Non-surjective finite alphabet iterative decoders

Thien Truong Nguyen-Ly, Khoa Le, Valentin Savin, D Declercq, F Ghaffari, O Boncalo

Non-Surjective Finite Alphabet Iterative Decoders

Thien Truong Nguyen-Ly*†, Khoa Le†, V. Savin*, D. Declercq†, F. Ghaffari† and O. Boncalo‡

*CEA-LETI, MINATEC Campus, Grenoble, France, {thientruong.nguyen-ly, valentin.savin}@cea.fr
†ETIS, ENSEA / CNRS UMR-8051 / University of Cergy-Pontoise, France, {le.khoa, ghaffari, declercq}@ensea.fr
‡University Politehnica Timisoara, Computer Engineering Department, Timisoara, Romania, boncalo@cs.upt.ro

***Abstract*—This paper introduces a new theoretical framework, akin to the use of imprecise message storage in Low Density Parity Check (LDPC) decoders, which is seen as an enabler for cost-effective hardware designs. The proposed framework is the one of Non-Surjective Finite Alphabet Iterative Decoders (NS-FAIDs), and it is shown to provide a unified approach for several designs previously proposed in the literature. NS-FAIDs are optimized by density evolution for WiMAX irregular LDPC codes and we show they provide different trade-offs between hardware complexity and decoding performance. In particular, we derive a set of 27 NS-FAIDs that provide decoding gains up to 0.36 dB, while yielding a memory / interconnect reduction up to 25% / 30% compared to the Min-Sum decoder.**

I. Introduction

This paper targets the design of cost-effective Low Density Parity Check (LDPC) decoders, suitable for the new generation of communication systems, requiring increased data rates and reduced energy footprint. One important characteristic of LDPC decoders is that the memory and interconnect blocks dominate the overall area/delay/power performance of the hardware design. To address this issue, we build upon the concept of Finite Alphabet Iterative Decoders (FAIDs), introduced in [1]–[3]. While FAIDs have been previously investigated for variable-node regular LDPC codes over the binary symmetric channel, this paper extends their use to any channel model, and to both regular and irregular LDPC codes.

The approach considered in this paper is to allow storing the exchanged messages using a lower precision (smaller number of bits) than that used by the processing units. The basic idea is to reduce the size of the exchanged messages, once they have been updated by the processing units. Hence, to some extent, the proposed approach is akin to the use of *imprecise storage*, which is seen as an enabler for cost-effective, high-throughput, and/or low-power decoder designs.

The proposed approach, referred to as *Non-Surjective FAIDs* (NS-FAIDs), is shown to provide a unified framework for several designs previously proposed in the literature, including the Normalized and Offset **M**in-**S**um (MS) decoders [4], [5], the Partially Offset MS decoder [6], the MS-based decoders proposed in [7], [8], or the recently introduced double-quantization domain MS decoder [9].

We show that NS-FAIDs can be optimized by using the Density Evolution (DE) technique, such as to obtain the best possible decoding performance for given hardware constraints, expressed in terms of memory/interconnect reduction. The DE optimization is illustrated for the WiMAX irregular LDPC codes [10], for which we propose a set of 27 *irregular* NS-FAIDs, with different trade-offs between hardware complexity and decoding performance. The DE analysis is further corroborated by Monte-Carlo simulations. We show that NS-FAIDs may provide decoding gains up to 0.36 dB, while yielding a memory / interconnect reduction up to 25% / 30% compared to the MS decoder.

The paper is organized as follows. NS-FAIDs are introduced in Section II, which also discusses their implementation benefits and the DE analysis. The optimization of irregular NS-FAIDs is presented in Section III. Numerical results are provided in Section IV, and Section V concludes the paper.

II. Non-Surjective Finite Alphabet Iterative Decoders

A. Preliminaries

LDPC codes are defined by sparse bipartite graphs, comprising a set of variable-nodes (VNs), corresponding to coded bits, and a set of check-nodes (CNs), corresponding to parity-check equations. Finite Alphabet Iterative Decoders (FAIDs) have been introduced in [1]–[3]. We state below the definition of FAIDs, in a slightly less general form than the one in [3]. Let Q be a positive integer. A $(2Q+1)$-level FAID is a 4-tuple $(\mathcal{M}, \mathcal{I}, \Phi_v, \Phi_c)$, where:

- $\mathcal{M} = \{-Q, \dots, -1, 0, +1, \dots, +Q\}$ is the alphabet of the exchanged messages, and is also referred to as the *decoder alphabet*,
- $\mathcal{I} \subseteq \mathcal{M}$ is the input alphabet of the decoder, *i.e.*, the set of all possible values of the quantized soft information supplied to the decoder,
- Φ_v and Φ_c denote the update rules for VNs and CNs, respectively.

