Optimizing LDPC Decoder Implementation on FPGA for Improved Wireless Communication Efficiency

Maryam Imad Subhi[1*], Qusay Al-Doori[2]

[1,2]Control and Systems Engineering Department, University of Technology, Baghdad 10066, Iraq
Email: cse.21.09@grad.uotechnology.edu.iq
*Corresponding Author

Received: 14.08.2024 Revised: 18.09.2024 Accepted: 01.10.2024

ABSTRACT
Low-density parity check (LDPC) decoders face challenges such as high computational complexity and limited scalability. This research explores the optimal implementation of an LDPC decoder on a Field Programmable Gate Array (FPGA) to improve communication efficiency. The decoder is implemented using the min sum algorithm in two approaches. The study highlights notable disparities in gate efficiency, pin count, and thermal dissipation metrics. The direct VHDL approach requires considerably fewer logic gates, at a reduction of 37.13%, and exhibits a minor improvement in core static thermal power dissipation, indicating a more streamlined design that might contribute to smaller and more economical FPGA designs. Conversely, the MATLAB to VHDL conversion is more effective in minimizing pin usage, with a 52.19% decrease, which could simplify design complexity and enhance scalability. Nonetheless, the direct VHDL method incurs higher input/output thermal power dissipation, marked by a 27.39% increase, which introduces additional considerations for thermal management in system design. Application requirements, available resources, and the target trade-off between design complexity and resource efficiency all play a role in the method selection process.

Keywords: FPGA, implementation, LDPC decoder, MSA, synthesis, VHDL

1. INTRODUCTION

Digital communication systems are widely used to transmit data through communication channels using digital signals. These systems use various encoding and decoding techniques to ensure accurate and reliable data transmission. The Low-Density Parity Check Code is one method that can be utilized to detect and correct errors that may be present in digital communications transmission networks. LDPC codes are widely utilized in contemporary communication systems due to the fact that they offer near-optimal performance and a minimal level of complexity [1].

Robert Gallagher's 1963 PhD thesis introduced LDPC linear error correcting codes [2]. However, it wasn't until the 1990s that LDPC codes gained significant attention due to their superior performance and low complexity. David Mackay and colleagues demonstrated actual LDPC coding applications. It is common practice to make use of LDPC codes in modern communication systems because of the efficient decoding algorithms and excellent error correction capability that they possess Therefore LDPC has become popular and is utilized by numerous standards for many applications [3]. Many popular communication standards make use of LDPC, including Wi-Fi [4], DVB-S2, IEEE 802.11n, 802.3an, 802.16a, and WiMAX [5, 6].

The parity check matrix is necessary for LDPC encoding and decoding. Iteratively generates code words and corrects errors during encoding and decoding. Tanner graphs, which demonstrate LDPC codes, assist in explaining the parity check matrix, code structure, and interconnections [7-9].

Hard decision and soft decision are the main LDPC decoding methods [10]. After analyzing the incoming data and encoded bits, the decoder chooses one of these approaches. Decoding LDPC codes initially used belief propagation (BP) methods, but the check node and variable node processors' complicated operations increased computational complexity. The complexity was lowered by implementing the Min-Sum (MS) algorithm at check nodes. It is possible to reduce the amount of computational effort required and simplify computations by carrying out basic arithmetic operations. The Min-Sum technique was favored due to its computational simplicity and decoding performance [11, 12].

Due to their near-capacity performance, many prominent message-passing algorithms decode LDPC codes. Variations of message-passing technology are used by LDPC decoders to optimize calculations.

Offset MS (OMS) and Normalized MS are common Min-Sum (MS) methods. Both methods effectively handle the computational load while providing good decoding performance [13, 14].
Despite their widespread use and effectiveness, LDPC decoders face scalability, flexibility, and high computational demands. These issues often result in higher power consumption and processing delays, particularly with emerging standards like 5G. To enhance processing speed and decoding efficiency while minimizing computational complexity and power usage, advancements in design and implementation strategies are necessary.
This paper's main focus is comparing two methods of implementing LDPC. The first is converting the min-sum algorithm from MATLAB to VHDL, and the second is the direct implementation in VHDL .The comparison is directed towards the efficiency of the conversion method, reducing computational demands, and optimizing processing speeds for LDPC decoders on FPGA. The min-sum algorithm was chosen for its simplified calculation procedure and decreased resource usage, rendering it an appropriate selection for assessing the effectiveness of the suggested implementations. ModelSim simulations confirmed the decoder functionality, and synthesis was carried out on an Altera Cyclone IVE FPGA using Quartus. The methodology is outlined in Figure 1.

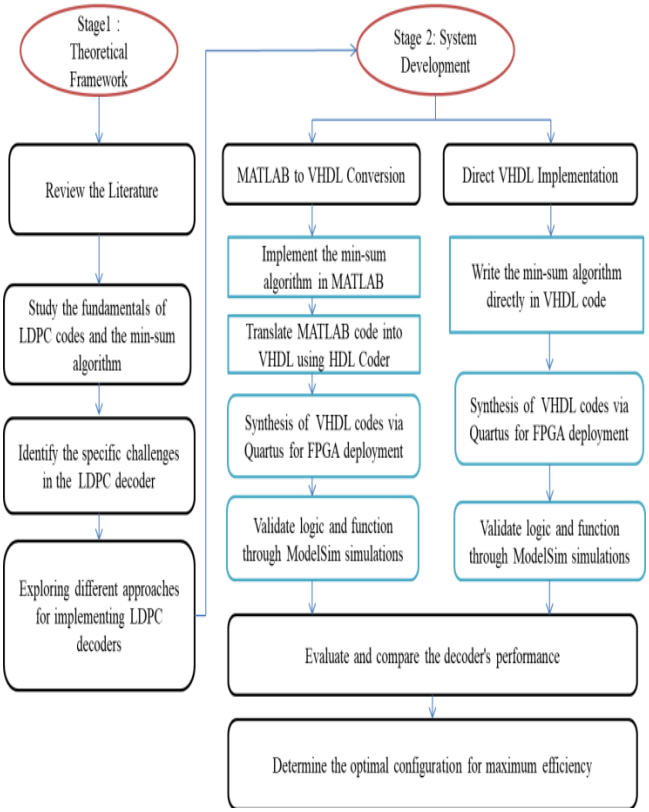


Figure 1. Methodology flowchart for LDPC decoder optimization

Structure of the paper's remainder: The Min-Sum LDPC decoding algorithm's background and related work are covered in Section 2. Section 3 covers the LDPC decoding algorithm. Section 4 covers LDPC decoder implementations. Finally, Section 5: conclusions and future work.

