$(3,3)$	0.74	1.000	1.000	1.000	**3.3963**	**0.1965**	0.74	(1, 1, 1)	**3.3963**	**0.1965**
	$(4,3)$	1.22	1.000	1.000	1.000	**3.1740**	–	1.22	(1, 1, 1)	**3.1740**	–
	$(4,4)$	1.18	1.000	1.000	1.000	**3.1787**	**−0.0102**	1.18	(1, 1, 1)	**3.1787**	**−0.0102**

4.8.2 Asymptotic Analysis of SP-MS Decoders for Irregular LDPC codes

Analyzing the regular LDPC codes we have seen that the optimum noise parameters $(\varphi_s^*, \varphi_a^*, \varphi_0^*)$ and the respective gains of SP-MS and SP-NA-MS decoders depend on the VN degree. Therefore, in order to optimize the SP-MS decoders for the LDPC codes with irregular VN distribution, we extend our optimization approach by considering a noise injection model Υ with different values of the transition probabilities for the different connection degrees. The precision considered for this optimization is $q_{ch} = q \in \{3, 4\}$.

Similar to Chapter 3 (Section 3.6), we consider $\Upsilon^{(2)} : \boldsymbol{\varphi}^{(2)} = \left(\varphi_s^{(2)}, \varphi_a^{(2)}, \varphi_0^{(2)}\right)$ for $d_v = 2$, $\Upsilon^{(3)} : \boldsymbol{\varphi}^{(3)} = \left(\varphi_s^{(3)}, \varphi_a^{(3)}, \varphi_0^{(3)}\right)$ for $d_v = 3$, and $\Upsilon^{(\geq 4)} : \boldsymbol{\varphi}^{(\geq 4)} = \left(\varphi_s^{(\geq 4)}, \varphi_a^{(\geq 4)}, \varphi_0^{(\geq 4)}\right)$ for $d_v \geq 4$.

The optimization of the transition probabilities for an irregular LDPC code with distribution $(\lambda(x), \rho(x))$ is still performed by the maximization of the noisy DE thresholds:

$$\left(\boldsymbol{\varphi}^{(2)*}, \boldsymbol{\varphi}^{(3)*}, \boldsymbol{\varphi}^{(\geq 4)*}, \alpha^*\right) = \arg\max_{(\boldsymbol{\varphi}^{(2)}, \boldsymbol{\varphi}^{(3)}, \boldsymbol{\varphi}^{(\geq 4)}, \alpha)} \left\{\tilde{\delta}\left(\lambda(x), \rho(x), q_{ch}, q, \alpha, \boldsymbol{\varphi}^{(2)}, \boldsymbol{\varphi}^{(3)}, \boldsymbol{\varphi}^{(\geq 4)}\right)\right\}. \tag{4.18}$$

The two WIMAX LDPC codes studied in Chapter 3 are used for our analysis, *i.e.* the WIMAX codes with rate $R \in \{1/2, 3/4\}$ and length $N = 2304$.

Noisy DE thresholds are summarized in Table 4.5, where we indicate the optimum channel gain factor α^* and the optimum noise parameters $\left(\boldsymbol{\varphi}^{(2)*}, \boldsymbol{\varphi}^{(3)*}, \boldsymbol{\varphi}^{(\geq 4)*}\right)$ obtained during the optimization process. Those results confirm the conclusions of the regular LDPC codes analysis: *(i)* the DE thresholds of SP-MS and SP-NA-MS decoders are better than the DE thresholds of NAN-MS, MS, and OMS decoders using the same precision (see Tables 3.4, 3.5, and 4.5), *(ii)* the DE thresholds of SP-MS decoders are

very close or equal to the DE thresholds of SP-NA-MS decoders since the optimized values of the transition probabilities are very close to 0 or 1, and *(iii)* the optimum value for φ_0^* is 0 or it is close to 0 for $d_v = 3$ VNs and for precision $q = 3$.

Table 4.5 DE thresholds of SP-NA-MS and SP-MS decoders for the WIMAX degree distribution.

		SP-NA-MS decoders							SP-MS decoders			
R	$q_{ch} = q$	α^*	d_v	φ_s^*	φ_a^*	φ_0^*	$\tilde{\delta}_{db}$	DE gain	α^*	$(\varphi_s^*, \varphi_a^*, \varphi_0^*)$	$\tilde{\delta}_{db}$	DE gain
1/2	3 bits	0.65	2	0.000	0.000	0.000	**1.3997**	**0.4313**	0.65	(0,0,0)	**1.4003**	**0.4307**
			3	0.000	0.162	0.000				(0,0,0)		
			≥ 4	1.000	1.000	1.000				(1,1,1)		
	4 bits	1.24	2	0.000	0.000	0.000	**0.9547**	**0.4394**	1.24	(0,0,0)	**0.9582**	**0.4359**
			3	0.250	1.000	0.375				(0,1,0)		
			≥ 4	1.000	1.000	1.000				(1,1,1)		
3/4	3 bits	0.81	2	0.000	0.000	0.000	**2.4433**	**0.3803**	0.81	(0,0,0)	**2.4451**	**0.3785**
			3	1.000	0.687	0.000				(1,1,0)		
			≥ 4	1.000	1.000	1.000				(1,1,1)		
	4 bits	1.48	2	1.000	0.788	0.000	**2.2110**	**0.0306**	1.49	(1,1,0)	**2.2111**	**0.0305**
			3	1.000	1.000	1.000				(1,1,1)		
			≥ 4	1.000	1.000	1.000				(1,1,1)		

Another conclusion can be derived from these tables for SP-MS decoders. From the DE analysis we can conclude that the offset $\lambda_v = 1$ should not be applied on degree $d_v = 2$ VNs for the case of low precision $q = 3$, since we obtain always $(\varphi_s^{(2)}, \varphi_a^{(2)}, \varphi_0^{(2)}) = (0, 0, 0)$. While for the largest precision $q = 4$, the offset $\lambda_v = 1$ should be applied on degree $d_v = 2$ VNs for some cases. These observations, combined with the fact that the optimum values for $\boldsymbol{\varphi}^{(\geq 4)*}$ are always 1 which correspond to the offset $\lambda_v = 1$, lead to the conclusion that the offset $\lambda_v = 1$ in SP-MS decoders must be chosen carefully for irregular LDPC codes, especially for the VN degree $d_v = 2$ and $d_v = 3$.

Finally, the gains of the SP-MS decoders are greater when using irregular codes. The gain of the rate 1/2 code is 0.4313 dB for the lower precision $q = 3$, and 0.4394 dB for the largest precision $q = 4$. In the case of the rate 3/4 code, the gains for the two considered precision $q = 3$ and $q = 4$ are smaller than for the rate 1/2 code. Obtaining gains of 0.3803 dB and 0.0306 dB for $q = 3$ and $q = 4$, respectively.

4.9 Finite Length Performance of Sign-Preserving Min-Sum Decoders

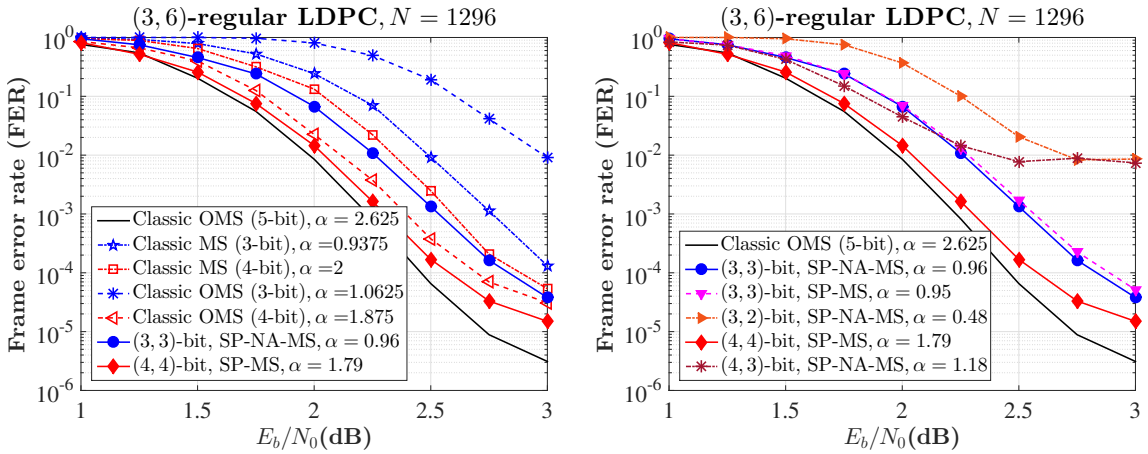
In this section we present the frame error rate (FER) performance for noiseless classical MS, noiseless classical OMS, SP-NA-MS, and SP-MS decoders. We analyze the performance of these quantized decoders over the BI-AWGN channel in order to corroborate the asymptotic results obtained in the previous section.

For this purpose, the PEG algorithm from [52] is used to design the regular QC-LDPC codes. We designed: *(i)* two (d_v, d_c)-regular QC-LDPC codes with length $N = 1280$, $R \in \{1/2, 3/4\}$ for $d_v = 5$, and $L = 128$ for $R = 1/2$ and $L = 64$ for $R = 3/4$, and *(ii)* one (d_v, d_c)-regular QC-LDPC codes with length $N = 10240$, $R = 3/4$ for $d_v = 5$, and $L = 512$. In addition, the four (d_v, d_c)-regular QC-LDPC codes presented in Chapter 3 are used.

The considered decoders are the ones with the best DE thresholds, indicated in bold in Table 4.2 and Table 4.4. The noiseless classical OMS decoder performance with the largest precision $q = 5$ is also shown as benchmark.

A maximum of 100 iterations has been set for the regular $(d_v = 3, d_c = 6)$ QC-LDPC code. Fig. 4.5 shows the FER performance comparisons between the classical MS, classical OMS, SP-MS, and SP-NA-MS decoders, for three considered precisions of messages $q \in \{2, 3, 4\}$, and for the regular $(d_v = 3, d_c = 6)$ QC-LDPC code. Fig. 4.6 draws the same curves for the regular $(d_v = 4, d_c = 8)$ QC-LDPC code.

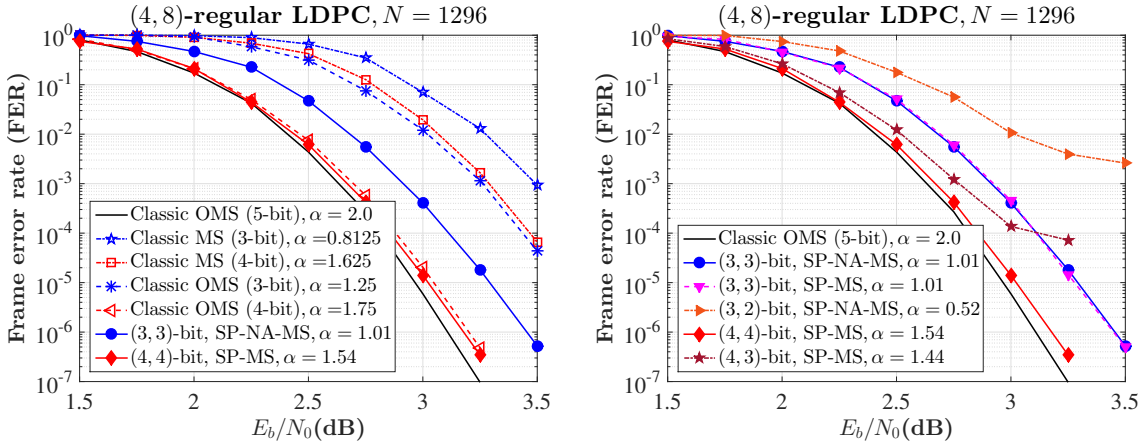
A first conclusion is that the finite length FER performances are in accordance with the gains predicted by the DE analysis. We observe in the waterfall region (*i.e.* at FER $= 10^{-2}$) an SNR gain for the SP-MS decoders which corresponds to the threshold differences (Table 4.4): around 0.27 dB for $(q_{ch} = q = 3,\ d_v = 3,\ R = 1/2)$ and 0.06 dB



(a) Precision $q = q_{ch}$ (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$

Fig. 4.5 FER performance for $(3,6)$-regular QC-LDPC code.

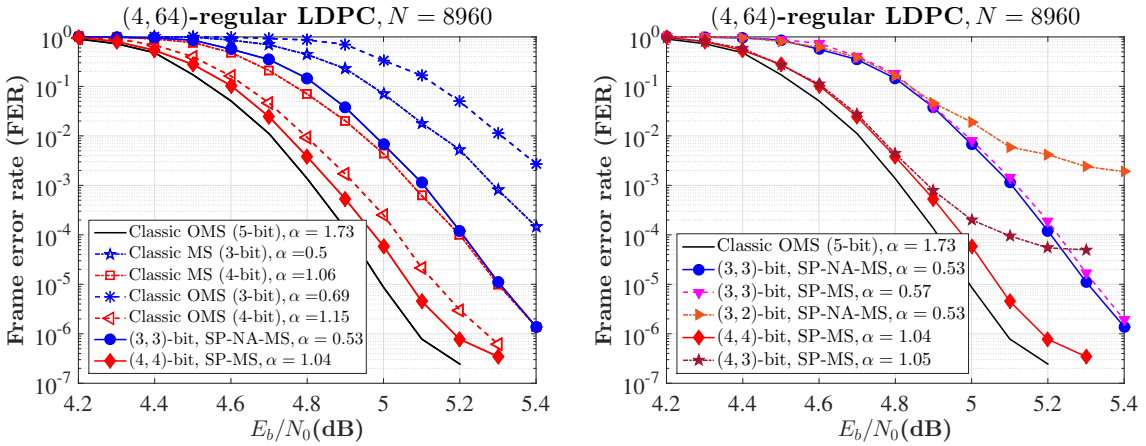
for $(q_{ch} = q = 4,\ d_v = 3,\ R = 1/2)$, 0.32 dB for $(q_{ch} = q = 3,\ d_v = 4,\ R = 1/2)$, and the same performance for $(q_{ch} = q = 4,\ d_v = 4,\ R = 1/2)$.



(a) Precision $q = q_{ch}$ (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$

Fig. 4.6 FER performance for $(4,64)$-regular QC-LDPC code.

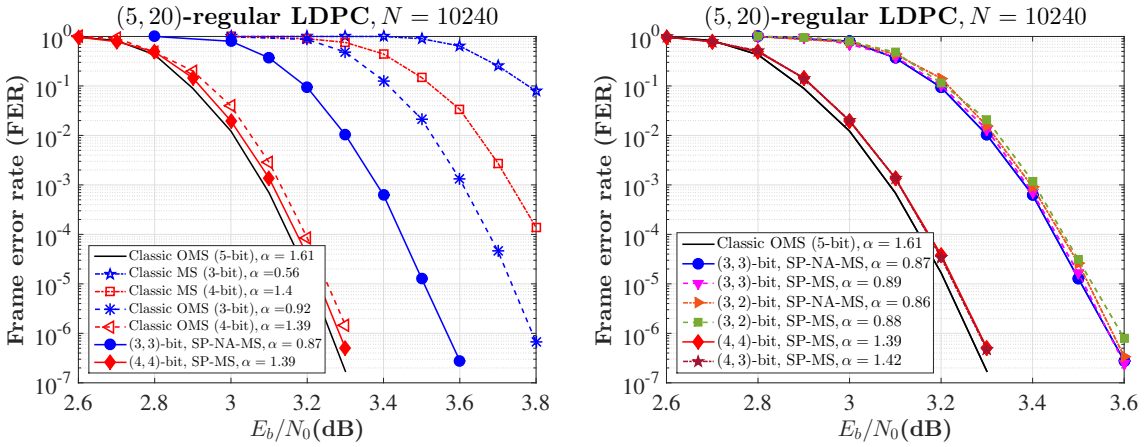
Simulation results for the regular $(d_v = 4, d_c = 64)$ QC-LDPC code of [9] are provided on Fig. 4.7 with a maximum of 30 iterations. Again, the SNR gains in the waterfall correspond to what was predicted with the DE analysis, with a 0.16 dB gain for $q_{ch} = q = 3$ and 0.04 dB for $q_{ch} = q = 4$.



(a) Precision $q = q_{ch}$ (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$

Fig. 4.7 FER performance for $(4, 64)$-regular QC-LDPC code.

On the other hand, from Fig. 4.5b, it is evident that the SP-MS decoders that use the precision $q = q_{ch} - 1$ exhibit very poor performance having an early error floor. In the case of $d_v = 4$ QC-LDPC decoders, see Fig. 4.7b, we can confirm again that the SNR gains in waterfall correspond to the DE gains for the precision message $q = q_{ch} - 1$, but we can observe that the $d_v = 4$ QC-LDPC decoders exhibit an early error floor.



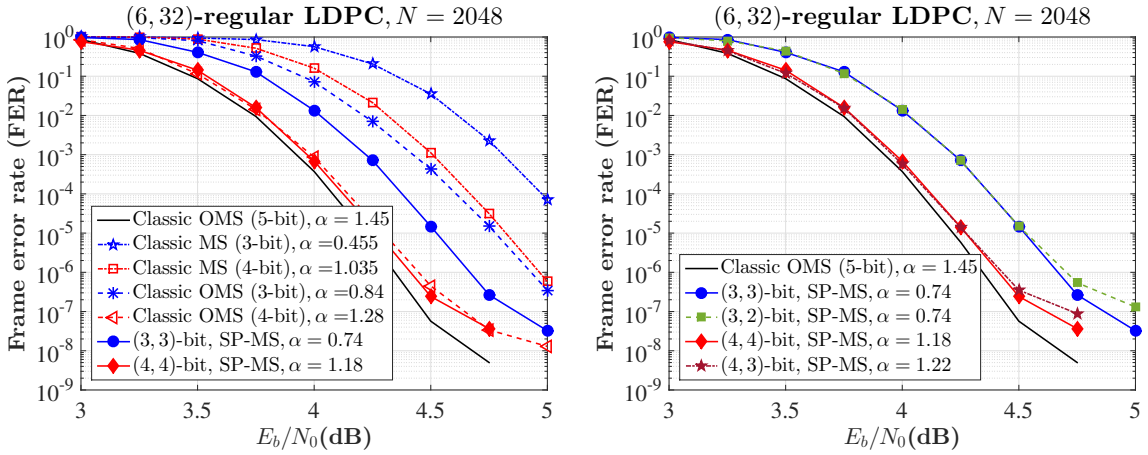
(a) Precision $q = q_{ch}$ (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$

Fig. 4.8 FER performance for $(5, 20)$-regular QC-LDPC code.

More simulation results for the case of the $d_v = 5$ QC-LDPC decoders are shown in Fig. 4.8. Once again it can be confirmed that gains predicted by DE correspond to the

SNR gains, 0.20 dB for ($q_{ch} = q = 3$, $d_v = 5$, $R = 1/2$), and 0.02 dB for ($q = 4$, $d_v = 5$, $R = 1/2$). Similarly, when the precision of messages equals $q = q_{ch} - 1$, the SNR gains correspond to the DE gains with a 0.20 dB gain for ($q_{ch} = 3$, $q = 2$) and a 0.02 dB gain for ($q_{ch} = 4$, $q = 3$).

Simulation results for the IEEE 802.3 ETHERNET code are provided on Fig. 4.9 with a maximum of 30 iterations. Again, the SNR gains in the waterfall correspond to what was predicted with the DE analysis, with a 0.19 dB gain for ($q_{ch} = 3, q = 3$) and ($q_{ch} = 3, q = 2$), and the same performance is achieved for ($q_{ch} = 4, q = 4$) and ($q_{ch} = 4, q = 3$).



(a) Precision $q = q_{ch}$ (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$

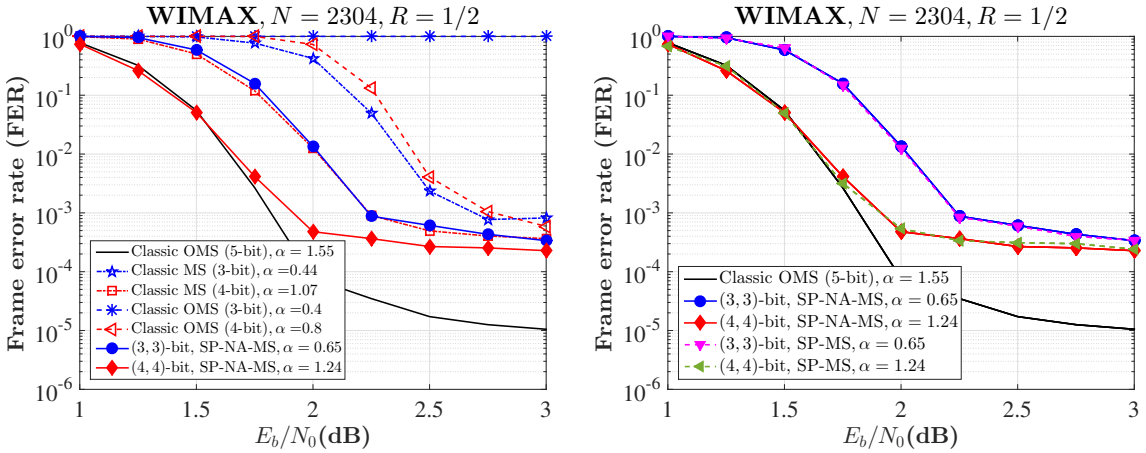
Fig. 4.9 FER performance for the ETHERNET code.

From Fig. 4.8b for $d_v = 5$ QC-LDPC decoders and Fig. 4.9b for $d_v = 6$ Reed-Solomon LDPC (RS-LDPC) decoders, we can confirm and conclude that the ($q_{ch}, q = q_{ch} - 1$)-bit SP-MS decoders have the same performance as the ($q_{ch}, q = q_{ch}$)-bit SP-MS decoders, except for the $d_v = 5$ QC-LDPC decoders and precision ($q_{ch} = 3, q = 2$), in that case a negligible loss of performance is observed. In the implementation part this has a great impact, since it goes from the precision message $q = q_{ch}$ to the precision $q = q_{ch} - 1$, this implies a considerable reduction in the complexity of the CNs and VNs, as well as a reduction in the number of wires in an ASIC implementation.

We have made the same analysis for the other LDPC codes designed with rate $R = \{1/2, 3/4\}$, *i.e.* $(d_v = 4, d_c = 8, N = 1296)$, $(d_v = 5, d_c = 10, N = 1280)$, $(d_v = 3, d_c = 12, N = 1296)$, $(d_v = 4, d_c = 16, N = 1296)$, and $(d_v = 5, d_c = 20, N = 1280)$, and obtained the same conclusions.

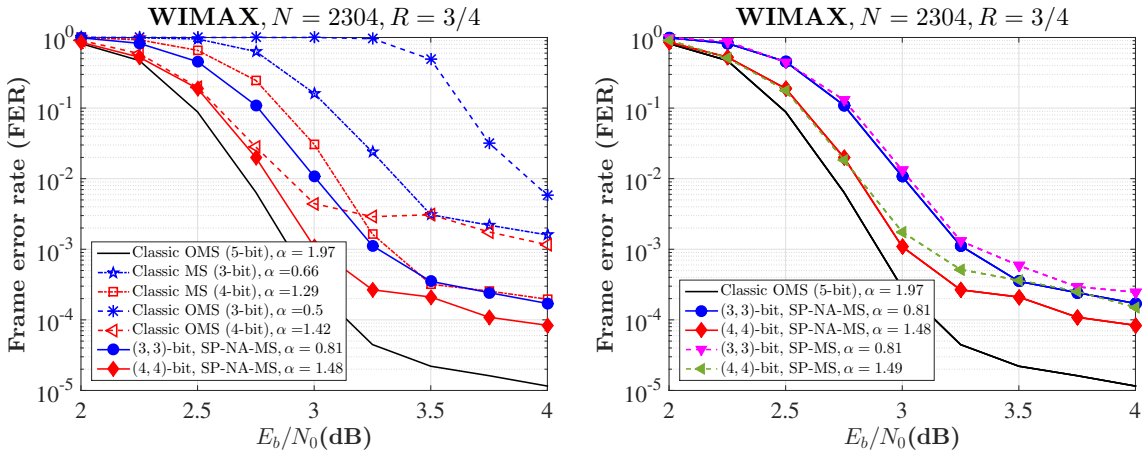
Similarly, Fig. 4.10 and Fig. 4.11 present simulation results for the WIMAX rate 1/2 LDPC code and the WIMAX rate 3/4 B LDPC code, respectively, for a maximum of 100 iterations. We observe again that the SNR gains in the waterfall region are in agreement with the gains predicted by the DE analysis, with a 0.40 dB gain for $(q_{ch} = q = 3, R = 1/2)$, a 0.40 dB gain for $(q_{ch} = q = 4, R = 1/2)$, a 0.37 dB gain for $(q_{ch} = q = 3, R = 3/4)$, and a 0.03 dB gain for $(q_{ch} = q = 3, R = 3/4)$.

Additionally, for the WIMAX rate 1/2 LDPC code, the $(q_{ch} = 3, q = 3)$-bit SP-MS decoder has the same FER performance as the 4-bit MS decoder. In the waterfall region, the $(q_{ch} = 4, q = 4)$-bit SP-MS decoder has the same FER performance as the 5-bit OMS decoder, while in the error floor region, the 5-bit OMS decoder has better FER performance than the $(q_{ch} = 4, q = 4)$-bit SP-MS decoder. In the case of the WIMAX rate 3/4 B LDPC code, the SP-MS decoders have better FER performance than the MS and the OMS decoders in the error floor region using the same precision.



(a) FER performance of SP-NA-MS. (b) FER performance of SP-MS.

Fig. 4.10 FER performance for the WIMAX LDPC code with $R = 1/2$.



(a) FER performance of SP-NA-MS. (b) FER performance of SP-MS.

Fig. 4.11 FER performance for the WIMAX LDPC code with $R = 3/4$.

On the other hand, one can note that for regular and irregular LDPC codes, the SP-MS decoders have a performance very close or even the same as the SP-NA-MS decoders when the same precision is used, this means that the noise injected in SP-NA-MS decoders can be ignored without affecting the performance of the SP-MS decoders which is the same conclusion obtained in the previous section. Therefore, the most important thing about SP-MS decoders is their nature, which is the preservation of the sign of the messages exchanged between VNs and CNs during the decoding process.

Comparing the FER performance of the SP-MS decoders and the NAN-MS decoders, we can clearly see that the SP-MS decoders have a better performance for both regular codes and irregular codes, especially when the precision of the messages is low. We can confirm again that the FER performances are in accordance with the gains predicted by the DE analysis.

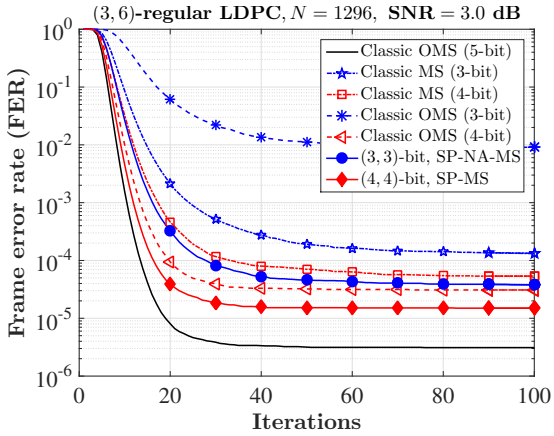
As a last remark, we can also see that the preservation of the sign of messages does not seem to have an influence in the error floor of the decoders, since all the curves have similar slopes in the low FER region. This means that the preservation of the sign of messages does not correct the dominant error events due to trapping sets.