The CN-update function Φ_c is the same for any FAID decoder, and is equal to the update function used by the MS decoder. Precisely, for a CN of degree d_c, the update function $\Phi_c : \mathcal{M}^{d_c-1} \to \mathcal{M}$ is given by:

$$\Phi_c\left(m_1, \dots, m_{d_c-1}\right) = \left(\prod_{i=1}^{d_c-1} \operatorname{sgn}(m_i)\right) \min_{i=1,\dots,d_c-1} |m_i| \quad (1)$$

The VN-update function $\Phi_v : \mathcal{I} \times \mathcal{M}^{d_v-1} \to \mathcal{M}$, for a VN of degree d_v, is defined in closed form as:

$$\Phi_v\left(\gamma, m_1, \dots, m_{d_v-1}\right) = F\left(\gamma + \sum_{j=1}^{d_v-1} m_j\right) \quad (2)$$

where the function $F : \mathbb{Z} \to \mathcal{M}$ is defined based on a set of threshold values $\mathcal{T} = \{T_0, T_1, \ldots, T_{Q+1}\} \subset \bar{\mathbb{R}}_+$, with $T_0 = 0$, $T_{Q+1} = +\infty$, and $T_i < T_j$ for any $i < j$:

$$F(x) = \text{sgn}(x)m, \quad \text{where } m \text{ is s.t. } T_m \leq |x| < T_{m+1} \quad (3)$$

In the following, F will be referred to as the *framing function*.

Definition 1: A q-bit FAID is a $(2Q+1)$-level FAID, with $Q = 2^{q-1} - 1$. It follows that messages exchanged within the FAID decoder are q-bit messages (including 1 bit for the sign).

We also note that the following proposition holds.

Proposition 2: For any function $F : \mathbb{Z} \to \mathcal{M}$, there exists a threshold set $\mathcal{T}$ such that F is given by Eq. (3), if and only if F satisfies the following properties:

(i) $F(-x) = -F(x)$, $\forall x \in \mathbb{Z}$
(ii) F is a non-decreasing function, that is $F(x) \leq F(y)$ for any $x < y$.

Note that the above proposition also implies that $F(0) = 0$ and $F(x) \geq 0, \forall x > 0$.

B. Non-Surjective FAIDs

Definition 3: A non-surjective FAID (NS-FAID) is a FAID such that the framing function $F : \mathbb{Z} \to \mathcal{M}$ is non surjective. Hence, the image set of F, denoted by $\text{Im}(F) \subset \mathcal{M}$ is a strict subset of $\mathcal{M}$.

The reason behind NS-FAIDs is that it allows reducing the size of the memory required to store the exchanged messages, as well as the size of the interconnect network that carries the messages from the memory to the processing units. This will be explained in Section II-C.

Assumption: As the focus of this work is on practical implementations, we will further assume that the sum $\gamma + \sum_{j=1}^{d_v-1} m_j$ in Eq. (2) is saturated to $\mathcal{M}$, prior to applying F on it. Consequently, in the sequel we shall only consider framing functions $F : \mathcal{M} \to \mathcal{M}$, and the VN-update function Φ_v from Eq. (2) is redefined as:

$$\Phi_v\left(\gamma, m_1, \ldots, m_{d_v-1}\right) = F\left(s_{\mathcal{M}}\left(\gamma + \sum_{j=1}^{d_v-1} m_j\right)\right) \quad (4)$$

where $s_{\mathcal{M}} : \mathbb{Z} \to \mathcal{M}$, $s_{\mathcal{M}}(x) = \text{sgn}(x)\min(|x|, Q)$, is the saturation function. Since $F(-x) = -F(x), \forall x \in \mathcal{M}$, F is completely determined by the vector $[F(0), F(1), ..., F(Q)]$, further referred to as the *Look-Up Table (LUT)* of F, which satisfies the following inequalities (Proposition 2):

$$0 = F(0) \leq F(1) \leq \cdots \leq F(Q) \leq Q \quad (5)$$

Definition 4: The weight of F, denoted by W, is the number of distinct entries in the vector $[F(0), F(1), ..., F(Q)]$. It follows that $1 \leq W \leq Q + 1$. By a slight abuse of terminology, we shall also refer to W as the weight of the NS-FAID.

As an example, the framing function in Table I has $Q = 7$, image set $\text{Im}(F) = \{0, \pm 1, \pm 3, \pm 7\}$, and weight $W = 4$.

Table I
FRAMING FUNCTION F WITH LUT = [0, 1, 1, 3, 3, 7, 7, 7]

m	0	1	2	3	4	5	6	7
$F(m)$	0	1	1	3	3	7	7	7

Proposition 5: The number of $(2Q+1)$-level NS-FAIDs of weight W is given by:

$$N_{\text{NS-FAID}}(Q, W) = \binom{Q}{W-1}^2 \quad (6)$$

For the sake of simplicity, we only consider transmission over *binary-input* memoryless noisy channels. We assume that the channel input alphabet is $\mathcal{X} = \{-1, +1\}$, with the usual convention that $+1$ corresponds to the 0-bit and -1 corresponds to the 1-bit, and denote by $\mathcal{Y}$ the output alphabet of the channel. We denote by $\underline{\boldsymbol{x}} = (x_1, \ldots, x_N) \in \{-1, +1\}^N$ the transmitted codeword, and by $\underline{\boldsymbol{y}} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ the received word.

