A Survey on High-Throughput Non-Binary LDPC Decoders: ASIC, FPGA, and GPU Architectures

Oscar Ferraz, *Student Member, IEEE*, Srinivasan Subramaniyan, Ramesh Chinthala, João Andrade, Joseph R. Cavallaro, *Fellow, IEEE*, Soumitra K. Nandy, *Senior Member, IEEE*, Vitor Silva, Xinmiao Zhang, *Senior Member, IEEE*, Madhura Purnaprajna, and Gabriel Falcao, *Senior Member, IEEE*

***Abstract*—Non-binary low-density parity-check (NB-LDPC) codes show higher error-correcting performance than binary low-density parity-check (LDPC) codes when the codeword length is moderate and/or the channel has bursts of errors. The need for high-speed decoders for future digital communications led to the investigation of optimized NB-LDPC decoding algorithms and efficient implementations that target high throughput and low energy consumption levels. We carried out a comprehensive survey of existing NB-LDPC decoding hardware that targets the optimization of these parameters. Even though existing NB-LDPC decoders are optimized with respect to computational complexity and memory requirements, they still lag behind their binary counterparts in terms of throughput, power and area optimization. This study contributes to an overall understanding of the state-of-the-art on application-specific integrated-circuit (ASIC), field-programmable gate array (FPGA) and graphics processing units (GPU) based systems, and highlights the current challenges that still have to be overcome on the path to more efficient NB-LDPC decoder architectures.**

***Index Terms*—Non-binary low-density parity-check codes, error-correcting codes, resilient communications, non-binary LDPC decoders, ASIC, FPGA, GPU.**

Manuscript received October 22, 2020; revised April 24, 2021 and August 14, 2021; accepted October 11, 2021. Date of publication November 8, 2021; date of current version February 24, 2022. This work was supported in part by the Project ECHO, a joint work supported through the Indo-Portugal Bilateral Scientific and Technological Cooperation funded by Instituto de Telecomunicações and Fundação para a Ciência e Tecnologia in Portugal under Grant UIDB/EEA/50008/2020 and Grant PTDC/EEI-HAC/30485/2017 and the Ph.D. Scholarship 2020.07124.BD, and in part by the Department of Science and Technology, Government of India, under Grant INT/PORTUGAL/P-12/2017. The work of Joseph R. Cavallaro was supported in part by the U.S. NSF under Grant CNS-1717218, Grant CNS-2016727, and Grant CNS-1827940, for the "PAWR Platform POWDER-RENEW: A Platform for Open Wireless Data-driven Experimental Research with Massive MIMO Capabilities." The work of Xinmiao Zhang was supported by the National Science Foundation under Award 2052641. *(Corresponding author: Oscar Ferraz.)*

Oscar Ferraz, Vitor Silva, and Gabriel Falcao are with the Department of Electrical and Computer Engineering, Instituto de Telecomunicações, University of Coimbra, 3030-290 Coimbra, Portugal (e-mail: oscar.ferraz@co.it.pt; vitor@co.it.pt; gff@co.it.pt).

Srinivasan Subramaniyan and Madhura Purnaprajna are with the Department of Computer Science, Amrita Vishwa Vidyapeetham, Bengaluru 560035, India (e-mail: srinivasansubramaniam74@gmail.com; p_madhura@blr.amrita.edu).

Ramesh Chinthala is with the Department of Electronics and Communication Engineering, School of Engineering, Amrita Vishwa Vidyapeetham, Bengaluru 560035, India (e-mail: c_ramesh@blr.amrita.edu).

João Andrade is with the Solutions Group, Synopsys Portugal, 4470-605 Moreira da Maia, Portugal (e-mail: joao.andrade@synopsys.com).

Joseph R. Cavallaro is with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: cavallar@rice.edu).

Soumitra K. Nandy is with the Department of Computational and Data Sciences, Indian Institute of Science Bangalore, Bengaluru 560012, India (e-mail: nandy@iisc.ac.in).

Xinmiao Zhang is with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: zhang.8952@osu.edu).

Digital Object Identifier 10.1109/COMST.2021.3126127

I. Introduction

ERROR correcting codes (ECCs) are an important component employed in modern communication systems. They minimize the impact of errors in data over a transmission channel. Error correcting codes (ECCs) can be classified into two main branches, algebraic and probabilistic. The algebraic branch focuses on exploiting finite-field arithmetic to maximize the minimum hamming distance between codewords. However, the designs from this branch are not suitable for achieving high error-correcting capability [1]. The probabilistic branch is capable of relatively good throughput performance while keeping a moderate design complexity. The main advantage of the latter is the strong error-correcting capability, which allows a lower bit error rate (BER) to be achieved [1]. One class of ECCs from this branch is the low-density parity-check (LDPC) codes. First proposed by R. Gallager in the 1960s [2], LDPC codes have been shown to approach the Shannon limit [3]. These were considered impractical for many decades, in real-time communication systems, due to the computational complexity involved.

Fig. 1 represents a structure of an ECC system. Data transmitted through a channel is prone to noise that causes errors in the received data. Several ECC algorithms can be employed to correct those errors. The channel encoder transmits data with N bits. In order to achieve resilience to noise, $N - K$ redundant bits are added to the payload or information bits (K). The ratio between N and K or code rate ($R = \frac{K}{N}$), with $N > K$, can be controlled to add resilience to noise, where a low code rate is used in high noise channels.

Several ECCs have been proposed along with the design and development of the successive generations of mobile telecommunications. At present, the most relevant are Turbo code [4], binary LDPC codes and Polar codes [5]. However, the literature lacks an organized structure of reported works regarding Non-binary low-density parity-check (NB-LDPC) codes and their implementations, which are capable of achieving better BER performance for moderate code lengths [6].

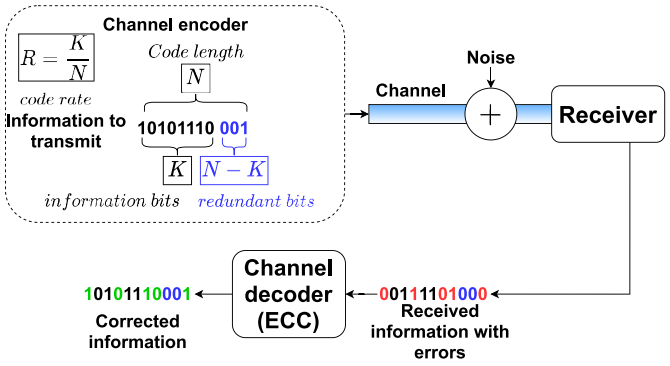


Fig. 1. Representation of an ECC system. $N - K$ redundant bits are added to the payload or information bits (K) and the codeword of length N is sent through a channel. The transmitted information is prone to noise which causes errors in the received data. ECC systems repair those errors and are the backbone of reliable data transmission systems.

On the one hand, the code characteristics can be altered since the code length and the number of iterations can be increased, putting pressure on the hardware. On the other hand, the hardware has limited resources and the decoder implementation can only perform to a limited degree in terms of throughput, latency, flexibility and memory. The three main performance goals consist of achieving (1) high throughput and (2) low energy consumption, while guaranteeing (3) performance scalability to higher Galois fields for reaching even lower BER, and keeping the computational complexity manageable for supporting real-time processing, all limited by realistic latency, area and power constraints.

A. Motivation

The formulation of NB-LDPC decoding algorithms has been reported in the literature for twenty years. In the past decade, more algorithms that take architecture constraints into consideration have been proposed, while very large scale integration (VLSI) architectures have reached maturity for processing such complex codes. The literature reports dozens of decoders but lacks comprehensive studies about the relationship between NB-LDPC coding theory and the computer architectures to process them efficiently.

This paper focuses on algorithms and NB-LDPC decoder designs that target the graphics processing units (GPU), field-programmable gate array (FPGA) and application-specific integrated-circuit (ASIC)-based architectures. While the former provides high energy efficiency due to the high number of cores (i.e., data-parallelism) and memory bandwidth available, they also require power two orders of magnitude above the ones provided by the latter. FPGAs demand low-power (ASICs even lower) and provide high-throughput (ASICs even higher) performance, as demonstrated in the last sections of the article. They provide a high degree of customization and can lead to superior energy efficiency. However, both present constraints in the form of coding and development effort, parallelization complexity and cost, compared with GPUs.

TABLE I
LIST OF ACRONYMS AND ABBREVIATIONS USED IN THIS PAPER

10GBASE-T	10 Gigabit Ethernet
ADBP	Analog Digital Belief Propagation
AMSA	Adaptive Multiset Stochastic Algorithm
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BP	Belief Propagation
CCSDS	Consultative Committee for Space Data Systems
CN	Check Node
CN-to-VN	Check-to-Variable Node
CUDA	Compute Unified Device Architecture
DVB-S2	Digital Video Broadcasting - Satellite - second generation
ECC	Error-Correcting Code
EDA	Electronic Design Automation
ELB	Equivalent Logic Block
eMBB	enhanced Mobile Broadband
EMS	Extended Min-Sum
FBRB	Full Bit Reliability-Based
FER	Frame Error Rate
FF	Flip-Flop
FFT	Fast Fourier Transform
FHT	Fast Hadamard Transform
FPGA	Field-Programmable Gate Array
GBFDA	Generalized Bit-Flipping Decoding Algorithm
GF	Galois Field
GPU	Graphics Processing Unit
HLS	High-Level Synthesis
IHRB	Iterative Hard Reliability-Based
ISRB	Iterative Soft Reliability-Based
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood Ratio
LUT	Lookup Table
LTE	Long Term Evolution
MIMO	Multiple-Input Multiple-Output
MLGD	Majority-Logic Decoding
MM	Min-Max
NASA	National Aeronautics and Space Administration
NB-LDPC	Non-Binary Low-Density Parity-Check
PCM	Parity-Check Matrix
PLC	Power-Line Communication
QAM	Quadrature Amplitude Modulation
RS	Reed-Solomon
RTL	Register Transfer Level
SMSA	Simplified Min-Sum Algorithm
SNR	Signal-to-Noise Ratio
SPA	Sum-Product Algorithm
SRB	Symbol Reliability-Based
TDP	Thermal Design Power
UMTS	Universal Mobile Telecommunications System
VLSI	Very Large Scale Integration
VN	Variable Node
VN-to-CN	Variable-to-Check Node
WBRB	Weighted Bit Reliability-Based
WiMAX	Worldwide interoperability for Microwave Access

B. Contributions

This survey gives a comprehensive description of popular NB-LDPC decoding algorithms and provides a study comparing advantages and disadvantages between GPUs, FPGAs and ASICs for various NB-LDPC decoder implementations. Quite a few surveys have been conducted on several ECCs schemes and/or their implementation architectures, as shown in Table II. However, those surveys only briefly describe a few NB-LDPC decoding algorithms without covering their implementations

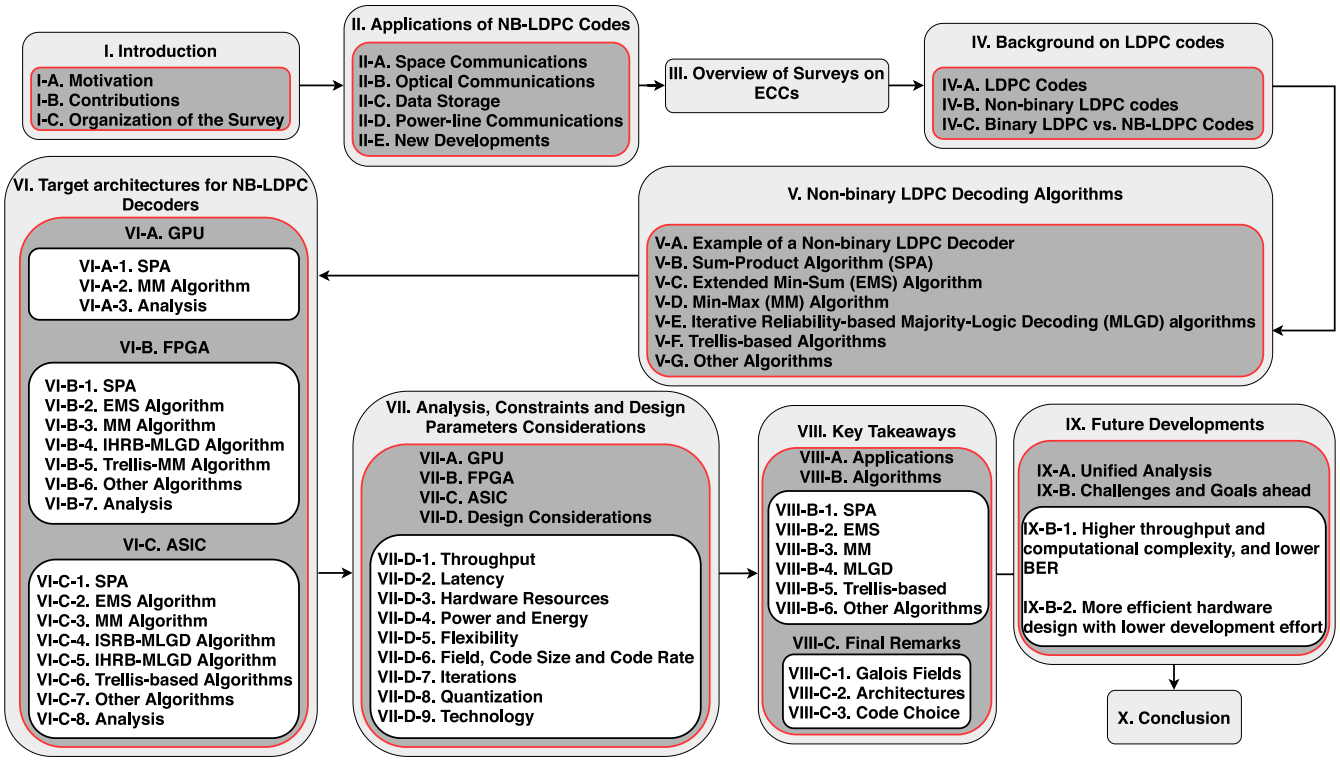


Fig. 2. Structure of the survey.

or only report a small fraction of NB-LDPC decoder implementations available in the literature. Our survey is the only one that (1) describes NB-LDPC decoding algorithms found in the literature, (2) compares more than 200 NB-LDPC decoders in three different architectures (GPUs, FPGAs and ASICs) and (3) provides a good basis for new researchers to understand, choose the algorithm and the architecture for developing NB-LDPC decoders for various applications.

Furthermore, this work seeks to achieve the following **contributions**:

- explain the differences between binary LDPC and NB-LDPC codes, discussing the qualities and disadvantages of each;
- provide examples of applications that require NB-LDPC codes;
- analyze the most relevant NB-LDPC decoding algorithms;
- compare implementations on GPUs, FPGAs and ASICs for the presented algorithms;
- discuss the trade-offs between the algorithmic characteristics and the chosen architecture;
- recommend the best design guidelines for implementing highly efficient NB-LDPC decoders;
- evaluate future trends of NB-LDPC decoders.

C. Organization of the Survey

Fig. 2 shows the structure of this survey. Section I provides a brief introduction to ECC and describes the motivation and contributions of this survey. Section II reports cases of NB-LDPC applications. Section III reports surveys on ECCs, in particular, Polar codes, Turbo codes, binary and non-binary LDPC codes. Section IV gives the background on LDPC codes, discussing the details of binary (Section IV-A) and non-binary LDPC codes (Section IV-B) and discusses their relative advantages and disadvantages (Section IV-C). Section V provides a formal description of NB-LDPC decoding algorithms: Section V-A provides a numerical example of an NB-LDPC decoder, serving as a tutorial, while the following subsections describe various NB-LDPC decoding algorithms. Section VI presents the decoder implementations found in the literature. Section VII provides a standardized comparison between decoders implemented in the same platform and discusses the constraints and advantages when choosing a device or developing an architecture to implement an NB-LDPC decoder. Section VIII presents the main conclusions of this survey by comparing and discussing the relationship between decoding algorithms, architectures and applications. Section IX provides a uniform analysis by comparing the number of used transistors for every implementation (Section IX-A) and analyzes future research challenges (Section IX-B), and finally, Section X offers our conclusions from this survey. Table I is provided to improve the readability of this paper.

II. Applications of NB-LDPC Codes

This section presents examples of NB-LDPC applications and discusses the requirements for each one. NB-LDPC codes offer better error-correcting performance than their binary counterparts, particularly for short to moderate lengths, at the cost of increased decoder complexity [7], [8]. However, high order modulations do not require binary-to-NB

mapping/demapping operations, thus proposing a good solution for high spectral efficiency [9].

A. Space Communications

NB-LDPC codes are particularly useful for reducing BER in deep space downlink communications and interactive uplink satellite communications, which use moderate data rates and short to moderate packet lengths [10]. In [11], the authors simulated an Additive White Gaussian Noise (AWGN) satellite communication channel to transmit information using an NB-LDPC code encapsulated in a Fountain encoder/encoder scheme, achieving a BER of 10^{-6} at a signal-to-noise ratio (SNR) of 1.85 *dB*. Furthermore, the use of NB-LDPC codes has been proposed by the German Aerospace Center [12] and National Aeronautics and Space Administration (NASA) [13].

Satellites communicate through free space channels which cause signal attenuation. Atmospheric absorption, weather conditions, pollution, electromagnetic and other interference contribute to increase BER and thus decrease throughput performance on free-space communications systems [14].

B. Optical Communications

Another class of applications suitable for massive adoption of NB-LDPC codes is optical transport networks, particularly those used over long transmission distances [35]–[39]. Unlike satellite communications, terrestrial optical networks can provide less noisy channels, allowing for higher transmission rates (Tbps) at lower BERs ($< 10^{-15}$). Currently, 100Gb/s terrestrial optical transmission schemes are based on concatenated codes, Turbo and LDPC codes [40]. However, for the next generation of terrestrial optical communications, BER in the order of 10^{-15} are required, a performance that NB-LDPC decoders can deliver [40]. For example, in [37], a simulation for 100Gb/s optical transport systems achieved a BER of 10^{-15} at an SNR of 10.8 *dB*. Furthermore, by exploring all available electrical and optical degrees of freedom and applying NB-LDPC codes in a spectral-spatial-multiple-input multiple-output (MIMO) scheme, it is possible to achieve a 100 Tb/s serial optical transport network with a BER of 10^{-15} at an SNR of 12.3 *dB* [41].

C. Data Storage

NB-LDPC codes can also be employed in data storage. Unlike wireless or wired communications, data storage applications require an extremely low Frame Error Rate (FER) since a single fault results in irreversible data loss [8].

D. Power-Line Communication

Also, in highly noisy Power-Line Communication (PLC) channels, NB-LDPC codes can be employed to overcome attenuation, impulsive noise and multipath frequency selectivity and can be used to reliably transmit voice data and media signals, thereby removing pressure from communication-only infrastructure [42], [43].

E. New Developments

These are the most recent themes of research in NB-LDPC codes. However, new applications have been emerging such as the Internet-of-Things, autonomous vehicles [44], that require vehicle-to-vehicle communications, and even wearable computing devices, where NB-LDPC codes can be exploited. The evolution of communication systems will increasingly incorporate more elements of optical communications, providing higher transmission rates at lower BER and reduced latency while maintaining low energy consumption levels [27], [40].

The range of applications that can employ ECC techniques, such as NB-LDPC codes, poses additional challenges on top of the already hard problem of developing efficient hardware designs [7]. A number of system parameters that have an impact on the operating frequency, latency, area, power and throughput of the decoder can be controlled and manipulated. There are several that can improve the error correction capability of the decoder.

III. Overview of Surveys on ECCs

A review of ECC surveys found in the literature is presented in this section. Surveys found in the literature provide works in several ECCs. Table II summarizes the surveys related to Turbo, polar and LDPC codes. Turbo codes are surveyed in [15], [20], [26] with Brejza *et al.* [25] analyzing implementations in ASICs. Polar codes are surveyed in [1], [31], [32], [34] but none of these papers provide decoder implementations.

Some surveys report binary LDPC decoders in ASICs [17], [18] and FPGAs [24]. Moreover, Andrade *et al.* [23] report decoder implementations in both GPUs and FPGAs, while Guilloud *et al.* [16] and Thameur *et al.* [28] report implementations in both FPGAs and ASICs. Only the binary LDPC decoding are reported in [19], [21], without implementations.

From the surveyed literature, only three works provide surveys across different ECCs. Polar and LDPC codes are studied in [22] but do not present decoder implementations. In [27], the author presents a tutorial using LDPC and Turbo codes applied in optical networks. Finally, [7] provides a comprehensive survey on ASIC implementations of binary LDPC, Turbo and polar codes.

Arikan *et al.* [22] compare Polar codes, binary and non-binary LDPC codes. However, the authors only dedicate one section to discuss advantages and disadvantages between binary and non-binary LDPC codes, briefly describing four NB-LDPC decoding algorithms and focuses on the analog digital belief propagation (ADBP) algorithm. This survey does not provide a detailed description of the algorithms and only mentions a small fraction of the implementations found in the literature. Djordjevic proposes a tutorial in [27] on forward error correction and coded modulation for optical communications. For NB-LDPC decoders, the author presents one subsection describing decoding algorithms and cites one paper containing a dozen FPGA implementations. In [29], the authors focus on the construction and techniques of binary and non-binary LDPC codes applied to

TABLE II
COMPARISON OF THE SURVEYS FOUND IN THE LITERATURE ON TURBO, POLAR AND LDPC CODES

Year	Work	Binary LDPC	Non-binary LDPC	Turbo Codes	Polar Codes	Decoding algorithms	Implementations		
							GPU	FPGA	ASIC
2000	[15]	×	×	✓	×	✓	×	×	×
2007	[16]	✓	×	×	×	✓	×	✓	✓
2007	[17]	✓	×	×	×	✓	×	×	✓
2011	[18]	✓	×	×	×	✓	×	×	✓
2011	[19]	✓	×	×	×	✓	×	×	×
2013	[20]	×	×	✓	×	✓	×	×	×
2015	[21]	✓	×	×	×	✓	×	×	×
2015	[22]	✓	✓	×	✓	×	×	×	×
2016	[23]	✓	×	×	×	✓	✓	✓	×
2016	[24]	✓	×	×	×	✓	×	✓	×
2016	[25]	×	×	✓	×	✓	×	×	✓
2016	[26]	×	×	✓	×	✓	×	×	×
2016	[27]	✓	✓	✓	×	✓	×	✓	×
2017	[28]	✓	×	×	×	✓	×	✓	✓
2018	[29]	✓	✓	×	×	×	×	×	×
2018	[30]	✓	✓	×	×	✓	×	×	×
2019	[7]	✓	×	✓	✓	×	×	×	✓
2019	[31]	×	×	×	✓	✓	×	×	×
2019	[32]	×	×	×	✓	✓	×	×	×
2020	[1]	×	×	×	✓	✓	×	×	×
2020	[33]	×	×	×	✓	✓	×	×	×
2020	[34]	×	×	×	✓	✓	×	×	×
2020	**Our survey**	×	✓	×	×	✓	✓	✓	✓

magnetic record systems without describing algorithms and implementations. The work from [30] is the most recent survey NB-LDPC codes. The author surveys methods and decoding algorithms for both binary and non-binary LDPC codes. For NB-LDPC codes, they describe four decoding algorithms. However, the survey only mentions 11 decoder implementations and does not provide a qualitative comparative analysis.

