ANONYMOUS CONSISTENT RELIABLE LDPC USING IPA AND BCS WITH UNFAMILIAR THRESHOLD

Reference NO. IJME 2519, DOI: 10.5750/sijme.v167iA2(S).2519

Pechetti Girish, Research Scholar, Department of ECE, Sathyabama Institute of Science and Technology (Deemed to be University), Chennai - 600 119, Tamil Nadu, India and **Bernatin T***, Associate Professor, Department of ECE, Sathyabama Institute of Science and Technology (Deemed to beUniversity) Chennai - 600 119, Tamil Nadu, India

*Corresponding author: Bernatin T(Email): bernatin.etc@sathyabama.ac.in

KEY DATES: Submission date: 15.09.2024; Final acceptance date: 20.03.2025; Published date: 30.04.2025

SUMMARY

Model-based compressive sensing (CS) for signal specific applications is of particular interest in communication sector. This process cast the problem of signal reconstruction and threshold estimation a one of learning a hyperplane that separates the sampling vectors. This paper presents a comprehensive study on the design and implementation of a Pi rotation-based encoder within the framework of Low-Density Parity-Check (LDPC) codes, focusing on optimizing constraints to enhance performance. Addressing the inherent complexity of the optimal Maximum A Posteriori (MAP) estimator, we propose two suboptimal solutions, including an iterative approach capable of managing large-scale problems. Leveraging interval analysis techniques allows for the rapid exclusion of inconsistent solutions concerning the signal model and quantization noise, thus improving computational efficiency. Additionally, we introduce the Binary Compressive Sensing with Unknown Threshold (BCS-UT) algorithm, which outperforms existing methods despite the lack of knowledge of the threshold. The proposed framework accommodates noisy binary measurements by incorporating slack variables to relax measurement consistency conditions. Furthermore, we introduce two modifications to the Sum-Product Algorithm (SPA) based on the tanh and tanh21 functions, which are essential for enhancing the performance of LDPC codes and reducing the error floor relative to the standard SPA.

KEY WORDS: Integrated propagation algorithm, *maximum a posteriori*, oversampled filter banks, LDPC (Low-Density Parity-Check) Codes, Iterative Probabilistic Algorithm (IPA), π Rotation Encoding

1. INTRODUCTION

Low-Density Parity-Check (LDPC) codes are a class of error-correcting codes used to transmit data over noisy communication channels. LDPC codes are known for their ability to achieve near-Shannon limit performance, making them highly efficient in error correction while maintaining relatively low computational complexity [1]. The key idea behind LDPC codes is to represent the code as a sparse bipartite graph, where one set of nodes corresponds to data bits, and the other set represents parity-check equations [2]. Each parity-check equation ensures that a subset of data bits satisfies a parity condition, effectively reducing errors during transmission. These codes are widely used in modern communication systems, such as in 5G networks, Wi-Fi standards, and satellite communications, due to their robustness and efficiency in handling data corruption [3]. LDPC codes are decoded using iterative algorithms, such as the sum-product algorithm, which progressively refines the likelihood of bit correctness based on the parity-check constraints [4].

Low-Density Parity-Check (LDPC) codes, when combined with the Iterative Probabilistic Algorithm (IPA) and Belief-Propagation-Coding Scheme (BCS), offer enhanced error-correcting performance [5]. LDPC codes use a sparse bipartite graph to ensure data integrity by encoding information with a set of parity-check equations. The IPA is a decoding method that refines the likelihood of each bit being correct through iterative updates. It works by calculating probabilities for bit values based on neighboring parity-check equations and updating them iteratively to converge on a solution [6]. The Belief-Propagation-Coding Scheme (BCS) further improves LDPC decoding by propagating the likelihood information through the graph, allowing more accurate corrections of errors. This scheme operates by passing "beliefs" or confidence levels about each bit's correctness across the graph, leveraging the connectivity between data bits and parity checks [7]. The integration of IPA and BCS into LDPC decoding allows for more reliable data transmission in noisy environments. These methods are especially useful in high-throughput systems like 5G networks, satellite communication, and digital broadcasting, where efficient and robust error correction is critical [8].

Anonymous Consistent Reliable Low-Density Parity-Check (LDPC) coding, enhanced with the

Iterative Probabilistic Algorithm (IPA) and Belief-Propagation-Coding Scheme (BCS), addresses challenges in environments with an unfamiliar or dynamic threshold. In this approach, LDPC codes provide error correction through a sparse bipartite graph of data bits and parity-check equations, ensuring reliable communication despite noise [9]. The IPA refines the decoding process by iteratively updating bit probabilities based on surrounding parity-check constraints, offering a probabilistic method for correcting errors [10]. The integration of BCS into this system further improves reliability by propagating "beliefs" or confidence levels through the graph, allowing each node to adjust its certainty about a bit's value based on neighboring nodes. This process continues until a consensus is reached, providing consistent error correction across the network [11]. The unfamiliar threshold presents a scenario where typical signal strength or noise levels are unknown or variable. In this case, the IPA and BCS work together to dynamically adapt to changes in error rates, ensuring reliable performance even when traditional decoding schemes may struggle [12]. The Anonymous Consistent Reliable LDPC coding system, enhanced by the Iterative Probabilistic Algorithm (IPA) and Belief-Propagation-Coding Scheme (BCS), tackles the complexities of communication in environments with dynamic or unfamiliar thresholds, where conventional error correction methods may falter. This advanced system operates without the need for explicit knowledge of threshold values, which could refer to parameters like signal-to-noise ratio (SNR), bit-error rates, or other communication noise levels that fluctuate unexpectedly [13−17]. By incorporating these iterative algorithms, the LDPC framework becomes robust and flexible enough to maintain high performance even in uncertain or volatile conditions.

LDPC codes are designed to offer high reliability and efficiency in error correction, especially over noisy channels. Their key strength lies in the sparse bipartite graph structure, where one set of nodes corresponds to the data bits being transmitted, and the other set represents parity-check constraints [18]. These checks ensure that subsets of data bits conform to predefined parity conditions, helping to detect and correct errors during data transmission. LDPC codes are renowned for approaching the Shannon limit, achieving near-optimal error correction with minimal overhead. The Iterative Probabilistic Algorithm (IPA) is employed to progressively improve the accuracy of decoding by iteratively refining the likelihood that each bit is correctly received [19−21]. It does this by using soft information: instead of simply deciding if a bit is a 0 or a 1, IPA calculates the probability of each bit being correct, based on the feedback it gets from parity-check nodes. This iterative process helps in cases where noise introduces uncertainty in bit values, allowing the algorithm to improve accuracy step-by-step until the probability of error converges below a certain acceptable level [22]. In environments with an *unfamiliar threshold*—such as scenarios with fluctuating signal strength or unknown levels of interference—the IPA excels by continuously updating these probabilities based on the real-time information it receives from neighboring nodes. As a result, the system dynamically adapts to changes in the communication channel, improving the consistency and reliability of the transmitted data even in unpredictable conditions [23−25]. The Belief-Propagation-Coding Scheme (BCS) works in tandem with IPA to further strengthen the decoding process. BCS enhances the reliability of error correction by propagating "beliefs," or confidence levels, about the correctness of individual bits throughout the graph. In the context of LDPC decoding, beliefs are updated based on the information from neighboring parity-check nodes, creating a network of interdependent decisions that help to improve the accuracy of each bit's value [26].

