Energy modeling and optimization of protograph-based LDPC codes

Mohamed Yaoumi

DOCTORAT BRETAGNE LOIRE MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

l'École Nationale Supérieure Mines-Télécom
Atlantique Bretagne Pays de la Loire - IMT Atlantique

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Télécommunication

Par
Mohamed YAOUMI

Energy modeling and optimization of protograph-based LDPC codes

Thèse présentée et soutenue à Brest , le 14/12/2020
Unité de recherche :Lab-STICC
Thèse N° : 2020IMTA0224

Rapporteurs avant soutenance :

Florence ALBERGE Maître de Conférences, IUT Orsay
Christophe JEGO Professeur, ENSEIRB-MATMECA

Composition du Jury :

Président :	Emmanuel BOUTILLON	Professeur, Université de Bretagne Sud
Rapporteurs:	Florence ALBERGE	Maître de Conférences, IUT Orsay
	Christophe JEGO	Professeur, ENSEIRB-MATMECA
Examinateurs :	Fakhreddine GHAFFARI	Maître de Conférences, University of Cergy-Pontoise
Dir. de thèse :	Frédéric GUILLOUD	Professeur, IMT Atlantique
Co-encadrante:	Elsa DUPRAZ	Maître de Conférences, IMT Atlantique

Invités :

Co-encadrant: François LEDUC-PRIMEAU Professeur adjoint, Polytechnique Montréal

Acknowledgement

To Mée, You will never be forgotten.

Mam, Dad, thank you for EVERYTHING.

Elsa, Frederic, Francois, thank you for your help.

Family and friends, thank you for your support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Energy Efficiency of Electronic Systems

In the design of electronic systems, energy efficiency can be defined as reducing the amount of energy required to realize a certain function. Energy efficiency is now a crucial issue for environmental reasons, but also to increase battery lifetime, or to improve computational capabilities of systems under limited resources. At the same time, many applications in telecommunications, signal processing, and machine learning, are very energy consuming, due to the increasing demand for these applications, and due to the increasing amount of data to be processed. The energy consumption of these systems can be reduced either by working on their on-circuit implementation (hardware design), or from the design of processing algorithms suitable for energy efficiency (software design). This thesis focuses more on the second aspect.

1.1.1 Energy efficient hardware implementation

A large body of literature is dedicated to the design of energy efficient hardware implementations. For this, one may first optimize the architecture itself. For instance, [2] proposes multiplier architectures and a multiplication technique that uses less area and can consume 58% less energy with a insignificant cost of accuracy. Also, using dedicated processors for a specific intensive computing application can provide 10–100x higher energy efficiency compared to general use processors [3]. One may also consider working on the degree of parallelism and on pipelining [4] [5].

Apart from optimizing the architecture, over the time, the size of circuits was considerably reduced, as well as their power supply. In order to keep decreasing the amount of energy per operations circuits should now operate in a near-threshold regime, where the supply voltage is approximately the same as the threshold voltage of the transistors [6]. Near-threshold circuits may provide a 10× improvement in energy efficiency [7] [8] [9] but

they may decrease the reliability of circuits and introduce faults in their computation operations. Considering the near-threshold computing would then require to design reliable algorithms working on unreliable hardware.

1.1.2 Energy efficient signal processing and machine learning algorithms

Energy efficiency can also be addressed by working on the signal processing algorithm itself, either by optimizing the algorithm for low energy consumption, or by designing methods which tolerate faults in their computation operations. Several applications were already considered in the literature. For instance, [10] proposes an energy minimization approach for learning in sensor networks. In addition, [11], [12] consider linear dot-product computation on unreliable hardware, while [13] addresses the effect of faults in estimation and hypothesis testing applications, and [14] focuses on distributed logistic regression with both unreliable processing units and unreliable memory units. Finally, [15] considers recursive binary estimation under computation noise and [16], [17] study the robustness of deep neural network under computation and memory failures.

One critical issue with these methods is to develop energy models in order to estimate the algorithm energy consumption, and to relate this energy consumption to the amount of faults introduced by the hardware. Such models may then be used in order to optimize the parameters of the algorithm (number of quantization bits, number of iterations, etc.) for a minimum energy consumption while satisfying a given performance criterion.

1.1.3 Energy consumption in telecommunication networks

In this thesis, we focus on energy efficiency in telecommunication systems. Indeed, Information and Communication Technology (ICT) is a domain in which there is a strong need for energy efficiency. Even-though ICT can be considered environmental-friendly by providing services in the daily life that help reduce the energy consumption (smart buildings, power management, etc), providing these services requires an important amount of energy. As a matter of facts, the ICT sector represents 8% of the worldwide energy consumption [18].

Telecommunication networks represent a non-negligible part of the ICT energy consumption [19]. Energy efficiency was also pointed out as an important concern in the design of modern and future telecommunication networks, see [20] [21] [22]. In addition,

from a state of the art realized in 2010, [23] asserts that error correction (evaluated with LTE Turbo codes) is responsible for approximately 1/3 of the overall energy consumption of baseband processing (which includes OFDM, modulation and demodulation, channel coding and decoding, etc.). This shows the importance of lowering the energy consumption of the error correction part, as we want to address in this thesis.

1.2 Energy efficient Error-correction Codes

In this work, we focus more on the energy efficiency in channel coding, more precisely, the energy efficiency of Error Correction Codes (ECC). ECC are very used in channel coding because they use less power in transmission but at the decoding level, they are limited by the Energy, hence the compromise between performance and decoding energy.

In the literature, there are different types of EEC, each of them presents a compromise between performance and decoding energy. Among these types of codes, we mention Turbo codes [24], LDPC codes [25], Polar codes [26], or Spatially Coupled codes [27]. In this work, we have chosen to treat this compromise in the case of Low-Density Parity Check (LDPC) codes. LDPC codes are a capacity-approaching codes, used in several standers, like 10Gbps Ethernet on CAT6 cables (IEEE 802.3an), WiMAX (IEEE 802.16), Wifi (IEEE 802.11n/ac) and 5G standard.

The EF-FECtive (Energy-First Forward Error-Correction) project focuses on the design of energy-efficient LDPC decoders by addressing the coding theory and hardware implementation areas. As presented in Figure 1.1, EF-FECtive project consist of three parts: (i) develop an LDPC decoder architecture for ASIC implementation, (ii) propose models to estimate the decoder energy consumption, (ii) optimize the code and decoder parameters to minimize the energy consumption.

This thesis aims to address the third point (code and decoder parameter optimization), by relying on the architecture and the energy models developed in this other parts of the project. More precisely, we seek to develop LDPC codes and decoders which allow for a significant reduction in the energy consumption of the decoder, while preserving the same decoding performance. We consider standard (non-faulty) decoders, and also faulty decoders which are implemented on circuits working in a near-threshold regime.

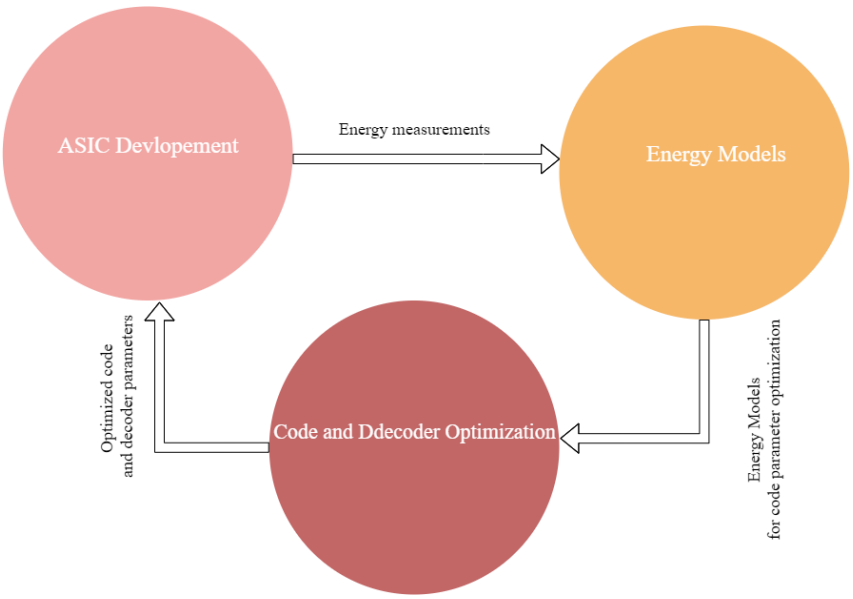

Figure 1.1 – Global view of the EF-FECtive project

1.3 Design of Energy-efficient LDPC codes

LDPC codes are linear codes, and the LDPC encoder [28] constructs a codeword which is then transmitted through a noisy channel. The LDPC decoder is an iterative message passing algorithm which should retrieve the original information sequence from the received noisy codeword. There exists different types of LDPC decoders. Hard-decision decoders such as Gallager A and B decoders [25], [29] and bit-flipping decoders [30] work with binary message values. More powerful soft-decision LDPC decoders like Sum-product [31], Min-Sum [32] [33] [34] [35] and c [36], work with integers or real message values. In practice, Min-Sum decoders together with Quasi-Cyclic LDPC codes are often considered, because they allow for efficient hardware implementations [37], [38], [39] [40].

All the above LDPC decoders where studied under hardware faults. Gallager A and B decoders were considered in [41] [42] [43] [44], Min-sum decoders were studied in [45] [46] [47] [48], bit-flipping decoders were examined in [49] [50] [51], and FAID decoders were considered in [52]. Density evolution is commonly used to predict the asymptotic performances of LDPC decoders, whether it is for the standard decoders [53, 54] or for faulty decoders [45]. Density Evolution can also be adapted to evaluate the finite-length performance of non-faulty [55] and faulty decoders [56].

In this work, we consider a quantized version of the Min-Sum decoder for which a specific architecture was developed in the EF-FECtive project [1] [57]. We then develop

high-level energy models for the considered architecture, and we introduce methods to optimize the code and decoder parameters in order to minimize the decoder energy consumption under given performance constraints. We adapt the finite-length Density Evolution method of [56] to the considered architecture, in order to estimate the decoding performance both in terms of Bit Error Rate and in terms of average number of iterations.

1.4 Contributions

We now present the thesis contributions. First, we adapted the finite-length Density evolution to the non-faulty Min-Sum decoder implemented on the considered architecture, and we further proposed a method to evaluate the average number of iterations at finite-length. Then, we introduced two high-level models to estimate the energy consumption of the quantized Min-Sum decoder. The first model uses the average number of operations required for decoding a codeword as a proxy for energy consumption. The second model considers the total number of bits that must be written in memory during the decoding process. From these models, we developed an optimization method in order to select protographs that minimize the decoder energy consumption while satisfying a given performance criterion. The proposed optimization method was based on a genetic algorithm called differential evolution.

In the second part of the thesis, we considered a faulty LDPC decoder, and we assumed that the circuit introduces some faults in the memory units used by the decoder. When considering faulty decoders, there is a tradeoff between the amount of faults introduced by the circuit and its energy consumption. In order to characterize this tradeoff, we used a noise model that connects the noise level in the bits stored in memory to the energy consumption of a memory cell. We then update the memory energy model so as to take into account the noise in the decoder. This energy model depends on the noise level, the number of quantization bits for the messages in the decoder, the codeword length, and the number of iterations performed by the decoder. Therefore, we proposed an alternate method in order to optimize these parameters so as to minimize the decoder energy consumption for a given protograph.

1.5 Organization of the manuscript

The manuscript is organized as follows. In Chapter 2, we present LDPC codes and decoders. We then show how to use Density Evolution for protographs, and describe the finite-length Density Evolution method. We also introduce the method to estimate the number of iterations at finite length. In Chapter 3, we introduce the complexity energy model and the memory energy model. In Chapter 4, we present the method to optimize the protograph for energy reduction, while maintaining a certain level of performance. Then in Chapter 5, we consider faulty Min-sum decoders and we introduce a faulty memory energy model. We then describe a method to optimize the decoder parameters for energy reduction. Finally in chapter 6, we optimize the protograph considering a faulty Min-Sum decoder.

Publications

1 - Yaoumi, M., Leduc-Primeau, F., Dupraz, E., & Guilloud, F. (2019, August). Optimization of Protograph LDPC Codes based on High-Level Energy Models. In 2019 16th International Symposium on Wireless Communication Systems (ISWCS) (pp. 6-10). IEEE.

2 - Yaoumi, M., Dupraz, E., Leduc-Primeau, F., & Guilloud, F. Optimisation de la Consommation d'Energie pour des Codes LDPC Construits à Partir de Protographes. GRETSI 2019, August.

3 - Yaoumi, M., Dupraz, E., Leduc-Primeau, F., & Guilloud, F. Energy Optimization of Quantized Min-Sum Decoders for Protograph-Based LDPC Codes. Accepted 20 August 2020 In Annals of Telecommunications.

4 - Yaoumi, M., Dupraz, E., Leduc-Primeau, F., & Guilloud, F. Energy Optimization of Faulty Quantized Min-Sum LDPC Decoders. In preparation for IEEE Communication Letters

Chapter 2

LDPC codes and Decoders

In this chapter, we describe LDPC codes and their constructions from degree distributions or protographes. We also present LDPC decoders, with a special focus on the quantized Min-Sum decoder which will be considered in the rest of this thesis. Then, we show how we use density evolution to evaluate the asymptotic performance of the quantized Min-Sum decoder. We also present a method based on Density Evolution to evaluate the finite-length performance of LDPC codes for long codewords. In addition, we show that this method can be used to estimate the average number of decoder iterations at finite length.

Throughout the chapter, we assume that each codeword bit is transmitted over an additive white Gaussian noise (AWGN) channel using binary phase-shift keying (BPSK) modulation. The i-th received value y_i is thus given by $y_i = x_i + b_i$ where b_i are independent centered Gaussian random variables with variance σ^2 and where $x_i \in \{-1, 1\}$ is the $i-$th modulated coded bit. The channel signal-to-noise ratio (SNR) is given by $\xi = 1/\sigma^2$.

2.1 LDPC Codes and Decoders

LDPC codes are a type of error-correction codes which provide near capacity performance. They were first introduced by Gallager in 1962 [25] in his PhD dissertation. In 1987, Tanner introduced a graphical representation of these codes called the Tanner graph [58]. However, LDPC codes were long ignored due to low computation capabilities at this time, so that practical implementation was not possible. In 1997, LDPC codes were re-discovered by Mackay [59] and the study of these codes was resumed.

In this work, we consider binary LDPC codes which can be represented by a sparse binary $M \times N$ parity-check matrix $\boldsymbol{H} = \{h_{i,j}\}$, where $h_{i,j}$ is the value in the i-th row and j-th column of $\boldsymbol{H}$. The matrix $\boldsymbol{H}$ is sparse in the sense that it has a low density of 1's. The parameter N gives the code length, $K = N - M$ denotes the information length if $\boldsymbol{H}$ is full rank, and $R = K/N$ is the code rate.

The parity check matrix $\boldsymbol{H}$ can also be represented by a bipartite Tanner graph that connects N variable nodes to M check nodes. Variables nodes correspond to the channel outputs, check nodes correspond to parity check equations. The i-th variable node v_i and the j-th check nodes c_j are connected in the Tanner graph if $h_{j,i} = 1$. We use d_{v_i} to denote the number of connections of the variable node v_i (variable node degree), and we use d_{c_j} to denote the number of connections of the variable node c_j (check node degree).

An example of a parity check matrix $\boldsymbol{H}$ of size 4×6 and code rate $R = \frac{1}{3}$ is given by:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

where every variable node v_i, $i \in \{1, \cdots, 6\}$ is connected to 2 check node, ($d_{v_i} = 2$) and every check node c_j, $j \in \{1, \cdots, 4\}$ is connected to three variable nodes ($d_{c_j} = 3$). The Tanner graph of the matrix H is given in Figure 2.1.

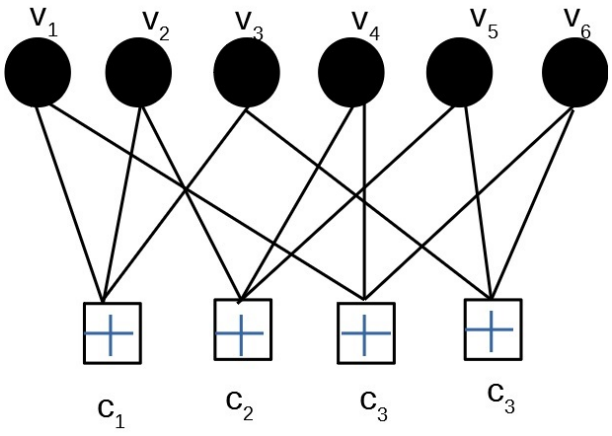

Figure 2.1 – Tanner graph of a parity check matrix H of size 4×6.

2.1.1 Regular and Irregular LDPC codes

We say that an LDPC code is regular [29] if all the variable nodes have the same variable node degree d_v and all the check nodes have the same check node degree d_c. The

code rate for regular codes is given by:

$$R = 1 - \frac{d_v}{d_c}$$

On the opposite, for irregular [53] LDPC codes, variable node and check node degrees are not constant. The node degrees are chosen according to predefined degree distributions. For irregular codes, the code rate is given by:

$$R(\lambda, \rho) = 1 - \frac{\sum_i \frac{\rho(i)}{i}}{\sum_i \frac{\lambda(i)}{i}} = 1 - \frac{\int_1^0 \rho(x)d(x)}{\int_1^0 \lambda(x)d(x)} \tag{2.1}$$

Degree distributions for the variable node degrees and check node degrees are given respectively by :

$$\lambda(x) = \sum_{i=2}^{dv_{max}} \lambda(i)x^{i-1} \quad and \quad \rho(x) = \sum_{i=2}^{dc_{max}} \rho(i)x^{i-1} \tag{2.2}$$

where the coefficients $\lambda(i)$ and $\rho(i)$ are defined as

$\lambda(i) = \mathbb{P}[$ an edge is connected to a variable node with $d_v = i]$

$\rho(i) = \mathbb{P}[$ an edge is connected to a check node with $d_c = i]$

The performance of an LDPC code depends on its degrees. We may find irregular codes with better performance than regular codes, but the degree distribution has to be optimized properly.

2.1.2 LDPC code construction from Protographs:

Alternatively, here, we consider LDPC codes constructed from a protographs [60]. An LDPC code constructed from protograph can be considered as an irregular code with a precise control of the connections between different types of variable nodes and check nodes. A protograph is specified by a matrix $\boldsymbol{S}$ of size $m \times n$ whose elements indicate the number of edges connecting the variable and check nodes of the Tanner graph [58] associated with $\boldsymbol{S}$. In the protograph, let d_{vi} be the total number of edges connected to a variable node of type $i \in \{1, \cdots, n\}$ and d_{cj} the total number of edges connected to a check node of type $j \in \{1, \cdots, m\}$. The protograph can contain multiple edges between a variable node v_i and a check node c_j if $S(i, j) > 1$, but the graph of the resulting LDPC code has at most one edge between a variable node and a check node. For LDPC codes constructed from protographs, the code rate is given by $R = 1 - \frac{m}{n}$.

A length-N LDPC code of rate R can be constructed from a protograph by applying a "copy-and-permute" operation on the protograph. The protograph is copied Z times, where $Z = N/n$ is called the lifting [61] factor. The parity check matrix $\boldsymbol{H}$ (which will be assumed full-rank hereafter) is then obtained by interleaving the edges. The degree distribution of the LDPC code is the one of the protograph, provided by the entries in $\boldsymbol{S}$.

For example, let us consider the protograph S of size 3×4:

$$S = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

In order to construct an LDPC code of length 8, we need $Z = \frac{8}{4} = 2$ copies of the protograph $\boldsymbol{S}$. Therefore, we interleave the edges so that the variable node and the check node satisfy the degree distribution provided by the Protograph $\boldsymbol{S}$. Figure 2.2 represents the Tanner graph of the protograph $\boldsymbol{S}$, while Figure 2.3 shows the copy operation from the protograph $\boldsymbol{S}$ and Figure 2.4 presents the final graph of the parity check matrix $\boldsymbol{H}$.

In this work, we are more interested in quasy-cyclic LDPC [62] codes. Quasy-cyclic LDPC codes are a class of structured type LDPC codes, characterized by there less complex encoding [63] and their parallel decoding structure that allows for higher error-correcting capacity. The quasi-cyclic nature of LDPC codes will provide a better decoding performance and allow for easier decoder implementations [64]. To construct quasy-cyclic LDPC codes, we need a two-step lifting procedure described in [65] [1]. The first lifting step aims to construct a base matrix B of size $m_b \times n_b$ from the protograph S using the same procedure described above, where $m_b = Z_1 m$ and $n_b = Z_1 n$. Z_1 is called the first lifting factor. The second lifting aims to construct the quasy-cyclic parity check matrix H of size $(M \times N)$ using the base matrix B, where $M = Z_2 m_b$ and $N = Z_2 n_b$. Z_2 is called the second lifting factor. In the second lifting, we replace every non-zero element of the base matrix B by a circulant matrices of size $Z_1 \times Z_2$. In this process, we use the PEG algorithm [66] [67] to place the circulant matrices in order to reduces the amount of short cycles in the final parity-check matrix $\boldsymbol{H}$.

2.1.3 LDPC decoding algorithms

LDPC decoding algorithms are iterative message-passing algorithms. They are called "message passing algorithms" because the decoding process is based on passing messages

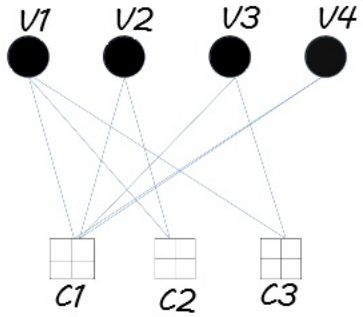

Figure 2.2 – The protograph $\boldsymbol{S}$

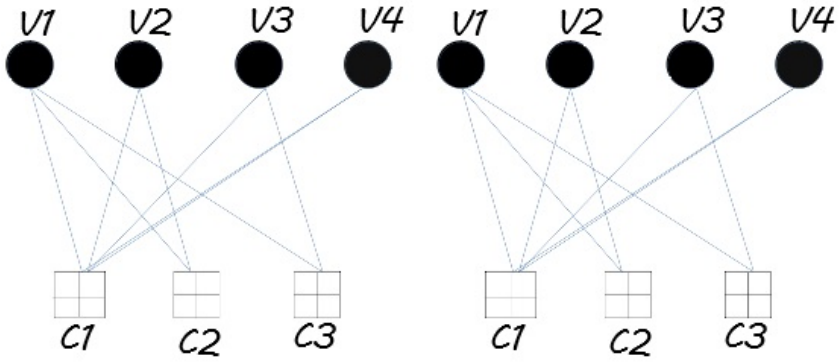

Figure 2.3 – The Protograph copied 2 times.

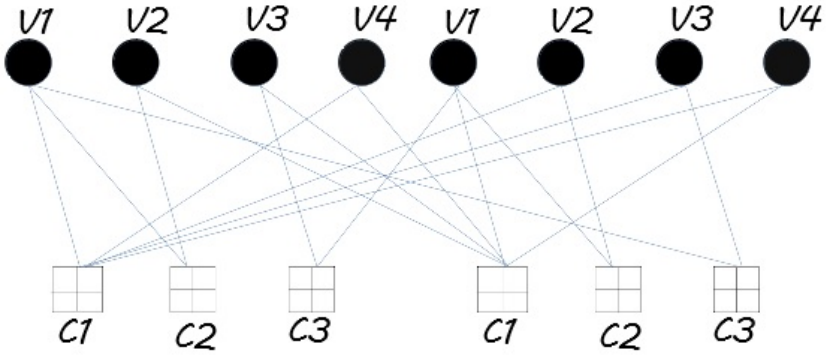

Figure 2.4 – The final LDPC code

from variable nodes to check nodes, and from check nodes to variable nodes through successive iterations. The messages sent from a variable node to a check node are calculated from the received messages from the channel and from the neighbouring check nodes of this variable node. However, due to the extrinsic principle, the message sent from a variable node v_i to a check node c_j does not use the previous message sent from the check node c_j to the variable node v_i. The extrinsic principle also holds when computing messages at the check nodes. The functions used to compute messages at variable nodes and check nodes are based on the chosen decoding algorithm. There are 2 types of decoders: "Hard" decoders which work with binary message values, Gallager A, B, bit-flipping decoders are hard-decision decoders. They are not very efficient in terms of performance, and will not be presented here. On the opposite, "Soft" LDPC decoders like Sum-product [31] and Min-Sum [32], work with integers or real message values, are much more efficient,and will be presented here.

Sum-Product decoder

The Sum-Product decoder is also known as Belief-Propagation decoder. In this decoder, the exchanged messages represent either probabilities or Log-Likelihood Ratio (LLR) for the codeword bits. In the following, we describe the LLR version of the Sum-Product decoder. The initial log likelihood ratio r_i for the the considered AWGN channel is given by :

$$\begin{aligned} r_i &= \log\left(\frac{\mathbb{P}\left(x_i = 1|y_i\right)}{\mathbb{P}\left(x_i = -1|y_i\right)}\right) \\ &= \log\left(\frac{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i-1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i+1)^2}{2\sigma^2}\right)}\right) \\ &= -\frac{(y_i-1)^2}{2\sigma^2} + \frac{(y_i+1)^2}{2\sigma^2} \\ r_i &= \frac{2y_i}{\sigma^2} \end{aligned} \tag{2.3}$$

where x_i is the i-th message sent through the channel and y_i is is the i-th channel output.

The Sum-Product decoder works in L iterations. In the Sum-Product decoder, the message sent from variable v_i to check c_j is denoted $\beta_{i\to j}^{(\ell)}$ at iteration ℓ and is calculated

as:

$$\beta_{i\to j}^{(\ell)} = r_i + \sum_{j'\in\mathcal{N}_{v_i}\setminus\{i\}} \gamma_{j'\to i}^{(\ell-1)}, \tag{2.4}$$

where $\mathcal{N}_{v_i}$ is the set of all check nodes connected to variable node v_i. Messages $\gamma_{j\to i}^{(\ell)}$ sent from the check node c_j to the variable node v_i are calculated according to:

$$\gamma_{j\to i}^{(\ell)} = 2\tanh^{-1}\left(\prod_{i'\in\mathcal{N}_{c_j}\setminus\{i\}} \tanh\left(\frac{1}{2}\beta_{i'\to j}^{(\ell)}\right)\right) \tag{2.5}$$

where $\mathcal{N}_{c_j}$ is the set of all check nodes connected to variable node c_j. At iteration ℓ, $\beta_i^{(l)}$ represents the A Posteriori LLR value, where:

$$\beta_i^{(\ell)} = r_i + \sum_{j\in\mathcal{N}_{v_i}} \gamma_{j\to i}^{(\ell-1)}. \tag{2.6}$$

The decision on the message x_i is taken based on the value of $\beta_i^{(\ell)}$ as:

$$x_i = \begin{cases} 1 & \text{if } \beta_i^{(\ell)} \geqslant 0 \\ -1 & \text{if } \beta_i^{(\ell)} < 0 \end{cases}$$

Offset Min-Sum decoder [68]

The Sum-Product algorithm is based on hyperbolic tangent calculations, which makes the hardware implementation more complex in case of e.g. FPGA implementation [69]. So the Min-Sum algorithm represents a simplified version of the Sum-Product algorithm and is designed to reduce hardware complexity [70]

In Min-Sum decoder , the message $\beta_{i\to j}^{(\ell)}$ sent from the variable v_i to check c_j at iteration ℓ is calculated as:

$$\beta_{i\to j}^{(\ell)} = \alpha r_i + \sum_{j'\in\mathcal{N}_{C_i}\setminus\{j\}} \gamma_{j'\to i}^{(\ell-1)} \tag{2.7}$$

In this expressions, α is a scaling parameter which should be optimized for good decoding performance.

The message $\gamma_{j\to i}^{(\ell)}$ sent back from the check node c_j to the variable node v_i is given

by:

$$\gamma_{j\to i}^{(\ell)} = \left(\prod_{i' \in \mathcal{N}_{c_j} \setminus \{i\}} \operatorname{sgn}\left(\beta_{i'\to j}^{(\ell)}\right) \right) \times \max\left[\min_{j' \in \mathcal{N}_{c_j} \setminus \{i\}} \left|\beta_{i'\to j}^{(\ell)}\right| - \lambda, 0 \right] \tag{2.8}$$

where $\operatorname{sgn}(x) = 1$ if $x \geq 0$ and $\operatorname{sgn}(x) = -1$ if $x < 0$. In the above equation λ is an offset parameter. We will explain later how to optimize it.

The Min-Sum algorithm simplifies the calculation at the check node by recognizing that small incoming messages $\beta_{i'\to j}^{(\ell)}$ dominates the product in (2.5). As a results the Min-Sum decoder reduces the complexity but has an inferior performance. However, a well optimized offset parameter can push the offset Min-Sum decoder to have a closer performance to the Sum-Product decoder.

Finally, $\beta_i^{(\ell)}$ represents the result of the decoding process at the iteration ℓ where:

$$\beta_i^{(\ell)} = r_i + \sum_{j \in \mathcal{N}_{v_i}} \gamma_{j\to i}^{(\ell-1)}, \tag{2.9}$$

The decision on the message x_i is taken based on the value of $\beta_i^{(\ell)}$ as:

$$x_i = \begin{cases} 1 & \text{if } \beta_i^{(\ell)} \geqslant 0 \\ -1 & \text{if } \beta_i^{(\ell)} < 0 \end{cases}$$

In this work, we consider quantized Min-Sum decoder [45–47]. For this, we use messages between $-Q$ and Q, with a quantization step-size s. The quantization function is given by

$$\Delta(x) = \operatorname{sgn}(x) \min\left(Q, s\left\lfloor \frac{|x|}{s} + \frac{1}{2} \right\rfloor \right), \tag{2.10}$$

where $\operatorname{sgn}(x) = 1$ if $x \geq 0$ and $\operatorname{sgn}(x) = -1$ if $x < 0$. For implementation efficiency, we usually choose $Q = 2^{q-1} - 1$, where q is the number of bits used to represent messages, and $\mathcal{M} = \{-Q, -Q+1, \ldots, Q\}$,

2.1.4 Serial and Parallel Scheduling

Message passing decoders can be implemented with different types of scheduling [71]. In flooding or parallel scheduling all the variable nodes (or check nodes) update their edges simultaneously. Alternatively, in *serial-C* scheduling, the check nodes are updated in a serial manner. After each check node update, all the variable nodes connected to it are updated. Serial-C scheduling enables to reduce the size of the circuit and requires

fewer decoding iterations [71].

2.2 Density Evolution for the quantized Min-Sum decoder

Density Evolution [53, 54] is a standard tool to evaluate the asymptotic performance of an LDPC decoder. For a given SNR, Density Evolution provides the decoder error probability p_{e_∞} averaged over the ensemble of codes described by protograph $\boldsymbol{S}$. The principle of Density Evolution is to follow the evolution of the probability distributions of the exchanged messages during the decoding process at successive iterations. The decoder error probability is then calculated from the probability distributions obtained by Density Evolution. In the Density Evolution method, we consider an infinite codeword length, we assume that the code is cycle free, and finally, we consider the all-zero codeword assumption.

In the following, we present how to use Density Evolution for a quantized min-sum decoder.

