Proposed early-stopping algorithm for the FPGA implementation of a Min-Sum decoder for NB-LDPC codes

Pham Tuan Dat, Pham Xuan Nghia*

Faculty of Radio-Electronic Engineering, Le Quy Don Technical University, 236 Hoang Quoc Viet, Nghia Do, Hanoi, Vietnam.

*Corresponding author: nghiapx@lqdtu.edu.vn

Received 20 Mar. 2026; Revised 06 May 2026; Accepted 14 May 2026; Published 25 Jun. 2026.

DOI: https://doi.org/10.54939/1859-1043.j.mst.112.2026.47-55

ABSTRACT

This paper proposes an early-stopping algorithm applied to the design of a Min-Sum decoder for NB-LDPC (32,16) codes on FPGA. The decoder is implemented in SystemVerilog for NB-LDPC codes over GF(16) and evaluated on AWGN channels with performance metrics including bit error rate (BER), frame error rate (FER), average iteration count, and decoding latency. The early-stopping algorithm is integrated into the loop control block to terminate the decoding process as soon as convergence is achieved, thus reducing the average number of decoding iterations from 18 to 7.44 at E_b/N_0 = 6 dB. Implementation results show that the average decoding latency is reduced to 134.4 ns, corresponding to BER ≈ 5.31 × 10^{-3} and FER = 0.12, consistent with reference MATLAB simulations. These results demonstrate that implementing NB-LDPC coding and decoding algorithms on FPGA, combined with an appropriate early-stopping mechanism, can significantly reduce hardware processing time, enabling real-time communication applications.

Keywords: NB-LDPC; Min-Sum; Early-stopping algorithm; FPGA; SystemVerilog; Throughput; Latency.

1. INTRODUCTION

Recent FPGA-oriented LDPC/NB-LDPC studies show a clear trend toward balancing algorithmic simplification and implementation efficiency. In addition to foundational NB-LDPC works, more recent hardware papers emphasize trellis-based Min-Max simplification, architecture-level pruning, and implementation-aware decoding strategies on modern FPGA/SoC platforms. This manuscript is updated to explicitly position the proposed early-stopping Min-Sum decoder within that recent landscape [5-11].

Low-density parity-check (LDPC) codes were proposed by Gallager in 1962 [1]. However, for a long time, these codes were not widely adopted due to high computational complexity from iterative decoding and significant hardware implementation costs. With advances in FPGA technology and digital design tools, variants of LDPC codes, particularly NB-LDPC codes, have become increasingly feasible for high-reliability, low-latency communication systems.

For NB-LDPC codes, hardware implementation is more challenging than binary LDPC codes due to arithmetic operations performed over finite fields GF(q), larger message exchange volume, and more complex check-node updates. Nevertheless, NB-LDPC codes offer superior error correction performance at short and medium block lengths, making them particularly suitable for real-time processing applications. Therefore, research on FPGA implementation of NB-LDPC decoders is meaningful for both theoretical and practical perspectives [2, 4, 6, 7].

In iterative decoders, the average latency depends directly on the number of iterations executed. If the decoder always runs to the maximum number of iterations, the processing latency will be unnecessarily large, especially at medium and high SNR regions where many frames converge early. Thus, proposing and integrating a rational early-stopping algorithm is necessary to reduce latency while maintaining decoding performance [7, 10].

This paper focuses on implementing a Min-Sum decoder for NB-LDPC (32,16) codes on FPGA and proposes an early-stopping algorithm to reduce the average number of decoding iterations. The main contributions include: description of NB-LDPC codes and Min-Sum decoding principles, proposal of early-stopping conditions suitable for hardware architecture, and performance evaluation on AWGN channels with comparison between cases with and without early-stopping.