2. RELATED WORK

LDPC codes have gained prominence in modern communication in recent years due to their remarkable -correction capabilities. LDPC codes are a type of linear error-correcting code characterized by a sparse parity-check matrix. The strength of LDPC codes comes from their iterative decoding process, which corrects errors by sending messages between variable nodes and check nodes in a Tanner graph representation. The demand for efficient error correction has spurred developments in LDPC decoder structures and algorithms, especially for FPGA implementations. This section highlights the latest LDPC decoder implementations, focusing on min-sum algorithm application in FPGA-based devices.
Recent studies on FPGA-based LDPC decoders show that the Min-Sum algorithm balances resource efficiency and decoding performance. One study [15] implemented a Min-Sum Algorithm-based quasi-cyclic LDPC decoder. The partially parallel approach reconciles the conflict between hardware resource

usage and decoding efficiency. With a code length of 8,176 bits, the decoder can reach a bit error rate (BER) of 10^-3 at 4 dB and a throughput of 300 Mbps per iteration. Another study [16] implemented the Min-Sum algorithm within a Differential Chaos Shift Keying (DCSK) system on a Kintex7 FPGA, showing enhanced BER performance and real-time processing capabilities. Although increasing the spreading factor affected system performance, the study highlighted the algorithm's resource efficiency, making it suitable for real-time communication systems that require both performance and efficiency.

Recent LDPC decoder designs for 5G applications have reduced complexity while maintaining performance. The reduced-complexity offset min-sum (smOMS) algorithm was introduced in [13] to reduce hardware resource utilization in partially parallel layered decoders. This approach approximates a second minimum with a weight parameter to improve irregular 5G NR code SNR. However, preserving SNR efficiency while reducing complexity, especially across code inconsistencies, remains a challenge. Similarly, [17] introduced a Simplified Offset Min-Sum (SOMS) algorithm to minimize computational demands for QC-LDPC codes further. This design achieved 5G-NR with 13.3 Gbps FPGAs. Advanced decoders are expensive to design and produce, making their use challenging.

LDPC decoder optimization on FPGAs has improved power efficiency and flexibility. A study offered a partially parallel DVB-S2 decoder design that reduced FPGA power consumption by 2.5 W by addressing double-diagonal parity-check matrix submatrices. Although memory access conflicts and the complexity of parity-check operations were issues, the system maintained high decoding performance [18]. Nadal and Baghdadi [19] also provide an FPGA-based, high-speed LDPC decoder for 5G networks that can handle a variety of frame sizes and coding rates. It can decode at 10 Gb/s. When trying to keep data rates high across different configurations, however, striking a balance between the design's complexities and processing efficiency becomes quite difficult.

Improvements in FPGA implementations have been motivated bythe development of high-throughput LDPC decoders for standards like 5G and DVB-S2X. Likhobabin et al. [20] created a DVB-S2X LDPC decoder using field-programmable gate arrays (FPGAs), which can achieve up to 10 Gbps at 250 MHz and has 360 cores. Despite the design's area efficiency and versatility, the complexity of managing QC-LDPC codes, particularly with long code words and large matrices, remained a significant challenge. Similarly, Selvakumari, Shantha [11] explored high-throughput architectures for a 5G NR LDPC decoder using the Min-Sum algorithm. Although additional iterations improved BER and FER, the reliance on multiple iterations could be problem in time-critical or power-limited scenarios. In FPGA implementations of LDPC decoders, memory efficiency and reduced resource utilization are two of the most important factors. In [21], the focus was on FPGA technology for the purpose of efficiently decoding LDPC codes in the logarithmic domain. The implementation utilized 67% of logic elements and 15% of memory bits, reflecting the complexity of the approach and its resource-intensive nature. A hardware-efficient 5G LDPC decoder using the Hybrid Offset Min-Sum (HOMS) algorithm was developed [22]. This approach lowered memory consumption by 10% and achieved 2.82 Gbps, although implementing the HOMS algorithm challenged.

Based on the information from previous studies, there has been a tendency to overlook the comprehensive balancing of computational demands, processing speed, and thermal management in LDPC decoder implementations. Most methods strive to increase throughput or reduce complexity, but at the expense of other important variables. The study compares two implementation methods, MATLAB to VHDL conversion and direct VHDL coding, with a more integrated strategy that incorporates resource utilization, thermal dissipation, and system efficiency. This understanding helps design more balanced and effective FPGA-based LDPC decoders.

3. LDPC DECODING ALGORITHM

This section provides an introduction to fundamental aspects of LDPC coding, including the Tanner graph and definition. In addition, it discusses the Min-Sum Algorithm, the most prevalent hardware implementation method, which balances hardware resources with decoding performance.

3.1 Representation of LDPC codes

LDPC codes are efficient encoding and decoding codes because they detect and correct data transmission errors. The Parity Check Matrix (PCM) and Tanner graph commonly represented LDPC codes, revealing their structure and behavior [23].

First, the sparse binary matrix PCM in LDPC codes connects code bits and parity checks. A '1' in the matrix indicates a bit-parity check connection, while a '0' indicates no connection. This structure helps visualize and analyze code and clarifies bit-parity check relationships during encoding and decoding, enabling mistake discovery and rectification. PCM node degree is the number of nonzero entries in a row or column. These degrees classify LDPC codes as regular or irregular. Regular LDPC codes have uniform

node degrees, meaning each variable and check node has the same number of connections. In contrast, irregular LDPC codes feature varying node degrees, with different numbers of connections for variable and check nodes[24, 25].The parity check matrix in standard LDPC codes has fixed row and column weights. Although parity check matrix building techniques provide design and construction process flexibility due to their random nature, the complexity of hardware design is increased due to the lack of row and column regularity. In order to simplify hardware implementation, structured methods can generate matrices with simply described patterns and levels of distribution [26, 27].

Second, in an LDPC code, the Tanner graph, a bipartite graph, provides a visual representation of the connections between bits and parity checks. This graph divides into variable nodes for bits and check nodes for parity check equations, with edges indicating bit involvement in parity checks. By representing the LDPC code as a Tanner graph, the connections among the bits and the parity checks can be more easily visible, allowing for a deeper understanding of the code's functionality[28].

3.2 Min-sum decoding algorithm

An LDPC code is defined by a sparse parity-check matrix and operates through iterative processes between two groups: check nodes (CNs) and variable nodes (VNs). With near-optimal decoding performance nearing the Shannon limit, the Belief Propagation (BP) algorithm is the most powerful message-passing method for LDPC decoding. Nevertheless, a substantial amount of computing complexity is required to achieve this level of high performance. One approach to this problem is the Min-Sum (MS) method, which simplifies the process while reducing computing requirements and improving error correction. The steps of the Min-Sum algorithm are as follows [29]:

(1) Initialization: Soft information bits, also known as log likelihood ratios (LLRs), are initially used to initialize variable nodes from the channel.