Initial probability values for quantized messages

For a quantized decoder, Density Evolution tracks the evolution of the probability values of the quantized messages at successive iterations. At the channel output, every initial quantized message $m_k \in \mathcal{M}$, has a probability $p_k = \mathbb{P}\left(m_{k-1} + \frac{1}{2} \leq r \leqslant m_i + \frac{1}{2}\right)$ calculated as follows:

$$\begin{aligned}
p_k &= \mathbb{P}\left(m_{k-1} + \frac{1}{2} < r \leqslant m_k + \frac{1}{2}\right) \\
&= \mathbb{P}\left(r \leqslant m_k + \frac{1}{2}\right) - \mathbb{P}\left(r < m_{k-1} + \frac{1}{2}\right) \\
&= \mathbb{P}\left(\frac{2\alpha y}{\sigma^2} \leqslant m_k + \frac{1}{2}\right) - \mathbb{P}\left(\frac{2\alpha y}{\sigma^2} < m_{k-1} + \frac{1}{2}\right) \\
&= \mathbb{P}\left(y \leqslant \frac{(m_k + \frac{1}{2})\sigma^2}{2\alpha}\right) - \mathbb{P}\left(y < \frac{(m_{k-1} + \frac{1}{2})\sigma^2}{2\alpha}\right) \\
&= \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{\frac{(m_k+\frac{1}{2})\sigma^2}{2\alpha} - x}{\sigma\sqrt{2}}\right)\right) - \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{\frac{(m_{k-1}+\frac{1}{2})\sigma^2}{2\alpha} - x}{\sigma\sqrt{2}}\right)\right)
\end{aligned} \tag{2.11}$$

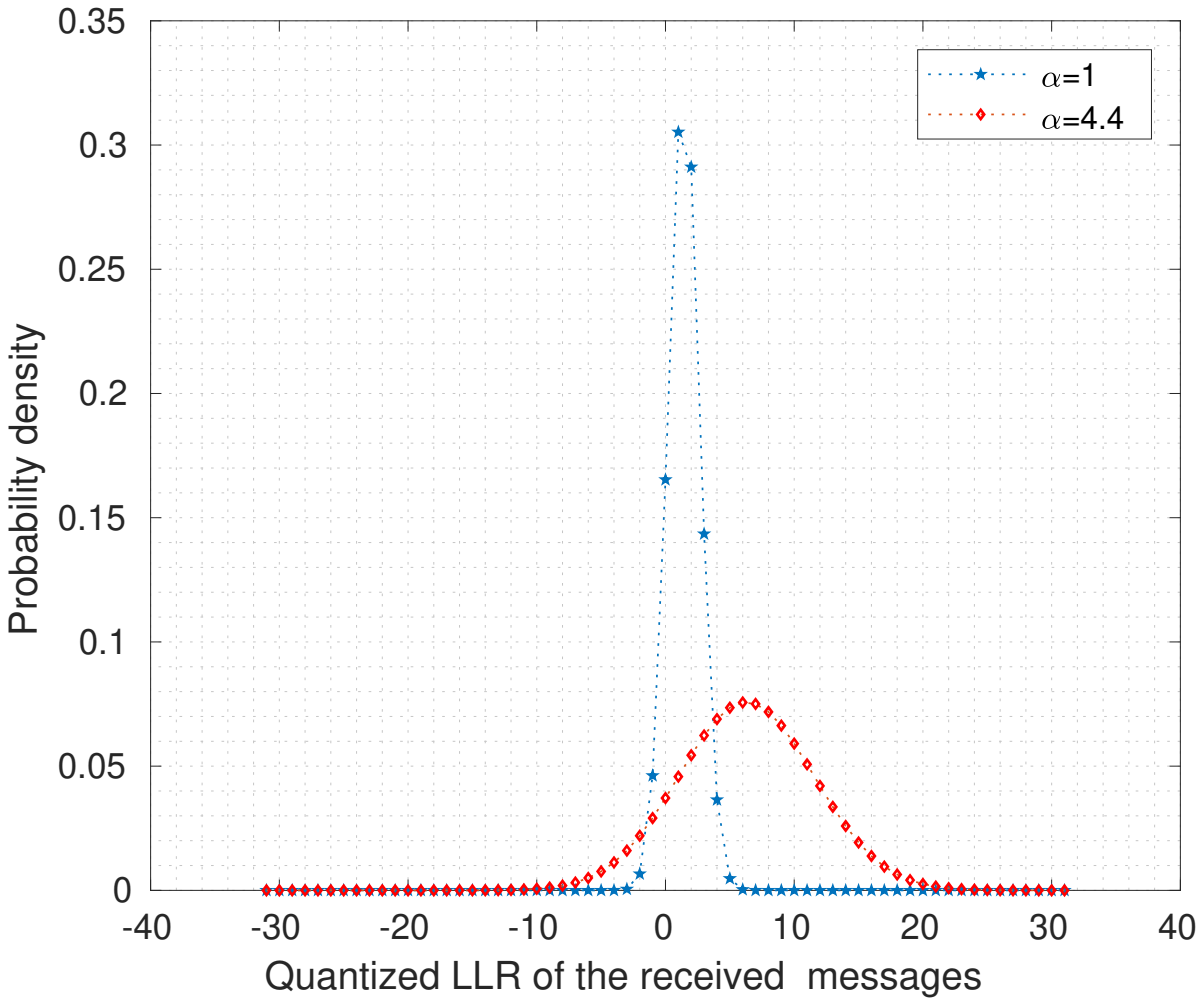

Figure 2.5 – Probability density of the received message at SNR value $\xi = 1.45$dB and $q = 6$

Figure 2.5 represents the density function of the received symbol y, calculated with 2 different values of α, (1 and $\alpha = 4.4$), for SNR value $\xi = 1.45$dB and $q = 6$. As we can see, the value of the parameter α can change the probability values of the quantized messages, which will affect the performance of the decoder. In order to have the best decoding performance, we will explain later how to optimize α using density evolution.

Density Evolution iterations

At first, suppose that we have a regular (d_v, d_c) LDPC code. Let $p^{(0)}$ be the initial probability vector of size $(2Q+1)$ that contains the probability values of every quantized message. Let $q^{(\ell)}$ be the probability vector of size $(2Q+1)$ of messages sent from check nodes to variable nodes at iteration ℓ. Let $p^{(\ell)}$ be the probability vector of size $(2Q+1)$ of messages sent from variable nodes to check nodes at iteration ℓ.

At iteration ℓ every check node receives the probability vector $p^{(\ell)}$ as an input. At the check nodes to calculate $q^{(\ell)}(m_i)$ the new probability of m_i at the ℓ-th iteration, we have to find all possible combinations of d_c-1 quantized messages that give the value m_i as result of the check node function. For example, suppose that $\phi(x)$ is the check node function, and that $m_i = \phi(m_1, m_2, \dots m_{d_c-1})$, where $m_x \in \mathcal{M}$ is the quantized message associated to the $x-th$ edge connected to the check node. The value of $q^{(\ell)}(m_i)$ is calculated by:

$$q^{(\ell)}(m_i) = \sum_{\boldsymbol{M_c}:\phi(\boldsymbol{M_c})=m_i} \left(\prod_{j=1}^{d_c-1} p^{(\ell-1)}(m_j) \right), \tag{2.12}$$

where $\boldsymbol{M_c} = (m_1, m_2, ..., m_{d_c-1})$ is a vector that contains the $d_c - 1$ input messages.

Based on this method we have now an updated vector probability $q^{(\ell)}$. For the variable nodes we follow the same process. The only differences are that the input vector probability is $q^{(\ell)}$ and that the LLR r is included as an edge associated with the initial probability $p^{(0)}$. Let $\omega(x)$ be the variable node function, and that $m_i = \omega(m_0, m_1, m_2, ...m_{d_c-1})$, where m_x is the quantized message associated to the $x-th$ edge connected to the variable node. The message $m_0 \in \mathcal{M}$ is the quantized message associated to the initial LLR. The value of $p^{(\ell)}(m_i)$ at the iteration ℓ is calculated by:

$$p^{(\ell)}(m_i) = \sum_{\boldsymbol{M_v}:\omega(\boldsymbol{M_v})=m_i} \left(\prod_{j=1}^{d_v-1} \left(q^{(\ell)}(m_j) \right) p^0(m_0) \right) \tag{2.13}$$

where $\boldsymbol{M_v} = (m_0, m_1, m_2, ..., m_{d_v-1})$. is a vector that contains the d_v input messages.

Error probability

From the above probability computation, we now want to evaluate the decoder error probability. Suppose that the algorithm stops after L iterations. Then the decoder error probability is calculated by :

$$p_{e_\infty} = \sum_{k=1}^{Q} p^{(L)}(k) + \frac{1}{2} p^{(L)}(Q+1) \tag{2.14}$$

This error probability depends on the SNR value and the node degrees d_v and d_c. The Density Evolution method described above considers regular LDPC Codes. We now will explain how to implement density evolution for protographs.

2.2.1 Density Evolution for protographs

Density Evolution for protographs is different compared to the case of regular and irregular degree distributions. In case of protographs, we can apply multi-edge [54] Density Evolution, where we consider not only the degree distribution, but also the type of edge connecting a variable node to a check node. In density evolution for protographs, each edge of the Protograph can correspond to a different probability distribution. So for a better explanation of the multi-edge Density Evolution, we will focus on the example of protograph S given below:

$$S = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \end{pmatrix}$$

Connection table:

With regular codes, for instance at the check node, we update messages using $d_c - 1$ identical edges. But with protographs, edges incoming to a check node are not of the same type. For example, Figure 2.7 shows that, to update the message $\gamma_{1\to4}$, we use the two incoming messages from the variable node v_1. But to update the message $\gamma_{1\to1}$ sent over the first edge connecting v_1 to c_1, we use only one remaining edge from v_1 to c_1, and the other messages come from v_2, v_3 and v_4 as shown in Figure 2.6.

Therefore, before applying the Density Evolution process, we need to identify all the potential connections between each variable node v_i and all the check nodes, while satisfying the extrinsic principle. This is why, for each variable node v_i, we create a connection

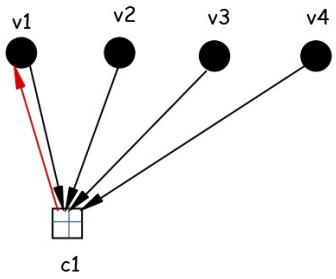

Figure 2.6 – $\gamma_{1\to1}$ update

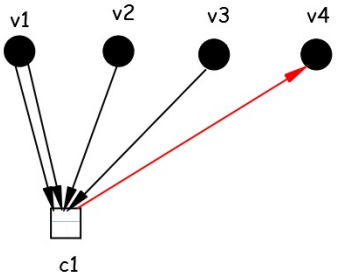

Figure 2.7 – $\gamma_{1\to4}$ update

table that lists the incoming edges involved in the computation of the message sent to each check node c_j. We now give an example of how we construct the connection table.

In the protograph example S, the first variable node is connected to the first check nodes by 2 edges ($S_{1,1} = 2$) and the second variable is connected to the same check node by one edge ($S_{1,2} = 1$) and so on. So to compute the message from check node c_1 to variable node v_1, we use the following connection vector: [$S_{1,1} - 1 = 1$, $S_{1,2} = 1$, $S_{1,3} = 1$, $S_{1,4} = 1$]. To compute the message from check node c_1 to variable node v_2, we use the following connection vector: [$S_{1,1} = 2$, $S_{1,2} - 1 = 0$, $S_{1,3} = 1$, $S_{1,4} = 1$]. We use the same method for the rest of the variable nodes connected to c_1 until we have the connection table T_{dc_1} of size $(n \times n)$. For example, we have here the connection table for the first check node:

$$T_{dc_1} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{pmatrix}$$

We use the same method for the rest of the check nodes until we have the final connection table T_{dc} of size $(mn \times n)$.

To construct the connection table T_{dv} we use the same process for every variable node. For example, the connection table for the first variable node T_{dv_1} is:

$$T_{dv_1} = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}$$

We continue for the rest of the variables until we have the final connection table for the variable nodes T_{dv} of size $(mn \times m)$.

Here we have the final connection table T_{dv} for the protograph example considered in

this section:

$$T_{dv} = \begin{pmatrix} 1 & 1 \\ 2 & 0 \\ 0 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

We use the same method for the connection table T_{dc}.

Density Evolution iterations:

For the density Evolution iterations we use the same method as for regular LDPC codes, but instead of using $\mathcal{N}_{c_i}$ *and* $\mathcal{N}_{v_j}$, we use T_{dv} and T_{dc} and we consider every line of the table as an edge to be updated. Suppose that S is a photograph of size $(m \times n)$, and T_{dv} and T_{dc} are the connection tables for the variable nodes and the check nodes respectively. So at the variable node level the updated probability $p_i^{(\ell)}(m_w)$ of the quantized message m_w at the ℓ-th iteration for the i-th connection(i-th line in the connection table) is given by :

$$p_i^{(\ell)}(m_w) = \sum_{\boldsymbol{M_v}:\omega(\boldsymbol{M_v})=m_w} \left(\prod_{j=1}^{m} \left(\prod_{k=1}^{T_{dv}(i,j)} q_{i'}^{(\ell-1)}(m_r) \right) \right) \tag{2.15}$$

where i' represents the index of the connection needed to update $p_i^{(\ell)}(m_w)$ given by $i' = \left(\frac{i-1}{n}\right) \times n + j$. In addition, $\boldsymbol{M}_v = (m_1, m_2, ...m_W)$, where $W = \sum_{j=1}^{m} T_{dv}(i,j)$.

At the check node level, the updated probability $q_i^{(\ell)}(m_w)$ of the quantized message m_w at the ℓ-th iteration for the i-th connection(i-th line in the connection table) is given by :

$$q_i^{(\ell)}(m_w) = \sum_{\boldsymbol{M_c}:\phi(\boldsymbol{M_c})=m_w} \left(\prod_{j=1}^{n} \left(\prod_{k=1}^{T_{dc}(i,j)} p_{i'}^{(\ell-1)}(m_r) \right) \right) \tag{2.16}$$

where $i' = i - \left(\frac{i-1}{m}\right) m + (j-1)\ n$. In addition, $M_c = (m_1, m_2, ...m_W)$, where $W = \sum_{j=1}^{n} tab_{dc}(i,j)$.

Error probability :

The final error probability is calculated over all the edges as:

$$p_{e_\infty} = \frac{\sum_{i=1}^{mn}\left(\sum_{k=1}^{2Q} p_i^{(L)}(k)\right) + \frac{1}{2}\sum_{i=1}^{mn} p_i^{(L)}(2Q+1)}{mn}$$

This error probability depends on the channel parameters, the SNR ξ for an AWGN channel.

2.2.2 Finite-length Density Evolution

With the density evolution method presented in the previous section, the error probability p_{e_∞} is evaluated assuming an infinite codeword length. In order to predict the finite-length performance of the quantized Min-Sum decoder described in Section 7.3, we rely on the method proposed in [56]. In this method, in order to evaluate the decoder error probability $p_{e_N}(\xi)$ at SNR ξ for a codeword length N, we use the following equation:

$$p_{e_N}(\xi) = \int_0^{\frac{1}{2}} p_{e_\infty}(x)\, \phi_{\mathcal{N}}\left(x;\, p_0, \frac{p_0(1-p_0)}{N}\right) dx \tag{2.17}$$

In this expression, $p_0 = \frac{1}{2} - \frac{1}{2}\mathrm{erf}\left(\sqrt{\xi/2}\right)$, and $p_{e_\infty}(x)$ is the error probability evaluated with standard density evolution at SNR value $2(\mathrm{erf}^{-1}(1-2x))^2$. The function $\phi_{\mathcal{N}}(x;\mu,\sigma^2)$ is the probability density function of a Gaussian random variable with mean μ and variance σ^2.

We can use the same method to estimate the FER performance of the decoder is evaluated for a codeword of length N, with SNR ξ, and iteration ℓ, from the following formula:

$$B_N^{(\ell)}(\xi) = \int_0^{\frac{1}{2}} B_\infty^{(\ell)}(x)\, \Phi_{\mathcal{N}}\left(x; p_0, \frac{p_0(1-p_0)}{N}\right) dx. \tag{2.18}$$

where $B_\infty^{(\ell)}(x) = 1-(1-p_{e\infty}^{(\ell)}(x))^N$, and $p_{e\infty}^{(\ell)}(x)$ gives the decoder error probability calculated by standard DE for variance value $v^2 = \frac{1}{2\left(\mathrm{erf}^{-1}(1-2x)\right)^2}$.

This method takes into account the variability in the channel at finite-length. It does not take into account the effect of cycles in the code parity check matrix, which can also affect the finite-length performance of the decoder. This method is therefore well suited for codes from moderate to long length N. Finally, this method works for both regular, irregular codes, or codes constructed from protographs.

2.2.3 Estimation of the number of iterations at finite length

In order to evaluate its energy consumption, we also need to predict the number of iterations required by the decoder at a given length N. In order to do so, we now introduce our method to evaluate the average number of iterations $L_N(\xi)$ at length N. For this, we use $L_{N,\xi}(j)$ to denote the number of iterations needed to decode a given frame $j \in [\![1, 2^K]\!]$, where $K = N - M$, depending on the codeword length N and the SNR ξ. The average number of iterations $L_N(\xi)$ can then be expressed as

$$L_N(\xi) = E\left[L_{N,\xi}^{(\ell)}\right] = \sum_{j=1}^{2^K} \frac{1}{2^K} L_{N,\xi}(j) = \sum_{j=1}^{2^K} \frac{1}{2^K} \sum_{\ell=1}^{L} \mathbb{1}_{\mathcal{A}_{N,\xi}^{(\ell)}}(j) \tag{2.19}$$

where $\mathcal{A}_{N,\xi}^{(\ell)}$ is the set of frames which need at least iteration ℓ in the decoder, and $\mathbb{1}$ is the indicator function. We then denote by $\mathcal{B}_{N,\xi}^{(\ell-1)}$ the set of frames perfectly decoded at iteration $\ell - 1$. We assume that the stopping criterion considered in the decoder is perfect, which gives that $\overline{\mathcal{B}_{N,\xi}^{(\ell-1)}} = \mathcal{A}_{N,\xi}^{(\ell)}$, where $\overline{\mathcal{B}_{N,\xi}^{(\ell-1)}}$ is the complement of $\mathcal{B}_{N,\xi}^{(\ell-1)}$. Under this assumption, we have that

$$L_N(\xi) = \sum_{j=1}^{2^K} \frac{1}{2^K} \sum_{\ell=1}^{L} \left(1 - \mathbb{1}_{\mathcal{B}_{N,\xi}^{(\ell-1)}}(j)\right) \tag{2.20}$$

$$= \sum_{\ell=1}^{L} \left(1 - B_N^{(\ell-1)}(\xi)\right) \tag{2.21}$$

where $B_N^{(\ell-1)}(\xi)$ is defined in (7.5).

The above derivation relies on the condition that $\overline{\mathcal{B}_{N,\xi}^{(\ell-1)}} = \mathcal{A}_{N,\xi}^{(\ell)}$. In order to determine if it should stop at iteration $\ell - 1$, the decoder relies on a stopping criterion. However, for some frames j, this stopping criterion may be verified while the frame is not correctly decoded, thus leading to $j \in \overline{\mathcal{B}_{N,\xi}^{(\ell-1)}}$ but $j \notin \mathcal{A}_{N,\xi}^{(\ell)}$. The probability of this event depends on the code cycle distribution and minimum distance. In what follows, since we consider long codewords, we choose to neglect this event and evaluate the average number of iterations as (7.8).

The FER (7.5) and the average number of iterations (7.8) are evaluated by taking into account channel uncertainty at length N. However, these evaluations do not take into account code cycles which can also degrade the decoder performance at finite length. These methods are therefore well-suited to evaluate the code performance for codewords of moderate to long sizes, starting from around 3000 bits.

2.3 Conclusion

In this chapter, we presented LDPC codes and how to construct quasy-cyclic LDPC codes from protographs. We presented also the sum-product decoder and the offset min-sum decoder. We then showed how to use Density evolution for quantized min-sum decoder, and protographs. Finally we introduced a finite-length density evolution method, and we proposed a method to estimate of the number of iterations at finite length. In the next chapter, we will study the energy consumption of the quantized Min-Sum decoder.

Chapter 3

Energy model for non-faulty quantized Min-Sum decoders

In the previous chapter, we showed how to use Density Evolution to evaluate the performance of the quantized Offset Min-Sum decoder at finite-length, both in terms of decoder error probability and in terms of number of iterations. In this chapter, we aim to evaluate the energy consumption of the quantized Min-sum decoder, for a specific hardware architecture. We start by the existing methods to estimate the energy consumption of LDPC decoders. . We then present the hardware architecture which we consider for the quantized Min-Sum decoder, and we introduce two high-level models in order to estimate the decoder energy consumption for this architecture.

3.1 State of the art on energy consumption evaluation of LDPC decoders

In addition to the decoding performance and the transmission power, the decoder energy consumption is a design criterion of importance for hardware implementation. However, it was taken into account only recently in the literature [72]. In this section, we describe existing models to evaluate the energy consumption of LDPC decoders.

In [72], two energy consumption models are introduced for LDPC decoders. The first model considers the energy consumed in each variable and check node for message computation. For this model, the average power per information bit is given by :

$$P_n = \frac{E_n(2N-K)L}{K} R_s = \mathcal{E}_n L \tag{3.1}$$

where E_n represents the energy consumed in a variable node or a check node during one decoding iteration, N is the number of variable nodes, R_s is the number of source bits

decoded per second, and L denotes the total number of decoding iterations. Then, the decoder energy consumption is evaluated as:

$$\mathcal{E}_n = \frac{E_n(2N-K)}{K} R_s = \frac{E(d_c+d_v)}{(d_c-d_v)} R_s \tag{3.2}$$

This model takes into account only the energy consumed in the processing units, and does not consider message exchange between nodes.

The second model introduced in [72] evaluates the energy consumed by wires in the decoder. For this model, the decoding power is:

$$P_w = CAV^2 f = \mathcal{E}_w A \tag{3.3}$$

where C represents the capacitance per unit-area of a wire, A is the total area occupied by the wires in the circuit, V represents the supply voltage of the circuit, and f is the clock-frequency of the circuit. Finally in the wire model, the energy consumption $\mathcal{E}_w$ is evaluated as:

$$\mathcal{E}_w = CV^2 f \tag{3.4}$$

In both models, energy consumption depends on the code length and on the code degree distribution. In addition, these two energy models depend on the considered decoder, through energy per node E_n which depends on the functions used in variable nodes and check nodes. In addition, these two models do not evaluate the memory energy consumption in the decoder. However, as stated in [73], the energy consumption due to memory access is non-negligeable and should be taken into account during energy consumption evaluation. In order to decrease the memory energy consumption, [73] considers Finite Alphabet Iterative Decoders (FAIDs), and seeks to minimize the size of message alphabets while maintaining a good level of performance. The approach of [73] reduces both memory and wire energy consumption. This observation is however performed from numerical simulations, and no model is given to estimate the decoder energy consumption.

While previous works [74] and [73] optimize the decoder, the code itself can play an important role in energy consumption. Therefore, [75, 76] seek to minimize the decoding complexity for a target decoding performance by a numerical optimization of the code rate and irregular degree distributions. For example, [75] considers irregular codes, and

evaluates the code complexity by using the following model:

$$\mathcal{K} = L\frac{\mathcal{I}}{Rn} \tag{3.5}$$

where $\mathcal{I} = n\frac{1}{\int_1^0 \lambda(x)d(x)} = m\frac{1}{\int_1^0 \rho(x)d(x)}$ gives the total number of edges in the graph of the code. However, both works [75, 76] assume an infinite codeword length, and [75] considers the Gallager B decoder while [76] studies the Sum-Product decoder.

Therefore, in the following, we aim to complete the above models, and to investigate them in the case of the quantized Min-Sum decoder. Therefore, we introduce two models that estimate the energy consumption of a quantized Min-Sum decoder for protograph-based LDPC codes. The first model counts the total number of operations required by the decoder.The second model considers the total number of bits that must be written in memory during the decoding process.

3.2 Min-Sum decoder architecture

In the above section, we have seen that the energy consumption depends on the considered decoder. Therefore, before providing the energy models, we first introduce the Min-sum architecture of [1] [57] which we consider in this work. This architecture uses a large number of processing units in parallel, and relies on datapath pipelining, which is a technique for increasing parallelism at a small cost in the circuit. However, the use of pipelining requires that the operations performed in parallel have no data dependencies. In addition, in this architecture the Serial-C scheduling [71] is considered, because it reduces the total number of iterations by a factor of 2, and is well-suited for Quasy-Cyclic LDPC code constructions. However, the use of the Serial-C scheduling imposes additional data dependencies, which should be handled by the architecture in order to allow for pipelining. Figure 3.1 gives a global view of the considered architecture.

In the architecture proposed in [1], messages $\beta_i^{(\ell)}$ calculated in variable nodes at iteration ℓ are given by:

$$\beta_i^{(\ell)} = \Delta(r_i) + \sum_{j \in \mathcal{N}_{v_i}} \gamma_{j\to i}^{(\ell-1)}, \tag{3.6}$$

where $\gamma_{j\to i}^{(\ell)}$ represents the message sent from the check node c_j to the variable node v_i at iteration ℓ. In this architecture, in order to update the messages at the variable node, we

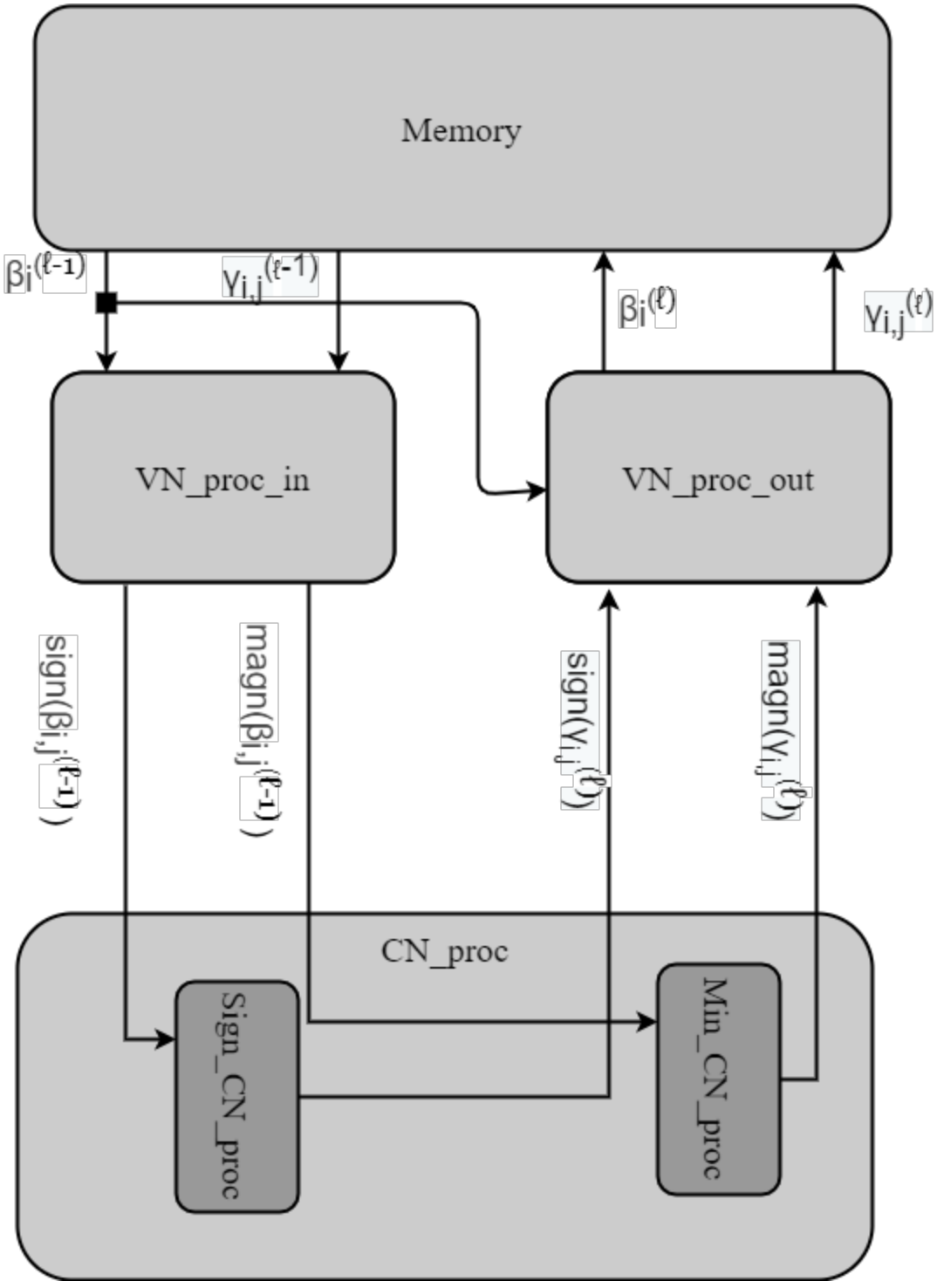

Figure 3.1 – Architecture of the decoder core presented in [1]

use the following equation:

$$\beta_i^{(\ell)} \leftarrow \beta_{i\to j}^{(\ell-1)} + \gamma_{j\to i}^{(\ell-1)} \tag{3.7}$$

As a result, in this architecture, at each variable node, only the total sum of the incoming messages $\beta_i^{(\ell)}$ is stored in memory. Thus, when check node computations are performed successively in the pipeline, they can benefit from the newest version of variable node messages.

In the considered architecture, variable node i sends the same message $\beta_i^{(\ell)}$ to all its neighboring check nodes. The specific message $\beta_{i\to j}^{(\ell)}$ of every check will be calculated during the check node update, and check to variable node messages $\gamma_{j\to i}^{(\ell)}$ are calculated as

$$\begin{aligned} \beta_{i\to j}^{(\ell)} &= \beta_i^{(\ell)} - \gamma_{j\to i}^{(\ell-1)} \\ \gamma_{j\to i}^{(\ell)} &= \left(\prod_{i'\in\mathcal{N}_{c_j}\setminus\{i\}} \operatorname{sgn}\left(\beta_{i'\to j}^{(\ell)}\right) \right) \times \max\left[\min_{j'\in\mathcal{N}_{c_j}\setminus\{i\}} \left|\beta_{i'\to j}^{(\ell)}\right| - \lambda, 0 \right], \end{aligned} \tag{3.8}$$

where λ is the offset parameter.

The decoder is initially parametrized with a number of quantization bits q. However, when calculating the sum in (7.1), a saturation problem can occur. For example suppose that $d_{v_i} = 3$, $\gamma_{1\to i} = Q$, $\gamma_{2\to i} = -Q$, and $\gamma_{3\to i} = Q$. Then $\beta_i^{(\ell)}$ can be calculated in two ways:

(i) $\beta_i^{(\ell)} = \Delta\left(\Delta\left(\gamma_{1\to i} + \gamma_{2\to i}\right) + \gamma_{3\to i}\right) + \Delta(r_0) = Q + \Delta(r_0)$
(ii) $\beta_i^{(\ell)} = \Delta\left(\Delta\left(\gamma_{1\to i} + \gamma_{3\to i}\right) + \gamma_{2\to i}\right) + \Delta(r_0) = \Delta(r_0)$

As we can see, due to message saturation in the quantization function, message values differ depending on the computation order. To avoid this saturation error, we should choose a large number of quantization bits, so as to be able to represent the maximum possible message value given by $M_s = \lceil \frac{d_v+1}{2} \rceil |Q|$, with $|Q| = 2^{q-1}$, and

$$q_s = \lceil log_2 \left(\lceil \frac{d_v+1}{2} \rceil \right) \rceil - 1 \tag{3.9}$$

so that the maximum message value $M_s = 2^{q_s+q}$ lies in the quantization alphabet. Since the variable node degrees d_{v_i} vary, q_s is calculated from $d_{vmax} = \max(d_{vi}), i \in \{1, \cdots, n\}$. This choice of q_s ensures that no saturation error occurs when computing the total belief $\beta_i^{(\ell)}$. Note that other messages in the decoder are quantized on q bits only.

3.3 Memory and Complexity analysis

We now introduce the two energy models for the previous architecture. For this, we start by analyzing the total number of memory access and the total number of operations required for decoding.

3.3.1 Memory analysis

For a check node c with degree d_c, we must store one sign bit for every output message, and two minimum values of $q-1$ bits each. Thus the total number of stored bits is $d_c + 2q - 2$. In a variable node v of degree d_v, only $\beta_i^{(\ell)}$ has to be saved in memory, which requires $q + q_s$ bits.

3.3.2 Complexity analysis

Due to the serial-C scheduling, a variable node is updated each time one of its neighboring check nodes is updated. Considering that variable node v_i is connected to check node c_j that is being updated, we first compute (7.13), and once the check node has been updated, finish the variable node update with

$$\beta_i^{(\ell)} \leftarrow \beta_{i\to j}^{(\ell-1)} + \gamma_{j\to i}^{(\ell-1)},$$

this requires $2d_{v_i}$ additions during one iteration, each applied to inputs of $q + q_s$ bits.

For the check node update, the processing of the sign in (7.14) requires $(2d_{c_j} - 1)$ 2-input exclusive-OR (XOR-2) operations. Finally, we assume that the calculation of the two minimum values of (7.14) is performed using a merge-sort circuit architecture [47]. This circuit requires $\lfloor \frac{d_{c_j}}{2} \rfloor + 2(\lceil \frac{d_{c_j}}{2} \rceil - 1)$ comparisons, and all the comparisons are performed on inputs of $q-1$ bits.

3.4 Energy Models

3.4.1 Complexity energy model

In order to derive the complexity energy model, we denote by E_{add}, E_{xor}, E_{comp}, the elementary energy consumption of a 1-bit addition, an XOR-2 operation, and a 1-bit comparison, respectively. Consider an LDPC code of length N, rate R, and constructed

from a protograph S. For a target SNR ξ and bit error rate (BER) p_e, the complexity energy model is given by:

$$E_c = \frac{L_N(\xi, p_e)N}{n}\Bigg(2(q+q_s)E_{\text{add}}\sum_{i=1}^{n} d_{v_i} + (1-R)\left(E_{\text{xor}}\sum_{j=1}^{m}\left(2d_{c_j}-1\right) + E_{\text{comp}}(q-1)\left(\lfloor\frac{d_{c_j}}{2}\rfloor + 2\left(\lceil\frac{d_{c_j}}{2}\rceil - 1\right)\right)\right)\Bigg) \quad (3.10)$$

where $L_N(\xi, p_e)$ (7.7) is the total number of iterations used by the decoder.

The number of operations performed by check nodes with d_{c_j} even and by check nodes with d_{c_j} odd only differ by a constant $\frac{1}{2}$. We can thus approximate E_c with the worst case where all the d_{c_j} are odd. If we also assume that a comparison has the same complexity as an addition, i.e. $E_{\text{comp}} = E_{\text{add}}$, the complexity energy model simplifies to:

$$E_c \;=\; L_N(\xi, p_e)N\Bigg(2E_{\text{add}}(q+q_s)\tilde{d}_v + E_{\text{xor}}(2\tilde{d}_c - 1) + \frac{3}{2}E_{\text{add}}(q-1)(\tilde{d}_c - 1)\Bigg), \quad (3.11)$$

where $\tilde{d}_v = \frac{1}{n}\sum_{i=1}^{n} d_{v_i}$ and $\tilde{d}_c = \frac{1}{m}\sum_{j=1}^{m} d_{c_j}$ are respectively the average variable and check node degrees of the code. Note that for protographs, we have $\tilde{d}_v = (1-R)\tilde{d}_c$.

