Computer Engineering and Applications Vol. 9, No. 3, October 2020

An Improved Throughput for Non-Binary Low-Density-Parity-Check Decoder

Omowuyi Olajide*, Bashir Abdulrazaq, Adewale Adedokun, Ime Umoh, Akan Bello

Department of Computer Engineering, Ahmadu Bello University, Zaria, Nigeria
** omowuyi@gmail.com*

ABSTRACT

Low-Density-Parity-Check (LDPC) based error control decoders find wide range of application in both storage and communication systems, because of the merits they possess which include high appropriateness towards parallelization and excellent performance in error correction. Field-Programmable Gate Array (FPGA) has provided a robust platform in terms of parallelism, resource allocation and excellent performing speed for implementing non-binary LDPC decoder architectures. This paper proposes, a high throughput LDPC decoder through the implementation of fully parallel architecture and a reduction in the maximum iteration limit, needed for complete error correction. A Galois field of eight was utilized alongside a non-uniform quantization scheme, resulting in fewer bits per Log Likelihood Ratio (LLR) for the implementation. Verilog Hardware Description Language (HDL) was used in the description of the non-binary error control decoder. The propose decoder attained a throughput of 10Gbps at 400-MHz clock frequency at 10 iterations when synthesized on a ZYNQ 7000 Series FPGA. The decoder's percentage utilization of the Look Up Table (LUT), Register (FF), and Latch are 96.00, 14.00 and 7.00 percent respectively.

Keywords: Non-binary low-density parity-check (NB-LDPC) codes, Field-Programmable Gate Array (FPGA), error correction, Variable Node (VN), Check Node (CN), Log Likelihood Ratio (LLR), iterative decoding.

1. INTRODUCTION

A unique type of block codes called Low Density Parity Check (LDPC) codes have found extensive application in storage architectures and communication channels. Within a communication channel, they are used primarily to correct errors transmitted in the system. With respect to storage architectures, they are implemented to ensure data recovery and storage reliability. LDPC codes have enjoyed much attention from the research community. Gallager in 1962, was the first to propose them [1], but as a result of their very high complexity, they were regarded impractical for implementation. This resulted in them being unused for several years. As a result, little or nothing could be achieved in avenues requiring specific implementation of the code. But these all changed in 1996, when LDPC codes were discovered again, this time by Mackay and Neil [2]. This was as a result of their superior performance and capability, and perhaps also because of the license fee resulting from the patenting of turbo codes. Since that time, LDPC codes have experienced a reawakening in various applications [3].

Given the development and advancement in computation power and efficiency today, LDPC codes have become an integral part in diverse standardized and commercial communication infrastructures. They have important characteristics that make them most suitable for implementation in various systems. LDPC codes also work with certain decoding algorithms [4]. These algorithms are implemented using reduced degree of complexity. This complexity is measured in the number of calculations and the length of time of decoding. Algorithms with low complexity will often result in a simple architecture and as a result, a reduced cost of implementation [5]. This will also translate into lower hardware resources that will be needed to implement such an architecture. The amount of hardware resources used in the implementation of algorithms contribute significantly in determining the effectiveness of the architecture. LDPC codes have a feature of iterative decoding, unlike the turbo codes. This makes it possible for the codes to attain excellent performance in error correction. Such performance can attain the limit theoretically established called the Shannon limit. This is achieved when messages with large block lengths are decoded [6].

LDPC decoders have various decoding algorithms that can be implemented to achieve specified design. A very important factor that also comes to play when decoding algorithms are being implemented is what is called the degree of parallelism. This is the number of nodes allowed to be processed at the same time. Designers are therefore provided with various techniques to attain the desired architecture [7]. Furthermore, an LDPC decoder consists of individual elements being processed together. An interworking of a number of architectural features constitute a complete implementation of an LDPC decoder in its entirety. They include, hardware resource utilization, processing throughput and error correction performance [4]. These features rely on certain architectural parameters, namely the type of architecture, feature of the LDPC code used, number of iterations and the algorithm employed. This structure is shown in Figure 1. These features depend on capability of the LDPC code to correct errors. The error correction capability of an LDPC code plays a critical role in the functionality of the decoder. It can be improved through various means, one of which is the alteration in the decoding iteration limit. A more powerful LDPC code would require fewer iterations for error correction [8].