(2) Variable to check node: Following initialization, bit nodes update their check nodes with bits to verify the message. Where Qij is the bit node -to-check node message.

$$Qij = Li = \log \frac{P(ai = 0|bi)}{P(ai = 1|bi)} \quad (1)$$

(3) Check node update: The Q message that the bit node sends is used to calculate the messages that the check node sends, with Rij standing for the message that the check node sends to the bit node.

$$R_{ij} = \prod_{j' \in V(i) \setminus j} \text{sign}(Qij) \times \min |Qij'| \quad (2)$$

To calculate Rij′s min |Qij′|, check node processing must omit V(j), therefore two minima must be found: Min1 and Min2. More specifically, min |Qij′| is:

$$\min_{j' \in V(i) \setminus j} |Qij'| = \begin{matrix} \text{Min1, if } j \neq \text{argmin(Min1i)} \\ \text{Min2, if } j = \text{argmin(Min2i)} \end{matrix} \quad (3)$$

(4) Bit node update: Bit nodes determine their messages Qij using the following formula:

$$Q_{ijnew} = Q_{ijold} + \sum_{i' \in C(j) \setminus i} Ri'j \quad (4)$$

(5) A message from bit node V(i) to check node C(j) is calculated using all R messages from neighboring check nodes, except for the R message from check node C(j). This computation is similar to that of the check-to-bit messages.

(6) Bit reliability calculation: Following the bit nodes' updates, a reliability value yi is determined using the following formula:

$$yi = Q_{ijold} + \sum_{i \in C(j)} Rij \quad (5)$$

From Eq. (5), the Y vector represents:

$$Y = (y1, y2, y3, \dots, yn)$$

(7) Bit value determination: The following formula determines the bit values:

$$Zi = \begin{cases} 1 \text{ if } yi \leq 0 \\ 0 \text{ if } yi > 0 \end{cases} \quad (6)$$

(8) Decoding verification: The procedure of decoding checks if $H \times Z^T = 0$. The procedure continues until either a valid code is found or the maximum number of iterations is achieved if this condition is not met. Figure 2 shows the steps of the Min-Sum decoding algorithm, which uses iterative message-passing to decode LDPC codes.

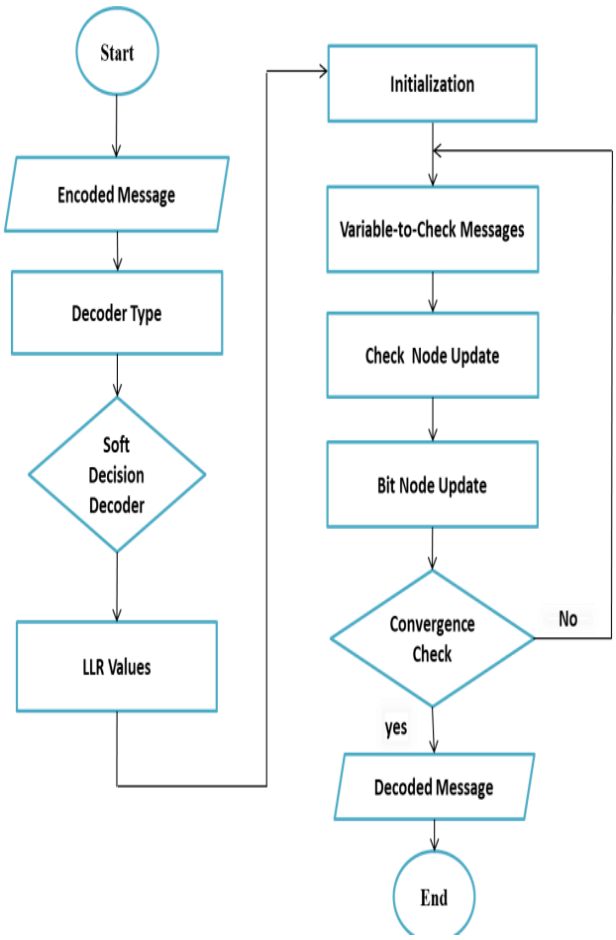


Figure 2. Flowchart of Min-Sum algorithm for LDPC decoding

In contrast to the BP algorithm, the MS method's minimum function in the CN process produces an approximated message. Because of this, decoding performance is reduced. Many versions of the modified original MS algorithm, such as Offset Min-Sum (OMS) and Normalized Min-Sum (NMS), were devised to compensate for this performance reduction, both of which aim to reduce computational complexity while maintaining decoding performance [30]. The Simplified Offset Min-Sum (SOMS) form of the OMS algorithm reduces computing complexity and routing requirements, improving FPGA throughput and hardware efficiency [17]. However, optimizing error correction performance and resource utilization with NMSA on FPGA considerably improves processing efficiency [31].

The LDPC encoder on the transmitter side generates code words by adding parity bits to the message bits using the generator matrix G, creating error-correcting code words. Conversely, the receiver's LDPC decoder employs the parity-check matrix H to correct transmission errors, ensuring accurate message delivery. It identifies and corrects errors by comparing received bits against parity-check equations. This streamlined error-correction process guarantees the integrity of the message, as depicted in Figure 3.

4. LDPC Decoder Implementations

The LDPC decoder used in this study was applied in two different methodologies: MATLAB-to-VHDL and direct VHD. It operated in both approaches in a completely parallel processing mode and made use of a 5 × 10 binary regular LDPC parity-check matrix and the min-sum algorithm. With 5 message bits and 10-bit code words, the architecture achieves a code rate of 0.5. The decoding process is designed to complete in 5 iterations.

The matrix used in the implementation is as follows:

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

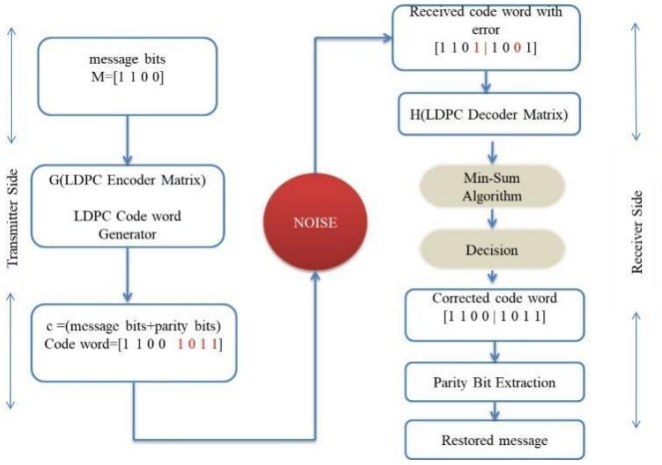


Figure 3. The communication process between sender and receiver, utilizing an LDPC encoder and decoder

The decoder uses a sparse matrix to connect variable nodes (V1 to V10) and check nodes (C1 to C5), optimizing computation and communication for efficiency. As shown in Figure 4, LLRs are refined through a maximum of five iterations, stopping when parity is achieved, or when the maximum number of iterations is reached.