The current paper surveys a variety of implementations of NB-LDPC decoding algorithms and decoder implementations on GPUs, FPGAs and ASICs, being the only work to survey implementations on three distinct systems. This work also discusses the best approaches for different algorithms and provides insightful information regarding NB-LDPC decoding and architectural design.

In the literature, numerous implementations of Turbo, polar and LDPC decoders have been proposed. From the surveyed literature, Turbo codes achieve the lowest throughput performance of 15.8 Gbps [45]. Data dependencies prevent parallel implementations from improving throughput performance [7], even with a higher degree of parallelization than polar decoders, which also experience data dependencies. However, polar decoders can achieve a peak throughput performance of 25.6 Gbps by unrolling and pipelining large amounts of data blocks [46]. The parallelization degree is higher for LDPC decoders, which allows throughput performance to increase up to 172.4 Gbps for binary decoders [47] and 21.6 Gbps for non-binary decoders [48].

IV. BACKGROUND ON LDPC CODES

This section provides a background on LDPC codes. Section IV-A introduces the historical context and a small example is given of a binary LDPC code. Section IV-B details the evolution from binary to NB-LDPC codes and the analogous example from the previous section is given in the non-binary form. Section IV-C compares the advantages and disadvantages between binary and non-binary LDPC codes.

A. LDPC Codes

LDPC codes were invented by Gallager in 1962 [2] and allow transmission rates close to the Shannon limit [49]. Due to the computational complexity, these codes were deemed impractical and were forgotten by the scientific community. Fueled by the invention of turbo codes in 1993 [4], MacKay and Neal rediscovered the LDPC codes after 34 years, when processing systems could process such computational complexity.

An LDPC code is a linear block code defined by a sparse parity-check matrix (PCM) $\mathbf{H}$ or the equivalent Tanner graph representation [50]. These types of codes contain K information bits on an N-bit codeword. The redundant bits ($M = N - K$, with $K < N$) can be added to provide immunity against noise, thus increasing the robustness of the codeword [51], as depicted in Fig. 1. check nodes (CNs) and variable nodes (VNs) are some of the main components of LDPC codes, where some data processing occurs (detailed in Section V-A). The number of CNs and VNs is defined by the

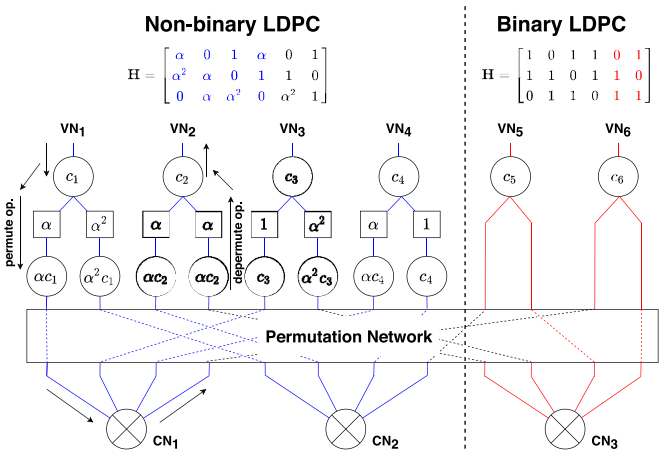


Fig. 3. LDPC Tanner graph representation for the matrix presented in (1) (right side) and (2) (left side). The number of VNs (*N*) is equal to the number of columns of the **H** matrix, while the number of CNs (*M*) equals the number of rows. The permutation network is defined by the non-zero entries of the **H** matrix and occurs on both binary and non-binary cases. Data is sent through connected nodes and processed by VNs and CNs. For NB-LDPC codes, the data from VNs is multiplied by the corresponding symbol when transmitted to the CN and divided by the symbol when transmitted from the CN to the VN. The permutation/depermutation operations on the upper left part of the figure are not required in the binary case.

number of rows and columns of the PCM. A binary example of a PCM **H** with $M = 3$ rows, corresponding to the number of CNs and $N = 6$ columns, corresponding to the number of VNs is illustrated in (1), below. The code rate is the number of information bits per transmitted bits [52]. In this example, the code rate is equal to $\frac{K}{N} = 0.5$.

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

The PCM in (1) indicates the connections between CNs and VNs, which can be illustrated by the Tanner graph in Fig. 3. The LDPC code is considered regular if the number of non-zero elements is equal in all rows (d_c) and the number of non-zero elements is equal in all columns (d_v). If the CN degree d_c and the VN degree d_v are not constant, the code is irregular. Irregular codes have better coding performance but are computationally more complex [53], [54].

A transmission system that uses an LDPC code encodes data to transmit (**v**), of size *K*, by multiplying the data by generator matrix (**G**), outputting a codeword ($\mathbf{c} = \mathbf{v} \cdot \mathbf{G}$). In the receiver, the received codeword is multiplied by the PCM in an operation named parity check equation ($\mathbf{c} \cdot \mathbf{H}^T = 0$). However, the codeword can be corrupted with noise, resulting in the parity check equation to fail (not equal to zero). In this case, a decoding algorithm must be employed to correct errors (further details can be found in Section V-A).

Several methods can be used to construct both **G** and **H** matrices. Gallager codes [2] and Mackay codes [3] (also known as progressive edge growth) were the first to be used and result in a semi-random construction. Quasi-cyclic LDPC code construction methods are some of the most used in the literature due to their decreased complexity, which results in a simpler decoder [55], [56]. In quasi-cyclic LDPC codes, the construction method uses shifted identity sub-matrices and zero sub-matrices to create the PCM matrix through multiplicative or additive groups of finite fields or even through masking methods [56], [57].

TABLE III
ARITHMETIC RULES FOR ADDING AND MULTIPLYING b_1 AND b_2 $GF(2^2)$

b_1 \ b_2	**0**	**1**	$\boldsymbol{\alpha}$	$\boldsymbol{\alpha^2}$
0	0	1	α	α^2
1	1	0	α^2	α
$\boldsymbol{\alpha}$	α	α^2	0	1
$\boldsymbol{\alpha^2}$	α^2	α	1	0

(a) Addition in $GF(2^2)$.

b_1 \ b_2	**0**	**1**	$\boldsymbol{\alpha}$	$\boldsymbol{\alpha^2}$
0	0	0	0	0
1	0	1	α	α^2
$\boldsymbol{\alpha}$	0	α	α^2	1
$\boldsymbol{\alpha^2}$	0	α^2	1	α

(b) Multiplication in $GF(2^2)$.

B. Non-Binary LDPC Codes

Fueled by the invention of Turbo codes [4] in the 1990s, Davey and MacKay revisited LDPC codes and proposed an extension for a non-binary formulation [6]. By then, with the evolution of Moore's Law, new processing systems could create enough computing power to execute moderate-length versions of these codes efficiently.

Davey and MacKay an extension of binary LDPC codes over the Galois field *GF*(q) with $q > 2$ [6]. Barnault and Declercq proposed a modification to NB-LDPC codes that allowed the definition of higher orders with a complexity that scaled as $q \times log_2(q)$ [58]. The new proposal allowed codes with order $GF(2^m)$ to be decoded up to a maximum of $GF(2^8)$.

A Galois field is defined as a mathematical field with a finite number of elements and is denoted as *GF*(q), where q is the cardinality of the field [59]. Galois fields obey a set of proprieties, which dictates that any result of a Galois field operation must result in a number contained in the field (see Table III). An irreducible primitive polynomial can be used to define an *m*-order Galois field. Therefore, α^m is used to represent symbols in NB-LDPC codes [59] (further details in Section V-A).

Extending (1) to $GF(2^2)$, (2) represents a possible NB-LDPC code with the same connections as (1). Contrary to binary LDPC codes, where the symbols are represented either by a '0' or '1', NB-LDPC codes are represented by symbols contained in the respective Galois field. For instance, for codes defined over $GF(2^2)$, each symbol is represented by 2 bits or 0, 1, α and α^2 using a irreducible polynomial [59]. For those defined over $GF(2^4)$, 4 bits are required to define a symbol.

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

The main difference from binary LDPC codes is the permutation and depermutation operators, as depicted on the upper left side of Fig. 3. In an arbitrary connection between a VN and a CN, the probability is multiplied by the Galois symbol corresponding to the PCM entry of the n-*th* VN (n-*th* PCM

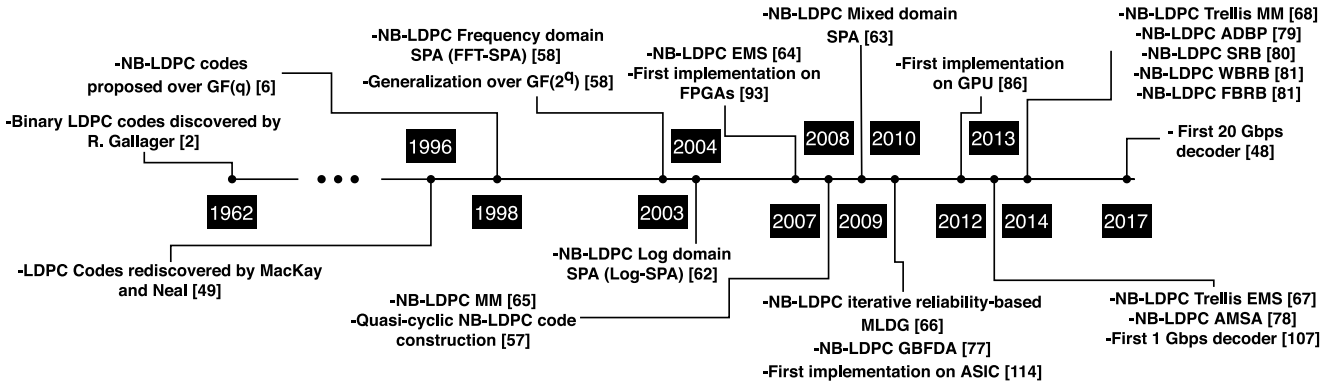


Fig. 4. Non-binary LDPC codes and decoders evolution timeline.

column) and the m-*th* CN (m-*th* PCM row). When the data is propagated from the CN to the VN, the same process happens but instead, the probability is divided by the corresponding symbol.

C. Binary LDPC vs. NB-LDPC Codes

NB-LDPC codes can achieve better error-correcting performance than binary LDPC codes when the code length is moderate, but this is at the cost of higher decoding complexity [60]. In [6], by moving an irregular $\frac{1}{3}$ rate code from binary to a *GF*(8) construction, it is possible to achieve a 0.3 dB improvement. In a MIMO system, NB-LDPC codes over a small Galois field (up to *GF*(16)) outperform certain binary LDPC codes, both employing joint MIMO detection and channel decoding. At BER of 10^{-4}, for 16 quadrature amplitude modulation (QAM), [61] shows that a separate detection and decoding MIMO system employing an NB-LDPC code over *GF*(256) outperforms the joint detection and decoding system of [6], by 0.37 *dB*.

NB-LDPC decoders show a high computational complexity in CN processing and require a high volume of memory to store the intermediate results. There are various methods in the literature where researchers tried to reduce both computational complexity and memory requirement.

V. Non-Binary LDPC Decoding Algorithms

The present section describes NB-LDPC decoding algorithms. An example of an NB-LDPC decoding algorithm, serving as a tutorial, is given in Section V-A. The following sections present the NB-LDPC decoding algorithms found in the literature based on two criteria: (1) literature relevance, based on the popularity and impact on the evolution of NB-LDPC decoders, and (2) timeline. The algorithms in Sections V-B, V-C, V-D, V-E, and V-F are presented chronologically. In contrast, in Section V-G, each paragraph describes one algorithm and it is also presented chronologically. One exception to this criteria is the sum-product algorithm (SPA) which contains three variants (fast Fourier transform (FFT)-SPA, Log domain SPA and mixed domain SPA). For the sake of simplicity, they are grouped in the same category (SPA) but in Tables IV, V and VI, they are presented in different groups for better analysis.

Fig. 4 provides an evolution timeline of NB-LDPC codes and decoding algorithms; SPA (also known as belief propagation (BP) algorithm) was the first to be proposed in 2003 [58]. In this proposal, the decoder's calculations are executed in the frequency domain (FFT-SPA). However, some variants have been proposed. In 2004, a decoder was proposed on the Log domain [62] and afterward, in 2009, a mixed domain (both frequency and Log domain) decoder was proposed [63] (further details in Section V-B). In 2007, the extended min-sum (EMS) algorithm was introduced [64], reducing the computational complexity of the CN processor in SPA.

Then, in 2008, the min-max (MM) algorithm further increased performance by replacing the sum operation on the EMS algorithm CN processor with the *max*() operation [65].

In 2010, two methods were proposed in [66] that belong to the majority-logic decoding (MLGD) class. The iterative hard reliability-based (IHRB) and the iterative soft reliability-based (ISRB) methods reduce memory utilization.

In 2013 and 2014, Trellis-EMS [67] and Trellis-MM [68] proposed a configuration set path by only considering the most reliable symbols and further reducing the decoder's complexity. Apart from the intense arithmetic operations, the complexity lies in the permutation network that connects CNs and VNs, order of the Galois field and memory requirements.

A. Example of a Non-Binary LDPC Decoder

LDPC codes are defined by a sparse PCM $\mathbf{H}$. The decoding process works by satisfying the parity check equation ($\mathbf{H}^T \cdot \mathbf{c} = 0$), where $\mathbf{c}$ is the received codeword. In binary codes, $\mathbf{H}$ and $\mathbf{c} \in GF(2) = \{0, 1\}$. However, NB-LDPC codes use finite fields in $GF(2^m)$ ($m \in \mathbb{Z}^+$), which requires arithmetical rules. All the elements of $GF(2^m)$ can be expressed as $0, 1, \alpha, \alpha^2, \ldots, \alpha^{m-2}$, where α is a primitive element of $GF(2^m)$. An irreducible primitive polynomial defines these fields with degree *m* for each $GF(2^m)$, which is used to derive addition and multiplication for two elements $\in$ $GF(2^m)$. For example, Table III depicts the addition and multiplication

tables generated by the primitive polynomial $\alpha^2 + x^1 + 1$ in $GF(2^2)$ [52].

Decoders use XOR operators to implement additions, while multiplication operators are efficiently implemented using look-up table (LUT) [69].

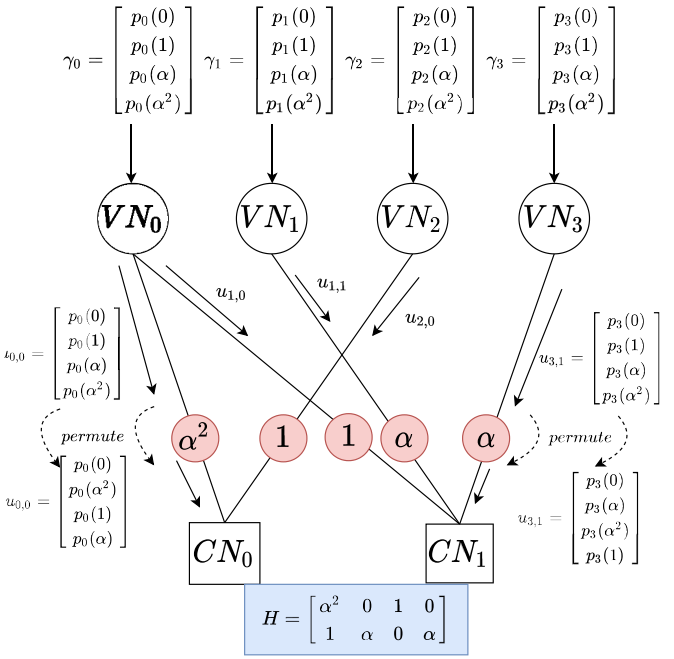
Data transmission works by transmitting a codeword $\mathbf{c} = [c_0, \ldots, c_n]$ of n symbols over a noisy channel. The received message $\mathbf{y} = [y_0, \ldots, y_n]$ may contain errors that need to be corrected using an iterative decoding process. On binary LDPC decoding, the process calculates the probability of a received symbol being 0 or 1. On NB-LDPC decoding, the probabilities are computed for every element in $GF(2^m) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{m-2}\}$.

The first step is to initialize the probabilities $p_n = P(c_n = \beta | y_n)$, which are the probability of a symbol (β) belonging to an element of the codeword (c_n), knowing the received codeword element (y_n). For older algorithms, such as the BP algorithm (also known as SPA), the probabilities are represented by a probability mass function ($p_n(0) + p_n(1) + \cdots + p_n(\alpha^{m-2}) = 1$). However, newer algorithms exchange and compute log-likelihood ratio (LLR) as $ln\frac{P(c_n=\widehat{\beta})}{P(c_n=\beta_n)}$ where $\widehat{\beta}$ is the most likely $GF(2^m)$ element for the n-th codeword symbol. Instead of using probability mass functions, the conversion to LLRs allows to convert multiplications into additions and reduce quantization errors [62].

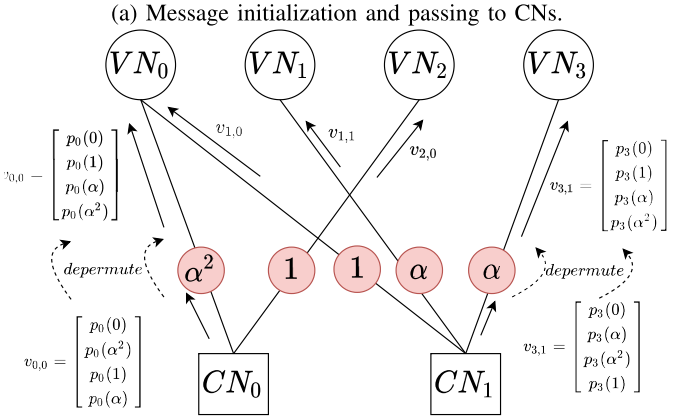
The decoder structure can be described as Tanner graph, as depicted in Fig. 5. In this figure, the example NB-LDPC decoder contains four VNs and two CNs equal to the number of columns (n) and rows (m) of $\mathbf{H}$. Each VN is connected to the CNs defined by the non-zero entries in $\mathbf{H}$. In this example, VN_0 connects to CN_0 and CN_1 because $\mathbf{H_{0,0}}$ and $\mathbf{H_{1,0}}$ are non-zero entries. However, VN_1 does not connect to CN_0 because $\mathbf{H_{0,1}} = 0$. These connections serve the purpose of transmitting probabilities vectors. Those are initialized with $\gamma_\mathbf{n} = [p_n(0), \ldots, p_n(\alpha^{m-2})]$ and transmitted to connected CNs in the form of the message vector ($\mathbf{u_{m,n}} = \gamma_\mathbf{n}$). These messages are multiplied by the corresponding Galois symbol on $\mathbf{H_{m,n}}$ in an operation named permutation, to be processed in the CNs.

Then, CN processing is executed. The probabilities are recalculated for each symbol on each CN. Each algorithm applies different operations to the messages (addition, multiplication, minimum, maximum), defined in the following sections. The general rule determines that the new probability vector to be sent to a connected VN ($\mathbf{v_{m,n}}$) is calculated through the received messages ($\mathbf{u_{m,n}}$) from adjacent connections on the same CN. For example, the CN_1 in Fig. 5(a) receives messages $\mathbf{u_{1,0}}$, $\mathbf{u_{1,1}}$ and $\mathbf{u_{3,1}}$. To calculate the message $\mathbf{v_{1,0}}$, the CN processor uses $\mathbf{u_{1,1}}$ and $\mathbf{u_{3,1}}$. For message $\mathbf{v_{1,1}}$, it will use $\mathbf{u_{1,0}}$ and $\mathbf{u_{3,1}}$ and finally for message $\mathbf{v_{3,1}}$, it will use $\mathbf{u_{1,0}}$ and $\mathbf{u_{1,1}}$. In CN_0, the calculation of message $\mathbf{v_{2,0}}$ only uses the $\mathbf{u_{0,0}}$, while the calculation of message $\mathbf{v_{0,0}}$ will use message $\mathbf{u_{2,0}}$.

After calculating all check-to-variable (CN-to-VN) messages, the next step depermutes the vector messages and sends them to the connected VNs as shown in Fig. 5(b). VN processing differs in each algorithm. Similar to CN processing,



(a) Message initialization and passing to CNs.



(b) Message passing to VNs.

Fig. 5. NB-LDPC message exchange and permutations.

the new probability vectors ($\mathbf{u_{m,n}}$) are calculated through the received messages ($\mathbf{v_{m,n}}$) from adjacent connections on the same VN. The main difference from CN processing is that new messages $u_{m,n}$ are added to or multiplied (depending on the algorithm) by the initial probability vector $\gamma_\mathbf{n}$, generating a $\mathbf{z}$ vector with the most likely symbols.

For the decoding to end, the parity check equations must be verified ($\mathbf{z} \cdot \mathbf{H^T} = 0$). If this condition is true, the process stops and the codeword is successfully corrected. If the premise is false, the messages $u_{m,n}$ are sent to CNs for the probabilities to be recalculated and the process repeats until the parity equation or a maximum number of iterations (I_{max}) is verified. This I_{max} variable is set to limit the maximum latency but should be large enough to allow the decoding process to converge, and is chosen to achieve a given FER goal. The decoding process checks the parity equation before starting the decoding process. If the received codeword γ_n contains no errors, the decoding process is not executed.