The remainder of this paper is organized as follows. Section 2 presents the theoretical basis of NB-LDPC codes, the Min-Sum decoding process, and the proposed early-stopping criterion with analytical discussion. Section 3 describes the hardware-oriented implementation flow for NB-LDPC encoding/decoding with the early-stopping mechanism. Section 4 reports the simulation procedure, validation methodology, and experimental results. The final section concludes the paper and outlines future work.

2. NB-LDPC CODES AND MIN-SUM DECODING ALGORITHM

2.1. NB-LDPC code model (32,16)

Let $H \in \mathrm{GF}(q)^{M \times N}$ be the parity-check matrix of an NB-LDPC code over GF(q) where $q = 2^m$. In this design, we use $q = 16$ (i.e., $m = 4$), and the code parameters are:

$$N = 32, \quad K = 16, \quad M = N - K = 16 \tag{1}$$

with code rate

$$R_c = \frac{K}{N} = 0.5. \tag{2}$$

Each code symbol belongs to GF(16) and is represented by 4 bits. Therefore, the message frame length is 64 bits and the codeword length is 128 bits. The parity-check condition is:

$$\boldsymbol{H}\boldsymbol{c}^T = \mathbf{0}, \tag{3}$$

where all operations are performed over GF(16).

2.2. Sparse representation of parity-check matrix

Instead of storing the full matrix H in dense form, the decoder only stores the positions of non-zero elements and their coefficients on each edge of the Tanner graph. This representation reduces memory requirements and aligns with row-wise processing in the check-node update block. If $N(m)$ is the set of variable nodes connected to check node m, then:

$$h_{m,n} \neq 0 \Leftrightarrow n \in N(m), \quad |N(m)| = d_c, \tag{4}$$

where N(m) denotes the set of variable nodes connected to check node m in the prototype considered in this paper.

2.3. Min-Sum decoding algorithm for NB-LDPC

For BPSK transmission over AWGN channels, the received signal is:

$$y_k = x_k + n_k, \tag{5}$$

where y_n is the received channel sample and n is the symbol index.

From the received signal, the decoder generates initial symbol reliability values $L_n(a)$ for each symbol $a \in \mathrm{GF}(16)$ at position n.

In each iteration, messages from variable nodes to check nodes are updated according to:

$$Q_{n \to m}(a) = L_n(a) + \sum_{m' \in M(n) \setminus m} R_{m' \to n}(a), \tag{6}$$

where M(n) is the set of check nodes adjacent to variable node n. Then, the hard decision is made:

$$\hat{c}_n = \arg \max_{a \in GF(16)} Q_n(a), \tag{7}$$

At the check node, the Min-Sum algorithm approximates the message update problem by reducing complexity compared to full probabilistic propagation. For edge (m, n), the outgoing message $Rm \rightarrow n(a)$ is computed under the constraint of the parity-check equation at the check node. This approximation follows the well-known low-complexity belief-propagation family in the log domain [2, 4, 6].

In hardware implementation, this process is optimized using sparse storage structures, finite field lookup tables, and row-wise computation organization of matrix H.

2.4. Clarification of Min-Sum approximation

The exact sum-product rule at a check node requires log-sum-exp operations over multiple symbol configurations, which is expensive in hardware. Min-Sum replaces this operation with a minimum (or maximum in reliability form), significantly reducing arithmetic complexity [4, 6]. In practical terms, the check-node update can be interpreted as selecting the most reliable configuration that satisfies the parity constraint, rather than accumulating all possible configurations.

To compensate for approximation bias, practical implementations use normalized and offset Min-Sum corrections [3, 4, 6]. In generic reliability-domain form:

$$R_{m\rightarrow n}^{\mathrm{NMS}}(a) = \alpha\, R_{m\rightarrow n}^{\mathrm{MS}}(a), \quad 0 < \alpha \leq 1. \tag{8}$$

$$R_{m\rightarrow n}^{\mathrm{OMS}}(a) = \max\left(R_{m\rightarrow n}^{\mathrm{MS}}(a) - \beta,\, 0\right), \quad \beta \geq 0. \tag{9}$$

In this work, we adopt the Min-Sum principle as the baseline because it offers a favorable trade-off between decoding performance and hardware cost for short NB-LDPC codes. This choice is consistent with prior non-binary reduced-complexity decoder studies [4, 6, 11].