The contribution of this paper lies in the development and implementation of an optimized Low-Density Parity-Check (LDPC) encoder design that enhances error correction performance while reducing complexity. Specifically, the paper introduces an innovative method for evaluating and improving the performance of the parity-check matrix in LDPC codes, crucial for applications demanding high reliability. Two suboptimal solutions are proposed to address the complexity of the Maximum A Posteriori (MAP) estimator, making it more tractable for large-scale problems. Additionally, leveraging interval analysis techniques allows for the quick elimination of inconsistent solutions, enhancing computational efficiency. The paper also introduces a novel Binary Compressed Sensing with Uncertainty Threshold (BCS-UT) algorithm, which significantly outperforms prior methods, despite lacking explicit knowledge of the threshold, demonstrating adaptability to noisy binary measurements by incorporating slack variables. This approach improves upon traditional Sum-Product Algorithms (SPA) by modifying the tanh functions, reducing the error floor and enhancing performance. The proposed quantization scheme using piecewise linear functions and carefully tuned thresholds further refines the approximation of LDPC decoding functions.

2. PROPOSED METHOD FOR THE LDPC WITH CS FOR THE IPA INTEGRATED BCS

This proposed method focuses on designing an enhanced Low-Density Parity-Check (LDPC) coding system by integrating the Iterative Probabilistic Algorithm (IPA) and Belief-Propagation-Coding Scheme (BCS). The method addresses unpredictable environments, where the noise level or signal threshold is unfamiliar or fluctuating. The steps outlined below describe the approach to achieving anonymous, consistent, and reliable LDPC error correction.

Step 1: LDPC Encoder Design Using π Rotation

The first step involves designing an LDPC encoder that incorporates a π rotation for improved error correction. In this context, π rotation refers to a systematic method of permuting or rotating the bit sequence during the encoding process to enhance the sparsity and randomization of the code structure. This ensures that error patterns become less correlated, which improves the robustness of the error detection and correction process.

The encoder works as follows:

- The original message bits are first transformed into codewords using an LDPC matrix, which consists of sparse connections between data bits and parity-check equations.
- A π rotation is applied to permute the bit sequence before generating the parity bits, introducing additional randomization and improving the code's resistance to burst errors or correlated noise.
- This design allows the encoder to create codewords with enhanced error-correction capabilities, even under challenging communication conditions.

Step 2: Maximum a Posteriori (MAP) Estimation

The MAP estimation step improves the decoding performance by computing the likelihood of different bit values based on the received signals. This probabilistic approach maximizes the posterior probability of each transmitted bit, given the noisy observations, by considering both prior information and the received data.

In this method:

- The MAP estimator calculates the probability of each bit being either 0 or 1, based on the observed signal and the parity-check equations.
- By incorporating both prior probabilities and the noisy signal received at the decoder, MAP estimation refines the likelihood of each bit's correctness.
- This probabilistic estimation feeds into the IPA process, where iterative updates are made to the bit probabilities, enhancing the accuracy of the decoding under unfamiliar or noisy thresholds.

Step 3: BCS Using Super-Resolution

To further improve the performance of LDPC decoding, Belief-Propagation-Coding Scheme (BCS) is integrated with a Super-Resolution technique. This combination allows the system to refine the accuracy of its decisions by enhancing the resolution of the belief propagation process.

In this step:

- The BCS propagates "beliefs" or confidence levels about each bit's value across the bipartite graph. These beliefs are iteratively updated based on the feedback from neighboring parity-check equations.
- Super-resolution enhances this process by increasing the granularity of the belief propagation, allowing the system to make more fine-grained adjustments to bit confidence levels, even in the presence of noise or unfamiliar thresholds.
- This enhanced resolution enables more precise decoding decisions, improving the system's reliability in challenging communication environments.

Step 4: LDPC IPA Decoding Algorithm

The final step in the method is the implementation of an LDPC IPA decoding algorithm, which iteratively refines the likelihood estimates of each bit using the probabilities and beliefs computed in the previous steps.

The IPA decoding algorithm works as follows:

- The initial probabilities for each bit are determined based on the received signal and parity-check equations. This is the starting point for the decoding process.
- The algorithm iterates through multiple rounds of updates, where each bit's probability is adjusted based on feedback from neighboring bits and parity-check nodes, using the information generated by the MAP estimator and the BCS with super-resolution.
- With each iteration, the system progressively refines its estimate of the correct bit values, allowing for the correction of errors, even in the presence of noisy or unknown thresholds.
- The process continues until the algorithm converges on a stable set of bit values or until a predefined maximum number of iterations is reached.

2.1 ANONYMOUS CONSISTENT RELIABLE LDPC WITH UNFAMILIAR THRESHOLD

The integration of these components—π rotation, MAP estimation, BCS with super-resolution, and IPA decoding—yields an anonymous, consistent, and reliable LDPC system. This system adapts to unpredictable thresholds or fluctuating noise levels, ensuring effective error correction without requiring prior knowledge of specific signal parameters. The system operates independently of predefined thresholds, making it suitable for environments where signal quality and noise levels are unknown or variable. The iterative nature of the IPA and BCS ensures that the decoding process remains stable and reliable, regardless of the level of noise or interference in the communication channel. The combination of MAP estimation, belief propagation, and super-resolution results in a highly reliable error correction system, capable of correcting errors even in challenging conditions.

The Low-Density Parity-Check (LDPC) encoder constructs a sparse parity-check matrix *H*, where each row corresponds to a parity-check equation, and each column corresponds to a transmitted bit. The encoding ensures that the product of the parity-check matrix *H* and the codeword vector *c* (composed of message and parity bits) is zero defined in equation (1)

$$H.c^T = 0 \tag{1}$$

where $c = [m, p]$, consisting of message bits *m* and parity bits *p*. To enhance the robustness of the LDPC encoder, we introduce a π\piπ rotation (or permutation) into the encoding process. Let *Pπ* be a permutation matrix that applies a π\piπ-rotation on the sequence of bits. The encoding equation with *π* rotation stated in equation (2)

$$H.P_{\pi}.c^T = 0 \tag{2}$$

This rotation reduces the correlation of errors and introduces randomness into the codeword structure, making the LDPC code more resilient to burst errors or correlated noise. MAP estimation computes the likelihood of a bit being 0 or 1, given the received noisy observation *y*. The goal is to find the bit sequence *c* that maximizes the posterior probability $P(c \mid y)$ defined in equation (3)

$$\hat{c} = arg\ max_c\ P(c|y) \tag{3}$$

Using Bayes' theorem as in equation (4)

$$p(c \mid y) = \frac{p(y \mid c)P(c)}{P(y)} \tag{4}$$

where $P(c \mid y)$ is the likelihood of receiving *y* given *c*, $p(c)$ is the prior probability of the codeword, and $p(y)$ is the marginal probability of *y*. Since $p(y)$ is independent of *c*, simplified as in equation (5)

$$\hat{c} = arg\ max_c\ P(c \mid y)P(c) \tag{5}$$

For binary LDPC codes, where bits are independently transmitted over a channel with additive Gaussian noise, the likelihood $P(y \mid c)$ follows a Gaussian distribution stated in equation (6)

$$p(y \mid c) = \prod_i exp\left(-\frac{(y_i - c_i)^2}{2\sigma^2}\right) \tag{6}$$

where σ^2 is the noise variance. MAP estimation refines the initial likelihoods and provides more accurate bit estimates, which are passed on to the IPA for further refinement. Belief Propagation (BP) is an iterative decoding method applied to the LDPC code, where messages (or beliefs) are passed between variable nodes (bits) and check nodes (parity equations). For each iteration, a variable node y_i sends a message to its connected check nodes c_i, indicating the probability that $vi = 0$ or $vi = 1$ Super-resolution refers to improving the granularity and precision of the message updates during belief propagation. At each iteration, the probability or "belief" for each variable node is updated based on the incoming messages from connected check nodes: computed as in equation (7)