Algorithm 1 Belief Propagation (BP) for NB-LDPC Decoding

input: γ_n, $\mathbf{H}$; I_{max}

initialization: $u_{m,n}^{(0)}(\beta) = \gamma_n(\beta)$; $z_n^{(0)} = \arg\max_\beta(\gamma_n(\beta))$

for $k = 1 : I_{max}$

Compute $z^{(k-1)}\mathbf{H}^T$; Stop if $z^{(k-1)}\mathbf{H}^T = 0$

check node processing

for each check node *m*, each $n \in S_v(m)$, each $\beta \in GF(q)$

$$v_{m,n}^{(k)}(\beta) = \sum_{(a_j)\in\mathcal{L}(m|a_n=\beta)} \left(\prod_{j\in S_v(m)\backslash n} u_{m,j}^{(k-1)}(a_j) \right) \quad (3)$$

variable node processing

for each variable node *n*, each $m \in S_c(n)$, each $\beta \in GF(q)$

$$u_{m,n}^{(k)}(\beta) = \gamma_n(\beta) \prod_{i\in S_c(n)\backslash m} v_{i,n}^{(k)}(\beta)$$

a posteriori information computation & tentative decision

for each variable node *n*, each $\beta \in GF(q)$

$$z_n^{(k)} = \arg\max_\beta(\gamma_n(\beta) \prod_{i\in S_c(n)} v_{i,n}^{(k)}(\beta))$$

Algorithm 2 Extended Min-Sum (EMS) Decoding Algorithm

input: γ_n; $\mathbf{H}$; I_{max}

initialization: $u_{m,n}^{(0)}(\beta) = \gamma_n(\beta)$; $z_n^{(0)} = \arg\min_\beta(\gamma_n(\beta))$

for $k = 1 : I_{max}$

Compute $z^{(k-1)}\mathbf{H}^T$; Stop if $z^{(k-1)}\mathbf{H}^T = 0$

check node processing

for each check node *m*, each $n \in S_v(m)$, each $\beta \in GF(q)$

$$v_{m,n}^{(k)}(\beta) = \min_{(a_j)\in\mathcal{L}(m|a_n=\beta)} \left(\sum_{j\in S_v(m)\backslash n} u_{m,j}^{(k-1)}(a_j) \right) \quad (4)$$

variable node processing

for each variable node *n*, each $m \in S_c(n)$, each $\beta \in GF(q)$

$$u_{m,n}^{\prime(k)}(\beta) = \gamma_n(\beta) + \sum_{i\in S_c(n)\backslash m} v_{i,n}^{(k)}(\beta)$$

$$u_{m,n}^{(k)}(\beta) = u_{m,n}^{\prime(k)}(\beta) - \min_{\omega\in GF(q)}(u_{m,n}^{\prime(k)}(\omega))$$

a posteriori information computation & tentative decision

for each variable node *n*, each $\beta \in GF(q)$

$$z_n^{(k)} = \arg\min_\beta(\gamma_n(\beta) + \sum_{i\in S_c(n)} v_{i,n}^{(k)}(\beta))$$

B. Sum-Product Algorithm (SPA)

The BP decoding (also known as SPA) for binary LDPC codes can be extended to NB-LDPC codes. However, for a code constructed over *GF*(*q*), given the observation of a received symbol, the corresponding transmitted symbol can be any element of *GF*(*q*). Therefore, vectors of *q* probability messages instead of single probabilities need to be computed and stored during the decoding. This increases the complexity and the size of the memory for message storage by a factor of *q*. Moreover, the CN processing complexity is increased much more due to the non-binary check equations. As a result, NB-LDPC decoders have much higher hardware complexity than binary LDPC decoders, and their complexity increases quickly with the order of the finite field.

Use $u_{m,n}$ ($v_{m,n}$) to represent the message vector from CN *n* (check node *m*) to check node *m* (variable node *n*). Let $S_c(n)$ ($S_v(m)$) be the set of check (variable) nodes connected to variable (check) node *n* (*m*). Let $h_{i,j}$ be the entry of the PCM matrix in the *ith* row and *jth* column. Define the configuration set, $\mathcal{L}(m|a_n = \beta)$, as the set of sequences of finite field elements (a_j) $(j \in S_v(m)\backslash n)$ such that $\sum_{j\in S_v(m)\backslash n} h_{m,j}\, a_j = h_{m,n}\beta$. Let z_n be the hard-decision of the *n*-th received symbol, the BP for NB-LDPC decoding is listed in Algorithm 1 [70], where I_{max} is the maximum iteration number.

The VN processing and *a posteriori* message computation in NB-LDPC BP are direct extensions of those in binary BP. However, for a code with row weight d_c, the cardinality of a configuration set is $\mathcal{O}(q^{d_c-2})$. As a result, the CN processing in NB-LDPC decoding needs to sum up substantially more products and it is much more complicated.

To simplify the CN processing, a forward-backward method was proposed in [6] to break down the computations in (3) into an iterative process and share intermediate results. However, a large number of intermediate vectors need to be stored and the iterative process causes long latency. The computations in (3) can be also interpreted as convolution. A frequency-domain decoder was developed in [58] to convert convolutions to term-by-term multiplications. The hardware-consuming multiplications become additions in log-domain decoding algorithms [62], which are also more resilient to quantization noise. The mix-domain decoder in [63] tries to take advantage of both domains. Nevertheless, domain conversions implemented as expensive look-up tables are needed. Overall, log-domain decoders lead to lower hardware complexity.

At the cost of very little loss in the error-correcting performance, the hardware complexity is greatly reduced by approximations of the non-binary BP in the log domain, such as the EMS [64], [71], MM [65], simplified min-sum algorithm (SMSA) [72], syndrome-based EMS [67], and iterative reliability-based MLGD algorithms [66]. Many hardware implementations are based on these approximate algorithms. These algorithms are briefly discussed in the next section.

C. Extended Min-Sum (EMS) Algorithm

In the EMS and MM algorithms, messages are LLRs. For the *n*-th received symbol, its LLR associated with $\beta \in GF(q)$ is defined as $\ln(P(c_n = \hat{\beta}_n)/P(c_n = \beta))$, where $\hat{\beta}_n$ is the most likely *GF*(*q*) element for the *n*-th symbol. Hence, the LLR for the most likely field element in each vector is always zero and all the other LLRs are positive. Moreover, a smaller LLR means its associate field element is more likely to be the transmitted symbol. Using these LLRs, the non-binary BP can be approximated by the EMS algorithm shown in Algorithm 2.

The CN processing in the EMS algorithm is an approximation of that in the BP. To compensate for the approximation

error, the sums of the CN-to-VN messages can be scaled by a constant factor or added with a constant offset before they are added to the channel LLR in the VN processing and *a posteriori* information computation. Besides, the normalization in the VN processing is needed to make the smallest LLR in each vector zero.

It was first proposed in [73] to represent the variable-to-check (VN-to-CN) messages by the nodes in a Trellis with d_c stages. Then a configuration set in $\mathcal{L}(m|a_n = \beta)$ is equivalent to a path that passes exactly one node in each stage, except stage *n*. Accordingly, the CN processing is mapped to a path construction process. The constraints on the paths are relaxed in the SMSA [72] by allowing multiple nodes from the same stage to be included in a path. As a result, complexity is reduced at the cost of very slight performance degradation. To eliminate the redundancy among the computations of different CN-to-VN messages, it was proposed in [67] to first calculate syndromes that include the contributions of nodes from every stage. Then the contributions from the node in stage *n* are excluded to derive the CN-to-VN messages to VN *n*.

D. Min-Max (MM) Algorithm

The MM algorithm [65] is very similar to the EMS algorithm, except that the sum computation in (4) is replaced by 'max' comparison. Comparators have lower hardware complexity than adders and the maximum of the messages equals one of the messages. As a result, MM decoders achieve more complexity reduction than the EMS decoders, with slight performance loss.

E. Iterative Reliability-Based Majority-Logic Decoding (MLGD) Algorithms

Compared to other LDPC decoding algorithms, MLGD algorithms have lower complexity but inferior error-correcting capability, especially when the column weight is small. Two algorithms have been proposed in [66] to improve the performance of MLGD for NB-LDPC codes by carrying reliability information over the decoding iterations and incorporating it in the decoding decision.

Define the LLR associated with $\beta \in GF(q)$ as $\ln(P(\beta)/P(0))$. Let γ_n be the LLR vector for the *n*th received symbol computed from the channel information. Algorithm 3 lists the ISRB-MLGD algorithm proposed in [66]. In this algorithm, the scalar λ is used for performance optimization. $\phi_{m,n}$ can be considered as an extrinsic measure of the reliability from the channel. $\psi_n(\beta)$ is the extrinsic reliability that the *n*th received symbol is β contributed by all the connected CNs. It is accumulated to $R_n(\beta)$, which is carried over the iterations and used to make decoding decisions.

The IHRB-MLGD algorithm [66] assumes only hard decisions, z_n, are available from the channel. Compared to the ISRB algorithm, its major differences lie in the message initialization and extrinsic reliability accumulation. In the IHRB algorithm, $R_n^{(0)}(\beta)$ is initialized to a pre-set positive integer γ if $\beta = z_n^{(0)}$ or 0 otherwise. $\psi_n(\beta)$ is replaced by the count of $\sigma_{n,m}$ that equals β. The ISRB algorithm has better performance than the IHRB algorithm. However, the IHRB algorithm has lower complexity since its $\psi_n(\beta)$ values are easier to compute and its LLRs require shorter word length [74].

Algorithm 3 ISRB-MLGD Algorithm

input: γ_n

initialization: $z_n^{(0)} = \arg\max_\beta(\gamma_n(\beta))$;

$R_n^{(0)}(\beta) = \lambda\gamma_n(\beta)$;

$\phi_{m,n} = \min_{j \in S_v(m)\backslash n} \max_\beta \gamma_j(\beta)$

for $k = 1 : I_{max}$

Compute $z^{(k-1)}H^T$; stop if $z^{(k-1)}H^T = 0$

for each variable node n

for each $m \in S_c(n)$

$\sigma_{m,n} = h_{m,n}^{-1} \sum_{u \in S_v(m)\backslash n} \left(z_u^{(k-1)} h_{m,u}\right)$

$\psi_n(\beta) = \sum_{\sigma_{m,n}=\beta, i \in S_c(n)} \phi_{i,n}$

$R_n^{(k)}(\beta) = R_n^{(k-1)}(\beta) + \psi_n(\beta)$

$z_n^{(k)} = \arg\max_\beta(R_n^{(k)}(\beta))$

F. Trellis-Based Algorithms

Algorithm 4 extends the relaxations on the Trellis paths for CN-to-VN message computation [72], [75]. The CN processor converts the received messages into the delta domain. In (5), the messages are subtracted from the most reliable symbol in the received message ($u_{m,n}(\widehat{\beta_n})$), ensuring that all messages in the delta domain are non-negative and the first index of each message in the delta domain is always the most reliable symbol [67], [68].

From these received messages, a configuration set can be established. This set is defined by a path that contains the most reliable elements from each received message, denoted as a 0-order configuration. Other paths that differ from this 0-order configuration are called deviations. The sets of 0-order configurations and their deviations are defined by $conf(n_r, n_c)$, where the paths are formed from the n_r most reliable messages for a symbol β and can have n_c elements that deviate from 0-order configuration [68].

This configuration set ($conf(n_r, n_c)$) is used to build a syndrome column in (7), where the n_r most reliable symbols are considered, reducing complexity and increasing the degree of parallelism [67].

The CN-to-VN messages are generated by calculating the difference in the reliability of the configurations ($\Delta u_{m,n}$) and the syndrome ($\Delta U_{m,n}$) in (8). If this operation is associated with an existing output message, the minimum of those values is considered.

In (9), the messages are converted to the normal domain, and the VN processing is the same as the EMS algorithm.

The Trellis-MM algorithm is obtained by changing (7) into:

$$\Delta U^{(k-1)}(\beta) = \min_{\eta(\beta) \in conf(n_r,n_c)} \left(\max_{j \in S_v(m)\backslash n} \Delta u_{m,j}^{(k-1)}(\eta_j(\beta)) \right).$$

Algorithm 4 Trellis Extended Min-Sum (EMS) Decoding Algorithm

input: γ_n; $\mathbf{H}$; I_{max}
initialization: $u^{(0)}_{m,n}(\beta) = \gamma_n(\beta)$; $z^{(0)}_n = \arg\min_\beta(\gamma_n(\beta))$
for $k = 1 : I_{max}$
Compute $z^{(k-1)}\mathbf{H}^T$; Stop if $z^{(k-1)}\mathbf{H}^T = 0$

delta computation
for each check node m, each $n \in S_v(m)$, each $\beta \in GF(q)$

$$\Delta u^{(k-1)}_{m,n}(\eta_n = \widehat{\beta^{(k-1)}_n} + \beta) = u^{(k-1)}_{m,n}(\widehat{\beta^{(k-1)}_n}) - u^{(k-1)}_{m,n}(\beta) \tag{5}$$

syndrome computation
for each check node m, each $n \in S_v(m)$

$$\omega^{(k-1)}_m = \sum_{j \in S_v(m)\backslash n} \Delta u^{(k-1)}_{m,j}(\widehat{\beta^{(k-1)}_j}) \tag{6}$$

check node processing
for each check node m, each $n \in S_v(m)$, each $\beta \in GF(q)$

$$\Delta U^{(k-1)}(\beta) = \min_{\eta(\beta) \in conf(n_r,n_c)} \left(\sum_{j \in S_v(m)\backslash n} \Delta u^{(k-1)}_{m,j}(\eta_j(\beta)) \right) \tag{7}$$

$$\Delta v^{(k)}_{m,n}(\beta + \eta_n(\beta)) = \min_{\eta(\beta) \in conf(n_r,n_c)} (\Delta v^{(k)}_{m,n}(\beta + \eta_n(\beta)), \Delta U^{(k-1)}(\beta) - \Delta u^{(k-1)}_{m,n}(\eta_n(\beta)))) \tag{8}$$

$$v^{(k)}_{m,n}(\beta + \omega^{(k-1)}_m + \widehat{\beta^{(k-1)}_n}) = \Delta v^{(k)}_{m,n}(\beta) \tag{9}$$

This operation further increases throughput performance and reduces the area required for ASIC implementations [68]. In particular, for codes over *GF*(4), only one syndrome needs to be computed and the decoder is greatly simplified [76].

G. Other Algorithms

Other algorithms with less representation in the literature are also presented in this survey. The generalized bit-flipping decoding algorithm (GBFDA) computes the syndrome using the hard decision symbols obtained from the most reliable Galois field values. The hard decision symbols are then propagated to the nodes and accumulated along with decoding iterations [77]. Some variations extend the algorithm by considering more symbols with a multiple-vote symbol system, which reduces complexity but decreases decoding performance (>0.7 *dB*) [77].

In the Adaptive Multiset Stochastic Algorithm (AMSA), instead of exchanging probabilities, symbols are generated by stochastic stream generators with the probabilities encoded in the statistics of the stream [78]. This algorithm reduces area cost and hardware complexity at the expense of an increased memory requirement [78].

In the ADBP algorithm, messages are not exchanged as probability mass functions or LLRs but instead are defined by a class of Gaussian-like distributions cast into the general class of expectation-propagation algorithms [79]. This algorithm reduces complexity and memory requirements since they are independent of the Galois field's size [79].

Symbol reliability based (SRB) algorithms combine features from MLGD algorithms and multiple voting systems, resulting in decreased complexity with a slight decoding performance loss compared to the EMS and MM algorithms [80].

The weighted bit reliability-based (WBRB) algorithm [81] exchanges the minimum bit-reliability values instead of symbol-reliability values, such as MLGD algorithms. This algorithm allows CN and VN processing to overlap, leading to higher throughput performance, and lower complexity and memory requirement than the EMS algorithm. In the full bit reliability-based (FBRB) algorithm, all bit-reliability values of a symbol are used [81].

VI. Target Architectures for Non-Binary LDPC Decoding

In this section, it is presented the NB-LDPC decoder implementations for programmable (GPU), reconfigurable (FPGA) and dedicated architectures (ASIC) using the algorithms mentioned in the previous section.

These architectures provide different characteristics that can be exploited to highlight features in the algorithms and applications.

GPUs have better-suited features for prototyping and cost-sensitive systems due to low development effort and low non-recurring engineering costs while providing a high degree of programming flexibility. However, GPUs have fixed instruction sets and fixed memory hierarchies. They also have a high energy consumption.

Compared to GPUs, FPGAs do not have a fixed instruction set and allow a customizable memory hierarchy. These systems offer more flexibility at a higher development effort but consume less energy. However, FPGAs systems may have inefficient resource utilization due to the complexity of the routing network.

ASICs have a very high non-recurring engineering cost and development effort. Although, they can achieve the highest throughput performance. However, they provide a highly efficient design that consumes less energy compared to FPGAs.

The relationship and analysis between algorithms presented in Section V and the architectures presented in this section can be found in Section VII.

Tables IV, V, and VI present the year of published work, characteristics of the device, characteristics of the used code and the output metrics of the implementation. Some works report more than one decoder implementation in the same paper. In this case, they are considered different implementations and thus, have their own entry in Tables IV, V, and VI.

A. GPU

GPU devices still represent a growing research trend in this field, with several articles detailing

TABLE IV

CHARACTERISTICS OF NON-BINARY LDPC DECODERS ON GPUS GROUPED BY ALGORITHMS MENTIONED IN SECTION V. THIS TABLE PRESENTS THE YEAR OF PUBLISHED WORK, CHARACTERISTICS OF THE DEVICE (DEVICE, NUMBER OF CORES, PROCESS NODE AND DIE SIZE), CHARACTERISTICS OF THE USED CODE (CODE SIZE, CONNECTIONS PER VN AND CN, SIZE OF GALOIS FIELD, NUMBER OF ITERATIONS OF THE DECODER AND SNR) AND THE OUTPUT METRICS OF THE IMPLEMENTATION (DECODING THROUGHPUT AND POWER OF THE GPU)

Ref.	Year	GPU	GPU cores	Process size (nm)	Die size (mm^2)	Code size (N, M)	Weights (d_v, d_c)	GF(2^m) (m)	Iter.	SNR (dB)	Throughp. ($Mbps$)	Power (W)
Frequency domain sum-product algorithm (FFT-SPA)												
[82]	2013	Tesla C1060	240	55	470	384, 256	2, 3	5	5	N/A	1.63	188^3
									10		0.82	
									15		0.55	
								6	5		0.78	
									10		0.39	
									15		0.26	
								7	5		0.42	
									10		0.21	
									15		0.14	
									5		3.34	
									10		1.67	
									15		1.11	
								8	5		0.26	
									10		0.13	
									15		0.08	
									5		2.37	
									10		1.2	
									15		0.79	
									5		1.52	
									10		0.77	
									15		0.52	
[83]	2018	GTX580	512	40	520	384, 192	2, 4	6	10	N/A	16.6	244^3
		GTX280	240	65	576	384, 128	2, 6	5	5		26.4	236^3
								6			13.1	
								7			6.61	
								8			4.02	
[84]	2018	GTX TITAN X	3072	28	601	1152, 576	N/A	2	10	N/A	98.8	250^3
						576, 288		4			48.2	
						384, 192		6			16.6	
						288, 144		8			3.63	
						384, 256	2, 3	5	3		49.2	
								6			23.3	
								7			13.1	
								8			7.39	
Mixed domain sum-product algorithm (Mixed domain SPA)												
[85]	2015	GTX 580	512	40	520	1152, 576	$1.372, 2.743^2$	2	100^1	0	0.2	244^3
										1	0.35	
										2	1.2	
										3	1.6	
						768, 384	$1.289, 2.577^2$	3		0	0.21	
										1	0.22	
										2	0.4	
										3	0.6	
						576, 288	$1.181, 2.362^2$	4		0	0.2	
										1	0.4	
										2	1.4	
										3	1.85	
						462, 231	$1.081, 2.161^2$	5		0	0.15	
										1	0.49	
										2	1.3	
										3	1.9	
						384, 192	$1.035, 2.07^2$	6		0	0.37	
										1	0.39	
										2	0.7	
										3	1	
						330, 165	$1.148, 2.296^2$	7		0	0.13	
										1	0.45	
										2	1.25	
										3	1.7	
										0	0.09	
										1	0.43	
										2	1	
										3	1.45	
										0	0.38	
										1	0.42	
										2	0.82	
										3	1.2	
										0	0.05	
										1	0.2	

continued on the next page

highly parallel software-based designs [82]. Compute Unified Device Architecture (CUDA) is a framework for developing parallel programs on Nvidia GPUs. Nvidia coined the term CUDA core to designate the main GPU compute unit. While CPUs contain a few dozen cores, GPUs feature thousands of CUDA cores. The implementations

TABLE IV
(Continued.) CHARACTERISTICS OF NON-BINARY LDPC DECODERS ON GPUS GROUPED BY ALGORITHMS MENTIONED IN SECTION V. THIS TABLE PRESENTS THE YEAR OF PUBLISHED WORK, CHARACTERISTICS OF THE DEVICE (DEVICE, NUMBER OF CORES, PROCESS NODE AND DIE SIZE), CHARACTERISTICS OF THE USED CODE (CODE SIZE, CONNECTIONS PER VN AND CN, SIZE OF GALOIS FIELD, NUMBER OF ITERATIONS OF THE DECODER AND SNR) AND THE OUTPUT METRICS OF THE IMPLEMENTATION (DECODING THROUGHPUT AND POWER OF THE GPU)

continued from the previous page

Ref.	Year	GPU	GPU cores	Process size (nm)	Die size (mm^2)	Code size (N, M)	Weights (d_v, d_c)	GF(2^m) (m)	Iter.	SNR (dB)	Throughp. ($Mbps$)	Power (W)
[85]	2015	GTX 580	512	40	520	330, 165	1.148, 2.296[2]	7	100[1]	2	0.7	244[3]
										3	0.95	
						288, 144	1.005, 2.01[2]	8		0	0.03	
										1	0.21	
										2	0.47	
										3	0.7	
										0	0.24	
										1	0.38	
										2	0.6	
										3	0.9	
Min-max (MM) algorithm												
[86]	2012	GTX 470	448	40	529	620, 310	3, 6	5	10	3	0.694	215[3]
[87]	2014	GTX 650 Ti	768	28	221	744, 93	3, 24	5	15	N/A	1.81	110[3]
[88]	2015	GTX 650	768	28	221	744, 93	3, 24	5	10[1]	≤ 3	0.166	65[3]
										4	0.666	
										5	2.084	
										6	2.779	
										7	4.168	
										8	5.557	
		GTX Titan Black	2880		561					≤ 3	0.274	250[3]
										4	1.1	
										5	3.437	
										6	4.582	
										7	6.874	
										8	9.795	
[89]	2020	Jetson TX2	256	16	N/A	384, 256	2, 3	2	N/A	N/A	1.856	15[3]
								3			2.33	
								4			1.013	

[1]with early termination

[2]average weights (irregular codes)

[3]thermal design power (TDP) value

on GPUs mostly exploit the FFT SPA, mixed domain SPA and MM algorithms.

1) SPA: Table IV contains the most important GPU implementations from the literature. For the FFT-SPA, Andrade *et al.* [82] proposed a decoder with efficient data structures exploiting coalesced memory accesses and introduced simplifications using a fast Hadamard transform (FHT), achieving 3.34 Mbps.

In [83], the authors proposed efficient decoders using layered and flooding schedules, achieving 26.4 Mbps. In the same year, Lui *et al.* [84] proposed a multi-codeword decoder, reaching the best throughput result on GPUs of 98.8 Mbps by exploiting coalesced memory accesses.

For the mixed domain SPA, [85] proposed layered and flooding schedules for irregular codes, achieving 1.9 Mbps.

2) MM Algorithm: In 2012, Wang *et al.* [86] proposed the first NB-LDPC decoder on GPU for the MM algorithm, achieving 0.694 Mbps. For the same algorithm, another followed in [87], improving the throughput performance to 1.81 Mbps by removing multiplications and introducing a merger step, thus reducing latency.

Later on, the authors in [88] proposed a parallel block-layered approach to the NB-LDPC decoder, further improving performance and reaching 9.795 Mbps. While the GPUs implementations require a few dozen to several hundreds of Watts (65 to 250 W), the authors in [89], [90] were capable of reaching 2.33 Mbps using an embedded system that requires less than 15 W of power.

3) Analysis: The works [82]–[84], [87] do not report SNR values and do not feature early termination. These works evaluate the decoders' throughput performance, exclude the analysis of error-correction capability and can be compared with decoders that report a fixed number of iterations (without early termination).

Moreover, some implementations in [83] do not report the code weights (d_v, d_c), which does not provide information about the parallelization degree and is hard to replicate the decoder's evaluation parameters.

The focus of work [86] is also on throughput performance and reported SNR is just a complementary metric increasing the paper's value.