4.10 Convergence Performance Analysis

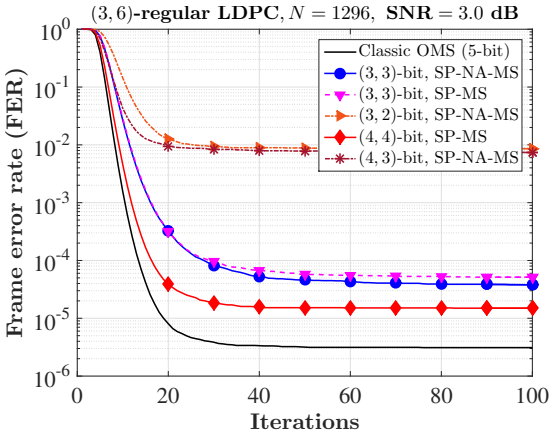
In this section, we present the FER convergence performance for the (d_v, d_c)-regular QC-LDPC codes, for the IEEE 802.3 ETHERNET code [6], and the two WIMAX LDPC codes [7] used in Section 4.9. The FER is evaluated at each decoding iteration for the classical MS, the classical OMS, the SP-NA-MS, and the SP-MS decoders over the BI-AWGN channel.

FER Convergence results are presented on Fig. 4.12 for the regular $(d_v = 3, d_c = 6)$ QC-LDPC code, on Fig. 4.13 and Fig. 4.14 for the $d_v = 4$ QC-LDPC codes, and on Fig. 4.15, Fig. 4.16, and 4.17 for the $d_v = 5$ QC-LDPC codes. Also, Fig. 4.18 shows the FER convergence performance for the IEEE 802.3 ETHERNET code. Similarly, Fig. 4.19 and Fig. 4.20 present FER convergence results for the WIMAX LDPC code. These figures show that the SP-NA-MS and the SP-MS decoders are faster to find the correct solution compared with classical MS and classical OMS decoders for a fixed precision $q = q_{ch}$, a fixed degree distribution $(\lambda(x), \rho(x))$, and a fixed E_b/N_0. It can be seen that the FER curves of the SP-MS decoders decrease faster than the MS and the OMS decoders. In addition, for low precision $(q_{ch}, q = 3)$, the SP-MS decoders perform much better than MS and OMS decoders along with the number of iterations increasing. It can also be observed that the SP-MS decoders have the same FER convergence performance as the SP-NA-MS decoders for a fixed precision $q = q_{ch}$.

One can note that for regular LDPC codes with $d_v > 4$, the FER convergence performance of the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoders are almost equal or equal to the FER convergence performance of $(q_{ch}, q = q_{ch})$-bit SP-MS decoders, see Fig. 4.15b, Fig. 4.16b, Fig. 4.17b, and Fig. 4.18b. In the case of the regular LDPC codes with $d_v = 3$ and $d_v = 4$, the SP-MS decoders implemented with precision $q = q_{ch} - 1$ exhibit

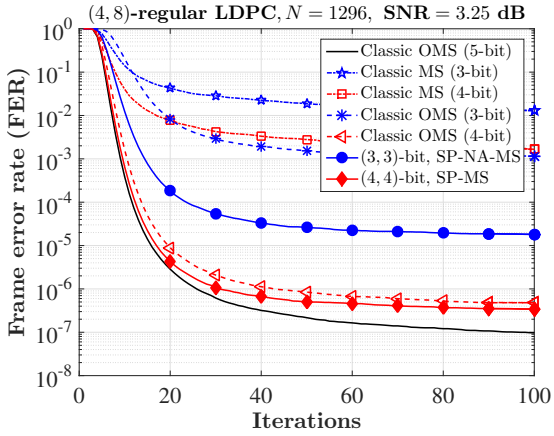


(a) Precision $q = q_{ch}$.

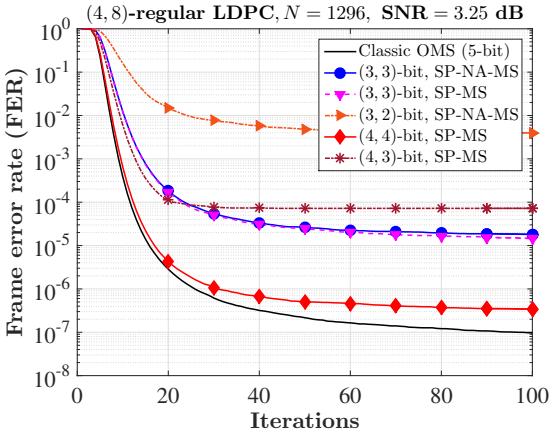


(b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$.

Fig. 4.12 FER convergence comparison on $(3,6)$-regular QC-LDPC code at $E_b/N_0 = 3.0$ dB.

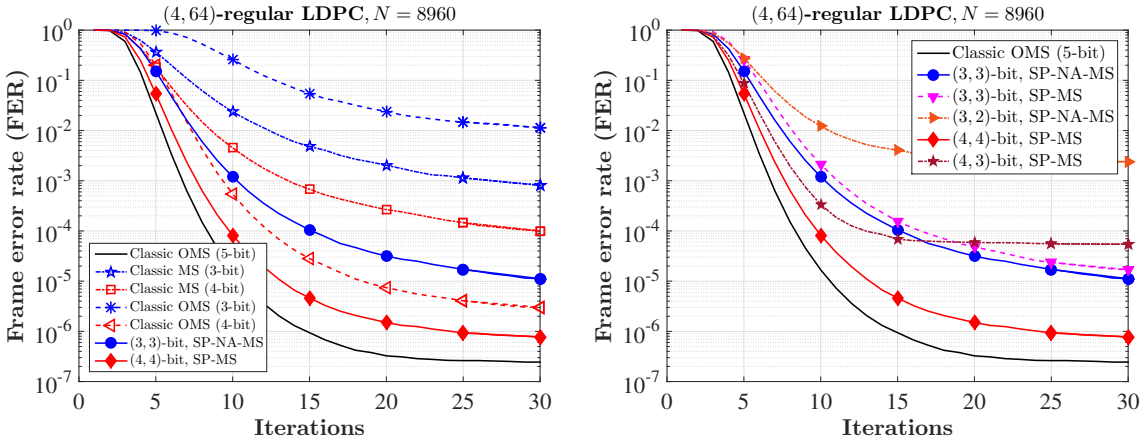


(a) Precision $q = q_{ch}$.



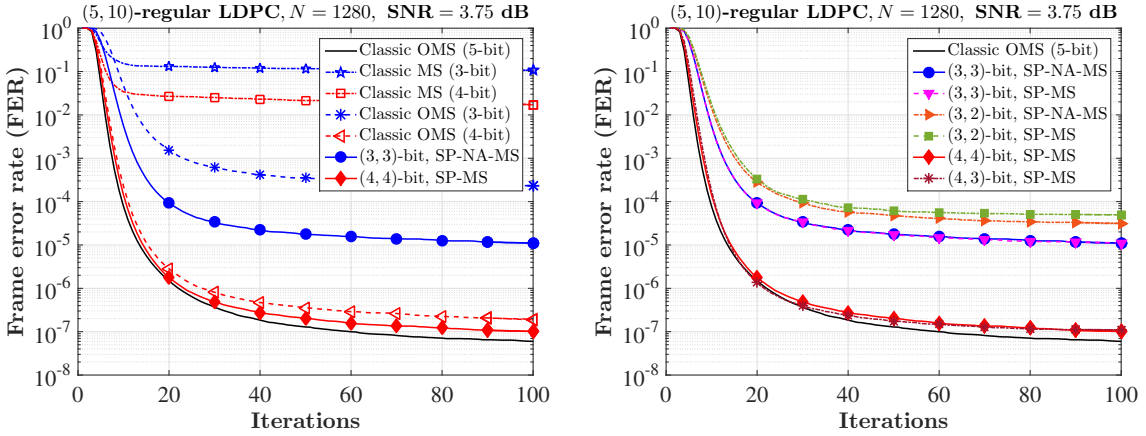
(b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$.

Fig. 4.13 FER convergence comparison on $(4,8)$-regular QC-LDPC code at $E_b/N_0 = 3.25$ dB.



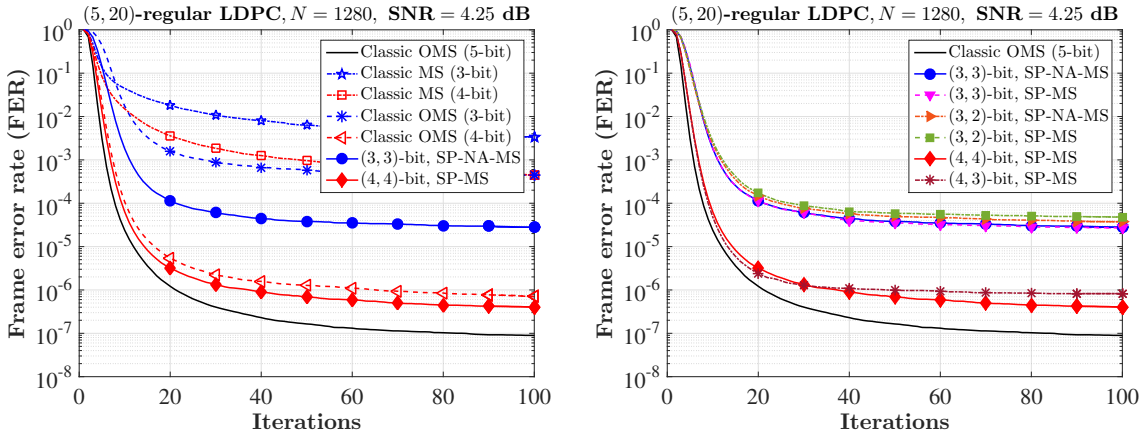
(a) Precision $q = q_{ch}$. (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$.

Fig. 4.14 FER convergence comparison on $(4, 64)$-regular QC-LDPC code at $E_b/N_0 = 5.2$ dB for $q_{ch} = \{4, 5\}$ and at $E_b/N_0 = 5.3$ dB for $q_{ch} = 3$.



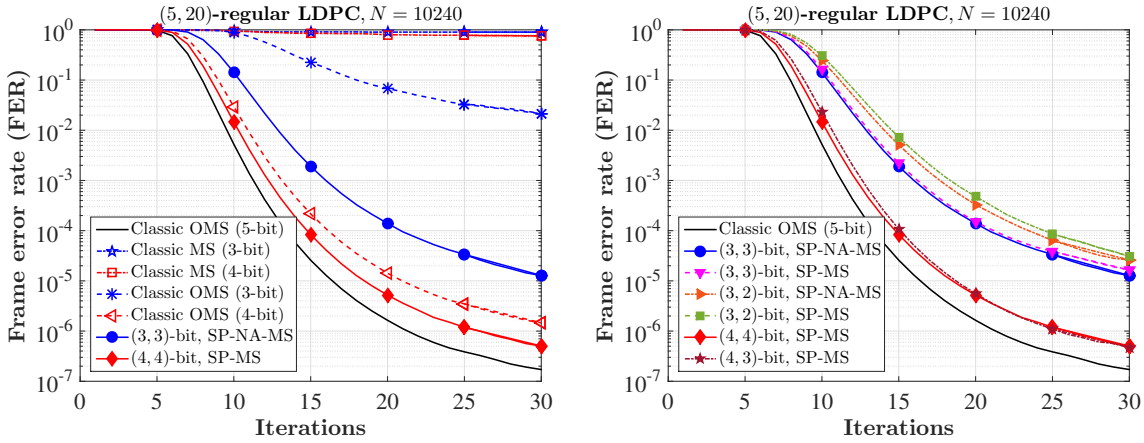
(a) Precision $q = q_{ch}$. (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$.

Fig. 4.15 FER convergence comparison on $(5, 10)$-regular QC-LDPC code at $E_b/N_0 = 3.75$ dB.



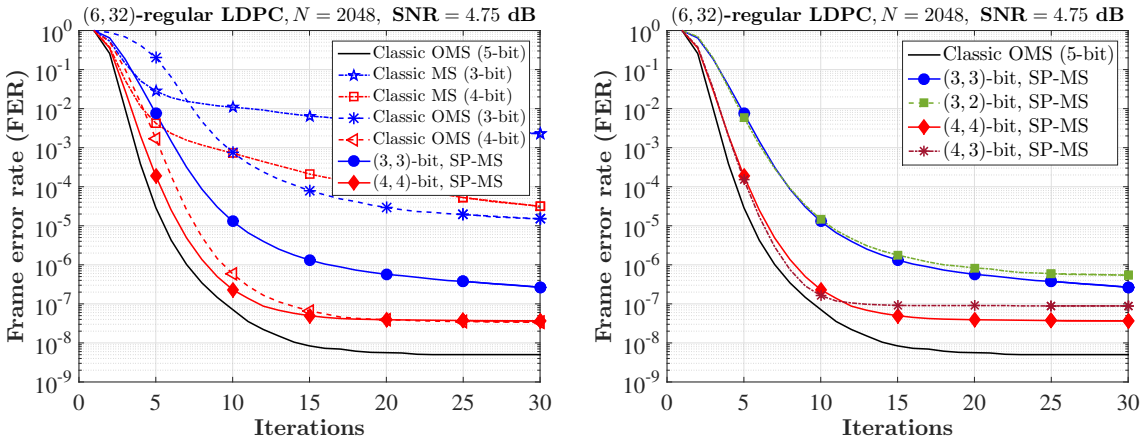
(a) Precision $q = q_{ch}$. (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$.

Fig. 4.16 FER convergence comparison on $(5, 20)$-regular QC-LDPC code at $E_b/N_0 = 4.25$ dB.



(a) Precision $q = q_{ch}$. (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$.

Fig. 4.17 FER convergence comparison on $(5, 20)$-regular QC-LDPC code at $E_b/N_0 = 3.3$ dB for $q_{ch} = \{4, 5\}$ and at $E_b/N_0 = 3.5$ dB for $q_{ch} = 3$.



(a) Precision $q = q_{ch}$. (b) Precision $q = q_{ch}$ and $q = q_{ch} - 1$.

Fig. 4.18 FER convergence comparison on the IEEE 802.3 ETHERNET code at $E_b/N_0 = 4.75$ dB.

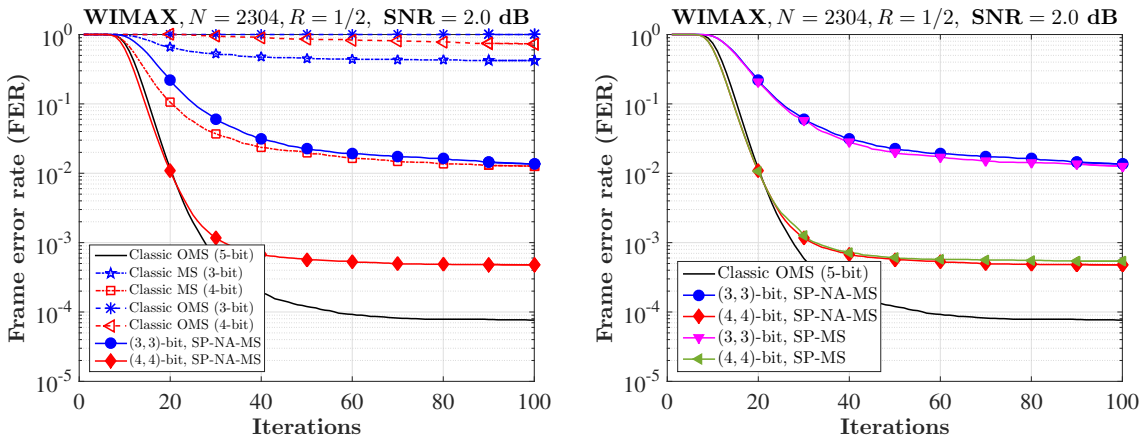
worse FER convergence performance than the SP-MS decoders using $q = q_{ch}$ due to the early appearance of the error floor.

On the other hand, for all decoders tested using QC-LDPC codes with length $N \in \{1296, 1280\}$, one can set the maximum number of iterations to 20 because after 20 iterations the slopes of the curves change drastically. For the ETHERNET code, a maximum of 10 iterations can be used for the SP-MS and decoders and precision $q \in \{3, 4\}$, while for the SP-MS decoders using precision $q \in \{2, 3\}$, a maximum of 15 iterations can be used. In the case of the WIMAX code, 30 and 20 iterations as maximum can be set for $R = 1/2$ and $R = 3/4$, respectively. From this analysis we can say that the maximum number of iterations allowed in a SP-MS decoder will depend on the length of the LDPC code N, the degree distribution $(\lambda(x), \rho(x))$, and the precision used (q_{ch}, q).

Let us analyse further the FER convergence performance for the case of the IEEE 802.3 ETHERNET code. During the first 10 decoding iterations, the $(4, 4)$-bit SP-MS and the $(4, 3)$-bit SP-MS decoders have almost the same FER performance as the 5-bit OMS decoders. We can observe that the SP-MS, MS, and OMS decoders

decrease slowly and in some cases they do not decrease after 15 iterations. Also, at $E_b/N_0 = 4.75$ dB (Fig. 4.18), the $(3,3)$-bit SP-MS and the $(3,2)$-bit SP-MS decoders with 15 iterations reach the FER$= 10^{-6}$. Similarly, the 4-bit OMS decoder with 9 iterations, the $(4,4)$-bit SP-MS and the $(4,3)$-bit SP-MS decoders with 8 iterations, and the 5-bit OMS decoder with 7 iterations reach the FER$= 10^{-6}$. We can also see that during the first 10 decoding iterations, the FER curve of the $(4,3)$-bit SP-MS decoder is almost an iteration away from the FER curve of the 5-bit OMS decoder.

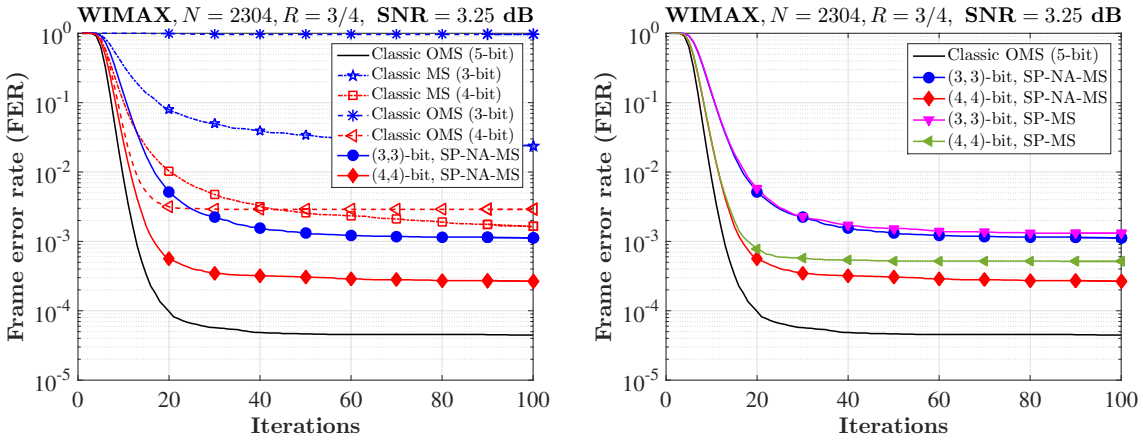
In the case of the WIMAX rate 1/2 LDPC code at $E_b/N_0 = 2.0$ dB (Fig. 4.19), we can see that $(4,4)$-bit SP-MS decoder has the same FER performance as the 5-bit OMS decoder throughout the first 30 iterations. It can also be seen that the $(3,3)$-bit SP-MS can achieve the same FER performance as the 4-bit MS decoder using 5 iterations more on average. For the WIMAX rate 3/4 B LDPC code at $E_b/N_0 = 3.25$ dB (Fig. 4.20), we observe that the SP-MS decoders have better FER performance than the MS and the OMS decoder throughout the iterations using the same precision (q_{ch}, q).



(a) SP-NA-MS decoders. (b) SP-MS decoders.

Fig. 4.19 FER convergence comparison on the WIMAX rate 1/2 LDPC code at $E_b/N_0 = 2.0$ dB.

From all the results presented, we can conclude that *(i)* the SP-MS and the SP-NA-MS decoders have almost the same convergence speed, *(ii)* the SP-MS decoders



(a) SP-NA-MS decoders.

(b) SP-MS decoders.

Fig. 4.20 FER convergence comparison on the WIMAX rate 3/4 B LDPC code at $E_b/N_0 = 3.25$ dB.

converge faster than the MS and the OMS decoders when the same precision is used, *(iii)* the SP-MS decoder uses fewer iterations than the the MS and the OMS to reach at a fixed value of FER.

The SP-NA-MS and the SP-MS decoders are the best decoders because these decoders use the sign information of the messages. Specifically at the VNs, the sign of the c-to-v messages is used to increase the reliability of the v-to-c messages and the APP, this causes the decoder to converge faster, *i.e.* a smaller number of iterations is used. We can note that if the reliability of the APPs increases then the reliability of the estimated bits also increases, *i.e.* the estimated codeword is more reliable.

4.11 Conclusion

In this chapter we have proposed a new message passing iterative LDPC decoder defining a sign-preserving factor which helps the decoder to keep the sign information of extrinsic messages during the VNU processing. The sign-preserving factor helps to improve the error-correcting performance of finite precision iterative decoders. We have also proposed a noise model to introduce randomness in the optimization process of

the proposed decoder. Density Evolution was used to optimize our SP-MS decoder performance. The analysis conducted with DE has shown that the preservation of sign of messages is always beneficial for low precision SP-MS decoders. Further DE analysis has shown that the noise injected in the SP-NA-MS decoders can be ignored without degrading their error-correcting performance, specially when the optimized transition probabilities are very close or equal to 0 or 1. Also, the DE thresholds results have shown that the precision of messages can be reduced by one bit maintaining the same error-correcting performance. The finite-length Monte Carlo simulations have corroborated the DE analysis and DE thresholds. We have also shown that the SP-MS decoders converge faster and use fewer iterations than the MS and OMS decoders.

From our study, an efficient implementation of SP-MS decoders can be made in a full-parallel architecture because the reduction of one bit in the precision of messages implies a considerable reduction in the complexity of the CNU and the VNU, and a great reduction in the total number of wires in the architecture. The cost that is paid for reducing the precision of the messages is the early appearance of the error floor, specially for the regular $d_v = 3$ and $d_v = 4$ LDPC codes.

The next chapter discusses the implementation of the (q_{ch}, q)-bit SP-MS decoders for regular LDPC codes, for the IEEE 802.3 ETHERNET code, and for the WIMAX LDPC code with rate $R = 1/2$.

Chapter 5

Implementation of Sign-Preserving Min-Sum Decoders

This Chapter presents the hardware implementation results of SP-MS decoders, first several LDPC decoders are studied, then the case of the IEEE 802.3 ETHERNET code is studied in detail. In particular, we presents two methods (one of them new) to efficiently mitigate the error floor at high levels of SNR. The resulting design of a 28 nm place and route ASIC shows very good results compared to state of the art implementations.

The outline of this chapter is as follows. The first section of this chapter presents rapidly the state of the art of highly parallel decoders. In the second section, we propose a fully parallel architecture to implement the MS, the OMS, and the SP-MS decoders for regular and irregular LDPC codes. The third section reports the the implementation results of the various regular and irregular LDPC decoders. The fourth section briefly discusses the trapping-sets for the IEEE 802.3 ETHERNET code. In the fifth section, we propose a post-processing architecture using the same precision for messages and LLRs. We also propose an architecture that implements the SP-MS decoder for the IEEE 802.3 ETHERNET code. In the sixth section, we introduce a new post-processing

algorithm for low precision decoders. Also, we present a fully parallel architecture that implements the proposed algorithm. The architecture is conceived for SP-MS decoders that use different precision for messages and LLRs. The seventh section reports the implementation results of the two proposed post-processing architectures. Also, the place and route results for two SP-MS decoders are listed, and the eighth section concludes this chapter.

5.1 Overview of Existing Hardware Implementations

In the past twenty years, the implementation of the OMS-based decoders in FPGAs and in CMOS technologies (ASIC implementations) has been studied and reported [53–58]. In the literature, we can find works that implement quantized decoders using a fully parallel architecture [55, 56, 58, 57]. A parallel architecture is desirable for high-throughput applications, but it does not efficiently use the implementation area and presents routing congestion problems. In [59], the authors have show that the area of a parallel architecture is determined by routing congestion and not by the gate count. The routing congestion problem of a parallel LDPC decoder can be reduced using a layer architecture [23, 60, 61]. In [62] a bit serial architecture is proposed to alleviate the routing problem, but bit serial architecture greatly reduces the decoding throughput. In [54], the authors use a grouping strategy of wires to reduce the routing congestion, the strategy consists of dividing the wires into global wires and local wires obtaining a partial-parallel architecture. Another strategy to reduce the routing congestion is to reduce the precision of messages, this strategy is used in [23] using the nonsurjective finite alphabet iterative decoder (NS-FAID), and in [63] using the FAID and an unrolled full-parallel architecture [64, 65] . The stochastic-based decoders

[66, 58] and the noisy gradient descent bit flipping decoder [67] use 1 bit of precision for the messages helping to solve the problem of routing congestion.

Although it is true that quantized decoders have good error correction performance at low SNR levels (waterfall region), at high SNR levels the slope of the FER curve often changes drastically degrading the error correction capability (error floor region). This phenomenon called floor error is usually caused by small trapping-sets [68]. In communication standards such as the IEEE 802.3 standard (10GBASE-T ETHERNET) that requires an error-free operation below the FER level of 10^{-10}, the early appearance of the error floor degrades the service. Many algorithms have been proposed in the literature in order to lower the error floor [54, 69–73]. The IEEE 802.3 ETHERNET code is one of the most studied regular LPDC codes to perform an implementation, several architectures and decoders have been proposed [63, 54–58, 74, 67]. The WIMAX LDPC code with rate $R = 1/2$ and length $N = 2304$ has also received important attention to carry out its implementation [60, 23, 61, 75–77].

In this work, in order to achieve high decoding speeds, we propose a fully parallel architecture to implement the MS, the OMS, and the SP-MS decoders for regular and irregular LDPC codes. We investigate whether the SP-MS decoder uses an area equivalent to or greater than the area used by the MS and the OMS decoder. We also investigate in what percentage the area of the SP-MS decreases when the precision of the messages goes from $q = q_{ch}$ bits to $q = q_{ch} - 1$ bits. On the other hand, the error floor of the IEEE 802.3 ETHERNET code is studied because the SP-MS decoder exhibit an error floor at a FER level of 10^{-8} making the SP-MS unacceptable.

In order to lower the error floor of SP-MS decoders, this work proposes a new post-processing algorithm based on the algorithm of [69, 54]. In addition, this work proposes two fully parallel post-processing architectures: *(i)* the first architecture adapts the post-processing algorithm of [69, 54] and uses the same precision for messages and

for LLRs, and *(ii)* the second architecture incorporates the proposed post-processing algorithm and uses different precision for messages and for LLRs.

Almost all the results of the area used and the power consumed by the decoders are obtained by making only the synthesis (FPGA and ASIC) of the decoders. At the moment that the place and route results are presented, it will be clearly stated.

5.2 Hardware Architecture of Sign-Preserving Min-Sum Decoders

In this section, we propose a fully parallel architecture for Sign-Preserving Min-Sum decoders that can be easily adaptable to implement the MS or the OMS decoders. The proposed architecture performs a decoding iteration per clock cycle in order to achieve very high throughput and ensure efficient use of hardware resources.

The input and output signals of the architecture are shown in Fig. 5.1. We can observe that the input signals are *clk* which is the global clock, *rst* which is a signal that puts the architecture in initial conditions, *go* which is a signal that allows the decoding process, and *DecInput* (of $q_{ch} \times N$ bits) which is the quantized version of the LLRs computed by $I_n = \mathcal{Q}^* \left(LLR(y_n) \right) \in \mathcal{A}_S$ for $n = 1, \ldots, N$. We can also see that the output signals are *Codeword* which is an estimated codeword $\hat{\boldsymbol{x}} = (\hat{x}_1, ..., \hat{x}_N) \in \{0,1\}^N$, *EndIter* indicates the end of the current decoding process, it can be set to 1 if a valid codeword is obtained or if the maximum number of iteration is reached, and *EndSyndr* which is a signal equal to 0 if and only if a valid codeword is obtained. At the beginning of the architecture's operation, the reset signal is set to $rst = 1$ in order to initialize all signals. To start the decoding process, the signal *go* is set to 1 while reset equals $rst = 0$.