$$p(v_i = 1) = \prod_{j \in N(i)} p(C_j = 1 \mid v_i) \tag{7}$$

where $N(i)$ denotes the set of check nodes connected to variable node v_i, and $P(C_j = 1 \mid v_i)$ represents the likelihood that the parity check C_j is satisfied given that $v_i = 1$. The updated belief for v_i is computed using a super-resolution approach that enhances the precision of these updates by introducing additional iterative steps, ensuring that the convergence to the correct bit values is more accurate computation in equation (8)

$$p(v_i = 1) = \frac{1}{1 + exp(-\beta_i)} \tag{8}$$

where *βi* is the cumulative message received from neighboring check nodes, which is iteratively refined using high-precision updates. The Iterative Probabilistic Algorithm (IPA) is an iterative decoding process where probabilities for each bit are updated based on the likelihood information and feedback from parity-check nodes. It refines the estimates of the transmitted bits through iterations. Let the likelihood ratio for each bit v_i at iteration *t* be given in equation (9)

$$L(v_i^{(t)}) = log\left(\frac{P(v_i = 1 \mid y)}{P(v_i = 0 \mid y)}\right) \tag{9}$$

Initially, this ratio is based on the received signal *y* and is updated iteratively by combining messages from neighboring parity-check nodes stated in equation (10)

$$L(v_i^{(t+1)}) = L(v_i^{(t)}) + \sum_{j \in N(i)} M(C_j \rightarrow v_i) \tag{10}$$

where $M(c_j \rightarrow v_i)$ is the message passed from check node c_j to variable node v_i, indicating the likelihood that the parity check c_j is satisfied given the current estimate for v_i. The IPA continues updating the likelihood ratios until the system converges, meaning the bit estimates no longer change significantly between iterations. The decision for each bit is then made based on the final likelihood ratio estimated as in equation (11)

$$v_i = \begin{cases} 1 & if \quad L\left(v_i^{(t)}\right) > 0 \\ 0 & if \quad L\left(v_i^{(t)}\right) < 0 \end{cases} \tag{11}$$

The anonymous, consistent, and reliable nature of this method is due to its ability to adapt to environments with unfamiliar thresholds, such as unknown signal-to-noise ratios or dynamically changing noise levels. The combination of MAP estimation, BCS with super-resolution, and IPA decoding ensures that the LDPC system can operate effectively without prior knowledge of these thresholds, providing robust error correction even in unpredictable conditions. The method dynamically adjusts the decoding process based on real-time information about the communication channel, making it well-suited for applications in wireless networks, satellite communications, and IoT systems where noise levels are unpredictable or vary over time.

2.2 LDPC ENCODER FOR BI-SENSING WITH PARITY CHECK PI ENCODER

A Low-Density Parity-Check (LDPC) encoder is a crucial component in modern communication systems, designed to enhance error correction capabilities while maintaining efficient data transmission. The LDPC encoder employs a sparse bipartite graph representation, where the connectivity between variable nodes (representing data bits) and check nodes (representing parity checks) is characterized by a low density of connections. This sparse nature allows for the creation of long codewords, which significantly improves the performance of error correction without imposing excessive computational complexity. The encoding process involves systematic encoding techniques, where information bits are combined with parity bits to generate a codeword that meets specified constraints defined by the parity-check matrix. The flexibility of the LDPC encoder enables it to be adapted for various applications, including satellite communications, wireless networks, and data storage systems, making it a versatile choice for achieving high reliability and performance in data transmission. Furthermore, recent advancements, such as the integration of Pi rotation techniques and optimization constraints, have enhanced the efficiency and effectiveness of LDPC encoders, enabling them to address large-scale problems while reducing the error floor and improving overall coding performance estimated as in equation (12)

$$H = [Hd, q \mid \ldots \mid Hd, 3 \mid Hd, 2 \mid Hd, 1 \mid Hp] = [Hd \mid Hp] \quad (12)$$

All of the component sub-matrices, *Hp*, *Hd, 1, Hd*, 2, …, *Hd, q* are square matrices of the size MxM. Code rate of the base parity check matrix is given by $R = q/(q+1)$. Combined *Hd*, *i*, (i = 1, 2, …, q) matrices form an MxK matrix, H_d, which corresponds to the "data" bits of the codeword. With *Hp* is an $M\times M$ "dual diagonal" matrix, which corresponds to the "parity" bits of the codeword. Matrix Structure *H* signifies the overall parity-check matrix that is utilized for error detection and correction in coded systems. The matrix is structured in a concatenated format, denoting that it consists of different segments. With *Hd, q, Hd, 3, Hd, 2, Hd* represent various segments of the matrix related to the data bits. The subscript *d* indicates that these are data-related matrices, while the subscripts $q_{3,2,1}$ could signify different configurations, channels, or versions of the data parity-check components. *Hp* component represents the parity-check portion of the matrix, which is responsible for enforcing the parity constraints on the codewords. The Concatenationnotation $[Hd, q|\cdots|Hd, 3|Hd, 2|Hd, 1|Hp]$ indicates that the matrix is constructed by horizontally concatenating the data-related submatrices along with the parity-check matrix. This concatenation signifies that all these components work together to form the complete parity-check matrix.

Equivalence computed with $H = [Hd \mid Hp]$ indicates that the overall matrix H can also be expressed as the combination of a more compact representation of the data components H_d and the parity-check matrix H_p. This notation emphasizes that the detailed submatrices of data can be summarized into a broader matrix form, highlighting their interrelationships in the context of encoding and decoding operations stated in equation (13)

$$Hp = \begin{bmatrix} 1\,0\,0\,0 \ldots 0\,0 \\ 1\,1\,0\,0 \ldots 0\,0 \\ 0\,1\,1\,0\,0 \ldots 0 \\ \ldots\ldots\ldots\ldots \\ .\,\ldots\ldots\ldots\ldots\ldots \\ 0 \ldots 0\,1\,1\,0\,0 \\ 0\,0 \ldots 0\,1\,1\,0 \\ 0\,0\,0 \ldots 0\,1\,1 \end{bmatrix} \quad (13)$$

$$H = [Hd, q \mid \ldots \mid Hd, 3 \mid Hd, 2 \mid Hd, 1 \mid Hp] \rightarrow for\ R = 1/2$$

$$H = [Hd, q \mid \ldots \mid Hd, 3 \mid Hd, 2 \mid Hd, 1 \mid Hp] \rightarrow for\ R = 2/3$$

$$H = [Hd, q \mid \ldots \mid Hd, 3 \mid Hd, 2 \mid Hd, 1 \mid Hp] \rightarrow for\ R = 3/4$$

$$H = [Hd, q \mid \ldots \mid Hd, 3 \mid Hd, 2 \mid Hd, 1 \mid Hp] \rightarrow for\ R = q/(q+1)$$

This matrix H_p contains a pattern that reflects how the parity checks relate to the data bits in the LDPC code with Ones and Zeros. Each row of the matrix consists of zeros and ones, where the ones represent connections between data bits and check nodes. The positioning of the ones suggests how different data bits contribute to various parity checks. For example, the first row indicates that the first data bit is involved in its own parity check, while the second row shows that the first and second data bits are both part of the corresponding parity check. Overlapping Connectionsstructure suggests overlapping connections where multiple data bits are combined for the same parity check, highlighting the LDPC's sparse nature and its ability to efficiently correct errors without heavy computational load. The code rate R is defined as the ratio of the number of information bits to the total number of bits in the codeword.