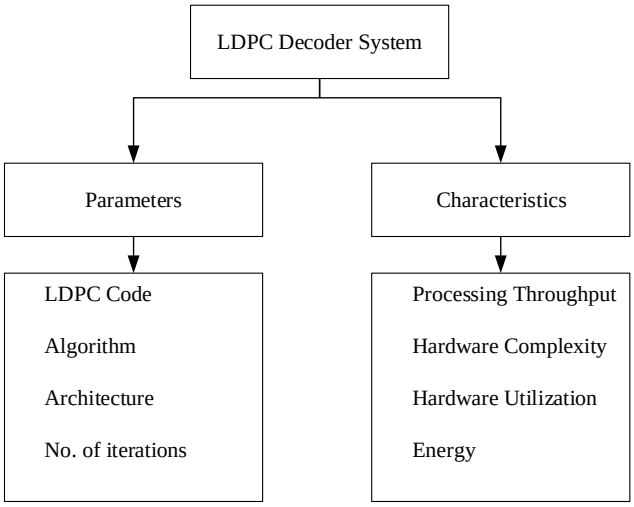

FIGURE 1. FPGA-based LDPC decoder system parameters and features

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

The number of iterations can also be increased to allow for errors present to be corrected. But this will have another effect on the decoder. As the number of iterations is increased, the degree of complexity is also increased. As a result, energy efficiency is compromised. The design of the decoder architecture involves a holistic view of all parameters so as to ensure optimal effectiveness and efficiency [7].

As a result of the demand for high throughput in error control decoders, recent research works [8] [9] [10], have focused on using large Galois fields (for example GF(256), GF(64) and GF(32)), for the representation of the processing nodes while only a subset of that Galois field is used to implement the intermediate messages passed between the nodes during processing. This approach results in the reduction in the hardware complexity, but does not necessarily increase the throughput of the decoder [11]. The implementation of partial parallelization in the decoder architecture is another technique adopted to increase the throughput. But, the maximum throughput that can be attained is limited because of the layered processing of the nodes. Though this technique slightly improves the throughput, higher values are achievable through a fully parallel design for LDPC codes [12]. The downside to this technique, is the large increase in the interconnect routing wires which eventually leads to a very complex decoder structure.

Hence motivated by the above literatures, a decoder that achieves higher throughput through the use of a small Galois field, and a fully parallel architecture is presented. The high decoder complexity that results from the full parallelization is lowered by the implementation of the small Galois field of eight. The proposed decoder also achieves an error correction capability at a reduced number of decoding iterations.

The rest of the paper is structured as follows, Section I covers the introduction of Low Density Parity Check Codes, section II discusses the materials and methods of code design and implementation, section III covers the results and discussion of decoder implementation, section IV covers the conclusion and section V lists the references.

2. MATERIAL AND METHODS

The LDPC code design comprises of two part which includes, the LDPC code characteristic design and the Tanner graph representation.

2.1 LDPC CODE CHARACTERISTIC DESIGN

A very important property of the LDPC code is its sparse parity check matrix. This property comes to play when decoding algorithms are implemented during error correction. The sparsity of the matrix involves distribution of both non-zero and zero elements, such that the zero elements occupy a majority of the slots. Important parameters that describe a parity check matrix include the word length, the parity bit number and the dimension. The non-zero elements that occupy a row constitute its row weight, while the non-zero elements that occupy the column determines the column weight. LDPC codes of our method are classified into two

categories based on constituent individual element position and weight:

i. A scenario where the column weight and the row weight of the parity check matrix are equal for each column and row. This is termed a regular LDPC code.
ii. A scenario where the column and row weights differ for each column and row. This is called an irregular LDPC code.

The parity check matrix is constructed by using a masked base matrix. The masked base matrix (B) is given in Equation (1). This matrix, B is a regular matrix. From the first row to the last row of B, it is seen there are non-zero elements occupying some edges, while other positions contain zero element The number of non-zero elements in B is equal for all rows. Likewise, the columns have an equal number of non-zero elements from the first to the last. There is constant column weight and constant row weight. The parity check matrix is built by first allocating the weight of the columns and rows, and then randomly placing the non-zero elements in their appropriate location. Definitely, the LDPC code should not be dense, having much less non-zero elements than zeros. The masked base matrix is the foundational structure for the construction of the LDPC code. It is constructed by concatenating many smaller matrices of the same size.

