A Novel FEC Implementation for VSAT Terminals Using High-Level Synthesis

Najmeh Khosroshahi *, Ron Mankarious , M. Reza Soleymani

Posted Date: 14 January 2026

doi: 10.20944/preprints202601.1080.v1

Keywords: FPGA implementation; High-Level Synthesis (HLS); IP core generation; LDPC decoding; QC-LDPC codes; satellite communication; VSATPlus® system

Article

A Novel FEC Implementation for VSAT Terminals Using High-Level Synthesis

Najmeh Khosroshahi [1,2], Ron Mankarious [2] and M. Reza Soleymani [3]

[1] Department of Electrical and Computer Engineering, Concordia University, Montréal, QC, Canada
[2] PolarSat Inc., Montréal, QC, Canada
[3] Department of Electrical and Computer Engineering, Concordia University, Montréal, QC, Canada
* Correspondence: najmeh.khosroshahi@mail.concordia.ca

Abstract

This paper presents a hardware-aware field-programmable gate array (FPGA) implementation of a layered 2-dimensional corrected normalized min-sum (2D-CNMS) decoder for quasi-cyclic low-density parity-check (QC-LDPC) codes in very small aperture terminal (VSAT) satellite communication systems. The main focus of this work is leveraging Xilinx Vitis high-level synthesis (HLS) to design and generate an LDPC decoder IP core based on the proposed algorithm, enabling rapid development and portability across FPGA platforms. Unlike conventional NMS and 2D-NMS algorithms, the proposed architecture introduces dyadic, multiplier-free normalization combined with two-level magnitude correction, achieving near-belief propagation (BP) performance with reduced complexity and latency. Implemented entirely in HLS and integrated in Vivado, the design achieves real-time operation on Zynq UltraScale+ multiprocessor system-on-chip (MPSoC) with throughput of 116-164 Mbps at 400 MHz and resource utilization of 8.7K-22.9K LUTs, 2.6K-7.5K FFs, and zero DSP blocks. Bit-error-rate (BER) results show no error floor down to 10^{-8} across additive white gaussian noise (AWGN) channel model. Fixed scaling factors are optimized to minimize latency and hardware overhead while preserving decoding accuracy. These results demonstrate that the proposed HLS-based 2D-CNMS IP core offers a resource-efficient, high-performance solution for multi-frequency time division multiple access (MF-TDMA) satellite links.

Keywords: FPGA implementation; High-Level Synthesis (HLS); IP core generation; LDPC decoding; QC-LDPC codes; satellite communication; VSATPlus® system

1. Introduction

Hubless full-mesh very small aperture terminal (VSAT) communication systems are increasingly adopted in satellite networks for their resilience against single points of failure, support for direct VSAT-to-VSAT connectivity, and enhanced security through private network isolation [52]. Unlike traditional star-configured VSAT architectures that rely on a central hub, hubless full-mesh systems eliminate hub dependency and enable direct remote-to-remote communication, which is critical for modern satellite applications such as broadband connectivity, IoT backhaul, and emergency communications. Reliable satellite communication requires robust forward error correction (FEC) to mitigate noisy channel conditions and maintain data integrity. Conventional FEC schemes such as turbo product codes (TPCs) [64], convolutional codes [59,61–63], and Reed–Solomon (RS) codes [59–61] have been widely deployed. However, low-density parity-check (LDPC) codes, and particularly quasi-cyclic LDPC (QC-LDPC) codes, have gained prominence due to their superior error-correction capability and hardware efficiency. QC-LDPC codes exploit structural regularity in the parity-check matrix, reducing memory and interconnect complexity while achieving near-optimal performance [50]. These advantages have led to their adoption in modern standards such as DVB-S2 [59], IEEE 802.11n (WiFi) [55], and 5G NR [58]. The sum-product (SP) algorithm, also known as belief propagation (BP),

 doi:10.20944/preprints202601.1080.v1

remains the performance benchmark for LDPC decoding [3], but its probability-domain arithmetic is computationally expensive for resource-constrained FPGAs. Simplified variants such as min-sum (MS) [2,35] and normalized min-sum (NMS) reduce complexity by replacing multiplications with add-compare-select operations, albeit at the cost of some coding gain. Further refinements, including second-minimum approximation (SAMS) [7], two-dimensional MS (2D-MS) [32], and two-dimensional normalized min-sum (2D-NMS) [30], improve accuracy with modest overhead. Traditional register-transfer-level (RTL)-based implementations of LDPC decoders are time-consuming, error-prone, and difficult to scale across evolving FPGA platforms. High-Level Synthesis (HLS) offers a modern design methodology that accelerates development, improves portability, and enables rapid design-space exploration without sacrificing hardware efficiency. For FPGA targets such as VSATPlus®, decoder microarchitecture strongly impacts throughput, latency, and power. Fully parallel layered designs achieve high per-iteration throughput but incur dense interconnect and tight timing closure [36]. Partially parallel designs reduce wiring and power at the cost of lower throughput [37], while pipelined block-serial decoders minimize area but increase latency [38]. Scheduling also influences convergence and memory organization: flooding requires more iterations and higher energy, whereas layered scheduling reduces iterations but demands careful memory banking [10,11]. Despite these advances, existing FPGA implementations struggle to balance error-correction performance, resource utilization, and power efficiency under stringent constraints typical of satellite systems. This gap motivates the need for an approach that combines algorithmic improvements with a scalable, hardware-aware design methodology leveraging HLS. To address this challenge, this work proposes a layered 2D-Corrected Normalized Min-Sum (2D-CNMS) decoder integrated into an HLS-based design flow for generating reusable LDPC IP cores. The proposed architecture incorporates dyadic, multiplier-free normalization and dual-scaling of the first and second minima, achieving near-BP performance while minimizing resource usage and power consumption.

The main contributions of this paper are as follows: (1) A hardware-aware LDPC decoder architecture based on the 2D-CNMS algorithm optimized for FPGA implementation; (2) An HLS-based design flow for generating reusable LDPC IP cores, reducing development complexity compared to traditional RTL approaches; (3) Comprehensive evaluation of post-implementation error-correction performance, resource utilization, and throughput under realistic satellite channel conditions. The remainder of this paper is organized as follows: Section 2 describes the VSATPlus® system and its architectural constraints. Sections 3 and 4 present the proposed LDPC decoder and algorithmic enhancements. Section 5 details the FPGA-oriented architecture. Section 7 reports simulation results and analyzes complexity-performance trade-offs. Finally, Section 8 concludes the paper and outlines future research directions.

2. VSATPlus System Overview

The VSATPlus [66] system provides full-mesh, single-hop connectivity within a satellite network, leveraging multi-frequency time division multiple access (MF-TDMA) technology. Each terminal buffers user data and transmits it in short, high-speed bursts scheduled to avoid overlap, ensuring efficient satellite resource utilization without requiring a central hub for scheduling. This hubless architecture supports diverse satellite applications, including broadband connectivity, IoT backhaul, emergency communications, and enterprise networking, where reliability and scalability are critical. Figure 1 illustrates one example of VSATPlus deployment in air traffic control (ATC), showing direct IP connectivity between area control centers (ACCs) and airports across regions. Similar principles apply to other mission-critical satellite services requiring secure, low-latency communication.

 doi:10.20944/preprints202601.1080.v1

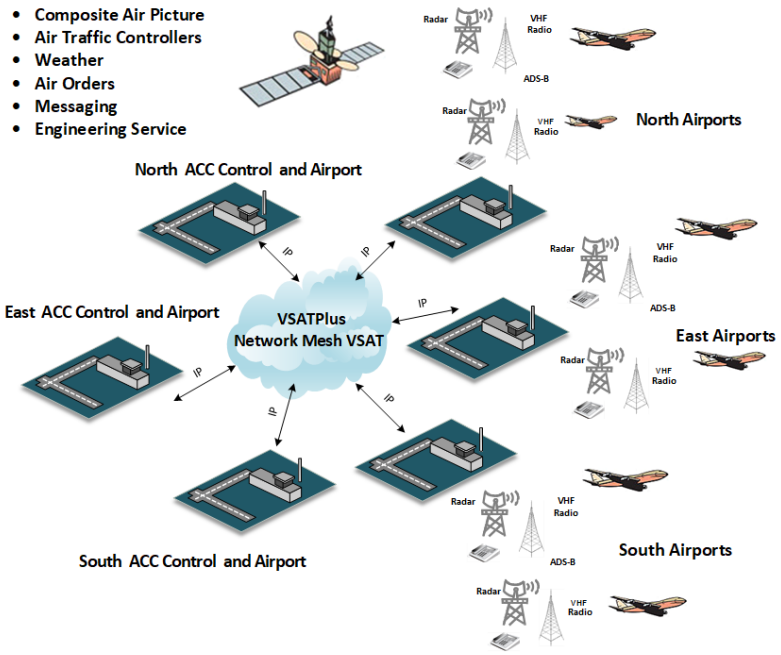


Figure 1. Example VSATPlus full-mesh, hubless deployment for ATC applications.

The VSATPlus modem is highly agile, capable of varying multiple parameters burst-by-burst. This flexibility enables carrier-to-carrier hopping for both transmit and receive over 32 carriers, along with adaptive modulation and FEC changes. This capability, referred to as Mesh-ACM^{TM}, dynamically adjusts modulation and coding to accommodate different terminal antenna sizes and varying channel conditions across the network, optimizing throughput and reliability [53]. Figure 2 shows the architecture of a VSATPlus full-mesh, hubless network employing Ku-band and C-band satellite links for high-throughput communication. The satellite facilitates signal transmission from ground stations operating on both frequency bands, enabling simultaneous multi-band connectivity.

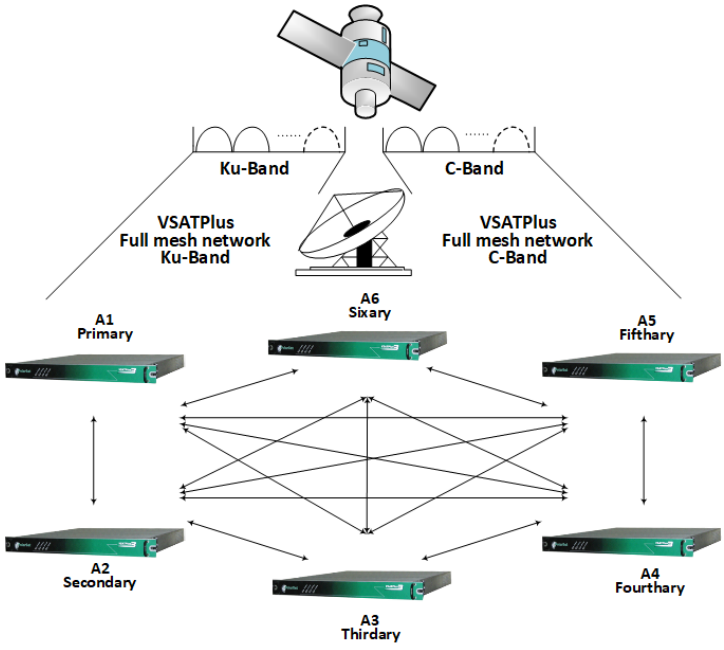


Figure 2. VSATPlus full-mesh, hubless network architecture for satellite communication.