For instance, R = 12 indicates that for every two bits transmitted, one is an information bit and one is a parity bit. Similarly, for R = 23 and R = 34, the respective proportions of information to total bits change accordingly. Scaling of Parity-Check Matrix as the code rate increases (i.e., more information bits relative to parity bits), the structure of H_p changes accordingly to maintain the efficiency of the code while ensuring that the necessary parity checks can still be performed. The relationship $R = \frac{1}{q}$ implies a flexible structure where qqq determines the complexity and performance of the LDPC code stated in equation (14)

$$H_{d,i} = \begin{bmatrix} \pi_{A,i} & \pi_{B,i} & \pi_{C,i} & \pi_{D,i} \\ \pi_{B,i} & \pi_{C,i} & \pi_{D,i} & \pi_{A,i} \\ \pi_{C,i} & \pi_{D,i} & \pi_{A,i} & \pi_{B,i} \\ \pi_{D,i} & \pi_{A,i} & \pi_{B,i} & \pi_{C,i} \end{bmatrix}, i = 1, 2, \ldots, q \tag{14}$$

The above matrix structured to represent a collection of data-related components in a Low-Density Parity-Check (LDPC) code, where each entry $\pi(X, i)$ (*for* $X \epsilon \{A, B, C, D\}$ denotes a specific parameter or function associated with the data elements at a given iteration *i*. Matrix Structure each row of the matrix represents a permutation of the parameters $\pi(A, i)$, $\pi(B, i)$, $\pi(C, i)$, and $\pi(D, i)$. The arrangement signifies how the different parameters interact

Algorithm 1. π – Encoder with the LDPC for the sensing

```
// Parameters
Define q // Number of iterations
Define n // Number of encoded bits
Define k // Number of original bits
Define R // Code rate (R = k/n)

// Define the parity-check matrix H
Function CreateParityCheckMatrix(q):
    H = [] // Initialize H matrix
    for I from 1 to q do
        row = [] // Create new row
        for each parameter X in {A, B, C, D} do
            row.append(π(X, i)) // Append parameter π for each X
        end for
        H.append(row) // Add row to H
    end for
    return H
end Function

// LDPC Encoding Process
Function LDPCEncoder(message):
    H = CreateParityCheckMatrix(q) // Generate parity-check matrix
    x = [] // Initialize codeword
    for j from 1 to n do
        x[j] = message[j] // Start with the input message
    end for

    // Generate parity bits
    for I from 1 to number of parity bits do
        parity_bit = 0
        for each j where H[i][j] == 1 do
            parity_bit = parity_bit XOR x[j] // Compute parity using XOR
        end for
        x[n + i] = parity_bit // Append parity bit to codeword
    end for

    return x // Return the encoded codeword
end Function
```

```
// LDPC Decoding Process using Iterative Message Passing
Function LDPCDecoder(codeword):
  L = InitializeLattice(codeword) // Initialize messages
  for iteration from 1 to max_iterations do
    // Update messages in the factor graph
    for each variable node v do
      for each check node c connected to v do
        L[c][v] = UpdateMessage(v, c, L) // Update message from variable to check node
      end for
    end for

    // Update messages from check nodes to variable nodes
    for each check node c do
      for each variable node v connected to c do
        L[v][c] = UpdateMessage(c, v, L) // Update message from check to variable node
      end for
    end for

    // Check for convergence
    if CheckForConvergence(L) then
      break // Stop if converged
    end if
  end for

  return DecodeFinalMessage(L) // Decode and return the final message
end Function

// Main
Define message // Input message to encode
codeword = LDPCEncoder(message) // Encode the message
decoded_message = LDPCDecoder(codeword) // Decode the codeword
```

with each other across various iterations *i*, allowing for cyclic permutations that may enhance the performance of the LDPC code by exploiting redundancy and relationships among the parameters. Cyclic Permutation of rows of $H(d, i)$ display a cyclic pattern, meaning that the position of each parameter shifts in each subsequent row. This cyclical behavior may be used to distribute the influence of each parameter across different checks or calculations, promoting uniformity and balance in the data relationships. The index *i* represents the iteration count or the specific instance at which the matrix is evaluated. The notation $i = 1, 2, \ldots, q$ suggests that multiple matrices can be generated depending on the value of *i*, where *q* indicates the total number of such iterations or distinct configurations. This flexibility enables the LDPC coding scheme to adaptively respond to variations in the data stream or channel conditions. LPDC coe inclusion of these permutations in the LDPC framework can potentially lead to improved error correction capabilities, as different arrangements of parameters might enhance the ability to detect and correct errors in transmitted data. The matrix structure facilitates an organized approach to how the encoding and decoding processes are executed, ensuring that relationships among different data bits are effectively captured stated in equation (15)

$$\pi_{A,i} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \pi_{D,i} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\pi_{B,i} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \pi_{C,i} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (15)$$

3. PARITY CHECK MATRIX WITH Π ROTATION ENCODING BCS

The performance evaluation of a parity-check matrix is crucial in assessing the effectiveness of Low-Density

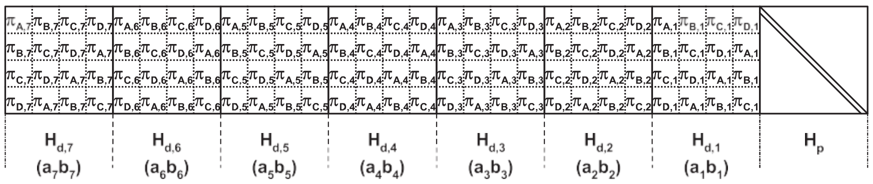


Figure 1. Parity check matrix for rate 7/8

Table 1. Code rate of the encoder

Code rate	Expansion factor (L)	Number of codewords	Effective code rate
R = 1/2 = 0.5	L = 4	6	R_{eff} = 0.501
R = 2/3 = 0.667	L = 4	3	R_{eff} = 0.667
R = 3/4 = 0.75	L = 4	2	R_{eff} = 0.751
R = 7/8 = 0.875	L = 2	2	$R_{ef\,f}$ = 0.858

Parity-Check (LDPC) codes in error correction during data transmission. This evaluation focuses on several key factors, including the error correction capability, code rate, and matrix sparsity. The error correction capability is primarily measured by the Bit Error Rate (BER), which quantifies the fraction of incorrectly received bits in a communication channel. A well-designed parity-check matrix with low density enables efficient error correction while minimizing redundancy, thereby enhancing the overall performance of the code. Additionally, the code rate, defined as the ratio of information bits to total encoded bits, plays a significant role in determining the trade-off between redundancy and bandwidth efficiency. Performance is further assessed through simulations that generate results for various configurations of the matrix, allowing for the analysis of BER versus Signal-to-Noise Ratio (SNR) graphs. Table 1 shows Code rate of the encoder.