We further consider a function $\varphi : \mathcal{Y} \to \mathcal{I}$ that maps the output alphabet of the channel to the input alphabet of the decoder. Hence, φ encompasses both the computation of the soft (unquantized) log-likelihood ratio (LLR) value and its quantization. By a slight abuse of terminology, we shall refer to φ as *quantization map*.

Assumption: For transmission over the binary-input Additive White Gaussian Noise (AWGN) channel, we shall consider that the decoder's input information and exchanged messages are quantized on the same number of bits; therefore $\mathcal{I} = \mathcal{M}$ unless otherwise stated. In this case, $y_n = x_n + z_n$, where z_n is the white Gaussian noise with variance σ^2, and the quantization map $\varphi : \mathcal{Y} \to \mathcal{M}$ is defined by:

$$\varphi(y) = [\mu \cdot y]_{\mathcal{M}} \quad (7)$$

where $\mu > 0$ is a constant referred to as *gain factor*, and $[x]_{\mathcal{M}}$ denotes the closest integer to x that belongs to $\mathcal{M}$ (see also [11] and the gain factor quantizer defined therein).

C. Implementation benefits

Since F is a non-decreasing function, it can be shown that the framing function F can alternatively be applied at the CN-processing step (instead of VN-processing), while resulting in an equivalent decoding algorithm. Whether F is applied at the VN-processing or the CN-processing step is rather a matter of implementation. When F is applied at the VN-processing step, both VN- and CN-messages belong to a strict subset of the alphabet $\mathcal{M}$, namely $\mathcal{M}' = \text{Im}(F) \subset \mathcal{M}$. This may result in significant memory savings for storing the exchanged messages. It is worth noting that many hardware implementations of Quasi-Cyclic (QC) LDPC decoders rely on a layered architecture, which only requires storing the check-node messages [12].

Proposition 6: Consider a particular implementation of the MS decoder, with exchanged messages of size q-bits. By applying a framing function with weight W (which turns the MS decoder into a NS-FAID), exchanged messages can be

represented using only $w = \lceil \log_2(W) \rceil + 1$ bits (including 1 bit for the sign).

We shall refer to the above w value as the *framing bit-length*. As an example, the framing function from Table I has bit-length $w = \lceil \log_2 4 \rceil + 1 = 3$ bits. As a consequence of the message size reduction, the size of the memory and the interconnect network that carries the messages from the memory to the processing units are also reduced.

A different benefit is the possible simplification of the CN processing unit. Since incoming messages span only a subset $\mathcal{M}' = \text{Im}(F) \subset \mathcal{M}$, depending on $\mathcal{M}'$, it may simplify the min computation within the CN-processing. Let us give a particular example to illustrate this phenomenon. Assume that $Q = 7$ and F is such that $\mathcal{M}' = \{0, \pm 1, \pm 3, \pm 7\}$ (*e.g.*, framing function from Table I). In this case, it can be easily seen that the $\min_{i=1,\ldots,d_c-1} |m_i|$ computation required by the CN-processing (Eq. (1)) is equal to the bit-wise AND of the absolute values $|m_i|, i = 1, \ldots, d_c - 1$. Hence, the CN-processing can be implemented using AND logic gates only, thus avoiding the use of comparator trees as in [9], [13].

D. Examples of NS-FAIDs

If the framing function F is the identity function, then the corresponding FAID is just the MS decoder with finite alphabet $\mathcal{M}$. Some examples of NS-FAIDs are provided below.

Example 1. Let $F : \mathcal{M} \to \mathcal{M}$ be defined by:

$$F(x) = \text{sgn}(x) \max(|x| - \lambda, 0) \tag{8}$$

where $0 < \lambda < Q$. Then, the corresponding NS-FAID decoder is the Offset Min-Sum (OMS) decoder with offset factor λ.

Example 2. Let $F : \mathcal{M} \to \mathcal{M}$ be defined by:

$$F(x) = \begin{cases} x, & \text{if } |x| \text{ is even} \\ \text{sgn}(x)(|x| - 1), & \text{if } |x| \text{ is odd} \end{cases} \tag{9}$$

Then, the corresponding NS-FAID decoder is the Partially OMS (POMS) decoder from [6].

Moreover, it can be seen that the MS-based decoders proposed in [7], [8] and the dual-quantization domain decoder proposed in [9] are particular realizations of NS-FAIDs.

While the main reason behind the NS-FAIDs definition consists in their ability to reduce memory and interconnect requirements, we can also argue that they may allow improving the error correction performance (with respect to MS). This is the case of both OMS and POMS decoders mentioned above. Given a target message bit-length w (*e.g.*, corresponding to some specific memory constraint), one may try to find the framing function F of corresponding weight W, which yields the best error correction performance. The optimization of the framing function can be done by using the DE technique, which will be discussed in Section II-F.