The network consists of six nodes (A_1 to A_6), interconnected in a full-mesh topology to ensure direct, bidirectional communication between all nodes. This architecture demonstrates the robustness

 doi:10.20944/preprints202601.1080.v1

and scalability of VSATPlus for distributed satellite applications beyond ATC, including broadband and enterprise networking.

3. Decoding Algorithmic Principles

QC-LDPC codes apply parity-check matrices tiled by $Z \times Z$ submatrices that are either circulant permutations of the identity or zeros [29]. This regular structure yields compact representations and lower memory requirements while enabling highly parallel, hardware-efficient decoders—an attractive performance/complexity trade-off for high-throughput satellite links. In satellite communication systems, where long propagation delays and limited link budgets make re-transmissions costly, efficient LDPC decoding is essential to maintain reliability and throughput. Although QC-LDPC codes may be degree-irregular (row and column weights (d_c, d_v) can vary), the locations of ones in H are deterministically specified by the permutation matrix and its circulant shifts, producing predictable, conflict-free access patterns rather than random placement [17]. Furthermore, the QC form organizes H as a regular array of blocks, with Z distinct memory banks for variable node (VN) data, enabling blockwise schedules that achieve scalable parallelism. A compressed permutation matrix, illustrated in Table 1, stores only shift offsets, where dash marks a zero block and non-negative entries specify the cyclic shift of the identity, minimizing footprint and simplifying decoder configuration. Owing to these advantages, QC-LDPC codes are widely adopted in modern communication standards.

Table 1. Permutation Matrix of QC-LDPC Code with R = 2/3 [65]

37	-	-	32	19	27	30	-	4	-	-	-	-	-	21	33	0	0	-	-	-	-	-
36	-	35	-	20	22	-	1	-	13	-	39	-	-	39	-	-	0	0	-	-	-	-
-	-	-	-	-	-	-	20	17	18	18	29	7	1	22	-	-	-	0	0	-	-	-
-	17	32	-	-	16	28	-	26	-	-	10	-	20	-	7	-	-	-	0	0	-	-
5	12	-	7	29	11	-	-	-	-	12	11	30	-	-	-	0	-	-	-	0	0	-
-	-	6	23	-	-	4	14	37	-	-	-	-	35	2	26	-	-	-	-	-	0	0
27	2	27	-	6	-	19	-	-	36	33	-	22	-	-	-	-	-	-	-	-	-	0
-	38	-	7	-	-	-	33	-	21	14	-	33	15	-	25	0	-	-	-	-	-	-

The decoding of QC-LDPC codes typically employs BP algorithms, which are broadly classified based on the nature of the message information exchanged during iterations. Hard-decision decoders, such as the bit-flipping (BF) algorithm, operate on binary decisions extracted directly from the channel [39]. Conversely, soft-decision decoding techniques, including the sum-product (SP) algorithm and its computationally simplified variant, the min-sum (MS) algorithm, leverage soft information in the form of log-likelihood ratios (LLRs) [26,68]. These iterative message-passing algorithms exchange extrinsic information between variable nodes (VNs) and check nodes (CNs) over a bipartite graph representation of the parity-check matrix H, known as a Tanner graph [27]. This graph-based framework underpins the inference mechanism that progressively refines the reliability of decoded bits with each iteration.

The structure of the LDPC parity-check matrix H used in this work is derived from the IEEE 802.16e (WiMAX) standard. The matrix is constructed from a permutation matrix, shown in Table 1, composed of integer shift values and null entries, expanded by a lifting factor Z. The resulting H matrix is sparse and quasi-cyclic as illustrated in Figure 3. Each diagonal or off-diagonal band in the figure corresponds to a circulant permutation matrix. Their distribution reflects the structured connectivity of VNs (columns) and CNs (rows).

 doi:10.20944/preprints202601.1080.v1

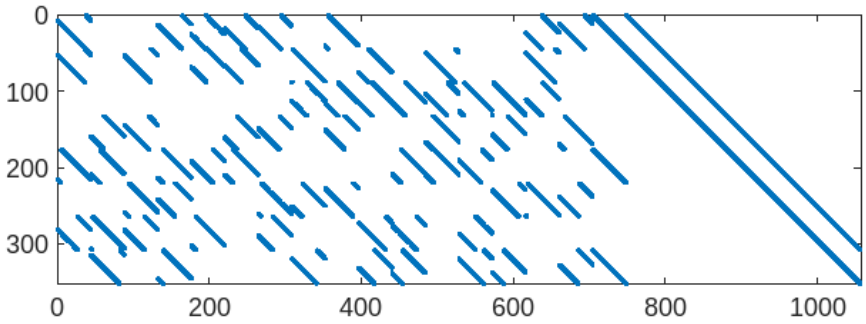


Figure 3. Sparsity pattern of the parity-check matrix H [65].

3.1. Sum-Product / Belief Propagation

The SP decoding algorithm, also known as the BP [30], initiates by calculating the LLRs for each received channel symbol r_i at time i under an AWGN channel model, as follows:

$$LLR_i = \log \frac{\Pr(r_i|t_i = 1)}{\Pr(r_i|t_i = -1)} = \frac{2r_i}{\sigma^2/2} \tag{1}$$

where σ^2 is the channel noise power following the Gaussian distribution. Initially, messages passed from variable node v to the connected check node c are set to the LLR value derived from the received symbol, as given by:

$$L_{v\to c} = \sum_{c'\in\mathcal{M}(v)\setminus c} L_{c'\to v} \tag{2}$$

where $c' \in \mathcal{M}(v) \setminus c$ represents the set of check nodes connected to the variable node v, excluding c. Next, the check node information is updated as follows:

$$L^{(\ell)}_{c\to v} = 2\tanh^{-1}\Big(\prod_{v'\in\mathcal{N}(c)\setminus v} \tanh\frac{L^{(\ell-1)}_{v'\to c}}{2}\Big) \tag{3}$$

Once the convergence condition is satisfied or the maximum number of iterations is reached, the operation terminates. Subsequently, the hard decision for each variable node, v, is computed based on its LLR:

$$L_v = LLR + \sum_{c'\in\mathcal{M}(v)} L_{c'\to v} \tag{4}$$

If $L_v \geq 0$, then the estimated transmitted bit value is 0, otherwise it is considered as 1.

3.2. Min-Sum

While BP achieves excellent error-correction performance, its reliance on nonlinear functions and probability-domain operations results in high computational complexity, motivating the development of simplified algorithms such as Min-Sum. The MS decoding algorithm simplifies the computational complexity of the BP algorithm by replacing nonlinear hyperbolic tangent functions with a minimum operation. Thus, the check node update equation simplifies to:

$$L^{(\ell)}_{c\to v} = \Big(\prod_{v'\in\mathcal{N}(c)\setminus v} \operatorname{sign}(L^{(\ell-1)}_{v'\to c})\Big) \min_{v'\in\mathcal{N}(c)\setminus v} \Big|L^{(\ell-1)}_{v'\to c}\Big| \tag{5}$$

The variable nodes update and hard decision processes remain consistent with the BP algorithm.

3.3. Normalized Min-Sum

Although the MS significantly reduces hardware resource requirements and processing latency compared to the BP algorithm, it experiences degradation in BER performance, typically around $0.5-1$ dB [39]. This degradation primarily results from overestimation in extrinsic information exchange

 doi:10.20944/preprints202601.1080.v1

between nodes. To mitigate this effect, normalized min-sum (NMS) applies scaling factor, $\beta \in (0,1)$, to correct extrinsic information estimates, thereby enhancing practical decoding performance [24]:

$$L_{c\to v}^{(\ell)} = \Big(\prod_{v' \in \mathcal{N}(c)\setminus v} \operatorname{sign}\big(L_{v'\to c}^{(\ell-1)}\big) \Big)\, \beta \min_{v' \in \mathcal{N}(c)\setminus v} \big| L_{v'\to c}^{(\ell-1)} \big| \tag{6}$$

Whereas NMS attenuates the check-node minima by multiplying the raw minimum magnitude by a constant $0 < \beta < 1$, offset min-sum (OMS) instead applies a fixed subtraction to correct for the systematic over-estimation introduced by the plain min-sum rule [25]. In OMS, each check-to-variable message is computed as:

$$L_{c\to v}^{(\ell)} = \Big(\prod_{v' \in \mathcal{N}(c)\setminus v} \operatorname{sign}\big(L_{v'\to c}^{(\ell-1)}\big) \Big) \max\Big[\Big(\min_{v' \in \mathcal{N}(c)\setminus v} \big| L_{v'\to c}^{(\ell-1)} \big| \Big) - \eta,\ 0 \Big] \tag{7}$$

where $\eta > 0$ is the offset parameter chosen to minimize the performance loss relative to the full belief-propagation algorithm [25]. Because the offset operation requires only a single subtraction per edge rather than a multiplication, OMS offers reduced critical-path latency and lower hardware resource usage compared to NMS.

3.4. 2-Dimensional Normalized Min-Sum

In the standard NMS algorithm, a single constant scaling factor compensates the overestimated magnitudes of all incoming check-to-variable messages. However, the extrinsic magnitude sent to a VN equals the smallest incoming magnitude, $\min_1$, for all edges except the argmin edge, which uses the second-smallest incoming magnitude, $\min_2$, due to edge exclusion. Consequently, applying one scale for both cases can miscalibrate some messages and degrade performance. The 2D-NMS addresses this by applying distinct scaling (β_1, β_2), conditioned on whether $\min_1$ or $\min_2$ is used, yielding more accurate LLRs with only modest added complexity [31]:

$$L_{c\to v}^{(\ell)} = \Big(\prod_{v' \in \mathcal{N}(c)\setminus v} \operatorname{sign}(L_{v'\to c}^{(\ell-1)}) \Big) \times \begin{cases} \beta_2 \cdot \min_2 \\ \beta_1 \cdot \min_1 \end{cases} \tag{8}$$

For LDPC decoders employing the 2D-NMS algorithm, the optimal normalization factors (β_1, β_2), are predominantly determined by the check node degree (d_c), which is determined by the parity check matrix.

3.5. 2-Dimensional Min-Sum

While 2D-NMS compensates the check-node magnitude bias via the scaling pair (β_1, β_2), it applies the same gain to both message flows. Density-evolution analysis in [32] shows that the residual bias on the variable-to-check stream differs statistically from that on the check-to-variable stream. Therefore, introducing a second, direction-specific scaling pair eliminates this asymmetry and closes the remaining gap to SP decoding with only two extra multiplications per edge. To address this residual asymmetry more effectively, [32] proposes applying iteration-dependent scale factors $(\alpha^{(\ell)}, \beta^{(\ell)})$ directly to the extrinsic LLRs on each edge, resulting in:

$$\tilde{L}_{v\to c}^{(\ell)} = \alpha^{(\ell)}\, L_{v\to c}^{(\ell)} \tag{9}$$

While the corresponding check-to-variable updates is as follows:

$$\tilde{L}_{c\to v}^{(\ell)} = \Big(\prod_{v' \in \mathcal{N}(c)\setminus v} \operatorname{sign}(\tilde{L}_{v'\to c}^{(\ell-1)}) \Big) \beta^{(\ell)} \min_{v' \in \mathcal{N}(c)\setminus v} \big| \tilde{L}_{v'\to c}^{(\ell-1)} \big| \tag{10}$$

 doi:10.20944/preprints202601.1080.v1

Although $(\alpha^{(\ell)}, \beta^{(\ell)})$ can be optimized adaptively based on the code structure and channel SNR, [32] demonstrates that exhaustive AWGN profiling across the operational E_b/N_0 range yields two fixed scale factors that preserve virtually all of the SP decoder's extrinsic-information fidelity.