3.4.2 Memory energy model

In order to derive the memory energy model, we denote by E_{bit} the elementary energy consumption for writing one bit in memory. For an LDPC code of length N and rate R constructed from protograph S, the memory energy model is given by

$$E_m = \frac{L_N(\xi, p_e)N}{n}E_{\text{bit}}\left(\sum_{i=1}^{n} d_{v_i}(q+q_s) + (1-R)\left(\sum_{j=1}^{m}\left(2q + d_{c_j} - 2\right)\right)\right), \quad (3.12)$$

which can be simplified to

$$E_m = L_N(\xi, p_e)E_{\text{bit}}N\Big((q+q_s)\tilde{d}_v + (1-R)(\tilde{d}_c + 2q - 2)\Big). \quad (3.13)$$

The two energy models E_c and E_m depend on the SNR and BER targets ξ and P_e through the average number of iterations L_N. In addition, only the average degrees $\tilde{d}_v$ and $\tilde{d}_c$ of the protograph S explicitly appear in these energy models. The protograph S

Table 3.1 – Finite-length energy values of the protograph S for $\xi = 1.45dB$ and $N = 10^4$

Energy	$q = 5$	$q = 6$
Em	7.76×10^{-7} J	6.85×10^{-7}J
Ec	2.92×10^{-8} J	2.55×10^{-8} J

however also has an influence on the number L_N of iterations, see (7.8). In the considered architecture, there are as many "writes" as there are "reads" in the memory, therefore we only evaluate the "writes".

3.5 Numerical energy evaluation

In this section, we first use the memory energy model and the complexity energy model to evaluate the energy consumption of the protograph S of size 2×4:

$$S = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \end{pmatrix}$$

The objective of this section is to see how the energy consumption varies with the value of the SNR, maximum number of iterations L, and the quantization step q.

For illustrative purposes, we substitute the energy constants with rough estimates. Based on the estimate of 0.1pJ for a 32-bit addition reported in [77], we set $E_{\text{add}} = 3.13$ fJ, and since a 1-bit adder contains two XOR-2 gates, $E_{\text{xor}} = 1.56$ fJ. We base the storage energy on the estimate of 10 pJ for a 64-bit access from an 8KB cache, yielding an average of $E_{\text{bit}} = 0.156$ pJ.

Table (3.1) provides the energy values E_m and E_c for the two models for the protograph S at $\xi = 1.45dB$ and $N = 10000$, for $q = 5$ bits and $q = 6$ bits. As we can see, at this SNR, and because the decoder need more iterations, the protograph S consumes more energy at $q = 5$ bits compared to $q = 6$ bits. The value of q changes the energy consumption of the decoder in both the memory and the complexity model, and the optimal value of q depends on the protograph and also on the SNR, as we now illustrate.

Figure (3.2) and Figure (3.3), give the energy consumption evaluated with the two models, for the protograph S for $N = 10000$, $q = 5$ bits and $q = 6$ bits for different values of SNR. As we can see, the energy consumption decreases with the SNR, which is due to the fact that the decoder needs fewer iterations for the decoding process as

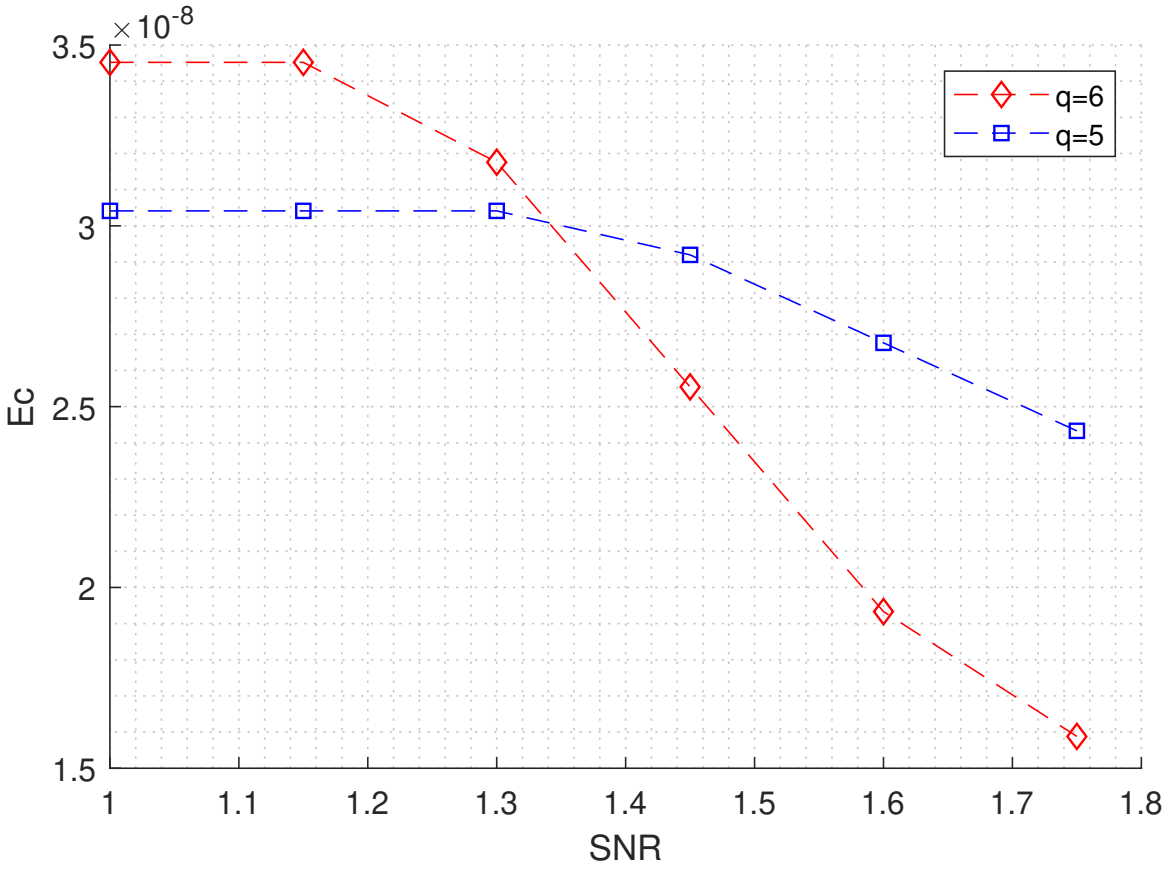

Figure 3.2 – Energy consumption of the protograph S evaluated using the complexity energy model for different value of SNR, for $q = 5$ and $q = 6$

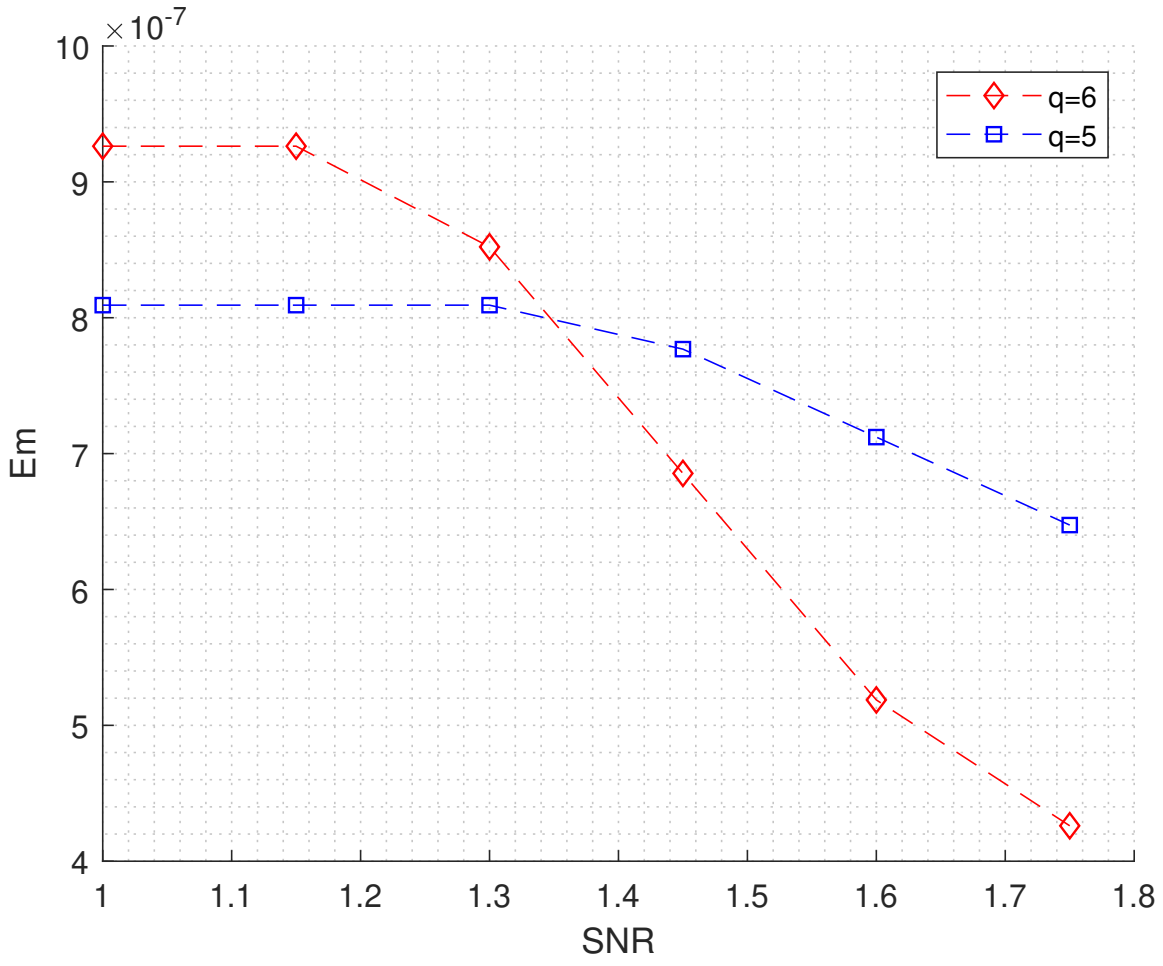

Figure 3.3 – Energy consumption of the protograph S evaluated using the memory energy model for different value of SNR, for $q = 5$ and $q = 6$

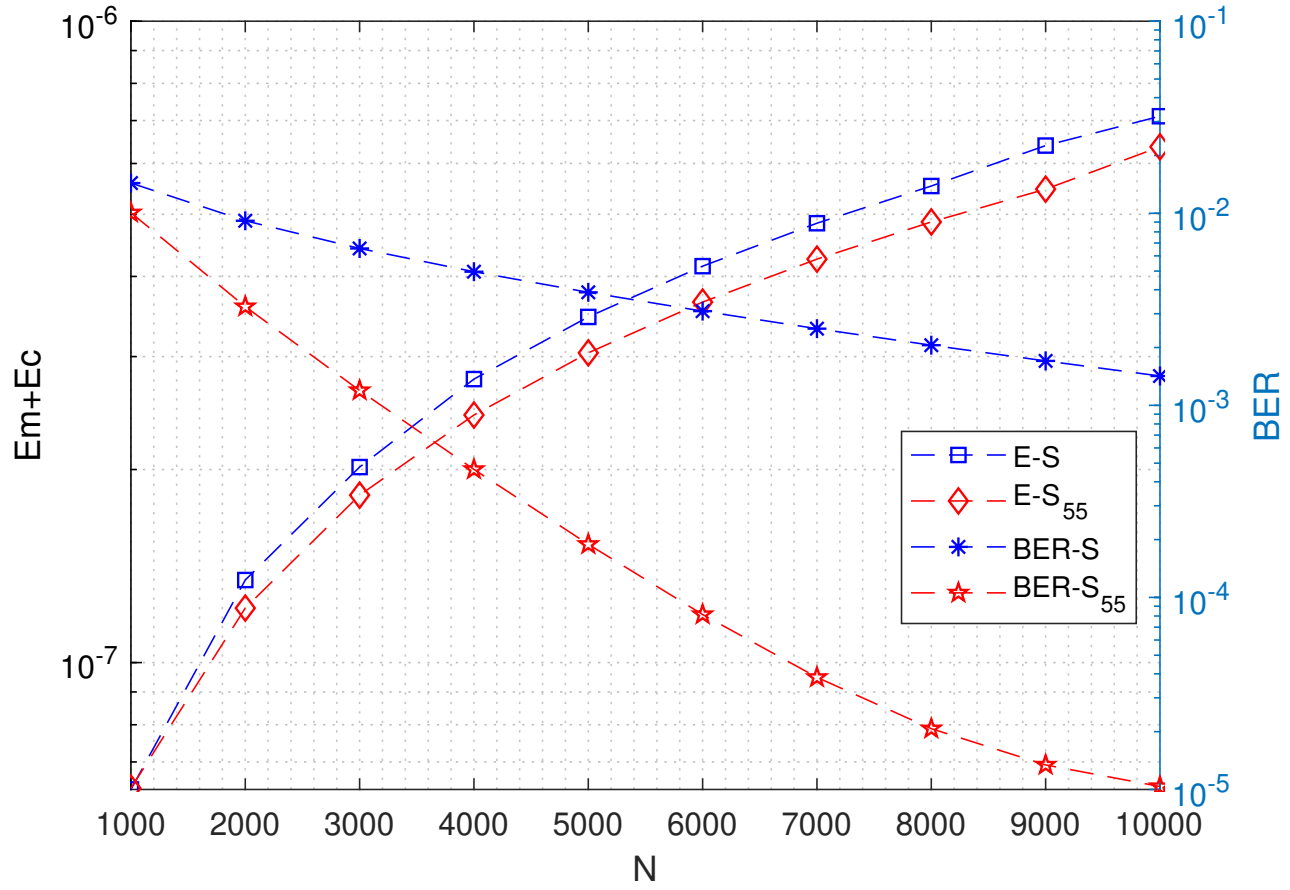

Figure 3.4 – Energy consumption and the BER of the protograph S and S_{55} evaluated using Finite length Density Evolution for different value of N, at SNR $\xi = 1.45$ dB and $q = 6$ bits

the SNR increases. We also remark that for SNR values lower than 1.45 dB, the decoder consumes less energy at $q = 5$ bits compared to $q = 6$ bits. It is worth noting that for such SNR values, the decoder uses the maximum number of iterations in both cases. So the minimum energy score is provided by the smallest quantization step. On the opposite, for SNR values higher than 1.45dB, the decoder consumes less energy for $q = 6$ bits because of the number of iterations.

We now compare the decoder energy consumption for two protographs. Figure (3.4), gives the total energy consumption $(E_c + E_m)$ for protograph S and protograph S_{55} given by,

$$S_{55} = \begin{pmatrix} 1 & 0 & 5 & 1 \\ 3 & 2 & 1 & 1 \end{pmatrix}$$

Figure (3.4) also gives BER evaluated using finite length Density Evolution presented in Section 2.2.2 for different values of N, at SNR $\xi = 1.45$ dB and $q = 6$ bits. As we can see, the energy consumption increases with the code length N. In addition, while the two protographs show close energy consumption values, they get very different BER performance. In particular, protograph S shows a very degraded performance compared to S_{55}. This shows the importance of taking into account the two criterion (energy and performance) during the code design process.

Finally, we see that many parameters have a major influence on the energy consumption: the quantization step q, the degree values d_v and d_c of the protograph, the code length N, and the average iteration number L. In the next chapter, we optimize these parameters so as to minimize the decoder energy consumption while guaranteeing a certain performance criterion.

3.6 Conclusion

In this chapter, we provided two energy models in order to estimate the energy consumption of the quantized Min-Sum decoder. The first model called the complexity model, calculates the total number of operations in the decoding process. The second model, called the memory energy model, calculates the total number of bits stored in memory. In the next chapters, we provide optimization methods so as to select the code and decoder parameters that minimize the decoder energy consumption for a target decoding performance. We start in the next chapter with a method to optimize the protograph when all the other parameters are fixed.

Chapter 4

Protograph optimization for energy minimization

In the previous chapter, we presented the quantized Min-Sum decoder architecture considered in this work, and we introduced two energy models in order to evaluate the energy consumption of the decoder. In this chapter, we start by formulating a general optimization problem that corresponds to minimizing the decoder energy consumption while satisfying a given performance criterion. In this formulation, the energy consumption is minimized with respect to the code and decoder performance (protograph and iteration number) that participate to the energy models. To evaluate the energy consumption, we use the energy models presented in Section 7.5, and we rely on the finite-length Density Evolution (2.2.2) to evaluate the decoding performance. We then propose an optimization method to solve this problem, using the Differential Evolution [78] algorithm.

4.1 Optimization Problem

We want to find the protograph S minimizing the decoder energy consumption while maintaining a certain level of decoding performance. In order to simplify the optimization problem, we first assume that the code rate R, the codeword length N, the number q of quantization bits, and the dimensions m, n of the protograph are fixed, and we optimize the protograph S. Then, in order to specify the decoding performance, we set a target SNR ξ and a target performance to be achieved at that SNR. Once these parameters are set, we formulate the optimization problem as

$$\min_{\boldsymbol{S}} E(\boldsymbol{S}) \quad \text{s.t.} \quad \mathcal{P}(\xi) < \mathcal{P}_{\max} \tag{4.1}$$

In (7.11), the energy function E can be given either by the complexity energy model (7.9), by the memory energy model (7.10), or by a weighted combination of both. $\mathcal{P}(\xi)$ represents the decoding performance at the SNR value ξ and $\mathcal{P}_{\max}$ is the target performance to be

achieved at a SNR value ξ. In what follows, we consider two examples of the decoding performance $\mathcal{P}$.

4.1.1 BER performance criterion

In the first example, the performance criterion is given by the BER. In this case, in order to specify the decoding performance, we set a target SNR ξ and a target error probability p_e to be achieved at that SNR. The optimization problem (7.11) then becomes:

$$\min_{\boldsymbol{S}} E(\boldsymbol{S}) \quad \text{s.t.} \quad p_{e,\text{opt}}(\xi) < p_{e,\max} \tag{4.2}$$

where

$$p_{e,\text{opt}}(\xi) = \min_{\alpha,\lambda} p_{e_N}(\xi)$$

and $p_{e_N}(\xi)$ is the BER defined in (7.4). Recall that α and λ are the scaling parameter and the offset parameter of the quantized Min-Sum decoder, see Section(7.3). Note that $p_{e_N}(\xi)$ also depends on $\boldsymbol{S}$, and L, although it is is not explicitly stated in order to simplify the notation.

4.1.2 FER performance criterion

In the second optimization problem, the performance criterion is given by the FER. In this case, the performance criterion is defined as a condition on the maximum FER $B_{\max}$ that can be tolerated at SNR value ξ. As a result, the optimization problem can be expressed as

$$\min_{S} E(\boldsymbol{S}) \quad \text{s.t.} \quad B_N(\xi) < B_{\max}, \tag{4.3}$$

where $B_N(\xi)$ is the FER estimated at length N and defined in (7.5). In the above formulation, the terms E and $B_N(\xi)$ are evaluated for parameters $\overline{\gamma}$ and $\overline{\alpha}$ such that

$$\overline{\gamma}, \overline{\alpha} = \arg\min_{\alpha,\lambda} B_N(\xi) \tag{4.4}$$

Note that the optimal values of α and λ strongly depend on the considered protograph.

4.1.3 Discussion on the optimization method

The above two optimisation problems involve discrete matrices S to be optimized. These two optimisation problems are non-continuous, non-linear, and non-differentiable.

In addition, the BER $p_{eN}(\xi)$, the FER $B_N(\xi)$ and the average number of iterations $L_N(\xi)$ do not have explicit analytical expressions, although they can be evaluated numerically. A common solution for protograph optimization [79] consists of adapting a genetic algorithm called Differential Evolution [78] which was initially proposed for the optimization of continuous variables. However, while previous works [79] consider protograph optimization for performance only, the main difficulty in our case is to address the tradeoff between energy consumption and performance. In particular, Differential Evolution is not well-suited for constrained optimization, as the algorithm should be initialized inside the feasible set, that is the set of protographs that satisfy the constraints $p_{eN}(\xi) < p_{e,max}$ in (4.2) or $B_N(\xi) < B_{max}$ in (4.3).

We now describe the generic Differential Evolution algorithm. We then show how it can be applied to the previous optimization problems.

4.2 Differential Evolution algorithm

Differential Evolution [78] is a genetic algorithm that was initially introduced for nonlinear and non-differentiable continuous space functions. It is a parallel search method that uses vectors with D parameters. We now describe the generic standard unconstrained version of the Differential Evolution algorithm. Denote by $\mathcal{F}$ the function to be minimized. The Differential Evolution algorithm first generates randomly an initial population $\mathcal{G}_1$ of NP vectors $\boldsymbol{x}_1^{(1)}, \cdots, \boldsymbol{x}_{\mathrm{NP}}^{(1)}$, each of length D. Then, in order to generate a new population $\mathcal{G}_{i+1}$ of NP vectors from the previous population $\mathcal{G}_i$, Differential Evolution relies on two functions called Mutation and Crossover. These two functions realize NP random combinations $\boldsymbol{u}_1^{(i+1)}, \cdots, \boldsymbol{u}_{\mathrm{NP}}^{(i+1)}$ of the vectors $\boldsymbol{x}_1^{(i)}, \cdots, \boldsymbol{x}_{\mathrm{NP}}^{(i)}$ of the population $\mathcal{G}_i$. The population $\mathcal{G}_{i+1}$ is then constructed from a given selection rule that ensures that a newly generated vector $\boldsymbol{u}_k^{(i+1)}$ is included into the population only if it improves the optimization criterion. We now describe the mutation, crossover, and selection operations which are used to construct the new population $\mathcal{G}_{i+1}$.

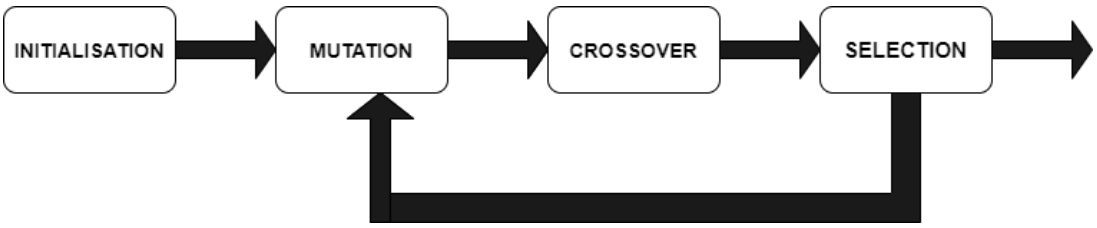

Figure 4.1 – Differential Evolution algorithm elements

4.2.1 Mutation

Let us consider a vector $\boldsymbol{x}_k^{(i)}$ of the population $\mathcal{G}_i$, where $k \in \{1, \cdots, \mathrm{NP}\}$. A mutant vector $\boldsymbol{v}_k^{(i+1)}$ is first generated as follows:

$$\boldsymbol{v}_k^{(i+1)} = \boldsymbol{x}_{r_1}^{(i)} + F.(\boldsymbol{x}_{r_2}^{(i)} - \boldsymbol{x}_{r_3}^{(i)}) \tag{4.5}$$

where r_1, r_2, $r_3 \in \{1, \cdots, \mathrm{NP}\}$ are random indexes, and must be different from each other and from the index k. The parameter $F \in [0\ , 2]$ is a constant factor that controls the amplification of the difference $(\boldsymbol{x}_{r_2}^{(i)} - \boldsymbol{x}_{r_3}^{(i)})$.

4.2.2 Crossover

Then, a new vector $\boldsymbol{u}_k^{(i+1)} = (u_{1k}^{(i+1)},\ u_{2k}^{(i+1)}, \ldots, u_{Dk}^{(i+1)})$ of length D is generated from a combination of the mutant vector $\boldsymbol{v}_k^{(i+1)}$ and of the original vector $x_k^{(i+1)}$ in order to increase the diversity in the new candidate vector. The trial vector $\boldsymbol{u}_k^{(i+1)}$ is formed as follow:

$$u_{jk}^{(i+1)} = \begin{cases} v_{jk}^{(i+1)} & \text{if } (\nu_j \leq \mathrm{CR}) \text{ or } j = r_i \\ x_{jk}^{(i)} & \text{if } (\nu_j > \mathrm{CR}) \text{ and } j \neq r_i \end{cases} \tag{4.6}$$

where $j \in \{1, 2, \ldots, D\}$, and $k \in \{1, \cdots, \mathrm{NP}\}$. The parameter $\nu_j \in [0, 1]$ is a randomly generated number. The parameter CR is the crossover constant $\in [0, 1]$ and it is fixed for all the algorithm. Finally $r_i \in \{1, 2, \ldots, D\}$ is a randomly generated index to make sure that the trial vector $\boldsymbol{u}_k^{(i+1)}$ has at least one component from the mutant vector $\boldsymbol{v}_k^{(i+1)}$.

4.2.3 Selection

Finally the trial vector $\boldsymbol{u}_k^{(i+1)}$ is compared to the population vector $\boldsymbol{x}_k^{(i)}$, in order to decide whether the trial vector $\boldsymbol{u}_k^{(i+1)}$ should replace $\boldsymbol{x}_k^{(i)}$ into the population $\mathcal{G}_{i+1}$. For

this, the function $\mathcal{F}$ to be minimized is used as the selection criterion $\forall k \in \{1, \cdots, \mathrm{NP}\}$ as:

$$\boldsymbol{u}_k^{(i+1)} = \begin{cases} \boldsymbol{u}_k^{(i+1)} & \text{if } \mathcal{F}(\boldsymbol{u}_k^{(i+1)}) < \mathcal{F}(\boldsymbol{x}_k^{(i)}) \\ \boldsymbol{x}_k^{(i)} & \text{otherwise.} \end{cases} \tag{4.7}$$

4.2.4 Algorithm termination

Finally, after I iterations, the algorithm outputs the vector $\boldsymbol{x}_k^{(I)}$ which minimizes the function $\mathcal{F}$. Although this algorithm does not give guarantees on the quality of the retained solution, is ensures that the criterion is reduced from iteration to iteration. The result of this optimization depends on the parameters fixed at the beginning of the algorithm (F, CR, NP, D). In order to improve the performance of the algorithm, it is recommended in [78] to choose $5D < \mathrm{NP} < 10D$, $F = 0.5$ and $\mathrm{CR} = 0.1$. To speed up the convergence of the algorithm a larger CR value is recommended $\mathrm{CR} = 0.9$ or $\mathrm{CR} = 1.0$.

4.3 Application of Differential Evolution to Protograph Optimisation

In order to solve the optimization problems (7.11) and (4.3), we need to adjust the Differential Evolution algorithm so as to consider discrete protographs that satisfy some constraints related to LDPC code construction. For this, the following changes are required. First, the populations $\mathcal{G}_i$ is composed by protographs in a vectorized form.Second, during the population initialization and when applying the Mutation and Crossover operations, the components of each vector of the population are rounded to the closest integer values, and forced to be between 0 and a given value $S_{\max}$. Then, for a protograph to be included into a population, it is necessary that $d_{v,\min} = \min_i(d_{v_i}) > 1$ and $d_{c,\min} = \min_i(d_{c_i}) > 1$ in order to avoid degree 0 and degree 1 nodes. In particular, we eliminate degree 1 variable nodes that could show good performance under density Evolution, but a poor minimum distance, which would affect the code performance at finite length [80]. We also choose a small value for S_{max} (usually $S_{max} = 6$) in order to avoid high degrees which would make it difficult to construct a good finite-length code without short cycles.

4.3.1 Protograph optimisation for performance only

In this section, we consider protograph optimization for performance only, and we do not take into consideration the energy consumption. The performance criteria may be given by one of the following two options:

(i) The minimum SNR value that allows to achieve a given error probability.
(ii) The error probability at a given SNR value.

When considering criterion (i), the selection step prioritizes the protograph with better threshold defined as the minimum SNR value to achieve a given error probability p_e. The selection function is then given by:

$$\mathcal{F}(S) = \mathrm{SNR}_{op}(S, p_e) \tag{4.8}$$

where

$$\mathrm{SNR}_{op}(S, p_e) = \min_{\alpha,\lambda} \mathrm{SNR}(S, p_e)$$

. The selection step (4.7) checks whether the protograph $\boldsymbol{u}_k^{(i+1)}$ has a lower threshold than $\boldsymbol{x}_k^{(i)}$ in order to add it in the next generation.

Criterion (ii) is usually used when the objective is to reduce the running time of the algorithm. In this case, the selection function is given by:

$$\mathcal{F}(S) = p_{e_\infty,opt}(S, \xi^*) \tag{4.9}$$

where

$$p_{e_\infty,opt}(\xi^*) = \min_{\alpha,\lambda} p_{e_\infty}(\xi^*)$$

The selection step (4.7) checks whether the protograph $\boldsymbol{u}_k^{(i+1)}$ has a lower error probability than $\boldsymbol{x}_k^{(i)}$ at SNR value ξ in order to add it in the next generation. Figure 4.2 and Figure 4.3 show the steps used for the protograph selection in the performance only optimization method.

In the following, we consider protographs optimization for performance only as baselines to evaluate the energy gains raised by the protographs optimized for reduced energy consumption.