E. Irregular NS-FAIDs

In case of irregular LDPC codes, *irregular NS-FAIDs* are NS-FAIDs using different framing functions F_{d_v} for VNs of different degrees d_v. Framing functions F_{d_v} may have different weights W_{d_v}. In this case, messages outgoing from degree-d_v VNs can be represented by using only $w_{d_v} = \lceil \log_2(W_{d_v}) \rceil + 1$ bits. However, the message size reduction does not necessarily apply to CN-messages, due to the fact that a CN may be connected to VNs of different degrees. Let $\mathcal{M}'_{d_v} = \text{Im}(F_{d_v})$. Then, messages outgoing from a CN c can be represented by using:

$$\left\lceil \log_2 \left(\left| \cup_{d_v \in \mathcal{D}_c} \mathcal{M}'_{d_v} \right| \right) \right\rceil \text{ bits}, \tag{10}$$

where $\mathcal{D}_c$ is the set of degrees of VNs connected to c, and $|\cdot|$ is used to denote the number of elements of a set.

One advantage of irregular NS-FAIDs is that they allow different protection levels for VNs of different degrees (through the use of framing functions F_{d_v} with different weights). Alternatively, it is also possible to define CN-irregular NS-FAIDs in a similar manner. However, in this work we only deal with VN-irregular NS-FAIDs, since most of the practical irregular LDPC codes are irregular on VNs, while almost regular (or semi-regular) on CNs. In order to reduce the size of the CN-messages, in Section III we will further impose certain conditions on the framing functions F_{d_v}, by requiring their images being included in one another.

F. Density Evolution Analysis

The objective of the DE technique is to recursively compute the probability mass function (pmf) of the exchanged messages, through the iterative decoding process. This is done under the assumption that exchanged messages are independent, which holds in the asymptotic limit of the code length. In this case, the decoding performance converges to the cycle free case. DE equations for the NS-FAID decoder can be derived in a similar way as for the finite-alphabet MS decoder [11, Appendix B]. The only modification required is to take into account the framing function F applied at the VN-processing step, which can be easily done using the following:

Proposition 7: Let $A^{(\ell)}_{\text{MS}}$ denote the pmf of VN-messages $\alpha_{\text{MS}} \stackrel{\text{def}}{=} s_{\mathcal{M}} \left(\gamma + \sum_{j=1}^{d_v - 1} m_j \right)$ computed by the MS decoder at iteration ℓ, and $F : \mathcal{M} \to \mathcal{M}$ be a framing function. Then the pmf of $\alpha_{\text{NS-FAID}} = F(\alpha_{\text{MS}})$, denoted by $A^{(\ell)}_{\text{NS-FAID}}$, is given by:

$$A^{(\ell)}_{\text{NS-FAID}}(m) = \sum_{x \in \mathcal{M} : F(x) = m} A^{(\ell)}_{\text{MS}}(x)$$

Similar to [11], the DE is used to compute the asymptotic error probability, defined as:

$$p_e^{(+\infty)} = \lim_{\ell \to +\infty} p_e^{(\ell)} \tag{11}$$

where $p_e^{(\ell)}$ is the bit error probability at iteration ℓ.

For a target bit error probability $\eta > 0$, the *η-threshold* is defined as the worst channel condition for which decoding error probability is less than η. Assuming the binary-input AWGN channel model, the η-threshold corresponds to the maximum noise variance σ^2 (or equivalently minimum SNR), such that the asymptotic error probability is less than η:

$$\sigma^2_{\text{thres}}(\eta) = \sup \left\{ \sigma^2 \mid p_e^{(+\infty)} \leq \eta \right\} \tag{12}$$

In case that $\eta = 0$, the η-threshold is simply referred to as DE threshold [14]. However, the asymptotic decoding performance of finite-precision MS-based decoders is known to exhibit an error floor phenomenon at high SNR [11]. This makes the η-threshold definition more appropriate in practical cases, when the target bit error rate can be fixed to a practical non-zero value.

Finally, it is worth noting that the above threshold value depends on: (i) the irregularity of the LDPC codes, defined as usual by the degree distribution polynomials λ and ρ [14], (ii) the FAID decoder, *i.e.*, the size of the decoder alphabet and the framing function F, (iii) the channel quantizer, *i.e.*, the gain factor μ used in Eq. (7). Therefore, assuming that the degree distribution polynomials λ and ρ and the size of the decoder alphabet are fixed, we use the DE technique to **jointly optimize** the framing function and channel quantizer.