The LDPC decoder was implemented using two approaches, both utilizing the same Altera Cyclone IV E FPGA board (model EP4CE15F23C6). These approaches are outlined below:

4.1 The process of implementing the LDPC decoder from MATLAB to VHDL included several crucial steps

The development of the LDPC decoder began with implementing and validating the algorithm in MATLAB. To assure functionality, MATLAB tools were used to write and test decoder logic. After validation, HDL Coder converted the MATLAB script to VHDL for FPGA execution. The algorithm had to be translated to the hardware description language and FPGA architecture.

After translation, Altera's Quartus FPGA synthesis tool synthesized VHDL code. Mapping the code to the Altera Cyclone IV E FPGA board's design, specifically the EP4CE15F23C6 model, optimizing it for timing and resource limitations, and guaranteeing efficient operation within the FPGA's logic and resource limits was required.

The final step involved using ModelSim simulations to verify that the VHDL code was prepared for FPGA deployment and accurately represented the original MATLAB design. This process is illustrated in Figure 5.

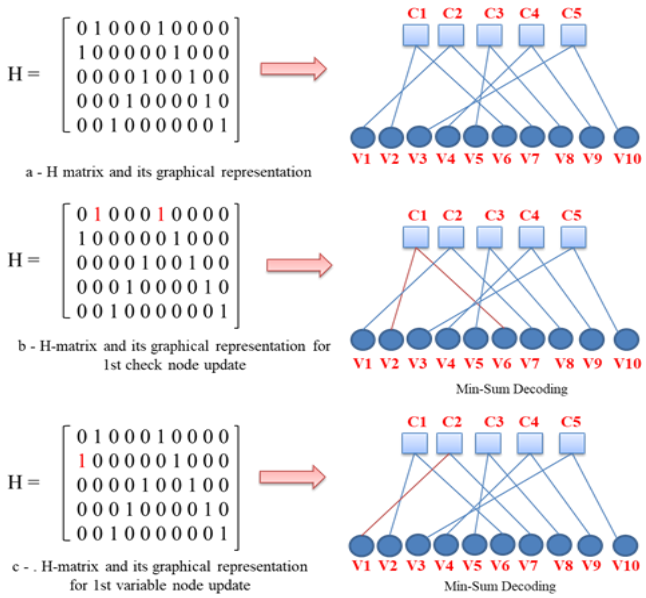


Figure 4. LDPC iterative decoding stages with a Tanner graph representation

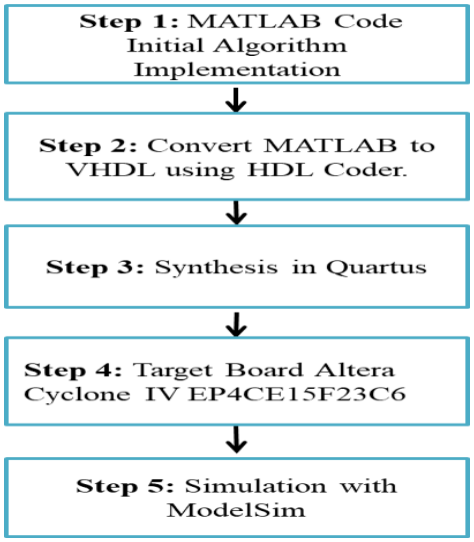


Figure 5. LDPC decoder implementation diagram (MATLAB to VHDL)

4.2 Implementing the LDPC decoder directly in VHDL required a series of essential steps

The LDPC decoder was implemented directly by writing VHDL code specifically optimized for FPGA execution. This method provided precise control over the design, enabling efficient utilization of FPGA resources such as logic gates. Quartus synthesized the VHDL code, which was mapped onto the Altera Cyclone IV E FPGA (model EP4CE15F23C6) and optimized for performance and resource efficiency. Finally, the correctness, dependability, and readiness of the FPGA were validated by ModelSim simulations,the steps are depicted in Figure 6.

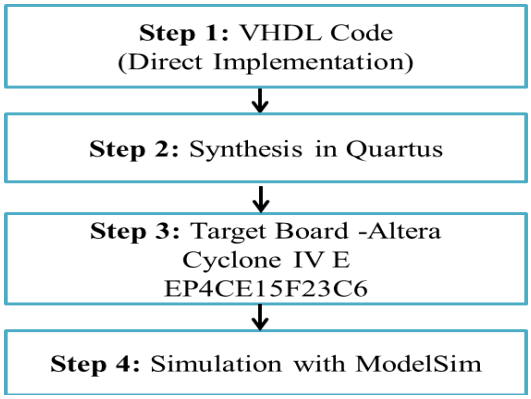


Figure 6. LDPC decoder implementation design flow (direct VHDL)

LDPC decoders use parallel processing and the Min-Sum algorithm in both implementation methods. It stores LLR input data, which indicates a bit's possibility of being '0' or '1', in buffers. One iteration controller verifies this data before each decoding cycle. During the cycle, variable nodes update bit values based on the LLR data, which are sent to check nodes for further refinement. Finally, the Bit Decision component determines the most likely value of each bit, ensuring accurate decoding. This process is illustrated in Figure 7.

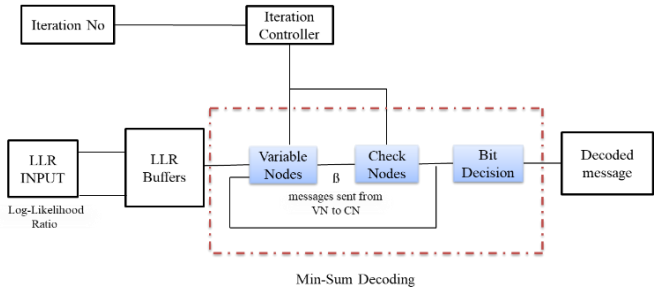


Figure 7. Parallel LDPC decoder architecture

The following pseudocode outlines the Min-Sum decoding process implemented in the LDPC decoder architecture:

```
Procedure Min-Sum Decoding Process
I=0      Initialization
for each VN in parallel, do
 for each CN connected to VN, do
Qij = LLR(VN) Initialize message from VN to CN
      end for
   end for
   repeat
  Step 1: Check Node Update
    for each CN in parallel, do
        for each VN connected to CN, do
  Rij = Product of signs (Qij from all other VNs)
Rij = Minimum (|Qij| from all other VNs)
          end for
      end for
  Step 2: Variable Node Update
       for each VN in parallel, do
          for each CN connected to VN, do
Qij_new = LLR (VN) + Sum (Rij from all other CNs)
        end for
     end for
  Step3: Decision Making
  for each VN in parallel, do
 yi = LLR (VN) + Sum (Rij from all connected CNs)
  if yi ≤ 0 then
  Z = 1
      else
 Z = 0
  end if
    end for
    I =I+1
 Step4:  Test
  if H * Z^T = 0 then
  Finished
    end if
  Until I = IMAX or H * Z^T = 0
  return Decoded Message(Z)
end procedure
```

5. RESULTS AND DISCUSSION

This section presents the performance evaluation of the LDPC decoder using MATLAB-to-VHDL conversion and direct VHDL coding. Simulations were conducted on a system with robust hardware and advanced software tools.

