




# A Dynamic Hybrid Decoder Approach Using EG-LDPC Codes for Signal Processing Applications

J. Chinna Babu<sup>1</sup> · Nuka Mallikharjuna Rao<sup>2</sup> · Kadiyala Ramana<sup>2</sup> · Vidhyacharan Bhaskar<sup>3</sup> 

Accepted: 9 August 2021 / Published online: 23 September 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

## Abstract

The Low-Density Parity Check (LDPC) codes of Euclidean Geometry (EG) are encrypted and decrypted in numerous ways, namely Soft Bit Flipping (SBF), Sequential Peeling Decoder (SPD), Belief Propagation Decoder (BPD), Majority Logic Decoder/Detector (MLDD), and Parallel Peeling Decoder (PPD) decoding algorithms. These algorithms provide a extensive range of trade-offs between latency decoding, power consumption, hardware complexity-required resources, and error rate performance. Therefore, the problem is to communicate a sophisticated technique specifying the both soft and burst errors for effective information transmission. In this research, projected a technique named as Hybrid SBF (HSBF) decoder for EG-LDPC codes, which reduces the decoding complexity and maximizes the signal transmission and reception. In this paper, HSBF is also known as Self Reliability based Weighted Soft Bit Flipping (SRWSBF) Decoder. It is obvious from the outcomes that the proposed technique is better than the decoding algorithms SBF, MLDD, BPD, SPD and PPD. Using Xilinx synthesis and SPARTAN 3e, a simulation model is designed to investigate latency, hardware utilization and power consumption. Average latency of 16.65 percent is found to be reduced. It is observed that in considered synthesis parameters such as number of 4-input LUTs, number of slices, and number of bonded IOBs, excluding number of slice Flip-Flops, hardware utilization is minimized to an average of 4.25 percent. The number of slices Flip-Flops resource use in the proposed HSBF decoding algorithm is slightly higher than other decoding algorithms, i.e. 1.85%. It is noted that, over the decoding algorithms considered in this study, the proposed research study minimizes power consumption by an average of 41.68%. These algorithms are used in multimedia applications, processing systems for security and information.

**Keywords** Low Density Parity Check (LDPC) · EG-LDPC · Soft errors · Burst errors · Decoding algorithm · HSBF (SRWSBF) · Multimedia applications · Security

---

✉ Vidhyacharan Bhaskar  
meetvidhyacharan@yahoo.com

Extended author information available on the last page of the article

## 1 Introduction

### 1.1 Low Density Parity Check Codes

The LDPC codes were a better-quality nature of fault detection and pattern improvement schemes, which are considered to be difficult to decode, operating speed and error rate performance [1]. Many techniques have been developed to create LDPC codes to effectively transmit the Shannon theorem to the channel. In this field of coding theory, rapid progress has been shown to take improvement of the LDPC codes in areas like digital streaming in satellite systems, Data communication systems, data stored and mobile telephony systems [2].

### 1.2 LDPC Representation

As recommended by the classification, LDPC codes are the superior form of direct mass codes with different sizes of equivalence regulated matrices that contain fewer number of ones. This parity check matrix is generally created by using random method depends on certain rigorous limitations. LDPC codes are distinct from the useless interplanetary matrix (H) with structural characteristics of the equivalence verified. Each row consists of 'ρ' number of 1's. Every column is made up of the 'γ' number of 1's. The common number of the 1's in any two columns is 'λ' and should be no better than one. Both 'ρ' and 'γ' are less related to code distance and number of rows in H [3].