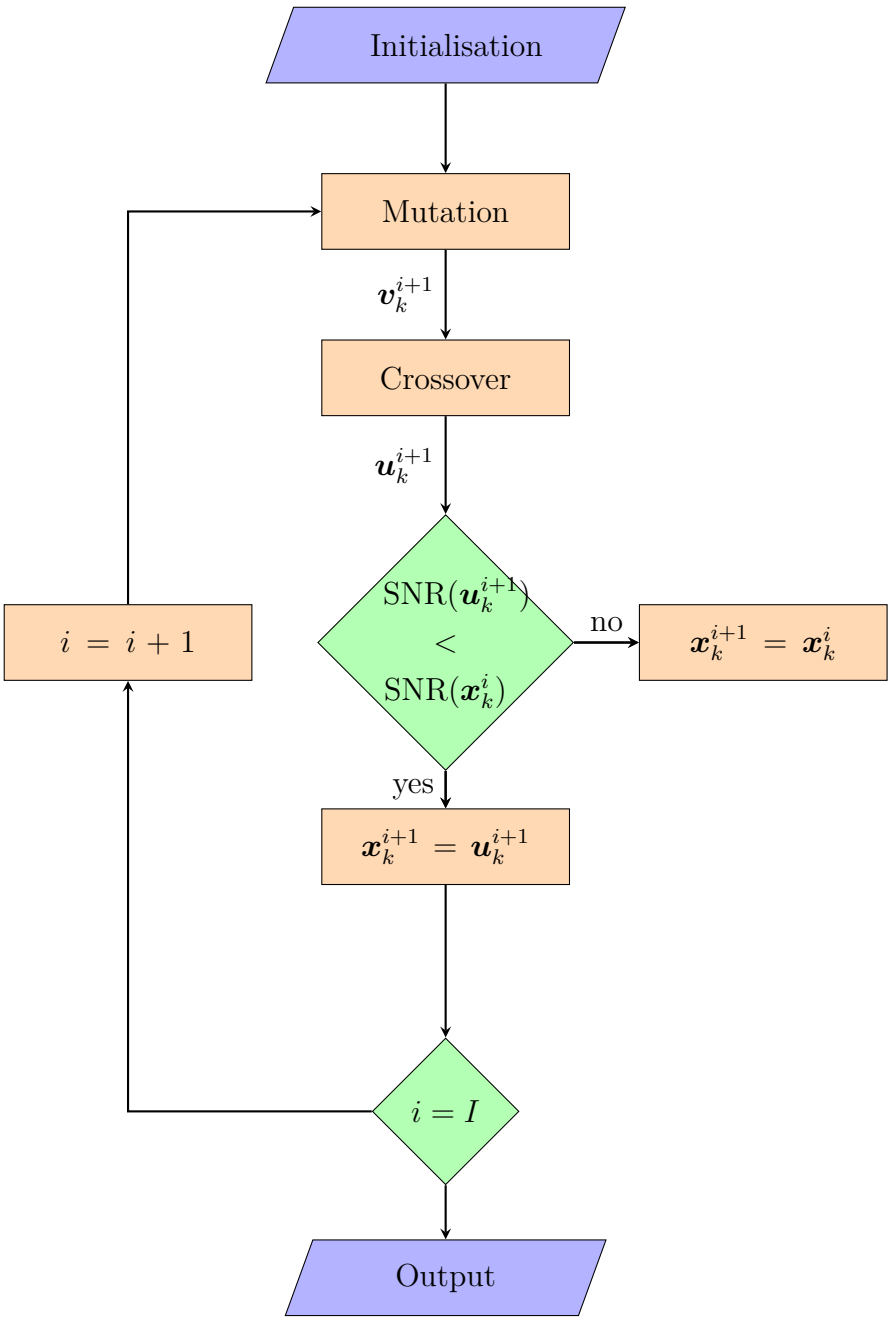

Figure 4.2 – The selection process for SNR based performance only protograph optimization

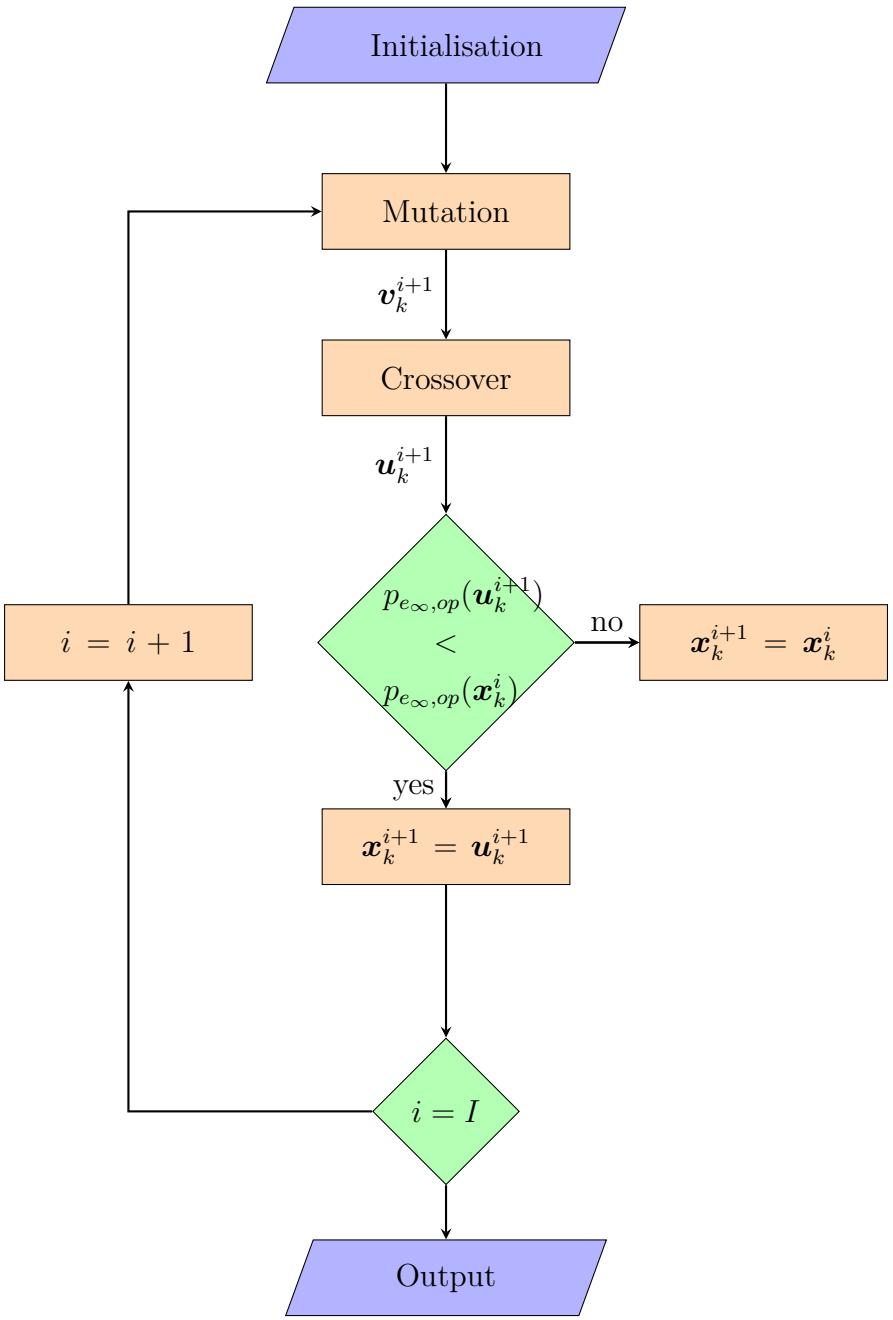

Figure 4.3 – The selection process for BER based performance only protograph optimization

4.3.2 Energy optimization with BER performance criterion

We now want to solve optimisation problem (4.2) with the performance constraint on the BER. In this case, the selection criterion is given by the energy consumption of the

decoder evaluated using the complexity energy model or the memory energy model. When applying the selection step we use the function:

$$\mathcal{F} = E(S, \xi). \tag{4.10}$$

Ideally, in order to take the constraint into account, before applying the selection step (4.7), we check whether the protograph $\boldsymbol{u}_k^{(i+1)}$ verifies the performance constraint $p_{e,\text{opt}}(\xi) < p_{e,\max}$. However, computing $p_{e,\text{opt}}(\xi)$ is computationally expensive, because of the integral in (7.4). This is why we introduce a second SNR value ξ^* and only verify a condition on the asymptotic error probability p_{e_∞} calculated by standard Density Evolution, that is:

$$\min_{\alpha,\lambda} p_{e_\infty}(\xi^*) < p_{e_\infty,\max}.$$

If protograph $u_k^{(i+1)}$ satisfies the above constraint, then we check if the protograph satisfies the finite length performance constraint, we then compute the minimum number of iterations $L_N^*(\xi)$ (see (7.8)) for $\boldsymbol{u}_k^{(i+1)}$. Finally, we apply the selection step.

4.3.3 Energy optimization with FER performance criterion

We now describe the method we developed to optimize the energy consumption under a FER performance criterion. First, for the two considered energy models, given that the parameters R, N, q are fixed, the optimization problem (4.3) is equivalent to:

$$\min_S \left(f(\tilde{d}_v) L_n(\xi) \right) \quad \text{s.t.} \quad B_N(\xi) < B_{\max}, \tag{4.11}$$

where $f(\tilde{d}_v) = \lceil \log_2(\max d_v + 1) \rceil \tilde{d}_v$. In addition, by replacing $L_N(\xi)$ by its expression in (7.8), we further show that the optimization problem (7.11) is equivalent to

$$\min_{S,\lambda,\alpha} \left(f(\tilde{d}_v) \sum_{\ell=1}^{L} B_N^{(\ell-1)}(\xi) \right) \quad \text{s.t.} \quad \overline{B}_N(\xi) < B_{\max} \tag{4.12}$$

This formulation shows that the two key criterion for energy optimization are the average variable node degree $\tilde{d}_v$ and the FER performance. Therefore, improving the FER performance of the decoder may lower its energy consumption. We now exploit the equivalent formulation (4.12) to propose an efficient optimization method based on Differential Evolution.

In order to apply the Differential Evolution algorithm, we start from the optimization problem (4.12). As a criterion for protograph selection at each iteration, we consider a function $\mathcal{F}$ defined as

$$\mathcal{F}(S) = \left(f(\tilde{d}_v) \sum_{\ell=1}^{L} B_N^{(\ell-1)}(\xi) \right), \tag{4.13}$$

In addition, during the I iterations of the algorithm, we consider a relaxed constraint on the performance, that is that the protograph threshold obtained from standard DE must be larger than a certain value $\xi^\star$ in order for this protograph to be included in the population.

$$\min_{\alpha,\lambda} \mathrm{SNR}(S) < \xi^*$$

This relaxed constraint allows to increase the feasible set when runing Differential Evolution. Finally, after the I iterations, we select the protograph that both minimizes the function $\mathcal{F}$ and satisfies the FER constraint $B_{\max}$. Figure 4.4 and Figure 4.5 show the steps used for the protograph selection in the BER based energy optimization method and the FER based energy optimization method respectively.

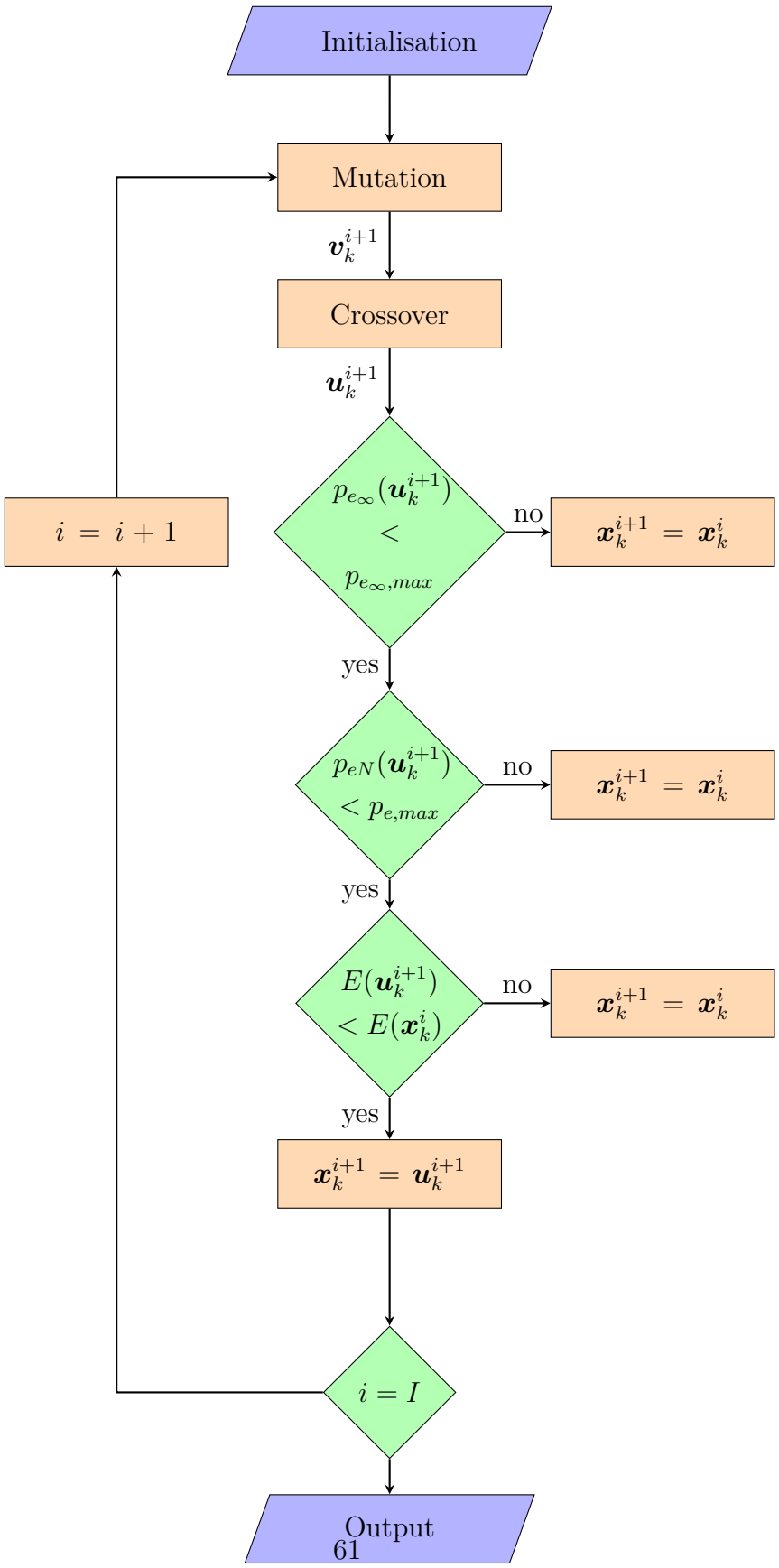

Figure 4.4 – The Selection process for Energy optimization with BER performance criterion

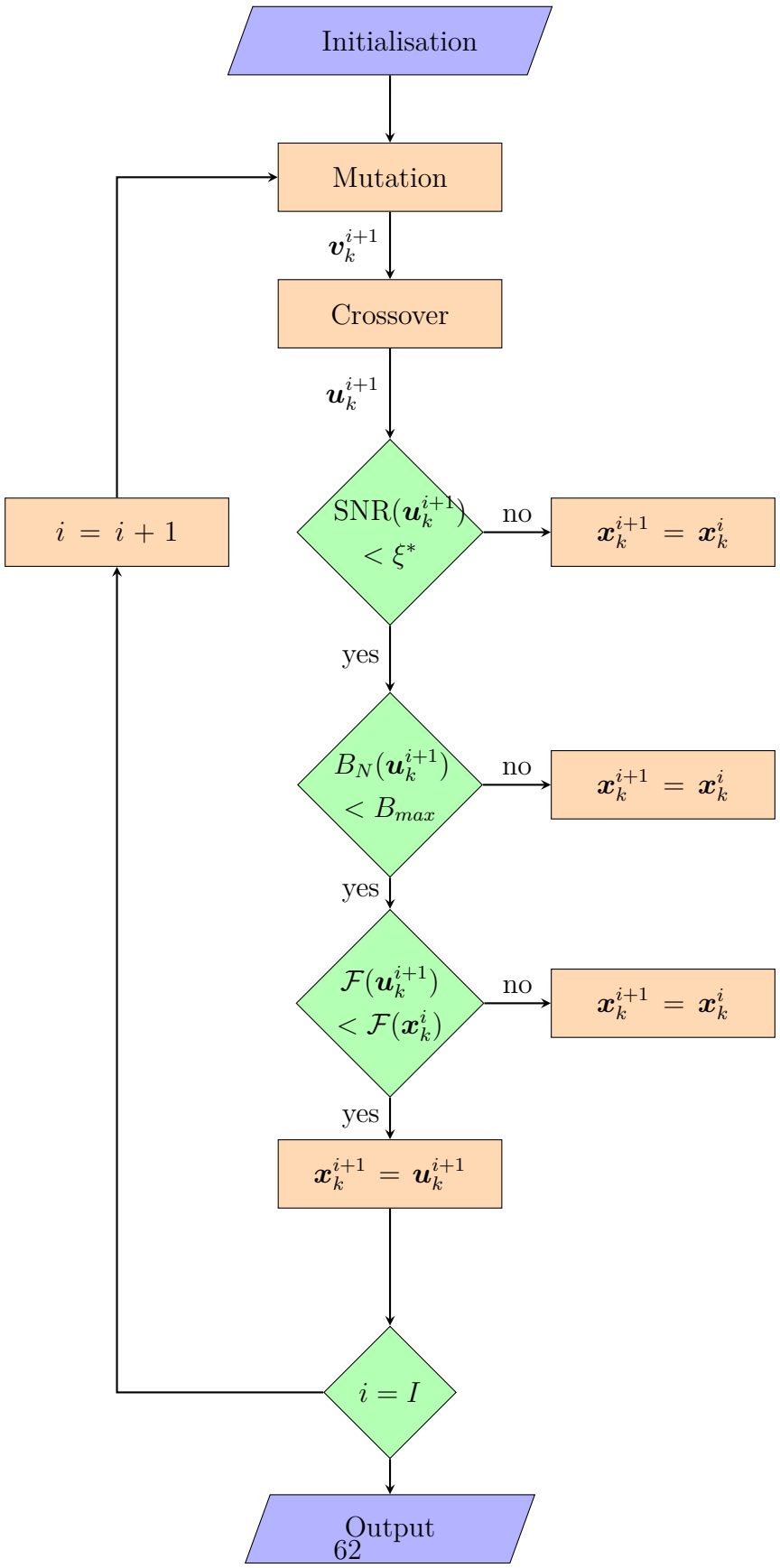

Figure 4.5 – The Selection process for Energy optimization with FER performance criterion

Table 4.1 – Infinite-length thresholds and finite-length energy values of the protographs for $\xi = 1.45$dB and $N = 10000$.

Protograph	Threshold	E_c	E_m
$S_{\text{opt}} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 0 & 4 & 1 \end{bmatrix}$	1.18 dB	19.1 nJ	508 nJ
$S_0 = \begin{bmatrix} 2 & 1 & 3 & 2 \\ 5 & 1 & 1 & 0 \end{bmatrix}$	1.15 dB	28.7 nJ	733 nJ
$S_c = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{bmatrix}$	1.20 dB	21.9 nJ	585 nJ
$S_m = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 4 \end{bmatrix}$	1.21 dB	21.9 nJ	585 nJ

4.4 Simulation results

We now provide results for the optimization methods described in Section 4.3.1, Section 4.3.2 and Section 4.3.3. For the optimization, we consider protographs of size $m = 2$, $n = 4$ with $S_{max} = 6$, code parameters $N = 10^4$, $R = 0.5$, and decoder parameters $q = 6$, $L = 50$. In the optimization problem (4.2), $p_{e,\max} = 10^{-3}$ is set as the maximum error probability at an SNR $\xi = 1.45$dB and $p_{e_\infty,\max} = 10^{-6}$ is set as the maximum error probability at an SNR $\xi = 1.25$dB. In case of the optimization problem (4.3), for the relaxed performance constraint, we considered $\xi^* = 1.25$dB, and for the final performance constraint, we considered $B_{\max} = 10^{-2}$ at an SNR value $\xi = 1.45$dB. In this Section we use the same energy values presented in Chapter 2. $E_{\text{add}} = 3.13$ fJ, $E_{\text{xor}} = 1.56$ fJ and $E_{\text{bit}} = 0.156$ pJ. These values do not affect the optimization result.

Table 7.1 shows protographs obtained by the optimization method proposeds in Section 4.3.1, Section 4.3.2 and Section 4.3.3. The protograph S_0 was optimized without the energy criterion (Section 4.3.1). Protograph S_m and S_c where obtained from the Energy optimization method with BER performance criterion (Section 4.3.2). S_m was obtained by considering the memory energy model, and S_c was obtained from the complexity energy model. Finally protograph S_{opt} was generated using the Energy optimization method with FER performance criterion (Section 4.3.3). The energy evaluated using the memory model is denoted E_m, and E_c is the energy evaluated using the complexity model. As we can see, S_0 achieves a better SNR threshold, but S_c and S_m satisfy the SNR threshold criterion and consume less energy based on both energy models. In the other hand, the protograph S_{opt} satisfies the performance criterion of the optimization problem (4.3), with

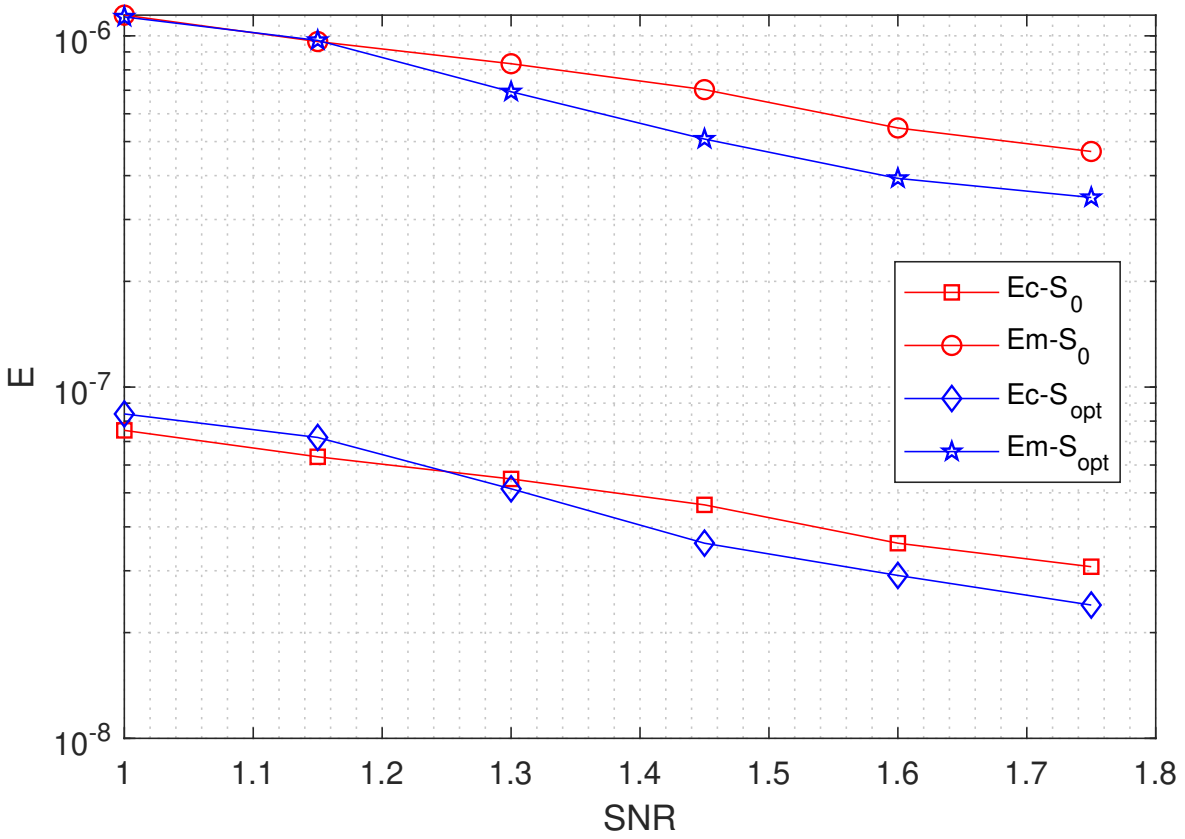

Figure 4.6 – Energy consumption of the protographs S_0 and S_{opt}

a FER value $B_N^L(\xi) = 0.0049$ at $\xi = 1.45dB$.

We then observe that S_0 shows a higher energy consumption, which is due to the fact that this protograph was optimized for performance only. On the opposite, S_{opt} has the lowest energy consumption, with a gain of approximately 15% compared to S_0. To verify these results, Figure 4.6 shows the energy values with respect to SNR for the two energy models, for protographs S_{opt} and S_0. The figure confirms that protograph S_{opt} has lower energy consumption than S_0 at any considered SNR.

Figure 4.7 and Figure 4.8 compare the BER and FER values for S_0, S_m, S_c and S_{opt}, obtained from both the finite-length method of Section2.2.2 and from Mont Carlo simulations. We first observe that the method proposed in Section 2.2.2 provides a good approximation of the decoder performance. Second, we see that although S_0 has a better performance in the waterfall, it shows an important error floor at higher SNR values, which can be observed from the FER curve.

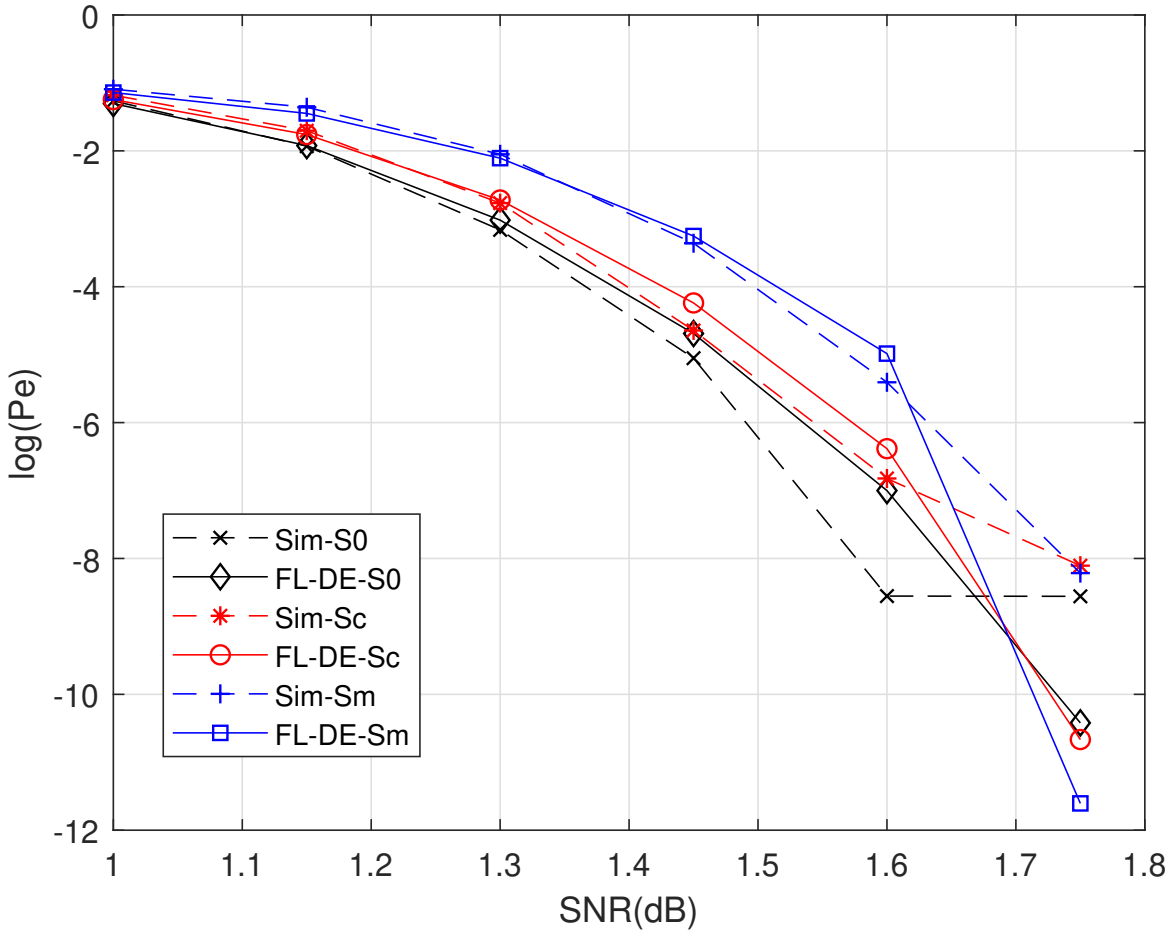

Figure 4.7 – Bit error rate of codes generated from protographs with energy criteria (S_c, S_m) and without the energy criteria (S0).

Finally, the proposed energy optimization methods provides good results in term of energy score and performance. As we can see from Table 7.1, the protographs S_m and S_c have similar energy scores, and it is due to the fact that, they have the same degree distribution and similar performance where they use the same number of iterations. In the next chapter, we will see how optimizing the decoder parameters can differentiate the decoding performance and the energy consumption of these two protographs.

4.5 Conclusion

In this chapter, we proposed an optimization method to minimize the decoder energy consumption with respect to the protograph, while satisfying a given decoding performance constraint. Simulation results showed that, using the energy based optimization method can provide protographs that minimizes the energy consumption by 15% compared to performance only optimized protographs. In the next chapter, we consider that the protograph is fixed, and we optimize the other code and decoder parameters (code length, number of quantization levels, etc.) so as to minimize the energy consumption while respecting the performance criterion.

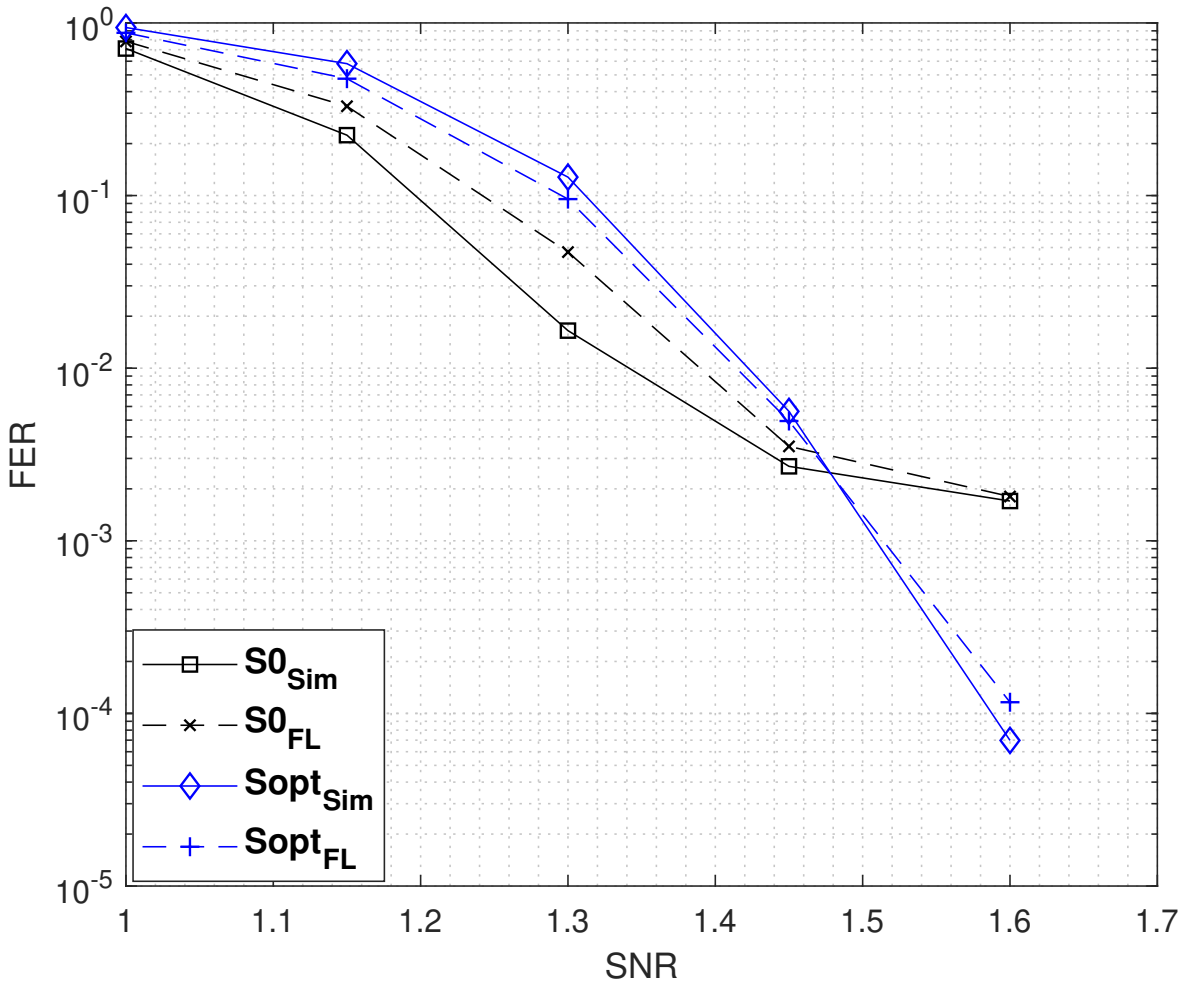

Figure 4.8 – Performance comparison between codes constructed from protographs S_0 and S_{opt}. The FER performance was evaluated from finite-length DE (FL) and from Monte Carlo simulations (sim)

.

Chapter 5

Energy optimization for faulty min-Sum decoders

In the previous chapters, we modeled the energy consumption according to the code and decoder parameters, and we proposed a protograph optimization only. Now we seek to optimize the other parameters, in particular the length of the code and the number of quantization bits. Energy consumption can also be greatly reduced by aggressive voltage scaling, but this may introduce faults in the computation operations realized on the circuit. Therefore, in this part, we propose to consider a circuit architecture that tolerates faults in the computation operations and memories of the circuit [81]. We update the memory energy model in order to apply it to a faulty decoder. We then propose a method to minimize decoder energy consumption , by optimizing the code length, the quantization level and the noise level.

5.1 State of the art

The performance of LDPC decoders implemented on faulty hardware was widely studied in the literature. In [41] the authors assume that the LDPC decoder is subject to both transient and permanent errors. Transient errors make faulty gates or memory units provide an erroneous output from time to time with a non-zero probability. Permanent errors make gates and memories provide all the time the same output. When dealing with energy consumption issues, we are mainly concerned with transient errors.

In this sense, [82] was the first to investigate the effect of transient noise on standard iterative decoders for (LDPC) codes. The authors in [82] proposed a density evolution method to take into account the effect of noise in LDPC decoders, and that the method applies to both Gallager-B and Belief-Propagation decoders.

Several models were proposed to represent the effect of noise in the decoder. For instance, [83] models the effect of timings violations caused by a reduced supply voltage

and increased clock frequency, while simultaneously capturing the energy consumption. Also [44] studies the effect of timing errors in the Gallager-B decoder. For this model the authors provide a theoretical analysis of the performance of LDPC decoders under timing errors. Based on this analysis, when the number of iterations goes to infinity, the error probability of the decoder with timing errors converge to the error probability of the error-free decoder.

In the other hand, [52] considers FAIDS decoders. For those decoders, it proposes a rigorous method for the analysis and the design of decoding rules robust to transient errors introduced by the hardware. The Belief-Propagation decoders was also considered in [84]. Finally, the Min-Sum decoder has been studied a lot with noise. For instance, [85] considers asymmetric noise models for Offset Min-Sum decoders. Also [45] studies faulty Min-Sum decoding to unreliable memory. Since memory elements have been shown to be the first point of failure in digital circuits and since memories represent is the largest part in term of area of most hardware LDPC decoders. [45] introduces a bit-level fault model for unreliable memory reads.

None of the previous works relate energy consumption and the amount of faults introduced, and that is what we will deal with in this chapter. The objective is to find the best compromise between decoder circuit energy consumption and LDPC decoding performance under circuit faults.