$$B_q = \begin{bmatrix} 0 & 0 & 6 & 0 & 8 & 0 & 10 & 0 & 0 & 0 & 0 & 15 & 0 & 17 & 0 & 19 & 0 & 0 & 0 & 0 & 51 & 0 & 55 & 0 & 58 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11 & 0 & 16 & 0 & 19 & 0 & 0 & 0 & 0 & 35 & 0 & 39 & 0 & 43 & 0 & 0 & 0 & 0 & 54 & 0 & 57 & 0 & 60 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 13 & 0 & 18 & 0 & 23 & 0 & 0 & 0 & 0 & 37 & 0 & 43 & 0 & 46 & 0 & 0 & 0 & 0 & 57 & 0 & 60 & 0 & 62 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 18 & 0 & 21 & 0 & 27 & 0 & 0 & 0 & 0 & 41 & 0 & 45 & 0 & 49 & 0 & 0 & 0 & 0 & 59 & 0 & 62 & 0 & 64 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 20 & 0 & 25 & 0 & 34 & 0 & 0 & 0 & 0 & 45 & 0 & 48 & 0 & 52 & 0 & 0 & 0 & 0 & 62 & 0 & 64 & 0 & 66 \\ 9 & 0 & 14 & 0 & 0 & 0 & 0 & 23 & 0 & 29 & 0 & 37 & 0 & 0 & 0 & 0 & 47 & 0 & 51 & 0 & 56 & 0 & 0 & 0 & 0 & 64 & 0 & 66 & 0 \\ 0 & 12 & 0 & 16 & 0 & 0 & 0 & 0 & 27 & 0 & 36 & 0 & 40 & 0 & 0 & 0 & 0 & 50 & 0 & 54 & 0 & 59 & 0 & 0 & 0 & 0 & 66 & 0 & 68 \\ 11 & 0 & 16 & 0 & 18 & 0 & 0 & 0 & 0 & 31 & 0 & 39 & 0 & 42 & 0 & 0 & 0 & 0 & 53 & 0 & 58 & 0 & 62 & 0 & 0 & 0 & 0 & 68 & 0 \\ 0 & 14 & 0 & 18 & 0 & 30 & 0 & 0 & 0 & 0 & 38 & 0 & 42 & 0 & 46 & 0 & 0 & 0 & 0 & 56 & 0 & 61 & 0 & 64 & 0 & 0 & 0 & 0 & 70 \end{bmatrix} \quad (1)$$

To get the parity check matrix from the masked base matrix, each non-zero element of B is replaced by a square matrix over Galois field of eight, called circular permutation matrix (CPM), and each zero element of B is replaced by a square matrix of zero element. The CPM is illustrated in equation (2) while the zero matrix is illustrated in equation (3). The null space of the parity check matrix gives the non-binary LDPC code.

$$CPM = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 4 & 5 & 6 & 7 & 1 \\ 3 & 4 & 5 & 6 & 7 & 1 & 2 \\ 4 & 5 & 6 & 7 & 1 & 2 & 3 \\ 5 & 6 & 7 & 1 & 2 & 3 & 4 \\ 6 & 7 & 1 & 3 & 3 & 4 & 5 \\ 7 & 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \quad (2)$$

$$ZM = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3)$$

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

2.2 TANNER GRAPH REPRESENTATION

The LDPC code is depicted using a bipartite graph. This is also called a Tanner graph. The tanner graph gives the pictorial view with important information about the degree of the check and bit nodes. The rows that make up the parity check matrix illustrate the parity check equation and each column illustrate a code-word bit. Taking the matrix in Equation (1), the top row of B gives the equation of parity check. The bipartite graph shows the connections between the code-word bits and the parity check equation. If a non-zero element occupies a node, then that node represents a connection between the equation of parity check and the corresponding code-word bit. A section of the Tanner graph for the LDPC code is shown in Figure 2. The complete parity check matrix cannot be shown because of its enormous size. When a path in the parity check matrix starts with a code-word bit and ends with that same code-word bit, that trajectory is forms a cycle.

A bipartite graph can have a cycle of smallest length called the girth. Girth plays a very important role during error correction. A value of 4 is established from literature to cause degradation in the error correction performance of LDPC codes. As a result, a minimum girth size of six is recommended when designing LDPC codes, and is thus used in constructing the non-binary LDPC code. Figure 2 presents an illustration of a section of the tanner graph utilized in the proposed decoder.