In [88], the focus is on the evaluation of the error-correction capability. The setup includes early termination and reports SNR values. Early termination stops the decoding process after the codeword is successfully decoded. Comparing works with and without early termination is not accurate since the exact number of iterations is unknown and influences the decoder's throughput. Comparisons between

TABLE V
CHARACTERISTICS OF NON-BINARY LDPC DECODERS ON FPGAS GROUPED BY THE ALGORITHMS MENTIONED IN SECTION V. THIS TABLE PRESENTS THE YEAR OF PUBLISHED WORK, THE NAME OF THE DEVICE, CHARACTERISTICS OF THE USED CODE (CODE SIZE, CONNECTIONS PER VN AND CN, SIZE OF GALOIS FIELD, NUMBER OF ITERATIONS OF THE DECODER AND SNR) AND THE OUTPUT METRICS OF THE IMPLEMENTATION (NUMBER OF REPLICATED CORES, LUTS, FLIP-FLOPS (FFS), EQUIVALENT LOGIC BLOCKS (ELBS), CLOCK FREQUENCY AND DECODING THROUGHPUT)

Ref.	Year	FPGA	Code size (N, M)	Weights (d_v, d_c)	GF(2^m) (m)	Iter.	SNR (dB)	Cores	LUTs	FFs	ELBs	Freq. (MHz)	Throughp. ($Mbps$)
Frequency domain sum-product algorithm (FFT-SPA)													
[91]	2014	PCIe385N D5	384, 256	2, 3	2	N/A	N/A	1	N/A	N/A	1764942	163.07	1.08
								4			1573808	206.52	3.36
					3			1			1715864	188.96	0.82
								4			1519138	216.07	1.73
					4			1			1663473	193.16	0.68
								4			1597208	203.5	0.98
[92]	2015	Virtex 7 XC7VX690T	384, 256	2, 3	2	10	N/A	1	60648	60648	60648	250	1.17
								14	346560	303240	346560	219	14.54
					3			1	90972	77976	90972	250	0.95
								6	350892	294576	350892	210	4.81
					4			1	129960	112632	129960	216	0.66
								3	316236	277248	316236	201	1.85
Log domain sum-product algorithm (Log domain SPA)													
[93]	2007	Virtex2P XC2VP100	720, 360	3, 6	3	N/A	N/A	1	21626	59916	59916	68.5	N/A
[63]	2009	Virtex2P XC2VP125	720, 360	3, 6	3	N/A	N/A	5	7630	9172	9172	102.68	0.92
								1	64538	17004	64538	68.5	0.36
Mixed domain sum-product algorithm (Mixed domain SPA)													
[93]	2007	Virtex 2P XC2VP100	720, 360	3, 6	3	N/A	N/A	6	4946	6728	6728	99.73	N/A
[63]	2009	Virtex 2P XC2VP125	720, 360	3, 6	3	N/A	N/A	1	2592	6728	6728	99.73	1.09
[94]	2010	Virtex 5	1024, 512	2, 4	2	1^2	N/A	1	4396	1031	4396	N/A	33.16^2
					4				8668	1571	8668		13.22^2
					8				89712	10206	89712		1.56^2
[95]	2013	Virtex 6 XC6VLX240T	480, 240	2, 4	5	10	N/A	5	21832	N/A	21832	180	6
Extended min-sum (EMS) algorithm													
[96]	2009	Virtex 4 XC4VLX160	972, 486	N/A	2	N/A	N/A	1	87783	34908	87783	131.41	50
[97]	2012	Virtex 5 XC5VLX155T	960, 480	2, 4	6	10	N/A	2	39010	16533	39010	100	≤ 2.44
								4	30198	12444	30198		2.44
									39010	16533	39010		9.76
[98]	2013	Virtex 4 XC4VLX200	192, 64	2, 6	6	8	N/A	5	34846	11712	34846	61.33	2.95
			72, 36	2, 4				10	16894	6530	16894	62.53	1.73
			48, 24						15906	6330	15906	64.15	1.77
Min-max (MM) algorithm													
[99]	2010	Virtex 2P XC2VP125	744, 93	3, 24	5	15	N/A	1	78784	44659	78784	106.19	9.3
[100]	2011	Virtex 2P XC2VP125	744, 93	3, 24	5	15	N/A	1	62866	34838	62866	123.44	13
[89]	2020	Zynq Ultrascale+	384, 256	2, 3	2	N/A	N/A	1	N/A	N/A	N/A	476.2	22.7
					3							476.2	31.9
					4							434.8	38.7
Iterative hard reliability-based majority-logic decoding (IHRB-MLGD) algorithm													
[74]	2012	Virtex 5 XC5VLX200T	403, 248	8, 13	5	15	N/A	1	15682	529	15682	117.6	90.68
									18306	716	18306	109.8	84.67
Trellis min-max (Trellis MM) algorithm													
[101]	2015	Virtex 7	2304, 256	4, 36	4	10	N/A	1	163288	51995	163288	226	630.4
Generalized bit-flipping decoding algorithm (GBFDA)													
[102]	2012	Virtex 2P	837, 124	4, 27	5	10^1	N/A	1	119357	62190	119357	95.62	247
		Virtex 6							141652	22562	141652	169.9	439
[103]	2013	Virtex 6	837, 124	4, 27	5	15	N/A	3	N/A	16884	16884	N/A	44.6
[104]	2013	Virtex 6	837, 124	4, 27	5	20	N/A	1	N/A	4070	4070	250	44.6
									49346	22121	49346	200	240
									59930	21632	59930	222	267
Adaptive multiset stochastic algorithm (AMSA)													
[105]	2013	Altera Stratix IV	192, 96	2, 4	6	N/A	2.4	2	66885	23150	66885	108	698
					8				91376	30840	91376		512
Weighted-bit reliability-based (WBRB) algorithm													
[81]	2016	Virtex 5 XC5VLX200T	403, 248	8, 13	5	N/A	4.5	1	21034	1269	21034	106.28	118.98
Full-bit reliability-based (FBRB) algorithm													
[81]	2016	Virtex 5 XC5VLX200T	403, 248	8, 13	5	N/A	4.5	1	72294	4556	72294	85.28	95.73

[1] with early termination

[2] Throughput values normalized for 1 iteration

[3] Implemented only for CN processor

decoders reporting SNR with early termination are only valid for implementations using the same SNR with early termination.

Moreover, [85] reports the values for irregular codes, which achieve higher error-correcting performance at a decreased throughput performance.

The one-page paper [89] focuses on comparing throughput performance between a low-power GPU and an FPGA and does not report SNR and the number of iterations, thereby limiting comparisons against other implementations.

B. FPGA

FPGA-based solutionsprovide a middle ground for designing and testing NB-LDPC solutions. FPGAs require less design effort than ASICs and less power than GPUs. In terms of throughput performance, FPGAs present themselves in the middle of GPUs and ASICs. Compared with GPUs, the FPGAs' implementations presented in Table V address a higher number of algorithms and higher throughput performances.

1) SPA: The first implementations on FPGAs were proposed in 2007 in [63] and [93]. These works provide a detailed description for the SPA in the Log and mixed domain, achieving 0.92 Mbps and 1.09 Mbps, respectively.

Lehnigk-Emden and Wehn [94] studied the first FPGA implementation for GF(2^2), GF(2^4) and GF(2^8), achieving 33.16, 13.22 and 1.56 Mbps respectively for the mixed domain SPA. Using the same algorithm, in [95], the authors propose the use of multiplier cores and introduced message normalization and pipeline processing, achieving 6 Mbps.

Andrade *et al.* used OpenCL in [91] to provide a parallel FFT-SPA decoder architecture suited to the characteristics of a wide-pipeline accelerator, reaching 3.36 Mbps. Later, for the same algorithm, the authors showed that using high-level synthesis (HLS) can reduce the development effort compared to register transfer level (RTL), while they were still able to achieve 14.54 Mbps for 14 cores.

2) EMS Algorithm: The first implementation of the EMS algorithm was proposed in [96] using a semi-parallel approach that achieved 50 Mbps. Another work proposed an FPGA implementation that exploited skimming, prefetching and relaxed redundancy control to reduce latency and enable an efficient pipeline schedule, achieving 9.76 Mbps [97]. For the same algorithm, in [98], a serial architecture is proposed implementing horizontal shuffled scheduling that reaches 2.95 Mbps.

3) MM Algorithm: For the MM algorithm, the authors of [99] proposed a partial-parallel decoder employing an overlapped scheme and layered decoding to simplify the architecture, achieving 9.3 Mbps. This implementation was further improved in [100] with the proposal of a flexible decoder for regular and irregular codes for various Galois fields, reaching 13 Mbps.

In [89], [90], the authors show that using HLS can reduce the development effort while maintaining a high performance, reaching 38.7 Mbps.

4) IHRB-MLGD Algorithm: An enhanced IHRB decoder is proposed in [74] that achieves a significant coding gain with small hardware overhead, achieving 90.68 Mbps.

5) Trellis-MM Algorithm: Lacruz *et al.* provided a significant breakthrough in [101]. The authors proposed a decoder with a reduction in memory resources by exploiting a layered schedule. The implementation also innovates by compressing the CN output messages, reducing the information sent to the VNs, thus achieving a quite significant performance of 630.47 Mbps.

6) Other Algorithms: For the GBFDA, in [102], the authors provide an implementation that achieves 439 Mbps with early termination. For the same algorithm, modifications are proposed. One avoids the sorting process and storing of unnecessary vectors, achieving 44.6 Mbps [103], while the other presents a technique to limit data growth and a broadcast mechanism to reduce routing congestion that can reach 267 Mbps [104].

The AMSA is proposed in [105], making the use of multisets, a generalization of sets that allows multiple instances of the same element, and achieving the best throughput of all FPGA implementations, in the order of 698 Mbps.

In [81], the authors proposed a layered partial-parallel WBRB decoder that reduces the amount of memory required, achieving 118.98 Mbps, and an FBRB decoder that obtains a better error-correcting performance and faster convergence rate, achieving a throughput of 95.73 Mbps.

7) Analysis: As discussed in the previous section, works [74], [92], [95], [97]–[101], [103], [104] focus on implementation and throughput performance (without early termination) and do not report SNR. These works can be compared to implementations with fixed iterations.

Additionally, the authors in [95] did not report the number of FFs, while in [103], [104], the number of used LUTs is not reported. However, (10) gives a metric for calculating the ELBs, which is an acceptable metric to estimate the FPGA resources used (given by [24]).

$$\max(number\ of\ 4-input\ LUTs,\ number\ of\ FFs). \quad (10)$$

On the other hand, in [81], [105] the authors analyze the throughput performance for a specific SNR, although not reporting the number of iterations. These results are valid, but comparison is possible only for decoders with the same SNR.

The implementations missing SNR and the number of iterations [63], [91], [93], [96] do not provide a meaningful contribution in terms of error-correction performance and, alternatively, they focus on resource requirement. In particular, the first decoder implementations [63], [93], [96] pioneered in the analysis of memory requirements for NB-LDPC decoders. Furthermore, the one-page paper [89] provides a comparison with a low-power GPU but due to missing metrics, it is hard to make a comparison between other implementations.

In [94], [102] report decoders with early termination without reporting SNR values, making it hard to compare error-correction capability and, consequently, throughput performance against other decoders. However, the authors in [94] normalized throughput performance by the number of iterations, thus facilitating comprehension of throughput performance. However, conclusions can not be taken from the decoder's overall performance since the total number of iterations is unknown.

C. ASIC

Among some of the most important designs found in the literature, top-performing ASIC designs are the best in terms of throughput performance and energy efficiency.

TABLE VI
CHARACTERISTICS OF NON-BINARY LDPC DECODERS ON ASICS GROUPED BY THE ALGORITHMS MENTIONED IN SECTION V. THIS TABLE PRESENTS THE YEAR OF PUBLISHED WORK, THE SIZE OF THE PROCESS NODE DESIGN, CHARACTERISTICS OF THE USED CODE (CODE SIZE, CONNECTIONS PER VN AND CN, SIZE OF GALOIS FIELD, NUMBER OF ITERATIONS OF THE DECODER AND SNR) AND THE OUTPUT METRICS OF THE IMPLEMENTATION (NUMBER OF GATES, CLOCK FREQUENCY, DECODING THROUGHPUT, AREA AND POWER OF THE DECODER)

Ref.	Year	Tech. (nm)	Code size (N, M)	Weights (d_v, d_c)	GF(2^m) (m)	Iter.	SNR (dB)	Gates (K)	Freq. (MHz)	Throughp. ($Mbps$)	Area (mm^2)	Power (W)
Log domain sum-product algorithm (Log domain SPA)												
[106]	2013	90	837, 124	4, 27	5	5	N/A	8510	250	233.53	46.118	0.893
Mixed domain sum-product algorithm (Mixed domain SPA)												
[94]	2010	65	1024, 512	2, 4	2	1^2	N/A	N/A	300	99.48^2	0.131	N/A
					4					39.66^2	0.208	
					8					4.69^2	1.45	
Extended min-sum (EMS) algorithm												
[72]	2012	180	837, 124	4, 27	5	15	N/A	1290	200	64	7.44^7	N/A
			744, 93	3, 24				1050			6.06^7	
[107]	2013	65	960, 480	2, 4	6	30^1	3.6	2780	700	1150	7.04	3.866
									400	656		0.726
[108]	2014	65	160, 80	2, 4	6	30^1	5	2780	700	1221	7.04	3.704
									400	698		0.729
						30			700	477		3.993
												1.974
[109]	2016	90	112, 56	2, 4	6	10	N/A	564	277	124.6	2.24	0.274
[110]	2017	65	160, 80	2, 4	6	10	3.1	2960	625	2222	9.59	N/A
								2880		1875	1.52^7	
										1538		
										1276		
[111]	2018	28	144, 24	2, 12	6	30^1	N/A	790	650	19500^4	0.10^7	N/A
[112]	2018	28	1512, 189	2, 16	6	1.5^8	4	1000	350	267	0.12^7	N/A
[113]	2020	28	144, 24	2, 12	6	30^1	5	790	650	10100	0.10^7	N/A
							4.5			7950		
							4			5730		
							3.5			2680		
							3			1060		
							≤ 3			650		
Min-max (MM) algorithm												
[114]	2010	180	620, 310	3, 6	5	10^1	N/A	14900	200	60	85.94^7	N/A
[73]	2010	180	837, 124	4, 27	5	15	N/A	79	150	10	0.46^7	NA
[115]	2011	90	837, 124	4, 27	5	15	N/A	3280	260	29^4	2.57^7	N/A
[116]	2011	90	837, 124	4, 27	5	N/A	N/A	3280	260	29	2.57^7	N/A
			620, 310	3, 6				2140		66.6	1.68^7	
			248, 124	4, 8				1920		47.69	10.33	0.4798
[100]	2011	180	837, 124	4, 27	5	15	N/A	1370	200	16	7.90^7	N/A
			744, 93	3, 24				1070		21	6.17^7	
			620, 310	3, 6				1240			7.15^7	
[75]	2012	180	837, 124	4, 27	5	15	N/A	871	200	66	5.02^7	N/A
[117]	2012	130	837, 124	4, 27	5	15^1	N/A	2130	500	64.3	3.44^7	N/A
[118]	2013	180	372, 124	4, 12	5	10	3.9	4100	220	982	23.65^7	N/A
		28	110, 22	2, 10	8			2570	520	546	1.289	0.976
[101]	2015	90	2304, 256	4, 36	4	10	N/A	975	333.3	964.7	10.49	N/A
[8]	2018	40	9396^6	3, N/A	3	6	5	N/A	120	2551	4.73	212.4
Iterative soft reliability-based majority-logic decoding (ISRB-MLGD) algorithm												
[119]	2014	180	255, 175	N/A	8	25^1	N/A	86.7	145	46.4	0.50^7	N/A
[120]	2019	90	4080^6	16, N/A	8	25	N/A	25	427	136	0.87	0.164
Iterative hard reliability-based majority-logic decoding (IHRB-MLGD) algorithm												
[74]	2011	180	403, 248	8, 13	5	25	N/A	450	250	193	2.60^7	N/A
[121]	2013	130	403, 248	8, 13	5	10^1	N/A	30.5	500	779	0.05^7	N/A
[8]	2018	40	9396^6	3, N/A	3	6	5	N/A	120	544	4.73	12.35
Trellis extended min-sum (Trellis-EMS) algorithm												
[67]	2013	40	3888, 432	3, 27	2	10^1	N/A	4000	1000^5	3600	0.8	N/A
										2400	0.72	
[122]	2013	40	3888, 432	N/A	2	N/A	N/A	N/A	N/A	3600	0.8	N/A
		90	837, 111		5	12		2750	205	484	2.15^7	
[123]	2014	90	837, 124	4, 27	5	9	N/A	2070	250	729	19.02	N/A
Trellis min-max (Trellis-MM) algorithm												
[123]	2014	90	837, 124	4, 27	5	N/A	N/A	1790	250	818	16.1	N/A
[68]	2014	90	837, 124	4, 27	5	9	N/A	3280	238	660	14.75	5.2
[124]	2014	90	837, 124	4, 27	5	N/A	N/A	1250	300	981	10.67	N/A
[125]	2015	90	2304, 256	4, 36	4	10	4.55	1420	380	1047	11.65	N/A
			1536, 192	3, 24	6	8		4050	300	1345^4	3.17^7	
			837, 124	4, 27	5			1170	345	1080	8.97	

continued on the next page

1) SPA: The SPA has increased complexity and it is hard to implement in hardware; only 2 works were found in the literature. In 2013, [106] proposed an efficient implementation with layered decoding to reduce the number of iterations and increase the throughput performance for SPA in the log domain, and managed to achieve 233.53 Mbps.

TABLE VI
(Continued.) CHARACTERISTICS OF NON-BINARY LDPC DECODERS ON ASICS GROUPED BY THE ALGORITHMS MENTIONED IN SECTION V. THIS TABLE PRESENTS THE YEAR OF PUBLISHED WORK, THE SIZE OF THE PROCESS NODE DESIGN, CHARACTERISTICS OF THE USED CODE (CODE SIZE, CONNECTIONS PER VN AND CN, SIZE OF GALOIS FIELD, NUMBER OF ITERATIONS OF THE DECODER AND SNR) AND THE OUTPUT METRICS OF THE IMPLEMENTATION (NUMBER OF GATES, CLOCK FREQUENCY, DECODING THROUGHPUT, AREA AND POWER OF THE DECODER)

continued from the previous page

Ref.	Year	Tech. (nm)	Code size (N, M)	Weights (d_v, d_c)	GF(2^m) (m)	Iter.	SNR (dB)	Gates (K)	Freq. (MHz)	Throughp. ($Mbps$)	Area (mm^2)	Power (W)
[126]	2015	28	2016, 252	4, 32	2	6.5[8]	N/A	266	500	6780	0.03[7]	N/A
[127]	2016	90	1536, 192	3, 24	6	8	4.4	2970	271	1259	28.9	N/A
								2999	265	1231	29.09	
			837, 124	4, 27	5			1060	393	1070	9.8	
[128]	2017	90	837, 124	4, 27	5	3	N/A	800	370	1274[4]	0.63[7]	N/A
[129]	2018	90	2304, 256	4, 36	4	10[1]	N/A	716	433.8	1396[4]	0.56[7]	N/A
[130]	2018	90	1512, 192	3, 24	6	8	4.4	2090	360	1403[4]	1.64[7]	N/A
			837, 124	4, 27	5			756	395	1261	7.8	
[131]	2019	90	837, 124	4, 27	5	2.68[8]	4.5	N/A	526.62	4681.27	6.86	N/A
						8				1640		
[132]	2019	90	837, 124	4, 27	5	8	N/A	667	591	1155[4]	0.52[7]	N/A
Generalized bit-flipping decoding algorithm (GBFDA)												
[103]	2013	90	837, 124	4, 27	5	15	N/A	590	250	89	0.46[7]	N/A
								639	500	16	0.50[7]	
[104]	2013	90	837, 124	4, 27	5	20	N/A	814	476	573	13.4	N/A
								706	263	316	13.28	
								590	500[4]	89[4]	0.46[7]	
Adaptive multiset stochastic algorithm (AMSA)												
[78]	2014	90	136, 68	2, 4	5	N/A	N/A	740	455	120[4]	0.58[7]	N/A
Analog-digital belief propagation (ADBP) algorithm												
[79]	2014	45	5000, 2000	2, 5	6	10	N/A	27.6	300	27	0.038	N/A
Symbol reliability-based (SRB) algorithm												
[80]	2014	180	403, 248	8, 13	5	15[1]	N/A	1450	500	97	8.36[7]	N/A
Weighted-bit reliability-based (WBRB) algorithm												
[48]	2018	90	837, 124	4, 27	5	10[1]	N/A	4540	207.04	21661.56	3.56[7]	N/A
Full-bit reliability-based (FBRB) algorithm												
[133]	2016	65	16, 8	2, 4	8	N/A	N/A	N/A	222	4372	0.69	N/A
										789	0.41	

[1]with early termination

[2]Throughput values normalized for 1 iteration

[3]Implemented only for CN processor

[4]synthesis result

[5]After place & route

[6]Total # of symbols

[7]Calculated values using the following areas per gate ($180nm - 5.768\mu m^2/gate$; $130nm - 1.613\mu m^2/gate$; $90nm - 783003.334nm^2/gate$; $65nm - 527431.042nm^2/gate$; $28nm - 123817.224nm^2/gate$)

[8]average number with early termination

For the mixed domain SPA, the same authors that implemented the first FPGA design [94] also proposed the first ASIC designs for GF(2^2), GF(2^4) and GF(2^8), achieving 99.48, 36.66 and 4.69 Mbps, respectively.

2) EMS Algorithm: The majority of the ASIC designs found in the literature implement either the EMS algorithm or the MM algorithm. In 2012, [72] proposed the first EMS algorithm implementation in ASICs, achieving 64 Mbps.

Soon after, the authors of [107] applied dynamic clock gating to the CNs to achieve greater energy efficiency, reaching 1.15 Gbps. Park *et al.* further improve the algorithm by adding a one-step look-ahead on the CN, enabling an increase in the operating frequency to 700 MHz, and by overlapping the execution of CN and VN operations, reducing the decoding iterations and reaching 1.221 Gbps [108].

Other works include a partial-parallel layered decoder employing (1) a double-throughput CN, (2) overlapped CN and VN processors and (3) pipelined scheduling to achieve 124.6 Mbps [109].

In [110], the authors propose a fully overlapped decoder with (1) an early bubble check to reduce the initialization latency of the CN, (2) a backward memory scan method to overlap the CN and VN and hide CN latency within the VN. Also, they incorporate a redundant memory reuse technique to further decrease the latency of a single decoding iteration, allowing the decoder to reach 2.222 Gbps of throughput performance.

A year later, Harb proposes a pre-sorting procedure that sorts messages entering the CN based on their reliability values, where the less likely messages are omitted. The author also presents a hybrid CN model combining the forward-backward and syndrome-based algorithm. The decoder is capable of

achieving 19.5 Gbps [111]. Later, Harb proposed a processing block that updates CNs and their associated VNs in a fully pipelined way, reaching 10.1 Gbps [113].