Table 1 outlines the key specifications of the computing environment, including the processor, memory, and software tools.

Table 1. Specifications of the computing environment

Component		**Specification**
Hardware	CPU	IntelCorei7- 10750H
	Frequency	2.60 GHz
	RAM	16.0 GB
	Hard drive	476 GB
Software	Operating system	Windows 11 pro-64-bit
	Tools	MATLAB R2021a Quartus Prime 21.1, ModelSim 20.1 (VHDL)

Additionally, the detailed characteristics of the FPGA board used for implementation, such as logic blocks, RAM, and operational frequencies, are summarized in Table 2.
Figure 8 illustrates the MATLAB code to VHDL translation using an RTL (Register Transfer Level) diagram. The diagram shows the organized logic resulting from the VHDL translation, allowing for extensive decoder architecture exploration in Quartus.
Additionally, Figure 9 shows the ModelSim implementation results, which are useful in understanding the hardware's dynamic behavior. These simulations provide a comprehensive evaluation of the system's behavior, particularly the internal states and outputs of the LDPC decoder. Decoding is based on probabilistic estimates, with each received signal (rx) representing the corresponding LLR for the received bit. Each LLR is treated independently while the signals are processed. The decoder's ability to refine bit estimates through iterative processing is indicated by the iteration signal (101), which represents the number of iterations performed. The decoder's best estimate as to the original transmitted message after all iterations is displayed by the VHat signal, which is the final decoded message.

Table 2. Details of the FPGA board used for implementation

No.	Attribute	Details
1	Series	Cyclone IV E
2	Number of logic blocks	963
3	Embedded block RAM - EBR	504 Kbit
4	Number of I/Os	343
5	Maximum operating frequency	200 MHz
6	Operating supply voltage	1 V to 1.2 V
7	Maximum operating temperature	+ 70 C
8	Mounting style	SMD/SMT
9	Package/case	FPGA-484
10	Minimum operating temperature	0 C
11	Packaging	Tray

The second approach concentrates onimplementing the LDPC decoder directly in VHDL, with the min-sum algorithm programmed within the VHDL environment. Instead of using MATLAB, the algorithmic functions are built directly into the hardware using this approach.
Figure 10 illustrates the logic gates used in this direct VHDL implementation. Specifically, Figure 10 (a) shows the RTL layout, while Figure 10 (b) and Figure 10 (c) detail the horizontal and vertical phases of node selection, respectively.
Additionally, Figure 11 presents the ModelSim simulation results, highlighting the hardware's dynamic performance. In this direct VHDL approach, the decoder processes inputs as vectors, managing multiple signals simultaneously for efficient LLR processing. The rx signals represent the LLRs for each received bit, with the VHat signals showing the final decoded message after each iteration. The entire decoding process is completed in five iterations, ensuring accurate message recovery.
Regarding the outcomes of the comparison between the two methods, Table 3 presents a detailed comparison of key parameters between the two approaches, which are critical for assessing the performance and efficiency of the hardware designs.

Table 3**.** The comparison between MATLAB to VHDL and direct VHDL implementations

Key Metrics	MATLAB to VHDL	Direct VHDL	Improvement
Number of logic gates	1,702 gates (11%)	1,070 gates (7%)	37.13% fewer gates (Direct VHDL)
Number of registers	0	0	……..
Total pins	153 pins (44%)	320 pins (93%)	52.19% fewer pins (MATLAB to VHDL)
Core static thermal power dissipation	49.51 mW	49.47 mW	0.08% improvement (Direct VHDL)
Input/output thermal power dissipation	53.55 mW	68.22 mW	27.39% increase (Direct VHDL)
Total thermal power dissipation	103.06 mW	117.69 mW	14.20% increase (Direct VHDL)

Figure 12 highlights the differences in logic gates, total pins, core static thermal power dissipation, input/output thermal power dissipation, and overall thermal power dissipation between the methods. The MATLAB-to-VHDL approach excels in reducing input/output and total thermal power dissipation as well as minimizing the number of pins, while the direct VHDL approach is better at reducing gate count and core static thermal power dissipation. The choice of method depends on the specific application requirements, available resources, and the desired balance between resource efficiency and design complexity.