4. 2-Dimensional Corrected Normalized Min-Sum

For VSATPlus terminals, even modest coding-gain improvements translate into reduced power-amplifier back-off and increased fade margins. However, the SP decoding algorithm remains computationally impractical for the throughput and power envelopes of the target FPGA platforms. Building on [31,32], we propose a two-dimensional corrected normalized min–sum (2D-CNMS) decoder that preserves the add–compare–select arithmetic of MS while recovering a substantial fraction of the residual gap to SP algorithm. The update introduces two shift-based per-edge scalings per iteration and incurs no additional on-chip memory, thereby maintaining low architectural complexity. Therefore, the scaled variable-to-check update is defined as:

$$\tilde{L}_{v\to c}^{(\ell)} = \bar{\alpha}\, L_{v\to c}^{(\ell)} \tag{11}$$

While, the scaled check-to-variable update is given by:

$$\tilde{L}_{c\to v}^{(\ell)} = \Big(\prod_{v'\in\mathcal{N}(c)\setminus v} \operatorname{sign}(\tilde{L}_{v'\to c}^{(\ell-1)}) \Big) \times \begin{cases} \bar{\beta}_2 \cdot \min_2 \\ \bar{\beta}_1 \cdot \min_1 \end{cases} \tag{12}$$

Similar to 2D-NMS algorithm, $\min_1$ and $\min_2$ denote the smallest and second-smallest absolute values of the incoming messages $|\tilde{L}_{v'\to c}^{(\ell-1)}|$, where $v' \in \mathcal{N}(c) \setminus v$.

5. 2D-CNMS Hardware Implementation

Hand-optimized RTL design has traditionally been the standard approach for FPGA-based LDPC decoders. However, High-Level Synthesis (HLS) methodologies [16] and commercial toolchains such as Vivado HLS [67] now offer a more productive alternative by enabling designers to specify the decoder in C/C++ or SystemC and automatically generate synthesizable RTL. In practice, HLS significantly reduces design and verification effort [15], while achieving area, timing, and power results comparable to expert-tuned HDL implementations [15]. Furthermore, a single high-level source is portable across FPGA families and integrates seamlessly with the VSATPlus C-based simulation environment, facilitating rapid design-space exploration and system-level validation. Algorithm 1 summarizes the layered scheduling strategy employed in the proposed 2D-CNMS decoder. Compared to conventional layered MS, the proposed approach introduces a uniform variable-to-check scaling factor, $\bar{\alpha}$, and a two-level check-to-variable scaling pair $(\bar{\beta}_1, \bar{\beta}_2)$, conditioned on whether the outgoing message is derived from $\min_1$ or $\min_2$. These refinements improve extrinsic information accuracy while preserving the low-complexity arithmetic of MS.

 doi:10.20944/preprints202601.1080.v1

Algorithm 1 Layered 2D-CNMS QC-LDPC Decoder

Kernel 1: Initialization
Pipeline directive
Load $\mathrm{LLR}_v^{\mathrm{ch}}$ into VN accumulator memory
Initialize message memory
Kernel 2: Process Decoding Iterations
while $i = 1 : \mathrm{Itr}_{max}$ **do**
    Kernel 2.1: Compute VN Incoming Messages
    for $h = 1 : M/Z$ **do**
        # Pipeline directive
        for all check-nodes c in layer h **do**
            Load $L_{v\to c}^{(i)} : v \in \mathcal{N}(c)$ from VN–message RAM
            for all $v \in \mathcal{N}(c)$ **do**
                # Unroll directive
                $\mathrm{sign}(L_{c\to v}^{(i)}) = \prod_{v' \in \mathcal{N}(c)\setminus v} \mathrm{sign}(L_{v'\to c}^{(i-1)})$
                $|L_{c\to v}^{(i)}| = \begin{cases} \bar{\beta}_1 \cdot \min_{1\,\{v' \in \mathcal{N}(c)\setminus v\}} |L_{v'\to c}^{(i-1)}| \\ \bar{\beta}_2 \cdot \min_{2\,\{v' \in \mathcal{N}(c)\setminus v\}} |L_{v'\to c}^{(i-1)}| \end{cases}$
                Write $L_{c\to v}^{(i)}$ to CN–message RAM
            end for
        end for
    end for
    Kernel 2.2: Generate VN Output Messages
    for $u = 1 : N/Z$ **do**
        # Pipeline directive
        for all variable-node v in layer u **do**
            Load $L_{c\to v}^{(i)} : c \in \mathcal{M}(v)$ from CN–message RAM
            for all $c \in \mathcal{M}(v)$ **do**
                # Unroll directive
                $L_{v\to c}^{(i)} = L_v^{(i-1)} + \bar{\alpha} \sum_{c' \in \mathcal{M}(v)\setminus c} L_{c'\to v}^{(i-1)}$
                Write $L_{v\to c}^{(i)}$ to VN–message RAM
            end for
        end for
    end for
end while
Kernel 3: Compute VNs Hard Decision
for all $v = 1 : N$ **do**
    # Pipeline directive
    $\hat{c}_v \leftarrow 0$ if $L_v^{(i)} < 0$ else 1
end for
return Decoded Word, $\hat{c}_v$

The proposed decoder is implemented using Vivado HLS 2024, leveraging loop-level parallelism and pipelining to maximize throughput. Inner loops responsible for CN and VN updates are partially or fully unrolled to generate multiple CN→VN and VN→CN messages per cycle, while outer loops are pipelined to minimize the initiation interval (II), enabling iteration overlap and reducing latency. The inherent layered structure of QC-LDPC codes facilitates deterministic memory banking and partitioning across Z dual-port block RAMs (BRAMs), supporting concurrent read–modify–write operations aligned with layer boundaries. Address generation is performed using two ROM-based lookup tables (LUTs) precomputed from the base matrix: $\mathcal{N}(c)$ for CN updates and $\mathcal{M}(v)$ for VN updates. These index tables enable constant-time routing without additional buffering or pointer chasing, ensuring efficient memory access.

The HLS-based 2D-CNMS decoder architecture, illustrated in Figure 4, comprises two fully parallel processing engines: one dedicated to CN message computation and the other to VN message computation. Each engine interfaces directly with dual-port BRAM banks under the control of a compact ROM storing non-negative cyclic-shift offsets and connection degrees derived from the base matrix.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 January 2026 doi:10.20944/preprints202601.1080.v1

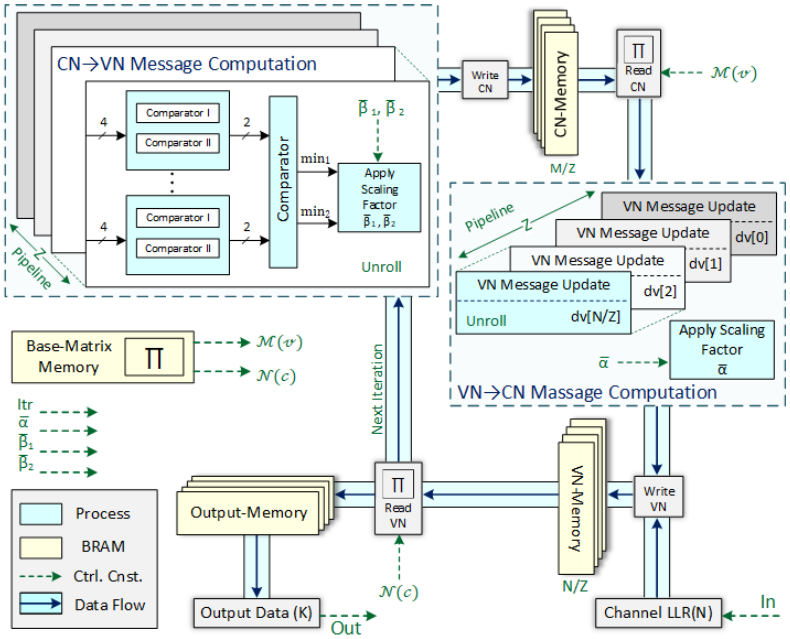


Figure 4. FPGA-based architecture of 2D-CNMS decoder.

In the CN engine, which comprises M/Z parallel pipelines, each pipeline retrieves a Z-wide vector of extrinsic LLRs from the VN RAM, applies a fixed-depth tree of four-input comparators to identify the smallest and second-smallest magnitudes, determines the overall output sign through a single XOR reduction stage, and adjusts these minima using normalization factors $(\bar{\beta}_1, \bar{\beta}_2)$. The normalized extrinsic messages are then written back to the CN-RAM via the dual-port interface. Similarly, the VN engine, consisting of N/Z parallel pipelines, reads the updated CN messages along with the original channel LLRs, accumulates them using an adder tree, subtracts each corresponding extrinsic to form new messages, scales the result by $\bar{\alpha}$, applies eight-bit saturation, and generates provisional hard decisions based on the sign. Both the updated soft-value vectors and hard-decision bits are subsequently written to the VN RAM and output RAM, respectively.

6. End-to-End Vivado Block Design

Figure 5 illustrates the integrated encoder–channel–decoder chain instantiated in Vivado by introducing external IP cores implemented using Vitis HLS 2024. Although the LDPC IP core was validated at a clock frequency of 400 MHz, the design employs a 100 MHz system clock generated by the Clocking Wizard IP and distributed via the Processor System Reset module. Each frame begins with a single-cycle start pulse, and all AXI4-Stream interfaces operate without backpressure, with the `ready` signal held asserted.

A pseudorandom source generates independent and identically distributed (i.i.d.) Bernoulli bits using a linear feedback shift register (LFSR) method. After AXI4-Stream handshakes, the QC-LDPC encoder accepts the payload and emits a serialized codeword. An RTL binary phase-shift keying (BPSK) modulator maps bits to signed 8-bit symbols with values $\{+127, -128\}$. Additive noise is produced by an external AWGN IP implementing the Box–Muller transform (see Appendix B). Variance and seed registers are programmed via AXI4-Lite memory-mapped writes issued by the Zynq UltraScale+ processing system (PS), enabling deterministic initialization and run-time reconfiguration of the programmable logic (PL) core. The noise stream is buffered in a FIFO to decouple producer and consumer timing. On readout, the noise is added to the BPSK waveform, saturated to $[-128, 127]$, and treated as the input LLR stream for the decoder. Similar to AWGN parameters, the iteration limit, $\mathrm{Itr}_{\max}$, is also configured by a control register for each frame.