In [112], a hybrid architecture is proposed that combine the extended-forward and the forward-backward approaches to reduce the total number of computed syndromes reaching 267 Mbps.

3) MM Algorithm: The first MM algorithm ASIC decoder proposed in [114]. It delivers 60 Mbps of throughput performance. Another version, such as that in [73], presents a MM algorithm decoder integrating a scheme that sorts out a limited number of VN-to-CN messages with the highest reliability. Then, the CN-to-VN messages are derived independently from the sorted messages, reducing computational complexity and memory usage without noticeable performance loss, achieving 10 Mbps.

The authors of [115] presented a selective-input layered decoder with an associated compensation technique. In the same year, in [116], they offered a barrel-shifter-based permutation network and a minimum value filter used to determine the first few smallest values from a given set. Those designs are capable of achieving 29 and 66.6 Mbps, respectively.

The authors in [100], proposed a flexible decoder for both regular and irregular codes for various Galois fields, achieving 21 Mbps. After that, Cai and Zhang [75] developed a relaxed CN processing scheme for the MM algorithm. By making use of the property that a linear combination of the elements can uniquely represent each finite field element on the minimum basis, all the entries in a message vector are computed simultaneously in and efficiently, reducing memory requirement and computational complexity, and achieving 66 Mbps.

Lin and Yan [117] proposed two shuffled scheduling schemes that increase throughput performance while reducing complexity and memory usage. These scheduling schemes have slightly better error performance and converge faster than flooding schedule and are capable of achieving 64.3 Mbps. The same authors demonstrated that a design can reach 982 Mbps if it adopts a scheme that approximates the CN-to-VN messages using a piecewise linear function, instead of using probabilities, and so reducing memory requirements [118].

From the same design developed for FPGA, Lacruz *et al.* provided an ASIC architecture in [101], lowering the need for memory resources by exploiting a layered schedule. The implementation also innovates by compressing the CN output messages, which reduces the information sent to the VNs, thus achieving a performance of 964.7 Mbps.

The last design of the MM algorithm belongs to Toriyama and Marković [8]. The authors proposed a decoder for storage applications with a reduction in message bitwidth, thus reducing memory requirements and achieving 2.551 Gbps. The same work also presents a design for the IHRB-MLGD, reaching 544 Mbps, respectively.

4) ISRB-MLGD Algorithm: Lu *et al.* [119] proposed the first partial-parallel ISRB-MLGD decoder, introducing alternate message-passing schemes, reaching almost the same BER performance as the MM algorithm. This design saves memory, produces a higher operating frequency and, consequently, a higher throughput performance with less hardware complexity, achieving 46.6 Mbps.

The latest developed work belongs to Song *et al.* [120]. They improved [119] by proposing a clipped-modified ISRB-MLGD algorithm, introducing two complexity-reduction modifications and an unsaturated clipping method, achieving higher performance with lower implementation complexity, raising the throughput to 136 Mbps.

5) IHRB-MLGD Algorithm: Reliability-based designs have been proposed both for hard and soft decoding. The authors in [74] presented an enhanced IHRB-MLGD decoder that achieves a significant coding gain with small hardware overhead, obtaining 193 Mbps.

For the same algorithm, in [121] the authors propose a partially parallel layered IHRB-MLGD algorithm, which delivers better error performance and faster convergence, achieving 779 Mbps.

The authors of [8] proposed a decoder for storage applications with a reduction in message bitwidth, thus reducing memory requirements and achieving 544 Mbps.

6) Trelis-Based Algorithms: Some designs are based on Trellis-based algorithms, with Li *et al.* [67] proposing the first Trellis-EMS algorithm implementation in an ASIC. The same authors added an extra column for message representation, achieving 3.6 Gbps [122], thus breaking the barrier of 1 Gbps decoding.

Reference [123] is a study that proposes implementations of the Trellis-EMS algorithm and Trellis-MM algorithm. The proposed architectures eliminate the computation of the second minimum operation in messages of the CN processor and present efficient estimators to infer the second minimum value, reducing the complexity and latency and increasing throughput, reaching 729 Mbps for the Trellis-EMS algorithm and 818 Mbps for Trellis-MM algorithm.

In [68], a simplified Trellis-MM decoder is proposed where the CN messages are computed in parallel, using only the most reliable information in a horizontal layered schedule, achieving 660 Mbps.

This work was further improved in [124], which uses compressed message passing between nodes to decrease the routing area and memory requirements; it increases throughput up to 981 Mbps. The work in [124] is further improved, allowing even fewer messages to be exchanged and achieving 1.345 Gbps, as described in [125].

In [126], the authors relaxed the constraints on the nodes included in the decoder configuration, reducing the hardware units by 3 times and allowing the CN-to-VN messages to be computed in one clock cycle achieving 6.78 Gbps. Lacruz *et al.* [127] proposed an approximation for the Trellis-MM algorithm to reduce complexity for the CN processor, achieving 1.259 Gbps.

A series of works presented by Thi and Lee proposed a two-extra-column Trellis-MM algorithm with layered scheduling and thus reducing the complexity of the CN processor and the decoder area, achieving 1.274 Gbps [128].

This work was further improved in [129] by reducing the number of exchanged messages and enabling the decoder to reach 1.396 Gbps. The same authors proposed a basic-set Trellis-MM algorithm in [130] that considers a set of independent field elements with the smallest LLRs, achieving 1.403 Gbps.

Their latest work [132] proposes a half-row modified two-extra-column Trellis-MM algorithm that reduces hardware complexity by 50%, keeping only the two elements with the highest reliabilities in each extra column, achieving 1.155 Gbps.

In [131], a row-parallel Trellis-MM decoder is proposed with compression, a two-stage minimum finder and an early termination scheme to improve throughput performance and decrease storage requirements. The design is capable of reaching 4.681 Gbps.

7) Other Algorithms: For the GBFDA, in [103] and [104], the authors describe implementations for FPGAs but also propose ASIC designs. The work reported in [103] achieves 89 Mbps, while the [104] study reaches 573 Mbps.

Some lesser known algorithms are also reported in the literature. In [78], a decoder is proposed using the AMSA with tracking forecast memory that achieves 540 Mbps.

In [79], the authors propose the ADBP algorithm that works on factor graphs over linear models and uses messages in the form of Gaussian-like probability distributions by tracking their parameters, capable of achieving 27 Mbps.

Zhou *et al.* [80] propose the SRB algorithm that performs well when the column weight is low, achieving 97 Mbps.

For the WBRB algorithm, Tian *et al.* achieve the best performance of all NB-LDPC decoders known in the literature, with a throughput of 21.661 Gbps.

Lastly, the authors of [133] provide an ASIC implementation for the FBRB algorithm using GF(2^8) that achieves 4.372 Gbps.

8) Analysis: The authors of [8], [108], [110], [118], [125], [127], [130], [131] focus on throughput performance by reporting the number of iterations without early termination and SNR values. The two reported values allow an analysis of the throughput performance and comparison with implementations without SNR and a fixed number of iterations (without early termination). Moreover, the decoders that include early termination [107], [108], [112], [113], [131], only allow for a fair comparison between implementations with the same SNR.

Contrarily, works that do not report SNR but report a fixed number of iterations (without early termination) [68], [72]–[75], [79], [100], [101], [103], [104], [106], [109], [115], [120], [122], [123], [128], [132], can be compared with decoders in the same class (no SNR and fixed number of iterations) since throughput performance can be normalized by the number of iterations.

Furthermore, works that do not report SNR and the number of iterations [78], [116], [122]–[124], [133] or employ early termination [48], [67], [80], [94], [111], [114], [117], [119], [121], [126] cannot be compared against other decoders since the SNR impacts the throughput performance and little can be inferred from the number of iterations.

In [8], [120], the authors do not report the code size but instead report the number of symbols, while in [122] they do not report code weights (d_v, d_c). These setups do not allow researchers to understand the parallelization degree of the decoders.

Of the surveyed papers, only [8], [68], [106]–[109], [116], [118], [120] report values for power. Power values help understand energy efficiency and to which extent these decoders can be deployed. In addition to the previous works, [74], [79], [94], [101], [104], [110], [121]–[125], [127], [130], [131], [133] report area values. Area is a constraint in systems with limited space, such as satellites. In Table VI, the missing area values were calculated through the average area per gate for a certain process node design.

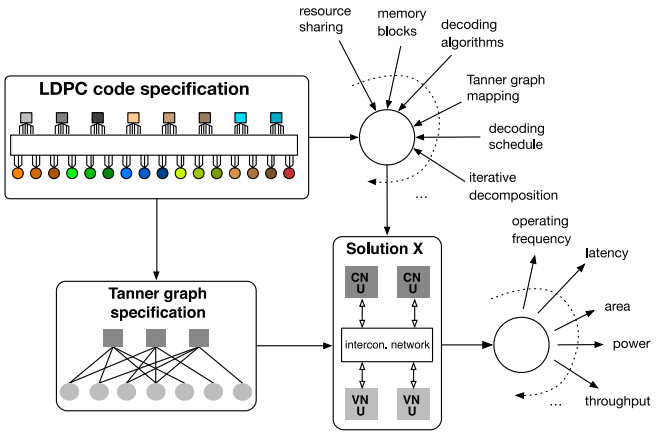


Fig. 6. Representation of the design process of an LDPC code. The ECC solution considers algorithmic and hardware aspects to produce a solution tailored to fulfill specific parameters.

Works [78], [104], [111], [115], [125], [128]–[130], [132], report synthesis values that do not represent the true decoder values but provide an approximation.

Finally in [8], [94], [122], [131], [133] the number of gates is not reported, which would offer insightful information about the amount of resources consumed to produce the decoders.

VII. Analysis, Constraints and Design Parameter Considerations

In this section, a comprehensive analysis is provided for the surveyed literature and the parameters that influence the design process of NB-LDPC decoders are considered and discussed.

As depicted in Fig. 6, several parameters (decoding algorithms, code size, etc.) and hardware specifications, such as memory, decoding schedules, etc., must be considered for producing a solution, which influences the decoder's latency, area, power and throughput.

Examining results without normalization increases the data analysis difficulty due to the huge number of variables of NB-LDPC codes. To make a fair comparison between designs, the plots are normalized by the number of connections between CNs and VNs or edges ($N \times d_v$ or $M \times d_c$), excluding GPUs.

A. GPU

GPUs represent the platforms with the least development effort and with the lowest throughput performance. The main advantage of GPUs is the faster development cycle and the hardware abstraction, allowing them to test and implement more complex decoders. GPUs also enable for flexible decoders to be implemented and eliminate the need

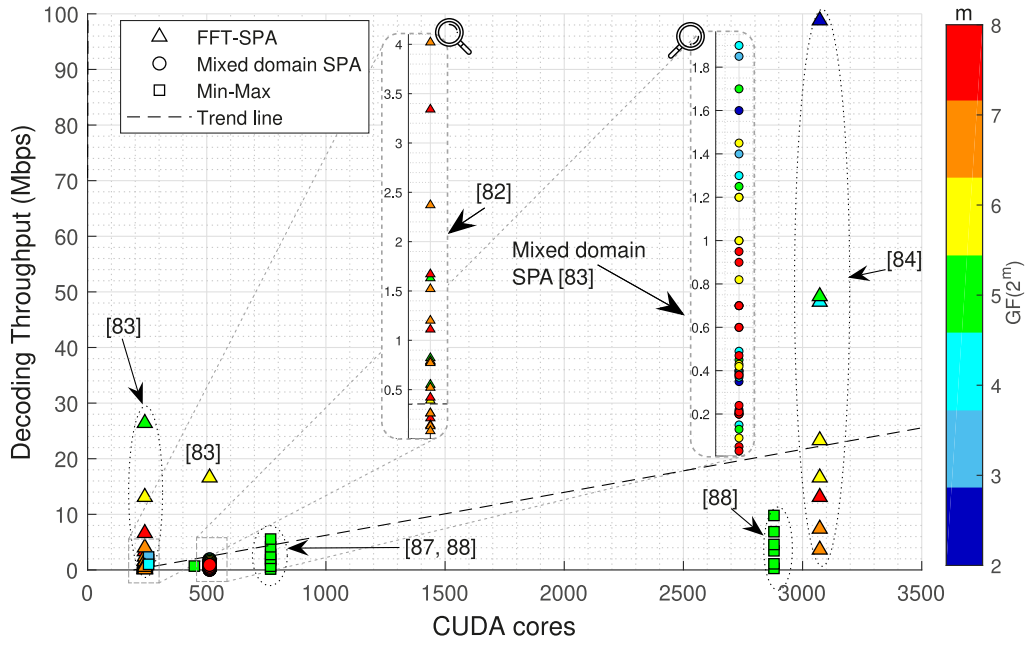


Fig. 7. Decoding throughput versus the number of CUDA cores for all the surveyed GPU decoders. The colored right vertical bar represents different Galois fields in the implementations. The throughput increases for smaller Galois fields and higher number of CUDA cores. The trend line is calculated using linear regression.

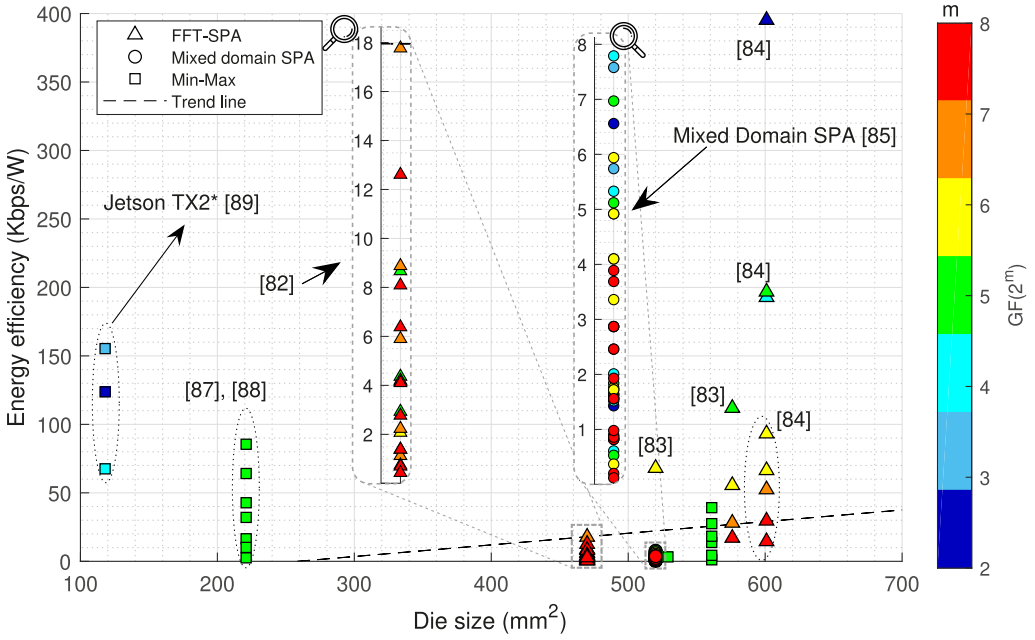


Fig. 8. Energy efficiency versus die size for all the surveyed GPU decoders. The throughput is higher for smaller Galois fields and slightly increases on GPUs with larger die sizes. It is assumed a die size of $118mm^2$ for the results from the Jetson TX2. These were not included in the calculation of the trend line.

for dedicated hardware with correspondingly high levels of non-recurring engineering cost.

As shown in Fig. 7 and Fig. 8, the decoding throughput tends to increase with the number of cores and, consequently, the GPU's die size, as shown by the trend lines. It is also observed that implementations in higher fields result in lower throughput performance. This is expected since higher Galois fields consume more resources and therefore, the performance drops.

The mixed domain SPA implementation [85] uses different code sizes and Galois fields with early termination on the same GPU. One major drawback of this approach is that it fails to make a fair comparison of throughput performance. In codes with early termination, the throughput performance can be the same for large codes and smaller fields as for shorter codes and higher fields. This can be identified by the clumped circles in Fig. 7 and Fig. 8.

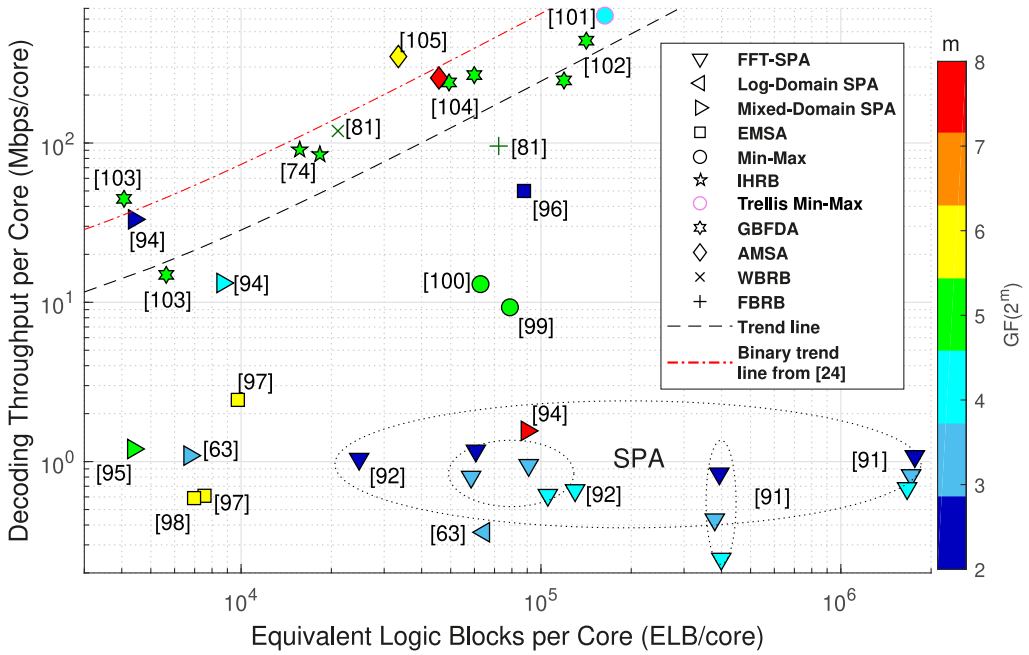


Fig. 9. Decoding throughput per core versus ELBs per core of the surveyed FPGA decoders. The trend line is calculated using linear regression, excluding the SPA points. A red trend line for $GF(2)$ from [24] is plotted for comparison.

The analysis of Fig. 8 is more intricate. The energy efficiency is related to the GPU technology and newer architectures are usually more efficient, as can be seen in the figure, particularly for the MM algorithm. The Jetson TX2 results are represented in the figure, although the actual die size is not publicly available, and it was assumed to be the same size as its predecessor (Jetson TX1), i.e., 118*nm*.

B. FPGA

Compared to GPUs, FPGAs allow developers to implement decoders at the hardware level, using several techniques, such as HLS and RTL. Electronic design automation (EDA) tools are used to automate processes, such as place & route, and reduce the implementation complexity, i.e., the programmer is detached from parts of the design.

Nonetheless, these systems have constraints in terms of area and resources, hindering the ability to create an efficient NB-LDPC decoder. In particular, FPGAs are vulnerable to codes with many nodes or high Galois fields since those increase connections and FPGAs can only support a limited number of complex wiring connections.

Due to the variation of resource organization in Intel (former Altera) and Xilinx FPGAs, a metric called ELB represents the maximum number of required FFs or 4-input LUTs that would be needed to implement the design. To compensate for the increased size of 6 LUTs in some devices, in comparison with 4-input LUTs, we approximate each 6-input LUTs to become equivalent to two 4-input LUTs. The total number of ELBs of an implementation corresponds to $max(total\ 4-LUTs, total\ FFs)$ [24].

Most of the FPGA designs replicate decoders to occupy the available resources of the FPGA. To provide a fair comparison, the points present in Fig. 9 are normalized by the number of reported cores, represented in Table V, ensuring the lowest number of ELBs to implement one decoder.

By applying a linear regression, an upward trend can be seen as the number of ELBs increases. SPA decoders were not included in the regression because they were from older implementations and this algorithm is computationally more complex than the rest.

Points above the trend line present efficient designs in terms of throughput performance per resource unit. Of those, the designs from [105] using the AMSA show a good performance for $GF(2^6)$ and $GF(2^8)$. Implementations using SPA show a poor performance due to the computational complexity of the algorithm. These designs achieve a throughput performance of less than 2 Mbps except for [94]. The designs from [91] are the largest decoders and occupy more than 300K ELBs per core. In [63], [92], [95], the solutions achieve less resource utilization. However, their efficiency falls short of the trend line. Of all SPA decoders, [94] is the one that achieves good performance for $GF(2^2)$ and $GF(2^4)$. For $GF(2^8)$, the design becomes too complex and the resource utilization per core increases 10-fold.

EMS and MM algorithms do not have efficient designs. However, [101] delivers the best throughput performance per core. The remaining solutions, implemented using GBFDA, AMSA, WBRB and FBRB, follow the expected trend and are considered good designs.

Characteristics of the NB-LDPC code, notably the code size and the operating field, impose restrictions on FPGA-based designs in terms of resources. The different PCM varieties found in the literature create difficulties in analyzing the results because to larger codes are more complex and use more resources.

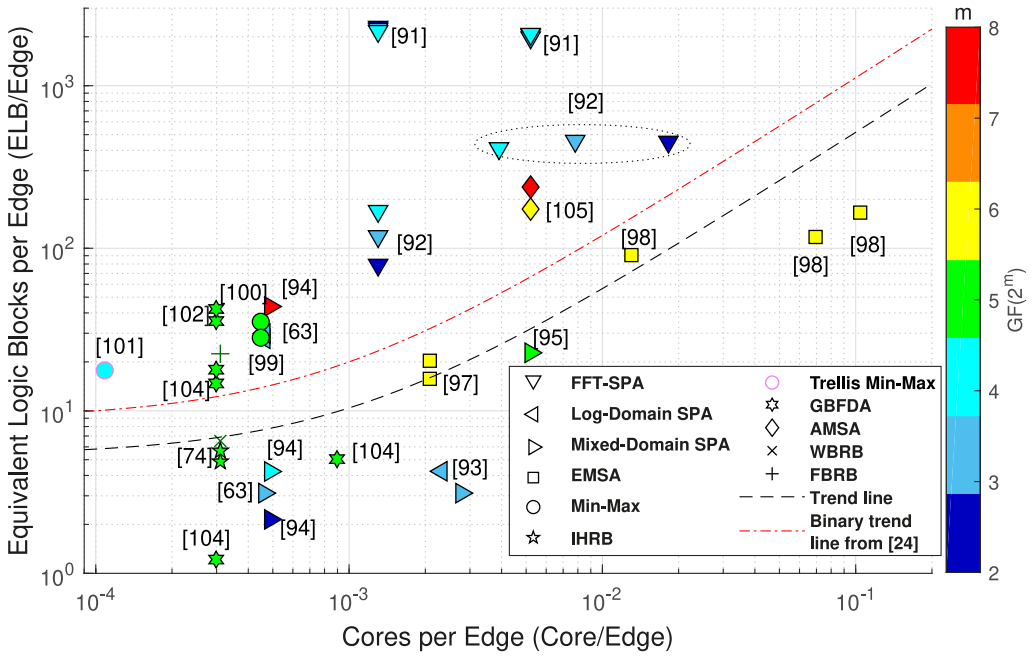


Fig. 10. ELBs per edge versus cores per edge of the surveyed FPGA decoders. The trend line is calculated using linear regression. A red trend line for $GF(2)$ from [24] is plotted for comparison.