Since 'ρ' and 'γ' are small, and the matrix H has a smaller concentration and a sparse matrix. Therefore the code generated by the H matrix is known as the LDPC code. The parity check matrix is structured at the earliest to design the LDPC codes and afterwards to measure the originator matrix for the pattern. The primary change between LDPC and traditional coding patterns is to decode patterns of code. Usually, these LDPC decoding patterns are classified into 2 categories. They are matrix form and pictorial form [4].

$$\text{Matrix representation : (Ex : H) = } \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

The presented matrix size is 8 by 4, comprising n-adjustable nodes and k-patterned nodes. Each row weights 6 and each column weight is determined by 3, whereas the total block length 'n' and number of message bits k and equivalency bits n-k are applied to each size word code (n,k). The H-matrix is only one when there is a relation between the Adjustable check nodes and modeled variable nodes intended for every column essentials and row essentials of the pictorial representation. [5–7]. Such a pictorial representation consists of n number of variable nodes (i.e. n=8) and k number of check nodes (i.e. k=4) as shown in Fig. 1 [8].

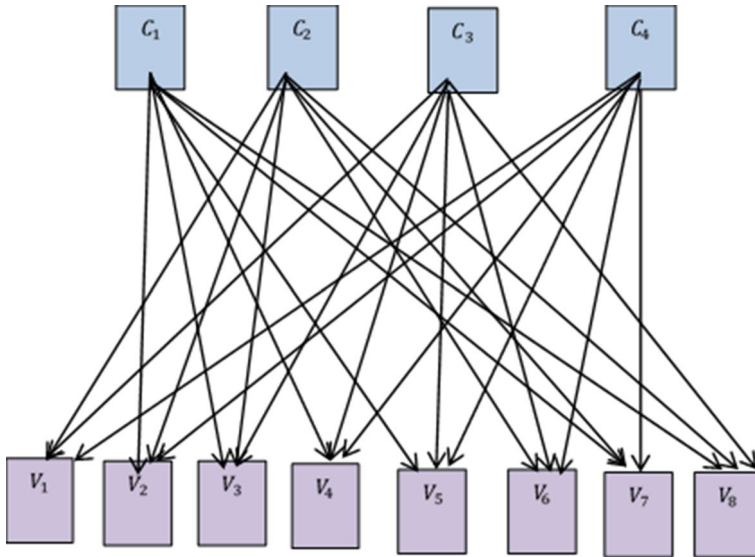


Fig. 1 H-matrix graphical representation for the above example

### 1.3 Creation of Patterned matrix

The  $H_{M \times N}$  equivalence form matrix is well-defined as  $(N, K)$ , where  $N$  is overall code pattern length,  $K$  is amount of definite data bits and it contains  $N - K$  quantity of equivalence bits [8, 9].

### 1.4 Process for scheming the equivalence patterned matrix for LDPC code patterns

#### Step 1 Scheming of Primeval polynomial

Let us deliberate the odds between 1 and 16 and write the odd numbers in binary form. Attach MSB bit '1.' Now reverse the bits which are obtained after appending. Then, associate the concatenated bits with the complemented bits. If both binary streams are equal then ignore the binary streams otherwise and store the numbers. Then the primeval polynomial can be considered as the least value of left over bit streams [10]. All the equations in Sect. 1.4 were considered from the study [10].

$$11001 \rightarrow x^4 + x^3 + 1 \text{ and Primitive polynomial } P(x) = x^4 + x^3 + x + 1$$

#### Step 2 Determination of Degree terms

While calculating the degree term value, if the value exceeds the 16, then XOR it with the primitive polynomial  $(x^4 + x^3 + x + 1)$ ,  $p(x) = 11001$ .

#### Step 3 Determination of Generated polynomial

The generalized method of generated polynomial  $G(x)$  is given by

$$G(x) = (x - \alpha^0)(x - \alpha^1)(x - \alpha^2) \dots \dots \dots (x - \alpha^{n-k-1})$$

where  $n, k = 16, 8$  therefore

$$G(x) = (x - \alpha^0)(x - \alpha^1)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)(x - \alpha^5)(x - \alpha^6)(x - \alpha^7)$$