 doi:10.20944/preprints202601.1080.v1

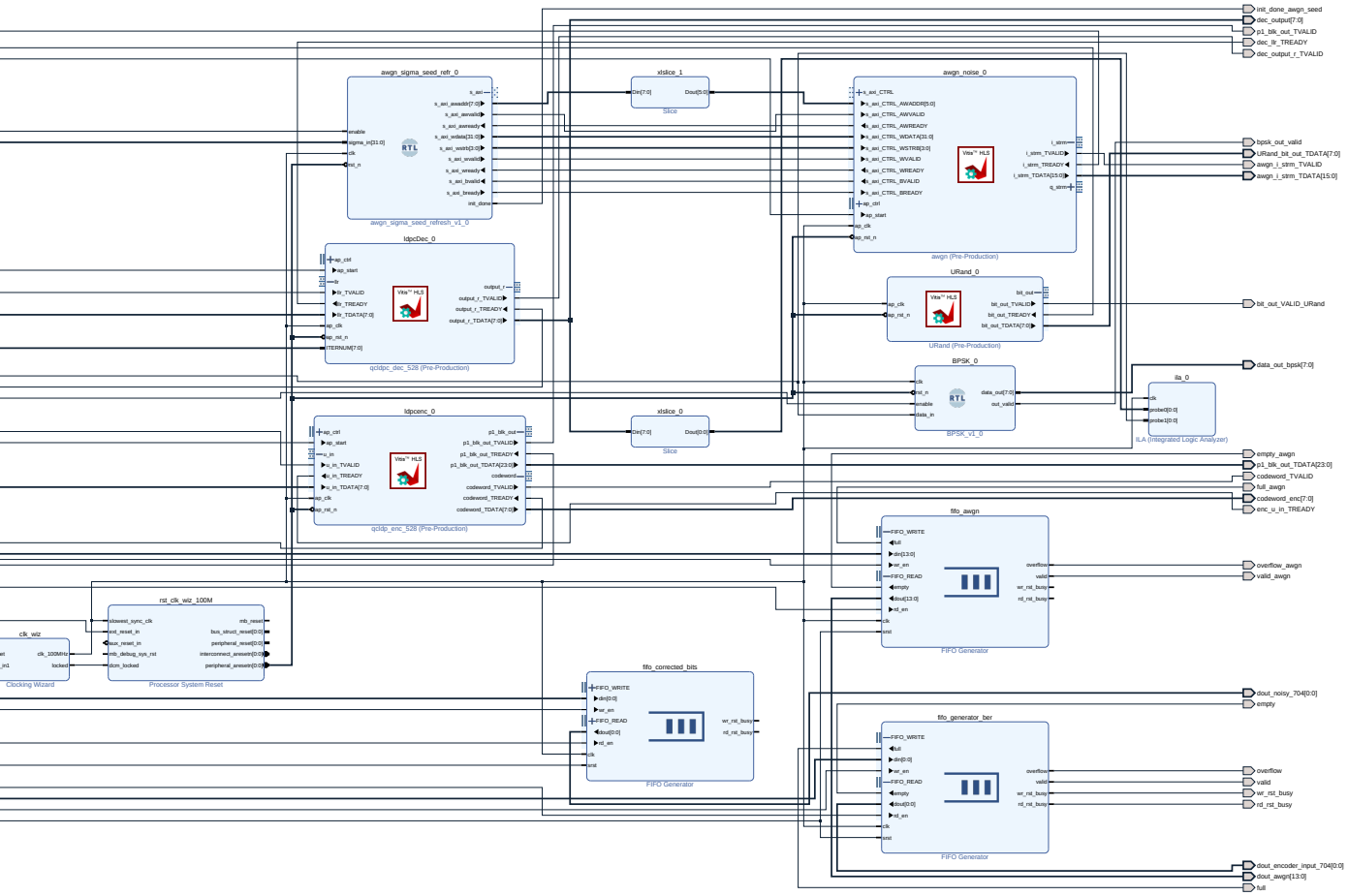


Figure 5. Vivado 2024 end-to-end block design integrating HLS-generated IP cores.

During transmission, the reference bit for each payload position is stored in a reference FIFO. In parallel, a channel hard decision is derived from the sign of the saturated received sample and stored in a second FIFO to measure uncoded performance. When decoded bits become available, they are compared against the reference FIFO to accumulate the post-decoder error count, while the channel-hard stream is compared against the same reference to count bit flips caused by noise.

7. Emulation Results

We evaluate decoding algorithms under a layered schedule on the QC-LDPC code $(1056, 704)$ with $\mathrm{Itr}_{\max} = 50$ using 8-bit fixed-point LLRs (clip/saturate) and syndrome-based early termination. The channel is AWGN and E_b/N_0 is swept on a uniform grid with sufficient frames per point to probe down to BER $\leq 10^{-8}$. All baselines use identical quantization, stopping criteria, and layered scheduling for a fair comparison. Unlike conventional NMS, the proposed 2D-CNMS introduces dual-direction scaling and normalization refinements that significantly improve convergence speed and error-floor performance while maintaining FPGA-friendly complexity.

According to [31], the optimal two-level CN normalization factors (β_1, β_2) decrease with increasing check-node degree (d_c) and approach constants at high SNR. Specifically, for the regular codes with $d_c = 6$ and SNR $\in [0, 4]$ dB, $\beta_1 \in [0.50, 0.90]$ and $\beta_2 \in [0.27, 0.50]$, trending toward the upper end as SNR rises. Consistently, [32] reports nearly iteration-invariant scalings for IEEE 802.11n $(1944, 1296)$, $R = 2/3$ as $\bar{\alpha} \approx 0.9007$ and $\bar{\beta} \approx 0.8973$, achieving BER within 0.02 dB of SP algorithm

 doi:10.20944/preprints202601.1080.v1

over $E_b/N_0 \in [1.4, 2.6]$ dB. In line with [1], the strongest single-scalar NMS baseline uses a fixed normalization schedule:

$$\beta^{(\ell)} = 3/4 + 2^{-(\ell+1)} \text{ if } 2 \le \ell \le 5, \text{ else } 3/4 \tag{13}$$

where ℓ represents the number of iterations.

These observations support fixed, iteration-invariant gains. Guided by coarse-to-fine emulation sweeps on the target IEEE 802.16e QC-LDPC codes (rates 1/2, 2/3, 3/4), we therefore map (β_1, β_2) to $\bar{\beta}_1 = 0.8125$ and $\bar{\beta}_2 = 0.875$, and set the VN gain to $\bar{\alpha} = 0.75$. All scalings are implemented via shift-and-subtract to eliminate multipliers and extra memory.

7.1. Performance and Complexity Comparison

Figure 6 compares full-BP, layered-BP, NMS ($\beta = 0.75$) [1,24], 2D-MS ($\bar{\alpha} = \bar{\beta} \approx 0.899$) [39], 2D-NMS $(\beta_1, \beta_2) = (0.75, 0.875)$ [31], and the proposed 2D-CNMS $(\bar{\alpha}, \bar{\beta}_1, \bar{\beta}_2) = (0.75, 0.8125, 0.875)$ on QC-LDPC code $(1056, 704)$ with $\text{Itr}_{max} = 50$. Across the operational E_b/N_0 range, 2D-CNMS closely follows layered-BP and consistently outperforms NMS, 2D-NMS, and 2D-MS. While NMS and 2D-CNMS are comparable in the waterfall region, 2D-CNMS exhibits a lower error floor in the high-SNR, outperforming even the layered-BP reference. While BP remains the theoretical benchmark, its computational complexity makes FPGA implementation impractical. 2D-CNMS achieves near-BP performance without multipliers or dividers, making it suitable for real-time FPGA deployment.

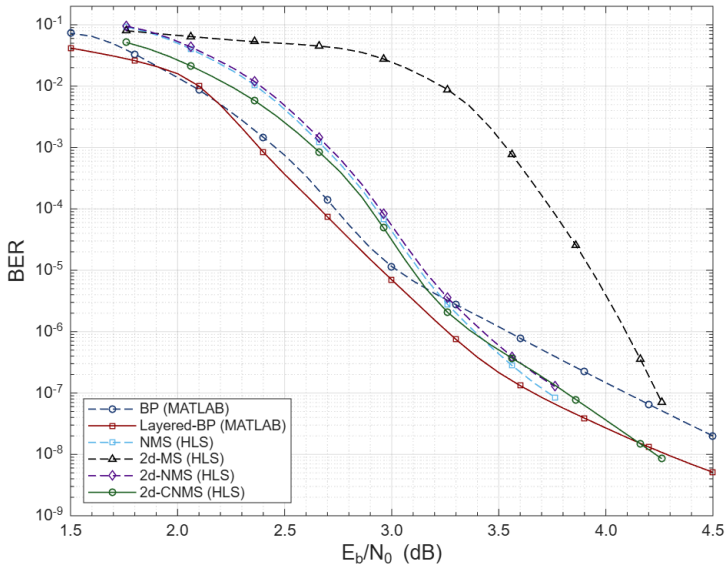
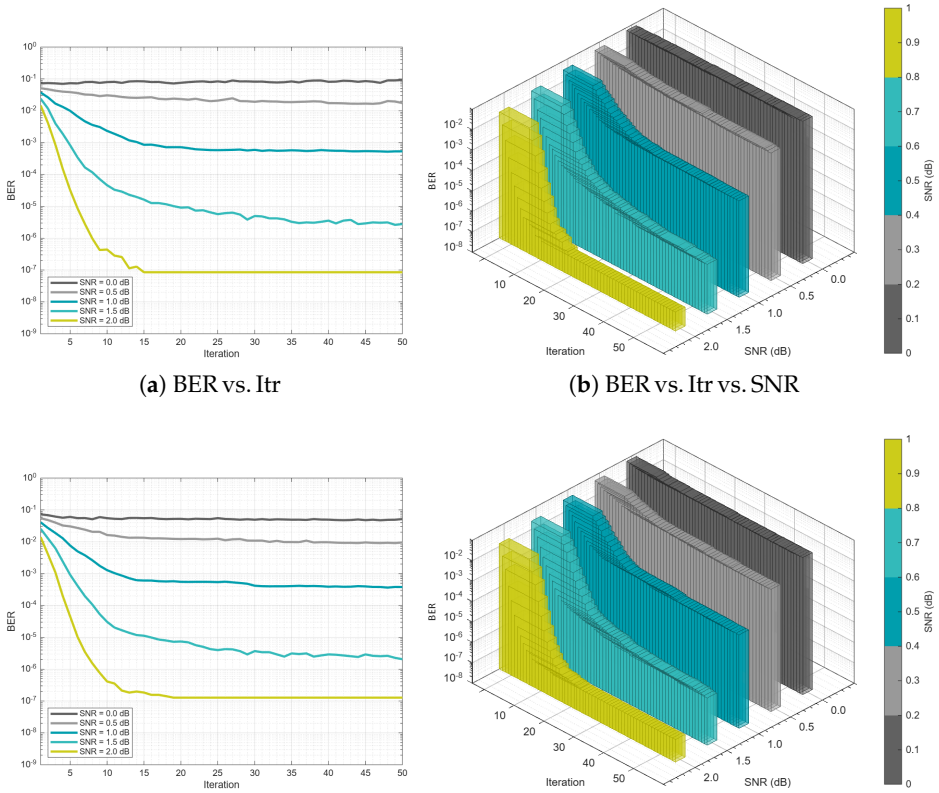


Figure 6. Performance comparison of various decoding algorithms with $\text{Itr}_{\max} = 50$ for QC−LDPC $(1056, 704)$.

Figure 7 reports BER versus decoder iterations for the $(1056, 704)$ code under BPSK/AWGN. At SNR = 2.0 dB, the proposed 2D-CNMS achieves BER $\le 10^{-7}$ by approximately 12 iterations, whereas the conventional NMS (with $\beta = 0.75$) requires roughly 15 iterations. For SNR $\le$ 1.5 dB, neither method reaches 10^{-7} within 50 iterations. Nevertheless, 2D-CNMS maintains a uniformly lower BER for the same iteration budget and, at SNR = 1.5 dB, attains BER $\approx 10^{-6}$ in the mid–30s iterations while NMS does not achieve 10^{-6} within 50 iterations. These iteration savings translate directly to reduced latency and higher throughput (see Section 7.3).

 doi:10.20944/preprints202601.1080.v1



(**a**) BER vs. Itr (**b**) BER vs. Itr vs. SNR

(**c**) BER vs. Itr (**d**) BER vs. Itr vs. SNR

Figure 7. BER performance versus SNR and iteration count. (a),(b): conventional NMS. (c),(d): proposed 2D-CNMS.