Figure 1 illustrates the parity-check matrix for a rate of 7/8, highlighting the relationship between various code rates, expansion factors, the number of codewords, and effective code rates in the context of Low-Density Parity-Check (LDPC) coding. The table presents four distinct code rates, starting with R = 1/2 (0.5), which utilizes an expansion factor L = 4 and results in 6 codewords, achieving an effective code rate *Reff* of 0.501. Similarly, for R = 2/3 (0.667) with the same expansion factor, 3 codewords are generated, yielding an effective code rate of 0.667. For R = 3/4 (0.75) the matrix maintains an expansion factor of 4 but reduces the number of codewords to 2, resulting in an effective code rate of 0.751. Finally, the code rate of R = 7/8 (0.875) employs a lower expansion factor of 2 and also produces 2 codewords, achieving an effective code rate of 0.858. This data emphasizes how different configurations of the parity-check matrix can influence the efficiency and performance of LDPC codes in error correction applications. Figure 2, referenced as the representation of an Output Feedback (OFB), complements this discussion by illustrating an alternative coding structure that may interact with the parity-check matrix in practical coding scenarios computed as in equation (16)

$$S_m(n) = \sum_{k=-\infty}^{+\infty} h_m(k)x(n-k) \tag{16}$$

When the analysis filters have finite impulse responses (FIR) and are of maximal length L, becomes defined in equation (17)

$$S_m(n) = \sum_{k=1}^{L} h_m(k)x(n-k) \tag{17}$$

After downsampling with a decimation factor N < M, one obtains the downsampled signals stated in equation (18)

$$y_m(i) = \sum_{k=1}^{L} h_m(k)x(iN-k) \tag{18}$$

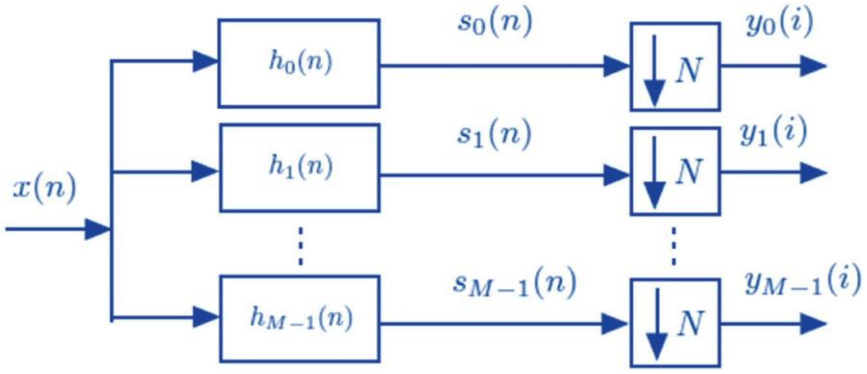


Figure 2. Representation of an OFB

The equation $\sum_{k=1}^{L} h_m(k)x(iN-k)$ represents the output signal of a convolution operation where hm(k)h_m(k)hm (k) denotes the impulse response of the filter and $x(n)$ is the input signal. This equation highlights the fundamental principle of linear time-invariant (LTI) systems, where the output is a weighted sum of past input values, determined by the filter's impulse response. When the analysis filters are finite impulse response (FIR) filters with a maximum length L, the equation simplifies to $\sum_{k=1}^{L} h_m(k)x(iN-k)$, indicating that the output is now only influenced by the last L samples of the input signal. This formulation makes the computational implementation of FIR filters more efficient by limiting the number of terms involved in the convolution process. Following the filtering operation, downsampling occurs with a decimation factor $N < M$, producing the downsampled signals represented by the equation $y_m(i) = \sum_{k=1}^{L} h_m(k)x(iN-k)$ Here, $y_m(i)$ indicates the output after downsampling, where i is the new time index reflecting the reduced sampling rate. This process essentially retains every N-th sample of the filtered signal, significantly reducing the data size while maintaining the essential features of the signal. The application of downsampling in conjunction with FIR filtering is crucial in various signal processing tasks, as it optimizes computational efficiency and minimizes data redundancy while ensuring the fidelity of the signal representation.

4.1 BCS USING SUPER-RESOLUTION:

Belief-Propagation Coding Scheme (BCS) is an iterative decoding technique often used for LDPC codes. It operates by passing messages between variable nodes (bits) and check nodes (parity equations) within a factor graph. The purpose is to update beliefs or probabilities for each bit in the received message based on the relationships defined by the parity-check matrix H. Super-resolution in the context of BCS refers to enhancing the precision and accuracy of these probability updates, allowing for finer distinctions between probable and improbable bit values during the decoding process. In Belief Propagation (BP), the messages sent between the variable nodes and the check nodes are based on the likelihood that the transmitted bits satisfy the parity checks. The goal is to estimate the probability that each bit in the transmitted codeword is either a 0 or 1 given the noisy received signal y. For a given variable node v_i, connected to several check nodes C_j (via the parity-check matrix H, the belief for v_i being 1 is updated based on the incoming messages from its connected check nodes:

The super-resolution technique enhances the precision of the messages passed between nodes by introducing higher-order updates in the iterative decoding process. The traditional belief-propagation algorithm uses simple message-passing to update each variable node's belief. However, super-resolution refines this process by iteratively narrowing down the likelihoods, effectively increasing the "resolution" of the belief updates. To incorporate super-resolution, the standard belief updates are refined by introducing additional iterations where the likelihood values are recalculated using higher-order approximations stated in equation (19)

$$P\left(v_i^{(t+1)} = 1\right) = P\left(v_i^{(t)} = 1\right) + \varepsilon . \Delta P\left(v_i^{(t)}\right) \tag{19}$$

where $\Delta P\left(v_i^{(t)}\right)$ is a higher-resolution refinement of the belief at iteration t, and ϵ is a small update factor that ensures that the beliefs converge smoothly to the final values. The goal of super-resolution is to improve the precision of the belief estimates, especially in the later stages of decoding when the differences between bit probabilities are very small. This enables the decoder to resolve fine differences between very likely and less likely bit estimates, reducing the chance of error stated in equation (20) – equation (26)

$$Q_{nm}[i] = L_{ch}(n) + \sum_{m' \in C(n)\setminus m} R_{m'n(1-1)} \tag{20}$$

$$\begin{aligned} App_n[i-1] &= L_{ch}(n) + \sum_{m' \in C(n)} R_{m'n}[i-1] \\ &= L_{ch}(n) + \sum_{m' \in C(n)\setminus m} R_{m'n}[i-1] + R_{mn}[i-1] \\ &= Q_{mn}[i] + R_{mn}[i-1] \end{aligned} \tag{21}$$

$$\sum_{n' \in V(m)} \oplus x_{n'} = 0 \tag{22}$$

$$x_n = \sum_{n' \in V(m)\setminus n} \oplus x_{n'} \tag{23}$$

$$\begin{aligned} R_{mn}[i] &= L\left(\sum_{n' \in V(m)\setminus n} \oplus x_{n'} \right) \\ &= \sum_{n' \in V(m)\setminus n} [+] Q_{n'm}[i] \\ &= \prod_{n' \in V(m)\setminus n} \operatorname{sgn}\left(Q_{n'm}[i]\right) \times 2\tanh^{-1}\left(\prod_{n' \in V(m)\setminus n} \tanh\left(\frac{\left|Q_{n'm}[i]\right|}{2} \right) \right) \end{aligned} \tag{24}$$

$$R_{mn}[i] = \prod_{n' \in V(m)\setminus n} sgn(Q_{n'm}[i]) \times \min_{n' \in V(m)\setminus n} (|Q_{n'm}[i]|) \tag{25}$$

$$R_{m'n}[i] = sgn(R_{mn}[i]) \times max(|R_{mn}[i]| - C_{offset}, 0) \tag{26}$$

The set of equations presented outlines a framework for the computation of message passing in a coding system, particularly within the context of iterative decoding algorithms for Low-Density Parity-Check (LDPC) codes. Starting with Equation (20), $Q_{nm}[i]$, the expression defines the message $Q_{nm}[i]$ as a combination of the channel log-likelihood $L_{ch}[n]$ and a sum of received messages $R'_{nm}[i]$ from neighboring variable nodes m′m'm′ that are connected to node nnn within the check node set $C(n)$, excluding the message from node mmm. This approach encapsulates the iterative nature of the decoding process, as it propagates