The points presented in the following two figures are normalized by the number of edges, and so comparison between designs is possible, regardless of the code size. Fig. 10 provides resource utilization per core. Ideally, the designs should use as few ELBs as possible. In this analysis, the most efficient designs are located in the bottom-left corner and below the trend line in terms of resource utilization.

Designs using FFT-SPA in [91], [92] and AMSA in [105] consume a great deal of resources. Log and mixed domain SPA shows a resource utilization below the trend line [63], [93]–[95], [134].

EMS and MM algorithms [97]–[101] follow the trend line and the MM design of LaCruz *et al.* [101] is one of the least resource-consumption solutions.

The implementation of [104] also achieves an efficient design, as it is the most efficient in resource utilization. Designs in [74], [81], [102] process the same edges per core but consume more ELBs. Nevertheless, these can be considered excellent designs.

Fig. 11 provides a measure of raw throughput performance. The data points follow a trend when the throughput is plotted against the cores. The optimal point is located in the top-left corner of the plot, where the throughput per edge is high and the designs can execute many of the edges in a single core.

Designs using SPA are located below the trend line [63], [91]–[93], [95] except for [94]. These implementations generally achieve a poor performance due to the algorithm's complexity and to it being one of the oldest. The EMS algorithm also achieves poor results, with all points being located below the trend line [63], [98]. A different story is told for the MM implementations. Those are located above the trend line [89], [100], [101], except for [99]. Implementations using the IHRB [74], GBFDA [102], [104], WBRB [81] and FBRB [81] algorithms achieve similar performance when decoding the same amount of edges by core.

In conclusion, the AMSA [105], GBFDA [102], [104] and Lacruz *et al.* [101] designs are the best-performing FPGA solutions in terms of throughput efficiency, according to Fig. 9 and Fig. 11. However, if resource allocation (Fig. 10) is taken into consideration, only the designs from [101] and [104] are considered since they have low resource utilization.

C. ASIC

The surveyed ASIC designs are primarily implemented for $GF(2^5)$ and $GF(2^6)$. For both Fig. 12 and Fig. 13, the optimal points are in the top-left corner of the plot since throughput should be maximized and the maximum number of edges should be processed in the minimum number of gates and area.

Data points in Fig. 12 follow a linear trend. Log-domain SPA [106] and MM [73], [75], [100], [114]–[117] designs show a poor performance that can be explained by their use of older process node technologies. The designs from [101], [118] use recent process node designs (28 and 90*nm*) and can achieve better performance even for $GF(2^8)$.

Designs using the EMS algorithm produce a good performance. This is particularly true for the designs in [111], [113], which amount to one of the best designs in terms of efficiency. Variants that use Trellis schemes for the EMS and MM algorithms achieve results in the same trend and use fewer gates per edge. Using the Trellis-MM algorithm, Zhang *et al.* were able to achieve the best ASIC designs [126].

The designs using ISRB [120], IHRB [121] and WBRB [48] achieve excellent performance and can be alternatives to better-known algorithms such as MM and EMS.

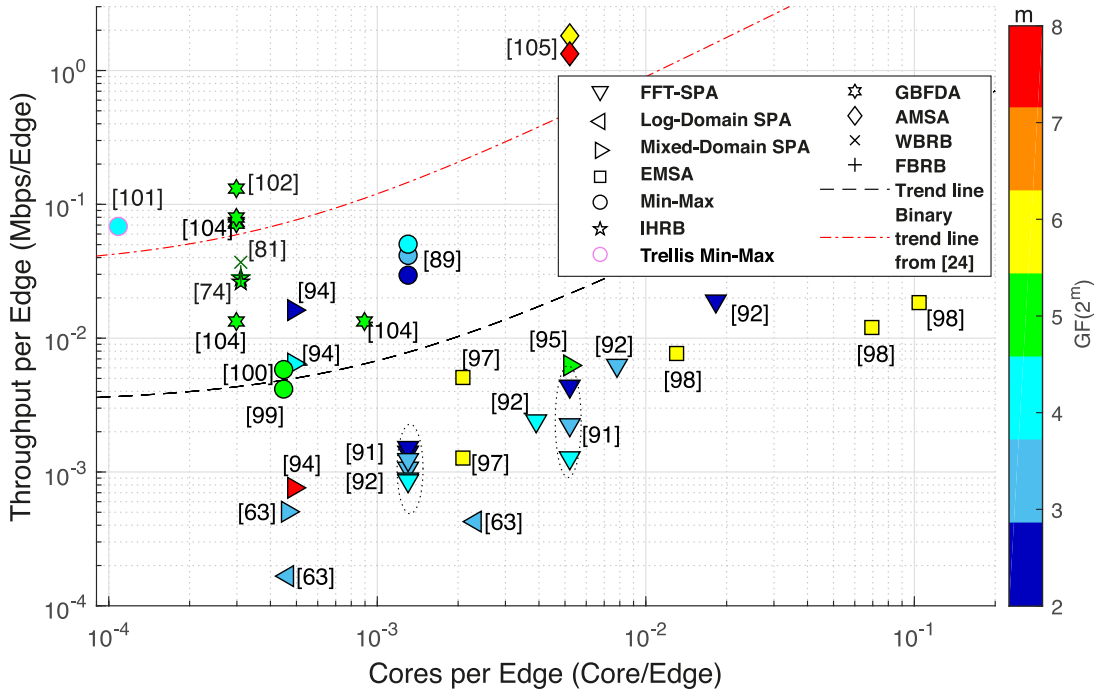


Fig. 11. Throughput per edge versus cores per edge of the surveyed FPGA decoders. The trend line is calculated using linear regression. A red trend line for $GF(2)$ from [24] is plotted for comparison.

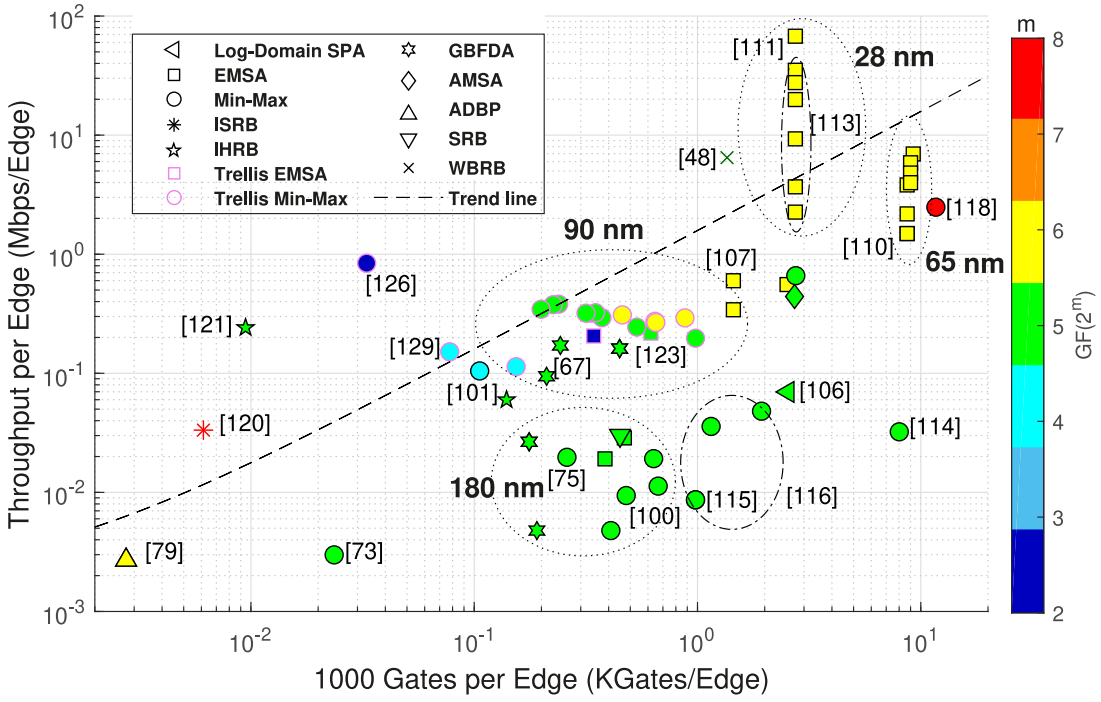


Fig. 12. Throughput per edge versus the number of gates per edge for all the surveyed ASIC decoders. The colored right vertical bar represents different Galois fields in the implementations. The trend line is calculated using linear regression.

In this figure, we can see that data points can be clustered in process node technologies. For this chart, there is a direct relationship between process node design and achieved throughput. For smaller technologies, the throughput is higher, even when adjusted for the trend.

Like previous figure, Fig. 13 presents the throughput and area normalized by the number of edges. Instead of a linear trend line, in this figure the trend is quadratic since increasing the area provides double the resources to increase the throughput.

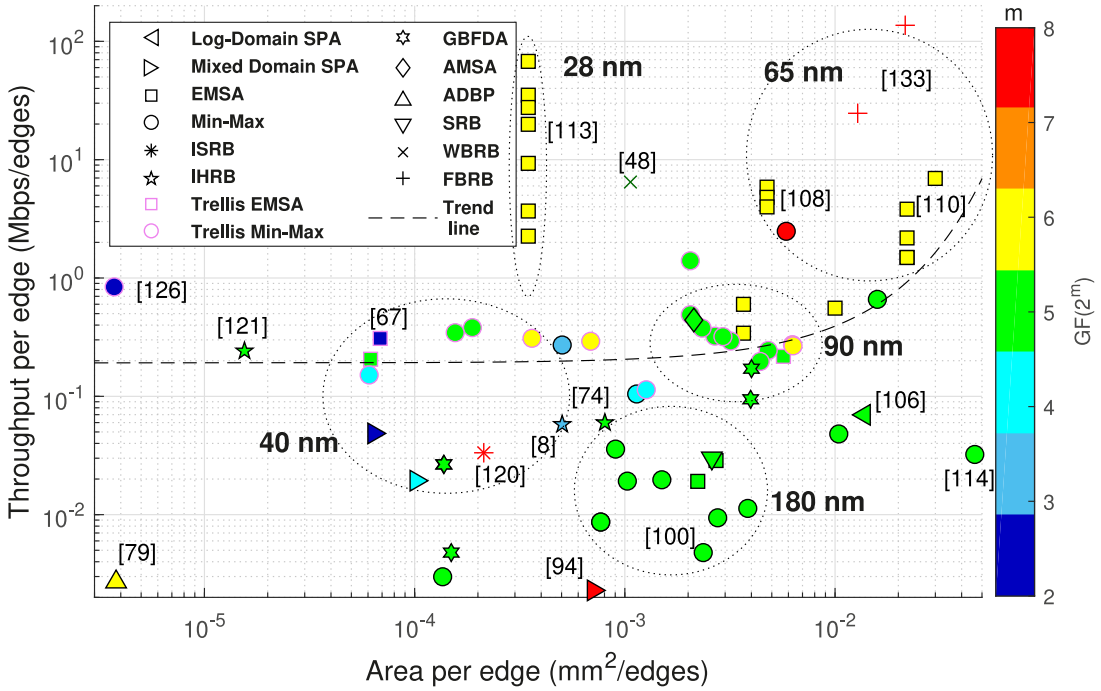


Fig. 13. Throughput per edge versus area per edge for all the surveyed ASIC decoders. The trend line is calculated using linear regression.

From Fig. 12, a similar analysis can be done for Fig. 13. SPA and MM decoders have poor area efficiency because they use older process node designs. However, MM designs from [8], [118], [126] stand out and achieve above-average performance, in particular for the design in [126], which has the lowest area per edge. Using a Trellis scheme for MM substantially improves the decoder's performance, placing the implementation on the trend line. For the EMS algorithm, in general the efficiency is excellent, with designs from [113] being among the candidates for the most efficient designs. The exception to the above-average performance is [72] since it uses an old process node technology (180*nm*). Contrary to MM decoders, using Trellis schemes for EMS designs lowers performance to Trellis-MM algorithm levels.

Designs using IHRB, ISRB, GBFDA and AMSA achieve similar performance that follows the trend. WBRB and FBRB algorithm implementations have shown impressive performance on ASIC. However, designs that use ADBP and SRB algorithms have one of the poorest performances.

The data points can be clustered by the type of process node design. ASICs using 180*nm* technology are mainly located below the trend line. This is expected since the technology is older and thus less efficient. The 90, 65 and 40*nm* technologies are situated around the trend line, with the designs using 40*nm* occupying less area. The cluster using 65*nm* is slightly higher than the expected throughput performance. This can be explained by most of the points being implemented using EMS, which is the best algorithm to achieve throughput performance per edge, as can be seen in Fig. 13. The cluster using 28*nm* technology achieves the best performance per area since it is the most efficient technology.

In conclusion, designs using Trellis-MM [126] and EMS [113] algorithms are the best performing implementations in terms of area and gate efficiency with the solution from [121], using the IHRB-MLGD algorithm falling short of those designs for lower Galois fields.

D. Design Considerations

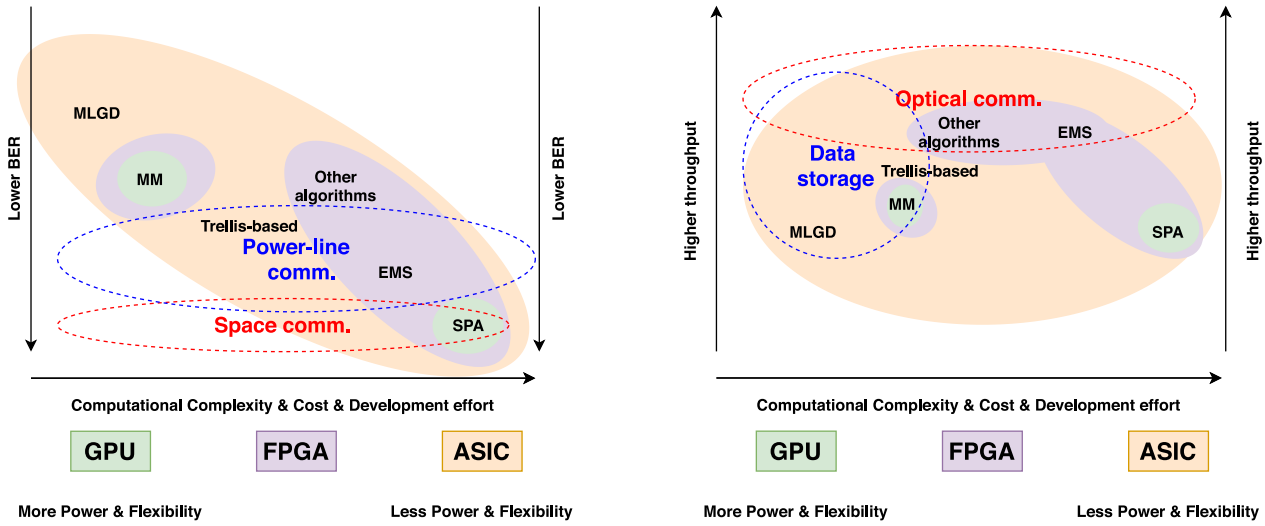
Fig. 6 generalizes the process of specifying and designing NB-LDPC decoders. From the code specification, multiple variables need to be considered to produce an NB-LDPC decoder solution. This solution is expected to comply with various performance parameters such as throughput, latency, area, power and flexibility.

1) Throughput: The most crucial output metric is throughput performance: the number of bits decoded per unit of time. This metric is required for high-speed data communication for applications described in Section II (Section II-B).

2) Latency: Second, latency is the time that the system takes to decode a codeword. Applications should have the lowest latency possible. In particular, safety-related applications (Section II-E) require low response times. Latency is intrinsically connected to throughput performance and the implementations often pose a trade-off between throughput and latency. As a result, the designers must consider the type of application and design the decoder in accordance with requirements.

3) Hardware Resources: Next, the hardware resources impose limitations on the performance and flexibility of the decoder. Specifically, memory plays a huge role in the decoder's performance, with high-performance decoders requiring more resources and larger footprints.

4) Power and Energy: As for power and energy-constrained systems such as satellites and smartphones (Section II-A), energy-efficiency must be contemplated. The works found



(a) Representation of a multi-domain design exploration relating architectures, applications and BER. Algorithms on the lower part of diagram, have better error-correcting performance (i.e., lower BER).

(b) Representation of a multi-domain design exploration relating architectures, applications and decoding throughput. Algorithms on the top part of diagram, can achieve higher throughput performance.

Fig. 14. Algorithms on the right side have an increased complexity, cost and development effort. The horizontal axis correlates with the architectures, with ASICs having higher cost, complexity and development effort, while requiring lower power levels and having less flexibility. Although, the decoding algorithms are not bound to a specific architecture, the literature shows GPU implementations for the SPA and MM. For FPGAs, implementations with SPA, MM, EMS and other algorithms can be found and all algorithms have implementations for ASICs, as shown in Table IV, V, and VI. Although not observed on this diagram, low complexity algorithms and different degrees of parallelism can improve throughput performance for the same silicon area.

in the literature do not report energy consumption values. Despite that, some works report power values and FPGA and ASIC's energy-efficiencies can be inferred. Concerning the GPU results, the thermal design power (TDP) values are considered.

5) Flexibility: Another important metric is the decoder's flexibility. In general terms, the literature reports implementations using fixed designs. However, if the decoders have the ability to process different PCMs they can improve the error-correcting capability by adapting to the changing noise levels of the environment. Admittedly, such flexibility increases the decoder complexity, particularly in ASICs, but FPGAs and GPUs can provide flexible solutions, too.

6) Field, Code Size and Code Rate: The error-correction performance increases for (1) high order fields, (2) longer code sizes, allowing more symbols to be decoded and (3) low code rates. However, these metrics scale exponentially with the increase of code edges, code size and, especially, high-order fields, making the implementation of decoders impractical for larger values.

Another parameter that the designer must consider when designing decoders is flexibility. Allowing the system to decode variable code sizes or different Galois fields, significantly increases complexity and decreases throughput performance.

Most of the works reported use designs that process different PCM sizes with different weights ranging from 200 to 10000 edges.

7) Iterations: The number of iterations imposed on the system can impact error-correction capability and the decoder's throughput performance. Received codewords with a high level of noise can result in undecodable codewords, and these systems usually employ a sufficient maximum number of iterations (I_{max}) to stop the non-converging decoding process. I_{max} is chosen in order to achieve a given FER goal.

However, various works employ a fixed number of iterations without early termination for benchmark purposes. This diversity imposes a barrier to make a fair comparison between implementations. Some works provided in Tables IV, V and VI either do not specify the number of iterations or give the maximum number of iterations with early termination but fail to pinpoint the exact number of iterations taken for that experiment, making it difficult to compare implementations.

8) Quantization: The quantization of the input probabilities (γ_n) also plays a role in the decoder's complexity. Converting floating-point arithmetic to binary produces a decoder with a reduced hardware complexity. This provides a trade-off between resolution and convergence. However, such a trade-off has implications for the decoder's throughput performance. Lower resolutions have higher throughput performance but reduce error-correction performance.

9) Technology: Lastly, the technology used influences the decoder capability. Faster clock frequencies translate into higher throughput performance. Usually, the implementation must be tailored to accommodate the platform requirements, resulting in different approaches to the same algorithm.

VIII. Key Takeaways

The analysis of the NB-LDPC decoders is a multi-domain design exploration with many dimensions. Fig. 14 depicts the relationship between decoding algorithms, architectures and

TABLE VII
COMPARISON OF DIFFERENT DECODING ALGORITHMS AND ARCHITECTURES FOR DIFFERENT METRICS USING RELATIVE SCORES RANGING FROM 1 TO 5. THE ARROWS ON THE LEFT COLUMN DESCRIBE THE GOAL OF THE METRIC, WHERE, FOR INSTANCE, A 5 ON THROUGHPUT MEANS THE HIGHEST THROUGHPUT PERFORMANCE AND 5 ON COMPLEXITY MEANS THE SIMPLEST DECODER

	SPA			**EMS**		**MM**			**MGLD**		**Trellis-based**		**Other algorithms**	
	GPU	FPGA	ASIC	FPGA	ASIC	GPU	FPGA	ASIC	FPGA	ASIC	FPGA	ASIC	FPGA	ASIC
Throughput $\uparrow$	2	2	3	2	5	1	2	4	2	3	3	4	3	5
BER $\downarrow$	5			4		3			2		$4-3$		3	
Complexity $\downarrow$	1			2		3			4		3		3	
Memory $\downarrow$	1			3		4			4		3		5	
Power $\downarrow$	1	3	5	3	5	1	3	5	3	5	3	5	3	5
Area $\downarrow$	1	3	3	3	4	1	3	3	3	5	3	3	4	3
Flexibility $\uparrow$	5	3	1	3	1	5	3	1	3	1	3	1	3	1
Development time/effort $\downarrow$	5	3	1	3	1	5	3	1	3	1	3	1	3	1

TABLE VIII
COMPARISON OF DIFFERENT DECODING ALGORITHMS FOR THE DIFFERENT APPLICATIONS MENTIONED IN SECTION II. (HIGHER NUMBER OF '+' IS BETTER)

	Space comm.	**Optical comm.**	**Data storage**	**Power-line comm.**
SPA	+++	+	+	++
EMS	++	+++	+	+++
MM	++	++	++	++
MGLD	+	+	+++	+
Trellis-based	++	++	++	+++
Other algorithms	++	+++	++	++

applications for the most important metrics, BER and throughput. The following sections describe the takeaways from this survey, with Table VII summarizing the relationship between decoding algorithms and architectures and Table VIII giving an overview of which decoding algorithms are better suited for the applications in Section II, according to the authors' opinion.

A. Applications

Most of the applications require a compromise between throughput and BER performance. High noise levels characterize space communications and throughput performance is not a crucial metric for these systems. The decoders employed in this should maximize the error-correction performance and consider other variables such as memory, power, and area since these are usually employed in resource-constrained systems such as satellites.

In terrestrial optical communications, these constraints are eliminated and noise is much less influential. Therefore, on these applications, throughput is prioritized to allow fast data communication.

For data storage applications, throughput and BER should be balanced. However, this application requires the decoder to have low complexity to decrease costs and consume low energy levels.

Noise is also present in power-line communications. However, the BER performance required in these applications is more relaxed than in space communications and the throughput performance requirement is more strict.

The new applications have different types of requirements. For example, communications for autonomous vehicles require a high level of throughput performance but, more importantly, need a decoder with small latency. A great effort has been put into exploring energetic efficiency decoders. Emergent applications such as Internet-of-Things and wearable devices require low-power communications.

B. Algorithms

1) SPA: SPA has the best error-correction performance at the cost of a decreased throughput performance. For this algorithm, ASICs provide a higher throughput rate. However, for a cost-effective decoder, GPUs represent a good solution. FPGA implementations have a similar throughput performance to GPUs. In applications where the throughput performance requirement is relaxed and in noisy communications channels, for instance in space and power-line communications (Sections II-A and II-D), this algorithm has an advantage over the rest.