Our architecture is composed of four main components: **VNU**, **CNU**, **Count of Iterations**, and **Compute Syndrome**. For a fixed degree distribution $(\lambda(x), \rho(x))$, it is sufficient to change the component **VNU** to implement the MS or the OMS for regular or irregular LDPC codes. It should be noted that for a regular code $(\rho(x) = x^{d_c-1})$, only one type of CNU is implemented, while in the case of irregular codes $(\rho(x) = \sum_{j=2}^{d_{c,max}} \rho_j x^{j-1})$, various types of CNUs are implemented. It is worth mentioning that a C code has been written to generate the architecture from the description of a sparse parity-check matrix H. Those components are shown in Fig. 5.1 and they are discussed as follows.

5.2.1 Count of Iterations

This component is responsible for counting the decoding iterations. From Fig. 5.1, we can see that the input signals are *clk, rst*, and *initialize*, and the output signal is *maxIter*. The count of the iterations is controlled by the signal *initialize*. When $initialize = 1$, the v-to-c messages are initialized with the quantized LLRs I_n using $\mathcal{S}(I_n, N_q)$, therefore, no iteration is counted and $maxIter = 0$. In the case that $initialize = 0$, the architecture performs the decoding process and one iteration is counted per each clock cycle, it should be noted that $maxIter = 1$ if and only if the maximum number of iteration is reached, otherwise $maxIter = 0$.

5.2.2 Compute Syndrome

This component is an asynchronous circuit and computes the syndrome at each decoding iteration. Its input signals are *rst, go, maxIter*, and *decision* which has N information bits, and its output signals are *EndSyndr, EndIter*, and *codeword* which is an estimation of the transmitted codeword over a noisy channel.

When the decoding process is enabled, *i.e.* $rst = 0$ and $go = 1$, the syndrome vector is computed in each decoding iteration using *decision* $\hat{\boldsymbol{x}}$. A correct codeword is obtained when the syndrome vector is all-zero, *i.e.* $H\hat{\boldsymbol{x}}^T = 0$, in this conditions we have $EndIter = 1$ and $EndSyndr = 0$. If a valid codeword is not computed, the value of *EndSyndr* is equal to 1. When the decoder reach the maximum number of iterations, *i.e.* $maxIter = 1$, we set $EndIter = 1$, and *decision* can be a valid codeword ($EndSyndr = 0$) or not ($EndSyndr = 1$).

5.2.3 VNU Component

The proposed architecture uses N VNU components. A VNU v_n component is a synchronous circuit that has as input signals to *clk, rst, initialize*, c-to-v messages $m_{c\to v_n}$, $c \in \mathcal{V}(v_n)$, of q bits, and the quantized LLR I_n of q_{ch} bits, while the output signals are v-to-c messages $m_{v_n\to c}$ of q bits and the estimated bit $\hat{x}_n$.

In order to implement the equations (4.6) and (4.5) in hardware we define the mapping functions $\mathcal{T}$, $\mathcal{T}_{ch}$, and $\mathcal{T}^{-1}$. The c-to-v messages are mapped with $\mathcal{T}(a) = \text{sign}(a) \times (2 \times |a| + 1)$, while the quantized LLR is mapped using $\mathcal{T}_{ch}(a) = \text{sign}(a) \times (2 \times |a| + \xi)$. A v-to-c message is obtained using the function $\mathcal{T}^{-1}(a) = (\text{sign}(a), (|a| - 1)/2)$. These functions are shown in Fig. 5.1, as well as the function Υ^*. The function Υ^* is defined as $\Upsilon^*(a) = (\text{sign}(a), \mathcal{S}(\max(|a| - \lambda_v^*, 0), 2N_q + 1))$, where $\lambda_v^* \in \{0, 2\}$ is the offset value. From Table 4.4 and Table 4.5, $\lambda_v^* = 0$ if the transition probability (φ_s^*, φ_a^*, or φ_0^*) of SP-MS decoders is 0, in the case that the transition probability is 1 we have $\lambda_v^* = 2$.

In the decoding process, *i.e.* $initialize = 0$, at each new clock cycle, the APP value γ_n^* is computed by $\gamma_n^* = \mathcal{T}_{ch}(I_n) + \mathcal{T}(m_{c_1\to v_n}) + ... + \mathcal{T}\left(m_{c_{d_v}\to v_n}\right)$. Then, each v-to-c message is computed by subtracting the corresponding c-to-v message from the APP value γ_n^* and by applying the functions Υ^* and $\mathcal{T}^{-1}$ to the result of the subtraction, *i.e.* $m_{v_n\to c_i} = \mathcal{T}^{-1}(\Upsilon^*(\gamma_n^* - \mathcal{T}(m_{c_i\to v_n})))$ for $i = 1, ..., d_v$.

The estimated bit $\hat{x}_n$ is obtained from the sign of γ_n^* and the sign of the quantized LLR I_n as follows: if $\gamma_n^* = 0$ we have $\hat{x}_n = \text{sign}(I_n)$, otherwise $\hat{x}_n = \text{sign}(\gamma_n^*)$.

5.2.4 Check Node Component

The proposed architecture uses M CNU components. Each CNU c_m is an asynchronous circuit that receives v-to-c messages $m_{v \to c_m}$ from the neighboring VNs $v \in \mathcal{V}(c_m)$. Note that each received message is represented by q bits. Among all received messages, the CNU c_m computes the smallest element min_1 and the second smallest element min_2 ($min_2 \geq min_1$), as well as the product of signs $sign$[1]of these messages. With those values, the function W is defined as $W(a, sign, min_1, min_2) = (\text{sign}(a) \times sign, min_2)$ if $|a| = min_1$, otherwise $W(a, sign, min_1, min_2) = (\text{sign}(a) \times sign, min_1)$. The c-to-v messages $m_{c_m \to v}$, which are the outputs of the CNUs, are computed using $m_{c_m \to v} = W(m_{v \to c_m}, sign, min_1, min_2)$ and sent to the VNUs v, with $v \in \mathcal{V}(c_m)$.

Fig. 5.2 shows a timing diagram of the main signals of the proposed architecture. The decoding process starts only when $go = 1$. When the architecture is enabled to perform the decoding, $go = 1$ and $rst = 0$, only for one clock cycle the signal $initialize$ is 1 and it is used to initialize the registers of the VNUs with the quantized LLRs I_n. The decoding process is performed when $initialize = 0$ and $go = 1$. When a valid codeword is obtained ($EndSyndr = 0$) or the maximum number of iteration is reached ($maxIter = 1$), the signal $EndIter$ is equal to 1 and the decoding process ends and other codeword can be decoded. One should note that at the same time that a codeword is decoded ($EndIter = 1$), *e.g.* $\boldsymbol{I} - b$, another codeword, *e.g.* $\boldsymbol{I} - c$, can initialize the registers of the VNUs (see Fig. 5.2 at clock cycle 12).

[1] In binary representation $sign$ can be obtained performing the xor function between the MSBs of $m_{v \to c_m}^{(\ell)}$, with $v \in \mathcal{V}(c_m)$.

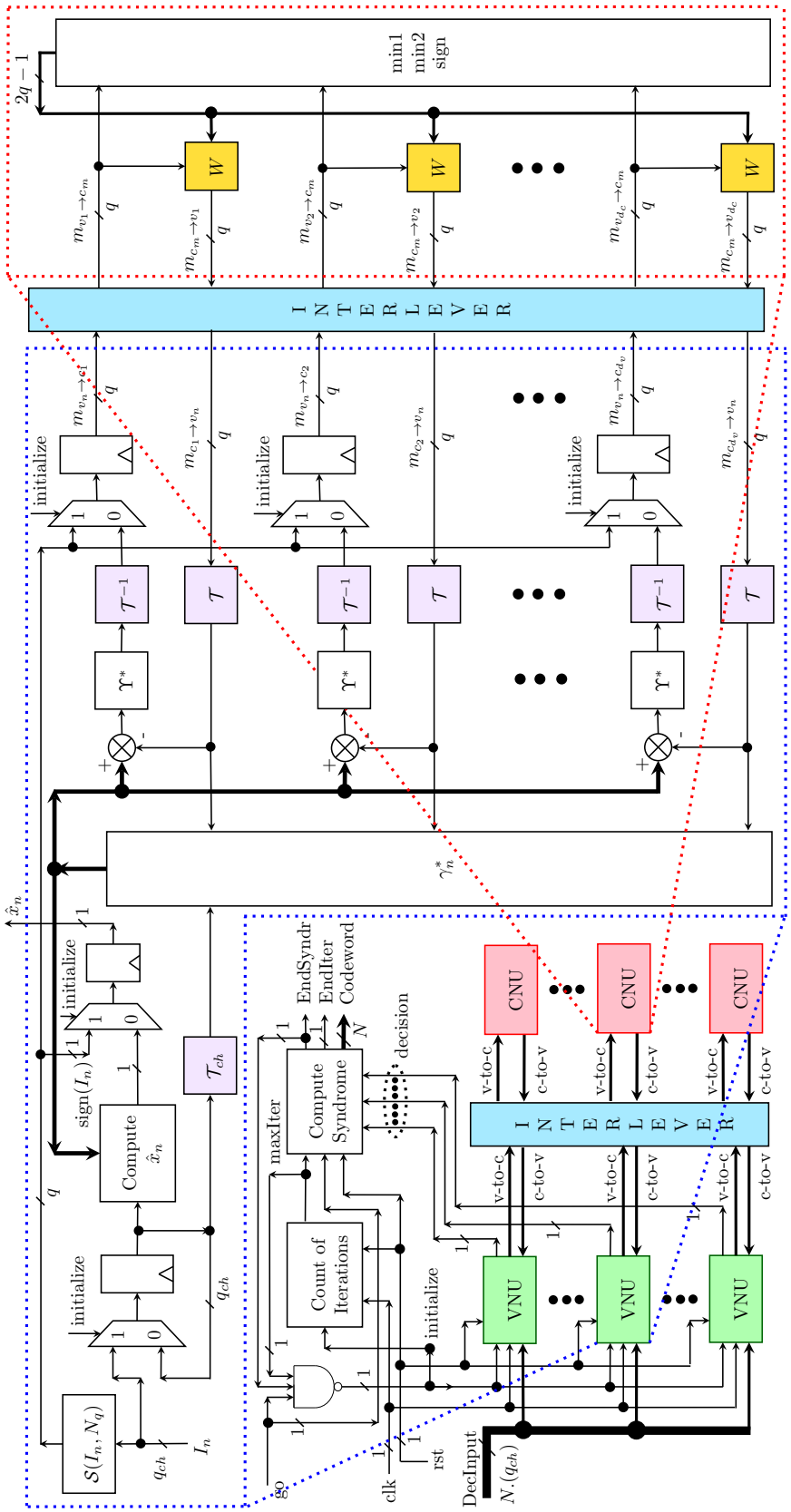


Fig. 5.1 Proposed architecture for the SP-MS decoders.

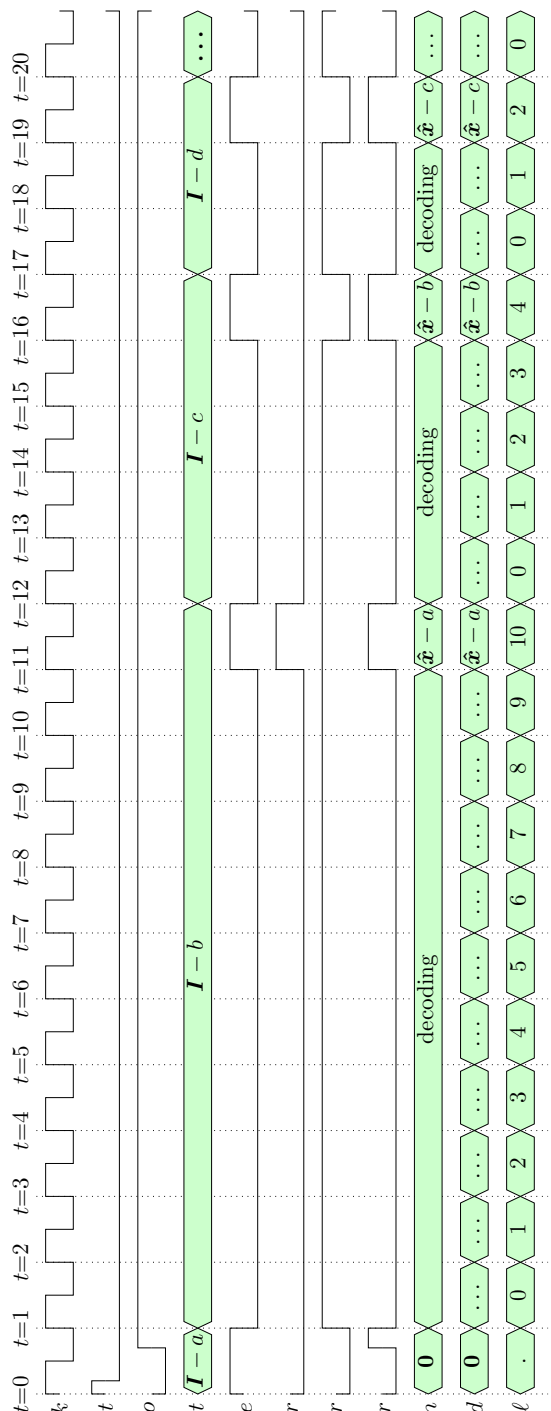


Fig. 5.2 Behavior of the main signals when an incorrect codeword is obtained using a maximum of 10 iterations, and when two correct codewords are computed.

We define the decoding throughput of the fully parallel architecture as

$$\text{Throughput} = \frac{N \times \text{Freq}}{L_{max}}, \text{ Gbit/s} \tag{5.1}$$

where Freq (in GHz) is the operating frequency and L_{max} is the maximum number of clock cycles to decode a codeword.

5.3 Implementation Results of SP-MS Decoders and MS-Based Decoders

5.3.1 Synthesis results on FPGA

In this section, we present the synthesis result of the proposed architecture on the Xilinx XC7V2000T-1FLG1925 FPGA chip for various LDPC codes with different rates and lengths. The main available resources of the FPGA are: *(i)* 2,443,200 Slice Registers, *(ii)* 1,221,600 Slice LUTs, and *(iii)* 351,321 LUT-FF pairs.

The FPGA resource utilization of the MS, OMS, and SP-MS decoders is listed in Table 5.1 for the (d_v, d_c)-regular LDPC codes considering fully parallel architecture. In addition, the maximum frequency that each decoder can reach is listed. The length of the regular LDPC code is $N = 1296$ for $d_v \in 3, 4$, $N = 1280$ for $d_v = 5$, and $N = 2048$ for $d_v = 6$ (ETHERNET code).

From the results obtained, one can see that the maximum frequency of $(q_{ch} = 3, q = 3)$-bit SP-MS decoders is slightly higher compared to the maximum frequency of the $(q_{ch} = 3, q = 3)$-bit MS and $(q_{ch} = 3, q = 3)$-bit OMS decoders. For the precision $(q_{ch} = 4, q = 4)$, the maximum frequency of SP-MS decoders is close or slightly higher compared to the maximum frequency of the MS and the OMS decoders. When comparing the use of FPGA resources by different decoders, one can observe that on average (in

Table 5.1 Synthesis results on FPGA of the MS, OMS, and SP-MS decoders for the (d_v, d_c)-regular LDPC codes.

(d_v, d_c)-regular LDPC code									
		Precision $(q_{ch}=3, q=3)$				Precision $(q_{ch}=4, q=4)$			
(d_v, d_c)	Decoder	Max. Freq. (MHz)	Number of slice registers	Number of slice LUTs	Number of fully used LUT-FF pairs	Max. Freq. (MHz)	Number of slice registers	Number of slice LUTs	Number of fully used LUT-FF pairs
$(3,6)$	MS	150.524	16854	152030	16854	128.295	22038	248147	16854
	OMS	143.184	16854	144252	12966	105.303	22038	205812	16854
	SP-MS	153.566	16854	148212	16854	120.613	22038	268019	16854
$(4,8)$	MS	127.389	20742	219608	16854	114.666	27222	284670	22038
	OMS	129.618	20742	212316	16854	107.968	27222	263046	22038
	SP-MS	133.631	20742	139741	16854	113.891	27222	250546	22038
$(5,10)$	MS	121.575	24326	285527	20486	98.338	32006	330973	26886
	OMS	122.066	24326	260454	20486	104.819	32006	318941	26886
	SP-MS	134.174	24326	294016	24326	103.864	32006	303822	26886
$(6,32)$	MS	112.705	45061	482236	45061	73.451	59397	696906	51205
	OMS	111.100	45061	463739	38917	87.790	59397	666300	51205
	SP-MS	114.071	45061	358221	45061	90.435	59397	638093	59397

some cases a little more and in other cases a little less) the SP-MS decoder uses as many resources as the MS and the OMS decoder. Therefore, we can conclude that the complexity increment of the SP-MS decoder is negligible compared to the MS and OMS decoders when the same precision is used.

Table 5.2 shows the case of SP-MS decoders in which the message precision is different from the LLR precision for the IEEE ETHERNET code. From the obtained synthesis results, we can clearly see that the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoders use less resources than the $(q_{ch}, q = q_{ch})$-bit SP-MS decoders. A large savings of FPGA resources can be observed: around 27% of slice registers and 35% of slice LUTs for the precision $(q_{ch} = 3, q = 2)$ compared to $(q_{ch} = 3, q = 3)$, and 20% of slice registers and 10% of slice LUTs for the precision $(q_{ch} = 4, q = 3)$ compared to $(q_{ch} = 4, q = 4)$.

The synthesis results of the WIMAX rate 1/2 LDPC code is listed in Table 5.3. The results show that the maximum frequency reached by the SP-MS decoders is a little higher than the maximum frequency of the MS and OMS decoders. Also, for both

Table 5.2 Synthesis results on FPGA of the SP-MS decoders for the IEEE 802.3 ETHERNET code.

IEEE 802.3 ETHERNET code, SP-MS decoders												
(q_{ch}, q)	Max. Freq. (MHz)	Number of slice registers		Number of slice LUTs		Number of fully used LUT-FF pairs		Throughput with 10 iter (Gbps)
(3,3)	114.071	45061	(0.0%)	358221	(0.0%)	45061	(0.0%)	23.3617
(3,2)	123.843	32773	(−27.27%)	232797	(−35.01%)	32773	(−27.27%)	25.3630
(4,4)	90.435	59397	(0.0%)	638093	(0.0%)	59397	(0.0%)	18.5211
(4,3)	91.329	47109	(−20.68%)	571194	(−10.48%)	38917	(−34.48%)	18.7042

precisions $q_{ch} = q = 3$ and $q_{ch} = q = 4$, when comparing the use of FPGA resources, the SP-MS decoders use a little less resources compared to the MS and OMS decoders. In this case, the complexity of the SP-MS decoder architecture is not increased.

Table 5.3 Synthesis results on FPGA of the SP-MS decoders for the WIMAX LDPC code.

WIMAX LDPC code with $R = 1/2$								
	Precision $(q_{ch} = 3, q = 3)$				Precision $(q_{ch} = 4, q = 4)$			
Decoder	Max. Freq. (MHz)	Number of slice registers	Number of slice LUTs	Number of fully used LUT-FF pairs	Max. Freq. (MHz)	Number of slice registers	Number of slice LUTs	Number of fully used LUT-FF pairs
MS	121.730	31110	234400	31110	106.646	40710	361542	31494
OMS	123.844	31110	250685	24198	103.659	40710	397438	31494
SP-MS	130.316	31110	223382	31110	111.557	40710	342105	31494

5.3.2 ASIC synthesis results

This section reports the ASIC synthesis results for the MS, OMS, and SP-MS decoders using the 28 nm Fully Depleted of Silicon-on-Insulator (FDSOI) library at 0.9 V supply voltage and temperature of 125 °C.

The timing constraints used to perform the synthesis of all decoders are: *(i)* 0.05 ns of clock skew, *(ii)* 0.5 ns of output delay, and *(iii)* $T - 0.2$ ns of input delay, where

T is the clock period (in ns) obtained from the operating frequency of the decoder, *i.e.* $T = 1/\text{Freq}$.

In order to measure the increase or decrease in complexity and power consumption of the SP-MS decoders, we list the area and power of each VNU/CNU of the MS, OMS, and SP-MS decoders in Table 5.4, Table 5.5, and Table 5.6. It must be taken into account that these tables were obtained considering an isolated VNU and an isolated CNU. One can note that for a fixed precision (q_{ch}, q) and a fixed degree distribution $(\lambda(x), \rho(x))$, the same CNU is used to implement the MS, the OMS, and the SP-MS decoder. Therefore, the difference of the MS, OMS, and SP-MS decoders lies in the VNU.

From the results presented in Table 5.4, we can observe that a VNU of the SP-MS decoder uses less area than a VNU of the OMS decoder and more area than a VNU of the MS decoder. In other words, the VNU of the proposed decoder (SP-MS) helps to reduce the area used by the architecture of the OMS decoder. Also, when comparing the power consumption of the VNU, the SP-MS consumes less power than the OMS and more power than the MS. Fig 5.3 shows the area utilization and power consumption of a VNU of degree $d_v = 6$ for the MS, OMS, and SP-MS decoders.

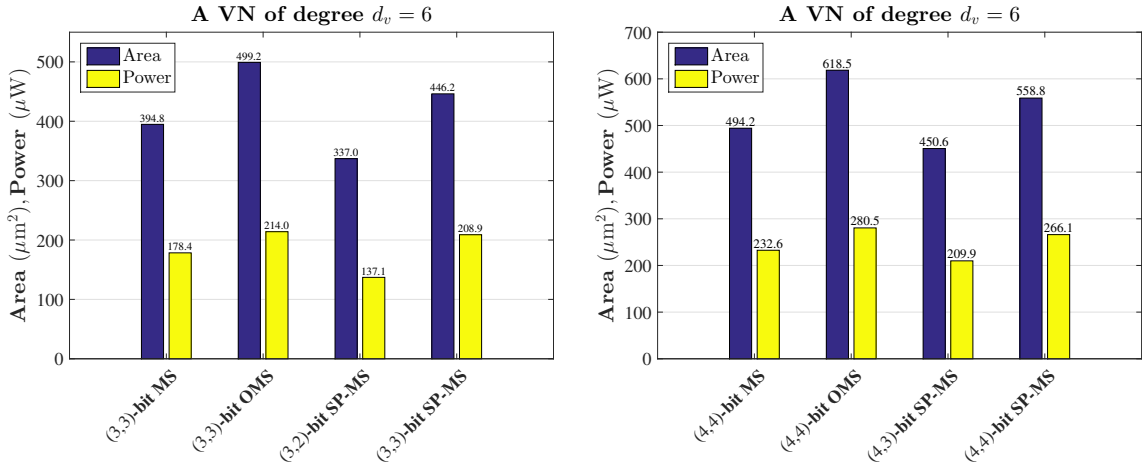
Table 5.5 and Fig. 5.4 show the area utilization and power consumption of a VNU of the SP-MS decoder when the precision of the messages goes from $q = q_{ch}$ bits to $q = q_{ch} - 1$ bits. One can note that a VNU/CNU of the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoder helps to reduce the area used and the power consumed by a VNU/CNU of the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder. For the case of the precision $(q_{ch} = 3, q = 2)$: *(i)* the CNU reduces an area of 459.73 μm^2 by 59.82% to 184.74 μm^2, *i.e.* a saving of around 60% is achieved, and *(ii)* the VNU reduces an area of 446.2 μm^2 by 24.47% to 337.0 μm^2. In the case of the precision $(q_{ch} = 4, q = 3)$, *(i)* the CNU helps to reduce an area of 883.24 μm^2 by 47.95% to 459.73 μm^2 (saving approximately 50% of the area), and

Table 5.4 ASIC synthesis results using the 28 nm FDSOI library for only one VNU of degree d_v and only one CNU of degree d_c.

(d_v, d_c)-regular LDPC code											
		Precision $(q_{ch}=3, q=3)$					Precision $(q_{ch}=4, q=4)$				
		Variable-Node			Check-Node		Variable-Node			Check-Node	
(d_v, d_c)	Decoder	Freq. (MHz)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Freq. (MHz)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)
(3,6)	MS	800	186.05	92.375	81.76	20.396	800	239.25	121.0	173.64	44.724
	OMS		225.05	101.90				291.15	140.2		
	SP-MS		195.51	100.70				258.84	134.7		
(4,8)	MS	700	269.28	124.3	110.81	29.005	700	337.66	162.9	228.15	58.902
	OMS		338.80	143.9				420.57	189.8		
	SP-MS		306.0	141.7				384.17	182.6		
(5,10)	MS	600	328.36	140.7	139.86	38.437	600	411.75	185.1	282.66	74.726
	OMS		415.99	167.0				515.38	220.0		
	SP-MS		362.47	161.2				456.63	207.5		
(6,32)	MS	600	394.8	178.4	459.73	165.7	600	494.2	232.6	883.24	237.3
	OMS		499.2	214.0				618.5	280.5		
	SP-MS		446.2	208.9				558.8	266.1		

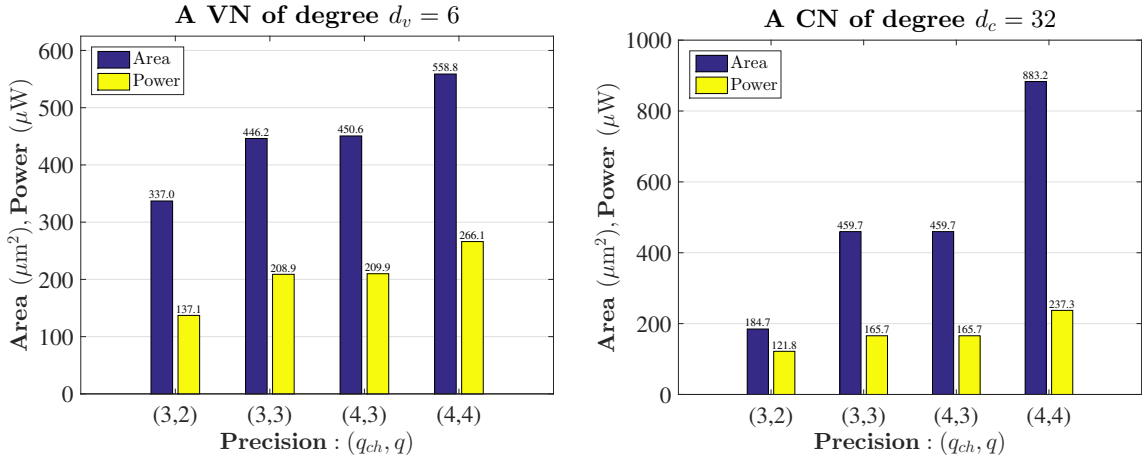
Table 5.5 ASIC synthesis results using the 28 nm FDSOI library for only one VNU of degree $d_v = 6$ and only one CNU of degree $d_c = 32$.

IEEE 802.3 ETHERNET code, SP-MS decoders									
Precision	Variable-Node					Check-Node			
(q_{ch}, q)	Freq. (MHz)	Area (μm^2)		Power (μW)		Area (μm^2)		Power (μW)	
(3,3)	600	446.2	(0.0%)	208.9	(0.0%)	459.73	(0.0%)	165.7	(0.0%)
(3,2)		337.0	(−24.47%)	137.1	(−34.37%)	184.74	(−59.82%)	121.8	(−26.49%)
(4,4)	600	558.8	(0.0%)	266.1	(0.0%)	883.24	(0.0%)	237.3	(0.0%)
(4,3)		450.6	(−19.36%)	209.9	(−21.12%)	459.73	(−47.95%)	165.7	(−30.17%)



(a) Precision $q_{ch} = 3$. (b) Precision $q_{ch} = 4$.

Fig. 5.3 The area utilization and power consumption of a VNU of the IEEE 802.3 ETHERNET code for the MS, OMS, and SP-MS decoders.



(a) A VNU of degree $d_v = 6$. (b) A CNU of degree $d_c = 32$.