5.2 Faulty Min-Sum decoder

Chapter 1 described the behavior of a deterministic min-Sum decoder, under a specific hardware architecture. We now assume that faults are introduced in this architecture. In the faulty decoder we now present, we assume that faults are introduced in the memory units, since memories are responsible for a large part of the decoder energy consumption [73]. Variable Node and Check Node operations 2.1.3 then become:

$$\widetilde{\beta}_i^{(\ell)} = \Delta(r_i) + \sum_{j \in \mathcal{N}_{v_i}} \widetilde{\gamma}_{j\to i}^{(\ell-1)} + \mathcal{B}, \tag{5.1}$$

$$\widetilde{\beta}_{i\to j}^{(\ell)} = \widetilde{\beta}_i^{(\ell)} - \widetilde{\gamma}_{j\to i}^{(\ell-1)} \tag{5.2}$$

$$\widetilde{\gamma}_{j\to i}^{(\ell)} = \left(\prod_{i' \in \mathcal{N}_{c_j} \setminus \{i\}} \operatorname{sgn}\left(\widetilde{\beta}_{i'\to j}^{(\ell)}\right) \right) \times \max\left[\min_{j' \in \mathcal{N}_{c_j} \setminus \{i\}} \left|\widetilde{\beta}_{i'\to j}^{(\ell)}\right| - \lambda, 0 \right] + \mathcal{B}_{j\to i}^{(\ell)}, \tag{5.3}$$

where $\mathcal{B}_i^{(\ell)}$ is a random variable that represents the noise introduced in the memory. The message equation (7.13) is not affected by noise because $\widetilde{\beta}_{i\to j}^{(\ell)}$ is not stored in memory.

The random variable $\mathcal{B}_i^{(\ell)}$ represents the noise at the quantized symbol level. This random variable can equivalently be represented on q bits $(\mathcal{B}_{i,1}, \cdots, \mathcal{B}_{i,q})$. We assume that the $\mathcal{B}_{i,k}$ are independent and identically distributed, and that each $\mathcal{B}_{i,k}$ follows a Bernoulli distribution with parameter ϵ such that $\mathbb{P}(\mathcal{B}_{i,k} = 1) = \epsilon$. It is the same model as in [82] , but it is applied in a specific way for our architecture.

5.3 Energy Model

In Chapter 2 , we introduced an energy model to estimate the energy consumption due to writes in memory during the decoding process. Then, as proposed in [86], the memory failure parameter ϵ and the energy e in a memory unit can be related as :

$$\epsilon = \epsilon_0 \exp\left(-ce_g\right) \tag{5.4}$$

where $e_g = e^{\theta}$, e is the gate energy, $\epsilon_0, \theta \in (0, 1]$ and $c > 0$. We consider this simple energy-vs-noise model in order to propose a first energy optimization method for Min-Sum decoders.

In the memory energy model defined in (7.10), E_{bit} was the elementary energy consumption for writing one bit in memory. In this chapter, in order to connect the value of E_{bit} to the noise parameter ϵ, we write

$$E_{\text{bit}} = e_g E_0 \tag{5.5}$$

In this expression, e_g represents the normalized energy given from (7.15). We further assume that e_g takes values in $[0, 1]$, where $e_g = 0$ means that the device does not consume any energy, and $e_g = 1$ means that the device operates with the nominal energy E_0. For $e_g = 0$ we consider a failure probability of $\frac{1}{2}$, which gives that $\epsilon_0 = \frac{1}{2}$. The values of c and E_0 both depend on the considered technology. For instance, for a typical 65nm SRAM cell, the failure probability at a nominal voltage is 10^{-7} [7], which gives $c = 12$. The value of E_0 is estimated based on [77], where, 10 pJ is the storage energy for a 64-bit access from a 8Kb cache, which gives $E_0 = 0.156$ pJ.

Finally, since, in the following, among other parameters, we want to optimize the codeword length N, we consider the memory energy model E_m of (7.10) normalized

by the number of information bits K. As a result, the following energy model will be considered in the optimization:

$$\mathcal{E} = \frac{E_m}{K} = \frac{L_N(\xi)}{Rn} e_g E_0 \left(\sum_{i=1}^{n} d_{v_i} (q + q_s) + (1 - R) \left(\sum_{j=1}^{m} \left(2q + d_{c_j} - 2 \right) \right) \right). \tag{5.6}$$

This equation depends on the parameters q, e_g, etc, that will be optimized. The average iteration L_N is quite important in this equation, it also depends on these parameters, including the code length N.

5.4 Energy optimization problem

We now propose an optimization method to minimize the decoder energy consumption with respect to parameters q, e_g and N, while satisfying a certain performance criterion. We consider that the protograph is fixed. In this part, we first formulate the optimization problem, and then described the optimization method which relies on alternating minimization.

The performance criteria is given by the BER defined as in 4.3.2, where we fix an SNR value ξ and a target error probability $p_{e,max}$ to be reached at that SNR. These are the same functions to be optimized, except that a noisy decoder is considered, and different parameters are optimized. Then, for simplicity, we assume that the code rate R and the protograph S are fixed. We propose to minimize the energy consumption with respect to the quantization level q, the noise parameter ϵ, and the codeword length N, while satisfying the performance criterion. The optimization problem can then be formulated as follows:

$$\min_{\epsilon,q,N} \mathcal{E}(\xi, q, \epsilon, N) \quad \text{s.t.} \quad p_{e,\text{opt}}(\xi, \epsilon, q, N) < p_{e,max} \tag{5.7}$$

where

$$p_{e,\text{opt}}(\xi, \epsilon, q, N) = \min_{\alpha,\lambda} p_{e_N}(\xi, \epsilon, q, N).$$

In the above optimization problem, $\mathcal{E}(\xi, q, \epsilon, N)$ is given by (7.17) and $p_{e_N}(\xi)$ is defined in (7.4). In addition, $p_{e,\text{opt}}(\xi, \epsilon, q)$ gives the minimum error probability that can be reached by optimizing the scaling parameter α and the Min-Sum offset λ.

5.5 Optimization Methods

In the following , we propose two optimization methods, in order to find the optimal parameters that minimise the energy consumption with respect to the performance criterion.

5.5.1 Optimization by exhaustive search

To solve the optimization problem (7.18), we first use exhaustive search to find the best couple (α, λ) that provides the minimum error probability $p_{e,\mathrm{N}}(\xi^*, \epsilon, q, N)$ at the SNR ξ^*, for every combination of (ϵ, q, N), where $(\lambda \in \mathcal{N}^+)$ and $(\alpha \in \mathcal{R}^+)$. Then we perform exhaustive search to find the best combination (ϵ, q, N), that minimizes the energy consumption while respecting the performance criterion. This optimization method is time consuming because it evaluate all the possible combinations of (ϵ, q, N), but at least it is guarantied to find the optimal solution.

5.5.2 Alternate optimization method

In this Section we propose another optimization method, less complex than the exhaustive method. The optimization problem defined in (7.18) is difficult to solve because it is non-differentiable and because it involves discrete parameters q and N . In addition, it is rather complex to evaluate the number of iterations $L_N(\xi)$ and the error probability $p_{e,N}(\xi)$ for given parameters q, ϵ, N, because this requires to numerically evaluate an integral. Therefore, we want to reduce the evaluation of these terms as much as possible.

In order to solve the optimization problem, we first define research intervals for the parameters q, N, ϵ involved in the optimization. We assume that $q \in [\![q_{\min}, q_{\max}]\!]$, $N \in [\![N_{\min}, N_{\max}]\!]$, and $\epsilon \in [\epsilon_{\min}, \epsilon_{\max}]$. The research interval for ϵ is continuous and corresponds to the whole range of potential values for this parameter, see Section 7.8.2. The range of values for q and N must be selected so as to satisfy the performance criterion at least for the largest values $N_{\max}$ and $q_{\max}$. For instance, in our simulations, we set $q_{\max} = 8$, since for this case, the performance of the quantized Min-Sum decoder is almost the same as the performance of the non-quantized decoder.

Once the research intervals are set, we then perform coordinate descent, which consists of optimizing alternatively each of the three parameters ϵ, q, and N. Since we consider a constrained optimization problem, we should verify at each iteration that the selected

parameters meet the performance criterion of the optimization problem. For this reason, we first initialize our algorithm with three parameters $\epsilon^{(0)} = \epsilon_{\max}$, $q^{(0)} = q_{\max}$, and $N^{(0)} = N_{\max}$. Then, at iteration $i \in [\![1, I]\!]$, we perform the successive three optimizations:

We first optimize q while parameters ϵ and N are fixed from previous iteration:

$$q^{(i)} = \arg\min_{q} \mathcal{E}(\xi, q, \epsilon^{(i-1)}, N^{(i-1)}) \text{ s.t.} p_{e,\text{opt}}(\xi, q, \epsilon^{(i-1)}, N^{(i-1)}) < p_{e,max} \tag{5.8}$$

We then optimize N using the optimal value of q from the last optimization and ϵ is fixed from previous iteration:

Finally We optimize ϵ using the optimal values of q and N from the last optimization:

$$\epsilon^{(i)} = \arg\min_{\epsilon} \mathcal{E}(\xi, q^{(i)}, \epsilon, N^{(i)}) \text{ s.t. } p_{e,\text{opt}}(\xi, q^{(i)}, \epsilon, N^{(i)}) < p_{e,max} \tag{5.9}$$

The following three optimizations are performed until the maximum number of iterations I is reached. In the above expressions, the parameter q is optimized by exhaustive search, since its research interval is small. Then, for the optimization of N and ϵ, we retain the parameters that satisfy the performance criterion $p_{e,\text{opt}}$ and minimize the energy $\mathcal{E}$ among a certain number of values between $N_{\min}$ and $N_{\max}$ and between $\epsilon_{\min}$ and $\epsilon_{\max}$. To further reduce the computation time, we first evaluate the performance criterion $p_{e,\text{opt}}$, and then evaluate the corresponding energy $\mathcal{E}$ only if the performance criterion is satisfied.

Finally, the interest of the coordinate descent approach is that it guarantees that the energy criterion is reduced at each iteration, in the sense that

$$\mathcal{E}(\xi, q^{(i)}, \epsilon^{(i)}, N^{(i)}) \leq \mathcal{E}(\xi, q^{(i-1)}, \epsilon^{(i-1)}, N^{(i-1)}) \tag{5.10}$$

It also ensures that the final solution verifies the performance criterion. In the next section, we evaluate through numerical simulations the proposed optimization method.

5.6 Numerical results

In this section, we consider five different protographs, all with parameters $m = 2$, $n = 4$, and code rate $R = 0.5$. The protographs S_{17}, S_{36} and S_{55} were constructed using the protograph optimization method for performance only 4.3.1. When applying the protograph optimization method, the protographs were optimized by considering a

large quantization level $q = 8$ in order to have a performance very close to the non-quantized decoder. We also consider the protographs S_m and S_c that were obtained in [87] by optimizing the decoder energy consumption.

For the five protographs, we set $p_{e,max} = 10^{-3}$ as the target error probability to be achieved at the SNR $\xi^{=}$1.45 dB. We then find the optimum parameters q_{op}, ϵ_{op} and N_{op} from the optimization method proposed in Section 5.5.1, and the optimization method proposed in Section 5.5.1.

5.6.1 Exhaustive method results

In order to find the best combination (ϵ, q, N), we test all the possible cases while using value of $e_g \in [0.1, 0.2, \ldots, 1]$, $q \in [3, 2, \ldots, 8]$ and $N \in [10^3, 2 \times 10^3, \ldots 10^4]$. We then use (7.17) to evaluate the energy consumption $\mathcal{E}(\xi, q, \epsilon, N)$ and (7.8) to evaluate the performance $p_{e,\text{N}}(\xi, \epsilon, q, N)$ at $\xi^{=}$1.45 dB, as to find the optimal combination $(\epsilon_{op}, q_{op}, N_{op})$ that minimises the energy consumption and respects the target error probability $p_e^{\star} = 10^{-3}$.

Figure 5.1 and Figure 5.2 provide the energy consumption and the BER of the protographs S_{55} and S_{33} respectively. In Figure 5.1 both protographs achieve a minimum energy consumption at $N = 4 \times 10^3$, however, in Figure 5.2 the protograph S_{36} can not achieve the target error probability until $N = 7 \times 10^3$, where S_{55} can easily achieve the target error probability with $N = 4 \times 10^3$.

In order to see how the energy varies with the quantization step, Figure 5.3 gives the energy consumption of the protograph S_{55} at $N = 4\times$ for different quantization step q. At $0.1 < e_g < 0.7$ the minimum energy consumption is given by $q = 3$, however in figure 5.4 we can see that only $q = 6$ and $q = 7$ can archive the target error probability where $q = 3$ and $q = 4$ can not even at $e_g = 1$. we also observe that the minimum energy consumption is given by $q = 5$ at $e_g = 0.9$.

Table 5.1 represents the optimum parameters $(e_{gop}, q_{op}, N_{op})$ the provides the minimum energy consumption per information bit $\mathcal{E}_{min}$ and the nominal energy consumption $\mathcal{E}_{nominal}$ evaluated using the energy model (7.17) for the five considered protographs. The nominal energy consumption $\mathcal{E}_{nominal}$ was estimated considering a quantization level $q = 8$, a code Lent $N = 10000$ and 100% of the nominal energy E_0 ($e_g = 1$). We observe that, for every protograph, the minimum energy score is achieved for a quantization level ($q_{op} = 5$) and for failure probability $\epsilon = 1.02 \times 10^{-5}$, which corresponds to the case where the device uses 90% of the nominal energy E_0 ($e_{gop} = 0.9$). On the other hand, the optimal code

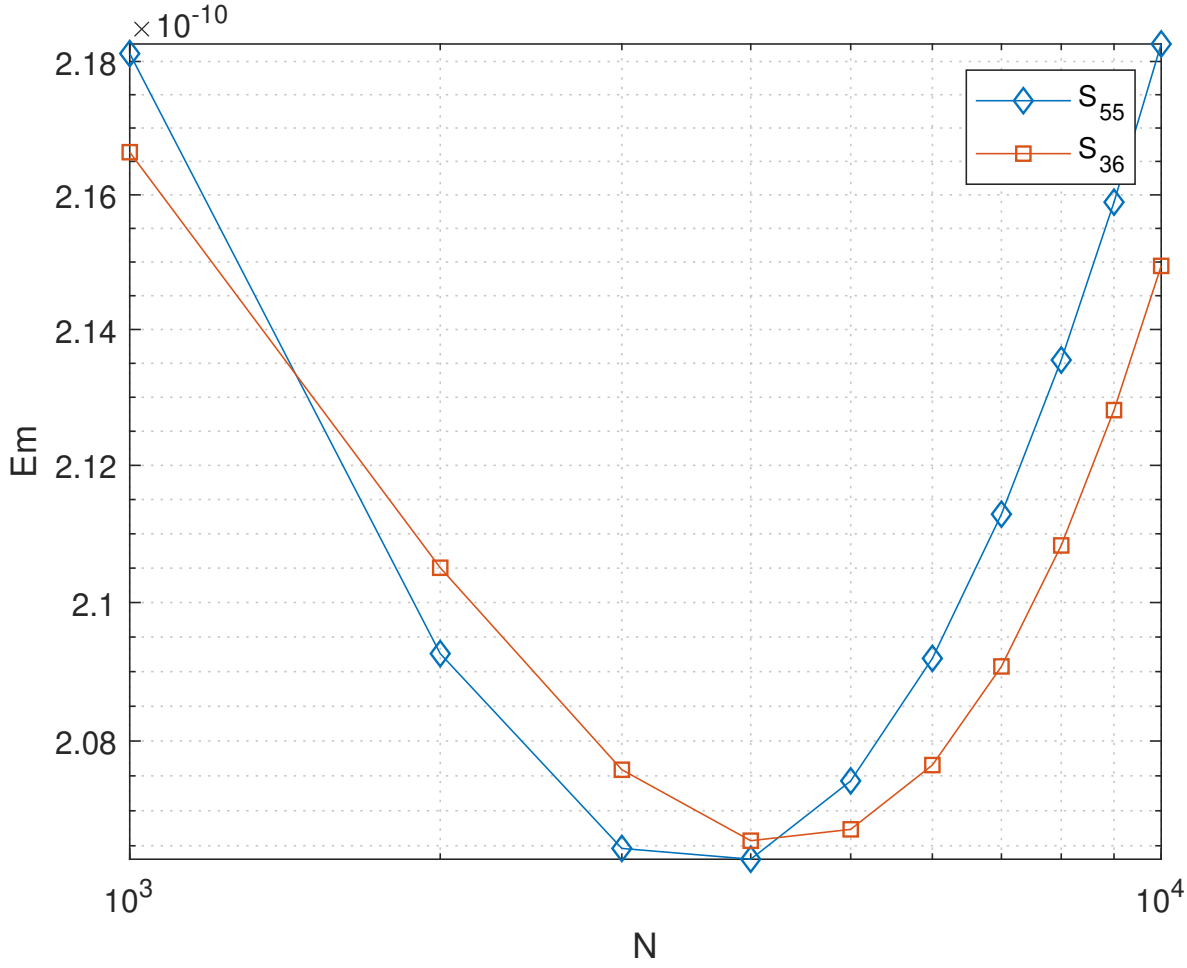

Figure 5.1 – Energy consumption for the protographs S_{55} and S_{36} at $\xi = 1.45$ dB for different code lengths

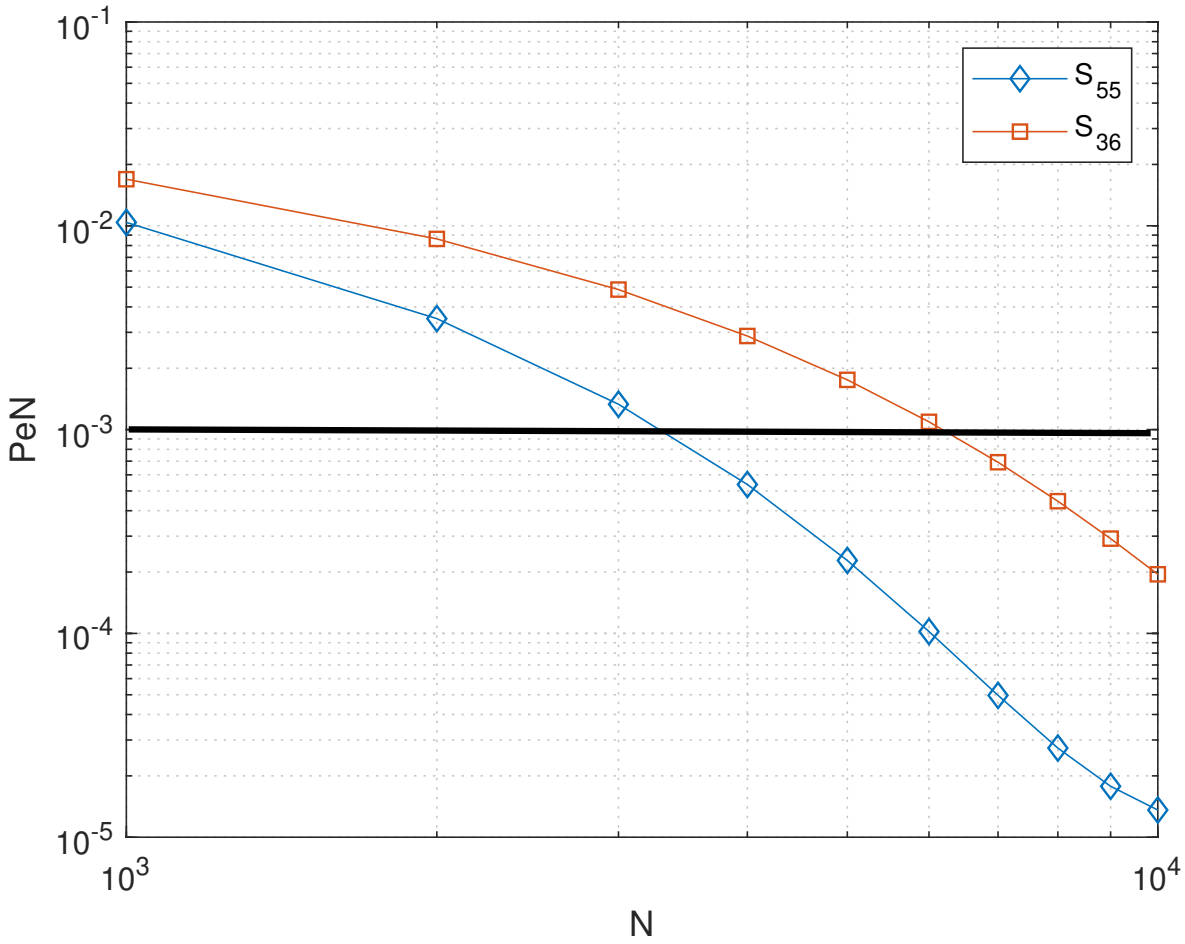

Figure 5.2 – Bit Error Rate of the protographs S_{55} and S_{36} at $\xi = 1.45$ dB for different code lengths. S_{17} and S_m

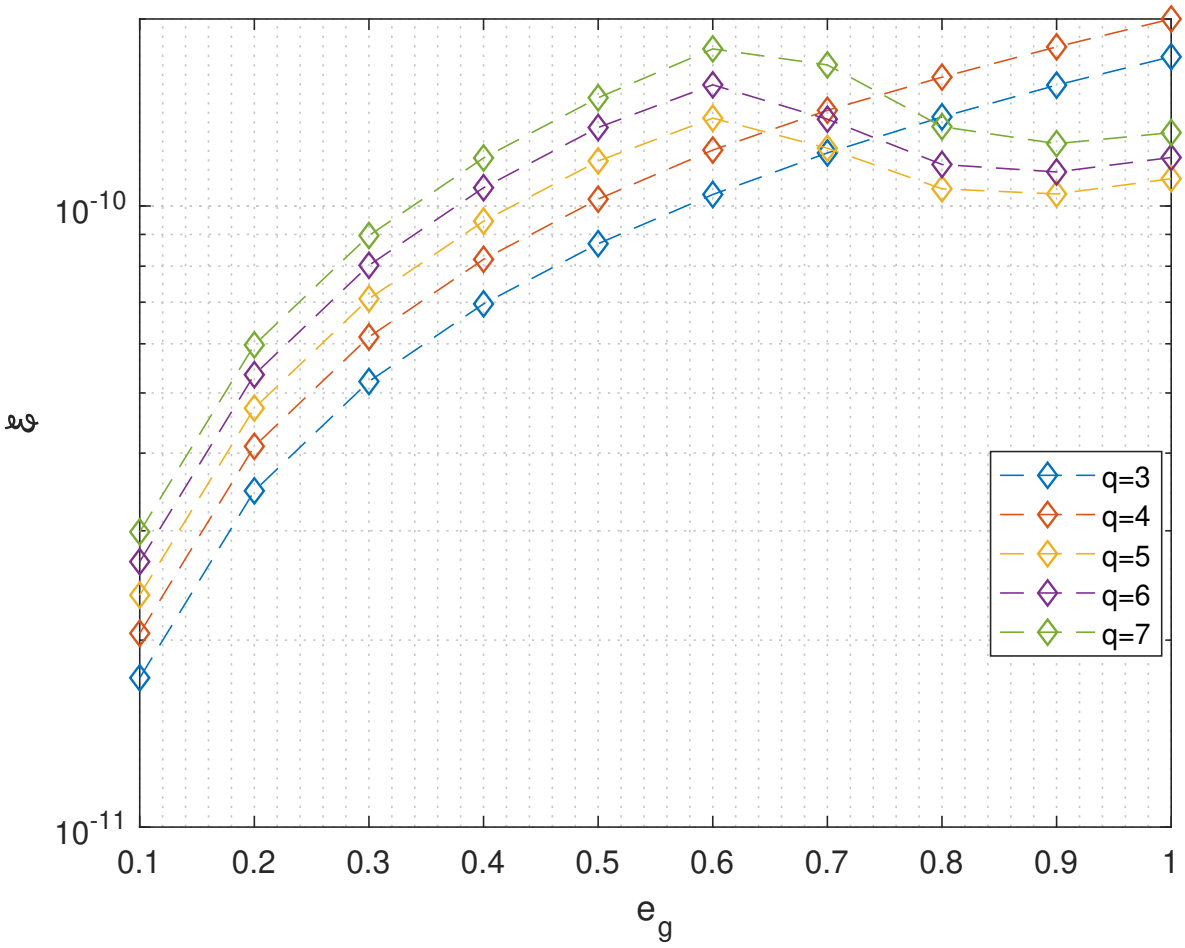

Figure 5.3 – Energy consumption of the protograph S_{55} for different value of quantization level q and different values of e_g at $N = 4 \times 10^3$

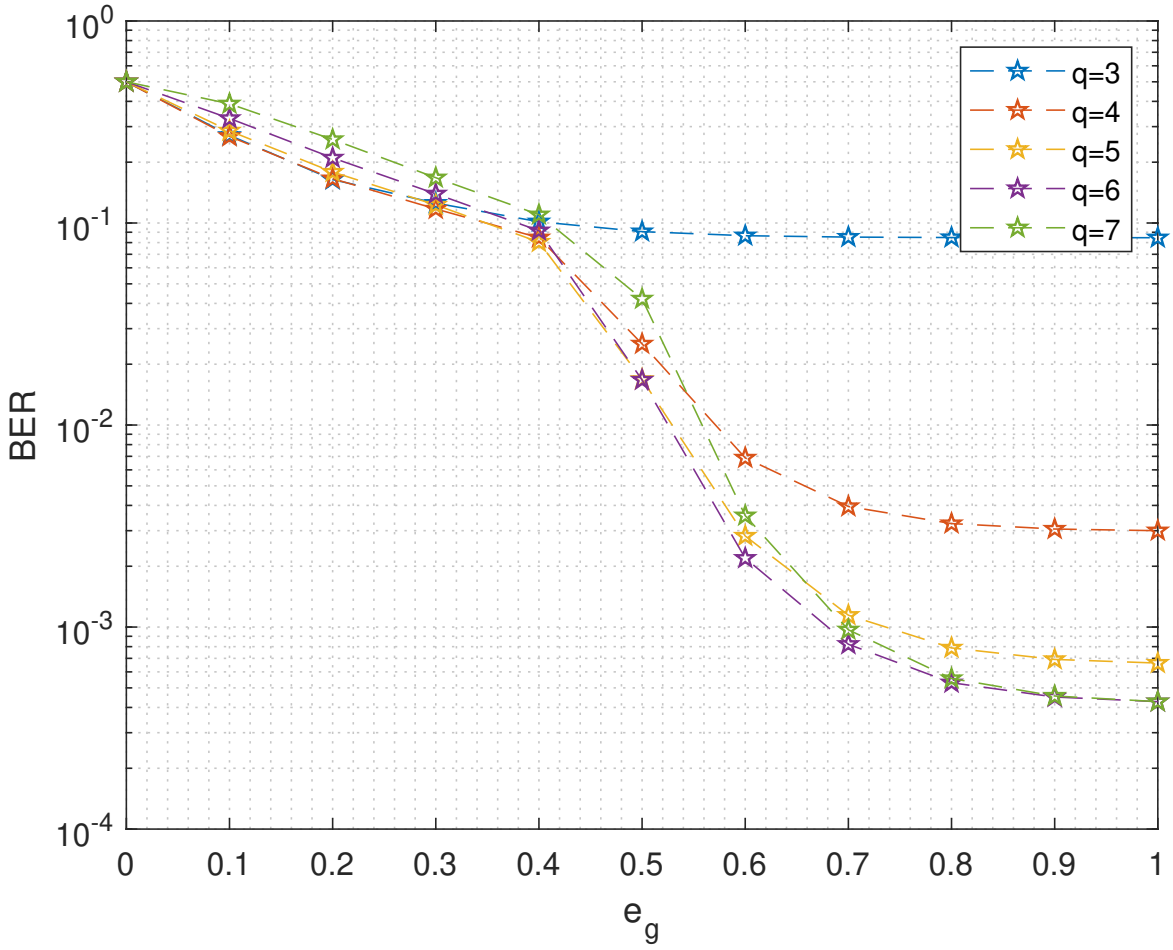

Figure 5.4 – Bit Error Rate of the protograph S_{55} for different value of quantization level q and different values of e_g at $N = 4 \times 10^3$

Table 5.1 – Minimum energy value $\mathcal{E}_{\min}$ and optimal parameters for the considered protographs compared to the nominal energy consumption $\mathcal{E}_{nominal}$

Protograph	$\mathcal{E}_{min}$	$e_{g_{\text{op}}}$	q_{op}	N_{op}	$\mathcal{E}_{nominal}$
$S_{17} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 0 & 1 & 4 & 1 \end{bmatrix}$	1.985×10^{-10} J	0.9	5	4000	2.96×10^{-10} J
$S_{36} = \begin{bmatrix} 2 & 1 & 2 & 3 \\ 1 & 4 & 0 & 1 \end{bmatrix}$	2.127×10^{-10} J	0.9	5	7000	3.43×10^{-10} J
$S_m = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 4 \end{bmatrix}$	2.006×10^{-10} J	0.9	5	8000	2.96×10^{-10} J
$S_{55} = \begin{bmatrix} 1 & 0 & 5 & 1 \\ 3 & 2 & 1 & 1 \end{bmatrix}$	2.007×10^{-10} J	0.9	5	4000	3.22×10^{-10} J
$S_c = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{bmatrix}$	2.229×10^{-10} J	0.9	5	5000	3.412×10^{-10} J

length N_{op} depends on the considered protographs.

We now evaluate the energy consumption and the decoding performance of the considered protographs. Figure 5.5 shows both the minimum energy per information bit and the Bit Error Rate (BER) for the considered protographs. As we can see, and for $e_g < 0.8$, the decoder cannot achieve the target error probability, so the minimum energy consumption will be estimated using the lowest value of the quantization step ($q = 3$ in our simulation). In the other hand, and for $0.8 \leq e_g \leq 1$ the decoder satisfies the performance criteria and provides a minimum energy consumption at $e_g = 0.9$ for all the considered protographs.

Figure 5.6 provides the BER of the considered protographs, evaluated using the finite length density evolution methode presented in (7.8), considering the optimal parameters $(e_{gop}, q_{op}, N_{op})$ for every protograph and in the nominal case. In the nominal case, the decoder provides a better performance, but all the protographs reach the target error probability $p_{e,max} = 10^{-3}$ at the SNR $\xi = 1.45$ dB in the optimized case.

The global optimization method can give a solution to the optimization problem but it come with a cost in simulation time where we need to evaluate all the possible combinations of (e_g, q, N).