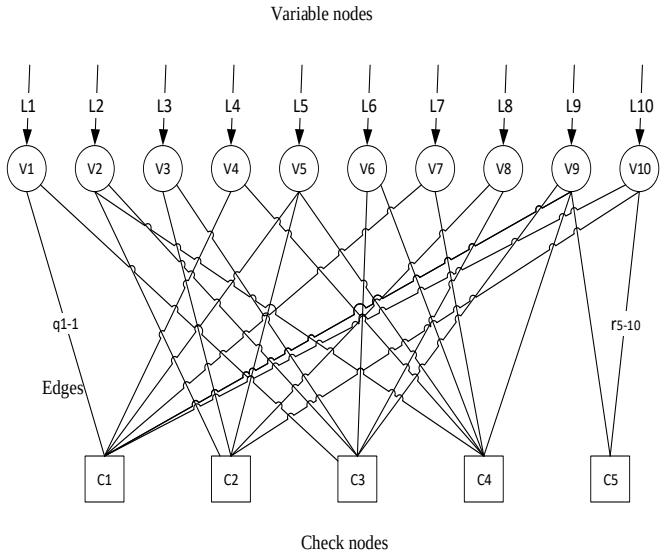

FIGURE 2. Tanner graph for example LDPC code

The connections L_i above each VN in Figure 2 pertain to LLRs associated with the N code-word bits of $\boldsymbol{c}$. When there is a non-zero element present in the parity check matrix, it creates an edge. An edge links the ith variable node, VN v_i to the jth check node CN c_j. An edge represents each element of $H_{ij} = 1$. A node's (VN/CN) degree is stated as the number of other nodes (VN/CN) that it is linked to. This parameter is also equivalent to the respective row/column weight in *the parity check matrix.* The check node degree as well as the variable node degree are vital parameters used in the construction of LDPC code. In a situation where all the

variable nodes and all the check nodes have the same degree, then a regular LDPC code is created. The tanner graph of the LDPC code used for the proposed decoder could not be shown because of the large number of interconnect wires and the processing nodes. In the row-column processing, all rows of one block column of the parity check matrix are processed at a time.

2.3 DECODER IMPLEMENTATION

The practical implementation of the decoder hardware design is determined by its architectural considerations. Architectural decisions influence the physical implementation and hardware used by the decoder. The proposed decoder implementation takes into consideration the architectural parameters LLR representation, clock frequency, degree of parallelization and number of decoding iterations. The first parameter for consideration is the degree of parallelization. A fully parallel architecture was synthesized, as shown in Figure 3. In this architecture, one computation unit is implemented for individual column and individual row of the parity check matrix. The processing of all rows and columns is carried out simultaneously, and hence fully-parallel decoders can achieve the highest throughput. Since there is one dedicated unit for each check node and each variable node, the check node units (CNUs) and variable node units (VNUs) are connected by hard wires to pass the messages. The V2C permutation in Figure 3 is the routing networks that cause the switching of the v2c messages to the appropriate CNUs. Likewise, the C2V inverse permutation represents the reverse routing links that are needed to direct the c2v messages to the correct VNUs. The number of routing networks required is determined by the number of CNUs and VNUs instantiated.

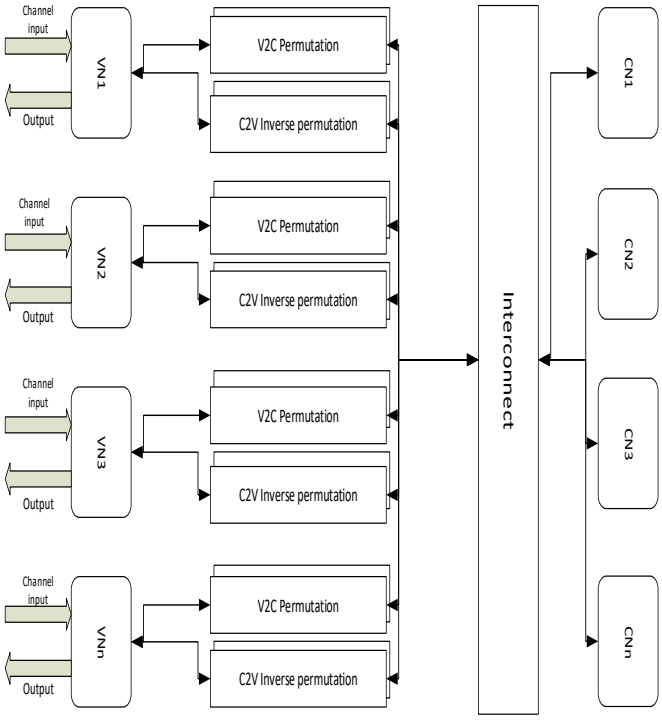

FIGURE 3. Architecture of the fully parallel non-binary LDPC (NB-LDPC) decoder.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