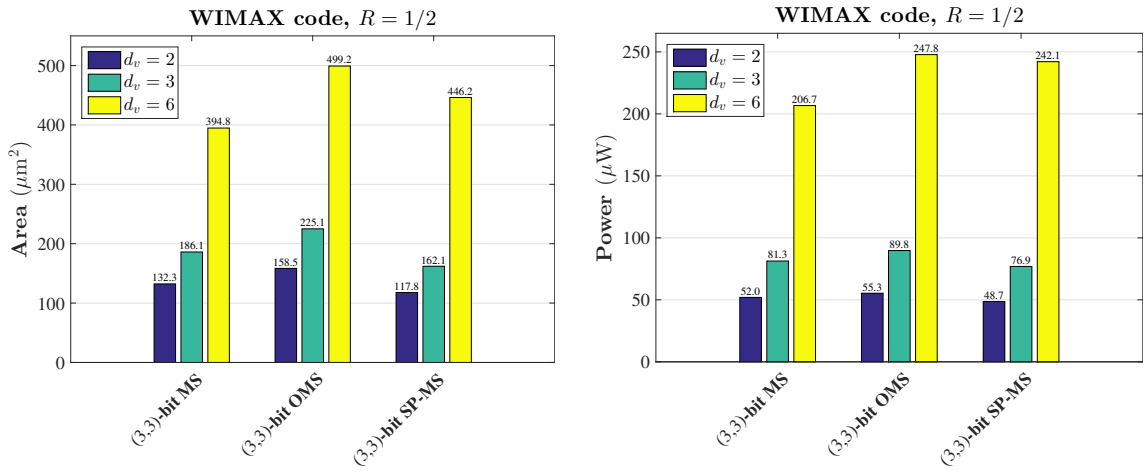
Fig. 5.4 The area utilization and power consumption of a VNU and a CNU of the IEEE 802.3 ETHERNET code for $q \in \{2, 3, 4\}$ bits of precision.

(ii) the VNU reduces the area by 19.36%, *i.e.*, from 558.8 μm^2 to 450.6μm^2. When comparing the power consumption, one can see a power consumption saving of 34.37% (from 208.9 μW to 137.1 μW) for the VNU and very low precision ($q_{ch} = 3, q = 2$). More results are shown in the Table 5.5.

We can conclude that using one bit less to represent the messages, The VNU and especially the CNU can greatly reduce the area used to implement the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoder compared to the area used by the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder.

Table 5.6 ASIC synthesis results using the 28 nm FDSOI library for the degree distribution $(\lambda(x), \rho(x))$, with $\lambda(x) = \frac{22}{76}x + \frac{24}{76}x^2 + \frac{30}{76}x^5$ and $\rho(x) = \frac{48}{76}x^5 + \frac{28}{76}x^6$

WIMAX rate 1/2 LDPC code												
q	Decoder	VN of $d_v = 2$			VN of $d_v = 3$		VN of $d_v = 6$		CN of $d_c = 6$		CN of $d_c = 7$	
		Freq. (MHz)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)
3	MS	700	132.35	51.979	186.05	81.314	394.78	206.7	81.76	20.396	93.84	24.760
	OMS		158.47	55.339	225.05	89.759	499.23	247.8				
	SP-MS		117.83	48.683	162.06	76.941	446.19	242.1				
4	MS	700	171.36	67.890	239.25	106.5	494.17	269.6	173.64	44.724	195.19	52.270
	OMS		205.96	75.328	291.15	123.4	618.53	325.0				
	SP-MS		156.83	63.223	259.98	116.2	558.79	308.6				



(a) Area. (b) Power.

Fig. 5.5 The area utilization and power consumption of VNUs of the WIMAX rate 1/2 LDPC code for the MS, OMS, and SP-MS decoders using $q = 3$ bits of precision.

The ASIC synthesis results for the WIMAX rate 1/2 LDPC code are shown in Table 5.6 considering three VNUs of degree $d_v = 2 \in \{2,3,6\}$, and two CNUs of degree $d_c \in \{6,7\}$. Similar to the case of regular LDPC codes, the CNUs are the same for the MS, OMS, and SP-MS decoders, hence, the complexity of the architecture lies in the VNUs. From the results, one can see that: *(i)* for $d_v = 2$, the VNU of the SP-MS decoders is the smallest (compared to the other two VNUs) and is the one that consumes less power, the same is true for $(d_v = 3, q = 3)$, and *(iii)* for the VNU with $(d_v = 3, q = 4)$, the SP-MS decoder uses less area than the OMS and more area than the MS, the same happens for the VNU of $d_v = 6$. Fig 5.5 shows the area utilization and power consumption of the three VNUs for the MS, OMS, and SP-MS decoders.

Let us now present the ASIC synthesis results for the fully parallel architecture. One should note that for a fixed precision (q_{ch}, q) and a fixed degree distribution $(\lambda(x), \rho(x))$, the total number of wires is the same for the MS, OMS, and SP-MS decoders. To implement the three decoders (MS, OMS, and SP-MS), we only need to use the appropriate VNU in the architecture presented in Fig. 5.1. Hence, the area used by the decoders is linear to the area used by a single VNU. But the power consumption of the decoder is not linear to the power consumption of a single VNU, this is because in the decoder a VNU is no longer isolated but interacts with the CNUs.

In order to measure the energy needed to decode a bit for the fully parallel architecture, we define the energy per decoded bit (EpB) given by

$$\text{EpB} = \frac{\text{Power}}{\text{Throughput}}, \text{ pJ/bit} \tag{5.2}$$

where Throughput (in Gbps) is the decoding throughput, and Power (in mW) is the power dissipation of the quantized decoder.

Analyzing the results obtained for (d_v, d_c)-regular LDPC decoders listed in Table 5.7, two conclusions can be stated: *(i)* the SP-MS decoder helps reduce the area used

by the OMS decoder when the same precision is used, and *(ii)* the SP-MS decoder always consumes slightly more power than the MS and OMS, *i.e.* the SP-MS decoder uses slightly more energy to decode a bit. To compute the decoding throughput, we consider $L_{max} = 20$ for $d_v \in \{3, 4, 5\}$ and $L_{max} = 10$ for $d_v = 6$.

Table 5.7 ASIC synthesis results using the 28 nm FDSOI library for (d_v, d_c)-regular LDPC codes, with $N = 1296$ for $d_v \in \{3, 4\}$, $N = 1280$ for $d_v = 5$, and $N = 2048$ for $d_v = 6$

(d_v, d_c)-regular LDPC code, BI-AWGN channel											
		Precision ($q_{ch} = 3, q = 3$)					Precision ($q_{ch} = 4, q = 4$)				
(d_v, d_c)	Decoder	Freq. (MHz)	Area (mm^2)	Power (mW)	EpB (pJ/bit)	Throughput (Gbps)	Freq. (MHz)	Area (mm^2)	Power (mW)	EpB (pJ/bit)	Throughput (Gbps)
$(3, 6)$	MS	800	0.298417	114.56	2.2099	51.84	800	0.426875	171.52	3.3086	51.84
	OMS		0.352821	102.10	1.9695			0.551994	182.48	3.5201	
	SP-MS		0.310467	127.62	2.4618			0.508916	193.39	3.7305	
$(4, 8)$	MS	700	0.428281	134.71	2.9698	45.36	700	0.588991	203.17	4.4791	45.36
	OMS		0.522132	119.08	2.6252			0.738115	200.73	4.4253	
	SP-MS		0.475490	151.35	3.3366			0.714808	225.82	4.9784	
$(5, 10)$	MS	600	0.519925	137.64	3.5844	38.40	600	0.714616	211.81	5.5159	38.40
	OMS		0.631893	121.01	3.1513			0.850135	197.48	5.1427	
	SP-MS		0.561353	168.54	4.3891			0.778968	224.12	5.8365	
$(6, 32)$	MS	600	1.007180	189.64	1.5433	122.88	600	1.365748	299.58	2.4380	122.88
	OMS		1.221662	211.45	1.7208			1.658770	290.65	2.3653	
	SP-MS		1.105530	336.91	2.7418			1.540191	419.17	3.4112	

Let us now study the SP-MS decoders considering that the message precision is different from the LLR precision. Table 5.8 shows the area utilization and power consumption for the $d_v = 6$ RS-LDPC decoders, also, the decoding throughput and the energy per decoded bit are listed. For a frequency of 600 MHz with a maximum of 10 iterations, the minimum throughput reached is 122.88 Gbps. The average number of decoding iterations in a SP-MS decoder at $E_b/N_0 = 4.75$ dB is around 3, hence, the decoding throughput reached is around 409.60 Gbps.

From the results, we can confirm and conclude that the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoder uses less area and consumes less power than the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder. We can observe that the $(q_{ch} = 3, q = 2)$-bit SP-MS decoder reduces an area of 1.105 mm^2 by 29.96% down to 0.774 mm^2, *i.e.* a saving of around 30% is achieved,

and the $(q_{ch}=4, q=3)$-bit SP-MS decoder reduces an area of 1.540 mm^2 by 27.60% to 1.115 mm^2 (saving approximately 28% of the area). When comparing the power consumption, one can see a power consumption saving of 28.11% (from 336.91 mW to 242.19 mW) for very low precision $(q_{ch}=3, q=2)$, while for precision $(q_{ch}=4, q=3)$, the reduction of power consumption is 22.73%, *i.e.*, from 419.17 mW to 323.91 mW.

Table 5.8 ASIC synthesis results using the 28 nm FDSOI library for the (q_{ch}, q)-bit SP-MS decoders.

IEEE 802.3 ETHERNET code, SP-MS decoders						
Precision (q_{ch}, q)	Freq. (MHz)	Area (mm^2)		Power (mW)		EpB (pJ/bit)		Throughput (Gbps)
(3,3)	600	1.105530	(0.0%)	336.91	(0.0%)	2.7418	(0.0%)	122.88
(3,2)		0.774331	(−29.96%)	242.19	(−28.11%)	1.9709	(−28.11%)	
(4,4)	600	1.540191	(0.0%)	419.17	(0.0%)	3.4112	(0.0%)	122.88
(4,3)		1.115132	(−27.60%)	323.91	(−22.73%)	2.6360	(−22.73%)	

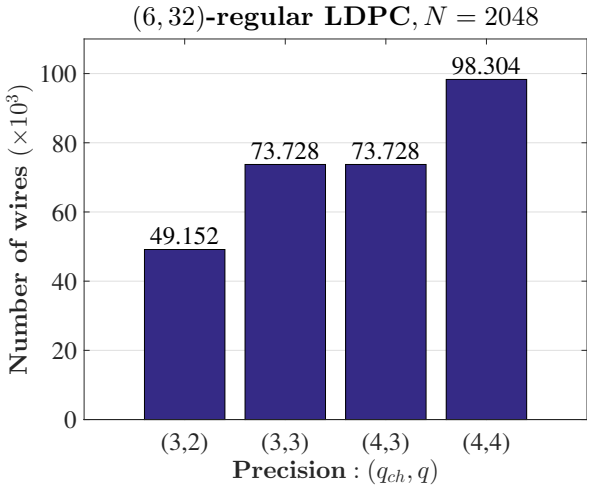


Fig. 5.6 Total number of wires of a fully parallel architecture for the (q_{ch}, q)-bit SP-MS decoders.

As we implement the decoders using a fully parallel architecture (N VNUs and M CNUs), the total number of wires used for the SP-MS decoder depends on the precision used, specifically of the precision used by the messages, *i.e* the total number of wires depends on q. Fig. 5.6 depicts the total number of wires of the ETHERNET code for

the (q_{ch}, q)-bit SP-MS decoders. When the precision q goes from 3 bits to 2 bits, the reduction of wires is 33.33%, in the case that the precision q goes from 4 bits to 3 bits, a reduction of 25% of wires is obtained. One can note that the $(4,3)$-bit SP-MS and the $(3,3)$-bit SP-MS use the same amount of wires.

Fig. 5.7 illustrates the area used for the (q_{ch}, q)-bit SP-MS decoders. We can see that the $(3,2)$-bit SP-MS decoder is the smallest decoder, the $(4,4)$-bit SP-MS decoder is the biggest decoder, and the $(4,3)$-bit SP-MS decoder occupies an area slightly greater than the $(3,3)$-bit SP-MS decoder. Fig. 5.8 shows the area used and the power consumption for the MS, OMS, and SP-MS decoders. One can observe that the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoder is the smallest decoder, and it consumes slightly more power than the MS and OMS decoder.

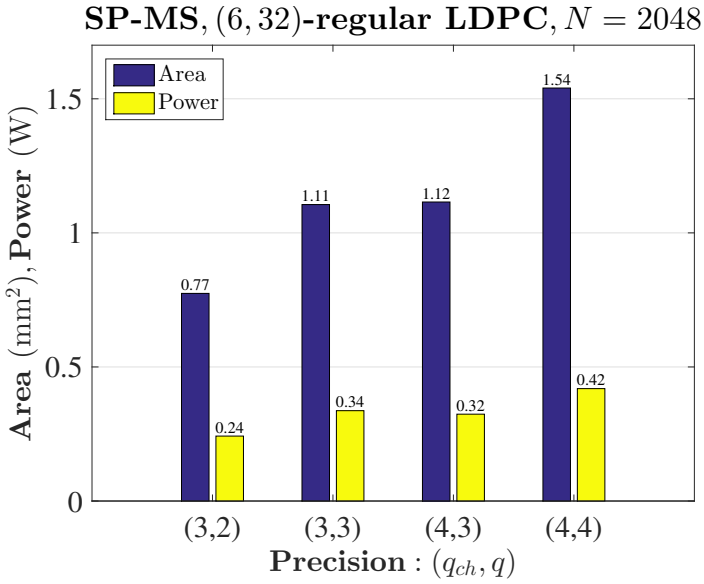
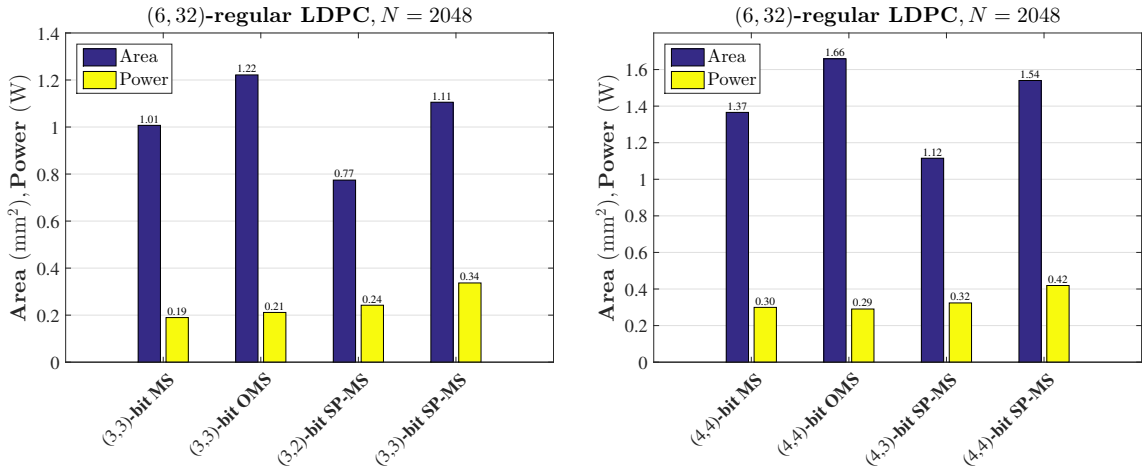


Fig. 5.7 The area utilization and power consumption of the (q_{ch}, q)-bit SP-MS decoders.

From the analysis of the SP-MS decoders using q_{ch} bits LLRs and q bits messages, with $q \in \{2,3,4\}$ and $q_{ch} \in \{3,4\}$, we can conclude that: *(i)* the $(3,2)$-bit SP-MS decoder will always be the smallest decoder, *(ii)* the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder always occupies an area smaller than the $(q_{ch}, q = q_{ch})$-bit OMS decoder, and *(iii)* the $(4,3)$-bit



(a) Precision $q = q_{ch} = 3$. (b) Precision $q_{ch} = 4$.

Fig. 5.8 The area utilization and power consumption of the MS, OMS, and SP-MS decoders considering the IEEE 802.3 ETHERNET code.

SP-MS decoder occupies an area slightly greater than the $(3,3)$-bit SP-MS decoder (both decoders have the same number of wires).

Although it is true that the results presented, considering $q = q_{ch} - 1$, are for the $(d_v = 6, d_c = 32)$-regular LDPC code (IEEE 802.3 ETHERNET code), the conclusions *(i)* and *(ii)* can be extended for other (d_v, d_c)-regular LDPC codes.

Table 5.9 ASIC synthesis results using the 28 nm FDSOI library for the WIMAX LDPC code with $R = 1/2$ and $N = 2304$

Irregular LDPC code, BI-AWGN channel										
	Precision $(q_{ch} = 3, q = 3)$					Precision $(q_{ch} = 4, q = 4)$				
Decoder	Freq. (MHz)	Area (mm^2)	Power (mW)	EpB (pJ/bit)	Throughput (Gbps)	Freq. (MHz)	Area (mm^2)	Power (mW)	EpB (pJ/bit)	Throughput (Gbps)
MS	800	0.598627	227.76	2.4714	92.16	700	0.816824	289.99	3.5961	80.64
OMS		0.747223	209.81	2.2766			1.003706	293.92	3.6448	
SP-MS		0.608703	247.91	2.6900			0.913862	306.86	3.8053	

Table 5.9 summarizes the ASIC synthesis results for the WIMAX rate 1/2 LDPC code. The decoding throughput is computed considering $L_{max} = 20$. Those results confirm the conclusions of the regular LDPC codes analysis: *(i)* the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder occupies less area than the $(q_{ch}, q = q_{ch})$-bit OMS decoder, *(ii)* the SP-MS decoder consumes slightly more power than the MS and OMS, *i.e.* the SP-MS

decoder uses slightly more energy to decode a bit when using $q = q_{ch}$. Fig. 5.9 depicts the area utilization and power consumption of the decoders considered.

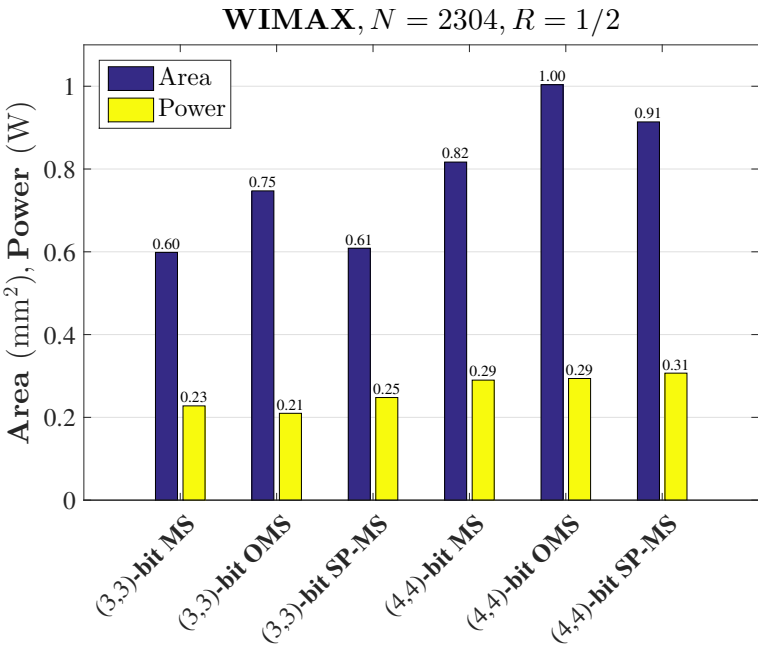


Fig. 5.9 The area utilization and power consumption of the WIMAX rate 1/2 LDPC code for the MS, OMS, and SP-MS decoders

5.4 The (8,8) absorbing set in SP-MS decoders

It is well known that trapping sets [68] are responsible for the loss of performance with the appearance of the error-floor of MP decoders in high SNR values. It is worth noting that the effect of quantization and saturation of messages can exacerbate the effect of the trapping sets. Of course the appearance of the error depends on the precision and the decoding algorithm used [17]. In the literature, a general (a, b) trapping set is defined as a set of a VNs which induces a subgraph with exactly b odd-degree CNs and an arbitrary number of even-degree CNs.

Absorbing sets were introduced in [78, 69, 79]. The authors have defined an absorbing set as a particular kind of trapping set [68]. Let us take the notations for an (a, b)

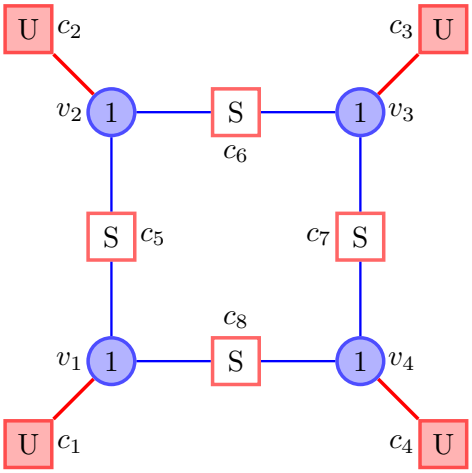
absorbing set given in [78, 69, 79]: *(i)* D denotes the absorbing set of size a, *(ii)* $O(D)$ denotes the set of unsatisfied CNs, where the size of $O(D)$ is b, *(iii)* $N(D)$ denotes the set of VNs connected to the unsatisfied CNs in $O(D)$, and *(iv)* $S(D)$ denotes the set of neighboring satisfied CNs to the VNs in $N(D)$, and it is composed for the falsely satisfied CNs and for the correctly satisfied CNs. An example of of an $(4,4)$ fully absorbing set is depicted in Fig. 5.10b.

This chapter studies and analyzes the $(8,8)$ trapping sets of the IEEE 802.3 ETHERNET ((2048,1723) RS-LDPC code). It has been shown that all $(8,8)$ trapping sets are fully absorbing sets [69], hence $|O(D)| = 8$, $|N(D)| = 256$, and $|N(D)\backslash D| = 248$. Also, the error-floor exhibited by quantized (2048,1723) RS-LDPC decoders (without an post-processing algorithm) is around 10^{-8} of FER and is mainly due to $(8,8)$ trapping-sets [69, 80]. The (q_{ch}, q)-bit SP-MS decoder also exhibits the appearance of the error between 10^{-7} and 10^{-8} of FER as shown in Fig. 5.11. The IEEE 802.3 ETHERNET code is selected to provide an error-free operation below the bit error rate (BER) level of 10^{-12} which corresponds to a frame error rate (FER) $\approx 10^{-10}$. Hence, the need to correct the $(8,8)$ absorbing set errors that are the dominant trapping sets and are the main responsible for the error-floor.

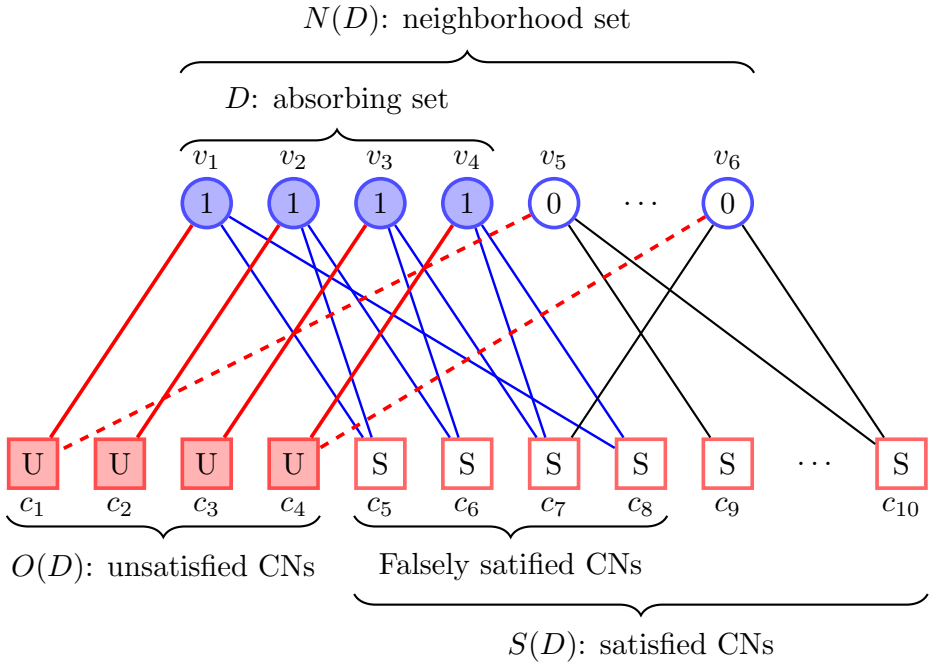
5.5 SP-MS Decoder with Post-Processing Using the Same Precision for Messages and LLRs

5.5.1 Post-Processing

In high-speed quantized decoders, the decoder can not know the absorbing set D, it is only possible to know the sets $N(D)$, $O(D)$, and $S(D)$ in order to correct the absorbing set errors. In [69, 54], the authors have defined a post-processing algorithm



(a) A $(4,4)$ trapping set.



(b) A $(4,4)$ fully absorbing set.

Fig. 5.10 Graphical representation of $(4,4)$ trapping set.

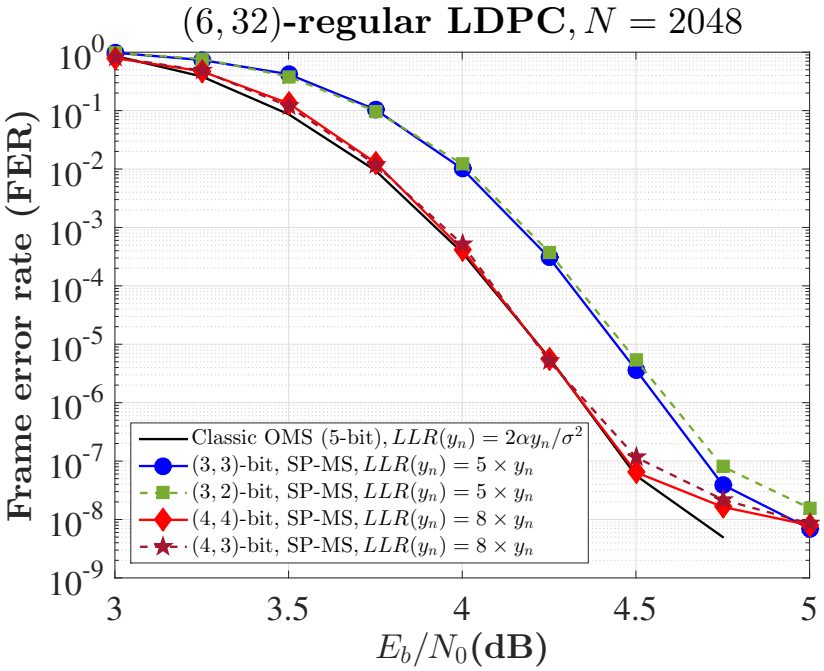


Fig. 5.11 Error-floor of (q_{ch}, q)-bit SP-MS decoders considering the IEEE 802.3 ETHERNET code.

implemented in a 4-bit OMS decoder (with $q_{ch} = q = 4$). The post-processing algorithm consists in biasing messages, specifically all messages from CNs in $O(D)$ to VNs in $N(D)$ are set to a strong value, denoted *strong*, and all messages from CNs in $S(D)$ to VNs in $N(D)$ are set to a weak value, denoted *weak*.

The post-processing algorithm implemented in the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder uses the same principle as the algorithm proposed in [69]. In SP-MS decoders, it is sufficient to know the set of unsatisfied CNs $O(D)$ to correct the $(8, 8)$ absorbing set errors. It is worth noting that knowing the set $O(D)$ one can identify all the VNs connected to the unsatisfied CNs.

Two decoding steps are performed to correct the absorbing set errors:

1. **[SP-MS decoder]**

   Decode a codeword with the SP-MS decoder for a fixed number of iterations. If the syndrome vector is not all-zero, continue with the next step.