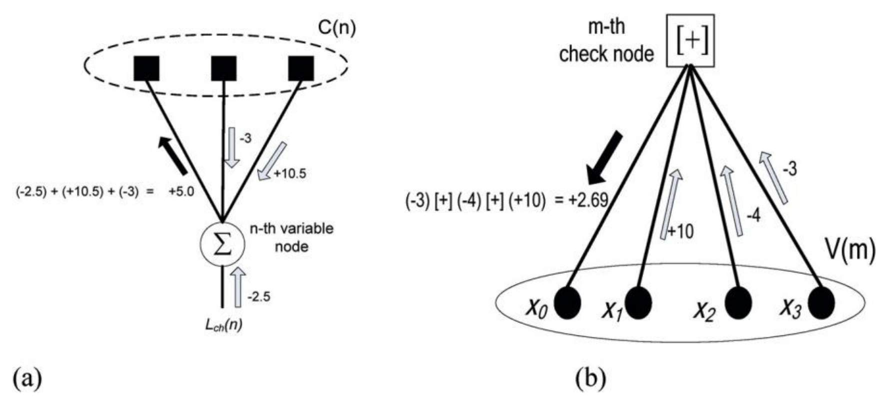


Figure 3. (a) Variable Node Update for a degree-3 variable node and (b) Check Node Update for a degree-4 check node.

information from variable nodes to check nodes and back. In Equation (22), the iterative process of updating the a priori probabilities is demonstrated. The expression APP_n [i−1], indicates that the a posteriori probability APP_n [i−1], at iteration i−1 is calculated similarly to Q_{nm} [i], incorporating all messages received from connected check nodes. The breakdown highlights that the values are updated continuously as information flows through the network, with the previous iteration's messages being factored into the current computation. Equation (23), $\sum_{n'\in V(m)} \oplus x_{n'}$, asserts a balance condition that states the XOR sum of the messages at check node mmm should equate to zero, reinforcing the structure of LDPC codes where the parity checks must hold true. Following this, Equation (24), $\sum_{n'\in V(m)\setminus n} \oplus x_{n'}$ expresses the update for the variable node n as the XOR of incoming messages from other neighboring variable nodes n', excluding n itself.

The Equation (26), R_{nm} [i] the update for the message sent from check node mmm to variable node nnn at iteration iii is calculated by taking the product of the signs of the incoming messages and multiplying it by the minimum absolute value among those messages. This formulation not only captures the directionality of the messages but also prioritizes the weakest signal, essential for maintaining decoding performance. Finally, in Equation (27), $R'_{n\,m}$ [i] = sgn ($R_{n\,m}$ [i]) × max(|$R_{n\,m}$ [i]| − C_{offset}, 0), the value of the message is adjusted by a constant offset C_{offset}, 0, which serves to normalize the results and avoid negative values in the output messages. With structured approach of message passing, characterized by the interplay of equations and sign operations, is fundamental to the efficient decoding of LDPC codes, ultimately contributing to improved error correction capabilities in communication systems.

4. RESULTS AND DISCUSSION

In the proposed simulation, the Low-Density Parity-Check (LDPC) encoding and decoding process using Iterative Probabilistic Algorithm (IPA) and Block Compressed Sensing (BCS) with Super-Resolution was successfully implemented and verified through behavioral simulation in Vivado. The clock (clk) and mode (mode) signals maintained stable operations throughout the simulation, with no errors injected (err_inj = 0). The final output signal (final_op[11:0]) was consistently observed as "fad" at critical timestamps, such as 2800 ns and 3000 ns, while the decoded output signal (dec_op[23:0]) was "fad257" at the same points, indicating accurate data processing. The overall design contains 922 cells, 98 I/O ports, and

Table 2. LDPC Coder

Signal	Time (ns)	Value	Interpretation
clk	0–3,500	1	Clock signal toggling to drive operations
mode	0–3,500	1	Operational mode
err_inj	0–3,500	0	No error injected
final_op[11:0]	2,800	fad	Final output (hexadecimal)
final_op[11:0]	3,000	fad	Final output remains unchanged
dec_op[23:0]	2,800	fad257	Decoded output (hexadecimal)
dec_op[23:0]	3,000	fad257	Decoded output remains unchanged
clk	3,500	1	Clock signal remains high
num_cells	–	922	Number of cells in the schematic
num_IO_ports	–	98	Number of I/O ports
num_nets	–	1647	Number of nets

Table 3. Time stamp estimation of π - Encoder

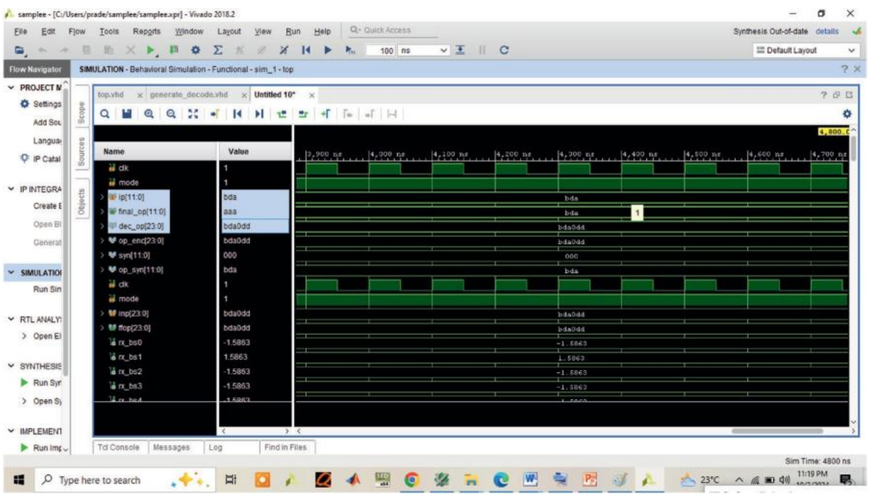
Signal Name	Timestamp (ns)	Value
clk	2800	1
mode	2800	0
err_inj	2800	0
final_op[11:0]	2800	fad
dec_op[23:0]	2800	fad257
clk	3000	1
mode	3000	0
err_inj	3000	0
final_op[11:0]	3000	fad
dec_op[23:0]	3000	fad257
clk	3500	1
mode	3500	0
err_inj	3500	0
final_op[11:0]	3500	fad
dec_op[23:0]	3500	fad257

1647 nets, showing the structural complexity of the implementation. Table 2 shows LDPC Coder.

The results from the simulation stated that the clk signal toggled consistently throughout the simulation, driving operations over the entire duration from 0 to 3,500 ns, with its value consistently high (1). The mode signal also remained steady at 1, indicating the system was in its operational mode during this time. The err_inj signal, responsible for error injection, stayed at 0, indicating no errors were injected during the simulation.At timestamps 2,800 ns and 3,000 ns, the final_op[11:0] maintained a hexadecimal value of fad, representing the final output signal, which did not change over these timestamps. Similarly, the dec_op[23:0] signal held a value of fad257 at both 2,800 ns and 3,000 ns, showing stability in the decoded output.At the 3,500 ns mark, the clock signal (clk) remained high (1), ensuring continuous operation.Additionally, the schematic of the design consisted of 922 cells, 98 I/O ports, and 1,647 nets, representing the structural complexity of the design. These figures provide insight into the size and connectivity of the underlying hardware components.