5.6.2 Alternate optimization method results

In this optimization method , we set $N_{min} = 10^3$, $N_{max} = 10^4$, $q_{min} = 3$ and $q_{max} = 8$ with $I = 3$ iterations. In Table 7.2, we observe that, for every protograph, the minimum energy is achieved for the same quantization level $q_{\text{op}} = 5$ and that the optimal failure

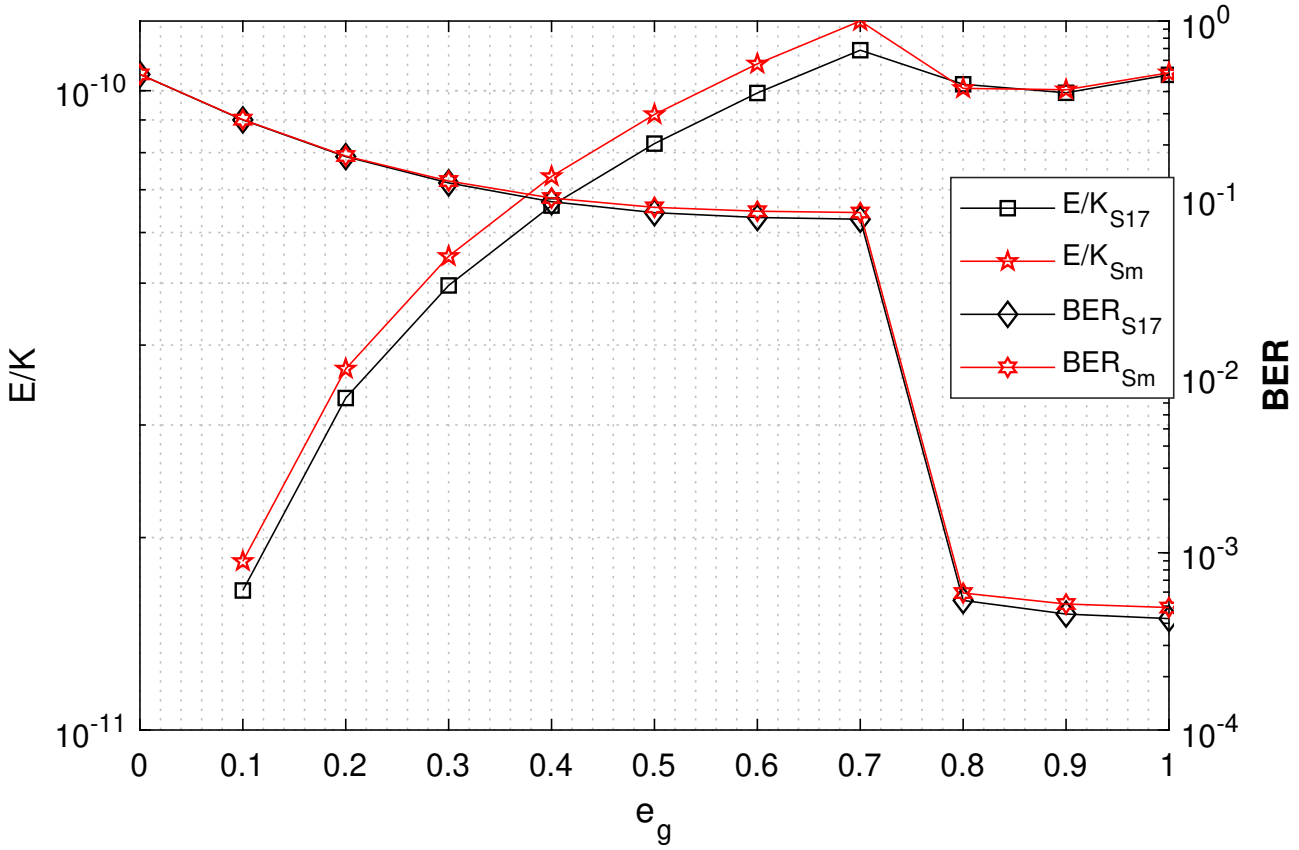

Figure 5.5 – Minimum energy per information bit and the Bit error rate for every value of the normalized energy e_g for the protographs. S_{17} and S_m

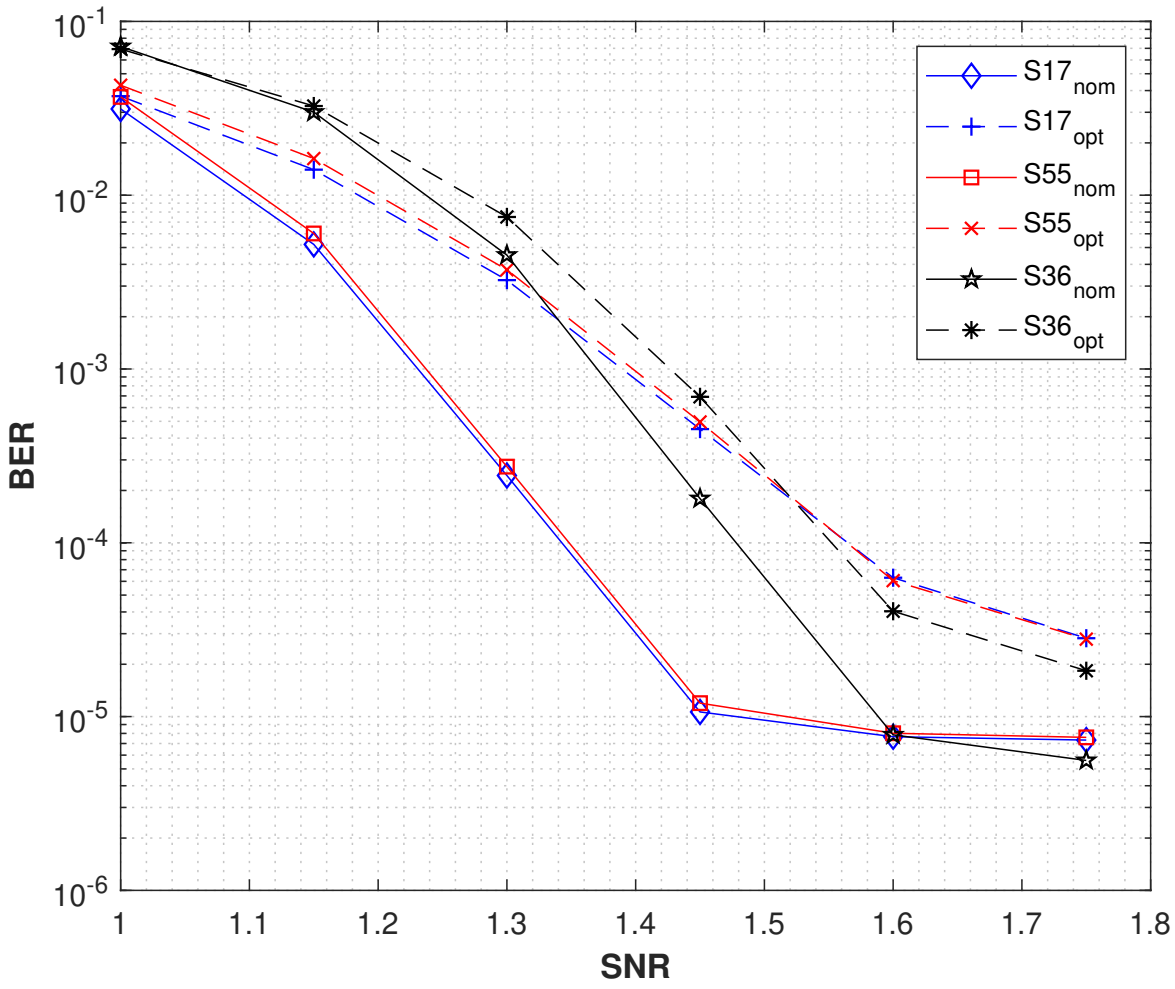

Figure 5.6 – BER of the protographs S_{17}, S_{55} and S_{36} evaluated using the optimal parameter compared to using the nominal parameters

Table 5.2 – Minimum energy value $\mathcal{E}_{\min}$ and optimal parameters for the four considered protographs. The left part of the table represents the case where the three parameters (q, N, ϵ) are optimized. The right part gives energy values $\mathcal{E}_{q_{\text{op}}}$ and $\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$, when only q is optimized, and when only q and N are optimized, respectively.

Protograph	$\mathcal{E}_{min}$	$e_{g_{\text{op}}}$	q_{op}	N_{op}	$\mathcal{E}_{q_{\text{op}}}$	$\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$	N_{op}
$S_{17} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 0 & 1 & 4 & 1 \end{bmatrix}$	1.86×10^{-10} J	0.86	5	3120	2.08×10^{-10} J	2.00×10^{-10} J	3056
$S_{36} = \begin{bmatrix} 2 & 1 & 2 & 3 \\ 1 & 4 & 0 & 1 \end{bmatrix}$	2.06×10^{-10} J	0.85	5	6264	2.21×10^{-10} J	2.22×10^{-10} J	6168
$S_m = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 4 \end{bmatrix}$	1.93×10^{-10} J	0.84	5	8000	2.09×10^{-10} J	2.09×10^{-10} J	10000
$S_{55} = \begin{bmatrix} 1 & 0 & 5 & 1 \\ 3 & 2 & 1 & 1 \end{bmatrix}$	1.93×10^{-10} J	0.88	5	3240	2.18×10^{-10} J	2.062×10^{-10} J	3192
$S_c = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{bmatrix}$	2.11×10^{-10} J	0.88	5	4280	2.35×10^{-10} J	2.22×10^{-10} J	4056

probabilities are close to each other, *i.e.*, $1.02 \times 10^{-5} < \epsilon_{\text{op}} < 3.38 \times 10^{-5}$, which roughly corresponds to using between 80% and 90% of the nominal energy E_0. On the other hand, the optimal code length N_{op} strongly depends on the considered protograph. We also compared the obtained minimum energy values with respect to two setups in which we optimize only a part of the parameters. In Table 7.2, $\mathcal{E}_{q_{\text{op}}}$ gives the minimum energy value when only q is optimized, and when $N = 10000$ and $e_g = 1$. And $\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$ gives the minimum energy value when both q and N are optimized, and when $e_g = 1$. We observe an energy gain of 10 to 15% in the case where all the parameters are optimized.

We now focus on the two protographs S_{17} and S_{36}. For these two protographs, Figure 5.7 gives the values of $\mathcal{E}_{\min}$, $\mathcal{E}_{q_{\text{op}}}$ and $\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$ with respect to SNR. In every considered case, the parameters q, N, and ϵ are optimized for the target SNR $\xi^\star$, and then used at every SNR. We see that although the optimization is performed at one single SNR value, the performance order is preserved at any SNR. FOr instance, the energy value $\mathcal{E}_{\min}$ for protograph S_{17} is always lower than $\mathcal{E}_{\min}$ for protograph S_{36}. The figure also confirms the energy gain at optimizing the three parameters, instead of just q or just q and N.

At the end, for protograph S_{17} Figure 5.8 shows the Bit Error Rate (BER) with respect to SNR, evaluated both from the finite-length method of Section 2.2.2 and from Monte-Carlo simulations. Again, the three sets of parameters leading to $\mathcal{E}_{\min}$, $\mathcal{E}_{q_{\text{op}}}$, and $\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$, are considered. We first observe that the finite-length method of Section 2.2.2 accurately predicts the decoder error probabilities. In addition, as expected, we see that the case

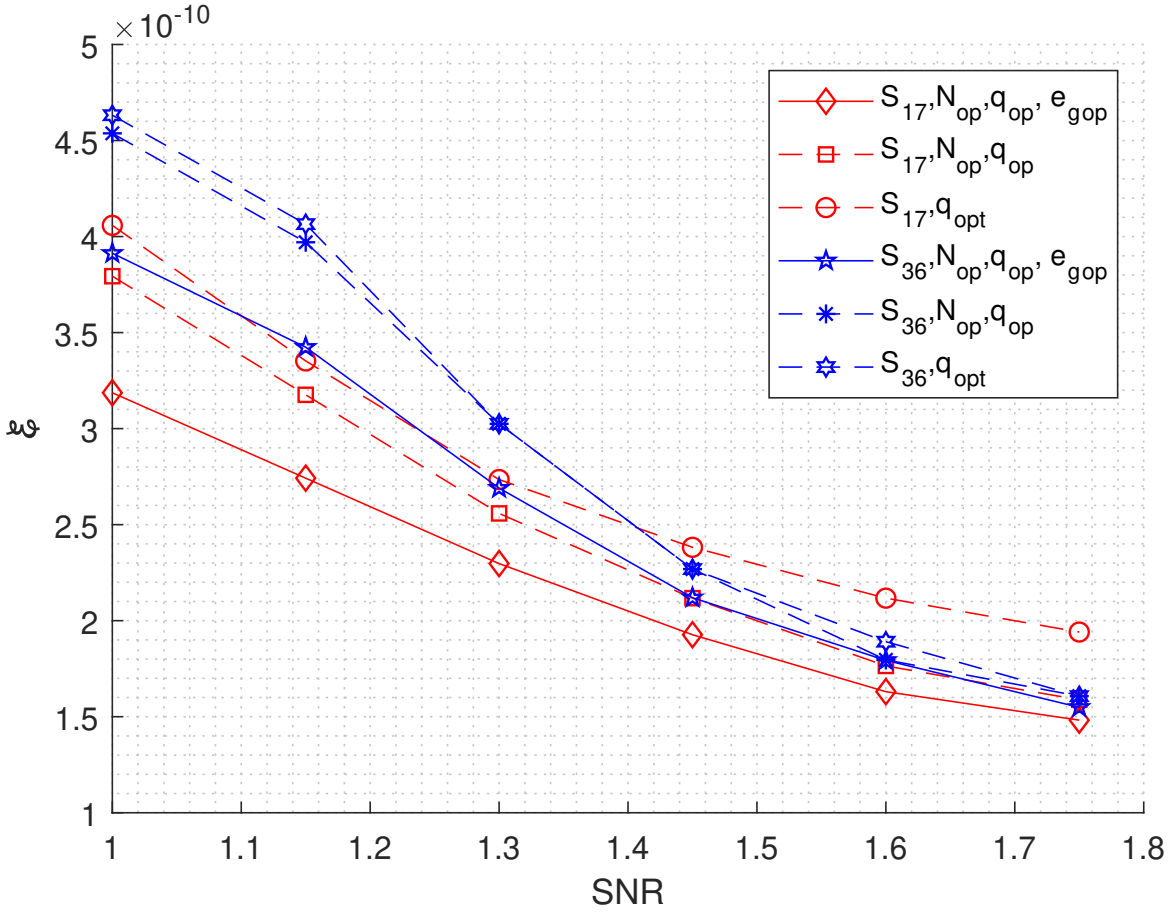

Figure 5.7 – Energy values $\mathcal{E}_{\min}$, $\mathcal{E}_{q_{\text{op}}}$, and $\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$, with respect to SNR, for the protographs S_{17} and S_{36}

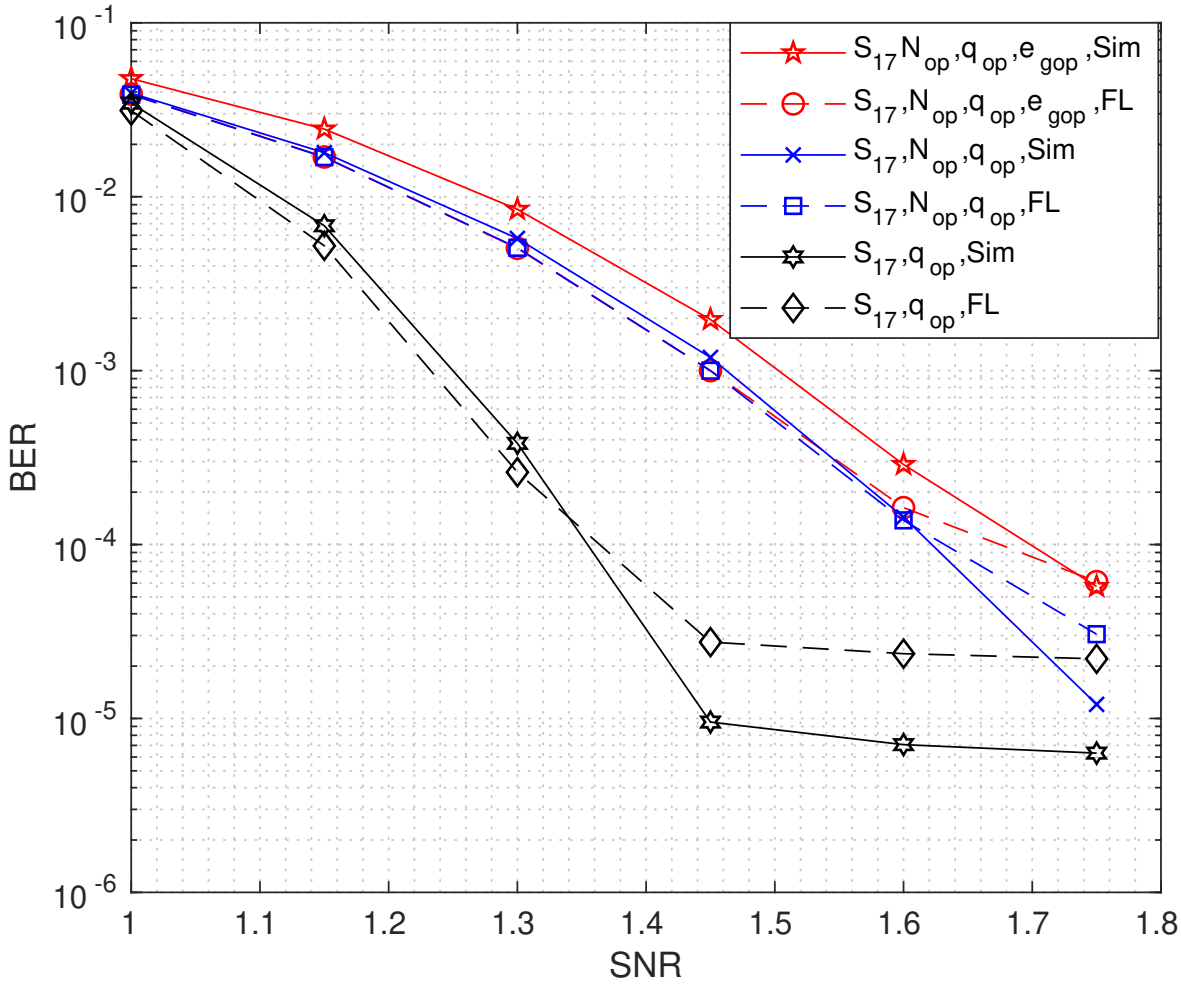

Figure 5.8 – BER with respect to SNR of the protograph S_{17}, evaluated from the finite-length density evolution method and from Monte Carlo simulations.

where the three parameters are optimized shows a degraded performance. Finally, we conclude that optimizing the decoder parameters allows to reduce the decoder energy consumption, although this has a cost in performance.

5.7 Bit energy optimization

In the previous sections of this chapter, we considered that all quantization bit have the same energy e_g. However, the bits do not have the same influence on the performance, therefore, in this section we consider that every quantization bit k has its own energy e_{gk}, and that the noise level depends also on the position of the bit, that is $\mathbb{P}(\mathcal{B}_k = 1) = \epsilon_k$, Where $k \in \{1, \ldots, q\}$. We use the same fault model presented in (7.15). We now want to optimize the energy levels per bit in order to minimise the energy consumption while providing certain decoding performance. In this section, we suppose that the code length N and the quantization level q are fixed, and we will follow the same process as before, we start by updating the faulty memory model, the we define the optimization problem, finally we present the alternate optimization method.

5.7.1 Energy Model

In order to estimate the energy consumption we rely on the energy model provided by (7.17), and we consider energy levels e_{gk} for each bit k, according to its position in the quantized messages. As a result, the following energy model will be considered in the optimization:

$$\mathcal{E} = \frac{L_N(\xi)}{Rn} E_0 \left(\sum_{i=1}^{n} d_{v_i} \left(\sum_{k=1}^{q} e_{gk} + q_s e_{g2} \right) + (1 - R) \left(\sum_{j=1}^{m} \left(2 \sum_{k=2}^{q} e_{gk} + d_{c_j} e_{g1} \right) \right) \right). \quad (5.11)$$

In this model we consider that the bit sign, which is the Most Significant Bit (MSB) is located at $k = 1$, and that bits from $k = 2$ to the Least Significant Bit (LSB) $k = q$ represent the absolute value of the quantized message. We assume that the q_s additional bits used in the variable nodes computation have the same energy e_{g2}.

5.7.2 Optimization problem

In this part, in order to simplify the optimization problem, we assume that the protograph S, the code rate R, the quantization level q and the code length N are fixed. As in

Section 4.3.2 we define the performance criterion as a target error probability $p_{e,max}$ to be achieved at an SNR value ξ. First, we mainly want to study the effect of a non-uniform energy distribution, and that is why we assume that the other parameters are fixed. Once these parameters are set the optimization problem is given by:

$$\min_{\epsilon_1,\ldots,\epsilon_q} \mathcal{E}(\xi, \epsilon_1, \ldots, \epsilon_q) \;\text{ s.t. }\; p_{e,\text{opt}}(\xi, \epsilon_1, \ldots, \epsilon_q)) < p_{e,max} \tag{5.12}$$

In this optimization problem, the energy term $\mathcal{E}$ to be minimized is given by (7.19).

5.7.3 Optimization method

In the above optimization problem, there are q parameters to optimize in order to achieve the minimum energy consumption. In this case, an exhaustive search may be time consuming. Therefore, we rely on the alternating optimization method presented in Section 5.5.2, where we optimize one parameter ϵ_k at a time, where $k \in 1, \ldots, q$.

First, we start by defining the search intervals such that $\epsilon_k \in [\epsilon_{max}, \epsilon_{min}]$, where $k \in \{1, \ldots, q\}$. The research interval for ϵ is continuous and corresponds to the whole range of potential values for this parameter, see Section 7.5. In order to satisfy the decoding performance, we set $\epsilon_k^0 = \epsilon_{max}$ for $k \in \{1, \ldots, q\}$. Then, at iteration $i \in [\![1, I]\!]$, we perform the successive q optimizations:

$$\epsilon_k^{(i)} = \arg\min_{\epsilon_k} \mathcal{E}(\xi, \epsilon_1^{(i)}, ..., \epsilon_k, \epsilon_{k+1}^{(i-1)}, ..., \epsilon_q^{(i-1)}) \text{ s.t. } p_{e,\text{opt}}(\xi, \epsilon_1^{(i)}, ..., \epsilon_k, \epsilon_{k+1}^{(i-1)}, ..., \epsilon_q^{(i-1)}) < p_{e,max} \tag{5.13}$$

where $k \in \{1, \ldots, q\}$. We perform these optimizations for every bit position in the quantized message representation, for a given number of iterations I. At the end of every optimization, we keep the parameter ϵ_k that minimizes the energy consumption and satisfies the performance criterion. In the next section, we present some numerical results of the proposed method.

5.7.4 Numerical results

We now evaluate the bit energy optimization method. Here, we consider the protographs S_{17}, S_{55}, S_{36}, S_m and S_c presented in Section 5.6. We also use the optimal parameter N_{op}, q_{op} that were obtained for each protograph so as to compare the energy values obtained in Section 5.6 to the energy values obtained after non-uniform energy allocation over bits. For the performance criterion, we set the error probability $p_{e,max} = 10^{-3}$

Table 5.3 – The final energy value $\mathcal{E}_f$, optimal energy per bit and the minimum energy value $\mathcal{E}_{\min}$ calculated with $e_{gk} = e_{gop}(S), \forall k \in \{1, \ldots, q\}$ for the considered considered protographs at SNR value $\xi = 1.45$dB

Protograph	$\mathcal{E}_{\min}$	e_{g1}	e_{g2}	e_{g3}	e_{g4}	e_{g5}	$\mathcal{E}_f$
$S_{17} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 0 & 1 & 4 & 1 \end{bmatrix}$	1.86×10^{-10} J	1	0.7	0.65	0.65	0.6	1.47×10^{-10} J
$S_{36} = \begin{bmatrix} 2 & 1 & 2 & 3 \\ 1 & 4 & 0 & 1 \end{bmatrix}$	2.06×10^{-10} J	1	0.7	0.8	0.7	0.94	1.77×10^{-10}
$S_m = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 4 \end{bmatrix}$	1.93×10^{-10} J	0.96	0.55	0.55	0.5	0.5	1.36×10^{-10}
$S_{55} = \begin{bmatrix} 1 & 0 & 5 & 1 \\ 3 & 2 & 1 & 1 \end{bmatrix}$	1.93×10^{-10} J	1	0.7	0.98	0.9	0.84	1.70×10^{-10}
$S_c = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{bmatrix}$	2.11×10^{-10} J	1	0.75	0.7	0.75	0.68	1.74×10^{-10}

to be achieved at an SNR valuer $\xi = 1.45$ dB. In the optimization method we use $I = 3$ iterations to find the optimal parameters.

Table 7.3 represents the final energy scores $\mathcal{E}_f$ calculated using the optimal energy per bit $e_{gk}, k \in 1, \ldots, q$, compared to the minimum energy scores $\mathcal{E}_{\min}$ presented in Table 7.2, and calculated using $e_{gk} = e_{gop} \forall k \in 1, \ldots, q$ for the considered protographs. As expected, the MSB is the bit that requires the highest level of energy, close to the nominal energy (90 % to 100 % of E_0) in order to satisfy the performance criterion. Indeed, the sign of the quantized messages is represented by the MSB and has a strong influence in the decoding process. On the opposite, the others bits can tolerate a higher level of noise, which allows to reduce the energy per level of those bits. As a result, the minimum energy score can be reduced by up to 30 %.

In case of protograph S_{55} or S_{36}, not only MSB but also LSB require a high energy level. this is due to the fact that using the optimal parameters N_{op} and q_{op} gives a performance level very close to the target error probability, which makes it hard to find lower energy per bit values awhile providing the required performance.

Now we evaluate the decoding performance of the considered protographs using the optimized energy per bit values. Figure 5.10 gives the BER of the five protographs evaluated using the finite length Density Evolution method presented in (7.8). As we can see, at SNR value $\xi = 1.45$dB, all the protographs satisfy the performance criterion. then Figure 5.7 provides the energy consumption of the considered protographs with respect to the SNR, evaluated with the energy model (7.19) , with the optimized energy values.

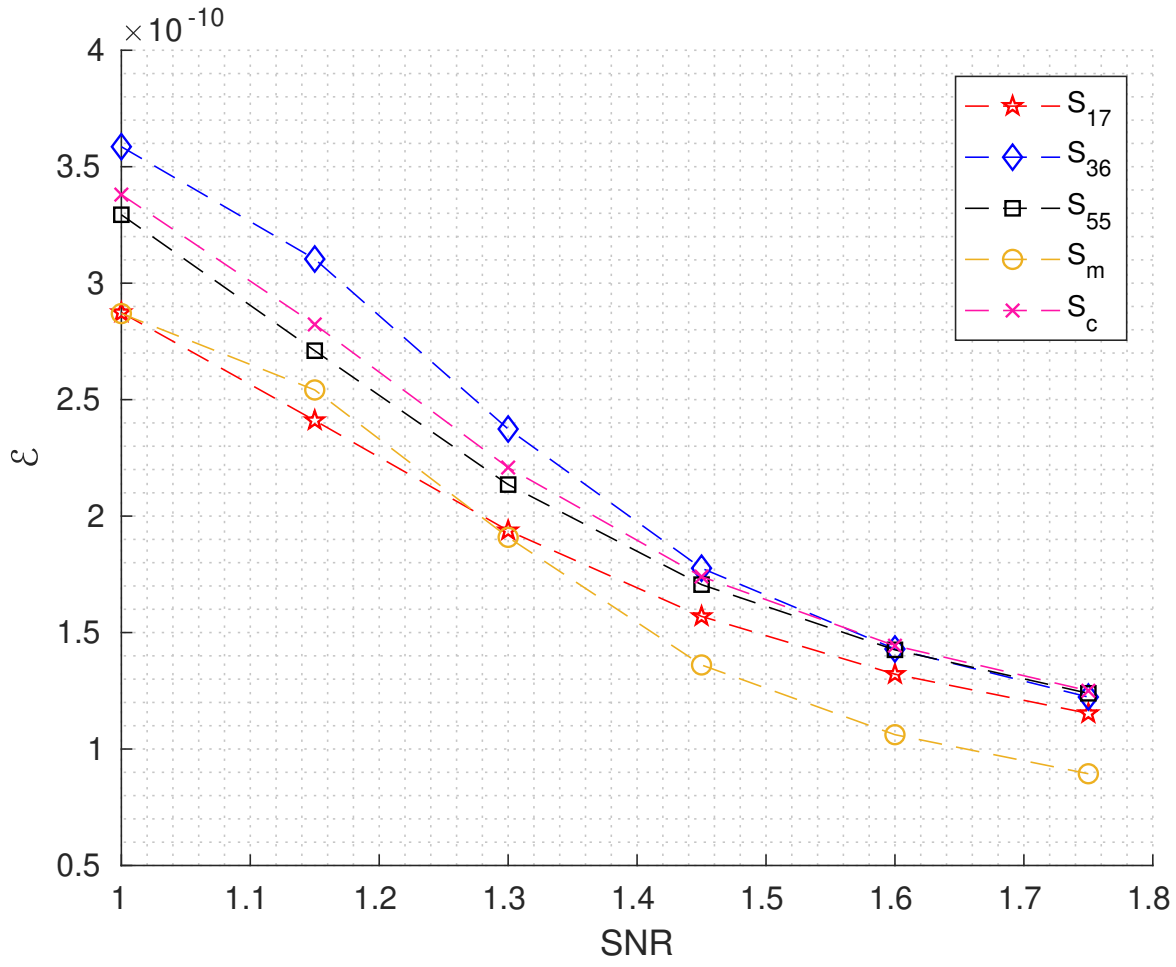

Figure 5.9 – Energy values $\mathcal{E}_f$, with respect to SNR, for the five protographs

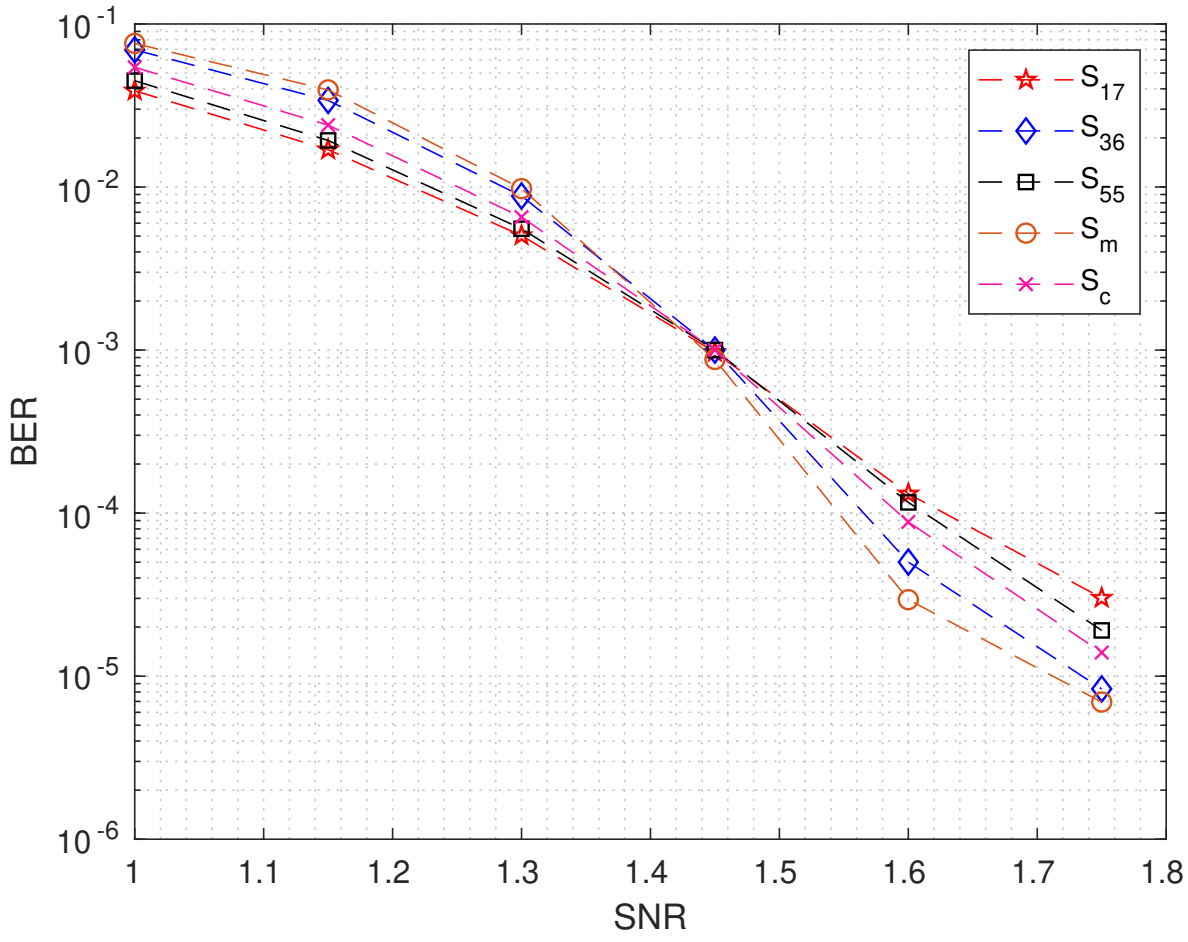

Figure 5.10 – BER with respect to SNR h of the considered protographs, evaluated from the finite-length density evolution method using the quantized energy per bit values.

As expected from Table 7.3, the The protograph S_m provides the minimum energy score at SNR $\xi = 1.3$ dB and higher.

5.8 Conclusion

In this chapter, we introduced an energy model for faulty quantized Min-Sum decoder. We then proposed a method to optimize the number of quantization bits, the code length, and the nominal energy, in order to minimize the energy consumption while satisfying a given decoding performance criterion. Simulation results showed that using the optimal parameters allows to reduce the energy consumption while satisfying the performance criterion. Finally, we optimized the energy per bit for the considered protograph in order to minimize the decoding energy consumption, which allowed to reduce the minimum energy scores by up to 30%.

Chapter 6

Protograph optimization for Faulty Min-Sum decoder

In the previous chapter, we considered a faulty Min-Sum decoder, and we assumed a fixed protograph. We then proposed a method to minimize the decoder energy consumption, by optimizing the parameters q, N, and the bit energy levels in the memories. In this chapter, we seek to minimize the decoder energy consumption by optimizing the protograph and the energy per bit. We also consider a fixed quantization level q and a fixed code length N, which is often the case in practice.

6.1 Optimization problem

The objective of this chapter is to find the best protograph and the energy per quantization bit for a given code length for a faulty Min-Sum decoder that minimises the energy consumption while respecting a given performance criterion. We now formulate the optimization problem.