2. [**Post-processing**]

   (a) Only in a single iteration, all messages from CNs in $O(D)$ to VNs in $N(D)$ are set to *strong*, and all messages from CNs in $S(D)$ to VNs in $N(D)$ are set to *weak*.

   (b) The decoding process continues with the SP-MS decoder until the syndrome vector is all-zero or the maximum number of iterations is reached.

Note that step (a) can be done locally at the VNU level and its procedure is described in the following section.

5.5.2 Post-Processing Architecture

This section presents an architecture designed mainly to correct the $(8,8)$ absorbing set errors. The $(q_{ch}, q = q_{ch})$-bit SP-MS decoder and the message biasing are implemented together, thus adding only a little complexity to the decoder. Fig. 5.12 shows the proposed fully parallel architecture to correct the $(8,8)$ absorbing set errors knowing only the unsatisfied CNs. The input signals of the architecture are *clk*, *rst*, *go*, *DecInput* of $q \times N$ bits, while the output signals are *Codeword*, *EndIter*, and *EndSyndr*. These input and output signals behave the same as the signals of the architecture shown in Fig. 5.1, *i.e.* the decoding process is enabled when $go = 1$ and $rst = 0$, a valid codeword is obtained when $EndSyndr = 0$, and the decoding process ends when $EndIter = 1$.

The proposed architecture is composed of five main components: **VNU**, **CNU**, **Count of Iterations**, **Compute Syndrome**, and **Usatisfied CNs**. The **VNU** component implements the post-processing, and the **CNU** component sends information about whether a CN is a satisfied CN or an unsatisfied CN. Let us discuss the five main components of the proposed architecture.

Count of Iterations

The input signals of this component are *clk, rst*, and *initialize*, and the output signals are *maxIter*, and *Itpp*. Let us denote by $L_{pp} \in \mathbb{N}$ the iteration number where the post-processing is enabled. One iteration is counted per each clock cycle when $initialize = 0$, in the case of $initialize = 1$ the v-to-c messages are initialized and no iteration is counted. Only when the count of iterations is equal to L_{pp}, the signal $Itpp$ is set to 1, otherwise $Itpp$ is always 0. Similarly, we have $maxIter = 1$ if and only if the maximum number of iterations is reached, otherwise we have $maxIter = 0$.

CNU Component

The proposed architecture uses $M = 384$ CNU components. Each CNU c_m is an asynchronous circuit that sends information about whether a CN is a satisfied CN or an unsatisfied CN.

Let us denote by $\boldsymbol{S} = (S_1, S_2, ..., S_M) \in \{0,1\}^M$ the syndrome vector, given by $\boldsymbol{S} = H\hat{\boldsymbol{x}}^T$, where $\hat{\boldsymbol{x}}$ is an estimation of a codeword $\boldsymbol{x}$. When the vector $\hat{\boldsymbol{x}}$ (computed from the APPs) is a valid codeword, the syndrome vector is all-zero ($\boldsymbol{S} = 0$), *i.e.* each CNU c_m is a satisfied CN. Each element S_m of the syndrome vector informs us about whether the CN is a satisfied CN ($S_m = 0$) or an unsatisfied CN ($S_m = 1$).

We can compute S_m as $S_m = [h_{mn}]\hat{\boldsymbol{x}}^T$ for $n = 1, ..., N$, but this calculation can be replaced by $S_m = \prod_{v_n \in \mathcal{V}(c_m)} \text{sign}\,(m_{v_n \to c_m})$ avoiding calculating the APPs and thus relaxing the computation of the syndrome vector $\boldsymbol{S}$. In other words, one can use the v-to-c message $m_{v_n \to c_m}$ instead of the APP γ_n to compute $\boldsymbol{S}$. This is because $m_{v_n \to c_m}$ and γ_n are estimates of the bit value associated with the VN v_n, of course, γ_n give us a higher reliability than $m_{v_n \to c_m}$.

The CNU c_m also computes min_1 and min_2, with $min_2 \geq min_1$. It is worth mentioning that the product of signs *sign* of all incoming messages to the CN c_m is

equal to S_m, *i.e.* $sign = S_m$. Fig. 5.12 shows *sign* as an output signal. Note that in binary representation, the sign product of the v-to-c messages is determined by the XOR logic of the MSBs of the v-to-c messages. Using those notations, the function W^* at a CNU c_m is defined as $W^*(a, S_m, min_1, min_2) = (S_m, min_2)$ if $|a| = min_1$, otherwise $W(a, S_m, min_1, min_2) = (S_m, min_1)$.

In the proposed architecture, the c-to-v messages $m^*_{c_m \to v}$, with $v \in \mathcal{V}(c_m)$, are computed as $m^*_{c_m \to v} = W(m_{v \to c_m}, S_m, min_1, min_2)$. The most significant bit of the c-to-v message $m^*_{c_m \to v}$ carries information about whether the CNU c_m is a satisfied CN or an unsatisfied CN.

Compute Syndrome

This component is an asynchronous circuit. Its input signals are *rst, go, maxIter*, and *SyndrVect* of M bits (syndrome vector $\boldsymbol{S}$), and its output signals are *EndSyndr* and *EndIter* which are computed at each decoding iteration.

When the decoding process is enabled, *i.e.* $rst = 0$ and $go = 1$, *EndSyndr* is determined with the OR logic of all the values of the syndrome vector, hence, a correct codeword is obtained when $EndSyndr = 0$ and the decoding process ends ($EndIter = 1$). When $EndSyndr = 1$, a valid codeword is not computed and the decoding process continues ($EndIter = 0$). When $maxIter = 1$, we have $EndIter = 1$, and $EndSyndr$ can be 0 or 1.

VNU Component

The proposed architecture uses $N = 2048$ VNU components. A VNU v_n component is a synchronous circuit that has as input signals to *clk, rst, initialize, post*, c-to-v messages $m^*_{c \to v_n}$, $c \in \mathcal{V}(v_n)$, and the quantized LLR I_n, while the output signals are v-to-c messages $m_{v_n \to c}$ and the estimated bit $\hat{x}_n$.

The mapping functions $\mathcal{T}$, $\mathcal{T}_{ch}$, and $\mathcal{T}^{-1}$ defined to implement SP-MS decoders do not change, see Section 5.2. The function V of Fig. 5.12 is defined as $V(a,b) = (\text{sign}(a) \times \text{sign}(b), |b|)$. The correct c-to-v message $m_{c \to v_n}$ is obtained as $m_{c \to v_n} = V\left(m_{v_n \to c}, m^*_{c \to v_n}\right)$.

Let denote b_i, with $i = 1, 2, ...d_v$, the sign of the c-to-v message $m^*_{c_i \to v_n}$, b_i carries information about whether the CNU c_i is a satisfied CN or not. The product of signs (prd) of the c-to-v message, *i.e.* $prd = \prod_{i \in \{1,...,d_v\}} b_i$, gives us the information about whether the VN v_n is connected to an unsatisfied CN c_i. Specifically, if $prd = -1$ (in binary representation $prd = 1$) then the VN v_n is connected to 1, 3, or 5 unsatisfied CNs.

The architecture shown in Fig. 5.12 is designed to correct the $(8,8)$ absorbing set errors. Only when the signal $post = 1$ (obtained in iteration L_{pp}) and when $prd = -1$, the message biasing is applied, *i.e.* the message $m_{c_i \to v_n}$ is mapped to *strong* if $b_i = -1$ (in binary representation $b_i = 1$), otherwise, the message $m_{c_i \to v_n}$ is mapped to *weak*. Let us take the example of a VN v_n connected to an unsatisfied CN c_1, and five satisfied CNs c_i with $i = 2, ..., 6$. Considering $post = 1$, we have: *(i)* $\gamma^*_n = \mathcal{T}_{ch}(I_n) + \mathcal{T}(strong) + 5 \times \mathcal{T}(weak)$, and *(ii)* $m_{v_n \to c_1} = \mathcal{T}^{-1}(\Upsilon^*(\gamma^*_n - \mathcal{T}(strong)))$, and $m_{v_n \to c_i} = \mathcal{T}^{-1}(\Upsilon^*(\gamma^*_n - \mathcal{T}(weak)))$ for $i = 2, ..., 6$.

For the proposed architecture, we determine by simulation that the optimal value of *weak* and *strong* are: *(i)* $weak = (\text{sign}(m), 00)$ and $strong = (\text{sign}(m), 11)$ for precision $q_{ch} = q = 3$, and *(ii)* while for precision $q_{ch} = q = 4$, $weak = (\text{sign}(m), 001)$ and $strong = (\text{sign}(m), 111)$, where $m = m_{c \to v_n}$. It must be taken into account that *weak* and *strong* always keep the sign of the c-to-v message $m_{c \to v_n}$.

Unsatisfied CNs

This component is a synchronous circuit. Its input signals are *clk, rst*, and *SyndrVect* of M bits ($\boldsymbol{S}$), and its output signal is *UnCN*. Let us denote by $S_{\hat{x}}$ the sum of all values of the syndrome vector $\boldsymbol{S}$ (computed from $\hat{\boldsymbol{x}}$), *i.e.* $S_{\hat{x}}$ is given by $S_{\hat{x}} = S_1 + S_2 + ... + S_M$. In other words, $S_{\hat{x}}$ gives us the number of unsatisfied CNs, for example, in an $(8,8)$ fully absorbing set, we have $S_{\hat{x}} = 8$.

In a VNU v_n when $post = 1$ and when $prd = -1$, the c-to-v message $m_{c \to v_n}$ can be mapped to *weak* or *strong*. The change in the magnitude of the message $m_{c \to v_n}$ is adequate if the codeword has converged into an $(8,8)$ absorbing set (or others absorbing sets like the $(7,12)$, $(11,6)$ absorbing sets, etc), that is, when the decoder operates in the error-floor region at high values of SNR ($E_b/N_0 > 4.5$ dB). In the case that the codeword has not yet converged into an absorbing set, *i.e.* the decoder operates in the waterfall region, the change in the magnitude of c-to-v message $m_{c \to v_n}$ will corrupt the decoding process making decoding much more complicated.

Unsatisfied CNs component adds a control signal $UnCN$ to the architecture and it is used to control the enabling of post-processing. The signal $UnCN$ is set to 1 if and only if the number of unsatisfied CNs is less than a threshold τ (*e.g.* $\tau = 31$), *i.e.* $S_{\hat{x}} < \tau$.

Using the signals $UnCN$ and $Itpp$, the signal *post* is calculated as $post = UnCN$ AND $Itpp$. Therefore, the message biasing is applied only when $UnCN = 1$ and $Itpp = 1$.

Fig. 5.13 shows a timing diagram of the main signals of the proposed post-processing architecture. We consider 9 iterations of SP-MS and 6 iterations of message biasing, a maximum of 15 iterations, $S_{\hat{x}} < 21$.

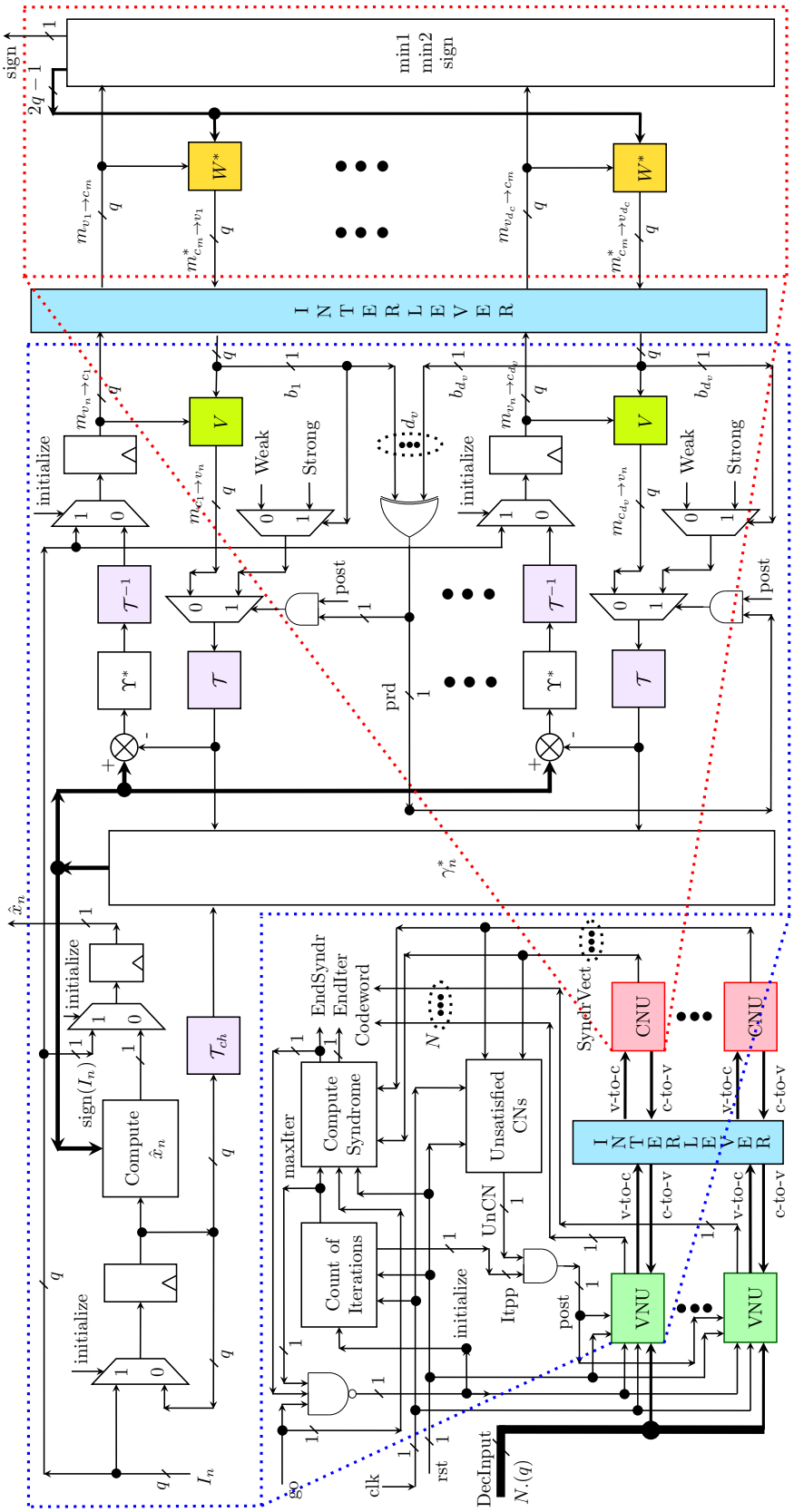
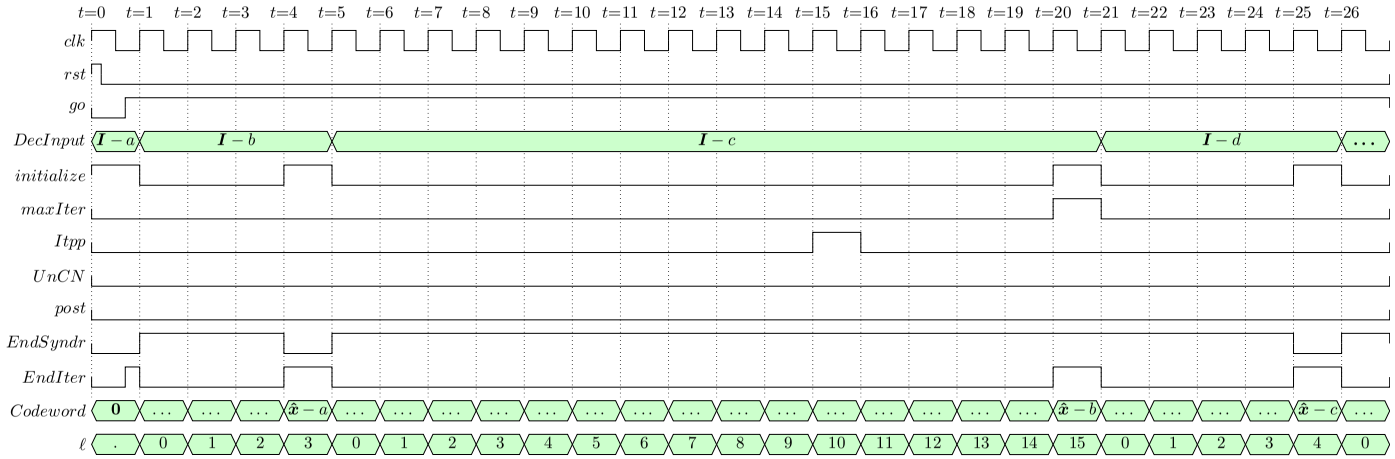
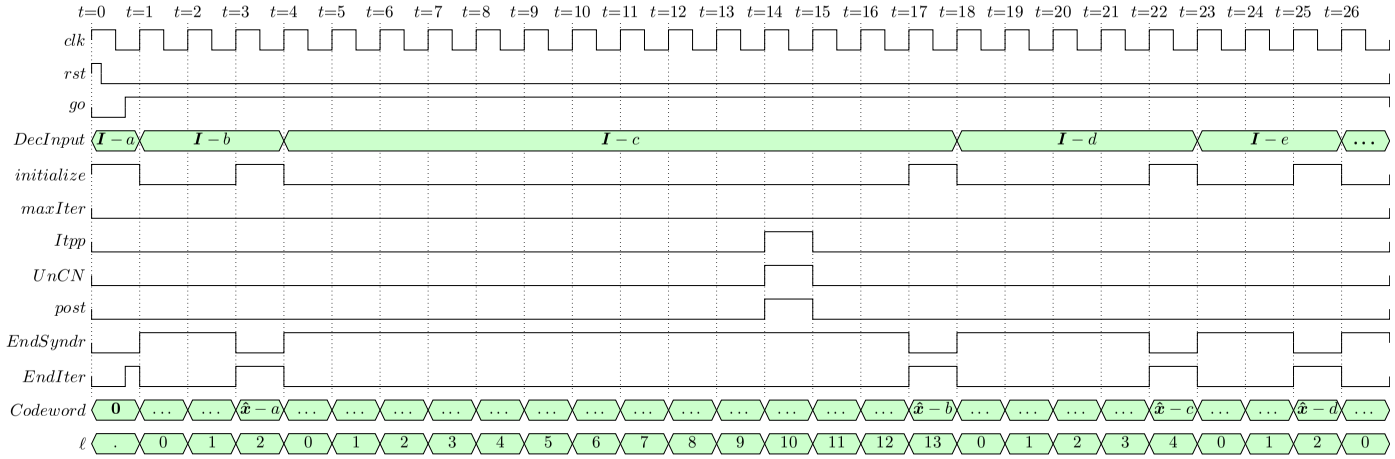


Fig. 5.12 Post-processing architecture of the $(q_{ch}, q = q_{ch})$-bit SP-MS decoders for the IEEE 802.3 ETHERNET code.



(a) Behavior of the main signals in the waterfall region.



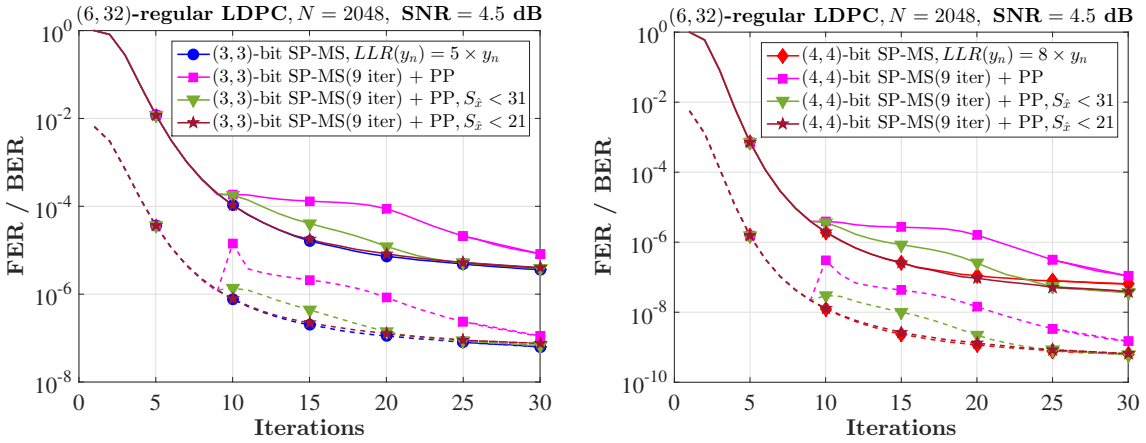
(b) Behavior of the main signals to correct an (8,8) absorbing set in the error-floor region.

Fig. 5.13 Behavior of the main signals of the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder with post-processing in the waterfall region and in the error-floor region. A maximum of 15 iterations is used.

5.5.3 Performance results

In this section, we present the FER performance of the proposed architecture for the two considered precisions $q_{ch} = q = 3$ and $q_{ch} = q = 4$.

Fig. 5.14 shows the FER convergence results at $E_b/N_0 = 4.5$ dB for the SP-MS, SP-MS + post-processing (PP) without the **unsatisfied CNs** component, and SP-MS + PP considering $S_{\hat{x}} < 21$ and $S_{\hat{x}} < 31$. We can see that the SP-MS + PP without controlling the number of unsatisfied CNs give us the worst FER/BER performance after enabling post-processing. The FER/BER performance of the SP-MS + PP is not degraded when the number of unsatisfied CNs is controlled considering $S_{\hat{x}} < 21$. In the case of $S_{\hat{x}} < 31$, a slight degradation of FER/BER performance is observed. Therefore, **Unsatisfied CNs** component with $\tau = 21$ helps the SP-MS decoder + PP to avoid decoding performance degradation in the waterfall region.



(a) Precision $q = q_{ch} = 3$.

(b) Precision $q_{ch} = 4$.

Fig. 5.14 FER (solid lines) and BER (dashed lines) convergence comparison on the IEEE 802.3 ETHERNET code at $E_b/N_0 = 4.5$ dB using a maximum of 9 iterations of $(q_{ch}, q = q_{ch})$-bit SP-MS and 21 iterations as maximum of post-processing.

The average number of iterations to correct the $(8,8)$ absorbing set errors is listed in Table 5.10. We can see that approximately 4 (resp. 3) iterations are required for post-processing when $q_{ch} = q = 3$ bits (resp. $q_{ch} = q = 4$) of precision is considered.

Table 5.10 Average number of iterations for post-processing using $(L_{pp}-1)$ iterations as maximum of the $(q_{ch}, q=q_{ch})$-bit SP-MS + $(L_{max}-L_{pp}+1)$ iterations as maximum of post-processing.

Average number of iterations for post-processing				
SNR (dB)	(3,3)-bit SP-MS, $L_{max}=15$		(4,4)-bit SP-MS, $L_{max}=15$	
	$L_{pp}-1=9$	$L_{pp}-1=10$	$L_{pp}-1=9$	$L_{pp}-1=10$
4.75	3.43	3.37	3.18	3.12
4.8	3.49	3.44	3.10	3.07
4.9	3.40	3.37	3.22	3.17
5.0	3.36	3.30	3.04	2.99

The proposed architecture of Fig. 5.12 for precision $(q_{ch}=3, q=3)$ is implemented in an FPGA to evaluate its performance. Two decoders are emulated: *(i)* the first one using a maximum of 11 iterations of the (3,3)-bit SP-MS + 8 iterations as maximum of post-processing, and the second one using a maximum of 9 iterations of the (3,3)-bit SP-MS + 6 iterations as maximum of post-processing. Emulations results show that the error-floor is lowered below the FER of 10^{-10} in the error-floor region, as is shown in Fig. 5.15. Also, the emulation confirms that in the waterfall region, the error correction performance is not degraded.

Simulation results for precision $(q_{ch}=4, q=4)$ are provided in Fig. 5.16 using a maximum of 9 iterations of the (4,4)-bit SP-MS + 6 iterations as maximum of post-processing. Again one can see that the FER performance is not degraded in the waterfall region, and the error floor can be lowered below the FER of 10^{-10}. The point at SNR level of 5.0 dB is obtained by dividing the number of error frames collected by the total number of frames. Similarly for the point at SNR level of 4.75 dB.

The emulation and simulation results show that the proposed architecture can be used for the IEEE 802.3 ETHERNET code.

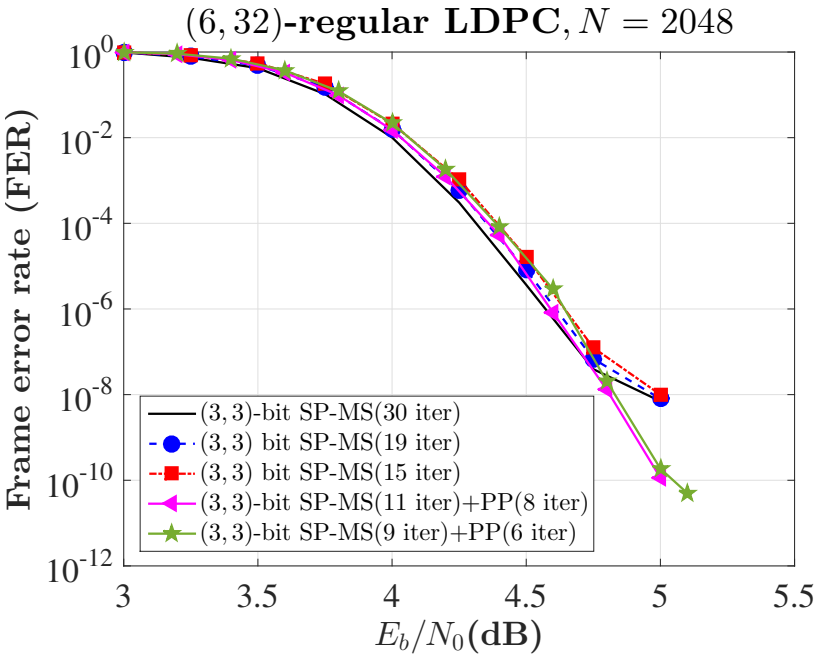


Fig. 5.15 FER performance of the $(3,3)$-bit SP-MS decoder with post-processing obtained by FPGA emulation for the IEEE 802.3 LDPC code.

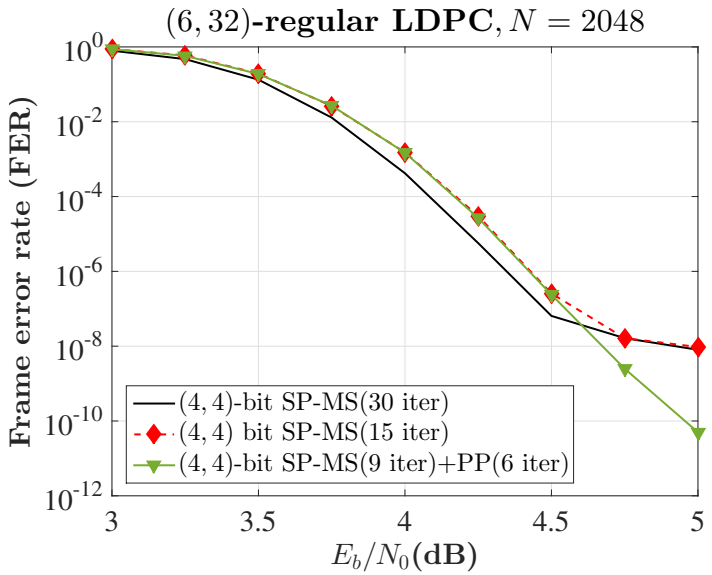


Fig. 5.16 FER performance of $(4,4)$-bit SP-MS decoder with post-processing for the IEEE 802.3 ETHERNET code..