III. Optimization of Irregular NS-FAIDs

A. Optimization procedure

Throughout this section, we consider $q = 4$-bit NS-FAIDs (hence, $Q = 7$). To illustrate the trade-off between hardware complexity and decoding performance, we consider the optimization of irregular NS-FAIDs for the WiMAX irregular LDPC codes with rate $1/2$ [10] (of course, the proposed method can be applied to any other irregular codes in the same manner). The *edge-perspective* degree distribution polynomials are given by $\lambda(x) = 0.2895x + 0.3158x^2 + 0.3947x^5$ and $\rho(x) = 0.6316x^5 + 0.3684x^6$. Hence, VNs are of degree $d_v \in \{2, 3, 6\}$. For each VN-degree d_v, we consider that the corresponding framing function F_{d_v} may be of any weight $W_{d_v} \in \{2, 4, 8\}$, corresponding to a message bit-length $w_{d_v} \in \{2, 3, 4\}$. Hence, the total number of framing functions is given by $N_{\text{NS-FAID}}(7, 2) + N_{\text{NS-FAID}}(7, 4) + N_{\text{NS-FAID}}(7, 8) = 1275$ (see Proposition 5). It follows that the total number of irregular NS-FAIDs is equal to $1275^3 = 2072671875$, since a different framing function may be applied for each VN-degree.

Clearly, even though we rely on DE, it is practically impossible to evaluate the decoding performance of all the irregular NS-FAIDs. To overcome this problem, we proceed as follows. First, we denote by NS-FAID-$w_2w_3w_6$ the ensemble of NS-FAIDs defined by a triplet of framing functions F_2, F_3, F_6, corresponding to variable node-degrees $d_v = 2, 3, 6$, with message bit-lengths w_2, w_3, w_6. Since $w_2, w_3, w_6 \in \{2, 3, 4\}$, there are 27 such ensembles. The number of NS-FAIDs in the ensemble NS-FAID-$w_2w_3w_6$ is given by $N_{w_2}N_{w_3}N_{w_6}$, where $N_w \stackrel{\text{def}}{=} N_{\text{FAID}}(Q, W)$. Note that $N_2 = 49$, $N_3 = 1225$, and $N_4 = 1$. These 27 ensembles are further divided into 2 groups, as follows:

Group-1: is composed of NS-FAID-$w_2w_3w_6$ ensembles, such that there is at least one $w_{d_v} = 4$ and at most one $w_{d_v} = 3$. In total, there are 16 NS-FAID-$w_2w_3w_6$ ensembles in this group (example: NS-FAID-443, NS-FAID-224, NS-FAID-432,...). It can be easily seen that the total number of NS-FAIDs in Group-1 (all 16 ensembles included) is equal to $N_{\text{NS-FAIDs}}^{\text{Group-1}} = 371176$. This number is small enough, so that the decoding performance of all the decoders in Group-1 can be evaluated by DE. It is worth noting that for NS-FAIDs in Group-1, the size of CN-messages cannot be reduced below $q = 4$: this follows from Eq. (10), since $\mathcal{D}_c = \{2, 3, 6\}$, for any CN c, and at least one $w_{d_v} = 4$. Therefore, these decoders present no advantages in terms of the memory size required for storing the CN-messages, or complexity of the interconnection network to carry CN-messages from CNU to VNU and/or memory.

Group-2: the remaining 11 NS-FAID-$w_2w_3w_6$ ensembles that do not belong to Group-1 (example: NS-FAID-333, NS-FAID-332, NS-FAID-433,...). Since the number of NS-FAIDs in Group-2 is still very large, we only evaluate part of them, by further imposing a number of constraints on the framing functions' images $\mathcal{M}'_{d_v} = \text{Im}(F_{d_v})$.

Decoding performance constraint: Using results from the exhaustive evaluation of NS-FAIDs in Group-1, we observed that good framing functions with $w_{d_v} = 3$ have as image set one of $\{0, \pm1, \pm2, \pm6\}$, $\{0, \pm1, \pm2, \pm7\}$, $\{0, \pm1, \pm3, \pm6\}$, or $\{0, \pm1, \pm3, \pm7\}$ (note that different framing functions may have the same image set). Hence, we only consider the NS-FAIDs in Group-2 such that $\text{Im}(F_{d_v})$ is one of the above 4 image sets, for any d_v such that $w_{d_v} = 3$.

Memory size reduction constraint: We further impose the following inclusion constraint between the image sets of framing functions used for different VN-degrees: Let d'_v, d''_v, d'''_v denote the VN-degrees, ordered according to increasing framing bit-length, that is $w_{d'_v} \leq w_{d''_v} \leq w_{d'''_v}$. Then we require that $\text{Im}\left(F_{d'_v}\right) \cup \text{Im}\left(F_{d''_v}\right) \subseteq \text{Im}\left(F_{d'''_v}\right)$. According to Eq. (10), this constraint ensures that CN-messages can be represented by using only $w_{d'''_v}$ bits, which is particularly suitable for layered architectures (in which case only CN-messages are stored).

It can be seen that the number of irregular NS-FAIDs in Group-2 that satisfy the above two constraints (and hence are evaluated by DE) is given by $N_{\text{NS-FAIDs}}^{\text{Group-2}} = 726621$.

B. Density Evolution evaluation

For each of the above $N_{\text{NS-FAIDs}}^{\text{Group-1}} + N_{\text{NS-FAIDs}}^{\text{Group-2}} = 1097797$ irregular NS-FAIDs, we compute its decoding threshold for a target bit error rate $\eta = 10^{-6}$, using the DE technique from Section II-F. The threshold computation also encompasses the optimization of the channel gain factor μ. Hence, for each NS-FAID, we first determine the gain factor μ that maximizes the η-threshold defined in Eq. (12). The corresponding η-threshold value is then reported as the η-threshold of the NS-FAID.