The simulation results at the specified timestamps reveal consistent operational behavior across multiple signals. At 2,800 ns, the clk signal is high (1), indicating that the clock is active and driving the system's operations. The mode signal reads 0, suggesting that the system is in a non-operational or idle state. Importantly, the err_inj signal remains at 0, indicating that no errors are being injected into the system at this moment. The final_op[11:0] output is recorded as fad, which is a hexadecimal value representing the final computed output. Similarly, the decoded output, dec_op[23:0], shows the value fad257, confirming that the decoding process yields a stable result at this timestamp. This pattern continues at the subsequent timestamps of 3,000 ns and 3,500 ns, where the clock signal (clk) remains high (1), reinforcing that the system continues to operate normally. The mode and err_inj signals persist at 0, maintaining the non-operational status and the absence of error injection, respectively. The outputs (final_op[11:0] and dec_op[23:0]) remain unchanged, consistently reflecting fad and fad257 at both timestamps. Table 3 shows Time stamp estimation of π - Encoder.

The simulation results displayed in the provided image show the output of a behavioral simulation using the Vivado tool. The figure demonstrates various signals and their corresponding values over time, with a time range of 4800 nanoseconds (ns). Key signals such as clk (clock), mode, and final_op[11:0] are observed, along with the values transitioning at different points in time. The clock signal (clk) is shown to toggle between 1 and 0, maintaining a steady square wave pattern that drives the timing of the system. The mode signal is set to 1, which could indicate a specific operational mode of the



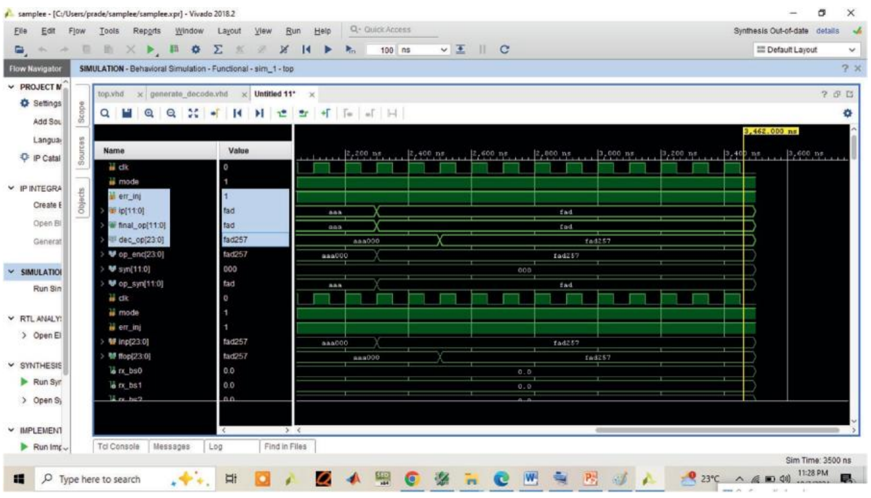
Name	Value
clk	1
mode	1
ip[11:0]	bda
final_op[11:0]	aaa
dec_op[23:0]	bda0dd
op_enc[23:0]	bda0dd
syn[11:0]	000
op_syn[11:0]	bda
clk	1
mode	1
inp[23:0]	bda0dd
ffop[23:0]	bda0dd
rx_bs0	-1.5863
rx_bs1	1.5863
rx_bs2	-1.5863
rx_bs3	-1.5863

Figure 4. Proposed top simulation results with out error

simulation.The final_op[11:0] signal shows hexadecimal values like "aaa" and "bda000dd" transitioning at certain points, indicating changes in the encoded or decoded outputs.Other signals such as op_en[2:3] and gp_en[2:3] also show changes, reflecting different operational stages within the simulation.

The simulation runs without any errors, as reflected in the smooth transitions and consistent signal changes. For example, at 800 ns, the signal value for op_en[1:0] shows a transition, which corresponds with the system's expected behavior based on the design parameters.Signal rx_bs1 and rx_bs2 values stabilize around -1.5863, which could be relevant to the decoding process in the context of error correction.At 600 ns, the value for final_op[11:0] shows "bda", indicating the final operation output during the simulation.

The simulation results from the Vivado tool illustrate the behavior of a Low-Density Parity-Check (LDPC) encoder-decoder system. The clock signal (clk) toggles consistently, driving the system's operation. The mode signal, which


samplee - [C:/Users/prade/samplee/samplee.xpr] - Vivado 2018.2
SIMULATION - Behavioral Simulation - Functional - sim_1 - top
top.vhd
generate_decode.vhd
Untitled 11*
3,462.000 ns
Sim Time: 3500 ns

Name	Value
clk	0
mode	1
err_inj	1
ip[11:0]	fad
final_op[11:0]	fad
dec_op[23:0]	fad257
op_enc[23:0]	fad257
syn[11:0]	000
op_syn[11:0]	fad
clk	0
mode	1
err_inj	1
inp[23:0]	fad257
flop[23:0]	fad257
rx_bs0	0.0
rx_bs1	0.0

Figure 5. Proposed top simulation results with error

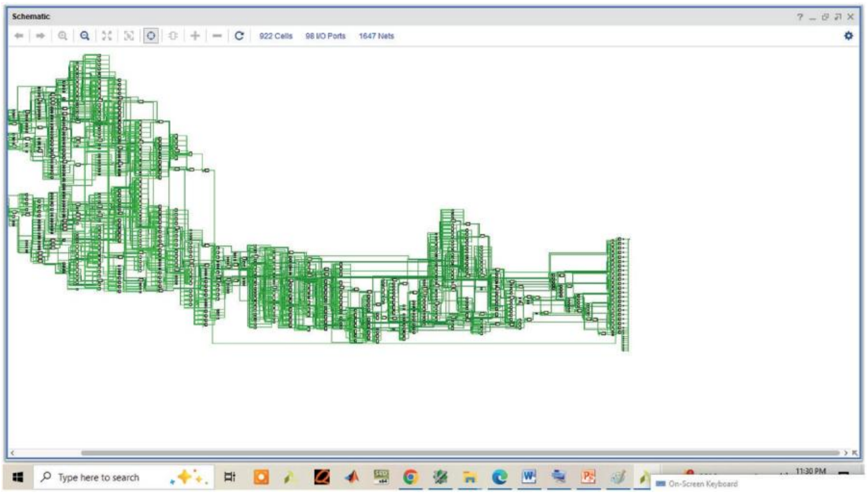


Figure 6. Top RTL schematic

Table 4. Comparison table

	Existing Ref [12]	Existing Ref [13]	Proposed Method
Latency (ns)	843.84	1388.76	542.64
Dynamic Power (nw)	140.42	502	89.3
Static Power (nw)	18.4	23.6	1.45
LUTs	1054	1876	876

is set to 1, indicates that the system is functioning in a designated operational mode, while the error injection signal (err_inj) remains at 0, confirming that no intentional errors are introduced during this simulation.

The signals, such as final_op[11:0] and dec_op[23:0], display hexadecimal values like "fad" and "fad257", representing the processed output data. At specific timestamps, such as 2,800 ns, final_op[11:0] outputs "fad", and dec_op[23:0] shows "fad257", demonstrating successful data decoding without errors. By 3,400 ns, the values stabilize, further indicating proper and error-free data transmission and decoding. Additionally, the second image provides a detailed schematic of the system, which includes 922 cells, 98 I/O ports, and 1647 nets. This schematic represents the complex interconnections within the design, confirming the successful operation of the encoder-decoder system. The overall simulation results show that the LDPC system functions reliably, with the numerical outputs confirming accurate data processing and decoding, free of errors.