By solving above equation the result is

$$G(x) = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1(11111111)$$

Step 4 Construction of parity patterned matrix

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

The first column is obtained by writing bits of  $\alpha^0 = 1 = 0001$  after LSB and the following step is downshifted to get the data in column 2 and consequently in column 1. If '1' is found in a column as the last bit, we can XOR shift data with  $G(x)$  to get that column for the next shift, otherwise the data would be moved to the left/down side. This is the last bit in column 3: '1' so that it is reversed to XOR the generator polynomial  $G(x) = [1 1 1 1 1 1 1]$  to get the information available in column 4. Like wise, it can find out the parity check matrix  $H$  for twofold LDPC patterns [11, 12]. From this method, we can generate the code vector input by taking the product of the initiator matrix 'G' and the message vector 'm' and the resultant is given by  $c = m.G$

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

Consider the analyse the 8-bit data input  $m = [10100]$ , which is encoded by reproducing this code in an originator matrix, i.e. the coding word resultant is  $C = [1001100]$ .

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

$C = [10011010010100]$ , subsequently the resulting data vector is considered to be valid, if and only if the vector measurement is  $Z = CH^T = 0$ , fulfilled, at this time the information amount presumed to be (12, 6) indicates the sum of bits and message bits between transmitter and receiver used for data transmission [13].

The remaining part of the paper in detailed is as follows. In Sect. 2, decoding of LDPC codes is reviewed and discussed. Section 3, few decoding algorithms are pre-

sented. In Sect. 4, recreation process used for the evaluation of data transmission and outcomes investigation is represented and Sect. 5 represents the conclusion work with a summarization.

## 2 Decoding of LDPC Codes

In the studies [14–16], the algorithm SBF, Min-Sum (MSA) algorithm and MLDD algorithm were proposed to decode LDPC codes on VLSI. These approaches are considered as two parameters for estimating system performance, such as hardware consumption and delay. MSA's hardware complexity is slightly higher than MLDD and SBF algorithms, relative to MSA and SBF algorithms, whereas the delay is slightly lower in MLDD algorithms. These studies describe the low complexity of hardware for use in applications for medical and signal processing.

In the study [17], the algorithm SBF and MLDD algorithm was proposed to decode LDPC codes on VLSI. These approaches are considered as two parameters for estimating system performance, such as hardware consumption and delay. SBF's hardware complexity is slightly higher than MLDD, whereas MLDD's delay compared to SBF's algorithm is slightly less. These studies identify the low complexity of the hardware to be used in applications for barcoding and signal processing. A novel Weighted Bit-Flipping (SRWBF) self-reliability-based decoding technique has been proposed for LDPC codes in the study [18]. This research examined the algorithms of Bit Flipping, namely weighted bit blipping algorithms, updated weighted bit flipping algorithms and weighted bit flipping algorithms with a reliability ratio. Both methods are a trade-off between complexity decoding, speed decoding and rate of error. The proposed approach performs data transmission considering two types of information transmission techniques to evaluate each bit's error term, check node information, and intrinsic information. Study has shown through simulations that the difficulty of decoding has substantially decreased. The latest algorithm comparison study is as shown in Table 8. The research analysis is carried out using an adjusted weighted bit flipping and weighted bit flipping decoding based on self-reliability.

The LDPC code is a simple computer-enhancing error; it is a communication technique through a noisy broadcast network. This paper analyzed the possible reasons leading to the problems for transmitting a message between transmitter and a receiver considering soft and burst errors in Soft Bit Flipping (SBF) [19], the Weighted Bit Flipping (WBF) [20], Belief Propagation Decoder (BPD) [21], Sequential Peeling Decoder (SPD) [22] and Parallel Peeling Decoder (PPD) [23] decoding algorithms. Soft error is an observable change in state of system where as burst error is a contiguous sequence of symbols received over a communication channel while transmitting messages between transmitter and a receiver. It is notice that the above mentioned algorithms are having minimum signal transmission rate and maximum power consumption, maximum decoding latency and higher decoding complexities. Thus, it motivated to formulate a model to considering Euclidean Geometric based Low Density Parity Patterned (EG-LDPC) decrypting approach by employing HSBF decoder for maximizing signal transmission rate and minimizes the power consumptions, decoding latency and decoding complexities [24]. The generic block diagram of decoding is shown in Fig. 2 [8, 15].