2) EMS: Compared to SPA, EMS decoders have less computational complexity, require less memory and area, while achieving higher throughput performance, particularly on ASICs. These decoders can accomplish this at the cost of reduced error-correction performance. This decoding algorithm is better suited for applications that require high decoding throughput, for instance in optical communications (Section II-B) or a good BER and throughput performance, as in power-line communications (Section II-D).

3) MM: The MM decoders have even less complexity but the reported results do not show higher throughput performance. Despite this, this algorithm has a reduced error-correction performance compared to the EMS. The area utilization on ASICs is one of the worst, together with the SPA. The reduced throughput of the algorithm does not suit high-speed applications. For example, this algorithm does not suit optical communications. However, it has a reduced complexity compared to the SPA and EMS, making a good implementation on data storage decoding systems(Section II-C).

4) MLGD: MLGD algorithms show to have poor throughput and BER performance. However, they require less memory usage and are less complex than the ones previously mentioned. They also have the most efficient power per area

ratio. Although they are similar, the IHRB-MLGD algorithm has a better performance per area than the ISRB-MLGD. These decoding algorithms are better for applications where throughput performance is not crucial and BER requirement is more relaxed. They provide the best solution for memory-constrained, as in data storage systems (Section II-C) and energy-constrained systems, for instance in Internet-of-Things applications or wearable computing communications devices (Section II-E).

5) Trellis-Based: Compared to the EMS and MM decoders, Trellis versions further reduce complexity with a minimal loss in error-correction performance. The Trellis-EMS [113] and Trellis-MM [126] provide a good trade-off between throughput performance and error-correction capability and are widely available, with good implementations for data storage (Section II-C) and power-line applications (Section II-D). The Trellis-EMS is more inclined to BER performance and Trellis-MM more suitable for throughput performance, and both provide the best throughput per area ratio.

6) Other Algorithms: For lesser reported algorithms, GBFDA decoders are a good fit for memory-constrained applications, for example, data storage applications (Section II-C). AMSA, WBRB, FBRB, ADBP and SRB algorithms are not featured in many studies published in the literature but can achieve high throughput performance, making them suitable for applications that require high-speed communications, as in optical communications (Section II-B). In particular, the ADBP decoder is shown to have a very low resource utilization and area, making them suitable for data storage and wearable communications devices (Sections II-C and II-E). However these decoding are still new and more research is needed.

C. Final Remarks

1) Galois Fields: Choosing higher Galois fields increases the decoder's complexity and requires greater amounts of memory, which creates constraints in terms of datapath and edge communications. Galois fields between $GF(2^2)$ and $GF(2^4)$ are standard for testing NB-LDPC decoders. For FPGAs and ASICs, an excellent field choice to implement decoders is in $GF(2^5)$ and $GF(2^6)$. Throughput performance dramatically decreases with the increase in the field order. However, decoders in $GF(2^7)$ and $GF(2^8)$ result in complex wiring and connections, making decoders hard to implement. With FPGA and ASIC process node design technology and tools development, it is expected to see the implementation of many new high-order fields. In GPUs, hardware abstraction allows for high-order fields to be implemented easily.

2) Architectures: The hardware designer must know the platform limitations when choosing the proper VLSI system to implement these decoders. As depicted in Fig. 14, a customized ASIC design is expensive and highly complex implementation-wise. However, this system will present the best throughput performance, low area utilization and high energy efficiency (as shown in Table VII). Using FPGAs reduces cost and can be helpful for specific scenarios where the run-time configuration is used (Fig. 14), but faces the wiring and connection restrictions that ASICs do, and has reduced throughput performance. GPUs detach the developer from the hardware and have a faster development cycle but significantly reduce the throughput performance, as illustrated in Fig. 14(a), Fig. 14(b) and Table VII. The latter is good for iterative co-design (exit chart analysis[135]) towards superior BER performance, to validate NB-LDPC decoders and test the BER performance of new codes.

3) Code Choice: In terms of code length, most decoders found in the literature employ codes between a few hundred (> 500) and a few thousand (< 2000) symbols for the decoder to have some practicality. It has been shown that NB-LDPC codes with low column weights perform well. In [50], the authors showed that for fields higher than $GF(2^6)$, codes with $d_v = 2$ have an optimal average Hamming weight spectrum, and thus have a higher error-correcting performance associated with low hardware complexity. The work in [55] presents methods for constructing NB-LDPC codes with low column weights (d_v) used in the majority of the surveyed literature. Furthermore, some authors provide a collection of the codes [136]–[138].

IX. Future Developments

This section provides an examination of all decoder implementations by analyzing the number of gates that each implementation requires and provides insights about future challenges and evolution of NB-LDPC decoding.

To compare designs in different systems, understand how future systems will evolve, and check if some designs satisfy the current requirements, a standardized evaluation should be adopted across all systems. Figs. 15 and 16 present the achieved throughput against the number of transistors normalized by the number of edges ($N \times d_v$ or $M \times d_c$). The number of transistors for GPUs is calculated according to the manufacturer's specifications. For an ASIC, it is considered that a gate has 2 transistors. Finally, for FPGA, it is assumed 1 ELB (14-input LUT (156 transistors) + 1 FF (18 transistors)) to have 174 transistors.

A. Unified Analysis

Fig. 15 illustrates the relative performance against the number of transistors for all surveyed systems. GPU implementations have a more compacted cloud than those of FPGAs. This is due to this type of device being more inflexible and containing more overheads. On the other hand, FPGA designs can achieve higher throughput using the same number of transistors as GPUs do. This is expected because FPGAs are more flexible and designs can be tailored to specific algorithms. For instance, GPUs can quantize samples in 8-bit variables, while FPGAs can use fewer customized bits, thus creating more efficient designs. ASICs sacrifice the flexibility of an FPGA system to achieve even higher throughput for the same number of transistors.

Outliers in the figure are found for both FPGA and ASIC designs. This is expected because these devices can provide custom hardware design and are implemented in different Galois fields, process node designs and algorithms, thus increasing variability in performance.

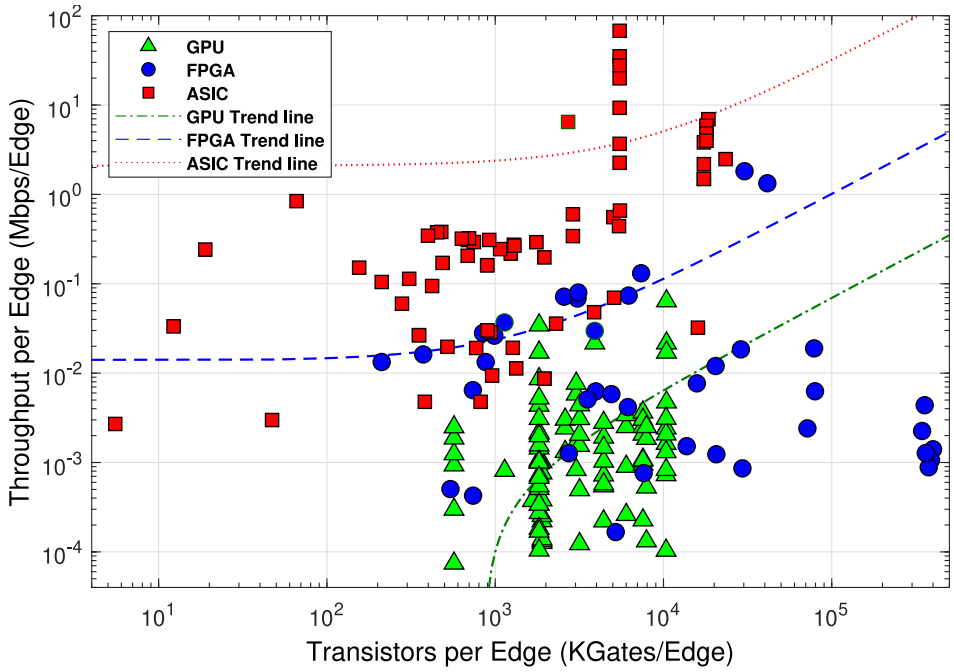


Fig. 15. Throughput per edge versus transistors per edge for all the surveyed decoders implemented on GPUs, FPGAs and ASICs. The trend lines are calculated using linear regression.

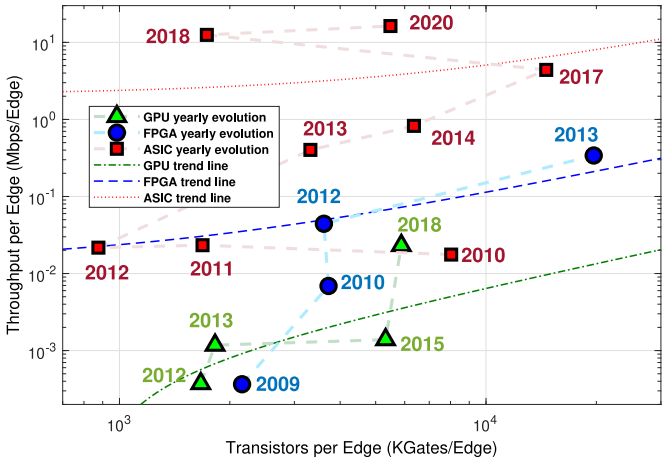


Fig. 16. Throughput per edge versus transistors per edge yearly evolution for the decoders implemented on GPUs, FPGAs and ASICs. The trend lines are the same from Fig. 15. Each point represents the center of mass of every implementation in the corresponding year.

The trend lines show how throughput evolves with the variation in the number of transistors. GPUs, FPGAs and ASICs have different efficiency trend lines. As observed in the trend lines of Fig. 15, it is clear that FPGAs are 10 times more efficient than GPUs in terms of throughput per transistor per edge. ASICs have a similar trend line to FPGAs. However, the transistors from these custom-made circuits are 30 times more efficient than FPGAs, allowing higher throughput performance using the same number of transistors per edge.

B. Challenges and Goals Ahead

An important step is to analyze how these implementations develop over time. Fig. 16 shows the trend for the landmark years for the three architectures, keeping the same trend lines as in Fig. 15 for a reference point. These points represent the center of mass of implementations published in the same year. Some outlier years in this figure are not shown due to implementations using older process node designs and being underrepresented, with one or two papers published in that mentioned year.

1) Higher Throughput and Computational Complexity, and Lower BER: Some conclusions can be taken from this evolution. Overall, it is expected that future designs may achieve higher throughput, enabling the execution of larger and more complex codes, and leading to superior BER performance. Furthermore, with the evolution of these architectures and the reduction of the process node design technology, the number of transistors per edge should increase as well as the designs' energy efficiency.

As depicted in Fig. 16, the trend shows an increase in throughput performance mainly due to process node design size reduction and the adoption of new hardware optimization techniques. Table IX shows the main research challenges from technological and algorithmic perspectives for developing future NB-LDPC decoders. Algorithmic parameters rely on developing new codes that will probably become larger and require the use of higher-order Galois field arithmetic, thus increasing BER performance and decoding complexity. Moreover, with more CNs and VNs processors incorporated into the design, more memory should be required. Also, new memory technologies have been proposed that allow part of the processing to be performed in memory, which will benefit throughput performance and energy efficiency, namely by reducing the data movement bottleneck.

2) More Efficient Hardware Design With Lower Development Effort: Regarding the interaction of future

TABLE IX
RESEARCH CHALLENGES FOR FUTURE NB-LDPC DECODERS

	GPU	FPGA	ASIC
Algorithmic parameters (for future codes)			
Throughput	Increase		
BER	Decrease (i.e., superior error-correcting performance)		
Complexity	Try to limit increase while keeping BER (depends on new codes and algorithms)		
Memory	Try to limit increase (depends on new codes and algorithms)		
Technological parameters (for future codes)			
Power	Likely to maintain or decrease		
Flexibility	Slight increase	Slight increase	Increase
Development time/effort	Decrease	Slight decrease	Slight decrease

codes and NB-LDPC decoding algorithms with technological developments of process node design, although codes may expectedly become larger and algorithms more complex, the size and power of the transistors are expected to decrease, leading to similar energy consumption levels and circuit area. The evolution of architectures and new EDA tools can potentially increase the decoder's flexibility and decrease development time and effort. Although a tremendous research effort has been invested in this field, it has mainly been dedicated to binary LDPC decoders, opening room to new research that is still required both for architectures and NB-LDPC codes, namely for targeting superior BER performance in the range of 10^{-15} dB and lower, while still guaranteeing very high throughput.

X. CONCLUSION

In this paper we have presented an exhaustive listing of NB-LDPC decoder architectures and systems found in the literature and discussed their practicalities and limitations, described best matching between algorithm and applications and highlighted the best performances and open challenges for future designs. The large number of algorithmic variables, and the range of code sizes, architectures, algorithms and hardware make the task of comparing decoders hard. Nevertheless, we concluded that SPA decoders have the best BER but a poor throughput performance. Trellis decoders provide the best trade-off between error-correction capacity and throughput and have excellent performance per area. The MLGD decoder further reduces complexity but shows poor BER performance. Other decoders (GBFDA, AMSA) are more suitable for memory-constrained applications.

The process of investigating NB-LDPC decoders involves (1) the prototyping of new architectures to test the error-capability performance of new NB-LDPC decoders and (2) the design of decoders for specific applications, taking into consideration tight constraints regarding throughput, latency and energy.

ASIC-based decoders are more suitable for energy-constrained applications that demand high throughput performance. GPUs, however, are suited to applications without power restrictions, such as base stations where arrays of GPUs can provide a cost-effective solution. In the middle term, FPGAs can provide throughput one order of magnitude higher than GPUs and one order below ASICs, and the opposite for energy. The reprogrammable and flexible nature of FPGAs and GPUs can also offer good platforms to prototype, test and validate new codes, to reduce costs and development time. ASIC-based decoders can generate prohibitive costs and should therefore be used in application-specific designs.

REFERENCES

[1] Z. Babar *et al.*, "Polar codes and their quantum-domain counterparts," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 123–155, 1st Quart., 2020.

[2] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. ICC IEEE Int. Conf. Commun.*, vol. 2, 1993, pp. 1064–1070.

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

[6] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over *GF*(*q*)," in *Proc. IEEE Inf. Theory Workshop*, 1998, pp. 70–71.

[7] S. Shao *et al.*, "Survey of turbo, LDPC, and polar decoder ASIC implementations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2309–2333, 3rd Quart., 2019.

[8] Y. Toriyama and D. Marković, "A 2.267-Gb/s, 93.7-pJ/bit non-binary LDPC decoder with logarithmic quantization and dual-decoding algorithm scheme for storage applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 8, pp. 2378–2388, Aug. 2018.

[9] A. Abdmouleh, E. Boutillon, L. Conde-Canencia, C. A. Nour, and C. Douillard, "A new approach to optimise non-binary LDPC codes for coded modulations," in *Proc. 9th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, 2016, pp. 295–299.

[10] L. Costantini, B. Matuz, G. Liva, E. Paolini, and M. Chiani, "On the performance of moderate-length non-binary LDPC codes for space communications," in *Proc. 5th Adv. Satell. Multimedia Syst. Conf. 11th Signal Process. Space Commun. Workshop*, 2010, pp. 122–126.

[11] L. Wen, J. Lei, and J. B. Wei, "Design of concatenation of fountain and non-binary LDPC codes for satellite communications," in *Proc. 2nd Int. Conf. Inf. Eng. Comput. Sci.*, 2010, pp. 1–4.

[12] G. Liva, E. Paolini, T. De Cola, and M. Chiani, "Codes on high-order fields for the CCSDS next generation uplink," in *Proc. 6th Adv. Satell. Multimedia Syst. Conf. (ASMS) 12th Signal Process. Space Commun. Workshop (SPSC)*, 2012, pp. 44–48.

[13] B. Chang, D. Divsalar, and L. Dolecek, "Non-binary protograph-based LDPC codes for short block-lengths," in *Proc. IEEE Inf. Theory Workshop*, 2012, pp. 282–286.

[14] A. Mansour, R. Mesleh, and M. Abaza, "New challenges in wireless and free space optical communications," *Opt. Lasers Eng.*, vol. 89, pp. 95–108, Feb. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0143816616300252

[15] R. Achiba, M. Mortazavi, and W. Fizell, "Turbo code performance and design trade-offs," in *Proc. 21st Century Military Commun. Archit. Technol. Inf. Superiority*, vol. 1, 2000, pp. 174–180.

[16] F. Guilloud, E. Boutillon, J. Tousch, and J.-L. Danger, "Generic description and synthesis of LDPC decoders," *IEEE Trans. Commun.*, vol. 55, no. 11, pp. 2084–2091, Nov. 2007.

[17] T. Brack *et al.*, "A survey on LDPC codes and decoders for OFDM-based UWB systems," in *Proc. IEEE 65th Veh. Technol. Conf.*, 2007, pp. 1549–1553.

[18] C. Roth, A. Cevrero, C. Studer, Y. Leblebici, and A. Burg, "Area, throughput, and energy-efficiency trade-offs in the VLSI implementation of LDPC decoders," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2011, pp. 1772–1775.

[19] N. Bonello, S. Chen, and L. Hanzo, "Low-density parity-check codes and their rateless relatives," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 1, pp. 3–26, 1st Quart., 2011.

[20] H. Chen, R. G. Maunder, and L. Hanzo, "A survey and tutorial on low-complexity turbo coding techniques and a holistic hybrid ARQ design example," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1546–1566, 4th Quart., 2013.

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

[22] E. Arikan, N. Ul Hassan, M. Lentmaier, G. Montorsi, and J. Sayir, "Challenges and some new directions in channel coding," *J. Commun. Netw.*, vol. 17, no. 4, pp. 328–338, Aug. 2015.

[23] J. Andrade, G. Falcao, V. Silva, and L. Sousa, "A survey on programmable LDPC decoders," *IEEE Access*, vol. 4, pp. 6704–6718, 2016.

[24] P. Hailes, L. Xu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "A survey of FPGA-based LDPC decoders," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1098–1122, 2nd Quart., 2016.

[25] M. F. Brejza, L. Li, R. G. Maunder, B. M. Al-Hashimi, C. Berrou, and L. Hanzo, "20 years of turbo coding and energy-aware design guidelines for energy-constrained wireless applications," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 8–28, 1st Quart., 2016.

[26] H. Mukhtar, A. Al-Dweik, and A. Shami, "Turbo product codes: Applications, challenges, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 3052–3069, 4th Quart., 2016.

[27] I. B. Djordjevic, "On advanced FEC and coded modulation for ultra-high-speed optical transmission," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1920–1951, 3rd Quart., 2016.

[28] H. B. Thameur, B. Le Gal, N. Khouja, F. Tlili, and C. Jego, "A survey on decoding schedules of LDPC convolutional codes and associated hardware architectures," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2017, pp. 898–905.

[29] Y. Fang, G. Han, G. Cai, F. C. M. Lau, P. Chen, and Y. L. Guan, "Design guidelines of low-density parity-check codes for magnetic recording systems," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1574–1606, 2nd Quart., 2018.

[30] S. Abdulghani, "Binary and non-binary low density parity check codes: A survey," *Int. J. Inf. Eng. Appl.*, vol. 1, no. 3, pp. 104–117, 2018.

[31] T. Tonnellier, A. Cavatassi, and W. J. Gross, "Length-compatible polar codes: A survey (invited paper)," in *Proc. 53rd Annu. Conf. Inf. Sci. Syst. (CISS)*, 2019, pp. 1–6.

[32] I. E. KAIME, A. A. MADI, and H. Erguig, "A survey of polar codes," in *Proc. 7th Mediterr. Congr. Telecommun. (CMT)*, 2019, pp. 1–7.

[33] Z. B. K. Egilmez, L. Xiang, R. G. Maunder, and L. Hanzo, "The development, operation and performance of the 5G polar codes," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 96–122, 1st Quart., 2020.

[34] V. Bioglio, C. Condo, and I. Land, "Design of polar codes in 5G new radio," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 29–40, 1st Quart., 2021.

[35] M. Arabaci, I. B. Djordjevic, R. Saunders, and R. Marcoccia, "A class of non-binary regular girth-8 LDPC codes for optical communication channels," in *Proc. Conf. Opt. Fiber Commun. Includes Post Deadline Papers*, 2009, pp. 1–3.

[36] M. Arabaci, I. B. Djordjevic, R. Saunders, and R. M. Marcoccia, "Rate-adaptive non-binary-LDPC-coded polarization-multiplexed multilevel modulation with coherent detection for optically-routed networks," in *Proc. 11th Int. Conf. Transparent Opt. Netw.*, 2009, pp. 1–4.

[37] C. Choi, H. Lee, N. Kaneda, and Y. Chen, "Concatenated non-binary LDPC and HD-FEC codes for 100Gb/s optical transport systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2012, pp. 1783–1786.

[38] X. Xiao, Y. Lu, and S. Goto, "A low-complexity coding scheme for non-binary LDPC code based on IDRB-MLGD algorithm," in *Proc. 9th Int. Conf. Inf. Commun. Signal Process.*, 2013, pp. 1–5.

[39] M. Arabaci, I. B. Djordjevic, R. Saunders, and R. M. Marcoccia, "Nonbinary quasi-cyclic LDPC-based coded modulation for beyond 100-Gb/s transmission," *IEEE Photon. Technol. Lett.*, vol. 22, no. 6, pp. 434–436, Mar. 15, 2010.

[40] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, "A survey on FEC codes for 100 G and beyond optical networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 209–221, 1st Quart., 2016.

[41] I. B. Djordjevic, "On the irregular nonbinary QC-LDPC-coded hybrid multidimensional OSCD-modulation enabling beyond 100 Tb/s optical transport," *J. Lightw. Technol.*, vol. 31, no. 16, pp. 2669–2675, Aug. 15, 2013.

[42] G. A. Al-Rubaye, C. C. Tsimenidis, and M. Johnston, "Non-binary LDPC coded OFDM in impulsive power line channels," in *Proc. 23rd Eur. Signal Process. Conf. (EUSIPCO)*, 2015, pp. 1431–1435.

[43] W. Abd-Alaziz, Z. Mei, M. Johnston, and S. Le Goff, "Non-binary turbo-coded OFDM-PLC system in the presence of impulsive noise," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, 2017, pp. 2576–2580.

[44] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1243–1274, 2nd Quart., 2019.

[45] A. Li, L. Xiang, T. Chen, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "VLSI implementation of fully parallel LTE turbo decoders," *IEEE Access*, vol. 4, pp. 323–346, 2016.

[46] P. Giard, G. Sarkis, C. Thibeault, and W. J. Gross, "Multi-mode unrolled architectures for polar decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 9, pp. 1443–1453, Sep. 2016.

[47] A. Naderi, S. Mannor, M. Sawan, and W. J. Gross, "Delayed stochastic decoding of LDPC codes," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5617–5626, Nov. 2011.

[48] J. Tian, J. Lin, and Z. Wang, "A 21.66 Gbps nonbinary LDPC decoder for high-speed communications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 2, pp. 226–230, Feb. 2018.

[49] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, 1996.