The reduced number of decoding iterations restriction implemented in the decoder played an important role in determining the throughput and error correction performance of the decoder. A stopping criterion is put in place to prevent the decoder from continuous decoding in case it encounters uncorrectable errors.

3. RESULTS AND DISCUSSION

The entire decoder architecture based on the variable node unit and the check node unit was synthesized and implemented on the ZYNQ FPGA. Verilog HDL was used in describing the hardware architecture. Xilinx Vivado SoC Design suite was used for the synthesis and implementation of the decoder architecture. Table 1 illustrates the result of the synthesis and implementation of the NB-LDPC decoder.

TABLE 1.
Output of the NB-LDPC decoder

Parameter	Value
Technology	28nm
Quantization	3 bits
Clock Frequency (MHz)	400
No. of iterations	10
Coded Throughput (Gbps)	10

The throughput of an error control decoder is perhaps its most important feature. It is defined as the total number of bits that is processed per second. Very high processing throughputs provide avenues for the implementation of very high speed data transfers. The throughput of the decoder is calculated as follows [8]:

$$T = \frac{f_{clk} \times N \times p}{I_{max} \times (M + d_v \times D) + (q-1)} Mbps \quad (4)$$

where f_{clk} denotes maximum clock frequency, I_{max} the maximum number of iterations N, the total number of variable node units, M, the total number of check node units p, the parallelism factor, d_v, the variable node degree D, the pipeline stages used in the design and q, the Galois field.

TABLE 2.
Summary of utilization report of resource usage

Resource	Utilization	Performance (%)
LUT	49763	96.00
Register (FF)	14598	14.00
Register (Latch)	7296	7.00

The Utilization Report breaks down the design utilization based on resource type of the logic elements contained in the FPGA. The percentage usage of the LUT, Register (FF), and Latch are 96.00, 14.00 and 7.00 percent respectively. Table 2

gives the available resource provided by the FPGA, along with the number of logic resource utilized. Various state of the art error control decoders are compared with the proposed decoder to illustrate the excellent performance of the proposed decoder. Table 3 gives the comparison of the parameter and results of various state of the art non-binary decoders. Empty slots are due to unavailable information from the authors of the work.

TABLE 3.
Performance Comparison of NB-LDPC Decoders

	TCAS II 2015 [14]	JSSC 2015 [13]	TVLSI 2014 [15]	IEEE 2018 [16]	IEEE 2019 [8]	This Decoder
Code	(168,84)	(160,80)	--	(327)	(837,726)	(576,288)
Galois Field	GF (16)	GF (64)	GF (256)	GF (8)	GF (32)	GF (8)
Algorithm	CTFM	EMS	RTBCP	A-IHRB & MM	TMM	MM
No. of Iterations	10	10	--	--	4	10
Quantization (bits)	5	5	5	3(log)	6	3
Technology	90nm	65nm	28nm	40nm	90nm	28nm
Utilization (%)	--	87	75.7	89.5	--	96
Frequency (MHz)	286	700	520	120	526.32	400
Power (mW)	--	3704	976	212.4	--	43244
Throughput (Gbps)	1.131	1.221	0.436	2.267	4.681	10.010

The relaxed half-stochastic non-binary LDPC decoder presented in [14] has a feature of (168,84), GF (16) code developed in 90nm CMOS. It focuses on lower algorithm complexity and achieved a 1.13Gbps throughput at 286MHz frequency. The fully parallel decoder of [13] implements a (160, 80), non-binary code in 65 nm CMOS. This decoder has a fully parallel nature and achieved a throughput of 1.22Gbps at 700MHz frequency. A technique of clock gating and node successive allocation were implemented to achieve low power and throughput. The decoder in [15] used a Relaxed Trellis Based Check Processing (RTBCP) algorithm to lower the complexity in the computation of the check node process. This was achieved with a fully parallel architecture in 28nm CMOS. A throughput of 546Mbps was attained using a GF (256).