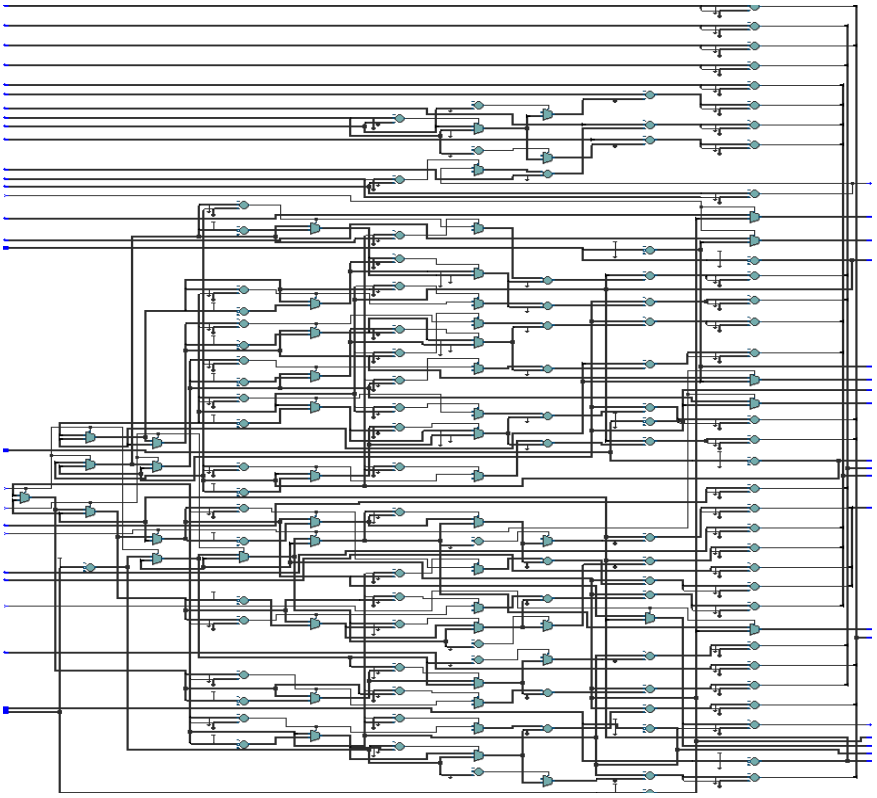


Figure 8. The logic gates involved in the MATLAB-to-VHDL conversion process

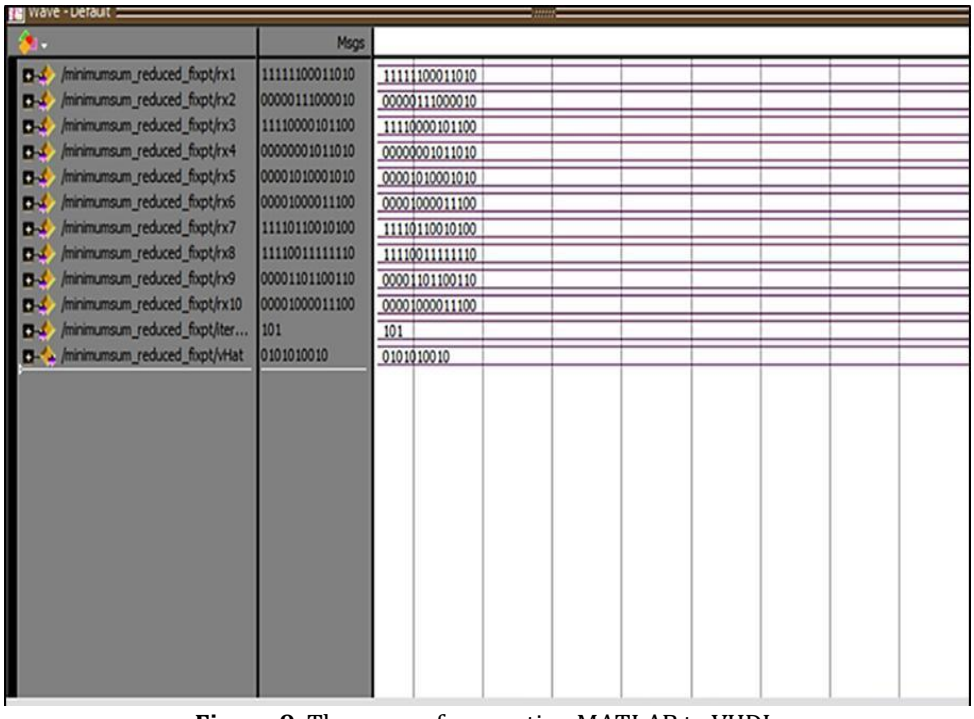
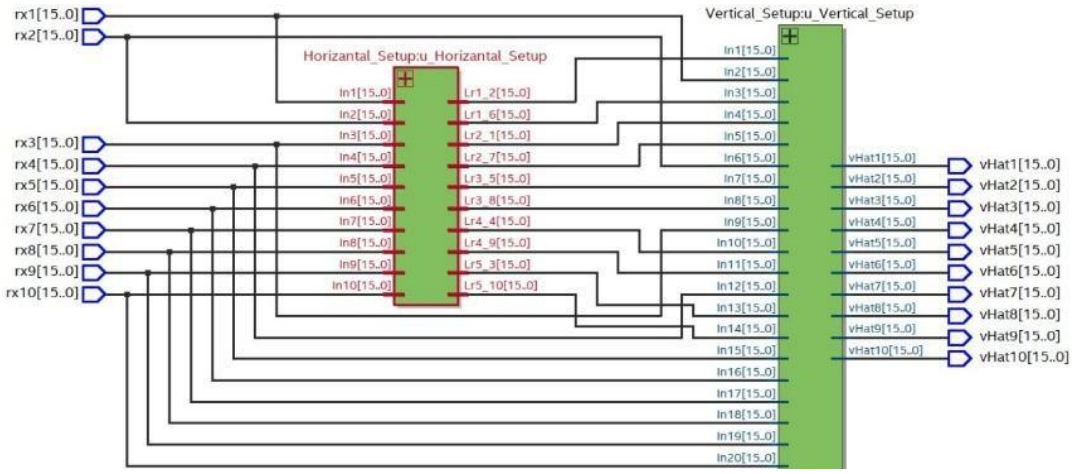
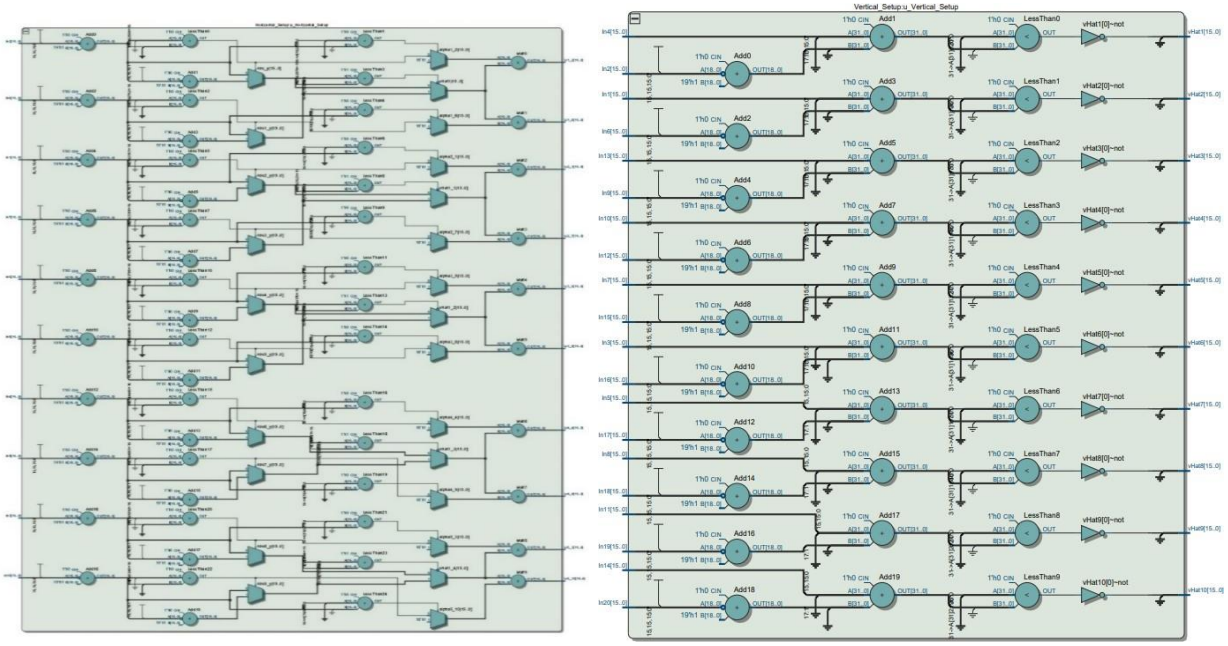