5.6 SP-MS Decoder with Post-Processing Using Different Precision for Messages and LLRs

5.6.1 Post-Processing

In the previous section, the post-processing (message biasing) proved to be very effective in correcting the $(8,8)$ absorbing set errors for precision $q_{ch} = q \in \{3,4\}$. But for precision $(q_{ch} = 3, q = 2)$, message biasing can only correct around 30% of all the $(8,8)$ absorbing set errors collected at high SNR levels, as is shown in Table 5.11. For example, from the 281 $(8,8)$ absorbing set errors obtained from the $(3,2)$-bit SP-MS decoder at an SNR of 4.9 dB, the message biasing can only correct 92 $(8,8)$ absorbing set errors, *i.e.* 67.26% of the $(8,8)$ absorbing set errors are not corrected. Since more than 60% of all the $(8,8)$ absorbing set errors are not corrected, an error floor emerges at a FER level of 10^{-8}, this makes the $(3,2)$-bit SP-MS decoder unacceptable for the IEEE 802.3 ETHERNET code.

The $(3,2)$-bit SP-MS decoder is a very special case because the message alphabet $\mathcal{A}_S = \{-1,-0,+0,+1\}$ and the decoder input alphabet $\mathcal{A}_L = \{-3,-2,-1,-0,+0,+1,+2,+3\}$ are very small. Fig. 5.17 shows the LLR distribution of the $(8,8)$ fully absorbing sets for the $(3,2)$-bit SP-MS decoder. The results are obtained considering the all-zero codeword sent over the BI-AWGN channel. We can see that the quantized LLR associated with the bits in the set D ($|D| = 8$) are in majority -3, -2, and -1. When message biasing is used, the maximum magnitude of *strong* can only be $|strong| = N_q = 1$, while the minimum magnitude of *weak* can only be $|weak| = 0$. With these values, an erroneously estimated bit could not be corrected if the associated LLR is very large (-3 or -2). It is for this reason that message biasing can not correct all the $(8,8)$ absorbing set errors.

Fig. 5.18 depicts the behavior of the APPs ($\boldsymbol{\gamma}$) for an $(8,8)$ absorbing set when post-processing (message biasing) is enabled in the $(3,2)$-bit SP-MS decoder. We can see the 8 APPs associated with the 8 wrong bits in iteration $L_{pp}-1$. In iteration L_{pp}, the message biasing changes the APP values as is shown in Fig. 5.18b. In iteration $L_{pp}+1$, two bits of the $(8,8)$ absorbing set are corrected because the two APPs associated with these two bits are positive. While in iteration $L_{pp}+2$, one bit corrected in iteration $L_{pp}+1$ becomes an incorrect bit again. Finally, after 4 iterations since the post-processing started, the same $(8,8)$ absorbing set is obtained again as is shown in Fig. 5.18e.

In order that the $(3,2)$-bit SP-MS decoder meets with the performance requirement of the ETHERNET standard, a new post-processing algorithm is introduced based on message biasing. The proposed post-processing algorithm to correct the $(8,8)$ absorbing set errors of the $(q_{ch}, q = q_{ch}-1)$-bit SP-MS decoder is described below:

1. [**SP-MS decoder**]

   Decode a codeword with the SP-MS decoder for a fixed number of iterations. If the syndrome vector is not all-zero, continue with the next step.

2. [**Post-processing**]

   (a) [**Only in a single iteration**]

   i. Message biasing: map all messages from CNs in $O(D)$ to VNs in $N(D)$ to $|strong| = N_q$, and map all messages from CNs in $S(D)$ to VNs in $N(D)$ to $|weak| = 0$.

   ii. Compute the APP: for each VN v_n in $N(D)$, the APP is computed from the channel observation I_n and from all c-to-v messages whose magnitudes are $|strong| = N_q$ or $|weak| = 0$, *i.e.* $\gamma_n^* = \mathcal{T}_{ch}(I_n) + \mathcal{T}(strong) + 5 \times \mathcal{T}(weak)$.

iii. Compute a new LLR: for each VN v_n, a new LLR is computed if the APP γ_n^* and the quantized LLR I_n satisfy the inequalities $A_1^* \leq |\gamma_n^*| \leq A_2^*$ ($A_1 \leq |\gamma_n| \leq A_2$) and $L_1 \leq |I_n| \leq L_2$, respectively. The magnitude of the new LLR is always equal to 0, and its sign is obtained flipping the sign of I_n.

(b) [**For a few more iterations**]

The decoding process continues with the SP-MS using the new LLRs until the syndrome vector is all-zero or a maximum number of iterations is reached.

We determine by simulation that the optimal values used in the inequalities are $A_1^* = 4$ ($A_1 = 2$), $A_2^* = 14$ ($A_2 = 7$), $L_1 = 0$, and $L_2 = 2$ for the $(3,2)$-bit SP-MS decoder. In the case of the $(4,3)$-bit SP-MS decoder, we have $A_1^* = 0$ ($A_1 = 0$), $A_2^* = 14$ ($A_2 = 7$), $L_1 = 0$, and $L_2 = 4$.

The proposed post-processing algorithm is very efficient to correct the $(8,8)$ absorbing set errors compared to the message biasing, especially for very low precision ($q_{ch} = 3, q = 2$). Table 5.11 shows the number of uncorrected $(8,8)$ absorbing set errors considering $S_{\hat{x}} < 21$. We can see that for very low precision ($q_{ch} = 3, q = 2$), the proposed post-processing corrects all the $(8,8)$ absorbing errors at SNR $= 4.9$ dB and SNR $= 5.0$ dB, while the message biasing can not correct around 68% of the $(8,8)$ absorbing set errors. For the case of the $(4,3)$-bit SP-MS decoder, the message biasing can also be used as post-processing, but a slight amount of the $(8,8)$ absorbing set errors can not be corrected using message biasing, as is shown in Table 5.11.

Using the proposed post-processing, Fig. 5.19 shows the behavior of the APPs to correct the $(8,8)$ absorbing set used in Fig. 5.18. In iteration $L_{pp} - 1$, we observe the 8 APPs associated with the 8 wrong bits. In iteration L_{pp}, all messages from CNs in $O(D)$ to VNs in $N(D)$ are set to *strong*, and all messages from CNs in $S(D)$ to VNs in $N(D)$ are set to *weak*. Hence, the APP γ_n is computed for each VN v_n, as is shown in Fig.

5.19b. The LLR I_n is mapped to $(-\text{sign}, 0)$ if $2 \leq |\gamma_n| \leq 7$ and $0 \leq |I_n| \leq 2$. In iteration $L_{pp}+1$, the first two bits of the $(8,8)$ absorbing set are corrected. Also, the APPs associated with the 6 wrong bits are close to 0. With 3 iterations of post-processing, five bits are already corrected. Finally, all bits of the $(8,8)$ absorbing set are corrected after 4 iterations of the proposed post-processing algorithm, as is shown in Fig. 5.19e.

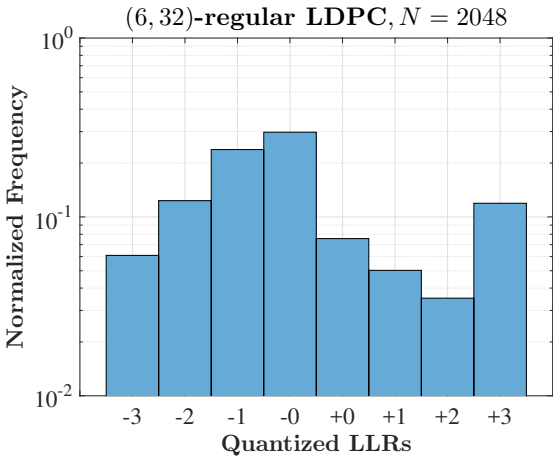
Table 5.11 Number of uncorrected $(8,8)$ absorbing set errors by the proposed post-processing and by message biasing (11 iterations of the $(3,2)$-bit SP-MS + 9 iterations of post-processing, and 9 iterations of the $(4,3)$-bit SP-MS + 6 iterations of post-processing).

IEEE 802.3 ETHERNET code, (q_{ch}, q)-bit SP-MS Decoder							
		Before post-processing		After post-processing			
Precision (q_{ch}, q)	SNR (dB)	run the SP-MS		proposed post-proc.		message biasing	
		Number of $(8,8)$ sets	Average $S_{\hat{x}}$	Number of $(8,8)$ sets	Average $S_{\hat{x}}$	Number of $(8,8)$ sets	Average $S_{\hat{x}}$
$(3,2)$	4.75	193	8.11	**3**	10	130	8.12
	4.8	257	8.12	**3**	10	162	8.27
	4.9	281	8.05	**0**	0	189	8.29
	5.0	88	8.09	**0**	0	61	8.23
$(4,3)$	4.75	369	8.10	**9**	8.67	26	9.62
	4.8	384	8.13	**11**	14.36	24	10.75
	4.9	88	8.05	**2**	8	2	8
	5.0	118	8	**2**	11	8	9

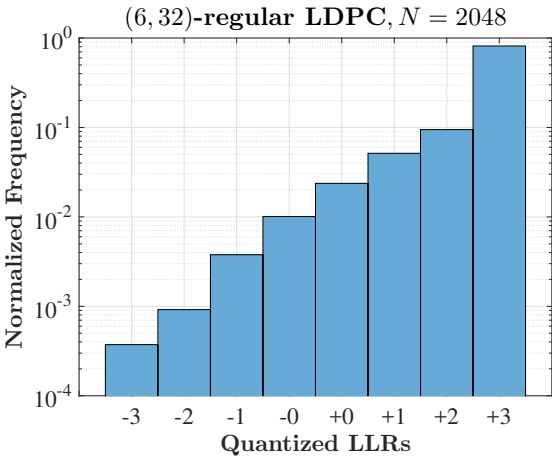
5.6.2 Post-Processing Architecture

The proposed post-processing architecture is a full parallel architecture which performs a decoding iteration per clock cycle. This architecture only needs to know the unsatisfied CNs in order to correct the $(8,8)$ absorbing set errors. Fig. 5.20 shows the main components and main signals of the proposed architecture, the input and output signals behave the same as the signals of the architecture shown in Fig. 5.12.

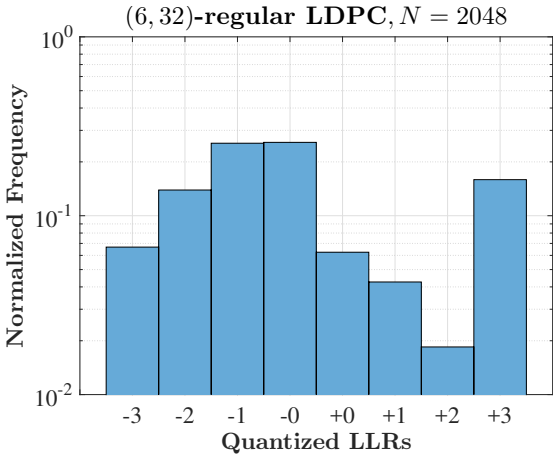
The behaviour of the main signals of the proposed post-processing architecture is depicted in Fig. 5.21. We consider 9 iterations of SP-MS and 6 iterations of post-processing with a maximum of 15 iterations. Also, the number of unsatisfied CNs is less



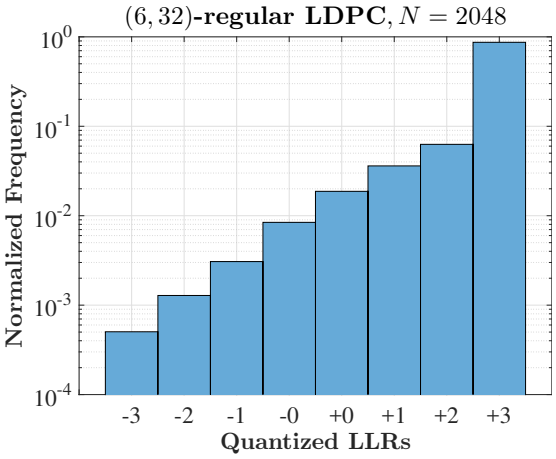
(a) LLR distribution of the bits in the set D at SNR $= 4.9$ dB.



(b) LLR distribution of the bits in the set $N(D)\backslash D$ at SNR $= 4.9$ dB.

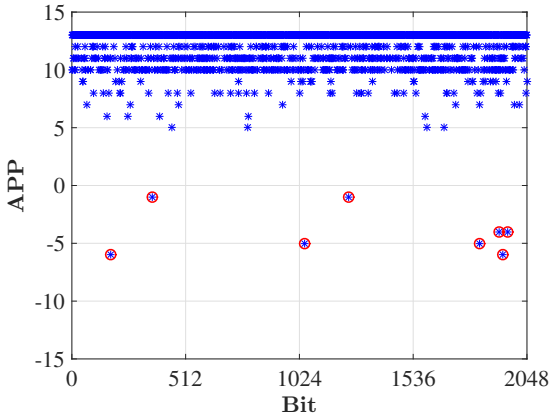


(c) LLR distribution of the bits in the set D at SNR $= 5.0$ dB.

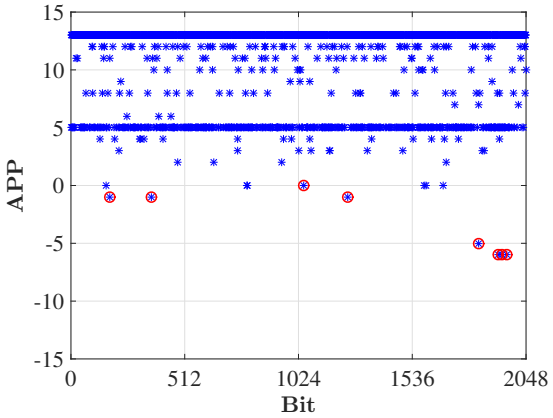


(d) LLR distribution of the bits in the set $N(D)\backslash D$ at SNR $= 5.0$ dB.

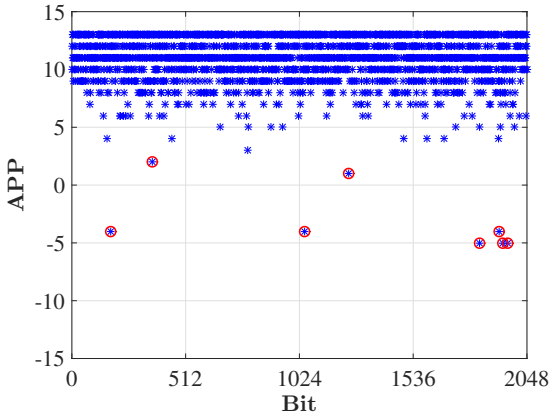
Fig. 5.17 LLR distribution of the $(8,8)$ fully absorbing sets obtained using the $(3,2)$-bit SP-MS decoder. The results provided are for 281 $(8,8)$ absorbing set at SNR $= 4.9$ dB, and for 88 $(8,8)$ absorbing set at SNR $= 5.0$ dB.



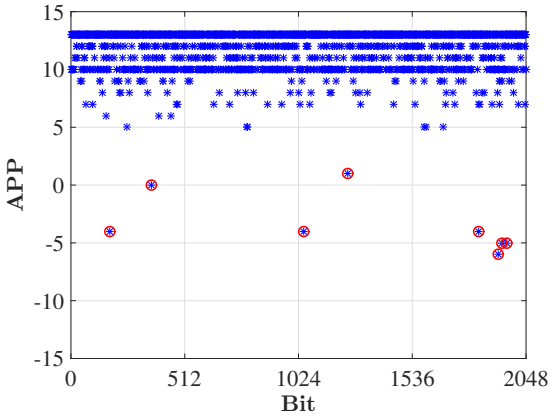
(a) $L_{pp}-1$.



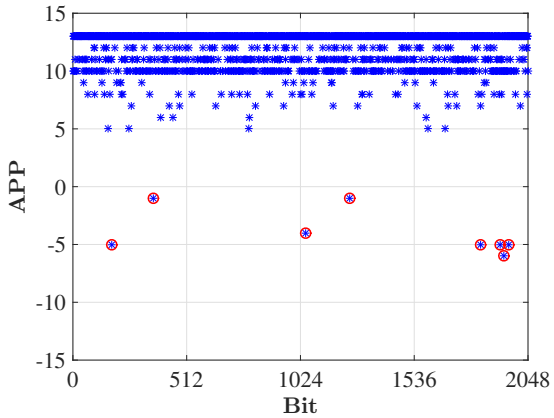
(b) L_{pp}.



(c) $L_{pp}+1$.

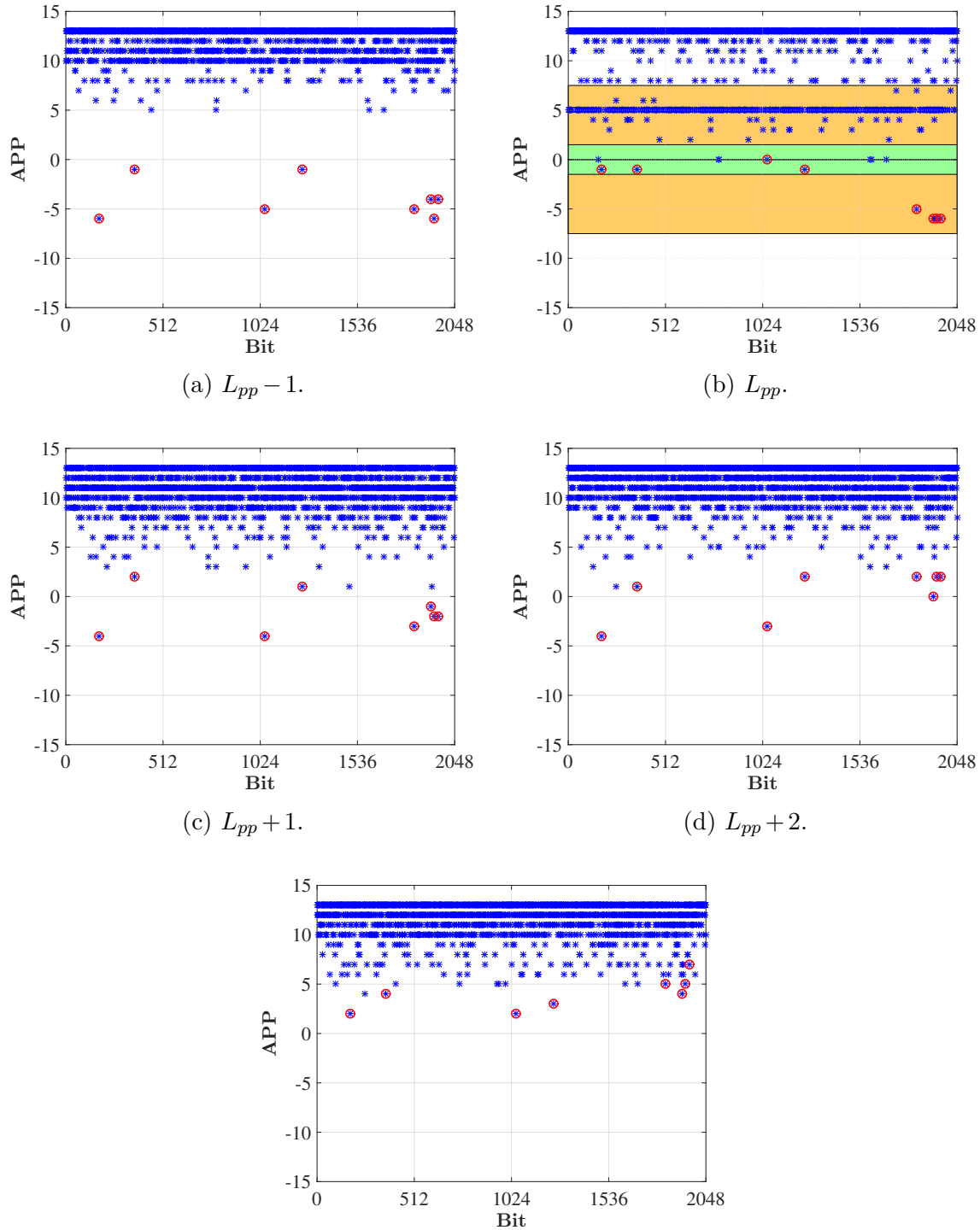


(d) $L_{pp}+2$.



(e) $L_{pp}+3$.

Fig. 5.18 A posteriori probability of the $(3,2)$-bit SP-MS decoder for an $(8,8)$ absorbing set. Behavior of the APPs in each iteration after enabling the post-processing (message biasing).



(a) $L_{pp}-1$. (b) L_{pp}.

(c) $L_{pp}+1$. (d) $L_{pp}+2$.

(e) $L_{pp}+3$.

Fig. 5.19 A posteriori probability of the $(3,2)$-bit SP-MS decoder for an $(8,8)$ absorbing set. Behavior of the APPs in each iteration after enabling the new post-processing.

than 21 ($S_{\hat{x}} < 21$). The post-processing starts when the iteration number ℓ equals L_{pp}. Only when the signal *post* is equal to 1, *i.e.* when the number of unsatisfied CNs is less than 21 ($UnCN = 1$) and when $\ell = L_{pp}$ ($Itpp = 1$), the messages from CNs in $O(D)$ to VNs in $N(D)$ are mapped to *strong*, the messages from CNs in $S(D)$ to VNs in $N(D)$ are mapped to *weak*, and the new LLRs are computed. Similar to the architecture of Fig. 5.12, the architecture for precision $(q_{ch}, q = q_{ch} - 1)$ is composed of five main components.

Count of Iterations

Like the case of precision $(q_{ch}, q = q_{ch})$, this component counts one iteration per each clock cycle when $initialize = 0$. Only when the count of iterations equals L_{pp}, the output signal *Itpp* is set to 1, otherwise *Itpp* is always 0. Similarly, only when the maximum number of iteration is reached, $maxIter = 1$, otherwise $maxIter = 0$.

Compute Syndrome

This component is an asynchronous circuit that computes the syndrome vector $\boldsymbol{S}$ in each decoding iteration using the input signal *decision* ($\hat{\boldsymbol{x}}$). The decoding process ends when a valid codeword is obtained ($\boldsymbol{S} = 0$), in this case the output signals are $EndIter = 1$ and $EndSyndr = 0$. In the case that $\boldsymbol{S} \neq 0$, the decoding process continues ($EndIter = 0$) until the maximum number of iterations is reached ($maxIter = 1$).

One should note that the output signal *SyndrVect* ($\boldsymbol{S}$) carries information about how many CNs are unsatisfied CNs. Also, unlike the architecture of Section 5.5 where the CNUs compute the signal *SyndrVect*, in this case the **Compute Syndrome** component is responsible for calculating the signal *SyndrVect*.

CNU Component

Similar to the architecture of Section 5.5, the proposed architecture consists of $M = 384$ CNU components. Each CNU c_m sends information about whether a CN is a satisfied CN or an unsatisfied CN, specifically, the most significant bit ($sign$) of the c-to-v messages ($m^*_{c_m \to v}$) carries that information. The value of $sign$ is computed as $sign = \prod_{v_n \in \mathcal{V}(c_m)} \text{sign}\,(m_{v_n \to c_m})$, obtaining $sign = +1$ (in binary representation $sign = 0$) for a satisfied CN or $sign = -1$ (in binary representation $sign = 1$) for an unsatisfied CN. The CNU c_m also computes min_1 and min_2, with $min_2 \geq min_1$. Using those values, the function W^* at a CN c_m is given by $W^*(a, sign, min_1, min_2) = (sign, min_2)$ if $|a| = min_1$, otherwise $W(a, sign, min_1, min_2) = (sign, min_1)$. The c-to-v messages $m^*_{c_m \to v}$, with $v \in \mathcal{V}(c_m)$, are computed using $m^*_{c_m \to v} = W(m_{v \to c_m}, sign, min_1, min_2)$.

VNU Component

The proposed architecture uses $N = 2048$ VNU components. The VNU component for precision $(q_{ch}, q = q_{ch} - 1)$ is very similar to the VNU component of the architecture presented in Section 5.5. Let us explain the two differences: *(i)* in the initialization stage, *i.e.* $initialize = 1$, the v-to-c messages ($m_{v_n \to c}$) are obtained by saturating the quantized LLR I_n with the function $\mathcal{S}(I_n, N_q)$, and *(ii)* during the decoding process, *i.e.* $initialize = 0$, and when the signal $post = 1$, a new LLR is computed if the APP γ^*_n and the quantized LLR I_n satisfy the inequalities $A^*_1 \leq |\gamma^*_n| \leq A^*_2$ and $L_1 \leq |I_n| \leq L_2$, respectively.

Unsatisfied CNs

This component is equal to the **Usatisfied CNs** component of the architecture presented in Section 5.5, *i.e.* this component produces the signal $UnCN = 1$ if and only if the number of unsatisfied CNs is less than τ ($S_{\hat{x}} < \tau$).

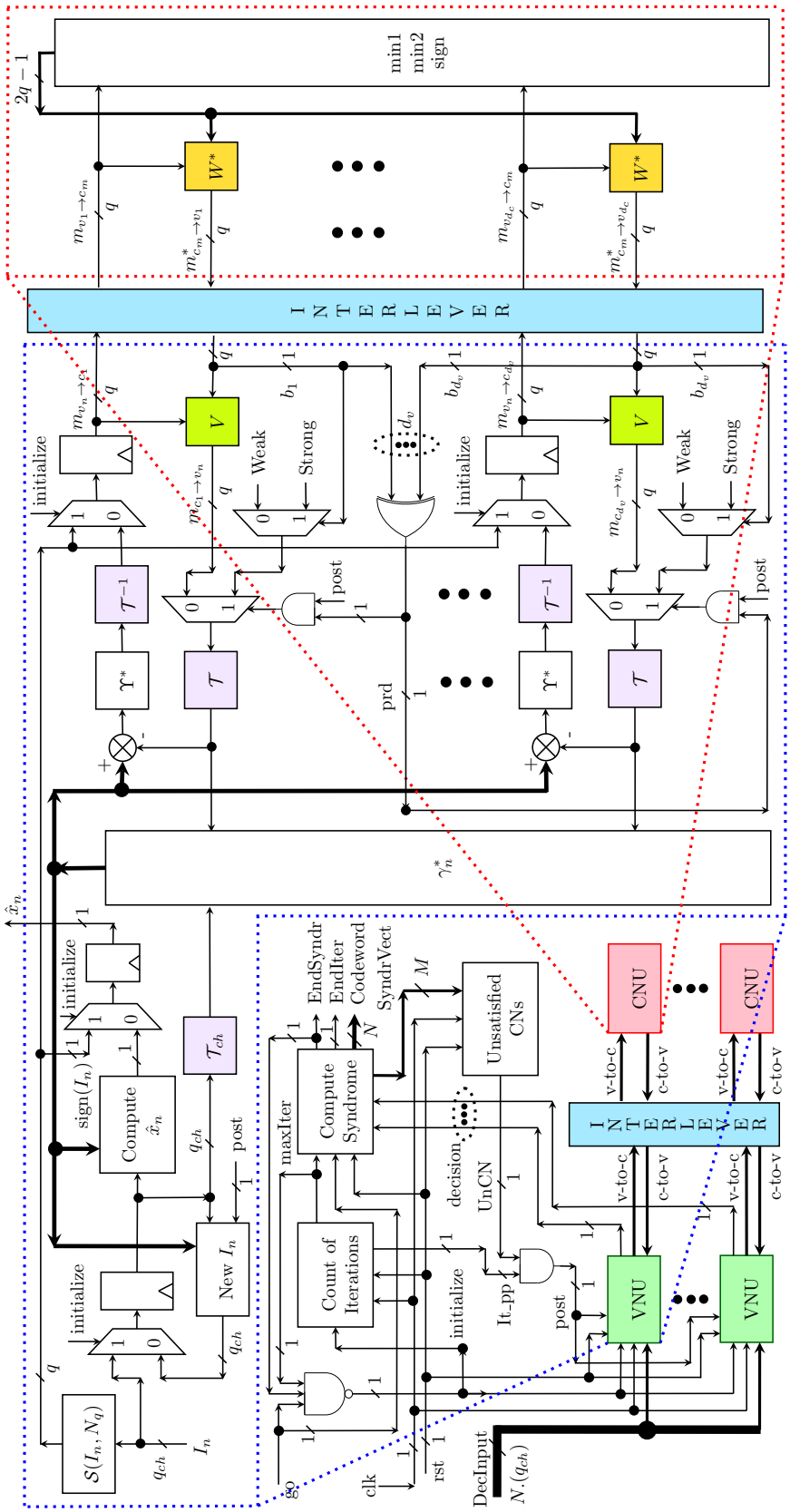


Fig. 5.20 Post-processing architecture of the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoders for the IEEE 802.3 ETHERNET code.