Figure 1 summarizes the theoretical iteration flow of the Min-Sum decoder used in this paper, from channel reliability initialization to the convergence/maximum-iteration stopping condition.

Implementation configuration clarification: the RTL decoder uses Min-Sum as the baseline reliability update rule. For reproducibility, this manuscript explicitly declares the finite-field initialization polynomial p(x) = x^4 + x + 1 for GF(16). In the baseline setting, no additional normalization or offset correction is enabled (equivalent to alpha = 1 and beta = 0).

3. MIN-SUM ALGORITHM IMPLEMENTATION ON FPGA AND EARLY-STOPPING MECHANISM

3.1. Min-Sum algorithm implementation on FPGA

In the proposed architecture, the Min-Sum decoder is organized following an iterative cycle with three main functional blocks: check-node update, variable-node update, and hard decision. Data flow is controlled by a central finite state machine (FSM) with states for input data loading, decoding iteration, convergence checking, and result output.

To align with FPGA implementation, the design applies the following principles:

- Store non-zero elements of the parity-check matrix sparsely to reduce memory and avoid redundant accesses.
- Organize data by symbol domain to reduce message access in each iteration.
- Use lookup tables for GF(16) operations to minimize combinational logic cost.
- Separate control path and data path for timing closure and easy extension of early-stopping functionality. Under the current measurement setup, the average latency model per frame is:

$$T(n) = 60 + 10n \text{ ns} \tag{10}$$

(within the Vivado 2020.1 evaluation scope at 100 MHz test clock, i.e., 10 ns cycle).

This model forms the basis for quantifying the effect of the early-stopping mechanism in subsequent sections.

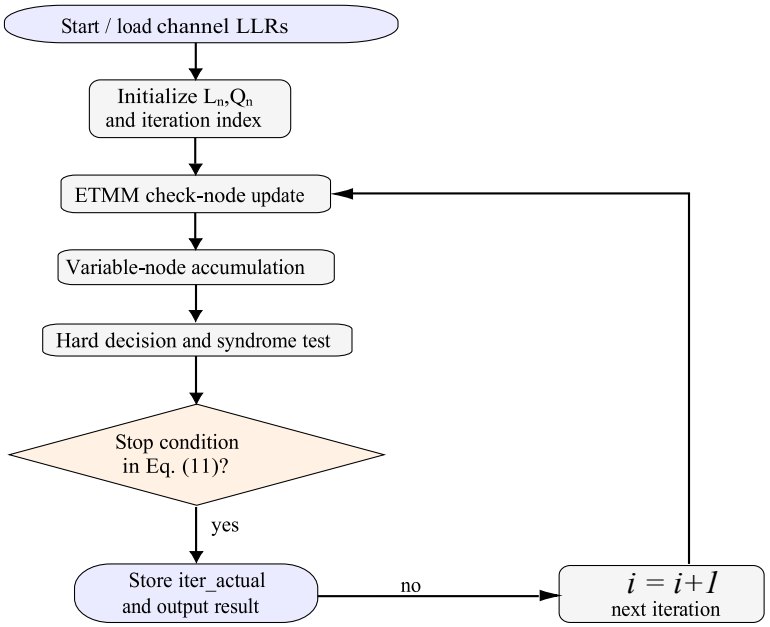

Figure 1. *Algorithm flowchart of the proposed NB-LDPC decoder.*

3.2. Early-stopping algorithm proposal

In conventional designs, the decoder stops when either convergence is achieved or the maximum iteration count I_{max} is reached. However, during decoding, when error floor phenomena appear, increasing the iteration count provides no further benefit. Information does not change, which signals the condition to stop decoding.