In order to simplify the protograph optimization, we consider a fixed code rate R, and fixed protograph size given by parameters n and m. As in Chapter 4, we set the decoding criterion as a target error probability $p_{e,max}$ to be achieved at SNR ξ. The optimization problem is then given by:

$$\min_{S,\epsilon_1,\ldots,\epsilon_q} \mathcal{E}(\xi, S, \epsilon_1, \ldots, \epsilon_q) \quad \text{s.t.} \quad p_{e,\text{opt}}(\xi, S, \epsilon_1, \ldots, \epsilon_q)) < p_{e,max} \tag{6.1}$$

The above optimization problem has $q+1$ parameter to be taken into consideration. In order to simplify the optimization, we break down the above problem into two optimization problems, starting with:

$$\min_{S} \mathcal{E}(\xi^*, S, \epsilon) \quad \text{s.t.} \quad p_{e,\text{opt}}(\xi^*, S, \epsilon)) < p_e^\star \tag{6.2}$$

Table 6.1 – The optimized protographs for the considered code lengths, the initial energy consumption $\mathcal{E}_i$, the optimized energy per quantization bit vales with the final energy consumption $\mathcal{E}_f$ Evaluated at SNR value $\xi = 1.45$

Protograph	$\mathcal{E}_i$	e_{g1}	e_{g2}	e_{g3}	e_{g4}	e_{g5}	$\mathcal{E}_f$	N
$S_4 = \begin{bmatrix} 0 & 1 & 1 & 4 \\ 2 & 1 & 3 & 1 \end{bmatrix}$	1.93×10^{-10} J	0.96	0.5	0.5	0.4	0.3	1.29×10^{-10} J	4000
$S_5 = \begin{bmatrix} 1 & 2 & 1 & 3 \\ 1 & 1 & 1 & 2 \end{bmatrix}$	2.11×10^{-10} J	0.96	0.5	0.5	0.484	0.4	1.40×10^{-10} J	5000
$S_6 = \begin{bmatrix} 2 & 3 & 1 & 0 \\ 0 & 1 & 3 & 2 \end{bmatrix}$	1.9×10^{-10} J	0.96	0.5	0.466	0.353	0.271	1.21×10^{-10} J	6000
$S_9 = \begin{bmatrix} 0 & 4 & 2 & 2 \\ 3 & 0 & 2 & 0 \end{bmatrix}$	2.54×10^{-10} J	0.898	0.484	0.4536	0.344	0.224	1.61×10^{-10} J	9000

In optimization problem (6.2), we fix the same energy level e_g for all quantization bits, and we optimize the protograph for the considered code length. Once we get an optimal protograph, we continue the energy minimization by optimizing the energy per quantization bits using the optimization problem given in Chapter 4:

$$\min_{\epsilon_1,\ldots,\epsilon_q} \mathcal{E}(\xi, S, \epsilon_1, \ldots, \epsilon_q) \quad \text{s.t.} \quad p_{e,\text{opt}}(\xi, S, \epsilon_1, \ldots, \epsilon_q)) < p_{e,max} \tag{6.3}$$

6.2 Optimization method

To solve the first optimization problem (6.2), we rely on the optimization method presented on Section 4.3.2. In this method we use the faulty memory energy model (7.17) to evaluate the decoding energy consumption and we use the BER to evaluate the decoding performance. Once we get the optimized protograph for the considered code length, we use the optimization method presented in Section 5.7.3 to solve the second optimization problem (6.3).

6.3 Numerical results

For the performance criterion, we set an error probability $p_{e,max} = 10^{-3}$ to be achieved at SNR $\xi = 1.45$ dB. For the protograph optimization, we used $R = 0.5$, $n = 2$ and $m = 4$. Table 6.1 gives the optimized protographs for the considered code length ($N = 4000$, $N = 5000$, $N = 6000$, $N = 9000$) and the initial energy consumption $\mathcal{E}_i$ evaluated with the

faulty memory energy model (7.19) using $e_{gk} = 0.9\ \forall k \in \{1, \ldots, q\}$ and $q = 5$ compared to the final energy consumption $\mathcal{E}_f$ evaluated using the optimal energy per bit values $e_{gk}, k \in \{1, \ldots, q\}$. The considered values of e_g and q in the protograph optimization were based on the results presented in table 5.1 of the exhaustive search optimization method presented in Section 5.5.1. In case of $\mathcal{E}_f$ the optimized protograph reduces the decoding energy consumption by 31% up to 32%, while using only 50% to 22% of the nominal energy E_0 in all bits except the MSB.

regarding the performance evaluation, Figure 6.2 provides the BER with respect to SNR of the optimized protographs, where the BER are evaluated with the finite length Density Evaluation method presented in Section 2.2.2 using the optimized energy. As we can see, all the protographs satisfy the target error probability $p_{e,max} = 10^{-3}$ to be achieved at SNR value $\xi = 1.45$dB.

Finally, Figure 6.1 gives the memory energy consumption with respect to SNR for the optimized protographs evaluated with the faulty memory energy model given in (7.19) using the optimized energy per bit values. As expected from Table 6.1, the protograph S_9 has the highest energy consumption while the protograph S_6 gives the lowest energy consumption.

The code length values considered here were set arbitrary in order to evaluate the optimization method. However, compared to the results of Table 7.3, using a non-optimized code length gives the possibility to well optimize the energy level per bit, which allows a better energy minimization.

6.4 Conclusion

In this chapter, we considered that the parameters N and q are fixed, and we optimized both the protograph S and the energy levels per bit, in order to minimize the decoder energy consumption. The proposed optimization method reduces the energy consumption of the optimized protographs by 32% compared to the initial energy consumption.

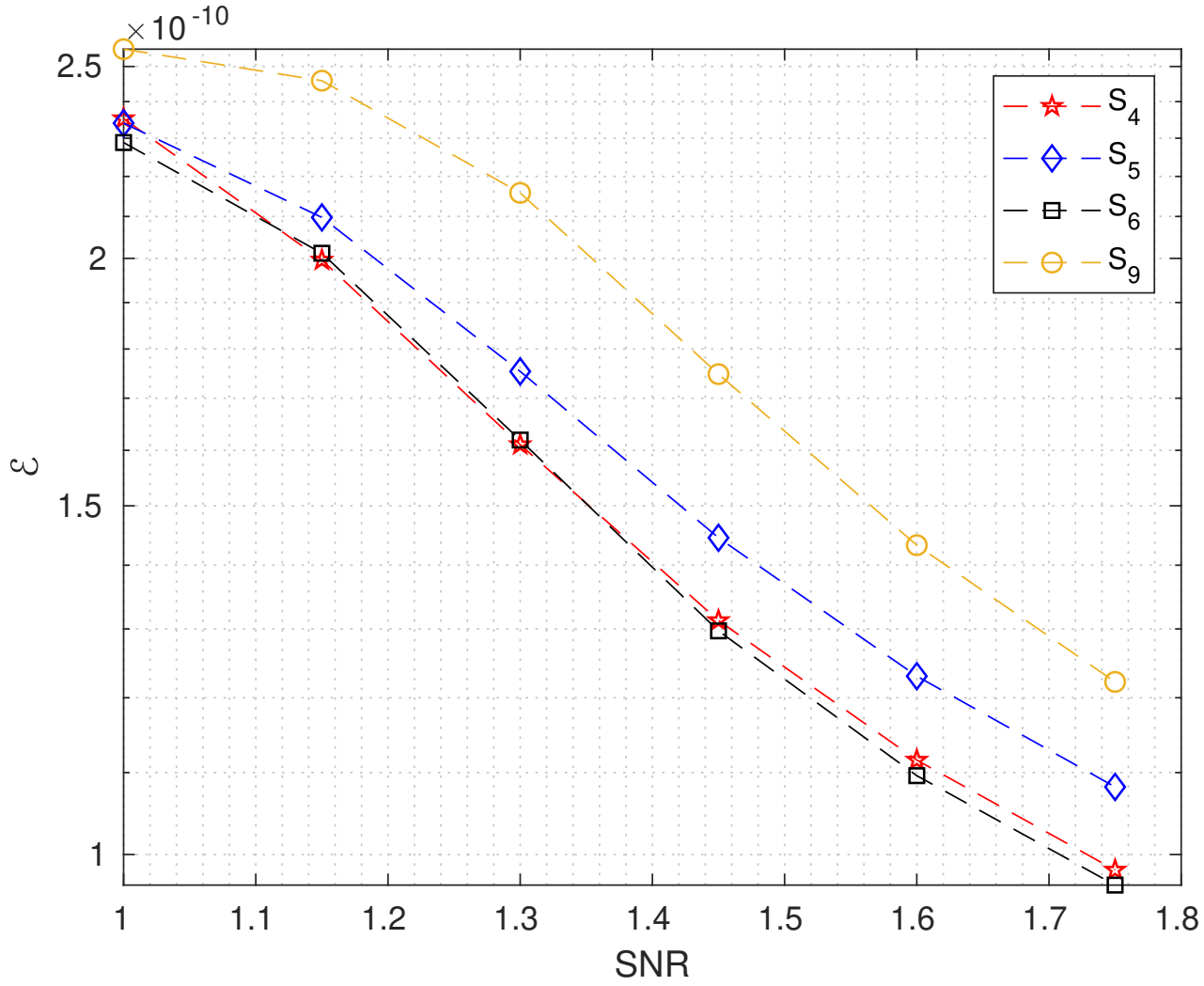

Figure 6.1 – Energy values $\mathcal{E}_{\min}$, $\mathcal{E}_{q_{\text{op}}}$, and $\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$, with respect to SNR, for the protographs S_{17} and S_{36}

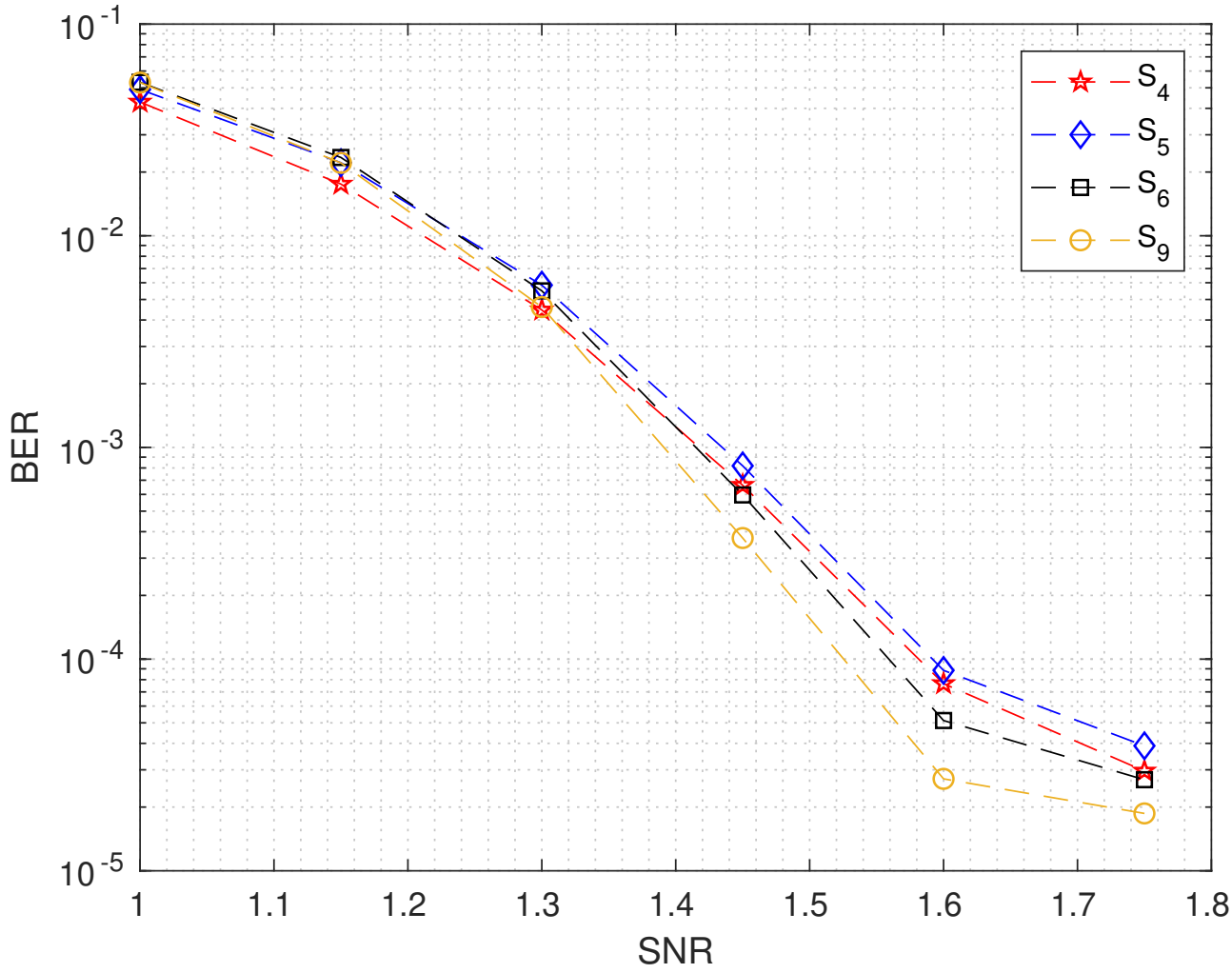

Figure 6.2 – BER with respect to SNR of the protograph S_{17}, evaluated from the finite-length density evolution method and from Monte Carlo simulations.

Conclusion

Summary

There are different types of error correction codes , each of which gives different tradeoffs in terms of decoding performance and energy consumption. We proposed to adress this tradoff for Low-Density Parity Check (LDPC) codes. In this thesis,which was part of the ANR project EF-FECtive, we developed LDPC codes and decoders which allow for a significant reduction in the energy consumption of the decoder, while preserving the same decoding performance.

In Chapter 2, we described LDPC codes and how to construct quasy-cyclic LDPC codes from protographs. We also presented the Sum-Product decoder and the offset Min-Sum decoder. We then showed how to use Density Evolution for quantized min-sum decoder, and protographs. Finally, we presented the finite-length Density evolution method, and we introduced a method to estimate the average iteration number at finite-length. Simulation results showed that the proposed method accurately predict the number of iterations compared to a Monte-Carlo simulation.

In Chapter 3, we introduced two high-level energy models in order to estimate the energy consumption of the quantized Min-Sum decoder. The complexity energy model calculates the total number of operations in the decoding process. The memory energy model calculates the total number of bits stored in memory. In the proposed two models, the energy consumption was related to the code and decoder parameters (protograph, code length, code rate, and number of levels).

In Chapter 4, we proposed an optimization method to minimize the decoder energy consumption with respect to the protograph, while satisfying a given decoding performance constraint. In this method, the energy consumption was evaluated using the energy models proposed in Chapter 3, and the performance was evaluated using the finite-length Density Evolution method. Numerical results showed that the proposed optimization method can provide protographs that minimizes the energy consumption by 15% compared to protographs optimized for performance only.

In Chapter 5, we introduced an energy model for faulty quantized Min-Sum decoder.

We then proposed a method to optimize the number of quantization bits, the code length, and the nominal energy, in order to minimize the energy consumption while satisfying a given decoding performance criterion. Numerical results showed that using the optimal parameters allows to reduce the energy consumption while satisfying the performance criterion. Finally, we optimized the energy per bit for the considered protograph in order to minimize the decoding energy consumption, which allowed to reduce the minimum energy scores by up to 30%.

In Chapter 6, we considered that the parameters N and q are fixed, and we optimized both the protograph S and the energy levels per bit, in order to minimize the decoder energy consumption. The proposed optimization method reduces the energy consumption of the optimized protographs by 32%

Perspectives

We now describe perspectives for this work.

1- Building on the proposed optimization methods: First, as presented in Chapter 3, the proposed energy models depend on the code rate, yet the code rate was fixed in all the proposed optimization methods. So a global optimization including the code rate could be an important step further. Second, in the proposed optimization methods, we fixed the performance criterion,and we then minimized the energy consumption. However, taking this other way by fixing an energy budget and optimizing the decoding performance could also be of great practical interest. The main difficulty in this case would be to set up a reasonable energy budget. We could also consider more realistic energy models, since the proposed methods can easily adapt to other models. Finally, the FER estimation method at finite length could also be improved.

2- Considering others ECC families and decoders: In this work, we focused on LDPC codes and on the quantized Min-Sum decoder.It could be interesting to adapt the proposed optimization methods to other families of codes and decoders so as to identify the best solutions in terms of energy consumption. It would be also interesting to apply the same approach to other signal processing and machine learning methods, to evaluate their performance on unreliable hardware, and to optimize their energy consumption in this case.

3- Using machine learning algorithms to improve the proposed optimizing methods:

The proposed methods provided interesting results in term of energy minimization. However, they were sometimes based on random exploration, and they add some drawbacks which reduced their efficiency.For instance, the protograph optimization method is time consuming, because of the finite-length performance evaluation and the iteration number estimation. Developing new optimization methods based on machine learning algorithms such as Deep Neural Networks may reduce execution time and provide better solutions.

Chapter 7

Résume de la thèse: Modélisation énergétique et optimisation des codes LDPC à base des protographes

7.1 Introduction

Dans la conception des systèmes électroniques, l'efficacité énergétique peut être définie comme la réduction de la quantité d'énergie nécessaire pour réaliser une certaine fonction. L'efficacité énergétique est aujourd'hui un enjeu crucial pour des raisons environnementales, mais aussi pour augmenter la durée de vie des batteries ou pour améliorer les capacités de calcul des systèmes aux ressources limitées. Dans le même temps, de nombreuses applications dans le domaine des télécommunications, du traitement des signaux et de l'apprentissage machine sont très consommatrices d'énergie, en raison de la demande croissante pour ces applications et de la quantité croissante de données à traiter. La consommation d'énergie de ces systèmes peut être réduite soit en travaillant sur leur mise en œuvre sur circuit (conception matérielle), soit en concevant des algorithmes de traitement adaptés à l'efficacité énergétique (conception logicielle). Cette thèse se concentre davantage sur le second aspect.

Dans cette thèse, nous nous concentrons sur l'efficacité énergétique dans les systèmes de télécommunication, plus précisément, sur l'efficacité énergétique des codes de correction d'erreurs (ECC) qui sont utilisés pour le codage des canaux dans la chaîne de communication. Les ECC sont utilisés dans la grande majorité des systèmes de communication car ils permettent une réduction importante de la puissance de l'émetteur. Cependant, les capacités des codes sont souvent limitées par la consommation d'énergie du circuit de décodage. Cela montre que l'optimisation du compromis entre l'énergie de décodage et la

performance est fondamentale dans le problème du codage des canaux.

Nous proposons d'aborder ce compromis pour les codes LDPC (Low-Density Parity Check), qui sont de puissants codes à forte capacité pouvant être décodés avec des décodeurs hautement parallèles pour obtenir des débits de données élevés. Les codes LDPC sont déjà présents dans de nombreuses normes, telles que l'Ethernet 10 Gbps sur les câbles CAT6 (IEEE 802.3an), le WiMAX (IEEE 802.16) et le Wifi (IEEE 802.11n/ac). De plus, ils ont été sélectionnés pour être utilisés dans la norme 5G.

Dans ce travail, nous considérons une version quantifiée du décodeur Min-Sum pour lequel une architecture spécifique a été développée dans le cadre du projet EF-FECtive [1] [57]. Nous développons ensuite des modèles énergétiques de haut niveau pour l'architecture considérée, et nous introduisons des méthodes pour optimiser les paramètres du code et du décodeur afin de minimiser la consommation d'énergie du décodeur sous des contraintes de performance données. Nous adaptons la méthode d'évolution de la densité de longueur finie de [56] à l'architecture considérée, afin d'estimer la performance de décodage à la fois en termes de taux d'erreur sur les bits et en termes de nombre moyen d'itérations.

7.2 Codes et décodeurs LDPC

Les codes LDPC sont un type de codes correcteur d'erreur qui permettent d'obtenir des performances proches de la capacité. Ils ont été introduits pour la première fois par Gallager en 1962 [25] dans sa thèse de doctorat. Dans ce travail, nous considérons les codes LDPC binaires qui peuvent être représentés par une matrice de contrôle de parité binaire $\boldsymbol{H} = \{h_{i,j}\}$de taille $M \times N$, où $h_{i,j}$ est la valeur dans la i-ème ligne et la j-ème colonne de $\boldsymbol{H}$. La matrice $\boldsymbol{H}$ est peu dense en ce sens qu'elle a une faible densité de 1. Le paramètre N donne la longueur du code, $K = N - M$ indique la longueur de l'information si $\boldsymbol{H}$ est de rang complet, et $R = K/N$ est le taux de code.

Alternativement, ici, nous considérons les codes LDPC construits à partir d'un protographs [60]. Un protographe est spécifié par une matrice $\boldsymbol{S}$ de taille $m \times n$ dont les éléments indiquent le nombre d'arêtes reliant les nœuds variables et de contrôle du graphe de Tanner [58] associé à $\boldsymbol{S}$.

Un code LDPC d'une longueur de N et d'un taux de codage de R peut être construit à partir d'un Protographe en appliquant une opération de "copie et permutation" sur le protograph. Le protographe est copié Z fois, où $Z = N/n$ est appelé le facteur de lifting [61]. La matrice de contrôle de parité $\boldsymbol{H}$ (qui sera supposée creuse ci-après) est

ensuite obtenue par entrelacement. La distribution des degrés du code LDPC est celle du protographe, fournie par les entrées de $\boldsymbol{S}$.

7.3 Min-Sum decoder architecture

Dans cette section, nous présenterons l'architecture Min-sum de [1] [57] dont nous tenons compte dans ce travail. Dans l'architecture proposée dans [1], les messages $\beta_i^{(\ell)}$ calculés dans les nœuds variables à l'itération ℓ sont donnés par :

$$\beta_i^{(\ell)} = \Delta(r_i) + \sum_{j \in \mathcal{N}_{v_i}} \gamma_{j \to i}^{(\ell-1)}, \tag{7.1}$$

où $\gamma_{j \to i}^{(\ell)}$ représente le message envoyé du noeud de contrôle c_j au noeud variable v_i à l'itération ℓ. Puis, afin de mettre à jour les messages au niveau du nœud variable, nous utilisons l'équation suivante :

$$\beta_i^{(\ell)} \leftarrow \beta_{i \to j}^{(\ell-1)} + \gamma_{j \to i}^{(\ell-1)} \tag{7.2}$$

Par conséquent, dans cette architecture, à chaque nœud de variable, seule la somme totale des messages entrants $\beta_i^{(\ell)}$ est stockée en mémoire. Par conséquent, lorsque les calculs des nœuds check sont effectués successivement dans le pipeline, ils peuvent bénéficier de la version la plus récente des messages des nœuds variables.

Dans l'architecture considérée, le nœud variable i envoie le même message $\beta_i^{(\ell)}$ à tous ses nœuds check voisins. Le message spécifique $\beta_{i \to j}^{(\ell)}$ de chaque contrôle sera calculé lors de la mise à jour du nœud de contrôle, et les messages de contrôle vers les nœuds variables $\gamma_{j \to i}^{(\ell)}$ sont calculés comme suit:

$$\begin{aligned} \beta_{i \to j}^{(\ell)} &= \beta_i^{(\ell)} - \gamma_{j \to i}^{(\ell-1)} \\ \gamma_{j \to i}^{(\ell)} &= \left(\prod_{i' \in \mathcal{N}_{c_j} \setminus \{i\}} \operatorname{sgn}\left(\beta_{i' \to j}^{(\ell)}\right) \right) \times \max \left[\min_{j' \in \mathcal{N}_{c_j} \setminus \{i\}} \left| \beta_{i' \to j}^{(\ell)} \right| - \lambda, 0 \right], \end{aligned} \tag{7.3}$$

où λ est le paramètre d'offset.

Le décodeur est initialement paramétré avec un nombre de bits de quantification q. Cependant, lors du calcul de la somme dans (7.1), un problème de saturation peut survenir. Pour éviter cette erreur de saturation, nous devons choisir un grand nombre de bits de

quantification, de manière à pouvoir représenter la valeur maximale possible du message donnée par $M_s = \lceil \frac{d_v+1}{2} \rceil |Q|$, avec $|Q| = 2^{q-1}$, et $q_s = \lceil log_2 \left(\lceil \frac{d_v+1}{2} \rceil \right) \rceil - 1$ de sorte que la valeur maximale du message $M_s = 2^{q_s+q}$ se trouve dans l'alphabet de quantification.

7.4 Évolution de Densité

Pour évaluer les performances du décodeur, nous nous appuyons sur Évolution de Densité [53, 54], qui est un outil standard pour évaluer les performances asymptotiques d'un décodeur LDPC. Pour un SNR donné, Évolution de Densité fournit la probabilité d'erreur du décodeur p_{e_∞} moyennée sur l'ensemble des codes décrits par le protographe $\boldsymbol{S}$. Cependant, la probabilité d'erreur p_{e_∞} est évaluée en supposant une longueur de mot de code infinie. Afin de prédire la performance en longueur finie du décodeur Min-Sum quantifié décrit dans la section 7.3, nous nous appuyons sur la méthode proposée dans [56]. Dans cette méthode, afin d'évaluer la probabilité d'erreur du décodeur $p_{e_N}(\xi)$ à SNR ξ pour une longueur de mot de code N, nous utilisons l'équation suivante :

$$p_{e_N}(\xi) = \int_0^{\frac{1}{2}} p_{e_\infty}(x)\, \phi_{\mathcal{N}}\left(x;\, p_0, \frac{p_0(1-p_0)}{N}\right) dx \tag{7.4}$$

Dans cette expression, $p_0 = \frac{1}{2} - \frac{1}{2}\text{erf}\left(\sqrt{\xi/2}\right)$, et $p_{e_\infty}(x)$ est la probabilité d'erreur évaluée avec l'évolution de la densité standard à la valeur SNR $2(\text{erf}^{-1}(1-2x))^2$. La fonction $\phi_{\mathcal{N}}(x; \mu, \sigma^2)$ est la fonction de densité de probabilité d'une variable aléatoire gaussienne avec une moyenne μ et une variance σ^2.

Nous pouvons utiliser la même méthode pour estimer la performance FER du décodeur est évalué pour un mot de code de longueur N, avec SNR ξ, et itération ℓ, à partir de la formule suivante :

$$B_N^{(\ell)}(\xi) = \int_0^{\frac{1}{2}} B_\infty^{(\ell)}(x)\, \Phi_{\mathcal{N}}\left(x; p_0, \frac{p_0(1-p_0)}{N}\right) dx. \tag{7.5}$$

où $B_\infty^{(\ell)}(x) = 1 - (1 - p_{e\infty}^{(\ell)}(x))^N$, et $p_{e\infty}^{(\ell)}(x)$ donne la probabilité d'erreur du décodeur calculée par l'DE standard pour la valeur de variance $v^2 = \frac{1}{2\left(\text{erf}^{-1}(1-2x)\right)^2}$.

Cette méthode tient compte de la variabilité du canal à longueur finie. Elle ne tient pas compte de l'effet des cycles dans la matrice de contrôle de parité du code, qui peut également affecter les performances du décodeur à longueur finie. Cette méthode est donc bien adaptée aux codes de longueur moyenne à longue N. Enfin, cette méthode fonctionne

à la fois pour les codes réguliers, irréguliers ou construits à partir de protographs.

7.4.1 Estimation du nombre d'itérations à longueur finie

Pour évaluer sa consommation d'énergie, il faut également prévoir le nombre d'itérations nécessaires au décodeur à une longueur donnée N. Pour ce faire, nous présentons maintenant notre méthode d'évaluation du nombre moyen d'itérations $L_N(\xi)$ à la longueur N. Pour ce faire, nous utilisons $L_{N,\xi}(j)$ pour indiquer le nombre d'itérations nécessaires pour décoder une trame donnée $j \in [\![1, 2^K]\!]$, où $K = N - M$, en fonction de la longueur du mot de code N et du SNR ξ. Le nombre moyen d'itérations $L_N(\xi)$ peut alors être exprimé comme

$$L_N(\xi) = E\left[L_{N,\xi}^{(\ell)}\right] = \sum_{j=1}^{2^K} \frac{1}{2^K} L_{N,\xi}(j) = \sum_{j=1}^{2^K} \frac{1}{2^K} \sum_{\ell=1}^{L} \mathbb{1}_{\mathcal{A}_{N,\xi}^{(\ell)}}(j) \tag{7.6}$$

où $\mathcal{A}_{N,\xi}^{(\ell)}$ est l'ensemble des trames qui nécessitent au moins l'itération ℓ dans le décodeur, et $\mathbb{1}$ est la fonction d'indicateur. Nous désignons ensuite par $\mathcal{B}_{N,\xi}^{(\ell-1)}$ l'ensemble des trames parfaitement décodées à l'itération $\ell - 1$. Nous supposons que le critère d'arrêt considéré dans le décodeur est parfait, ce qui donne ce $\overline{\mathcal{B}_{N,\xi}^{(\ell-1)}} = \mathcal{A}_{N,\xi}^{(\ell)}$, où $\overline{\mathcal{B}_{N,\xi}^{(\ell-1)}}$ est le complément de $\mathcal{B}_{N,\xi}^{(\ell-1)}$. Avec cette hypothèse, nous avons que

$$L_N(\xi) = \sum_{j=1}^{2^K} \frac{1}{2^K} \sum_{\ell=1}^{L} \left(1 - \mathbb{1}_{\mathcal{B}_{N,\xi}^{(\ell-1)}}(j)\right) \tag{7.7}$$

$$= \sum_{\ell=1}^{L} \left(1 - B_N^{(\ell-1)}(\xi)\right) \tag{7.8}$$

où $B_N^{(\ell-1)}(\xi)$ est défini dans (7.5).

La dérivation ci-dessus repose sur la condition que $\overline{\mathcal{B}_{N,\xi}^{(\ell-1)}} = \mathcal{A}_{N,\xi}^{(\ell)}$. Afin de déterminer s'il doit s'arrêter à l'itération $\ell - 1$, le décodeur s'appuie sur un critère d'arrêt. Cependant, pour certaines trames j, ce critère d'arrêt peut être vérifié alors que la trame n'est pas correctement décodée, ce qui conduit à $j \in \overline{\mathcal{B}_{N,\xi}^{(\ell-1)}}$ mais $j \notin \mathcal{A}_{N,\xi}^{(\ell)}$. La probabilité de cet événement dépend de la distribution du cycle de code et de la distance minimale. Dans ce qui suit, puisque nous considérons de longs mots de code, nous choisissons de négliger cet événement et d'évaluer le nombre moyen d'itérations comme (7.6). La précision de cette approximation sera évaluée ultérieurement à partir de simulations numériques.

La FER (7.5) et le nombre moyen d'itérations (7.8) sont évalués en tenant compte de l'incertitude du canal à la longueur N. Cependant, ces évaluations ne tiennent pas

compte des cycles de code qui peuvent également dégrader les performances du décodeur à longueur finie. Ces méthodes sont donc bien adaptées pour évaluer la performance du code pour des mots de code de taille moyenne à longue, à partir d'environ $3000 bits$.

7.5 Modèles d'énergie

Nous présentons maintenant les deux modèles d'énergie que nous proposons pour cette architecture. .

7.5.1 Modèle d'énergie de complexité

En raison de l'ordonnancement en série C, un nœud variable est mis à jour chaque fois qu'un de ses nœuds check voisins est mis à jour. Considérant que le nœud variable v_i est connecté au nœud de contrôle c_j qui est mis à jour, nous commençons par calculer (7.13), et une fois que le nœud de contrôle a été mis à jour, nous terminons la mise à jour du nœud de variable par

$$\beta_i^{(\ell)} \leftarrow \beta_{i\to j}^{(\ell-1)} + \gamma_{j\to i}^{(\ell-1)},$$

cela nécessite des ajouts de 2 d_{v_i} au cours d'une itération, chacun étant appliqué à des entrées de $q + q_s$ bits.

Pour la mise à jour du nœud de contrôle, le traitement du signe dans (7.14) nécessite des opérations $(2d_{c_j} - 1)$ 2 entrées OU exclusif (XOR-2). Enfin, nous supposons que le calcul des deux valeurs minimales de (7.14) est effectué en utilisant une architecture de circuit de fusion-tri [47]. Ce circuit nécessite $\lfloor \frac{d_{c_j}}{2} \rfloor + 2(\lceil \frac{d_{c_j}}{2} \rceil - 1)$ comparaisons, et toutes les comparaisons sont effectuées sur des entrées de $q - 1$ bits. Afin de dériver le Modèle d'énergie de complexité, nous indiquons par E_{add}, E_{xor}, E_{comp}, la consommation d'énergie élémentaire d'une addition de 1 bit, d'une opération XOR-2 et d'une comparaison de 1 bit, respectivement. Considérons un code LDPC de longueur N, au taux de codage R, et construit à partir d'un Protographe S. Pour un SNR cible ξ et un taux d'erreur sur les bits (TEB) p_e, le Modèle d'énergie de complexité est donné par :

$$E_c = \frac{L_N(\xi, p_e)N}{n}\Bigg(2(q+q_s)E_{\text{add}}\sum_{i=1}^{n} d_{v_i} + (1-R)\left(E_{\text{xor}}\sum_{j=1}^{m}\left(2d_{c_j}-1\right) + E_{\text{comp}}(q-1)\left(\lfloor\frac{d_{c_j}}{2}\rfloor + 2\left(\lceil\frac{d_{c_j}}{2}\rceil - 1\right)\right)\right)\Bigg) \quad (7.9)$$

où $L_N(\xi)$ (7.6) est le nombre total d'itérations utilisées par le décodeur.

7.5.2 Modèle d'énergie mémoire

Pour un nœud de contrôle c avec degré d_c, nous devons stocker un bit de signe pour chaque message de sortie, et deux valeurs minimales de $q-1$ bits chacune. Ainsi, le nombre total de bits stockés est de $d_c + 2q - 2$. Dans un nœud de variable v de degré d_v, il suffit de $\beta_i^{(\ell)}$ pour enregistrer en mémoire, ce qui nécessite $q + q_s$ bits.

Afin de dériver le modèle d'énergie de la mémoire, nous indiquons par E_{bit} la consommation d'énergie élémentaire pour écrire un bit en mémoire. Pour un code LDPC de longueur N et de taux R construit à partir du protographe S, le modèle d'énergie mémoire est donné par

$$E_m = \frac{L_N(\xi, p_e)N}{n} E_{\text{bit}} \left(\sum_{i=1}^{n} d_{v_i} (q + q_s) + (1 - R) \left(\sum_{j=1}^{m} \left(2q + d_{c_j} - 2 \right) \right) \right), \tag{7.10}$$

Les deux modèles énergétiques E_c et E_m dépendent des objectifs SNR et BER ξ et P_e par le nombre moyen d'itérations L_N.