Figure 8 presents the BER performance of the proposed 2D-CNMS across IEEE 802.16e QC-LDPC code profiles. In contrast to the MATLAB layered-BP reference in Figure 6, the 2D-CNMS maintain a sustained waterfall region with no observable error floor over the simulated range, reaching BER $< 10^{-8}$.

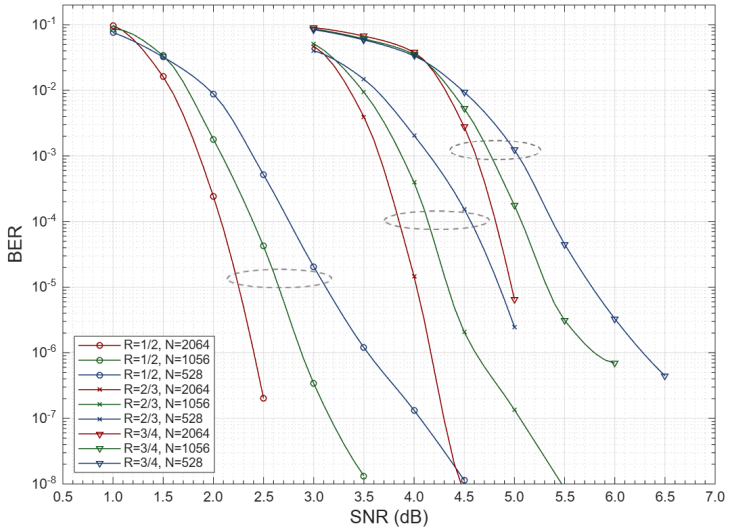


Figure 8. Performance analysis of QC−LDPC codes under 2D−CNMS decoding algorithm with different code rates.

Table 2 contrasts the arithmetic requirements of SP, MS, NMS, OMS, 2D-MS, 2D-NMS, and the proposed 2D-CNMS on a per-check-node basis. SP demands d_c multiplications and d_c-1 divisions. 2D-MS retains 2 multipliers. 2D-CNMS removes multipliers and dividers and keeps the comparator

 doi:10.20944/preprints202601.1080.v1

count $d_c + \log_2 d_c - 2$. Normalization is implemented by 3 subtractions and 12 bit-shifts. The resulting data path is DSP-free and FPGA-native while recovering near-SP message fidelity. In effect, 2D-CNMS trades multiplications/divisions for lightweight comparisons and shifts.

Table 2. Computational Complexity of a Check-node within a Single Iteration

Ref.	Algorithm	Mult.	Div.	Comparison	Add.	Sub.	Bit-shift	FPGA Impl.	Remarks
[3]	SP	d_c	$d_c - 1$	0	1	0	0	✗	$\tanh$, $\tanh^{-1}$
[23]	MS	0	0	$d_c + \log_2 d_c - 2$	0	0	0	✗	–
[24]	NMS	1	0	$d_c + \log_2 d_c - 2$	0	0	0	✗	–
[25]	OMS	0	0	$d_c + \log_2 d_c - 2$	0	1	0	✗	–
[45]	NMS	0	0	$d_c + \log_2 d_c - 2$	4	0	12	✓	$\beta = 0.75$
[32]	2D−MS	2	0	$d_c + \log_2 d_c + 2$	0	0	0	✗	*adaptive*
[39]	2D-MS	2	0	$d_c + \log_2 d_c - 2$	0	0	0	✓	$\bar{\alpha} = \bar{\beta} = 0.899$
[31]	2D−NMS	2	0	$d_c + \log_2 d_c - 2$	0	0	0	✗	–
[25]	2D−NMS	2	0	$d_c + \log_2 d_c - 2$	0	0	0	✗	$\beta_1 = 0.75, \beta_2 = 0.875$
Our work	2D−CNMS	0	0	$d_c + \log_2 d_c - 2$	0	3	12	✓	$\bar{\alpha} = 0.75$ $\bar{\beta_1} = 0.8125$ $\bar{\beta_2} = 0.875$

7.2. Resource Utilization

Table 3 benchmarks layered 2D-CNMS against representative HLS and hand-tuned FPGA decoders for WiMAX, Wi-Fi, and DVB-S2 standards under a uniform 100 MHz target and a fixed optimized iteration. The proposed instances deliver 29-41 Mbps coded-bit throughput with 0 DSP usage and moderate memory footprints near 97-217 BRAM-18K.

Table 3. Previous LDPC Decoder Implementations on Various FPGA Platforms

Method	Type	Decoder	Schedule	Std.	N	Rate	Itr.	Γ(Mbps)	Logic	RAM
[21]	HLS	MS	Flooding	802.16*e*	768	1/2	10	99	147,000	1,641
[21]	HLS	MS	Flooding	802.16*e*	1,152	1/2	10	104	161,000	1,703
[21]	HLS	MS	Flooding	802.16*e*	1,536	1/2	10	82	134,000	1,461
[21]	HLS	MS	Flooding	802.16*e*	1,920	1/2	10	81	144,000	1,516
[41]	HLS	MS	Flooding	802.11*n*	648	1/2	3	27	9,072	28
[42]	FPGA	MS	Layered	802.11*n*	1,944	1/2	10	21	187,000	1,349
[44]	FPGA	MS	Layered	802.11*n*	648	1/2	6	281	35,668	81
[47]	FPGA	MS	Layered	custom	1,024	1/2	30	-	25,035	72
[46]	FPGA	OMS	Layered	802.16*e*	576	1/2	20	33	8,685	106
[46]	FPGA	OMS	Layered	802.16*e*	2,304	1/2	10	307	34,104	281
[19]	FPGA	MS	Layered	802.11*n*	972	1/2	4	2,476	159,000	831
[22]	FPGA	OMS	Layered	802.11*n*	1,944	1/2	4	608	33,351	120
[43]	FPGA	OMS	Layered	802.11*n*	1,944	1/2	8	608	33,351	102
[48]	FPGA	MS	Flooding	802.16*e*	2,304	1/2	20	290	< 15,520	76
[45]	FPGA	OMS	Layered	802.11*n*	648	1/2	5	77	19,332	3
[49]	FPGA	OMS	Layered	DVB-S2	16,384	1/2	10	2,130	49,121	50

[1] Logic: LUTs (Xilinx) or ALMs (Intel Altera). [2]RAM: 18 K-bit BRAM (Xilinx) or 20 K-bit M20K (Altera).

Considering (15), throughput scales approximately linearly with the clock for fixed initiation interval and bandwidth, selecting a higher target clock in Vitis HLS (which packages the blocks as Vivado IP) proportionally increases system throughput. Retargeting the same design to $f_{\text{clk}} = 400$ MHz with II $= 1$ yields a $\times 4$ speedup. A comparator–adder–shift data path plus a layered schedule provides a predictable cost–throughput operating point without multiplier-based scaling. The IEEE 802.16e permutation-matrix geometries are 12×24 for rate 1/2, 8×24 for rate 2/3, and 16×48 for rate 3/4. These geometries directly set storage depth and intra-layer parallelism for each instance.

 doi:10.20944/preprints202601.1080.v1

Table 4 summarizes post-synthesis utilization and timing across multiple N, K settings and code rates. All designs remain DSP-free. LUT and FF counts stay modest in the ranges 8.7 k-22.9 k and 2.6 k-7.5 k. BRAM demand scales with the permutation geometry up to 217 at rate 3/4. Measured decoding latency follows the analytic model Λ_N in (14), with per-iteration costs C_{iter} that match the scheduled kernels. Achieved coded-bit throughput at 100 MHz spans 29-41 Mbps. These results confirm that the HLS formulation compiles to efficient RTL with stable timing and predictable scaling, while preserving the algorithmic benefits of two-dimensional dyadic normalization.

Table 4. Post-RTL Synthesis Optimized Resource Utilization of 2D-CNMS QC-LDPC Decoder

	R = 1/2			R = 2/3			R = 3/4		
Resource	(528, 264)	(1056, 528)	(2064, 1032)	(528, 352)	(1056, 704)	(2064, 1376)	(528, 396)	(1056, 792)	(2064, 1548)
LUT	10,264	8,668	8,793	10,649	11,406	9,735	21,874	22,974	20,172
FF	3,398	2,644	2,837	3,766	3,832	3,230	7,487	7,321	5,933
DSP	0	0	0	0	0	0	0	0	0
BRAM	0	95	97	0	0	0	0	0	217
URAM	0	0	0	0	0	0	0	0	0
SRL	211	0	0	235	0	0	298	262	0
LATCH	0	0	0	0	0	0	0	0	0
CLB	0	0	0	0	0	0	0	0	0
SLICE	0	0	0	0	0	0	0	0	0
Latency	1,777	3,094	5,866	1,865	3,270	6,210	1,579	2,833	5,092
C_{iter}	65	100	184	65	100	184	43	65	98
Γ (Mbps)	30	35	36	29	33	34	34	38	41

[1] Target clock period: 10.000 ns, Optimized iteration: 15; [2] Decoding latency is based on clock-cycle @ 100MHz; [3]Equation (14)

7.3. Decoding Latency Analysis

Decoding latency is measured in clock cycles to process one full codeword. Under non-overlapped AXI4-Stream I/O the total latency equals sequential input of all channel LLRs, sequential output of the final hard decisions, a fixed I/O and pipeline overhead, and an iteration-dependent compute term. For code rate 2/3 defined by an 8×24 permutation matrix, see Table 1, the lifting size is $Z = N/24$ and the information length is $K = 2/3 \times N$. Calibrated to the HLS schedule the per-iteration compute cost scales approximately linearly with Z. For a run with Itr_{max} decoding iterations the latency in cycles is:

$$\Lambda_N = N + K + \delta_{\text{pipe}} + Itr_{\text{max}} \times C_{\text{iter}}(Z) \tag{14}$$

with δ_{pipe} the fixed pipeline and I/O overhead and $C_{\text{iter}}(Z)$ the scheduled check-node and variable-node kernel cost per iteration. From HLS reports, as addressed in Table 4, at rate 2/3 we conservatively upper-bound $\delta_{\text{pipe}} = 10$ cycles, $C_{\text{iter}}(22) = 65$ cycles for QC-LDPC (528, 352), and $C_{\text{iter}}(44) = 100$ cycles for (1056, 704).