This approach conserves hardware resources and reduces decoding time without affecting decoding quality.

Based on this theoretical foundation, we propose an extended stopping condition:

$$\begin{aligned} \text{stop} = {} & \text{syndrome\_ok} \vee \text{hard\_stable\_2iter} \\ & \vee (\text{FAST\_EN} \wedge i \geq i_{min} \wedge \text{hard\_stable\_1iter}) \\ & \vee (i = I_{max}), \text{ with } I_{max} = 18 \text{ and } i_{min} = 6. \end{aligned} \tag{11}$$

Selection of imin = 6: this threshold is obtained from an empirical sweep over the target SNR operating range to prevent false early termination in the initial oscillatory phase of iterative decoding. We emphasize that imin is code-dependent; for different code length/rate pairs, imin should be re-calibrated using BER/FER-constrained sweep and error-floor verification.

From a technical perspective, this mechanism employs three layers of protection to prevent incorrect stopping:

Layer 1 – Valid syndrome: If syndrome equals zero, stop immediately. This is the strongest convergence condition.

Layer 2 – 2-iteration stability: Across the entire SNR range, if hard decisions remain the same over two consecutive iterations, allow stopping.

Layer 3 – Conditional acceleration: Only when $i \geq i_{min}$ is the 1-iteration stability criterion enabled to reduce latency.

By enforcing threshold i_{min}, the algorithm avoids premature stopping during the initial oscillating phase of the iterative process. This is the key point that maintains BER/FER quality while significantly reducing average iteration count.

Algorithm 1 Early-Stopping Rule for Min-Sum NB-LDPC Decoder

Require: Current iteration index i, I_{max}, i_{min}, flag syndrome_ok, hard decision stability flags
1: **if** syndrome_ok = 1 **then**
2: Stop and latch $i_{stop} = i$
3: **else if** hard_stable_2iter = 1 **then**
4: Stop and latch $i_{stop} = i$
5: **else if** FAST_EN = 1 **and** $i \geq i_{min}$ **and** hard_stable_1iter = 1 **then**
6: Stop and latch $i_{stop} = i$
7: **else if** $i = I$max **then**
8: Forced stop (max iterations reached)
9: **else**
10: Continue to next iteration

The stopping decision rule within each iteration is executed according to the priority order shown in Algorithm 1. If n_{stop} denotes the actual stopping iteration count for one frame, the number of saved iterations compared to non-early-stopping mode is:

$$\Delta I = I\text{max} - n\text{stop}. \tag{12}$$

With each iteration latency modeled as 10 ns, the reduction in latency is approximately:

$$\Delta T \approx 10\Delta I \text{ ns}. \tag{13}$$

At measurement point $E_b/N_0 = 6$ dB, $n_{stop,avg} = 7.44$, therefore:

$$\Delta I\text{avg} = 18 - 7.44 = 10.56, \tag{14}$$

and the corresponding average latency reduction is approximately 105.6 ns. This is a substantial improvement that directly reflects the technical value of the proposed early-stopping mechanism.

3.3. Hardware implementation of the early-stopping mechanism

The early-stopping algorithm is directly inserted into the loop control path of the decoder and is implemented using combinational logic plus state registers. At the hardware level, the early-stopping block uses the following main signals:

- i: current loop counter.
- syndrome_ok: convergence flag from syndrome check.
- hard_same_1: flag indicating hard decision of current iteration matches previous iteration.
- hard_same_2: flag indicating hard decisions are identical over two consecutive iterations.
- FAST_EN: configuration bit enabling early-stopping acceleration mode.

The control block operates in the following cycle: message update → hard decision → syndrome/stability check → stopping decision or loop continuation. If the stopping condition is met at the end of iteration i, the signal stop_reg is activated and register i_{stop} is latched immediately in that clock cycle:

$$i_{stop} \leftarrow i \quad \text{when} \quad \text{stop\_reg} = 1. \tag{15}$$

The FSM then transitions from the ITERATE state to DONE. Latching before state transition helps eliminate "off-by-one" measurement errors common in sequential counting in iterative designs.