IV. Numerical Results

This section provides the numerical results for the irregular NS-FAID optimization from the previous section. Table II show the NS-FAID decoder with the best η-threshold for each ensemble NS-FAID-$w_2w_3w_6$: the framing functions used for VN-degrees $d_v = 2, 3, 6$ are shown in columns 3, 4, and 5, while the η-threshold value (in dB) and the corresponding gain factor μ are shown in column 6. The framing functions' LUTs are reported in Table III (Lx in Table III corresponds to LUTx in Table II).

Table II
HARDWARE COMPLEXITY VS. DECODING PERFORMANCE TRADE-OFF FOR OPTIMIZED NS-FAIDS

Decoder Index	NS-FAIDs Ensemble	Framing functions applied to			SNR-thres (dB) & gain factor μ @BER $= 10^{-6}$	#Connects VNs-CNs	Memory size (bits)		Perform gain/loss (dB)	Connects reduction (%)	Memory reduction (%)	
		$d_v = 2$	$d_v = 3$	$d_v = 6$			store all messages	min1,min2 index,signs			store all messages	min1,min2 index,signs
Group 1												
1	**MS-444**	**LUT0**	**LUT0**	**LUT0**	**1.38-μ3.2**	**58368**	**29184**	**17664**	**0**	**0**	**0**	**0**
2	NS-FAID-443	LUT0	LUT0	LUT6	1.07-μ2.9	55488	29184	17664	+0.31	-4.93	0	0
3	NS-FAID-434	LUT0	LUT5	LUT0	1.08-μ2.7	56064	29184	17664	+0.30	-3.95	0	0
4	NS-FAID-344	LUT8	LUT0	LUT0	1.21-μ2.5	56256	29184	17664	+0.17	-3.62	0	0
5	NS-FAID-442	LUT0	LUT0	LUT19	1.48-μ3.2	52608	29184	17664	-0.10	-9.87	0	0
6	NS-FAID-424	LUT0	LUT15	LUT0	1.77-μ2.6	53760	29184	17664	-0.39	-7.89	0	0
7	NS-FAID-244	LUT16	LUT0	LUT0	2.93-μ3.2	54144	29184	17664	-1.55	-7.24	0	0
8	NS-FAID-432	LUT0	LUT11	LUT16	1.35-μ2.6	50304	29184	17664	+0.03	-13.82	0	0
9	NS-FAID-423	LUT0	LUT16	LUT12	1.60-μ2.9	50880	29184	17666	-0.22	-12.83	0	0
10	NS-FAID-324	LUT2	LUT15	LUT0	1.91-μ2.4	51648	29184	17664	-0.53	-11.5	0	0
11	NS-FAID-342	LUT8	LUT0	LUT16	1.53-μ2.4	50496	29184	17664	-0.15	-13.49	0	0
12	NS-FAID-234	LUT16	LUT8	LUT0	2.94-μ3.2	51840	29184	17664	-1.56	-11.18	0	0
13	NS-FAID-243	LUT15	LUT0	LUT1	2.98-μ2.5	51264	29184	17664	-1.60	-12.17	0	0
14	NS-FAID-422	LUT0	LUT19	LUT20	2.06-μ4.1	48000	29184	17664	-0.68	-17.76	0	0
15	NS-FAID-242	LUT16	LUT0	LUT17	2.93-μ3.3	48384	29184	17664	-1.55	-17.11	0	0
16	NS-FAID-224	LUT13	LUT15	LUT0	3.16-μ2.5	49536	29184	17664	-1.78	-15.13	0	0
Group 2												
17	**NS-FAID-433**	**LUT0**	**LUT9**	**LUT5**	**1.02-μ2.8**	**53184**	**29184**	**17664**	**+0.36**	**-8.88**	**0**	**0**
18	NS-FAID-343	LUT8	LUT0	LUT5	1.09-μ2.3	53376	29184	17664	+0.29	-8.55	0	0
19	NS-FAID-334	LUT8	LUT5	LUT0	1.11-μ2.3	53952	29184	17664	+0.27	-7.57	0	0
20	NS-FAID-233	LUT16	LUT7	LUT7	3.03-μ3.2	41664	21888	15360	-1.65	-28.62	-25.00	-13.04
21	NS-FAID-323	LUT7	LUT16	LUT10	1.78-μ2.6	41472	21888	15360	-0.40	-28.95	-25.00	-13.04
22	**NS-FAID-332**	**LUT8**	**LUT11**	**LUT17**	**1.43-μ2.8**	**40896**	**21888**	**15360**	**-0.05**	**-29.93**	**-25.00**	**-13.04**
23	**NS-FAID-333**	**LUT8**	**LUT8**	**LUT8**	**1.13-μ2.4**	**43776**	**21888**	**15360**	**+0.25**	**-25.00**	**-25.00**	**-13.04**
24	NS-FAID-223	LUT16	LUT18	LUT3	3.14-μ3.4	39360	21888	15360	-1.76	-32.57	-25.00	-13.04
25	NS-FAID-232	LUT16	LUT7	LUT16	3.03-μ3.2	38784	21888	15360	-1.65	-33.55	-25.00	-13.04
26	NS-FAID-322	LUT4	LUT18	LUT18	2.29-μ3.5	38592	21888	15360	-0.91	-33.88	-25.00	-13.04
27	**NS-FAID-222**	**LUT13**	**LUT14**	**LUT14**	**3.27-μ2.5**	**29184**	**14592**	**13056**	**-1.89**	**-50.00**	**-50.00**	**-26.09**