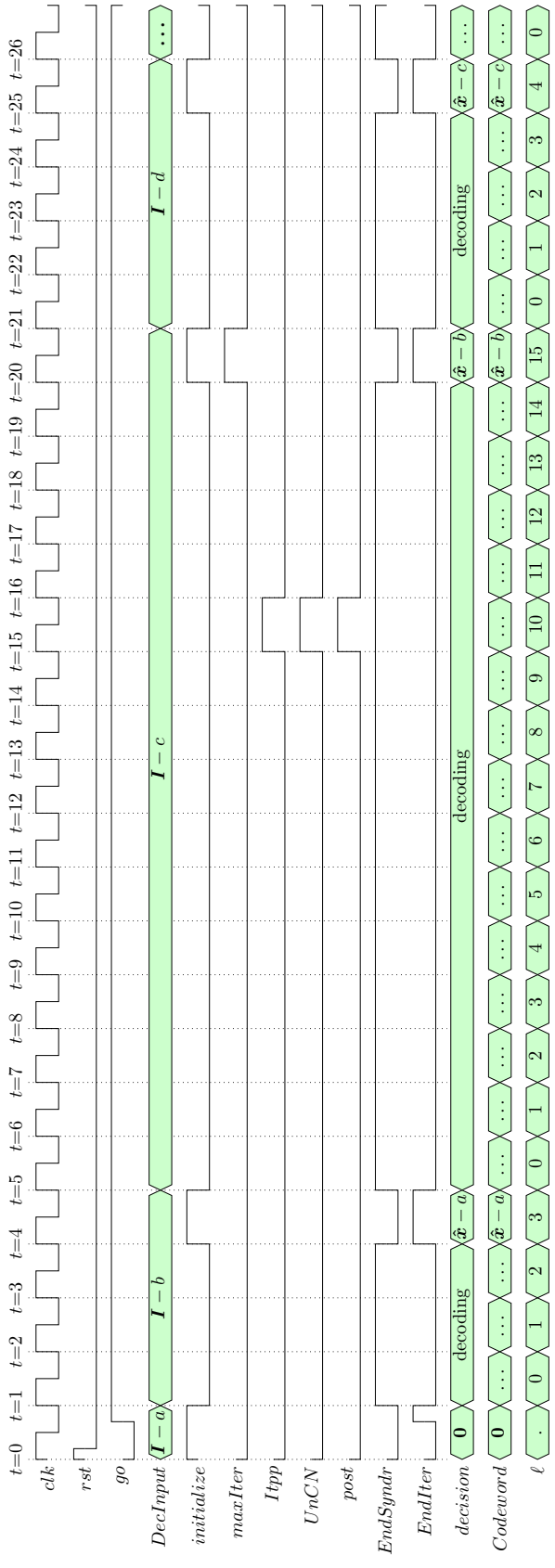
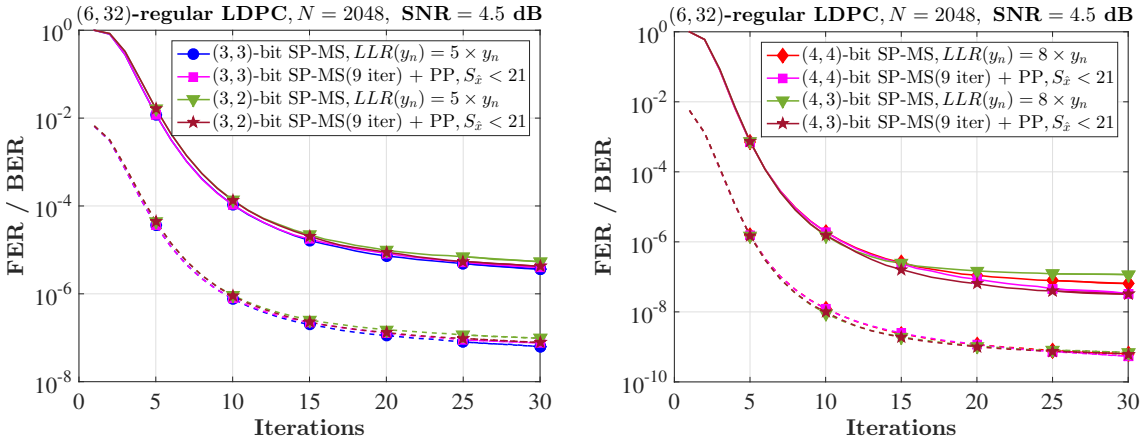


Fig. 5.21 Behavior of the main signals when the post-processing is enabled to correct an (8,8) absorbing set. An (8,8) absorbing set is corrected in the 15^{th} iteration.

5.6.3 Performance results

In this section, we present the FER performance of the proposed architecture for $(q_{ch}=3, q=2)$ and $(q_{ch}=4, q=3)$.

Fig. 5.22 shows the FER convergence results at $E_b/N_0 = 4.5$ dB for the SP-MS, and SP-MS + PP considering $S_{\hat{x}} < 21$. We can see that the FER/BER performance of the SP-MS + PP is not degraded when the number of unsatisfied CNs is controlled considering $S_{\hat{x}} < 21$. Therefore this confirms that the **Unsatisfied CNs** component helps the SP-MS decoders to avoid decoding performance degradation in the waterfall region when using post-processing.



(a) Precision $q_{ch} = 3$. (b) Precision $q_{ch} = 4$.

Fig. 5.22 FER (solid lines) and BER (dashed lines) convergence comparison on the IEEE 802.3 ETHERNET code at $E_b/N_0 = 4.5$ dB using a maximum of 9 iterations of $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS and 21 iterations as maximum of post-processing.

Table 5.12 lists the average number of iterations to correct the $(8,8)$ absorbing set errors. Similar to the message biasing, the proposed post-processing requires 4 and 3 iterations for precision $(q_{ch}=3, q=2)$ and $(q_{ch}=4, q=3)$, respectively.

The architecture of Fig. 5.20 that implements our post-processing algorithm is implemented in an FPGA for the case of very low precision $(q_{ch}=3, q=2)$. Two decoders are emulated to evaluate the performance of the architecture: *(i)* the first one using a

Table 5.12 Average number of iterations for post-processing using $(L_{pp}-1)$ iterations as maximum of the $(q_{ch}, q=q_{ch}-1)$-bit SP-MS + $(L_{max}-L_{pp}+1)$ iterations as maximum of post-processing.

Average number of iterations for the proposed post-processing				
SNR (dB)	$(3,2)$-bit SP-MS, $L_{max}=19$		$(4,3)$-bit SP-MS, $L_{max}=15$	
	$L_{pp}-1=9$	$L_{pp}-1=11$	$L_{pp}-1=9$	$L_{pp}-1=10$
4.75	3.86	3.82	3.36	3.29
4.8	3.85	3.76	3.29	3.25
4.9	3.79	3.76	3.19	3.16
5.0	3.89	3.78	3.27	3.27

maximum of 9 iterations of the $(3,2)$-bit SP-MS + 10 iterations as maximum of post-processing, and the second one using a maximum of 11 iterations of the $(3,2)$-bit SP-MS + 8 iterations as maximum of post-processing. Fig. 5.23 shows the emulations results, one can see that the error-floor is lowered and excelled error correction performance is obtained below the FER of 10^{-10}, which renders the proposed architecture of the $(3,2)$-bit SP-MS decoder acceptable for the IEEE 802.3 ETHERNET code. Also, the emulations results confirm that the error correction performance is not degraded in the waterfall region.

Fig. 5.24 shows the simulation results for precision $(q_{ch}=4, q=3)$ using a maximum of 9 iterations of the $(4,3)$-bit SP-MS + 6 iterations as maximum of post-processing. One confirms again that the FER performance is not degraded in the waterfall region, and the error floor can be lowered below the FER of 10^{-10}. The point at SNR level of 5.0 dB is obtained by dividing the number of error frames collected by the total number of frames. Similarly for the point at SNR level of 4.75 dB.

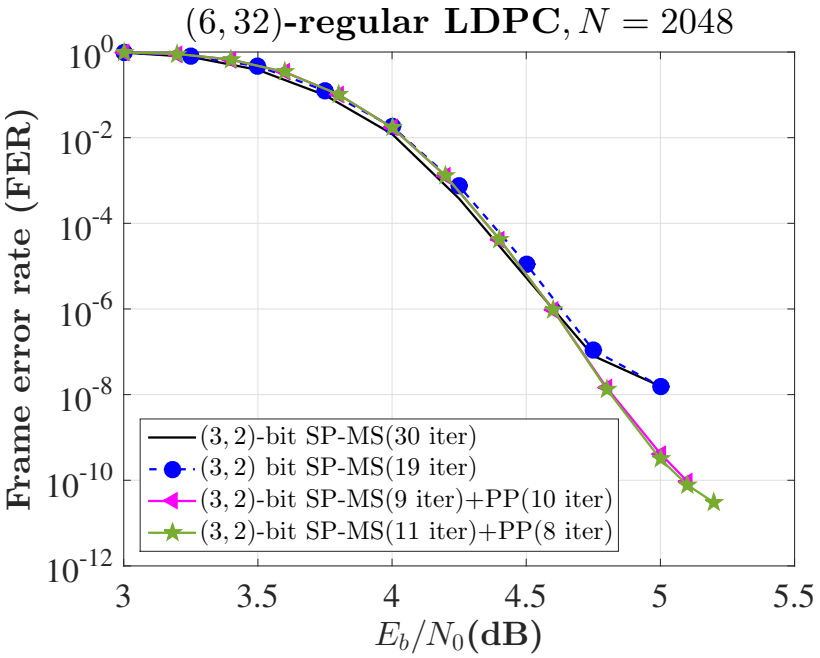


Fig. 5.23 FER performance of the $(3,2)$-bit SP-MS decoder with post-processing obtained by FPGA emulation for the IEEE 802.3 LDPC code.

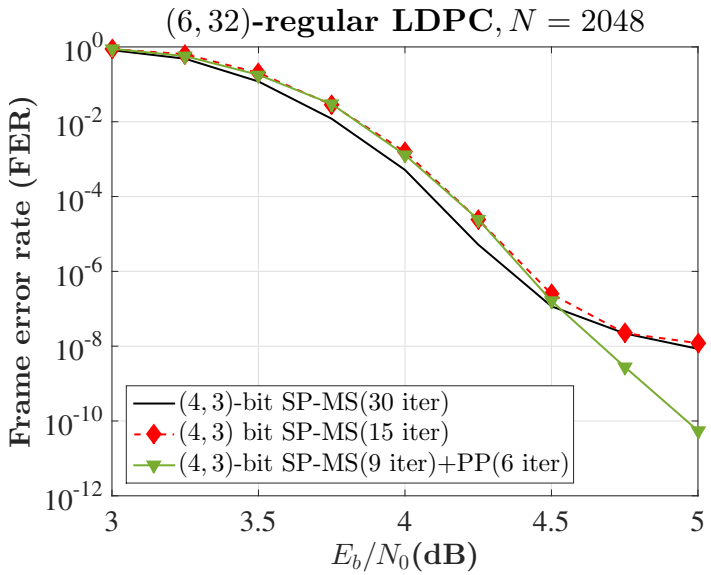


Fig. 5.24 FER performance of $(4,3)$-bit SP-MS decoder with post-processing for the IEEE 802.3 ETHERNET code..

5.7 Implementation Results of SP-MS Decoder with Post-Processing

In this section we present the implementation results of the post-processing architectures considering $S_{\hat{x}} < 21$ for all decoders. Also, We consider 19 iterations as maximum for precision $(3,2)$, and 15 iterations as maximum for precision $(3,3)$, $(4,3)$, and $(4,4)$.

5.7.1 Synthesis results on FPGA

This section reports the synthesis result of the proposed post-processing architectures on the Xilinx XC7V2000T-1FLG1925 FPGA chip for the IEEE 802. LDPC code.

The FPGA resource utilization of the $(q_{ch}, q = q_{ch})$-bit SP-MS and $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoders is listed in Table 5.13 for the $(d_v = 6, d_c = 32)$-regular LDPC code. Also, the maximum frequency that each decoder can reach is listed. From the results obtained, one can see that the $(q_{ch} = 3, q = 2)$-bit SP-MS decoder reaches the maximum frequency among the 4 SP-MS decoders. For the precision $q_{ch} = 4$, the maximum frequency of the $(q_{ch} = 4, q = 3)$-bit SP-MS decoder is higher compared to the maximum frequency of the $(q_{ch} = 4, q = 4)$-bit SP-MS decoder.

Comparing the resources used by the decoders on the FPGA, the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoders use less resources than the $(q_{ch}, q = q_{ch})$-bit SP-MS decoders. We can clearly see a large savings of FPGA resources: around 27% of slice registers and 31% of slice LUTs for the precision $(q_{ch} = 3, q = 2)$, and 20% of slice registers and 37% of slice LUTs for the precision $(q_{ch} = 4, q = 3)$.

Table 5.13 Synthesis results on FPGA of the SP-MS decoders with post-processing for the IEEE 802.3 ETHERNET code.

IEEE 802.3 ETHERNET code, SP-MS decoders									
(q_{ch}, q)	Max. Freq. (MHz)	Number of slice registers		Number of slice LUTs		Number of fully used LUT-FF pairs		Throughput with 15 iter (Gbps)
(3,3)	99.975	45064	(0.0%)	319809	(0.0%)	38920	(0.0%)	13.6499
(3,2)	103.659	32777	(−27.27%)	218923	(−31.55%)	32777	(−15.78%)	14.1529
(4,4)	87.987	59400	(0.0%)	587153	(0.0%)	51208	(0.0%)	12.0132
(4,3)	99.440	47112	(−20.69%)	366031	(−37.66%)	47112	(−8.0%)	13.5769

5.7.2 ASIC synthesis results

This section reports the ASIC synthesis results of the four SP-MS decoders with post-processing using the 28 nm FDSOI library at 0.9 V supply voltage and temperature of 125 °C. The timing constraints used to perform the synthesis of the post-processing architectures are the same as Section 5.3.

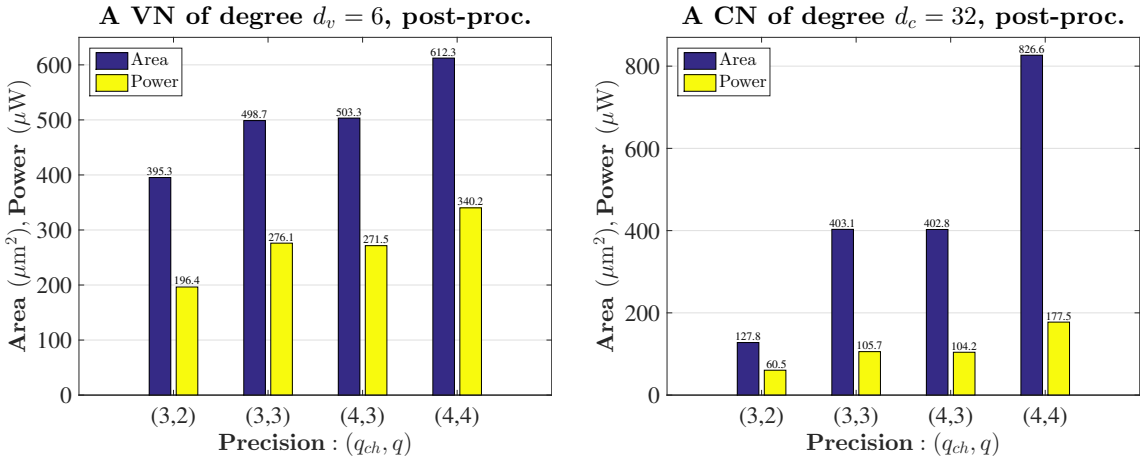
Table 5.14 ASIC synthesis results using the 28 nm FDSOI library for only one VNU of degree $d_v = 6$ and only one CNU of degree $d_c = 32$, the post-processing algorithm is implemented in the VNU.

IEEE 802.3 ETHERNET code, SP-MS decoders									
Precision	Variable-Node					Check-Node			
(q_{ch}, q)	Freq. (MHz)	Area (μm^2)		Power (μW)		Area (μm^2)		Power (μW)	
(3,3)	600	498.74	(0.0%)	276.1	(0.0%)	403.10	(0.0%)	105.7	(0.0%)
(3,2)		395.27	(−20.75%)	196.4	(−28.87%)	127.78	(−68.30%)	60.537	(−42.73%)
(4,4)	600	612.33	(0.0%)	340.2	(0.0%)	826.60	(0.0%)	177.5	(0.0%)
(4,3)		503.31	(−17.80%)	271.5	(−20.19%)	402.77	(−51.27%)	104.2	(−41.30%)

We list the area and power of an isolated VNU and an isolated CNU of the four SP-MS decoders with post-processing in Table 5.14. One can note that a VNU/CNU of the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoder helps to reduce the area used and the power consumed by a VNU/CNU of the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder. When the precision of the messages goes from $q = q_{ch} = 3$ bits to $q = q_{ch} - 1 = 2$ bits, we can see that *(i)* the CNU reduces an area of 403.10μm^2 by 68.30% to 127.74μm^2, *i.e.* a saving of

around 70% is achieved, and *(ii)* the VNU reduces an area of 498.74μm^2 by 20.75% to 395.27μm^2. In the case that the precision goes from $q = q_{ch} = 4$ bits to $q = q_{ch} - 1 = 3$ bits, *(i)* the CNU helps to reduce an area of 826.60μm^2 by 51.27% to 402.77μm^2 (saving approximately 51% of the area), and *(ii)* the VNU reduces the area by 17.80%, *i.e.*, from 612.33μm^2 to 503.31μm^2. When comparing the power consumption, one can see a power consumption saving of 28.87% (from 276.1μW to 196.4μW) for the VNU and very low precision ($q_{ch} = 3, q = 2$). More results are shown in the Table 5.14.

Like the SP-MS decoders without post-processing, for the SP-MS decoders with post-processing, we can conclude that using one bit less to represent the messages, the VNU and (especially) the CNU can greatly reduce the area used to implement the $(q_{ch}, q = q_{ch} - 1)$-bit SP-MS decoder compared to the area used by the $(q_{ch}, q = q_{ch})$-bit SP-MS decoder. Fig. 5.25 illustrates the are used for the power consumption by a single VNU/CNU of the SP-MS decoders with post-processing.



(a) A VNU of degree $d_v = 6$.

(b) A CNU of degree $d_c = 32$.

Fig. 5.25 The area utilization and power consumption by a single VNU/CNU of the IEEE 802.3 ETHERNET code for $q \in \{2, 3, 4\}$ bits of precision.

Let us now present the ASIC synthesis results for the fully parallel post-processing architecture. Table 5.15 shows the area utilization and power consumption for the $d_v = 6$ RS-LDPC decoders, also, the decoding throughput and the energy per decoded

bit are listed. We can observe that the $(q_{ch}=3, q=2)$-bit SP-MS decoder reduces an area of 1.185 mm^2 by 25.93% to 0.878 mm^2, *i.e.* a saving of around 26% is achieved, and the $(q_{ch}=4, q=3)$-bit SP-MS decoder reduces an area of 1.619 mm^2 by 24.12% to 1.229 mm^2 (saving approximately 24% of the area). When comparing the power consumption, one can see a power consumption saving of 22.59% (from 307.41 mW to 237.98 mW) for very low precision $(q_{ch}=3, q=2)$, while for precision $(q_{ch}=4, q=3)$, the reduction of power consumption is 20.57%, *i.e.*, from 383.25 mW to 304.41 mW.

For a frequency of 500 MHz, the minimum throughput reached is 68.27 Gbps for precision $(q_{ch}=3, q=2)$. While for precision $(q_{ch}=3, q=3)$, $(q_{ch}=4, q=3)$, and $(q_{ch}=4, q=4)$, the minimum throughput reached is 53.89 Gbps. The average number of decoding iterations in the SP-MS decoder at $E_b/N_0=4.75$ dB is around 3, hence, the decoding throughput reached is around 341.33 Gbps.

From the analysis of the SP-MS decoders with post-processing using $q_{ch} \in \{3,4\}$ bits LLRs and $q \in \{2,3,4\}$ bits messages, we can conclude that: *(i)* the $(3,2)$-bit SP-MS decoder is the smallest decoder, *(ii)* the $(q_{ch}, q=q_{ch}-1)$-bit SP-MS decoder uses less area and consumes less power than the $(q_{ch}, q=q_{ch})$-bit SP-MS decoder, and *(iii)* the $(4,3)$-bit SP-MS decoder occupies an area slightly greater than the $(3,3)$-bit SP-MS decoder (both decoders have the same number of wires).

Table 5.15 ASIC synthesis results using the 28 nm FDSOI library for only one VNU of degree $d_v=6$ and only one CNU of degree $d_c=32$.

IEEE 802.3 ETHERNET code, SP-MS decoders									
Precision (q_{ch}, q)	Freq. (MHz)	Area (mm^2)		Power (mW)		EpB (pJ/bit)		Throughput (Gbps)
$(3,3)$	500	1.185272	(0.0%)	307.41	(0.0%)	4.5029	(0.0%)	68.27
$(3,2)$		0.877948	(−25.93%)	237.98	(−22.59%)	4.4160	(−1.93%)	53.89
$(4,4)$	500	1.619418	(0.0%)	383.25	(0.0%)	5.6137	(0.0%)	68.27
$(4,3)$		1.228852	(−24.12%)	304.41	(−20.57%)	4.4589	(−20.57%)	68.27

Fig. 5.26 illustrates the are used for the (q_{ch}, q)-bit SP-MS decoders. We can see that the $(3,2)$-bit SP-MS decoder is the smallest decoder, the $(4,4)$-bit SP-MS decoder

is the biggest decoder, and the $(4,3)$-bit SP-MS decoder occupies an area slightly greater than the $(3,3)$-bit SP-MS decoder.

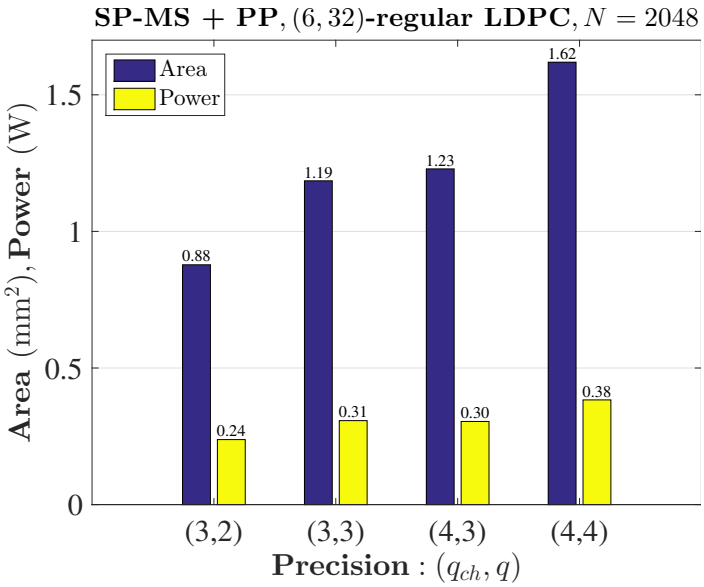


Fig. 5.26 The area utilization and power consumption of the (q_{ch}, q)-bit SP-MS decoders with post-processing.

Comparing the results of Table 5.8 and Table 5.15, one can see that for a fixed precision (q_{ch}, q), the post-processing increases the core area of SP-MS decoders. For very low precision $(q_{ch} = 3, q = 2)$, the area of 0.774 mm^2 is increased by 13.38% to 0.878 mm^2, while for precision $(q_{ch} = 4, q = 3)$, the area is increased by 10.20%, *i.e.*, from 1.115 mm^2 to 1.229 mm^2.

5.7.3 Place and Route results

The place and route results of SP-MS decoders presented in this section are preliminary results. We implement two decoders using $S_{\hat{x}} < 21$: *(i)* the first decoder considers a maximum of 9 iterations of $(3,3)$-bit SP-MS + 6 iterations as maximum of post-processing (message biasing), *(i)* the second decoder considers a maximum of 11 iterations of $(3,2)$-bit SP-MS + 8 iterations as maximum of post-processing (proposed algorithm of Section 5.6).

The place and route of the (3,3)-bit SP-MS decoder is done in 28 nm FDSOI obtaining 2.56 mm^2, a density of 40%, and clock frequency of 500 MHz. In the worst case that corresponds to 15 iterations, the decoding throughput is 68.27 Gbit/s and the guaranteed hardware efficiency is 26.67 Gbit/s/mm^2, see Table 5.16. A decoding throughput of 384.96 Gbit/s and a hardware efficiency of 150.38 Gbit/s/mm^2 are achieved using 2.66 iterations in average at an SNR level of 5.5 dB.

Table 5.16 Place and Route results of the $(3,2)$-bit SP-MS and the $(3,3)$-bit SP-MS decoders using post-processing.

Place and Route		
Decoder	$(3,2)$-bit SP-MS	$(3,3)$-bit SP-MS
Frequency (MHz)	421	**500**
Core area (mm^2)	**1.76**	2.56
Density	**60**%	40%
Throughput (Gbps)†	45.38	**68.27**
Hardware Eff. (Gbps/mm^2)†	25.78	**26.67**

† The worst case.

The place and route of the (3,2)-bit SP-MS decoder is also done 28 nm FDSOI, we obtain an area of 1.76 mm^2, a density of 60%, and a clock frequency of 421 MHz. In this case, using 19 iterations, the decoding throughput is 45.38 Gbit/s and the guaranteed hardware efficiency is 25.78 Gbit/s/mm^2. At a SNR level of 5.5 dB, where 2.70 iterations is used in average, the decoding throughput is 319.34 Gbit/s and the hardware efficiency is 181.44 Gbit/s/mm^2, as is shown in Table 5.17. When comparing the area used by the two SP-MS decoders, the precision ($q_{ch} = 3, q = 2$) reduces an area of 2.56 mm^2 by 31.25% to 1.76 mm^2, this result confirms the conclusions obtained in Section 5.7.2.

From the results of density, we can see that the (3,2)-bit SP-MS decoder helps alleviate the routing congestion problem of the (3,3)-bit SP-MS decoder. When comparing the hardware efficiency of both decoders at high SNR levels, where the average number

iterations is almost the same for both decoders, we can conclude that the $(3,2)$-bit SP-MS is the best decoder. It must be taken into account that at high SNR levels, the use of the post-processing is infrequent and most codewords are decoded using very few iterations.

Table 5.17 Average number of iterations, decoding throughput, and hardware efficiency of the $(3,2)$-bit SP-MS and the $(3,3)$-bit SP-MS decoders using post-processing.

Decoder	SNR (dB)	3.0	3.5	4.0	4.5	5.0	5.5
11 iter of $(3,2)$-bit SP-MS + 8 iter of post-proc.	Average iterations†	13	10.91	6.74	4.27	3.26	2.70
	Throughput (Gbps)	66.32	79.03	127.92	201.92	264.48	319.34
	Hardware Eff. (Gbps/mm^2)	**37.68**	**44.90**	**72.68**	**114.72**	**150.27**	**181.44**
9 iter of $(3,3)$-bit SP-MS + 6 iter of post-proc.	Average iterations†	12.75	10.12	6.45	4.16	3.19	2.66
	Throughput (Gbps)	80.31	101.19	158.76	246.15	321.0	384.96
	Hardware Eff. (Gbps/mm^2)	31.37	39.53	62.02	96.15	125.39	150.38

† A clock cycle used for the initialization of the variable-to-check messages is considered.

The $(3,2)$-bit SP-MS and the $(3,3)$-bit SP-MS decoders were synthesized from a VHDL description using Synopsys Design Compiler and the placed and routed using Cadence Encounter Digital Implementation. The layout of the $(3,2)$-bit SP-MS decoder is shown in Fig. 5.27.