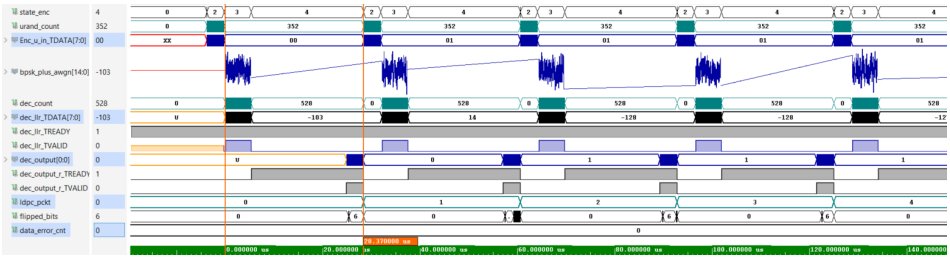
Given a clock frequency F_{clk} in MHz the coded-bit throughput in Mbps is:

$$\mathcal{T} = \frac{N \times F_{clk}}{\Lambda} \quad Mbps \tag{15}$$

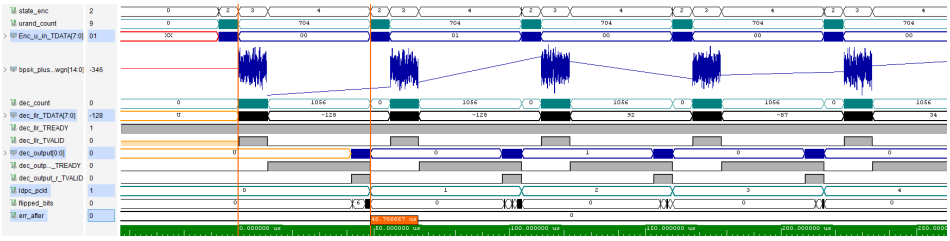
while the information-bit throughput equals $R \times \mathcal{T}$.

Figure 9 shows RTL simulation traces via Xilinx Vivado 2024. The left cursor aligns to the first valid AXI4-Stream LLR beat and the right cursor to the last valid hard-decision beat. The signal `dec_count` ramps over N samples during input, the decoder executes $\text{Itr}_{max} = 32$ iterations, and the design streams out K bits. A bounded fill and drain overhead is visible in the AXI handshakes and is consistent with $\delta_{\text{pipe}} \leq 10$ cycles.

 doi:10.20944/preprints202601.1080.v1



(**a**) IEEE802.16e QC-LDPC $(528, 352)$, $\mathrm{Itr}_{max} = 32$



(**b**) IEEE802.16e QC-LDPC $(1056, 704)$, $\mathrm{Itr}_{max} = 32$

Figure 9. 2D-CNMS decoder latency analysis of QC-LDPC instances with $F_{clk} = 100$ MHz via Xilinx Vivado 2024.

Equation (14) gives $\Lambda_{528} = 2970$ cycles and $\Lambda_{1056} = 4970$ cycles while Vivado reports 2037 cycles and 4876 cycles, respectively. The RTL values lie within the analytic bound and are shorter due to scheduler conservatism and the absence of backpressure.

8. Conclusion

This work presented an end-to-end FPGA implementation of a QC-LDPC decoder for VSATPlus systems using Vitis HLS and Vivado 2024. The primary contribution lies in demonstrating a practical design methodology for mapping advanced LDPC decoding techniques onto FPGA hardware through HLS-driven flows, enabling rapid prototyping and deployment without manual RTL development. The proposed architecture employs shift-and-subtract operations to eliminate multipliers and dividers, resulting in a DSP-free design optimized for resource-constrained satellite terminals.

The complete system integration was achieved using HLS-generated IP cores for the encoder, AWGN channel, and decoder. Under a 100 MHz and 400 MHz clock, the design achieves a coded-bit throughput of 29–41 Mbps and 116–164 Mbps, respectively, with moderate memory usage. Post-implementation Vivado power analysis reports an estimated dynamic power of 116 mW at 100 MHz for the encoder and 1.948 W at 400 MHz for the decoder, confirming suitability for low-power VSAT platforms. These values are consistent with reported figures for FPGA-based LDPC decoders of comparable block size (2064 bits) and code rates (1/2, 2/3, and 3/4) in the literature, where dynamic power typically ranges between 1.5 W and 3 W for high-throughput designs operating at 300–500 MHz. Fixed scaling factors were selected through empirical optimization to balance BER performance and hardware simplicity, avoiding the complexity overhead of adaptive scaling with negligible performance penalty.

Performance evaluation under AWGN, which is appropriate for GEO VSATPlus links with negligible Doppler and stable fading, shows near-BP decoding performance and no observable error floor down to BER $< 10^{-8}$. Compared to MS-family baselines, the implemented design achieves faster convergence and lower iteration counts, translating into reduced latency and higher throughput.

Future work will extend this HLS-based methodology to adaptive coding and modulation (ACM) and AI-assisted decoding for non-Gaussian channels, while incorporating detailed power and thermal profiling under realistic traffic conditions. These directions aim to further validate the practicality of HLS-driven FPGA deployment for next-generation satellite communication systems.

 doi:10.20944/preprints202601.1080.v1

Appendix A. QC-LDPC Encoder

The QC-LDPC encoder implemented in this work is based on the IEEE C802.16e−04/373r1 standard [65], which introduces a harmonized definition of QC-LDPC codes for the OFDMA physical layer, developed collaboratively by six major industry contributors including Intel, Motorola, Nokia, Nortel, Samsung, and Texas Instruments. The proposed LDPC codes are based on structured base matrices expanded into large parity-check matrices using circulant permutation matrices. This method utilizes a recursive structure with circular shifts and XOR operations facilitating streamlined hardware implementation and efficient computational logic on FPGA platforms. The standard supports multiple code-rates (1/2, 2/3, and 3/4) and a wide range of block sizes through scalable expansion, shortening, and puncturing techniques making it well-suited for modern, high-throughput communication systems of VSATPlus satellite communications system.

Appendix A.1. VSATPlus Encoder Algorithm

The VSATPlus QC-LDPC encoder leverages the Direct Encoding Method II as specified in the IEEE C802.16e-04/373r1 standard [65]. This approach is grounded in a structured matrix-partitioning technique that facilitates deterministic, low-complexity encoding suitable for hardware implementation. As illustrated in Figure A1, the parity-check matrix H is decomposed into six submatrices: $A_{(N_p-g)\times N_k}$, $B_{(N_p-g)\times g}$, $C_{g\times N_k}$, $D_{g\times g}$, $E_{g\times(N_p-g)}$, and $T_{(N_p-g)\times(N_p-g)}$, with T being a lower-triangular matrix characterized by ones along its main diagonal.

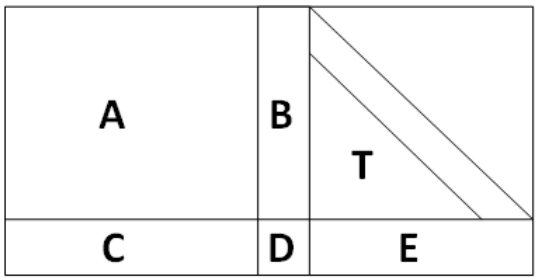


Figure A1. Structure of the QC-LDPC parity-check matrix implemented for VSATPlus based on IEEE 802.16e [65].

In the QC-LDPC selected for implementation and evaluation in this work, the instantiated parameters (N_p, N_k, g) are: (12, 12, 1), (8, 16, 1), and (12, 36, 1) for $R = 1/2, 2/3, 3/4$, respectively, where N_p and N_k denote the permutation-matrix rows and columns with lifting factor $Z = N/(N_k+N_p)$. The instantiate block lengths in our experiments are $\{528, 1056, 2064\}$ with the three code rates.

Given a systematic input vector u, the encoding procedure begins with the computation of intermediate products Au^T and Cu^T, followed by the application of the inverse transform $T^{-1}(Au^T)$. The first set of parity bits, denoted p_1, is then computed according to:

$$p_1^T = Cu^T + E^T T^{-1}(Au^T), \tag{A1}$$

which incorporates contributions from both the C and E matrices. Subsequently, the second set of parity bits, p_2, is obtained using:

$$p_2^T = T^{-1}(Au^T + Bp_1^T), \tag{A2}$$

where the input u and partial parity p_1 are jointly encoded via the lower-triangular transformation. A high-level depiction of this block-wise encoding flow is provided in Figure A2.

 doi:10.20944/preprints202601.1080.v1

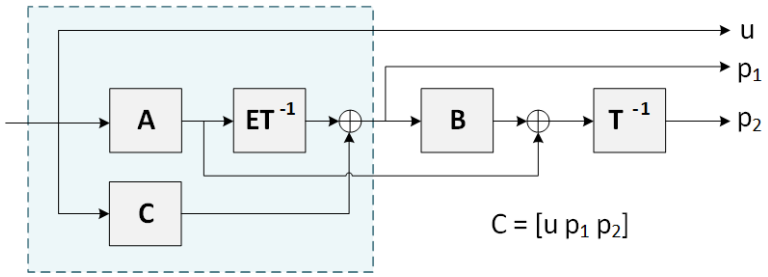


Figure A2. Block diagram of the IEEE 802.16e QC-LDPC encoder architecture [65].

The matrix sparsity and hierarchical decomposition yield a high-throughput, pipelined encoder design with minimal logic complexity. For a detailed derivation and theoretical justification of (A1) and (A2), refer to [65].

Appendix A.2. FPGA Implementation of VSATPlus Encoder

The QC-LDPC encoder is implemented using Vitis HLS and synthesized for the Xilinx Zynq UltraScale+ device with a target system clock of 100 MHz. The implementation leverages extensive high level synthesis optimizations including loop pipelining, aggressive unrolling, and complete array partitioning. These transformations enable deep parallelism across matrix-vector computations and circular shift operations, particularly within encoding stages. A detailed summary of the post-RTL optimized resource utilization, latency, and throughput for the specific QC-LDPC codes is presented in Table A1.

Table A1. QC-LDPC Enecoder Resource Utilization.

	R = 2/3			
Resource	(528, 352)	(1056, 704)	(2064, 1376)	Avail.
LUT	1,128	2,032	3,668	117,120
FF	1,079	1,940	3,570	234,240
DSP	2	2	2	1,248
BRAM−18K	0	0	0	288
URAM	0	0	0	64
SRL	58	61	66	–
LATCH	0	0	0	–
CLB	0	0	0	–
SLICE	0	0	0	–
Latency	911	1,792	3,473	–
$\Gamma\,(Mbps)$	58	59	60	–

[1]UltraRAM: 64-blocks, 288 Kb each. [2]Latency: In clock cycles at@100MHz.

Figure A4 illustrates RTL simulation traces of the QC-LDPC encoder at $F_{clk} = 100$ MHz in Vivado 2024. The left cursor is aligned to the first valid AXI4-Stream input beat, and the right cursor to the last valid output beat on the codeword port. Two counters ramps over the $K = 704$ accepted information bits, and the $N = 1056$ emitted code bits. The output path exhibits no backpressure and only a bounded fill/drain overhead is visible in the handshakes. The measured span between cursors is 1787 cycles at 100 MHz which is aligned with the Vitis HLS report provided in Table A1.

The simulation also verify the Gaussian noise source used for downstream link tests.

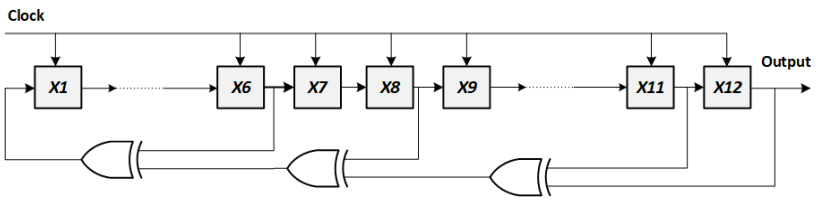


Figure A3. 12-bit LFSR with 1-bit output used in the Box-Muller method to generate i.i.d. random variable.

 doi:10.20944/preprints202601.1080.v1

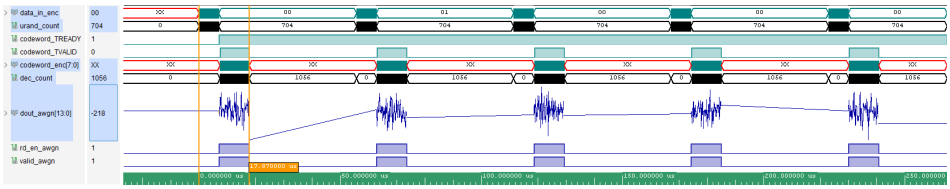


Figure A4. Encoder latency analysis of QC-LDPC $(1056, 704)$ with $F_{clk} = 100$ MHz via Xilinx Vivado 2024.