[50] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular $(2, d_c)$-LDPC codes over GF(q) using their binary images," *IEEE Trans. Commun.*, vol. 56, no. 10, pp. 1626–1635, Oct. 2008.

[51] S. Lin and D. J. Costello, *Error Control Coding*, vol. 2. Englewood Cliffs, NJ, USA: Prentice Hall, 2001.

[52] R. A. Carrasco and M. Johnston, *Non-Binary Error Control Coding for Wireless Communication and Data Storage*. Chichester, U.K.: Wiley, 2008.

[53] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

[54] A. Bazarsky, N. Presman, and S. Litsyn, "Design of non-binary quasi-cyclic LDPC codes by ACE optimization," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2013, pp. 1–5.

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

[56] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 53, no. 11, p. 1973, Nov. 2005.

[57] B. Zhou, L. Zhang, J. Kang, Q. Huang, S. Lin, and K. Abdel-Ghaffar, "Array dispersions of matrices and constructions of quasi-cyclic LDPC codes over non-binary fields," in *Proc. IEEE Int. Symp. Inf. Theory*, 2008, pp. 1158–1162.

[58] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over $GF(2^q)$," in *Proc. IEEE Inf. Theory Workshop*, 2003, pp. 70–73.

[59] G. L. Mullen and D. Panario, *Handbook of Finite Fields*. Boca Raton, FL, USA: CRC Press, 2013.

[60] A. A. Ghouwayel and E. Boutillon, "A systolic LLR generation architecture for non-binary LDPC decoders," *IEEE Commun. Lett.*, vol. 15, no. 8, pp. 851–853, Aug. 2011.

[61] R. Peng and R.-R. Chen, "Application of nonbinary LDPC cycle codes to MIMO channels," *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, pp. 2020–2026, Jun. 2008.

[62] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over $GF(q)$," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, 2004, pp. 772–776.

[63] C. Spagnol, E. M. Popovici, and W. P. Marnane, "Hardware implementation of GF(2^m) LDPC decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 12, pp. 2609–2620, Dec. 2009.

[64] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over $GF(q)$," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[65] V. Savin, "Min-max decoding for non binary LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2008, pp. 960–964.

[66] C. Chen, Q. Huang, C.-C. Chao, and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 11, pp. 3140–3147, Nov. 2010.

[67] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary LDPC codes and its hardware structure," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2600–2611, Jul. 2013.

[68] J. O. Lacruz, F. García-Herrero, D. Declercq, and J. Valls, "Simplified trellis min-max decoder architecture for nonbinary low-density parity-check codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1783–1792, Sep. 2015.

[69] K. M. Greenan, E. L. Miller, and S. J. T. J. E. Schwarz, "Optimizing galois field arithmetic for diverse processor architectures and applications," in *Proc. IEEE Int. Symp. Model. Anal. Simulat. Comput. Telecommun. Syst.*, 2008, pp. 1–10.

[70] X. Zhang, *VLSI Architectures for Modern Error-Correcting Codes*. Boca Raton, FL, USA: CRC Press, 2017.

[71] D. Declercq and M. Fossorier, "Extended minsum algorithm for decoding LDPC codes over *GF*(*q*)," in *Proc. Int. Symp. Inf. Theory*, 2005, pp. 464–468.

[72] X. Chen and C.-L. Wang, "High-throughput efficient non-binary LDPC decoder based on the simplified min-sum algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 11, pp. 2784–2794, Nov. 2012.

[73] X. Zhang and F. Cai, "Reduced-complexity decoder architecture for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 7, pp. 1229–1238, Jul. 2011.

[74] X. Zhang, F. Cai, and S. Lin, "Low-complexity reliability-based message-passing decoder architectures for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 11, pp. 1938–1950, Nov. 2012.

[75] F. Cai and X. Zhang, "Relaxed min-max decoder architectures for non-binary low-density parity-check codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 11, pp. 2010–2023, Nov. 2013.

[76] X. Zhang, "Low-complexity modified trellis-based min-max non-binary LDPC decoders," *J. Commun.*, vol. 10, no. 11, pp. 836–842, 2015.

[77] F. García-Herrero, E. Li, D. Declercq, and J. Valls, "Multiple-vote symbol-flipping decoder for nonbinary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 11, pp. 2256–2267, Nov. 2014.

[78] C.-W. Yang, X.-R. Lee, C.-L. Chen, H.-L. Chang, and C.-Y. Lee, "Area-efficient TFM-based stochastic decoder design for non-binary LDPC codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2014, pp. 409–412.

[79] M. Awais, G. Masera, M. Martina, and G. Montorsi, "VLSI implementation of a non-binary decoder based on the analog digital belief propagation," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3965–3975, Aug. 2014.

[80] L. Zhou, J. Sha, Y. Chen, C. Zhang, and Z. Wang, "Efficient symbol reliability based decoding for QCNB-LDPC codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2014, pp. 405–408.

[81] Q. Huang and S. Yuan, "Bit reliability-based decoders for non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 64, no. 1, pp. 38–48, Jan. 2016.

[82] J. Andrade, G. Falcao, V. Silva, and K. Kasai, "FFT-SPA non-binary LDPC decoding on GPU," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 5099–5103.

[83] Z. Liu, R. Liu, Y. Hou, H. Peng, and L. Zhao, "Efficient GPU-based implementation for decoding non-binary LDPC codes with layered and flooding schedules," *Concurrency Comput. Pract. Exp.*, vol. 30, no. 16, 2018, Art. no. e4442.

[84] Z. Liu, R. Liu, Y. Hou, and L. Zhao, "High-throughput multi-codeword decoder for non-binary LDPC codes on GPU," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 486–489, Mar. 2018.

[85] M. Beermann, E. Monzo, L. Schmalen, and P. Vary, "GPU accelerated belief propagation decoding of non-binary LDPC codes with parallel and sequential scheduling," *J. Signal Process. Syst.*, vol. 78, no. 1, pp. 21–34, 2015.

[86] G. Wang, H. Shen, B. Yin, M. Wu, Y. Sun, and J. R. Cavallaro, "Parallel nonbinary LDPC decoding on GPU," in *Proc. Conf. Rec. 46th Asilomar Conf. Signals Syst. Comput. (ASILOMAR)*, 2012, pp. 1277–1281.

[87] H. P. Thi, S. Ajaz, and H. Lee, "Efficient min-max nonbinary LDPC decoding on GPU," in *Proc. Int. SoC Design Conf. (ISOCC)*, 2014, pp. 266–267.

[88] H. T. Pham, S. Ajaz, and H. Lee, "Parallel block-layered nonbinary QC-LDPC decoding on GPU," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2015, pp. 1–6.

[89] O. Ferraz, S. Subramaniyan, G. Wang, J. R. Cavallaro, G. Falcao, and M. Purnaprajna, "Gbit/s non-binary LDPC decoders: High-throughput using high-level specifications," in *Proc. IEEE 28th Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM)*, 2020, p. 226.

[90] S. Subramaniyan *et al.*, "Pushing the limits of energy efficiency for non-binary LDPC decoders on GPUs and FPGAs," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2020, pp. 1–6.

[91] J. Andrade, G. Falcao, V. Silva, and K. Kasai, "Flexible non-binary LDPC decoding on FPGAs," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2014, pp. 1936–1940.

[92] J. Andrade *et al.*, "From low-architectural expertise up to high-throughput non-binary LDPC decoders: Optimization guidelines using high-level synthesis," in *Proc. 25th Int. Conf. Field Programmable Logic Appl. (FPL)*, 2015, pp. 1–8.

[93] C. Spagnol, W. Marnane, and E. Popovici, "FPGA implementations of LDPC over GF (2m) decoders," in *Proc. IEEE Workshop Signal Process. Syst.*, 2007, pp. 273–278.

[94] T. Lehnigk-Emden and N. Wehn, "Complexity evaluation of non-binary Galois field LDPC code decoders," in *Proc. 6th Int. Symp. Turbo Codes Iterative Inf. Process.*, 2010, pp. 53–57.

[95] W. Sulek, M. Kucharczyk, and G. Dziwoki, "GF(q) LDPC decoder design for FPGA implementation," in *Proc. IEEE 10th Consum. Commun. Netw. Conf. (CCNC)*, 2013, pp. 460–465.

[96] Y. Sun, Y. Zhang, J. Hu, and Z. Zhang, "FPGA implementation of nonbinary quasi-cyclic LDPC decoder based on EMS algorithm," in *Proc. Int. Conf. Commun. Circuits Systems*, 2009, pp. 1061–1065.

[97] Y. Tao, Y. S. Park, and Z. Zhang, "High-throughput architecture and implementation of regular (2, dc) nonbinary LDPC decoders," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2012, pp. 2625–2628.

[98] E. Boutillon, L. Conde-Canencia, and A. Al Ghouwayel, "Design of a GF(64)-LDPC decoder based on the EMS algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 10, pp. 2644–2656, Oct. 2013.

[99] X. Zhang and F. Cai, "Efficient partial-parallel decoder architecture for quasi-cyclic nonbinary LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 402–414, Feb. 2011.

[100] X. Chen, S. Lin, and V. Akella, "Efficient configurable decoder architecture for nonbinary quasi-cyclic LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 1, pp. 188–197, Jan. 2012.

[101] J. O. Lacruz, F. Garcia-Herrero, M. J. Canet, J. Valls, and A. Pérez-Pascual, "A 630 Mbps non-binary LDPC decoder for FPGA," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2015, pp. 1989–1992.

[102] F. García-Herrero, M. J. Canet, and J. Valls, "Decoder for an enhanced serial generalized bit flipping algorithm," in *Proc. 19th IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, 2012, pp. 412–415.

[103] F. Garcia-Herrero, M. J. Canet, and J. Valls, "Architecture of generalized bit-flipping decoding for high-rate non-binary LDPC codes," *Circuits Syst. Signal Process.*, vol. 32, no. 2, pp. 727–741, 2013.

[104] F. Garcia-Herrero, M. J. Canet, and J. Valls, "High-speed NB-LDPC decoder for wireless applications," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst.*, 2013, pp. 215–220.

[105] A. Ciobanu, S. Hemati, and W. J. Gross, "Adaptive multiset stochastic decoding of non-binary LDPC codes," *IEEE Trans. Signal Process.*, vol. 61, no. 16, pp. 4100–4113, Aug. 2013.

[106] Y.-L. Ueng, K.-H. Liao, H.-C. Chou, and C.-J. Yang, "A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using max-log-QSPA," *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 2940–2951, Jun. 2013.

[107] Y. S. Park, Y. Tao, and Z. Zhang, "A 1.15 Gb/s fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2013, pp. 422–423.

[108] Y. S. Park, Y. Tao, and Z. Zhang, "A fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating," *IEEE J. Solid-State Circuits*, vol. 50, no. 2, pp. 464–475, Feb. 2015.

[109] C.-L. Lin, S.-W. Tu, C.-L. Chen, H.-C. Chang, and C.-Y. Lee, "An efficient decoder architecture for nonbinary LDPC codes with extended min-sum algorithm," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 9, pp. 863–867, Sep. 2016.

[110] I. Choi and J.-H. Kim, "High-throughput non-binary LDPC decoder based on aggressive overlap scheduling," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 7, pp. 1937–1948, Jul. 2017.

[111] H. Harb, "Design of ultra high throughput rate NB-LDPC decoder," Ph.D. dissertation, Mathématiques Sci. Technol. Inf. Commun., Univ. Bretagne Sud, Lorient, France, 2018.

[112] C. Marchand, E. Boutillon, H. Harb, L. Conde-Canencia, and A. Al Ghouwayel, "Hybrid check node architectures for NB-LDPC decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 2, pp. 869–880, Feb. 2019.

[113] H. Harb, A. C. Al Ghouwayel, L. Conde-Canencia, C. Marchand, and E. Boutillon, "Ultra-high-throughput EMS NB-LDPC decoder with full-parallel node processing," to be published. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02494736

[114] J. Lin, J. Sha, Z. Wang, and L. Li, "Efficient decoder design for non-binary quasicyclic LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 5, pp. 1071–1082, May 2010.

[115] Y.-L. Ueng, C.-J. Yang, S.-W. Chen, and W.-X. Wu, "A selective-input non-binary LDPC decoder architecture," in *Proc. Int. SoC Design Conf.*, 2011, pp. 40–43.
[116] Y.-L. Ueng, C.-Y. Leong, C.-J. Yang, C.-C. Cheng, K.-H. Liao, and S.-W. Chen, "An efficient layered decoding architecture for nonbinary QC-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 2, pp. 385–398, Feb. 2012.
[117] J. Lin and Z. Yan, "Efficient shuffled decoder architecture for nonbinary quasi-cyclic LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 9, pp. 1756–1761, Sep. 2013.
[118] J. Lin and Z. Yan, "An efficient fully parallel decoder architecture for nonbinary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 12, pp. 2649–2660, Dec. 2014.
[119] Y. Lu, G. Tian, and S. Goto, "An efficient decoder architecture for cyclic non-binary LDPC codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2014, pp. 397–400.
[120] S. Song, H. Cui, J. Tian, J. Lin, and Z. Wang, "A novel iterative reliability-based majority-logic decoder for NB-LDPC codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*,vol. 67, no. 8, pp. 1399–1403, Aug. 2020.
[121] C. Xiong and Z. Yan, "Low-complexity layered iterative hard-reliability-based majority-logic decoder for non-binary quasi-cyclic LDPC codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2013, pp. 1348–1351.
[122] E. Li, F. García-Herrero, D. Declercq, K. Gunnam, J. O. Lacruz, and J. Valls, "Low latency T-EMS decoder for non-binary LDPC codes," in *Proc. Asilomar Conf. Signals Syst. Comput.*, 2013, pp. 831–835.
[123] J. O. Lacruz, F. García-Herrero, J. Valls, and D. Declercq, "One minimum only trellis decoder for non-binary low-density parity-check codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 1, pp. 177–184, Jan. 2015.
[124] J. O. Lacruz, F. García-Herrero, and J. Valls, "Reduction of complexity for nonbinary LDPC decoders with compressed messages," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2676–2679, Nov. 2015.
[125] J. O. Lacruz, F. Garcia-Herrero, M. J. Canet, and J. Valls, "High-performance NB-LDPC decoder with reduction of message exchange," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 5, pp. 1950–1961, May 2016.
[126] X. Zhang, "Modified trellis-based min-max decoder for non-binary LDPC codes," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, 2015, pp. 613–617.
[127] J. O. Lacruz, F. García-Herrero, M. J. Canet, and J. Valls, "Reduced-complexity nonbinary LDPC decoder for high-order Galois fields based on trellis min-max algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 8, pp. 2643–2653, Aug. 2016.
[128] H. P. Thi and H. Lee, "Two-extra-column trellis min-max decoder architecture for nonbinary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1787–1791, May 2017.
[129] H. P. Thi and H. Lee, "Reduced-complexity trellis min-max decoder for non-binary LDPC codes," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2018, pp. 1179–1183.
[130] H. P. Thi and H. Lee, "Basic-set trellis min-max decoder architecture for nonbinary LDPC codes with high-order galois fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 496–507, Mar. 2018.
[131] M.-R. Li, W.-X. Chu, H.-C. Lee, and Y.-L. Ueng, "An efficient high-rate non-binary LDPC decoder architecture with early termination," *IEEE Access*, vol. 7, pp. 20302–20315, 2019.
[132] H. P. Thi, H. Lee, and X. N. Pham, "Half-row modified two-extra-column trellis min-max decoder architecture for nonbinary LDPC codes," *Integration*, vol. 69, pp. 234–241, Nov. 2019.
[133] V. Rybalkin, P. Schlafer, and N. Wehn, "A new architecture for high speed, low latency NB-LDPC check node processing for GF(256)," in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC Spring)*, 2016, pp. 1–5.
[134] A. Haroun, R. Nasr, and A. Al-Ghouwayel, "On the implementation of vertical shuffle scheduling decoder for joint MIMO detection and channel decoding system," in *Proc. Int. Arab Conf. Inf. Technol. (ACIT)*, 2018, pp. 1–4.
[135] B.-Y. Chang, L. Dolecek, and D. Divsalar, "EXIT chart analysis and design of non-binary protograph-based LDPC codes," in *Proc. Military Commun. Conf.*, 2011, pp. 566–571.
[136] M. Helmling *et al.* "Database of Channel Codes and ML Simulation Results." 2019. [Online]. Available: www.uni-kl.de/channel-codes
[137] C. Marchand, H. Harb, T. Gendron, B. Orvoine, and E. Boutillon. "Free NB-LDPC Code Database of the Lab-STICC Laboratory." 2018. [Online]. Available: www-labsticc.univ-ubs.fr/nb_ldpc/
[138] D. Declerq. "Regular Ultra-Sparse Graphs and Related Nonbinary LDPC Codes." 2015. [Online]. Available: https://perso-etis.ensea.fr/declercq/graphs.php

Oscar Ferraz (Student Member, IEEE) received the M.Sc. degree from the University of Coimbra, Portugal, in 2019. He is currently pursuing the Ph.D. degree. He was a Visiting Researcher with Amrita Vishwa Vidyapeetham University, Bengaluru, India, and is currently a Researcher with the Instituto de Telecomunicações, Coimbra, Portugal. His research interests include parallel computer architectures, energy-efficient processing, and high-performance computing. He is a Student Member of the IEEE Signal Processing Society and IEEE Computer Society.

Srinivasan Subramaniyan received the bachelor's degree in electronics and communication engineering from Amrita Vishwa Vidyapeetham, Amritapuri, in August 2019. He was a Visiting Researcher with the Instituto de Telecomunicações Coimbra, Portugal, and was also a Junior Research Fellow of the Computer Architecture and High-Performance Lab, Amrita Vishwa Vidyapeetham, Bengaluru. His research interests include reconfigurable computing, signal Processing, and VLSI system design.

Ramesh Chinthala received the M.Tech. degree in VLSI from the Indian Institute of Technology Guwahati, Guwahati, India, and the Ph.D. degree from the Indian Institute of Science Bengaluru, Bengaluru, India. He is currently serves as an Assistant Professor (Sr. Gr.) with the Department of Electronics and Communication Engineering, School of Engineering, Amrita Vishwa Vidyapeetham, Bengaluru. His research interest includes high-performance computing architectures, FPGA-based Accelerators, FPGA-based embedded systems, hardware-software co-deign on FPGA platforms, parallel architectures for decoding algorithms like non-binary low-density parity check decoders, and hardware accelerators for machine learning algorithms.

João Andrade received the M.Sc. and Ph.D. degree in electrical and computer engineering from the University of Coimbra in 2010 and 2016. He was a Researcher with the Instituto de Telecomunicações from 2010 to 2016, and has since joined Synopsys, where he has worked as a Verification Engineer to HDMI controller IPs, and currently as an Applications Engineer to MIPI digital and PHY IPs. His research interests include error-correcting codes, computer architectures, and unreliable memory systems.

Joseph R. Cavallaro (Fellow, IEEE) received the B.S. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 1981, the M.S. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 1982, and the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY, USA, in 1988. In 1988, he joined the faculty of Rice University, Houston, TX, USA, where he is currently a Professor of Electrical and Computer Engineering and an Associate Chair. His research interests include computer arithmetic, and DSP, GPU, FPGA, and VLSI architectures for applications in wireless communications. He is an Advisory Board Member of the IEEE SPS TC on Design and Implementation of Signal Processing Systems (ASPS) and the Past Chair of the IEEE CAS TC on Circuits and Systems for Communications. He currently serves as an Associate Editor for the *Journal of Signal Processing Systems* and a Senior Editor for the IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS.

Soumitra K. Nandy (Senior Member, IEEE) received the B.Sc. (Hons.) degree in physics from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1977, the B.E. (Hons.) degree in electronics and communication in 1980, the M.Sc. (Engg.) degree in computer science and engineering in 1986, and the Ph.D. degree in computer science and engineering in 1989 from the Indian Institute of Science Bengaluru, Bengaluru, where he is a Professor with the Department of Computational and Data Sciences. His research interests are in areas of high-performance embedded systems on a chip, VLSI architectures for reconfigurable systems on chip, and architectures and compiling techniques for heterogeneous many core systems. He has over 170 publications in international journals, and proceedings of international conferences, and five patents.

Vitor Silva received the Graduation Diploma degree in electrical engineering and the Ph.D. degree from the University of Coimbra, Portugal, in 1984 and 1996, respectively. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Coimbra, where he lectures digital signal processing, and information and coding theory. He is currently the Director of the Instituto de Telecomunicações, Coimbra, coordinating the research activities of 40 collaborators. His research activities include signal processing, image and video compression, and coding theory.

Xinmiao Zhang (Senior Member, IEEE) received the Ph.D. degree from the University of Minnesota. She joined The Ohio State University as an Associate Professor in 2017. She was a Senior Technologist with Western Digital/SanDisk Corporation from 2013 to 2017 and was a Timothy E. and Allison L. Schroeder Associate Professor with Case Western Reserve University prior to that. She published close to 100 papers on error-correcting en/decoder design and a book "VLSI Architectures for Modern Error-Correcting Codes" (CRC Press, 2015). Her research spans the areas of VLSI architecture design, digital storage and communications, security, and signal processing. She is a recipient of the NSF CAREER Award in 2009. She is currently a member of the Board-of-Governors of the IEEE Circuits and Systems Society and the Chair of the Data Storage Technical Committee of the Communications Society. She also served on many conference organization and technical committees, and is currently an Associate Editor for IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS.

Madhura Purnaprajna received the master's degree in electrical and computer engineering from the University of Alberta, Canada, in January 2005, and the Ph.D. degree in electrical engineering from the Heinz Nixdorf Institute, University of Paderborn, Germany, in December 2009. She was a Postdoctoral Fellow with an International Research Fellowship from the German Research Foundation (Deutsche Forschungsgemenischaft) and MHV Fellowship (SNSF), at the Processor Architecture Lab, EPFL, Switzerland, and the High-performance Computing Lab, Indian Institute of Science Bengaluru, Bengaluru. She currently serves as an Associate Professor with the Department of Computer Science, Amrita Vishwa Vidyapeetham, Bengaluru. Her research interests are in reconfigurable computing and processor architectures.

Gabriel Falcao (Senior Member, IEEE) received the Ph.D. degree from the University of Coimbra in 2010, where he is currently a Tenured Assistant Professor with the Department of Electrical and Computer Engineering. He is a Researcher with the Instituto de Telecomunicações and his research activities include parallel computer architecture, GPU- and FPGA-based accelerators, and energy-efficient processing for compute-intensive signal processing applications, namely, in digital communications and imaging, where he published more than 100 papers. He was a recipient of a Google Faculty Research Award in 2013/14 (together with J. Barreto). From 2011 to 2017, he was a Visiting Professor with EPFL, Switzerland, and in the summer of 2018 he was a Visiting Academic with ETHZ, also in Switzerland. He was a General Co-Chair of the IEEE SiPS in 2020 and a Local Chair of Euro-Par in 2021. He is a member of the IEEE Signal Processing Society of the IEEE TRANSACTIONS ON COMPUTERS on Design and Implementation of Signal Processing Systems (ASPS) and a Full Member of the HiPEAC Network of Excellence.