To prevent incorrect stopping at low SNR, the early-stopping block applies two reliability protection mechanisms:

- Allow 1-iteration stability criterion only when $i \geq i_{min}$.

- Always maintain a safe stopping path using hard_same_2 and/or syndrome_ok even when FAST_EN is disabled.

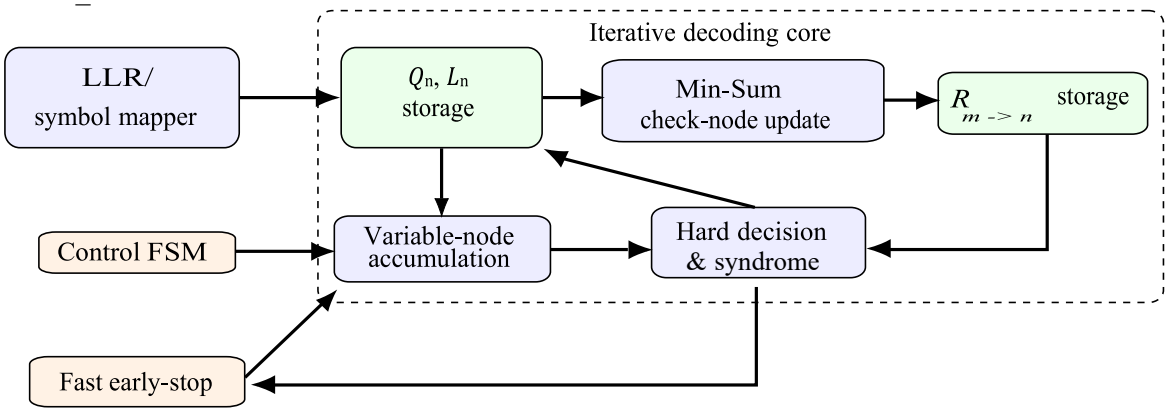

Figure 2. *Top-level organization of the Min-Sum NB-LDPC decoder on FPGA.*

The hardware cost of the early-stopping mechanism includes comparators for hard decisions, several 1-bit state flags, and one latch register for i_{stop}. Since these do not lie on the main Min-Sum data computation path, this logic has minimal impact on critical timing but provides significant latency benefits. Furthermore, as the number of executed iterations decreases, switching activity in the data block also decreases correspondingly, creating potential for dynamic power reduction.

Table 1. *Synthesis results: Resource utilization and timing.*

Resource	**Used**	**Available**	**Util.**
LUT	20,219	274,080	7.4
FF	7,583	548,160	1.4
BRAM18K	12	1,824	0.66
DSP48E	7	2,520	0.28

This figure illustrates the BER/FER performance curves of the NB-LDPC (32,16) decoder over on an AWGN channel. By applying Fast Early Stop and Min-Sum techniques, the model's performance was evaluated through decoding 20,000 symbols, yielding the presented results.