Table III
LUTS USED BY NS-FAIDS IN TABLE II

m	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
2	2	2	2	2	3	1	1	1	1	1	1	1	1	4	0	0	0	0	0	0	0
3	3	7	2	2	3	2	2	3	3	3	3	3	1	4	4	5	6	7	0	0	0
4	4	7	6	2	3	2	2	3	3	3	3	3	3	4	4	5	6	7	6	7	0
5	5	7	6	2	6	7	2	6	7	3	3	3	3	4	4	5	6	7	6	7	7
6	6	7	6	6	6	7	7	6	7	7	3	3	3	4	4	5	6	7	6	7	7
7	7	7	6	6	6	7	7	6	7	7	6	7	7	4	4	5	6	7	6	7	7
w	4	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2

The number of interconnects (wires) reported in column 7 corresponds to the number of wires required to carry both VN- and CN-messages, and is computed by the formula:

$$\#\text{Wires} = \sum_{d_v} N_{d_v} d_v w_{d_v} + (\max_{d_v} w_{d_v}) \sum_{d_c} N_{d_c} d_c \qquad (13)$$

where N_{d_v} and N_{d_c} denote the number of VNs and CNs of degree d_v and d_c, respectively. This formula accounts for the fact that there are d_v outgoing messages of bit-length w_{d_v} from each VN of degree d_v, and d_c outgoing messages from each CN of degree d_c, each CN-message being of bit-length $\max_{d_v} w_{d_v}$.

Columns 8 and 9 show the size of the memory required to store CN-messages (we only take into account CN-messages, because most of the practical hardware implementations do not store VN-messages). In column 8 all CN-messages are assumed to be stored, while column 9 assumes a *compressed format*, which only requires the storage of the signs, first and second minima, and the index of the first minimum [15]. Finally, the last 4 columns show the relative performance gain/loss (in dB) and the percentage of interconnect and memory reduction with respect to the MS decoder (which corresponds to the NS-FAID-444 decoder).

The trade-off between decoding performance and interconnect reduction is further illustrated in Figure 1 (the decoder index on the abscissa is the one from Table II). It can be seen that the NS-FAID-332 decoder (index 22) allows a significant reduction of the interconnection network (by 29.93%), with negligible performance degradation (0.05 dB) with respect to the MS decoder. It is also worth noting that it also provides up to 25% memory size reduction compared to MS. In addition, it can be observed that the amplitude of any VN-message belongs to $\{0, 1, 3, 7\}$. With these values, the min computation required by the CN-processing unit can be implemented by