Figure 9. The wave of converting MATLAB to VHDL



(a) The implementation of the LDPC decoder system usingdirect VHDL



(b) Horizontal setup block contents (c) Vertical setup block contents

Figure 10. The direct method VHDL

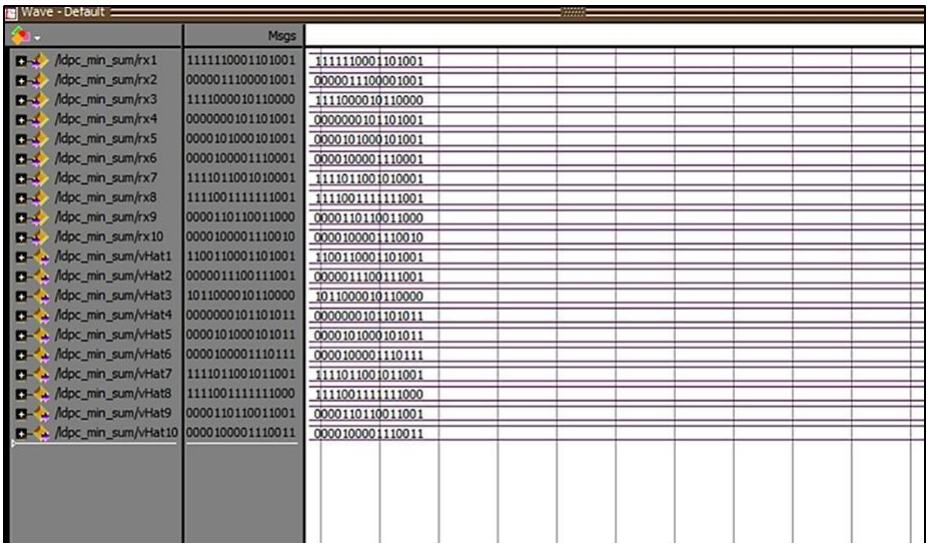


Figure 11. The wave of direct VHDL

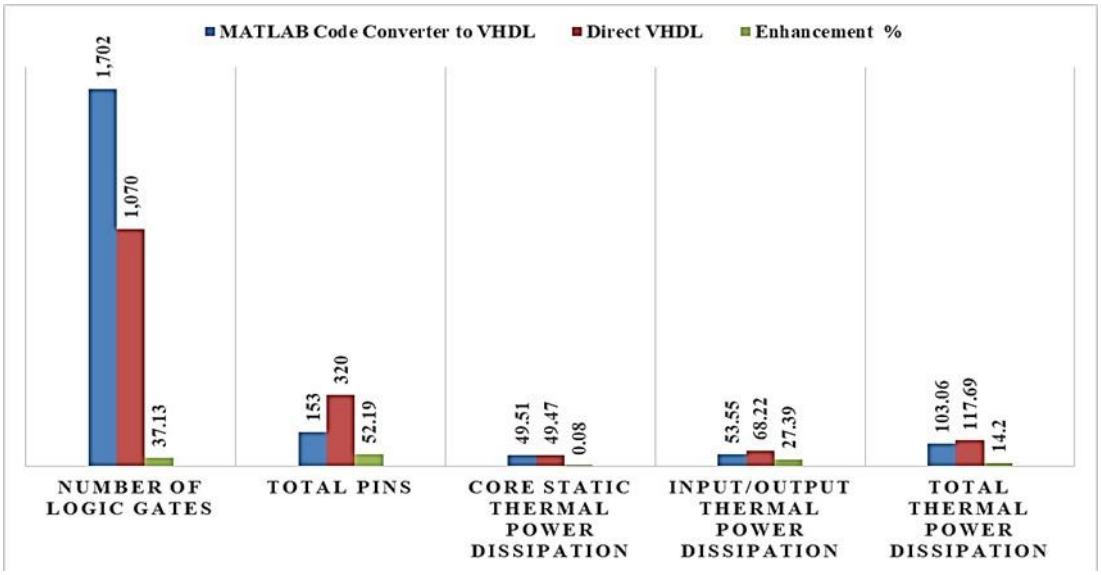


Figure 12. Comparison of logic gates, pins, and thermal power dissipation between MATLAB-to-VHDL and direct VHDL implementations

6. CONCLUSIONS

The min-sum algorithm for LDPC decoders achieves a balance between computational efficiency and error-correction performance. Implementing these decoders on FPGA platforms increases their speed and adaptability. Two strategies for efficient decoding are discussed: MATLAB-to-VHDL conversion and direct VHDL encoding, both utilizing the min-sum algorithm. Direct VHDL implementation shows a superior result in logic gates, with 37.13% fewer gates needed than the MATLAB to VHDL conversion. However, MATLAB to VHDL conversion is better in terms of pin count, requiring 52.19% fewer pins than the direct VHDL approach. When looking at thermal power dissipation, the direct VHDL has a slight 0.08% improvement in core static thermal power dissipation over MATLAB to VHDL. Nevertheless, MATLAB to VHDL is more efficient in overall thermal power dissipation, with direct VHDL showing a 14.20% increase in total thermal power dissipation and 27.39% increase in input/output thermal power dissipation.

Improving the overall efficiency of communication systems is possible through the reduction of computational complexity, which enhances response speed and enables effective operation at higher frequencies. Reducing heat losses also reduces power consumption, which in turn increases the device's lifetime. With communication standards evolving rapidly, these improvements are essential for a more efficient and durable wireless infrastructure.

Future research will focus on optimizing LDPC decoders for upcoming communication standards, especially 6G networks where speed is crucial. These decoders could improve wireless infrastructure efficiency and dependability in power-constrained settings for next-generation communication systems.

REFERENCES

[1] Le, K., Ghaffari, F., Declercq, D., Vasic, B., Winstead, C. (2017). A novel high-throughput, low-complexity bit-flipping decoder for LDPC codes. 2017 International Conference on Advanced Technologies for Communications (ATC), Quy Nhon, Vietnam.

[2] Gallager, R. (1962). Low-density parity-check codes. IRE Transactions on Information Theory, 8(1), 21-28. https://doi.org/10.1109/TIT.1962.1057683

[3] Davey, M.C., MacKay, D.J. (1998). Low density parity check codes over GF (q). 1998 Information Theory Workshop (Cat. No. 98EX131), Killarney, Ireland.

[4] Brack, T., Alles, M., Lehnigk-Emden, T., Kienle, F., Wehn, N., L'Insalata, N.E., Rossi, F., Rovini, M., Fanucci, L. (2007). Low complexity LDPC code decoders for next generation standards. 2007 Design, Automation & Test in Europe Conference & Exhibition, Nice, France.