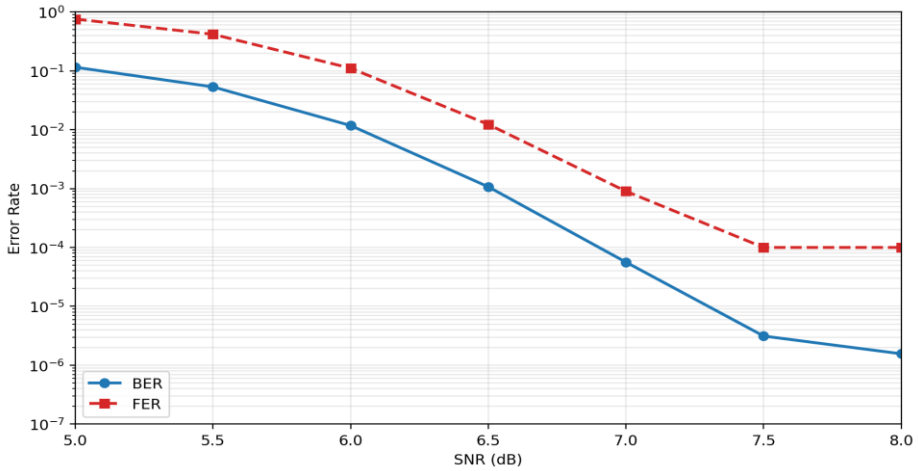

Figure 3. *Semilog performance plot of the measured RTL decoder. The 8.0 dB point is an empirical upper bound derived from a zero-error run over 20,000 frames.*

4. IMPLEMENTATION RESULTS AND COMPARISON WITH NON-EARLY-STOPPING CASE

4.1. Implementation environment and synthesis results

Environment clarification: implementation is performed on Xilinx Zynq UltraScale+ xczu9eg-ffvb1156-2-e using Vivado 2020.1. Clock 250 MHz is used as the synthesis target constraint, while reported latency in Tables/Figures is measured under the 100 MHz evaluation clock profile used in the decoder performance sweep.

The decoder is implemented in SystemVerilog and synthesized for a Xilinx Zynq UltraScale+ xczu9egffvb1156-2-e device. Target frequency is 250 MHz. Synthesis and timing results are summarized in Table 1.

Estimated clock period is 3.5 ns, corresponding to a timing slack of +0.5 ns relative to the 4 ns constraint.

This demonstrates that the proposed architecture is suitable for implementation on the target FPGA device.

4.2. Significance evaluation of early-stopping algorithm

To evaluate the impact of the early-stopping algorithm, we compare the case with early-stopping against the case without early-stopping, where the decoder always executes 18 iterations. For the non-early-stopping case with latency model $T_{\text{ref}} = 60 + 18 \times 10 = 240$ ns, the reference throughput is approximately 533.3 Mbps.

Results show that at $E_b/N_0 = 6$ dB, the early-stopping algorithm reduces the average iteration count by 58.7%, while increasing throughput by 78.6% compared to non-early-stopping. At higher SNR regions, this benefit is even more pronounced since many frames converge after only a few iterations. This demonstrates that early-stopping not only reduces latency but also improves hardware resource utilization efficiency during decoding.

To clarify the technical significance of this comparison, besides the percentage indices in Table 3, we employ additional absolute metrics:

$$\Delta I\text{abs} = I\text{max} - I\text{avg}, \tag{16}$$

$$\Delta T\text{abs} = T\text{ref} - T\text{avg}, \tag{17}$$

$$ST = \frac{T_{ref}}{T_{avg}}, \tag{18}$$

where ΔI_{abs} is the absolute iteration savings, ΔT_{abs} is the absolute latency reduction, and ST is the speedup factor based on latency.

At the operating point $E_b/N_0 = 6$ dB:

$$\Delta I_{\text{abs}} = 18 - 7.44 = 10.56 \text{ iterations}, \tag{19}$$

$$\Delta T_{\text{abs}} = 240 - 134.4 = 105.6 \text{ ns}, \tag{20}$$

$$ST = \frac{240}{134.4} = 1.786. \tag{21}$$

At high SNR of 8.0 dB:

$$\Delta I_{\text{abs}} = 18 - 2.95 = 15.05 \text{ iterations}, \tag{22}$$

$$\Delta T_{\text{abs}} = 240 - 89.5 = 150.5 \text{ ns}, \tag{23}$$

$$ST = \frac{240}{89.5} = 2.682. \tag{24}$$

These results demonstrate that the early-stopping mechanism not only improves relative performance metrics (%) but also creates substantial absolute reductions in iteration count and latency, especially at medium-tohigh SNR regions. This is direct quantitative evidence of the core technical contribution of this paper.