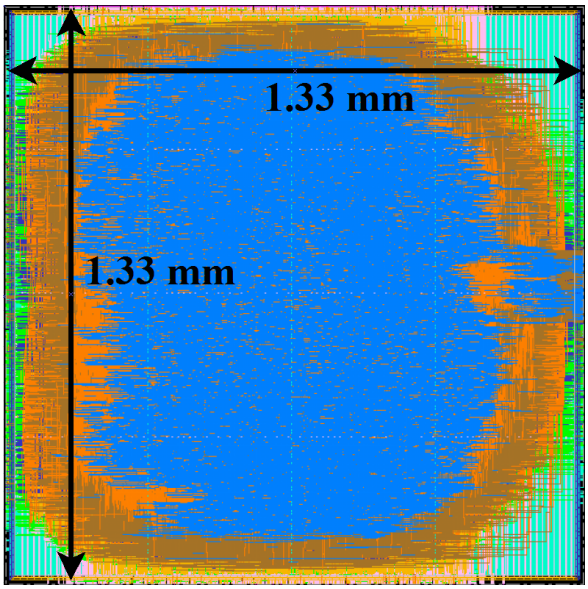


Fig. 5.27 Layout for the $(3,2)$-bit SP-MS decoder.

5.7.4 Comparison with other works

In the literature there are many implemented decoders for the IEEE 802.3 ETHERNET code, some of them are listed in Table 5.18 in order to compare them with our SP-MS decoders. Note that the maximum frequency of [54, 58] is set to 1000 MHz in 28 nm (after scaling) to have a more realistic and non-optimistic decoder. Comparing the hardware efficiency, we can see that the (3,2)-bit SP-MS is 5.0 (resp. 2.5) times more efficient than the decoder of [63] (resp. [54]).

The error correction performance of the proposed decoders are compared with the $(4,4)$-bit OMS decoder of [54] and with $(4,3)$-bit LUT-based decoder of [63]. Fig. 5.28 shows the FER performance of the SP-MS decoders. We can see that all the SP-MS decoders exhibit better FER performance than the $(4,3)$-bit LUT-based, For a FER $= 10^{-10}$ we obtain a SNR gain of 0.5 dB for the very low precision $(3,2)$, 0.52 dB for low precision $(3,3)$, and 0.56 dB for intermediate precision $(4,3)$ and the largest precision $(4,4)$. We can also observe that the $(4,4)$-bit SP-MS and the $(4,3)$-bit SP-MS decoders have slightly better FER performance than the $(4,4)$-bit OMS. Also, the SP-MS decoder using $(q_{ch} = 3, q = 2)$ bits (resp. $(q_{ch} = 3, q = 3)$ bits) of precision is 0.1 dB (resp. 0.08 dB) away from the $(4,4)$-bit OMS at a FER $= 10^{-10}$.

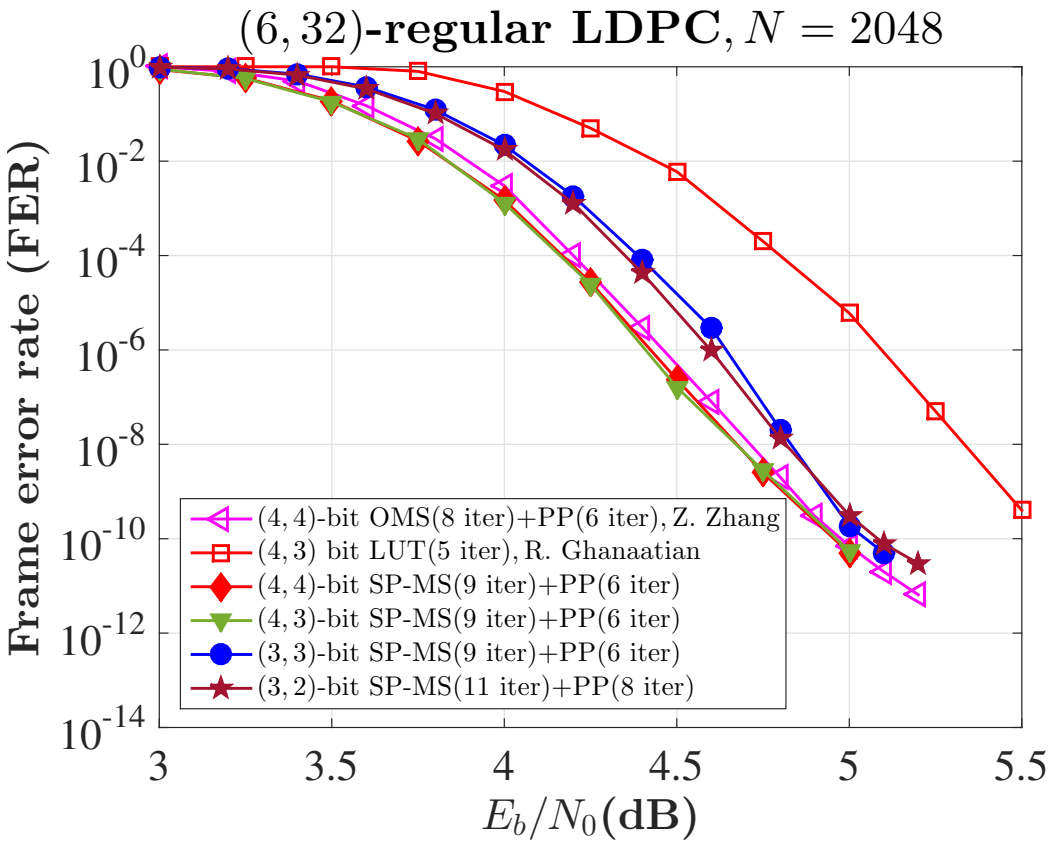


Fig. 5.28 Error correction performance of (q_{ch}, q)-bit SP-MS decoders.

Table 5.18 Implementation results for the IEEE 802.3 ETHERNET code and comparison with other works.

	This work		[63]	[54]	[55]	[56]	[57]	[58]
Technology	28 nm FDSOI		28 nm FDSOI	65 nm CMOS low-power	65 nm CMOS	90 nm CMOS	90 nm CMOS	90 nm CMOS
Supply voltage (V)	1.0		1.0	1.2	1.3	0.9	1.2	1.0
Decoder	SP-MS with post proc.		finite-alphabet	OMS with post proc.	split-row	Normalized Probabilistic MS	Reduced Complexity MS	Delayed Stochastic
precision (q_{ch}, q)	(3,2)	(3,3)	(4,3)	(4,4)	(5,5)	(4,3)	(6,6)	(5,1)
Iterations	11 + 8 PP	9 + 6 PP	5	8 + 6 PP	11	9	30	-
E_b/N_0 @ BER= 10^{-7} dB	4.51	4.53	4.95	4.25	4.55	4.4	4.32	4.7
Architecture	full-parallel		unrolled full-parallel	partial-parallel	full-parallel	full-parallel	layer parallel	full-parallel
Frequency (MHz)	421	500	862	700	195	199.6	226	750
Core area (mm^2)	1.76	2.56	16.2	5.05	4.84	9.6	3.84	3.93
Throughput (Gbps)	319.34†	384.96†	588	47.7*	92.8*	45.42*	12.8$^{\perp}$	172.4*
Hardware Eff. (Gbps/mm^2)	181.44†	150.38†	36.3	9.45*	19.1*	4.73*	3.33$^{\perp}$	43.86*
Scaled Hardware Eff.‡ (Gbps/mm^2)	181.44†	150.38†	36.3	72.71*	239.87*	157.1*	110.7$^{\perp}$	604.2*

† At a SNR level of 5.5 dB.

* The throughput reported is at $E_b/N_0 = 5.5$ dB. The maximum frequency is set to 1000 MHz in 28 nm, and the average number of iterations used is to compute the throughput and the hardware efficiency.

$^{\perp}$ Best throughput and and best hardware efficiency.

* It is not indicated an error-free operation below the BER level of 10^{-12}, the BER reported is only up to 10^{-7}.

‡ The scale factor for the frequency and throughput is $1/k$, for the area is k^2, and for the hardware efficiency is $1/k^3$, where k is the relative dimension to 28 nm.

5.8 Conclusion

In this chapter we have proposed an fully parallel architecture for regular and irregular LDPC codes, the architecture can be easily adapted to implement different decoders using different precision for messages and for LLRs. Using the proposed architecture, we have shown that the SP-MS decoder consumes slightly less area than the OMS decoder. We have also presented that when the precision of messages is reduced from $q = q_{ch}$ bits to $q = q_{ch} - 1$ bits, the area consumed by the SP-MS decoder is reduced by at least 25%.

From the study conducted in detail of the the IEEE 802.3 ETHERNET code, we have proposed a post-processing algorithm and two post-processing architectures. We have exhibited that our post-processing algorithm is very efficient in very low precision decoders, lowering the error floor below a FER of 10^{-10}. We have also presented that the algorithm is easily adaptable in a low precision decoder. Additionally, we have shown that the proposed algorithm converges rapidly using on average 3 to 4 iterations. On the other hand, we have demonstrated that the proposed architectures do not degrade the error correction performance in the waterfall region.

we have implemented the $(q_{ch} = 3, q = 2)$-bit SP-MS decoder that incorporates our post-processing algorithm and we have obtained an area of 1.76 mm^2, a decoding throughput of 319.34 Gbit/s, and a hardware efficiency of 181.44 Gbit/s/mm^2. When making the comparison with other decoders proposed in the literature, $(q_{ch} = 3, q = 2)$-bit SP-MS decoder is among one with the best hardware efficiency and good error correction performance.

Chapter 6

Conclusions

This thesis has proposed two low precision iterative decoders for low-density parity-check codes, the NAN-MS decoder and the SP-MS decoder. These decoders have been investigated and analyzed in the asymptotic limit of the code length using a noisy version of density evolution. The finite-length Monte Carlo simulations have corroborated the DE analysis. From the study carried out, it has been shown that the proposed decoders can reach a SNR gain of up to 0.43 dB. A much more exhaustive study of the SP-MS decoders has demonstrated that the precision of messages can be reduced by one bit maintaining the same error-correcting performance. It has also been presented that the SP-MS decoder consumes slightly smaller area than the OMS decoder when using the same precision. Additionally, it has been revealed that the reduction of one bit in the precision of the messages helps to reduce the area consumed by the SP-MS decoder by at least 25%.

This thesis has also proposed a post-processing algorithm. It has been shown that the post-processing algorithm is very efficient in correcting the absorbing set errors of iterative decoders, especially for very low precision decoders. The post-processing algorithm was implemented in a fully parallel architecture, and the emulation results have exhibited that the error floor is lowered below a frame error rate level of 10^{-10}

for the IEEE 802.3 ETHERNET code. Additionally, it has been determined that the algorithm converges quickly using on average 3 to 4 iterations.

The SP-MS decoder that incorporates the post-processing algorithm has been implemented obtaining an area of 1.76 mm^2, a decoding throughput of 319.34 Gbit/s, and a hardware efficiency of 181.44 Gbit/s/mm^2.

Future works that can be addressed from the results and conclusions presented in this thesis are: *(i)* the study of the SP-MS decoders in LDPC codes of the 5G, *(ii)* to use the concept of preserving the sign of the messages in FAIDs, and *(iii)* to adapt and extend the post-processing algorithm to other LDPC codes for storage applications. In addition, the concepts presented in this thesis can be adapted and extended to other codes such as the Turbo-code or the Polar code that need low precision decoders.

References

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. PhD thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, 1963.

[2] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[3] D. J. C. MacKay, *Information Theory, Inference, and Learning algorithms*. Cambridge University Press. ISBN:978-0-521-64298-9, 2003.

[4] ETSI, "ETSI EN 302 307-1 V1.4.1 (2014-07) Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 1: DVB-S2," 2014.

[5] ETSI, "ETSI EN 302 307-2 V1.1.1 (2014-10) Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 2: DVB-S2 Extensions (DVB-S2X)," 2014.

[6] "IEEE Standard for Ethernet," *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)*, pp. 1–4017, March 2016.

[7] "IEEE Standard for Local and metropolitan area networks - Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems - Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1," 2005.

[8] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in nand flash memory," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, pp. 429–439, Feb 2011.

[9] "IEEE Approved Draft Standard for Error Correction Coding of Flash Memory Using Low-Density Parity Check Codes," *IEEE P1890/D2, September 2017*, pp. 1–56, Jan 2018.

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

[11] F. R. Kschischang, B. J. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb 2001.

[12] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, March 1997.

[13] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the shannon limit," *IEEE Communications Letters*, vol. 5, pp. 58–60, Feb 2001.

[14] J. Chen and M. P. C. Fossorier, "Density evolution for two improved bp-based decoding algorithms of ldpc codes," *IEEE Communications Letters*, vol. 6, no. 5, pp. 208–210, May 2002.

[15] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Transactions on Communications*, vol. 50, pp. 406–414, March 2002.

[16] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of ldpc codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug 2005.

[17] L. Ping and W. K. Leung, "Decoding low density parity check codes with finite quantization bits," *IEEE Communications Letters*, vol. 4, no. 2, pp. 62–64, Feb 2000.

[18] R. Singhal, G. S. Choi, and R. N. Mahapatra, "Quantized ldpc decoder design for binary symmetric channels," in *2005 IEEE International Symposium on Circuits and Systems*, pp. 5782–5785 Vol. 6, May 2005.

[19] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (ldpc) codes," *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 549–554, April 2005.

[20] V. Savin, "Self-corrected min-sum decoding of ldpc codes," in *2008 IEEE International Symposium on Information Theory*, pp. 146–150, July 2008.

[21] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasic, "Finite alphabet iterative decoders—part i: Decoding beyond belief propagation on the binary symmetric channel," *IEEE Transactions on Communications*, vol. 61, pp. 4033–4045, October 2013.

[22] D. Declercq, B. Vasic, S. K. Planjery, and E. Li, "Finite alphabet iterative decoders—part ii: Towards guaranteed error correction of ldpc codes via iterative decoder diversity," *IEEE Transactions on Communications*, vol. 61, pp. 4046–4057, October 2013.

[23] T. T. Nguyen-Ly, V. Savin, K. Le, D. Declercq, F. Ghaffari, and O. Boncalo, "Analysis and design of cost-effective, high-throughput ldpc decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 508–521, March 2018.

[24] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding ldpc codes," *IEEE Transactions on Communications*, vol. 58, pp. 1610–1614, June 2010.

[25] A. Rasheed, P. Ivanis, and B. Vasic, "Fault-tolerant Probabilistic Gradient-Descent Bit Flipping Decoder," *IEEE Communications Letters*, vol. Vol. 18, no. 9, pp. 1487–1490, Sept. 2014.

[26] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, March 1999.

[27] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, Feb 2001.

[28] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, Feb 2001.

[29] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Analysis of low density codes and improved designs using irregular graphs," vol. [Online], Available: http://www.icsi.berkeley.edu/ luby/.

[30] S. Brkic, O. Al Rasheed, P. Ivaniš, and B. Vasić, "On fault tolerance of the gallager b decoder under data-dependent gate failures," *IEEE Communications Letters*, vol. 19, pp. 1299–1302, Aug 2015.

[31] E. Dupraz, D. Declercq, and B. Vasic, "Asymptotic error probability of the gallager b decoder under timing errors," *IEEE Communications Letters*, vol. 21, pp. 698–701, April 2017.

[32] D. Declercq, M. Fossorier, and E. Biglieri, *Channel Coding: Theory, Algorithms, and Applications*. Academic Press Library in Mobile and Wireless Communications. ISBN:978-0-12-396499-1, 2014.

[33] C. L. K. Ngassa, *LDPC Decoders Running on Error Prone Devices: Theoretical Limits and Practical Assessment of the Error Correction Performance*. PhD thesis, Université de Cergy-Pontoise, Cergy-Pontoise, France, 2015.

[34] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, Feb 2001.

[35] C. K. Ngassa, V. Savin, E. Dupraz, and D. Declercq, "Density evolution and functional threshold for the noisy min-sum decoder," *IEEE Transactions on Communications*, vol. 63, no. 5, pp. 1497–1509, May 2015.

[36] M. Fu, "On gaussian approximation for density evolution of low-density parity-check codes," in *2006 IEEE International Conference on Communications*, vol. 3, pp. 1107–1112, June 2006.

[37] G. Li, I. J. Fair, and W. A. Krzymien, "Density evolution for nonbinary ldpc codes under gaussian approximation," *IEEE Transactions on Information Theory*, vol. 55, no. 3, pp. 997–1015, March 2009.

[38] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press. 2007.

[39] E. Dupraz, D. Declercq, B. Vasic, and V. Savin, "Analysis and Design of Finite Alphabet Iterative Decoders Robust to Faulty Hardware," *IEEE Transactions on Communications*, vol. Vol. 63, no. 8, pp. 2797–2809, August 2015.

[40] C. K. Ngassa, V. Savin, and D. Declercq, "Min-sum-based decoders running on noisy hardware," in *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 1879–1884, Dec 2013.

[41] S. M. S. Tabatabaei Yazdi, H. Cho, and L. Dolecek, "Gallager b decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, pp. 1660–1673, May 2013.

[42] S. M. S. T. Yazdi, H. Cho, Y. Sun, S. Mitra, and L. Dolecek, "Probabilistic analysis of gallager b faulty decoder," in *2012 IEEE International Conference on Communications (ICC)*, pp. 7019–7023, June 2012.

[43] L. R. Varshney, "Performance of ldpc codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, July 2011.

[44] L. R. Varshney, "Performance of ldpc codes under noisy message-passing decoding," in *2007 IEEE Information Theory Workshop*, pp. 178–183, Sept 2007.

[45] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for ldpc codes," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3385–3400, Oct 2014.

[46] O. A. Rasheed, P. Ivaniš, and B. Vasić, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Communications Letters*, vol. 18, pp. 1487–1490, Sep. 2014.

[47] F. Leduc-Primeau, S. Hemati, S. Mannor, and W. J. Gross, "Dithered belief propagation decoding," *IEEE Transactions on Communications*, vol. 60, pp. 2042–2047, August 2012.

[48] C. L. K. Ngassa, V. Savin, and D. Declercq, "Unconventional behavior of the noisy min-sum decoder over the binary symmetric channel," in *2014 Information Theory and Applications Workshop (ITA)*, pp. 1–10, Feb 2014.

[49] K. Le, D. Declercq, F. Ghaffari, C. Spagnol, E. Popovici, P. Ivanis, and B. Vasic, "Efficient realization of probabilistic gradient descent bit flipping decoders," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1494–1497, May 2015.

[50] K. Le, F. Ghaffari, D. Declercq, and B. Vasić, "Efficient hardware implementation of probabilistic gradient descent bit-flipping," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, pp. 906–917, April 2017.

[51] F. Cochachin, D. Declercq, E. Boutillon, and L. Kessal, "Density Evolution Thresholds for Noise-Against-Noise Min-Sum Decoders," *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7, Oct 2017.

[52] A. Venkiah, D. Declercq, and C. Poulliat, "Design of Cages with a Randomized Progressive Edge Growth Algorithm," *IEEE Communications Letters*, vol. Vol. 12, no. 4, pp. 301–303, April 2008.

[53] T. Nguyen-Ly, K. Le, F. Ghaffari, A. Amaricai, O. Boncalo, V. Savin, and D. Declercq, "Fpga design of high throughput ldpc decoder based on imprecise offset min-sum decoding," in *2015 IEEE 13th International New Circuits and Systems Conference (NEWCAS)*, pp. 1–4, June 2015.

[54] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "An efficient 10gbase-t ethernet ldpc decoder design with low error floors," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 843–855, April 2010.

[55] T. Mohsenin, D. N. Truong, and B. M. Baas, "A low-complexity message-passing algorithm for reduced routing congestion in ldpc decoders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 5, pp. 1048–1061, May 2010.

[56] C. Cheng, J. Yang, H. Lee, C. Yang, and Y. 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.

[57] F. Angarita, J. Valls, V. Almenar, and V. Torres, "Reduced-complexity min-sum algorithm for decoding ldpc codes with low error-floor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 7, pp. 2150–2158, July 2014.

[58] A. Naderi, S. Mannor, M. Sawan, and W. J. Gross, "Delayed stochastic decoding of ldpc codes," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5617–5626, Nov 2011.

[59] A. J. Blanksby and C. J. Howland, "A 690-mw 1-gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 404–412, March 2002.

[60] W. Zhang, S. Chen, X. Bai, and D. Zhou, "A full layer parallel qc-ldpc decoder for wimax and wi-fi," in *2015 IEEE 11th International Conference on ASIC (ASICON)*, pp. 1–4, Nov 2015.

[61] K. Zhang, X. Huang, and Z. Wang, "High-throughput layered decoder implementation for quasi-cyclic ldpc codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 985–994, August 2009.

[62] A. Darabiha, A. Chan Carusone, and F. R. Kschischang, "Power reduction techniques for ldpc decoders," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 1835–1845, Aug 2008.

[63] R. Ghanaatian, A. Balatsoukas-Stimming, T. C. Müller, M. Meidlinger, G. Matz, A. Teman, and A. Burg, "A 588-gb/s ldpc decoder based on finite-alphabet message passing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 329–340, Feb 2018.

[64] P. Schläfer, N. Wehn, M. Alles, and T. Lehnigk-Emden, "A new dimension of parallelism in ultra high throughput ldpc decoding," in *SiPS 2013 Proceedings*, pp. 153–158, Oct 2013.

[65] A. Balatsoukas-Stimming, M. Meidlinger, R. Ghanaatian, G. Matz, and A. Burg, "A fully-unrolled ldpc decoder based on quantized message passing," in *2015 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, Oct 2015.

[66] S. Sharifi Tehrani, A. Naderi, G. Kamendje, S. Hemati, S. Mannor, and W. J. Gross, "Majority-based tracking forecast memories for stochastic ldpc decoding," *IEEE Transactions on Signal Processing*, vol. 58, pp. 4883–4896, Sep. 2010.

[67] G. Sundararajan and C. Winstead, "Asic design of a noisy gradient descent bit flip decoder for 10gbase-t ethernet standard," vol. [Online], pp. 1–12, Available: https://arxiv.org/abs/1608.06272.

[68] T. Richardson, "Error floors of ldpc codes," in *Proc. of the Allerton Conference on Communications, Control, and Computing*, pp. 1426–1435, Oct 2003.

[69] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. J. Wainwright, "Lowering ldpc error floors by postprocessing," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, pp. 1–6, Nov 2008.

[70] Y. Han and W. E. Ryan, "Low-floor decoders for ldpc codes," *IEEE Transactions on Communications*, vol. 57, pp. 1663–1673, June 2009.

[71] S. Landner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured ldpc codes," in *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 1, pp. 630–635 vol.1, June 2005.

[72] Y. Gong, X. Liu, W. Ye, and G. Han, "Effective informed dynamic scheduling for belief propagation decoding of ldpc codes," *IEEE Transactions on Communications*, vol. 59, pp. 2683–2691, October 2011.

[73] X. Chen, J. Kang, S. Lin, and V. Akella, "Hardware implementation of a backtracking-based reconfigurable decoder for lowering the error floor of quasi-cyclic ldpc codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, pp. 2931–2943, Dec 2011.

[74] K. Cushon, S. Hemati, C. Leroux, S. Mannor, and W. J. Gross, "High-throughput energy-efficient ldpc decoders using differential binary message passing," *IEEE Transactions on Signal Processing*, vol. 62, pp. 619–631, Feb 2014.

[75] C. Liu, S. Yen, C. Chen, H. Chang, C. Lee, Y. Hsu, and S. Jou, "An ldpc decoder chip based on self-routing network for ieee 802.16e applications," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 684–694, March 2008.

[76] B. Xiang, D. Bao, S. Huang, and X. Zeng, "An 847–955 mb/s 342–397 mw dual-path fully-overlapped qc-ldpc decoder for wimax system in 0.13μm cmos," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 6, pp. 1416–1432, June 2011.

[77] X. Shih, C. Zhan, C. Lin, and A. Wu, "An 8.29 $mm^2$52 mw multi-mode ldpc decoder design for mobile wimax system in 0.13μm cmos process," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 672–683, March 2008.

[78] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "Analysis of absorbing sets for array-based ldpc codes," in *2007 IEEE International Conference on Communications*, pp. 6261–6268, June 2007.

[79] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based ldpc codes," *IEEE Transactions on Information Theory*, vol. 56, pp. 181–201, Jan 2010.

[80] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. J. Wainwright, "Design of ldpc decoders for improved low error rate performance: quantization and algorithm choices," *IEEE Transactions on Communications*, vol. 57, pp. 3258–3268, Nov 2009.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

Université Bretagne Sud

Titre: Décodeur bruit contre bruit

Mot clés : Correction d'erreur, codes LDPC, Density Evolution

Résumé : Dans cette thèse, nous nous intéressons à l'amélioration des performances de décodeur Low-Density Parity-Check (LDPC) code fortement quantifié (3 ou 4 bits de précision en entrée). Le premier décodeur proposé, appelé décodeur Noise-Against-Noise Min-Sum (NAN-MS), intègre une certaine quantité de perturbations aléatoires dues à une injection délibérée de bruit dans le processus de décodage. L'autre décodeur, appelé décodeur Sign-Preserving Min-Sum (SP-MS), conserve toujours le signe des messages et utilise toutes les combinaisons possibles pouvant être générées pour une précision donnée. Nous montrons de plus qu'un décodeur SP-MS quantifiant ces entrées sur 3 ou 4 bits peut échanger des messages en internes quantifiés respectivement sur 2 ou 3 bits sans affecter le seuil de convergence du code dès lors que le degré des variables est supérieur à 4. Les décodeurs NAN-MS et SP-MS présentent un gain de SNR pouvant atteindre 0,43 dB dans la zone de convergence de la courbe de performances. D'autre part, nous avons proposé une modification d'un algorithme de post-traitement existant qui permet de réduire le taux d'erreur résiduelle (zone appelée « error floor ») pour les décodeurs avec seulement 2 bits de précision pour les messages échangés. Appliqué au code IEEE 10 Gigabit ETHERNET, l'algorithme SP-MS combiné avec l'algorithme de post-processing permet de réduire le niveau d'erreur au-dessous d'un taux d'erreur par trame de 10^{-10}. Nous avons implémenté ce décodeur en technologie ASIC 28 nm avec une architecture entièrement parallèle. La surface obtenue est de 1,76 mm^2, et le débit de décodage de 319,34 Gbit/s pour un rapport signal à bruit de 5,5 dB, soit une efficacité matérielle de 181,44 Gbit/s/mm^2.

Title: Noise-Against-Noise Decoder

Keywords : Error correction, LDPC codes, Density Evolution

Abstract : In this thesis, we are interested in improving the performance of the highly quantized Low-Density Parity-Check (LDPC) code decoder (3 or 4 bits of precision input). The first proposed decoder, named Noise-Against-Noise Min-Sum (NAN-MS) decoder, incorporates a certain amount of random perturbation due to deliberate noise injection into the decoding process. The other decoder, named Sign-Preserving Min-Sum (SP-MS) decoder, always preserve the sign of the messages and it uses all the possible combinations that can be generated for a given precision. We further show that a SP-MS decoder quantizing its inputs in 3 or 4 bits can reduce the precision of internal messages respectively to 2 or 3 bits without affecting the threshold of convergence of the code when the degree of the variable nodes is greater than 4. The NAN-MS decoder and the SP-MS decoder present a SNR gain of up to 0.43 dB in the waterfall region of the performance curve. On the other hand, we proposed a modification of an existing post-processing algorithm which makes it possible to reduce the residual error rate (region called "error floor") for the decoders with only 2 bits of precision for the exchanged messages. Applied to the IEEE 10 Gigabit ETHERNET code, the SP-MS algorithm combined with the post-processing algorithm reduces the error level below a frame error rate of 10^{-10}. We implemented this decoder in 28 nm ASIC technology with a completely parallel architecture. The resulting area is 1.76 mm^2, and the decoding rate of 319.34 Gbit/s for a signal-to-noise ratio of 5.5 dB, giving a hardware efficiency of 181.44 Gbit/s/mm^2.