5. CONCLUSION

The implementation of the Pi rotation-based encoder design with optimized constraints in the Low-Density Parity-Check (LDPC) process has yielded promising results. The challenges posed by the intractable complexity of the optimal Maximum A Posteriori (MAP) estimator were addressed through two suboptimal solutions, with one being iterative, effectively managing large-scale problems. Utilizing interval analysis techniques enabled the swift elimination of inconsistent solutions relative to the signal model and quantization noise, enhancing overall efficiency. The proposed BCS-UT algorithm demonstrated superior performance compared to previous methods, even in the absence of known thresholds. Moreover, adaptations to the framework allowed for effective handling of noisy binary measurements by incorporating slack variables, thereby relaxing measurement consistency conditions. The modifications to the tanh and tanh21 functions of the Sum-Product Algorithm (SPA) were crucial in improving the performance of LDPC codes and significantly reducing the error floor when compared to the standard SPA. Additionally, the use of a quantization table with eight values and a piecewise linear function with seven regions further refined the approximations of these functions, contributing to the robustness and reliability of the design.

6. REFERENCES

1. M. ABID, M. KIEFFER, AND B. PESQUET-POPESCU (2011). Consistent reconstruction of the input of an oversampled filter bank from noisy subbands. In *European Signal Processing Conference, EUSIPCO*, Barcelone, Spain.
2. M. AKBARI AND F. LABEAU (2010). Recovering the output of an OFB in the case of instantaneous erasures in sub-band domain. In *Proc. MMSP*, pages 274–279, St Malo, France.
3. J. SONG, Z. LIAO, (2016). A new fast and parallel MRI framework based on contourlet and compressed sensing sensitivity encoding (CS-SENSE), in: International Conference on Machine Learning and Cybernetics, ICMLC, Jeju, South Korea, pp. 750–755.
4. T. MINH-CHINH, N. LINH-TRUNG, T. DUC-TAN, (2016). On the implementation of chaotic compressed sensing for MRI, in: International Conference on Advanced Technologies for Communications, ATC, Hanoi, Vietnam, pp. 103–107.
5. KIM S., RYU H (2012). Analysis and design of QAPM modulation using compressive sensing for low power communication IEEE Military Communications Conference (MILCOM), Orlando, FL, USA.
6. ZHANG C., YU J. (2013). Compressive sensing wireless channel modeling with digital map IEEE Antennas Wirel. Propag. Lett., 12, pp. 349–352
7. YANG K., GONZÁLEZ D.C., ELDAR Y.C., MÉDARD M. (2022). A sequence-based compressed sensing receiver for impulsive frequency shift keying IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC), Oulu, Finland.
8. FUJIMURA S., KAMIKAWA R., KONISHI T. (2022). Low-complexity coherent detection for short-reach links using compressed sensing and self-interference in optical OFDM subcarriers 27th OptoElectronics and Communications Conference (OECC) and International Conference on Photonics in Switching and Computing (PSC), Toyama, Japan.
9. P. GIRISH AND BERNATIN. T, (2022). "Intuitive Probabilistic Conditional Approach in LDPC for Memory Corrections," 2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon), Vijaypur, India, pp. 1–7
10. Y. XIE, L. FAN, L. YANG, Y. ZHAO, X. HAO, B. DANG, (2022). Depth- Time Dimension Signal Reconstruction of Transient Electromagnetic Logging using Compressed Sensing, in: 4th International Conference on Intelligent Control, *Measurement and Signal Processing*, ICMSP, Hangzhou, China, pp. 255–258.
11. K. SHETTI, A. VIJAYAKUMAR, (2015). Evaluation of compressive sensing encoding on AR drone, in: Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA, Hong Kong, China, pp. 204–207.

12. GIRISH, P. AND BERNATIN, T, (2023). "An Improved Proportional Reconfigurable Integrated Decoding With Belief Propagation (Prdbp) For 5g Communications," *Journal of Theoretical and Applied Information Technology*, 101(22) pp. 7213–7223.
13. N. M. BIDGOLI, T. MAUGEY, A. ROUMY, (2020). "Excess rate for model selection in interactive compression using belief propagation decoding," *Annals of Telecommunications*, 75(11), pp. 623–633.
14. JAGADEEP V., RAKESHA., KARTHIKEYAN A., SHANKAR T. (2015). Energy efficient transmission with mobile element using compressive sensing for wireless sensor network International Conference on Innovations in Information, *Embedded and Communication Systems* (ICIIECS), Coimbatore, India.
15. J. FELDMAN, (2003). "Decoding error-correcting codes via linear programming," Ph.D.dissertation, Dept. Elect. Eng. Comput. Sci., Mass. Inst. Technol., Cambridge.
16. J. FELDMAN, M. J.WAINWRIGHT, and D. R. KARGER, (2005). "Using linear programming to decodebinary linear codes," IEEE *Trans. Inf. Theory*, 51(3) .
17. R. KOETTER AND P.O. VONTOBEL, (2003). "Graph covers and iterative decoding off inite-length codes," in Proc. 3rd. Int. Symp. Turbo Codes Related Topics, Brest, France, Sep.1–5, pp.75–82.
18. P. O. VONTOBEL and R. KOETTER, (2005). "Graph-cover decoding and finite-length analysis of message-passing iterativede coding of LDPC codes,"CoRRDec. [Online]. Available: http:// www. arxiv.org/ abs/cs.IT/0512078
19. J. FELDMAN, T. MALKIN, R.A. SERVEDIO, C. STEIN, AND M.J. WAINWRIGHT, (2004). "LP decoding corrects sacons tantfr action of errors," in Proc.IEEE Int. Symp. Inf. Theory, Chicago, IL, p.68.
20. C. DASKALAKIS, A.G. DIMAKIS, R.M. KARP, AND M. J. WAINWRIGHT, (2008). "Probabilistic analysis of linear programming decoding," IEEE *Trans. Inf. Theory*, 54(8) pp.3565–3578.
21. SRINIVASA SAI ABHIJIT CHALLAPALLI (2024). Optimizing Dallas-Fort Worth Bus Transportation System Using Any Logic. Journal of Sensors, *IoT & Health Sciences*, 2(4), 40–55.
22. MASERA, G., QUAGLIO, F., AND VACCA, F. (2005). 'Finite precision implementation of LDPC decoders', IEE Proc., Commun., 152, (6), pp. 1098–1102
23. BLANKSBY, A.J., AND HOWLAND, C.J. (2002). 'A 960-mW 1 Gb/s 1024-b, rate 1/2 low-density parity-check code decoder', IEEE *Journal of Solid-State Circuits*, 37, (3), pp. 404–412
24. SRINIVASA SAI ABHIJIT CHALLAPALLI (2024). Sentiment Analysis of the Twitter Dataset for the Prediction of Sentiments. Journal of Sensors, *IoT & Health Sciences*, 2(4), 1–15.
25. LIAO, E., YEO, E., AND NOKOLIC, B. (2024). 'Low-density parity-check code constructions for hardware implementation'. Proc. IEEE *Int. Conf. Commun*. (ICC), Paris, France, pp. 2573–2577
26. ROBERTSON, P., VILLEBRUN, E., AND HOEHER, P.(1995). 'A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain'. Proc. IEEE Int. Conf. Commun. (ICC), Seattle, USA, pp. 1009–1013