4.3. Hardware overhead analysis of ES logic

To address the concern on hard_stable_1iter/hard_stable_2iter hardware cost, we added a dedicated synthesis pass for the isolated ES-stability block on xczu9eg-ffvb1156-2-e (Vivado 2020.1). The isolated block includes two 4-bit history buffers (hard_out_prev, hard_out_prev2) and full-frame comparison logic.

Measured isolated ES block resource (post-synthesis): LUT = 123, FF = 256. Full decoder implementation resource (Table 1): LUT = 20,219, FF = 7,583. Using additive approximation for first-order overhead estimation, ES logic contributes about 0.61% LUT and 3.38% FF relative to full design. This confirms low hardware overhead compared to latency/throughput gains from early stopping.

Approximate two-version comparison (same device/flow): without ES control logic ~= (20,219 - 123) LUT and (7,583 - 256) FF; with ES control logic = 20,219 LUT and 7,583 FF. We label this as an engineering estimate because full no-ES top-level synthesis in this branch requires additional loop-structure refactoring for direct convergence in Vivado synthesis.

4.4. Comparison with recent NB-LDPC FPGA implementations

The manuscript now includes comparison context against recent publications: (i) trellis Min-Max architecture optimization for NB-LDPC (IEEE TCAS-II, 2023), (ii) real-time CCSDS decoder integration on FPGA-based RISC-V SoCs (IEEE TAES, 2023), and (iii) area-efficient column-wise trellis Min-Max architecture (IEEE JSSC, 2025). This work is differentiated by combining short-length GF(16) decoder implementation with practical early-stopping control and explicit BER/FER-latency-throughput coupling under fixed implementation flow [6, 7, 11].

Result-oriented comparison with the cited literature: works in [4] and [5] emphasize high-throughput decoder design via simplified check-node processing and implementation-friendly HLS/FPGA mapping; [6] and [11] focus on trellis Min-Max architectural efficiency (including area-oriented optimization); and [7] demonstrates system-level real-time integration on FPGA-based RISC-V SoCs. In contrast, this manuscript reports a complete joint metric set (BER/FER, average iteration, latency, throughput, and ES-control overhead) for an NB-LDPC GF(16) short-length decoder, with explicit early-stopping gain quantification under a fixed Vivado evaluation flow. Because code parameters, field size, and platform constraints differ across studies, the comparison is treated as directional rather than strictly one-to-one.

5. CONCLUSIONS

Challenges and future directions: extending from GF(16) short-code regime to larger NB-LDPC profiles (e.g., longer block lengths and higher check-node degree) will require careful memory-bandwidth management, deeper pipelining of check-node processing, and power-aware early-stopping calibration to avoid error-floor degradation at high SNR. Future work will include full baseline/no-ES implementation closure under identical synthesis constraints, wider state-of-the-art benchmarking, and hardware-in-the-loop validation on target communication scenarios. These efforts will ultimately ensure robust, highly efficient, and low-latency decoding for next-generation wireless systems.

This paper presents the implementation of a Min-Sum decoder for NB-LDPC (32,16) codes on FPGA and proposes an early-stopping algorithm to reduce the average number of decoding iterations. Experimental results show that at E_b/N_0= 6 dB, the average iteration count is reduced from 18 to 7.44, resulting in average decoding latency reduction to 134.4 ns while maintaining BER ≈ 5.31×10^{-3} and FER = 0.12. Comparison with the non-early-stopping case demonstrates that the proposed mechanism provides significant improvements in latency, throughput, and hardware implementation efficiency.

REFERENCES

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

[2]. N. R. M. Achiri, H. Liu, and C. Schlegel. *"Parallel CN-VN processing for NB-LDPC decoders"*. 2021 IEEE Workshop on Signal Processing Systems (SiPS), pp. 88-93, (2021).