The objectives of the paper are:

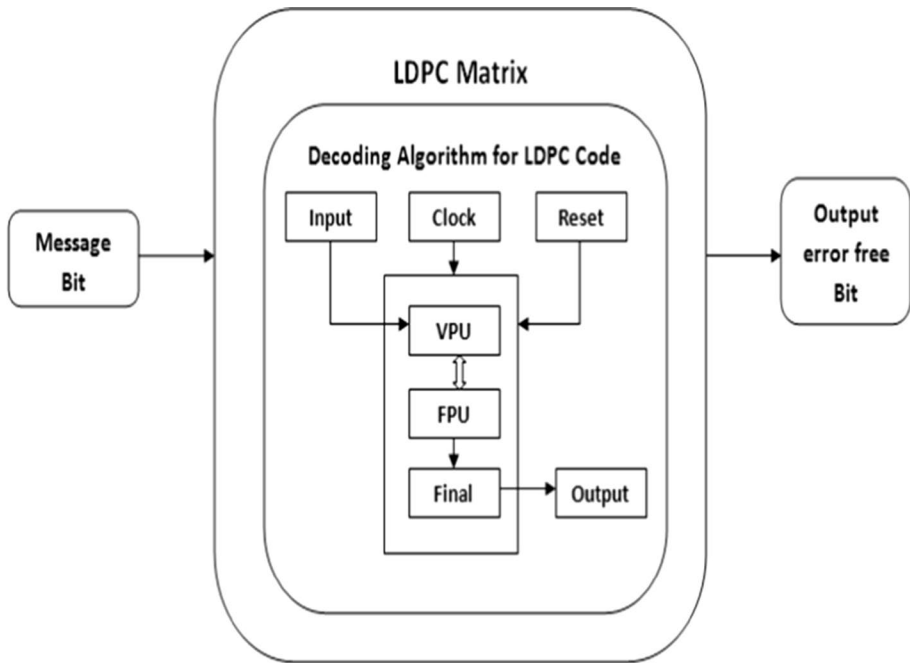


Fig. 2 Schematic diagram of LDPC decoder

1. To develop an algorithm that employs HSBF, this receives 16-bit code pattern and assigned to 16 Adjustable nodes. It verifies the nodes connections with all Adjustable nodes and compute patterned node values and also performs majority patterned operations are performed. This process is to continue till the end of operations.
2. To develop a simulation model for proving experimentally the performance of HSBF algorithm is better than that of SBF, MLDD, BPD, SPD and PPD decoding algorithms.

The criteria of optimization for such an algorithm are signal transmission rate and static and dynamic power consumptions and decoding complexity. Obviously, signal transmission rate is maximized while static and dynamic power consumption and decoder complexities to be minimized.

The common block schema for LDPC decrypting algorithms is as presented below and described in brief in Sect. 2.1.

## 2.1 General Procedure for encrypting and decrypting of LDPC code patterns

The equality checked matrix  $H_{M \times N}$  is distinct by a pattern  $(N, K)$ , here  $N$  is complete data size and  $K$  is sum of the definite information bits and  $N - K$  indicates the amount of equivalence bits. The originator matrix  $G$  can be attained from the  $H$ -matrix by considering the below metrics, by reorganising the  $H$ -matrix by using row and column operations  $H_{sys} = [Im|Pm \times k]$ . Reorganize the systematic parity patterned matrix ( $H_{sys}$ ) as  $G = [P^T k \times m | I_k]$ . Verify the results or shown as  $G.H^T = 0$ . It could generate the data

vector by multiplying message bits  $m$ , with the generator matrix  $G$ . The encryption data pattern is said to be permitted, if it fulfils  $C \cdot H^T = 0$ .