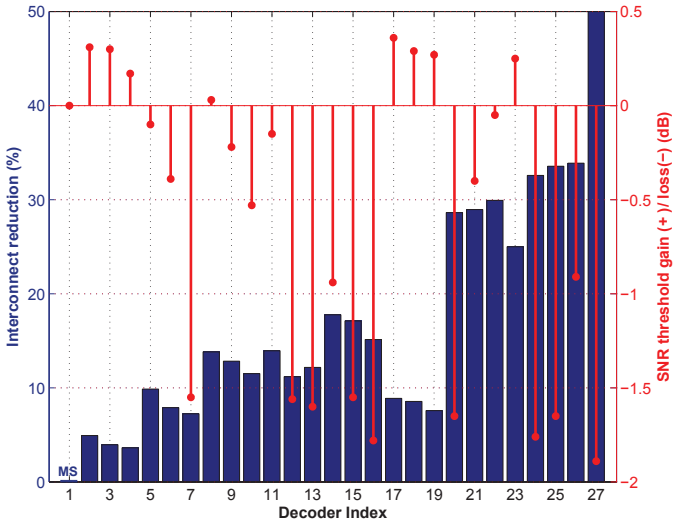

Figure 1. Interconnect reduction vs. decoding performance

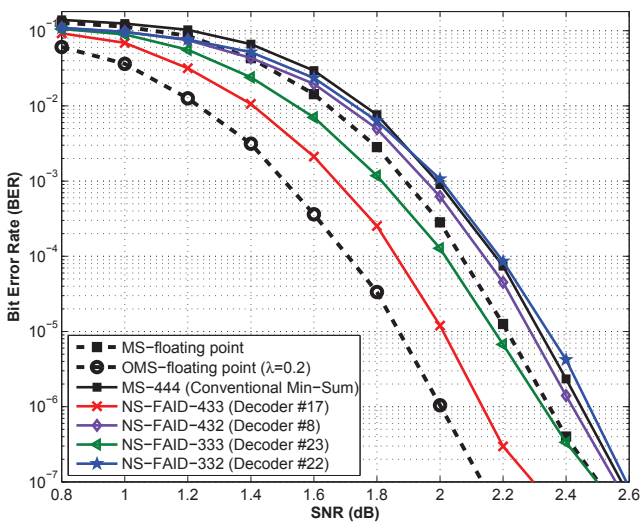

Figure 2. BER performance of proposed NS-FAIDs

using AND-gates only, as explained in Section II-C.

The NS-FAID-433 decoder (index 17) is also a very good candidate for applications requiring increased decoding performance: it achieves the best decoding gain (0.36 dB), while still providing an interconnect reduction by 8.88% with respect to the MS decoder.

To corroborate the analytic results obtained by DE, we have conducted Monte Carlo simulations for the WiMAX code with rate 0.5, and codeword length 2304 bits. The maximum number of decoding iterations is set to 100. The bit error rate (BER) performance of some of the optimized NS-FAIDs is shown in Figure 2. For comparison purposes, we have also included the BER performance of the infinite-precision (floating point) MS and Offset-MS decoders. It can be seen that simulation results corroborate the asymptotic analysis: at BER $= 10^{-6}$, the NS-FAID-433 decoder outperform the MS decoder by ≈ 0.31 dB, while the NS-FAID-332 decoder presents virtually the same decoding performance as the MS.

V. Conclusions

In this paper, we introduced the new framework of Non-Surjective FAIDs, which allows trading off decoding performance for hardware complexity reductions. NS-FAIDs have been optimized by density evolution and we showed that they exhibit better or similar decoding performance compared to the MS decoder, while providing significant savings in memory and/or interconnects. While this paper focused on the theoretical aspects of NS-FAIDs, hardware implementation results will be reported in future works.

Acknowledgment

The authors acknowledge support from the European H2020 Work Programme, project mmMagic, and the Franco-Romanian (ANR-UEFISCDI) Joint Research Programme "Blanc-2013", project DIAMOND.

References

[1] S. K. Planjery, S. K. Chilappagari, B. Vasić, D. Declercq, and L. Danjean, "Iterative decoding beyond belief propagation," in *IEEE Information Theory and Applications Workshop (ITA)*, 2010, pp. 1–10.

[2] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasic, "Finite alphabet iterative decoders for LDPC codes surpassing floating-point iterative decoders," *IET Electronics Letters*, vol. 47, no. 16, pp. 919–921, 2011.

[3] ——, "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.

[4] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. on Communications*, vol. 53, no. 8, pp. 1288–1299, 2005.

[5] V. Savin, *Channel coding: Theory, algorithms, and applications*. Elsevier, 2014, ch. LDPC Decoders, pp. 211–260.

[6] T. Nguyen-Ly, L. Khoa, F. Ghaffariy, A. Amaricai, O. Boncalo, V. Savin, and D. Declercq, "FPGA design of high throughput LDPC decoder based on imprecise offset min-sum decoding," in *IEEE International New Circuits And Systems Conference (NEWCAS)*, June 2015.

[7] D. Oh and K. K. Parhi, "Min-sum decoder architectures with reduced word length for LDPC codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 1, pp. 105–115, 2010.

[8] V. A. Chandrasetty and S. M. Aziz, "An area efficient LDPC decoder using a reduced complexity min-sum algorithm," *Integration, the VLSI Journal*, vol. 45, no. 2, pp. 141–148, 2012.

[9] S. Abu-Surra, E. Pisek, T. Henige, and S. Rajagopal, "Low-power dual quantization-domain decoding for LDPC codes," in *IEEE Global Communications Conference (GLOBECOM)*, 2014, pp. 3151–3156.

[10] IEEE-802.16e, "Physical and medium access control layers for combined fixed and mobile operation in licensed bands," 2005, amendment to Air Interface for Fixed Broadband Wireless Access Systems.

[11] Z. Mheich, T. Nguyen-Ly, V. Savin, and D. Declercq, "Code-aware quantizer design for finite-precision min-sum decoders," in *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, Varna, Bulgaria, June 2016, submitted.

[12] E. Boutillon and G. Masera, *Channel coding: Theory, algorithms, and applications*. Elsevier, 2014, ch. Hardware Design and Realization for Iteratively Decodable Codes, pp. 583–642.

[13] C.-L. Wey, M.-D. Shieh, and S.-Y. Lin, "Algorithms of finding the first two minimum values and their hardware implementation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 11, pp. 3430–3437, 2008.

[14] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. on Inf. Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[15] Z. Wang and Z. Cui, "A memory efficient partially parallel decoder architecture for quasi-cyclic LDPC codes," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 4, pp. 483–488, 2007.