7.6 Problème d'optimisation

Nous voulons trouver le Protographe S minimisant la consommation d'énergie du décodeur tout en maintenant un certain niveau de performance de décodage. D'abord on fixe les paramètres R, N, q m, n, nous fixons un SNR cible ξ et une performance cible à atteindre à ce SNR, et nous optimisons le protographe S. Une fois ces paramètres fixés, nous formulons le problème d'optimisation comme suit

$$\min_{\boldsymbol{S}} E(\boldsymbol{S}) \quad \text{s.t.} \;\; \mathcal{P}(\xi) < \mathcal{P}_{\max} \tag{7.11}$$

Dans (7.11), la fonction énergétique E peut être donnée soit par le Modèle d'énergie de complexité (7.9), soit par le Modèle d'énergie de mémoire (7.10), soit par une combinaison pondérée des deux. $\mathcal{P}(\xi)$ représente la performance de décodage à la valeur SNR ξ et $\mathcal{P}_{\max}$ est la performance cible à atteindre à une valeur SNR ξ. Dans ce qui suit, nous considérons deux exemples de performances de décodage $\mathcal{P}$.

Table 7.1 – Seuils de longueur infinie et valeurs énergétiques de longueur finie des protographs pour $\xi = 1.45$dB et $N = 10000$.

Protographe	Threshold	E_c	E_m
$S_{\text{opt}} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 0 & 4 & 1 \end{bmatrix}$	1.18 dB	19.1 nJ	508 nJ
$S_0 = \begin{bmatrix} 2 & 1 & 3 & 2 \\ 5 & 1 & 1 & 0 \end{bmatrix}$	1.15 dB	28.7 nJ	733 nJ
$S_c = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{bmatrix}$	1.20 dB	21.9 nJ	585 nJ
$S_m = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 4 \end{bmatrix}$	1.21 dB	21.9 nJ	585 nJ

7.7 Algorithme d'évolution différentielle et résultats de simulation

Pour résoudre le problème de l'optimisation 7.11, nous nous appuyons sur l'évolution différentielle [78], qui est un algorithme génétique qui a été initialement introduit pour les fonctions spatiales continues non linéaires et non différentiables. Nous avons adapté cet algorithme à notre problème d'optimisation.

Nous fournissons maintenant des résultats pour la méthode d'optimisation our les protographs de taille $m = 2$, $n = 4$ avec $S_{max} = 6$, les paramètres du code $N = 10^4$, $R = 0,5$, et les paramètres du décodeur $q = 6$, $L = 50$, avec $E_{\text{add}} = 3,13$ fJ, $E_{\text{xor}} = 1,56$ fJ et $E_{\text{bit}} = 0,156$ pJ. Ces valeurs n'affectent pas le résultat de l'optimisation.

Le protographe S_0 a été optimisé sans le critère énergétique. Le protographe S_m et S_c ont été obtenus par la méthode d'optimisation énergétique avec le critère de performance du TEB. S_m a été obtenu en considérant le Modèle d'énergie de la mémoire, et S_c a été obtenu à partir du Modèle d'énergie de la complexité. Enfin, le Protographe S_{opt} a été généré en utilisant la méthode d'optimisation énergétique avec critère de performance du FER. Comme nous pouvons le voir, S_0 atteint un meilleur seuil de SNR, mais S_c et S_m satisfont au critère de seuil de SNR et consomment moins d'énergie selon les deux modèles énergétiques. En revanche, le Protographe S_{opt} satisfait au critère de performance du problème d'optimisation, avec une valeur FER $B_N^L(\xi) = 0,0049$ à $\xi = 1,45dB$. On observe ensuite que S_0 montre une consommation d'énergie plus élevée, ce qui est dû au

fait que ce Protographe a été optimisé pour les performances uniquement. A l'inverse, S_{opt} a la plus faible consommation d'énergie, avec un gain d'environ 15% par rapport à S_0.

7.8 Optimisation de l'énergie pour les décodeurs Min-Sum Défectueux

Dans la partie précédente, nous avons modélisé la consommation d'énergie et nous avons proposé une optimisation du Protographe uniquement. Nous cherchons maintenant à optimiser les autres paramètres, en particulier N et q. La consommation d'énergie peut également être considérablement réduite par une mise à l'échelle agressive de la tension, mais cela peut introduire des fautes dans les opérations de calcul réalisées sur le circuit. Nous mettons à jour le modèle d'énergie de la mémoire afin de l'appliquer à un décodeur défectueux.

7.8.1 Faulty Min-Sum decoder

Dans cette partie, on suppose que le le décodeur est défectueux,et que les fautes sont introduits dans les unités de mémoire, puisque les mémoires sont responsables d'une grande partie de la consommation d'énergie du décodeur [73]. Les opérations variable noeud et noeud check deviennent alors :

$$\widetilde{\beta}_i^{(\ell)} = \Delta(r_i) + \sum_{j \in \mathcal{N}_{v_i}} \widetilde{\gamma}_{j \to i}^{(\ell-1)} + \mathcal{B}, \tag{7.12}$$

$$\widetilde{\beta}_{i \to j}^{(\ell)} = \widetilde{\beta}_i^{(\ell)} - \widetilde{\gamma}_{j \to i}^{(\ell-1)} \tag{7.13}$$

$$\widetilde{\gamma}_{j \to i}^{(\ell)} = \left(\prod_{i' \in \mathcal{N}_{c_j} \setminus \{i\}} \operatorname{sgn}\left(\widetilde{\beta}_{i' \to j}^{(\ell)}\right) \right) \times \max\left[\min_{j' \in \mathcal{N}_{c_j} \setminus \{i\}} \left|\widetilde{\beta}_{i' \to j}^{(\ell)}\right| - \lambda, 0 \right] + \mathcal{B}_{j \to i}^{(\ell)}, \tag{7.14}$$

où $\mathcal{B}_i^{(\ell)}$ est une variable aléatoire qui représente le bruit introduit dans la mémoire. L'équation du message (7.13) n'est pas affectée par le bruit car $\widetilde{\beta}_{i \to j}^{(\ell)}$ n'est pas stocké en mémoire.

7.8.2 Modèle d'énergie

Comme proposé dans [86], le paramètre de défaillance de la mémoire ϵ et l'énergie e dans une unité de mémoire peuvent être mis en relation de la manière suivantes :

$$\epsilon = \epsilon_0 \exp\left(-ce_g\right) \tag{7.15}$$

où $e_g = e^\theta$, e est l'énergie de l'unité de mémoire, $\epsilon_0, \theta \in (0, 1]$ et $c > 0$. Pour relier la valeur de E_{bit}, la consommation d'énergie élémentaire présentée dans le modèle d'énergie de la mémoire défini dans (7.10) au paramètre de bruit ϵ, nous écrivons

$$E_{\text{bit}} = e_g E_0 \tag{7.16}$$

Dans cette expression, e_g représente l'énergie normalisée donnée par (7.15).

Ensuite, puisque dans ce qui suit, parmi d'autres paramètres, nous voulons optimiser la longueur du mot de code N, nous considérons le modèle d'énergie de la mémoire E_m de (7.10) normalisé par le nombre de bits d'information K. Par conséquent, le Modèle d'énergie suivant sera pris en compte dans l'optimisation :

$$\mathcal{E} = \frac{E_m}{K} = \frac{L_N(\xi)}{Rn} e_g E_0 \left(\sum_{i=1}^{n} d_{v_i} \left(q + q_s\right) + (1 - R) \left(\sum_{j=1}^{m} \left(2q + d_{c_j} - 2\right) \right) \right). \tag{7.17}$$

. Cette équation dépend des paramètres q, e_g, etc, qui seront optimisés.

7.8.3 Problème et méthode d'optimisation énergétique

Pour simplifier, nous supposons que R et S sont fixés. Nous proposons de minimiser la consommation d'énergie en ce qui concerne le niveau de quantification q, le paramètre de bruit ϵ et la longueur du mot de code N, tout en satisfaisant au critère de performance. Le problème d'optimisation peut alors être formulé comme suit :

$$\min_{\epsilon, q, N} \mathcal{E}(\xi, q, \epsilon, N) \quad \text{s.t.} \;\; p_{e,\text{opt}}(\xi, \epsilon, q, N) < p_{e,max} \tag{7.18}$$

où

$$p_{e,\text{opt}}(\xi, \epsilon, q, N) = \min_{\alpha, \lambda} p_{e_N}(\xi, \epsilon, q, N).$$

Table 7.2 – Valeur énergétique minimale $\mathcal{E}_{\min}$ et paramètres optimaux pour les quatre prototypes considérés. La partie gauche du tableau représente le cas où les trois paramètres (q, N, ϵ) sont optimisés. La partie droite donne les valeurs énergétiques $\mathcal{E}_{q_{\text{op}}}$ et $\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$, lorsque seul q est optimisé, et lorsque seuls q et N sont optimisés, respectivement.

Protographe	$\mathcal{E}_{min}$	$e_{g_{\text{op}}}$	q_{op}	N_{op}	$\mathcal{E}_{q_{\text{op}}}$	$\mathcal{E}_{N_{\text{op}},q_{\text{op}}}$	N_{op}
$S_{17} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 0 & 1 & 4 & 1 \end{bmatrix}$	1.86×10^{-10} J	0.86	5	3120	2.08×10^{-10} J	2.00×10^{-10} J	3056
$S_{36} = \begin{bmatrix} 2 & 1 & 2 & 3 \\ 1 & 4 & 0 & 1 \end{bmatrix}$	2.06×10^{-10} J	0.85	5	6264	2.21×10^{-10} J	2.22×10^{-10} J	6168
$S_m = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 4 \end{bmatrix}$	1.93×10^{-10} J	0.84	5	8000	2.09×10^{-10} J	2.09×10^{-10} J	10000
$S_{55} = \begin{bmatrix} 1 & 0 & 5 & 1 \\ 3 & 2 & 1 & 1 \end{bmatrix}$	1.93×10^{-10} J	0.88	5	3240	2.18×10^{-10} J	2.062×10^{-10} J	3192
$S_c = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{bmatrix}$	2.11×10^{-10} J	0.88	5	4280	2.35×10^{-10} J	2.22×10^{-10} J	4056

Dans le problème d'optimisation ci-dessus, $\mathcal{E}(\xi, q, \epsilon, N)$ est donné par (7.17) et $p_{e_N}(\xi)$ est défini dans (7.4). En outre, $p_{e,\text{opt}}(\xi, \epsilon, q)$ donne la probabilité d'erreur minimale qui peut être atteinte en optimisant le paramètre d'échelle α et le décalage Min-Sum λ.

Pour résoudre ce problème d'énergie,nous avons utiliser deux méthode. Premièrement,nous avons mis en oeuvre une recherche exhaustive. Cette méthode consiste à tester tous les combinaison possible des paramètre à optimiser pour trouver le meilleur compromis, mais elle a un coût en temps de simulation. Deuxièmement, nous avons implémenté une méthode alternée. Dans cette méthode, nous optimisons une seul paramètre à la fois, et on utilise les nouvelle valeurs des paramètres déjà optimisés dans les optimisations suivantes. Cette méthode est rapide et assez précise.

Dans le tableau 7.2, on observe que, pour chaque protograph, l'énergie minimale est atteinte pour le même niveau de quantification $q_{\text{op}} = 5$ et que les probabilités optimales de défaillance sont proches les unes des autres, *i.e.*, $1,02 \times 10^{-5} < \epsilon_{\text{op}} < 3,38 \times 10^{-5}$, ce qui correspond à peu près à l'utilisation de 80 à 90 % de l'énergie nominale E_0. D'autre part, la longueur de code optimale N_{op} dépend fortement du Protographe considéré.

7.9 Optimisation de l'énergie des bits

Dans les sections précédentes de ce chapitre, nous avons considéré que tous les bits de quantification ont la même énergie e_g. Cependant, les bits n'ont pas la même influence sur

les performances, et c'est pourquoi, dans cette section, nous considérons que chaque bit de quantification k a sa propre énergie e_{gk}, et que le niveau de bruit dépend également de la position du bit. Nous utilisons le même modèle de défaillance que celui présenté dans (7.15). Nous voulons maintenant optimiser les niveaux d'énergie par bit afin de minimiser la consommation d'énergie tout en assurant certaines performances de décodage.

7.9.1 Modèle d'énergie

Afin d'estimer la consommation d'énergie, nous nous appuyons sur le Modèle d'énergie fourni par (7.17), et nous considérons les niveaux d'énergie e_{gk} pour chaque bit k, selon sa position dans les messages quantifiés. Par conséquent, le Modèle d'énergie suivant sera pris en compte dans l'optimisation :

$$\mathcal{E} = \frac{L_N(\xi)}{Rn} E_0 \left(\sum_{i=1}^{n} d_{v_i} \left(\sum_{k=1}^{q} e_{gk} + q_s e_{g2} \right) + (1-R) \left(\sum_{j=1}^{m} \left(2 \sum_{k=2}^{q} e_{gk} + d_{c_j} e_{g1} \right) \right) \right). \quad (7.19)$$

Dans ce modèle, nous considérons que le signe du bit, qui est le bit le plus significatif (MSB) est situé à $k = 1$, et que les bits de $k = 2$ au bit le moins significatif (LSB) $k = q$ représentent la valeur absolue du message quantifié. Nous supposons que les bits supplémentaires q_s utilisés dans le calcul des nœuds variables ont la même énergie e_{g2}.

7.9.2 Problème d'optimisation

Dans cette partie, le Protographe S, et les paramètres R, q et la longueur du code N sont fixés. Nous définissons le critère de performance comme une probabilité d'erreur cible $p_{e,max}$ à atteindre à une valeur SNR ξ. Le problème d'optimisation est donné par :

$$\min_{\epsilon_1,\ldots,\epsilon_q} \mathcal{E}(\xi, \epsilon_1, \ldots, \epsilon_q) \quad \text{s.t.} \quad p_{e,\text{opt}}(\xi, \epsilon_1, \ldots, \epsilon_q) < p_{e,max} \quad (7.20)$$

Dans ce problème d'optimisation, le terme énergétique $\mathcal{E}$ à minimiser est donné par (7.19).

Dans le problème d'optimisation ci-dessus, il y a q paramètres à optimiser afin d'atteindre la consommation d'énergie minimale. Dans ce cas, nous nous appuyons sur la méthode d'optimisation alternée présentée dans la section 7.8.3, où nous optimisons un paramètre ϵ_k à la fois, où $k \in \{1, \ldots, q\}$.

Table 7.3 – La valeur énergétique finale $\mathcal{E}_f$, l'énergie optimale par bit et la valeur énergétique minimale $\mathcal{E}_{\min}$ calculée avec $e_{gk} = e_{gop}(S), \forall k \in \{1, \ldots, q\}$ pour les prototypes considérés à la valeur SNR $\xi = 1,45$dB

Protographe	$\mathcal{E}_{\min}$	e_{g1}	e_{g2}	e_{g3}	e_{g4}	e_{g5}	$\mathcal{E}_f$
$S_{17} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 0 & 1 & 4 & 1 \end{bmatrix}$	1.86×10^{-10} J	1	0.7	0.65	0.65	0.6	1.47×10^{-10} J
$S_{36} = \begin{bmatrix} 2 & 1 & 2 & 3 \\ 1 & 4 & 0 & 1 \end{bmatrix}$	2.06×10^{-10} J	1	0.7	0.8	0.7	0.94	1.77×10^{-10}
$S_m = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 4 \end{bmatrix}$	1.93×10^{-10} J	0.96	0.55	0.55	0.5	0.5	1.36×10^{-10}
$S_{55} = \begin{bmatrix} 1 & 0 & 5 & 1 \\ 3 & 2 & 1 & 1 \end{bmatrix}$	1.93×10^{-10} J	1	0.7	0.98	0.9	0.84	1.70×10^{-10}
$S_c = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{bmatrix}$	2.11×10^{-10} J	1	0.75	0.7	0.75	0.68	1.74×10^{-10}

7.9.3 Résultats

Le tableau 7.3 représente les notes énergétiques finales $\mathcal{E}_f$ calculées en utilisant l'énergie optimale par bit $e_{gk}, k \in \{1, \ldots, q\}$, par rapport aux notes énergétiques minimales $\mathcal{E}_{\min}$ présentées dans le tableau 7.2, et calculé en utilisant $e_{gk} = e_{gop} \forall k \in \{1, \ldots, q\}$ pour les protographes considérés. Comme prévu, le MSB est le bit qui requiert le plus haut niveau d'énergie, proche de l'énergie nominale (90 % à 100 % de E_0) afin de satisfaire au critère de performance. En effet, le signe des messages quantifiés est représenté par le MSB et a une forte influence dans le processus de décodage. A l'inverse, les autres bits peuvent tolérer un niveau de bruit plus élevé, ce qui permet de réduire l'énergie par niveau de ces bits. En conséquence, le score énergétique minimum peut être réduit jusqu'à 30 % comparé au résultats trouves dans la section précédente.

7.10 Conclusion

Dans ce travail, nous avons considéré les codes LDPC construits à partir de protographes avec un décodeur Min-Sum quantifié, pour leurs bonnes performances et leur implémentation matérielle efficace. Nous avons utilisé une méthode basée sur l'évolution de densité pour évaluer les performances à longueur finie du décodeur pour un protographe donné. Ensuite, nous avons introduit deux modèles pour estimer la consommation d'énergie du décodeur Min-Sum quantifié. A partir de ces modèles, nous avons

développé une méthode d'optimisation afin de sélectionner des protographes qui minimisent la consommation d'énergie du décodeur tout en satisfaisant un critère de performance donné.

Dans la seconde partie de la thèse, nous avons considéré un décodeur LDPC bruité, et nous avons supposé que le circuit introduit des fautes dans les unités de mémoire utilisées par le décodeur. Nous avons ensuite mis à jour le modèle d'énergie de la mémoire afin de prendre en compte le bruit dans le décodeur. Par conséquent, nous avons proposé une méthode alternative afin d'optimiser les paramètres du model et minimiser la consommation d'énergie du décodeur pour un protographe donné.

BIBLIOGRAPHY

[1] E. Dupraz, F. Leduc-Primeau, and F. Gagnon, "Low-latency ldpc decoding achieved by code and architecture co-design," in *2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*. IEEE, 2018, pp. 1–5.

[2] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 23, no. 6, pp. 1180–1184, 2014.

[3] R. K. Krishnamurthy and H. Kaul, "Ultra-low voltage technologies for energy-efficient special-purpose hardware accelerators." *Intel Technology Journal*, vol. 13, no. 4, 2009.

[4] D. Jeon, M. Seok, C. Chakrabarti, D. Blaauw, and D. Sylvester, "A super-pipelined energy efficient subthreshold 240 ms/s fft core in 65 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 23–34, 2012.

[5] C. Zhang, D. Wu, J. Sun, G. Sun, G. Luo, and J. Cong, "Energy-efficient cnn implementation on a deeply pipelined fpga cluster," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, 2016, pp. 326–331.

[6] R. M. Swanson and J. D. Meindl, "Ion-implanted complementary mos transistors in low-voltage circuits," *IEEE Journal of Solid-State Circuits*, vol. 7, no. 2, pp. 146–153, 1972.

[7] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[8] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (ntv) design: opportunities and challenges," in *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 1153–1158.

[9] D. Markovic, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow-power design in near-threshold region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237–252, 2010.

[10] J. Chou, D. Petrovic, and K. Ramachandran, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, vol. 2. IEEE, 2003, pp. 1054–1062.

[11] Y. Yang, P. Grover, and S. Kar, "Computing linear transformations with unreliable components," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3729–3756, 2017.

[12] E. Dupraz, V. Savin, S. K. Grandhi, E. Popovici, and D. Declercq, "Practical ldpc encoders robust to hardware errors," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[13] H. Chen, L. R. Varshney, and P. K. Varshney, "Noise-enhanced information systems," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1607–1621, 2014.

[14] Y. Yang, P. Grover, and S. Kar, "Fault-tolerant distributed logistic regression using unreliable components," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 940–947.

[15] E. Dupraz and L. R. Varshney, "Binary recursive estimation on noisy hardware," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 877–881.

[16] J.-C. Vialatte and F. Leduc-Primeau, "A study of deep learning robustness against computation failures," *arXiv preprint arXiv:1704.05396*, 2017.

[17] G. B. Hacene, F. Leduc-Primeau, A. B. Soussia, V. Gripon, and F. Gagnon, "Training modern deep neural networks for memory-fault robustness," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[18] W. Adams, "Report of the iucn renowned thinkers meeting," 2006.

[19] T. Kelly and M. Adolph, "Itu-t initiatives on climate change," *IEEE Communications Magazine*, vol. 46, no. 10, pp. 108–114, 2008.

[20] G. Koutitas and P. Demestichas, "A review of energy efficiency in telecommunication networks," *Telfor journal*, vol. 2, no. 1, pp. 2–7, 2010.

[21] Y. Zhang, P. Chowdhury, M. Tornatore, and B. Mukherjee, "Energy efficiency in telecom optical networks," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 4, pp. 441–458, 2010.

[22] S. N. Roy, "Energy logic: a road map to reducing energy consumption in telecom munications networks," in *INTELEC 2008-2008 IEEE 30th International Telecommunications Energy Conference*. IEEE, 2008, pp. 1–9.

[23] G. Auer, O. Blume, V. Giannini, I. Godor, M. Imran, Y. Jading, E. Katranaras, M. Olsson, D. Sabella, P. Skillermark *et al.*, "D2. 3: Energy efficiency analysis of the reference systems, areas of improvements and target breakdown," *Earth*, vol. 20, no. 10, 2010.

[24] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC'93-IEEE International Conference on Communications*, vol. 2. IEEE, 1993, pp. 1064–1070.

[25] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[26] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[27] A. J. Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181–2191, 1999.

[28] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE transactions on information theory*, vol. 47, no. 2, pp. 638–656, 2001.

[29] ——, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 599–618, 2001.

[30] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding ldpc codes," in *2008 International Symposium on Information Theory and Its Applications*. IEEE, 2008, pp. 1–6.

[31] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[32] 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, 2005.

[33] F. Alberge, "Min-sum decoding of irregular ldpc codes with adaptive scaling based on mutual information," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, pp. 71–75.

[34] B. Le Gal and C. Jego, "High-throughput multi-core ldpc decoders based on x86 processor," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1373–1386, 2016.

[35] B. L. Gal, C. Jego, and J. Crenne, "A high throughput efficient approach for decoding ldpc codes onto gpu devices," *IEEE Embedded Systems Letters*, vol. 6, no. 2, pp. 29–32, 2014.

[36] 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, no. 10, pp. 4033–4045, 2013.

[37] C. Studer, N. Preyss, C. Roth, and A. Burg, "Configurable high-throughput decoder architecture for quasi-cyclic ldpc codes," in *2008 42nd Asilomar Conference on Signals, Systems and Computers*. IEEE, 2008, pp. 1137–1142.

[38] C. Marchand, L. Conde-Canencia, and E. Boutillon, "Architecture and finite precision optimization for layered ldpc decoders," *Journal of Signal Processing Systems*, vol. 65, no. 2, p. 185, 2011.

[39] 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, 2017.

[40] A. Balatsoukas-Stimming, N. Preyss, A. Cevrero, A. Burg, and C. Roth, "A parallelized layered qc-ldpc decoder for ieee 802.11 ad," in *2013 IEEE 11th International New Circuits and Systems Conference (NEWCAS)*. IEEE, 2013, pp. 1–4.

[41] C.-H. Huang, Y. Li, and L. Dolecek, "Gallager b ldpc decoder with transient and permanent errors," in *2013 IEEE International Symposium on Information Theory*. IEEE, 2013, pp. 3010–3014.

[42] S. S. T. Yazdi, H. Cho, and L. Dolecek, "Gallager b decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, no. 5, pp. 1660–1673, 2013.

[43] 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, no. 8, pp. 1299–1302, 2015.

[44] E. Dupraz, D. Declercq, and B. Vasic, "Asymptotic error probability of the gallager b decoder under timing errors," *IEEE Communications Letters*, vol. 21, no. 4, pp. 698–701, 2017.

[45] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of ldpc codes under unreliable message storage," *IEEE Communications Letters*, vol. 18, no. 5, pp. 849–852, 2014.

[46] C. K. Ngassa, V. Savin, E. Dupraz, and D. Declercq, "Density evolution and functional threshold for the noisy min-sum decoder," *IEEE Trans. on Commun.*, vol. 63, no. 5, pp. 1497–1509, 2015.

[47] F. Leduc-Primeau, F. R. Kschischang, and W. J. Gross, "Modeling and energy optimization of LDPC decoder circuits with timing violations," *IEEE Trans. on Commun.*, vol. 66, no. 3, pp. 932–946, 2018.

[48] 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)*, 2015, pp. 1–4.

[49] 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, 2014.

[50] O. Al Rasheed, P. Ivaniš, and B. Vasić, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487–1490, 2014.

[51] 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, no. 4, pp. 906–917, 2016.

[52] E. Dupraz, D. Declercq, B. Vasić, and V. Savin, "Analysis and design of finite alphabet iterative decoders robust to faulty hardware," *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2797–2809, 2015.

[53] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[54] T. Richardson, R. Urbanke *et al.*, "Multi-edge type LDPC codes," in *Workshop honoring Prof. Bob McEliece on his 60th birthday, California Institute of Technology, Pasadena, California*, 2002, pp. 24–25.

[55] R. Yazdani and M. Ardakani, "Waterfall performance analysis of finite-length ldpc codes on symmetric channels," *IEEE transactions on communications*, vol. 57, no. 11, pp. 3183–3187, 2009.

[56] F. Leduc-Primeau and W. J. Gross, "Finite-length quasi-synchronous ldpc decoders," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*. IEEE, 2016, pp. 325–329.

[57] J. Nadal, M. Fiorentino, E. Dupraz, and F. Leduc-Primeau, "A deeply pipelined, highly parallel and flexible ldpc decoder," in *2020 18th IEEE International New Circuits and Systems Conference (NEWCAS)*. IEEE, 2020, pp. 263–266.

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

[59] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

[60] Y. Fang, G. Bi, Y. L. Guan, and F. C. Lau, "A survey on protograph LDPC codes and their applications," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 4, pp. 1989–2016, 2015.

[61] M. Karimi and A. H. Banihashemi, "On the girth of quasi-cyclic protograph ldpc codes," *IEEE transactions on information theory*, vol. 59, no. 7, pp. 4542–4552, 2013.

[62] L. Lan, L. Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic ldpc codes for awgn and binary erasure channels: A finite field approach," *IEEE Transactions on Information Theory*, vol. 53, no. 7, pp. 2429–2458, 2007.

[63] S. Myung, K. Yang, and J. Kim, "Quasi-cyclic ldpc codes for fast encoding," *IEEE Transactions on Information Theory*, vol. 51, no. 8, pp. 2894–2901, 2005.

[64] 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, 2009.

[65] D. G. Mitchell, R. Smarandache, and D. J. Costello, "Quasi-cyclic LDPC codes based on pre-lifted protographs," *IEEE Trans. on Information Theory*, vol. 60, no. 10, pp. 5856–5874, 2014.

[66] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, 2005.

[67] J. Thorpe, K. Andrews, and S. Dolinar, "Methodologies for designing ldpc codes using protographs and circulants," in *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* IEEE, 2004, p. 238.

[68] J. Chen, R. M. Tanner, C. Jones, and Y. Li, "Improved min-sum decoding algorithms for irregular ldpc codes," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.* IEEE, 2005, pp. 449–453.

[69] T. Zhang and K. K. Parhi, "An fpga implementation of-regular low-density parity-check code decoder," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 6, p. 169168, 2003.

[70] N. Wiberg, "Codes and decoding on general graphs," 1996.

[71] E. Sharon, S. Litsyn, and J. Goldberger, "Efficient serial message-passing schedules for LDPC decoding," *IEEE Transactions on Information Theory*, vol. 53, no. 11, pp. 4076–4091, 2007.

[72] K. Ganesan, P. Grover, J. Rabaey, and A. Goldsmith, "On the total power capacity of regular-LDPC codes with iterative message-passing decoders," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 375–396, 2016.

[73] T. T. Nguyen-Ly, K. Le, V. Savin, D. Declercq, F. Ghaffari, and O. Boncalo, "Non-surjective finite alphabet iterative decoders," in *IEEE Int. Conf. on Communications (ICC)*, 2016, pp. 1–6.

[74] K. Ganesan, P. Grover, J. Rabaey, and A. Goldsmith, "On the total power capacity of regular-ldpc codes with iterative message-passing decoders," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 375–396, Feb 2016.

[75] W. Yu, M. Ardakani, B. Smith, and F. Kschischang, "Complexity-optimized low-density parity-check codes for gallager decoding algorithm B," in *2005 Int. Symp. on Information Theory (ISIT)*, 2005, pp. 1488–1492.

[76] B. Smith, M. Ardakani, W. Yu, and F. R. Kschischang, "Design of irregular LDPC codes with optimized performance-complexity tradeoff," *IEEE Trans. on Communications*, vol. 58, no. 2, 2010.

[77] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE Int. Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 10–14.

[78] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[79] F. Ye, E. Dupraz, Z. Mheich, and K. Amis, "Optimized rate-adaptive protograph-based ldpc codes for source coding with side information," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 3879–3889, 2019.

[80] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 876–888, 2009.

[81] P. Gupta, Y. Agarwal, L. Dolecek, N. Dutt, R. K. Gupta, R. Kumar, S. Mitra, A. Nicolau, T. S. Rosing, M. B. Srivastava *et al.*, "Underdesigned and opportunistic computing in presence of hardware variability," *IEEE Transactions on Computer-Aided Design of integrated circuits and systems*, vol. 32, no. 1, pp. 8–23, 2012.

[82] L. R. Varshney, "Performance of ldpc codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, 2011.

[83] F. Leduc-Primeau, F. R. Kschischang, and W. J. Gross, "Energy optimization of ldpc decoder circuits with timing violations," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 412–417.

[84] A. Tarighati, H. Farhadi, and F. Lahouti, "Design of ldpc codes robust to noisy message-passing decoding," *arXiv preprint arXiv:1501.02483*, 2015.

[85] E. Dupraz and F. Leduc-Primeau, "Noisy density evolution with asymmetric deviation models," *arXiv preprint arXiv:2005.05788*, 2020.

[86] A. Chatterjee and L. R. Varshney, "Energy-reliability limits in nanoscale circuits," *arXiv preprint arXiv:1607.03572*, 2016.

[87] M. Yaoumi, F. Leduc-Primeau, E. Dupraz, and F. Guilloud, "Optimization of protograph ldpc codes based on high-level energy models," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2019, pp. 6–10.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Modélisation énergétique et optimisation des codes LDPC à base des protographes

Mot clés : Efficacité énergétique, Codes correcteur d'erreur, LDPC, décodeurs Min-Sum

Résumé : Il existe différents types de codes correcteur d'erreurs (CCE), chacun offrant différents compromis entre la performance et la consumation d'énergie. Nous proposons de traiter ce problème pour les codes LDPC (Low-Density Parity Check).

Dans ce travail, nous avons considéré les codes LDPC construits à partir de protographes avec un décodeur Min-Sum quantifié, pour leurs bonnes performances et leur implémentation matérielle efficace. Nous avons utilisé une méthode basée sur l'évolution de densité pour évaluer les performances à longueur finie du décodeur pour un protographe donné. Ensuite, nous avons introduit deux modèles pour estimer la consommation d'énergie du décodeur Min-Sum quantifié. A partir de ces modèles, nous avons développé une méthode d'optimisation afin de sélectionner des protographes qui minimisent la consommation d'énergie du décodeur tout en satisfaisant un critère de performance donné.

Dans la seconde partie de la thèse, nous avons considéré un décodeur LDPC bruité, et nous avons supposé que le circuit introduit des défauts dans les unités de mémoire utilisées par le décodeur. Nous avons ensuite mis à jour le modèle d'énergie de la mémoire afin de prendre en compte le bruit dans le décodeur. Par conséquent, nous avons proposé une méthode alternative afin d'optimiser les paramètres du model et minimiser la consommation d'énergie du décodeur pour un protographe donné.

Title: Energy modeling and optimization of protograph-based LDPC codes

Keywords: Energy efficiency, Error code correction, LDPC codes, Min-Sum decoders

Abstract: There are different types of error correction codes (CCE), each of which gives different trade-offs interms of decoding performance and energy consumption. We propose to deal with this problem for Low-Density Parity Check (LDPC) codes.

In this work, we considered LDPC codes constructed from protographs together with a quantized Min-Sum decoder, for their good performance and efficient hardware implementation. We used a method based on Density Evolution to evaluate the finite-length performance of the decoder for a given protograph. Then, we introduced two models to estimate the energy consumption of the quantized Min-Sum decoder. From these models, we developed an optimization method in order to select protographs that minimize the decoder energy consumption while satisfying a given performance criterion. The proposed optimization method was based on a genetic algorithm called differential evolution.

In the second part of the thesis, we considered a faulty LDPC decoder, and we assumed that the circuit introduces some faults in the memory units used by the decoder. We then updated the memory energy model so as to take into account the noise in the decoder. Therefore, we proposed an alternate method in order to optimize the model parameters so as to minimize the decoder energy consumption for a given protograph.