## 2.2 Brief discussion of the Existing Decoding Algorithms

### 2.2.1 Soft Bit Flipping Algorithm

To decode LDPC codes, the SBF decoding method is considered a hard decision message passing technique. Each step begins with the detection of a binary (hard) bit, which is then passed to the decoder for further processing before being passed to the SBF algorithm via Tanner graph edges to indicate zeros and ones (called as binary data). Such bits are either zero or one when a bit node sends a message in SBF decoding for declaration. Every check node will send a message to every connected bit node based on the information available at the bit node. The check node uses the module-2 operation to establish its parity check restrictions, and the sum of the resultant bit value is zero.

Furthermore, received bits may differ from broadcast bit nodes, and if any erroneous bits are identified, they are verified bit by bit and flipped out. This procedure is repeated until the system is shut down or the decoder has completed the maximum number of iterations. If a valid codeword is identified, the SBF decoder operation may be terminated. Decoding delay, power consumption, and hardware complexity are all constraints of the SBF decoder.

### 2.2.2 Majority Logic Decoder/Detector Algorithm

When contrasted to the MLD (Majority Logic Decoder) algorithm, the MLDD functions as an error detector and corrector with less iterations. This approach can be used with binary LDPC codes. The majority logic decoding algorithm is used to decode the data. Instead of decoding all of the codeword bits, this algorithm moves straight to the third cycle, pausing the MLDD method in the middle and detecting up to five bit-flips in the third decoding cycle. To increase performance, the number of decoding cycles can be lowered.

### 2.2.3 Sequential Peeling Decoder Algorithm

During the decoding process, SPD promises to decrement in irresolvable variable nodes and discover the outcomes. Peeling decoder is a binary channel algorithmic translation tool. Sequential Peeling Decoding is the simplest instance for the iterative message-passing calculation.

### 2.2.4 Parallel Peeling Decoder Algorithm

Iterative message-passing calculation is supported by the PPD. In each cycle of the general code word, the message is transmitted from check nodes to variable nodes. For comparison to SPD, the PPD employs unexpected planning. Rather than settling a single grade-1 check node per cycle, all deg-1 check nodes that are available are settled. In comparison to SPD, PPD uses unanticipated planning. Instead of settling just one grade-1 CN per emphasis, all grade-1 CNs that are accessible are settled.

### 2.2.5 Belief Propagation Decoder Algorithm

The message is transmitted through the surrounding variable hubs and they are associated with each other in an iterative procedure. BPD is an important method, and it is one of the iterative decoding methods. In order to pass the message, the BPD uses an iterative method in which neighbouring variable nodes interact with one another. Belief propagation decoding can help with chart formation and message transmission. The belief propagation limit is reached when the channel parameter with the lowest unravelling error likelihood is equal to zero.

## 3 Proposed Decoding Algorithm

In this section, presented a proposed algorithm known to be HSBF decoding algorithm used in information processing systems.

### 3.1 HSBF (SRWSBF) Decoding Algorithm

The flow graph of HSBF algorithm is as shown in Fig. 3[13] and the pseudo code of HSBF algorithm is given below.