The non-binary LDPC decoder implemented in [16] is unique in that it incorporated two decoding algorithms namely the Iterative Hard reliability based algorithm and the Min-Max algorithm. A quantization scheme that is logarithmic in nature was also used to reduce the message bit size and lower the hardware complexity. A throughput of 2.267Gbps and 212.4mW was attained when implemented in 40nm CMOS. The Trellis Min–Max (T-MM) decoder in [8] implements an early termination technique that quickly lowers the decoding

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

iterations. This is achieved with appreciable error control and throughput. A throughput of 4.68 Gbps was achieved at 700MHz frequency using 90nm CMOS technology.

The non-binary decoder presented in this work was synthesized on the ZYNQ 7000 FPGA. This parallel decoder architecture processes all the nodes in the scheme. It consumes a lot of power as a result of the large amount of logic resource utilized. This decoder design synthesized on the FPGA achieves a throughput of 10Gbps, and consumed a total power of 43.244W. This throughput is desirable for enterprise cloud applications and the 5G network paradigm.

4. CONCLUSION

The proposed decoder implements a fully-parallel architecture and as a result has very high hardware resource requirements and complexity. The feature of the instantiation of more parallel processors enable the FPGA-based LDPC decoder architecture to decode more bits per second. Row-column processing is used, which aids in decreasing the processing time of each iteration. It also improves the throughput by utilizing a Galois field of eight. The decoder implements a full instantiation of the processing nodes and decoder termination to achieve a very high throughput of 10 Gb/s at 400 MHz, consuming a total power of 43.2W. A time-critical applicable throughput is demonstrated the decoder.

REFERENCES

[1] R. G. Gallager, "Low-Density Codes *," *IRE Trans. Inf. Theory*, vol. IT-8, pp. 21–28, 1962.

[2] R. Neal and D. Mackay, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. **32**, no. August, pp. 1645–1646, 1996.

[3] K. Zhao, Z. Wenzhe, S. Hongbin, T. Zhang, Z. Xiaodong, and N. Zheng, "LDPC-in-SSD: making advanced error correction codes work effectively in solid state drives," *Proc. 11th USENIX Conf. File Storage Technol.*, pp. 243–256, 2013.

[4] O. Lacruz, F. Garc, D. Declercq, S. Member, and J. V. Member, "Simplified Trellis Min-Max Decoder Architecture," *IEEE Trans. VERY LARGE SCALE Integr. Syst.*, no. 32, pp. 1–10, 2015.

[5] S. Cho, K. Cheun, and K. Yang, "Design of nonbinary LDPC codes based on message-passing algorithms," *IEEE Trans. Commun.*, vol. 66, no. 11, pp. 5028–5040, 2018.

[6] S. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the Design of Low-Density Parity-Check Codes within 0 . 0045 dB of the Shannon Limit," vol. **5**, no. 2, pp. 58–60, 2001.

[7] G. S. Geeta G Gunari, "FPGA Implementation of GF (q) LDPC Encoder and Decoder Using MD Algorithm," *Int. J. Electr. Electron. Comput. Syst.*, no. 3, pp. 26–31, 2014.

[8] 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.
[9] J. O. Lacruz, F. García-Herrero, M. J. Canet, and J. Valls, "High-Performance NB-LDPC Decoder With Reduction of Message Exchange," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 5, pp. 1950–1961, 2016.
[10] J. 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. Syst.*, vol. 24, no. 8, pp. 2643–2653, 2016.
[11] 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 Regul. Pap.*, vol. 66, no. 2, pp. 869–880, 2019.
[12] Y. Yu, W. Chen, J. Li, X. Ma, and B. Bai, "Binary Representaion for Non-binary LDPC Code with Decoder Design," *IEEE Trans. Commun.*, pp. 1–12, 2020.
[13] 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, 2015.
[14] X. R. Lee, C. W. Yang, C. L. Chen, H. C. Chang, and C. Y. Lee, "An area-efficient relaxed half-stochastic decoding architecture for nonbinary LDPC codes," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 62, no. 3, pp. 301–305, 2015.
[15] J. Lin and Z. Yan, "An efficient fully parallel decoder architecture for nonbinary LDPC codes," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 12, pp. 2649–2660, 2014.
[16] Y. Toriyama and D. Markovic, "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, 2018.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)