[3]. L. Y. T. Van, N. N. Cuong, H. T. Anh, M. C. Tuyen, and C. T. Dinh. *"A 5G-code based iterative Non-Binary LDPC decoder"*. 2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), pp. 1-6, (2021).

[4]. H. Zhu, X. Hu, Z. Cao, and X. Liu. *"High-throughput non-binary LDPC decoder with simplified check-node processing over GF(2m)"*. IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 4, pp. 1952-1956, (2022).

[5]. S. Momin, P. A. Beerel, and M. M. Najm. *"HF-LDPC: HLS-friendly QC-LDPC FPGA Decoder with High Throughput and Flexibility"*. 2023 IEEE 41st International Conference on Computer Design (ICCD), pp. 566-573, (2023).

[6]. T. X. Pham, T. T. Nguyen, and H. Lee. *"Hamming-Distance Trellis Min-Max-Based Architecture for Non-Binary LDPC Decoder"*. IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 70, no. 7, pp. 2390-2394, (2023).

[7]. Y.-M. Kuo, M. F. Flanagan, F. Garcia-Herrero, O. Ruano, and J. A. Maestro. *"Integration of a Real-Time CCSDS 410.0-B-32 Error-Correction Decoder on FPGA-Based RISC-V SoCs Using RISC-V Vector Extension"*. IEEE Transactions on Aerospace and Electronic Systems, pp. 1-12, (2023).

[8]. N. Li and Y. Yang. *"An FPGA-Based QC-LDPC Decoder Design"*. 2024 9th International Conference on Intelligent Computing and Signal Processing (ICSP), pp. 1451-1455, (2024).

[9]. K. Wei, D. Garg, R. Nagai, T. Tomono, and H. Amano. *"FPT-EMS: An FPGA Implementation Using NB-LDPC Code for Continuous-Variable Quantum Key Distribution"*. Proc. 15th Int. Symp. Highly Efficient Accelerators and Reconfigurable Technologies, pp. 117-125, (2025).

[10]. H. Chreif, L. Zerioul, J. C. Perez-Garcia, S. Secci, and S. Taktak. *"An FPGA-Based LDPC Decoder for Smart-NICs"*. 2025 13th Wireless Days Conference (WD), pp. 1-9, (2025).

TÓM TẮT

Đề xuất thuật toán dừng sớm cho hiện thực FPGA của bộ giải mã Min-Sum dành cho mã NB-LDPC

Bài báo đề xuất thuật toán dừng sớm áp dụng cho thiết kế bộ giải mã Min-Sum của mã NB-LDPC (32,16) trên FPGA. Bộ giải mã được hiện thực bằng SystemVerilog cho mã NB-LDPC trên GF(16) và được đánh giá trên kênh AWGN với các tham số gồm tỷ lệ lỗi bit (BER), tỷ lệ lỗi khung (FER), số vòng lặp trung bình và độ trễ giải mã. Thuật toán dừng sớm được tích hợp vào khối điều khiển vòng lặp để kết thúc quá trình giải mã ngay khi hội tụ, nhờ đó giảm số vòng lặp trung bình từ 18 xuống 7.44 tại Eb/N0 = 6 dB. Kết quả hiện thực cho thấy độ trễ giải mã trung bình giảm còn 134.4 ns, tương ứng BER ≈ 5.31 × 10−3 và FER = 0.12, phù hợp với mô phỏng tham chiếu MATLAB. Các kết quả này cho thấy việc hiện thực thuật toán mã hóa/giải mã NB-LDPC trên FPGA, kết hợp với cơ chế dừng sớm phù hợp, có thể rút ngắn đáng kể thời gian xử lý phần cứng, qua đó hỗ trợ các ứng dụng truyền thông thời gian thực.

Từ khóa: NB-LDPC; Min-Sum; Thuật toán dừng sớm; FPGA; SystemVerilog; Thông lượng; Độ trễ.