Appendix B. FPGA Implementation of an AWGN Channel

To evaluate QC-LDPC code in communication link, a dedicated AWGN noise generator is synthesized using Vitis HLS and integrated into the FPGA design via IP instantiation in Xilinx Vivado Design Suite. The generator is clocked at 100MHz and implements the Box-Muller transformation using fixed-point arithmetic and LUT-based approximations [51]. The Box-Muller method converts two statistically independent and identically distributed (i.i.d.) random variables $U_1, U_2 \in (0, 1)$ into a pair of independent, standard normally distributed variables.

$$\mathcal{N}_{\mathcal{I}} = \sqrt{-2 \ln U_1} \cdot \cos(2\pi U_2) \tag{A3}$$

$$\mathcal{N}_{\mathcal{Q}} = \sqrt{-2 \ln U_1} \cdot \sin(2\pi U_2) \tag{A4}$$

where, $\mathcal{N}_{\mathcal{I}}, \mathcal{N}_{\mathcal{Q}} \sim \mathcal{N}(0, 1)$. The FPGA-based AWGN generator is architected for cycle-accurate, pipelined synthesis of zero-mean Gaussian noise samples. To minimize hardware complexity and maximize throughput, computationally intensive functions, such as logarithmic operation, are replaced with high-resolution fixed-point LUTs generated offline using Python, which map i.i.d. random variables to radius values $R = \sqrt{-2 \ln U_1}$, and $\cos(2\pi U_2)$. Address generation for the LUTs is driven by two decorrelated 12-bit LFSRs, seeded independently and defined by the primitive polynomial of $g(X) = X^{12} + X^{11} + X^8 + X^6$ as illustrated in Figure A3. This primitive polynomial ensures maximal-length pseudo-random sequences. Each LFSR output is truncated to 10 bits, enabling indexed access to LUTs of size 1024, corresponding to a 10-bit address space. To maintain statistical zero-mean behavior, a DC offset correction stage is implemented using a programmable exponential moving average (EMA) applied to the noise sample. The DC bias estimate, $b[n]$, is iteratively updated and he DC-corrected output is then computed as follows:

$$\mathcal{N}^{\text{centered}}[n] = \mathcal{N}[n] - b[n+1] \tag{A5}$$

$$b[n+1] = b[n] + \frac{1}{2^{\zeta}}(\mathcal{N}[n] - b[n]) \tag{A6}$$

where ζ is a configurable smoothing coefficient. The bias-corrected sample is subsequently saturated to the 14-bit signed output range $[-8192, +8191]$. The entire architecture is fully pipelined, delivering one valid output sample per clock cycle.

On Xilinx Zynq UltraScale+ devices, the ARM-based PS provides the control plane for the AWGN IP, exposing all configuration knobs through an AXI4-Lite register file while the I/Q samples stream over AXI4-Stream. In our design, the PS programs noise standard deviation, EMA shift controlling the bias-removal time constant, ζ in (A6), and the independent seeds that decorrelate the two LFSR-driven LUT address generators.

Table A2 summarizes post-synthesis resource usage for the AWGN IP-core, comparing HLS estimates to Vivado results. Overall utilization on the target device is very low, with single-digit DSP and BRAM usage, no URAM.

 doi:10.20944/preprints202601.1080.v1

Table A2. AWGN IP-Core Design Resource Utilization.

	HLS		Vivado		
Resource	Used	Util. (%)	Used	Util. (%)	Avail.
LUT	1,545	1.30	4,325	3.68	117,120
FF	515	0.21	2,744	1.16	234,240
DSP	8	0.64	4	0.32	1,248
BRAM-18K	3	1.04	1	0.35	288
URAM	0	-	0	-	64
SRL	0	-	66	-	-
LATCH	0	-	0	-	-
CLB	0	-	0	-	-

[1]UltraRAM: 64-blocks, 288 Kb each.

References

1. J. Li, P. Zhang, L. Wang, and G. Wang, An FPGA-Based LDPC Decoder with Optimized Scale Factor in NMS Decoding Algorithm, *SSRN Electronic Journal*, 2022, doi: 10.2139/ssrn.4087994.
2. M. P. C. Fossorier, M. Mihaljevic, and H. Imai Reduced complexity iterative decoding of low-density parity check codes based on belief propagation, *IEEE Trans. Communications*, vol. 47, no. 5, pp. 673–680, May 1999, doi: 10.1109/26.768759.
3. D. J. C. MacKay Good error-correcting codes based on very sparse matrices *IEEE Trans. Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999, doi: 10.1109/18.748992.
4. Q. Wang, Q. Liu, S. Wang, L. Chen, H. Fang, L. Chen, Y. Guo, and Z. Wu Normalized Min-Sum Neural Network for LDPC Decoding *IEEE Trans. Cognitive Communications and Networking*, vol. 9, no. 1, pp. 70–81, Mar. 2023, doi: 10.1109/TCCN.2022.3212438.
5. D. Oh and K. K. Parhi Min-Sum Decoder Architectures With Reduced Word Length for LDPC Codes *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 57, no. 1, pp. 105–115, Jan. 2010, doi: 10.1109/TCSI.2009.2016171.
6. A. Verma and R. Shrestha Low computational-complexity SOMS-algorithm and high-throughput decoder architecture for QC–LDPC codes *IEEE Trans. Vehicular Technology* , vol. 72, no. 1, pp. 66–80, Jan. 2023, doi: 10.1109/TVT.2022.3203802.
7. H. Lopez, H.-W. Chan, K.-L. Chiu, P.-Y. Tsai, and S.-J. J. Jou A 75-Gb/s/mm^2 and energy-efficient LDPC decoder based on a reduced complexity second minimum approximation min-sum algorithm *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 926–939, Apr. 2020, doi: 10.1109/TVLSI.2019.2955925.
8. S. Yun, B. Y. Kong, and Y. Lee Area- and energy-efficient LDPC decoder using mixed-resolution check-node processing *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 999–1003, Mar. 2022, doi: 10.1109/TCSII.2021.3110953.
9. J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu Reduced-complexity decoding of LDPC codes *IEEE Trans. Communications*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005, doi: 10.1109/TCOMM.2005.852852.
10. M. M. Mansour A turbo-decoding message-passing algorithm for sparse parity-check matrix codes *IEEE Trans. Signal Processing*, vol. 54, no. 11, pp. 4376–4392, Nov. 2006, doi: 10.1109/TSP.2006.880240.
11. T. J. Richardson and R. L. Urbanke The capacity of low-density parity-check codes under message-passing decoding *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001, doi: 10.1109/18.910577.
12. M. Zhao, X. Zhang, L. Zhao, and C. Lee Design of a high-throughput QC–LDPC decoder with TDMP scheduling *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 62, no. 1, pp. 56–60, Jan. 2015, doi: 10.1109/TCSII.2014.2362661.
13. K. Zhang, X. Huang, and Z. Wang A high-throughput LDPC decoder architecture with rate compatibility *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 58, no. 4, pp. 839–847, Apr. 2011, doi: 10.1109/TCSI.2010.2089551.
14. A. Darabiha, A. Chan Carusone, and F. R. Kschischang Power reduction techniques for LDPC decoders *IEEE Journal of Solid-State Circuits*, vol. 43, no. 8, pp. 1835–1845, Aug. 2008, doi: 10.1109/JSSC.2008.925402.
15. Y. Delomier, B. Le Gal, J. Crenne, and C. Jego Model-based design of flexible and efficient LDPC decoders on FPGA devices *Journal of Signal Processing Systems*, vol. 92, no. 7, pp. 727–745, 2020, doi: 10.1007/s11265-020-01519-0

16. G. Martin and G. Smith High-Level Synthesis: Past, Present, and Future *IEEE Design & Test of Computers*, vol. 26, no. 4, pp. 18–25, 2009, doi: 10.1109/MDT.2009.83.
17. B. Le Gal, C. Jego, and C. Leroux, A flexible NISC-based LDPC decoder *IEEE Trans. Signal Processing*, vol. 62, no. 10, pp. 2469–2479, May 2014, doi: 10.1109/TSP.2014.2311964.
18. Y.-F. Zhang, L. Sun, and Q. Cao, TLP-LDPC: Three-level parallel FPGA architecture for fast prototyping of LDPC decoder using high-level synthesis *Journal of Computer Science and Technology*, vol. 37, no. 6, pp. 1290–1306, 2022, doi: 10.1007/s11390-022-1499-9.
19. S. Mhaske, H. Kee, T. Ly, A. Aziz, and P. Spasojevic FPGA-Based Channel Coding Architectures for 5G Wireless Using High-Level Synthesis, *International Journal of Reconfigurable Computing*, Article No. 3689308, 2017, doi: 10.1109/LCOMM.2017.2778727.
20. J. Yuan and J. Sha 4.7-Gb/s LDPC Decoder on GPU *IEEE Communications Letters*, vol. 22, no. 3, pp. 478–481, Mar. 2018, doi: 10.1109/LCOMM.2017.2778727.
21. J. Andrade, G. Falcão, and V. Silva Flexible design of wide-pipeline-based WiMAX QC-LDPC decoder architectures on FPGAs using high-level synthesis *IET Electronics Letters*, vol. 50, pp. 839–840, May 2014, doi: 10.1049/el.2013.3411.
22. S. Mhaske, H. Kee, T. Ly, A. Aziz, and P. Spasojevic FPGA-Based Channel Coding Architectures for 5G Wireless Using High-Level Synthesis *International Journal of Reconfigurable Computing*, Article ID 3689308, 2017. doi: 10.1155/2017/3689308.
23. C.-L. Wey, M.-D. Shieh, and S.-Y. Lin Algorithms of finding the first two minimum values and their hardware implementation *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3430–3437, Nov. 2008, doi: 10.1109/TCSI.2008.924892.
24. Q. Wang, Q. Liu, S. Wang, L. Chen, H. Fang, L. Chen, Y. Guo, and Z. Wu Normalized min-sum neural network for LDPC decoding *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 1, pp. 70–81, Mar. 2023, doi: 10.1109/TCCN.2022.3212438.
25. D. Oh and K. K. Parhi Min-sum decoder architectures with reduced word length for LDPC codes *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 105–115, Jan. 2010, doi: 10.1109/TCSI.2009.2016171.
26. M. P. C. Fossorier, M. Mihaljević, and H. Imai Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999, doi: 10.1109/26.768759.
27. R. M. Tanner A recursive approach to low complexity codes *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981, doi: 10.1109/TIT.1981.1056404.
28. Y. Cai, S. Jeon, K. Mai, and B. V. K. Vijaya Kumar Highly parallel FPGA emulation for LDPC error floor characterization in perpendicular magnetic recording channel *IEEE Trans. Magn.*, vol. 45, no. 10, pp. 3761–3764, Oct. 2009, doi: 10.1109/TMAG.2009.2022318.
29. M. P. C. Fossorier Quasicyclic low-density parity-check codes from circulant permutation matrices *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004, doi: 10.1109/TIT.2004.831841.
30. K. Chung, K. Cho, and W.-H. Lee Simplified 2-dimensional scaled min-sum algorithm for LDPC decoder *J. Electr. Eng. Technol.*, vol. 12, no. 3, pp. 1262–1270, May 2017, doi: 10.5370/JEET.2017.12.3.1262.
31. Z. Zhong, S. Guo, X. Xu, and H. Bai A classified normalized BP-based algorithm with 2-dimensional correction for LDPC codes *J. Commun.*, vol. 8, no. 5, pp. 315–321, May 2013, doi: 10.12720/jcm.8.5.315-321.
32. A. Hamad Estimation of two-dimensional correction factors for min-sum decoding of regular LDPC code *Wireless Eng. Technol.*, vol. 4, no. 4, pp. 181–187, Oct. 2013, doi: 10.4236/wet.2013.44027.
33. M. K. Roberts and R. Jayabalan, An improved low-complexity sum-product decoding algorithm for low-density parity-check codes *Frontiers of Information Technology & Electronic Engineering*, vol. 16, no. 6, pp. 511–518, Jun. 2015.
34. B. Le Gal and C. Jego Low-latency software LDPC decoders for x86 multi-core devices In *Proc. the 2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, 2017. doi: 10.1109/SiPS.2017.8110001.
35. N. Wiberg, H.-A. Loeliger, and R. Kotter Codes and iterative decoding on general graphs In *Proc. 1995 IEEE International Symposium on Information Theory (ISIT)*, Whistler, BC, Canada: IEEE, 1995, pp. 468, doi: 10.1109/ISIT.1995.550455.
36. O. Boncalo and A. Amaricai Ultra high throughput unrolled layered architecture for QC-LDPC decoders In *Proc. the 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Bochum, Germany, 2017, pp. 225–230, doi: 10.1109/ISVLSI.2017.47.