[5] Gupta, S.H., Virmani, B. (2009). LDPC for Wi-Fi and WiMAX technologies. 2009 international conference on emerging trends in electronic and photonic devices & systems, Varanasi, India.

[6] Kamal, N.N., Al-Doori, Q.F., Alani, O. (2023). A review of modern and recent studies on the Low-Density Parity-Check technology approach. Iraqi Journal of Computers, Communications, Control and Systems Engineering, 23(1), 165-178.

[7] Nazar, N., Al-Doori, Q., Alani, O. (2023). Designing a multi-frequency Low Power Low-Density Parity Check (LDPC) encoder with a dynamic voltage and frequency scaling (DVFS) approach. International

Journal of Intelligent Engineering & Systems, 16(4), 497-511. https://doi.org/10.22266/ijies2023.0831.40

[8] Pusane, A.E., Smarandache, R., Vontobel, P.O., Costello, D.J. (2011). Deriving good LDPC convolutional codes from LDPC block codes. IEEE Transactions on Information Theory, 57(2), 835-857. https://doi.org/10.1109/TIT.2010.2095211

[9] Majeed, A.H. (2012). LDPC error floor improvement. Engineering and Technology Journal, 30(3), 474-488.

[10] Hasan, F.S., Mosleh, M.F., Abdulhameed, A.H. (2021). FPGA implementation of LDPC soft-decision decoders based DCSK for spread spectrum applications. International Journal of Electrical and Computer Engineering (IJECE), 11(6), 4794-4809. https://doi.org/10.11591/ijece.v11i6.pp4794-4809

[11] Selvakumari, R.S. (2022). Performance analysis of Min-Sum based LDPC decoder architecture for 5G new radio standards. Materials Today: Proceedings, 62, 4965-4972. https://doi.org/10.1016/j.matpr.2022.03.693

[12] Subhi, M.I., Al-Doori, Q., Alani, O. (2023). Enhancing data communication performance: A comprehensive review and evaluation of LDPC decoder architectures. Ingénierie des Systèmes d'Information, 28(5), 1113. https://doi.org/10.18280/isi.280501

[13] Petrović, V.L., El Mezeni, D.M. (2020). Reduced-complexity offset min-sum based layered decoding for 5G LDPC codes. 2020 28th Telecommunications Forum (TELFOR), Belgrade, Serbia.

[14] Kingston Roberts, M., Kumari, S., Anguraj, P. (2021). Certain investigations on recent advances in the design of decoding algorithms using low-density parity-check codes and its applications. International Journal of Communication Systems, 34(8), e4765. https://doi.org/10.1002/dac.4765

[15] Zhao, H., Zhang, H. (2019). Design and FPGA implementation of a quasi-cyclic LDPC decoder. Communications, Signal Processing, and Systems.

[16] Mosleh, M.F., Hasan, F.S., Abdulhameed, A. H. (2021). Reduce the resources utilization of LDPC decoder based on Min-Sum decoder for spread spectrum applications. IOP Conference Series: Materials Science and Engineering, 1105, 012033. https://doi.org/10.1088/1757-899X/1105/1/012033

[17] Verma, A., & Shrestha, R. (2022). Low computational-complexity SOMS-algorithm and high-throughput decoder architecture for QC-LDPC codes. IEEE Transactions on Vehicular Technology, 72(1), 66-80. https://doi.org/10.1109/TVT.2022.3203802

[18] Zinchenko, M.Y., Levadniy, A.M., Grebenko, Y.A. (2020). LDPC decoder power consumption optimization. 2020 International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE), Moscow, Russia.

[19] Nadal, J., Baghdadi, A. (2021). Parallel and flexible 5G LDPC decoder architecture targeting FPGA. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29(6), 1141-1151. https://doi.org/10.1109/TVLSI.2021.3072866

[20] Likhobabin, E., Ovinnikov, A., Goriushkin, R., Nikishkin, P., Khokhryakov, E. (2022). High throughput FPGA implementation of LDPC decoder architecture for DVB-S2X standard. 2022 International Conference on Information, Control, and Communication Technologies (ICCT), Astrakhan, Russian Federation.

[21] Ji, Y., Ji, Y. (2021). Hardware implementation method of LDPC efficient decoding based on FPGA. 2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC), Guiyang, China.

[22] Tran-Thi, B.N., Nguyen-Ly, T.T., Hoang, T. (2023). An FPGA design with high memory efficiency and decoding performance for 5G LDPC decoder. Electronics, 12(17), 3667. https://doi.org/10.3390/electronics12173667

[23] Nazar, N., Al-Doori, Q., Alani, O. (2023). Low power Low-Density Parity Check encoder using dynamic voltage and frequency scaling approach. Mathematical Modelling of Engineering Problems, 10(1), 14-22. https://doi.org/10.18280/mmep.100102

[24] Abd Al-Kareem, L.G. (2011). Low density Parity check (LDPC) codes for a proposed slantlet transform OFDM system in a rayleigh fading channels with perfect and pilot channel estimation for M_ARY PSK modulation. Iraqi Journal of Computers, Communications, Control, and Systems Engineering, 11(2), 65-78.

[25] Mosleh, M.F. (2011). Evaluation of low density parity check codes over various channel types. Engineering and Technology Journal, 29(5), 961-971.

[26] Chandrasetty, V.A., Aziz, S.M. (2017). Resource efficient LDPC decoders: from algorithms to hardware architectures. Academic Press.

[27] Malema, G.A. (2007). Low-density parity-check codes: construction and implementation. PhD thesis. University of Adelaide, School of Electrical and Electronic Engineering.
[28] Karkooti, M., Radosavljevic, P., Cavallaro, J. R. (2008). Configurable LDPC decoder architectures for regular and irregular codes. Journal of Signal Processing Systems, 53, 73-88. https://doi.org/10.1007/s11265-008-0221-7
[29] Kakde, S., Khobragade, A., Ambatkar, S., Nandanwar, P. (2017). Implementation of layered decoding architecture for LDPC code using layered Min-Sum algorithm. IIUM Engineering Journal, 18(2), 128-136. https://doi.org/10.31436/iiumej.v18i2.677
[30] Tran-Thi, B.N., Nguyen-Ly, T.T., Hong, H.N., Hoang, T. (2021). An improved offset min-sum LDPC decoding algorithm for 5G new radio. 2021 International Symposium on Electrical and Electronics Engineering (ISEE), Minh, Vietnam.
[31] Kun, C., Qi, S., Shengkai, L., Chengzhi, P. (2019). Implementation of encoder and decoder for LDPC codes based on FPGA. Journal of Systems Engineering and Electronics, 30(4), 642-650. https://doi.org/10.21629/JSEE.2019.04.02