37. N. Yang, S. Jing, A. Yu, X. Liang, Z. Zhang, X. You, and C. Zhang Reconfigurable decoder for LDPC and polar codes In *Proc. 2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, Florence, Italy, 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8351337
38. T. Bhatt, V. Sundaramurthy, V. Stolpman, and D. McCain, Pipelined block-serial decoder architecture for structured LDPC codes In *Proc. 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings (ICASSP)*, Toulouse, France, 2006, vol. 4, pp. IV–IV, doi: 10.1109/ICASSP.2006.1660946.
39. S. A. Tamkeen and A. A. Hamad FPGA implementation of scaled "Quasi-Cyclic LDPC" decoder using high-level synthesis In *First International Conference on Mathematical Modeling and Computational Science: ICMMCS 2020*, Pattaya, Thailand, 2021, pp. 131–142. 10.1007/978-981-33-4389-4_14.
40. X. Wen, X. Jiao, P. Jääskeläinen, H. Kultala, C. Chen, H. Berg, and Z. Bie A high throughput LDPC decoder using a mid-range GPU In *Proc. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy: IEEE, 2014, pp. 7515–7519. doi: 10.1109/ICASSP.2014.6855061.
41. E. Scheiber, G. H. Bruck, and P. Jung Implementation of an LDPC decoder for IEEE 802.11n using Vivado™ High-Level Synthesis In *Proc. 2013 IEEE International Conference on Electronics, Signal Processing and Communication Systems*, no. 4, 2013.
42. J. Andrade, F. Pratas, G. Falcao, V. Silva, and L. Sousa Combining flexibility with low power: Dataflow and wide-pipeline LDPC decoding engines in the Gbit/s era In *Proc. 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, 2014, pp. 264–269. doi: 10.1109/ASAP.2014.6868671.
43. S. Mhaske, H. Kee, T. Ly, A. Aziz, and P. Spasojevic High-Throughput FPGA-Based QC-LDPC Decoder Architecture In *Proc. 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, 2015, pp. 1–5. doi: 10.1109/VTCFall.2015.7390967.
44. S. A. Zied, A. T. Sayed, and R. Guindi Configurable low complexity decoder architecture for Quasi-Cyclic LDPC codes In *Proc. 2013 21st International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2013, pp. 1–5. doi: 10.1109/SoftCOM.2013.6671861.
45. I. Tanyanon and S. Choomchuay A hardware design of MS/MMS-based LDPC decoder In *Proc. 2012 IEEE International Conference on Electron Devices and Solid State Circuits (EDSSC)*, 2012, pp. 1–4. doi: 10.1109/EDSSC.2012.6482804.
46. B. Le Gal and C. Jego Design of an ASIP LDPC Decoder Compliant with Digital Communication Standards In *Proc. 2012 IEEE Workshop on Signal Processing Systems (SiPS)*, 2012, pp. 19–24. doi: 10.1109/SiPS.2012.62.
47. W. H. Zhao and J. P. Long Implementing the NASA Deep Space LDPC Codes for Defense Applications In *Proc. 2013 IEEE Military Communications Conference (MILCOM)*, 2013, pp. 803–808. doi: 10.1109/MILCOM.2013.142.
48. A. Amaricai, O. Boncalo, and I. Mot Memory efficient FPGA implementation for flooded LDPC decoder In *Proc. 2015 23rd Telecommunications Forum (TELFOR)*, Belgrade, Serbia, 2015, pp. 500–503. doi: 10.1109/TELFOR.2015.7377516.
49. V. Pignoly, B. Le Gal, C. Jego, and B. Gadat High data rate and flexible hardware QC-LDPC decoder for satellite optical communications In *Proc. 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, 2018, pp. 1–5. doi: 10.1109/ISTC.2018.8625274.
50. N. Khosroshahi and T. A. Gulliver Quasi-cyclic low density parity check (LDPC) codes for dedicated short range communication (DSRC) systems In *Proc. 2010 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Calgary, AB, Canada: IEEE, 2010, pp. 1–5, doi: 10.1109/CCECE.2010.5575233.
51. A. Çağlan, E. İnceöz, E. Balcısöy, M. Özbek, and E. Çavuş FPGA implementation of AWGN noise generator using Box–Muller method In *Proc. IEEE SIU*, May 2016, pp. 1813–1816, doi: 10.1109/SIU.2016.7496114.
52. M. A. Cross and T. Fleetwood Hubless VSAT networks In *IEE Colloquium on VSATs—Trends and Technologies*, 1989, pp. 5/1–5/13.
53. N. Khosroshahi, R. Mankarious and M. R. Soleymani CNN-Based LDPC Decoder for Hubless Full-Mesh VSATPlus® System In *Proc. IEEE/AIAA Digital Avionics Systems Conference (DASC)*, 2025, to be published.
54. *IEEE Standard for Local and metropolitan area networks - Part 16: Air Interface for Broadband Wireless Access Systems*, IEEE Std 802.16TM, 2009.
55. *IEEE Standard for Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11n, 2009.
56. *IEEE Standard for Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11ad, 2007.

 doi:10.20944/preprints202601.1080.v1

57. *Verizon 5G Technical Forum; Air Interface Working Group; Verizon 5th Generation Radio Access; Multiplexing and channel coding (Release 1)*, TS V5G.212 V1.2, 2016.
58. *3rd Generation Partnership Project; Technical Specification; 5G; NR; Multiplexing and channel coding*, 3GPP TS 38.212 V15.2.0, 2018.
59. *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*, ETSI EN 302 307 V1.4.1, Apr. 2009.
60. *Digital Video Broadcasting (DVB); Frame structure, channel coding and modulation for a second generation digital transmission system for cable systems (DVB-C2)*, ETSI EN 302 769 V1.3.1, Jun. 2009.
61. *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television (DVB-T)*, ETSI EN 300 744 V1.6.1, Jul. 2009.
62. *Radio Broadcast systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers*, ETSI ES 300 401 V1.4.1, Sep. 1995.
63. *Digital cellular telecommunications system (Phase 2+); Radio transmission and reception (GSM)*, ETSI GSM 05.05, May 1996.
64. *Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer—General description*, ETSI TS 136 201 V15.2.0, Dec. 2017.
65. IEEE 802.16 Working Group LDPC coding for OFDMA PHY Contribution to IEEE 802.16e, Aug. 2004. [Online]. Available: http://ieee802.org/16/tge/contrib/C80216e-04_373r1.pdf
66. PolarSat Inc. VSATPlus3® Overview 2003. [Online]. Available: https://www.polarsat.com/vsatplus-3
67. AMD Xilinx (2021, May). *Vivado Design Suite User Guide: High-Level Synthesis UG902 V2020.1*. [Online]. Available: https://docs.amd.com/v/u/en-US/ug902-vivado-high-level-synthesis
68. N. Wiberg Codes and decoding on general graphs Ph.D. dissertation, Dept. Elect. Eng., Linköping Univ., Sweden, 1996.

Short Biography of Authors

Najmeh Khosroshahi (Member, IEEE) received the B.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2007, and the M.Sc. degree in electrical and computer engineering from the University of Victoria, Victoria, BC, Canada, in 2011. She is currently pursuing the Ph.D. degree in electrical and computer engineering at Concordia University, Montréal, QC, Canada. She is a Digital Communication Systems Engineer with PolarSat Inc., Montréal, QC, Canada, developing and verifying FPGA-centric signal-processing for VSAT platforms. Her current research interests include error-correcting codes, artificial intelligence (AI), quantum/learning-assisted decoding, and satellite communications. She is the author of *Inter-Vehicle Communication Systems Improvement* (LAP LAMBERT Academic Publishing, 2014).

 doi:10.20944/preprints202601.1080.v1

Ron Mankarious (Member, IEEE) received the B.Sc. degree in electrical engineering and the B.A. degree in economics from the University of California, Los Angeles (UCLA), Los Angeles, CA, USA, in 1985. He is currently the Executive Vice President of Sales and Marketing with PolarSat Inc., Montreal, QC, Canada, which he co-founded in 2003. Previously, he held management and engineering positions with NSI Communications, ComStream Corporation, Interstate Electronics, and Hughes Aircraft Company, all in California, USA. He has more than 40 years of experience in wireless and satellite communications and has authored IEEE papers on wireless adaptive routing and error-correction coding, as well as articles on satellite communications in leading industry publications. His current professional interests include satellite networking, adaptive routing, and forward error correction for MF–TDMA systems.

M. Reza Soleymani (Senior Member, IEEE) received the B.S. degree in electrical engineering from the University of Tehran, Tehran, Iran, in 1976, the M.S. degree in electrical engineering from San Jose State University, San Jose, CA, USA, in 1977, and the Ph.D. degree in electrical engineering from Concordia University, Montréal, QC, Canada, in 1987. From 1987 to 1990, he was an Assistant Professor in the Department of Electrical Engineering at McGill University, Montréal, QC, Canada. From 1990 to 1998, he was with EMS Technologies Ltd. (formerly Spar Aerospace Ltd.), where he had a leading role in the design and development of several satellite communications systems. In 1998, he joined the Department of Electrical and Computer Engineering at Concordia University, Montréal, QC, Canada, where he is presently a Professor. His current research interests include digital communications, satellite communications, communications networks, information theory and coding, and data compression and source coding. He holds several patents and has coauthored a book, *Turbo Coding for Satellite and Wireless Communications* (Kluwer Academic Publishers, 2002), as well as a number of book chapters in the field.