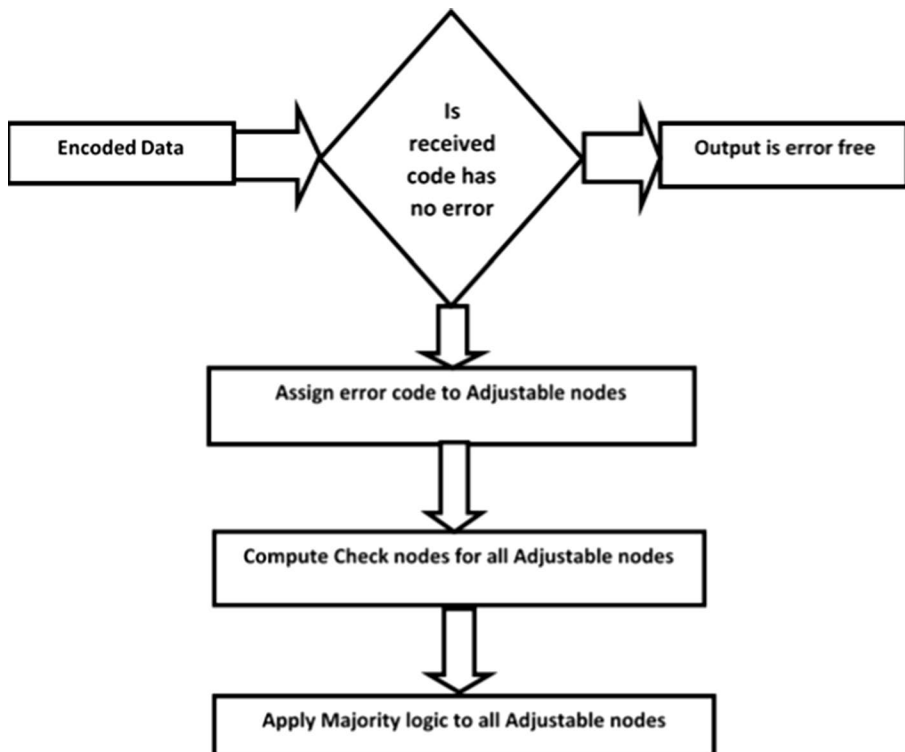


Fig. 3 Flow graph of HSBF decoding process



### 3.1.1 Mathematical Model

Assume the received real value sequence  $y = y_1, y_2, y_3 \dots y_n$  corresponding binary hard decision sequence be  $Z = Z_1, Z_2 \dots Z_n$ . LDPC code has a tanner graph in which all the variable nodes have degree  $j$ , and all the check nodes have degree  $k$ .

LDPC code can be represented as  $N, K, d_v, d_c$ , where  $N, K$  denotes the length of the codeword and the information bits respectively and  $d_v, d_c$  means the column and row weights.

A significant minimum gap ensures accuracy against random errors. However if the possibility of sending code words with nearby code words is poor, a code with a small minimum distance can be accurate.

Given  $n$ -bit codeword compute  $(n - k)$  syndrome bits.

$$\text{Received word } R = C + E$$

Where  $C$  is a received codeword, and  $E$  is a random variable.

$$\text{Compute syndrome vector } S = R.H^T$$

$$l \times k \qquad k \times n \qquad \Rightarrow \qquad l \times n$$

Message Vector Generator matrix Codeword Vector.

$$\text{The generator matrix } G_{k \times n} = [I_{k \times k}, A_{k \times (n-k)}]$$

$$\text{Extracting } d \text{ from } H. \text{ Where } H = [A^T, I_{(n-k) \times (n-k)}]$$

The Minimum distance  $d$  of the code  $'C'$  is the minimum number of linearly dependent columns of  $H$ , i.e., that can be combined to give the zero vector.

$d$  – Minimum weighted non-zero codeword in  $C$ .

### 3.2 Syndrome Vector Steps

1. For a given codeword error pattern,  $E_i$  is computed from the syndrome vector and update the syndrome vector using the expression [15].

$$S_i = E_i.H^T \tag{1}$$

2. Compute the code pattern for each received codeword at the receiver end.

HSBF (SRWSBF) Algorithm operation can be performed as follows:

*Step 1* Receive the input block  $y = \{y_1, y_2, y_3 \dots y_n\}$  and get the hard decision sequence  $z = \{z_1, z_2 \dots z_m\}$ . Based on the information (block size), compute the syndrome vector, and finds the most unreliable message word participating in each check mentioned in Eqs. (2) and (3) [17].

$$S_m = \sum_{n=1}^N Z_n.H_{mn} \tag{2}$$

$$y_m^{min} = \min_{n \in N(m)} |y_n| \tag{3}$$

where  $|y_n|$  denotes the absolute value of the  $n$ th message node soft value, while  $y_m^{min}$  is the lowest magnitude of all message nodes participating in the  $m$ th check.  $N(m)$  Denotes the set of variable nodes that can participated in them<sup>th</sup> the  $m$ th check node.

*Step 2* Compute  $E_n$  by using

$$E_n = \sum_{m \in M(n)} (2S_m - 1) \times t_n \quad (4)$$

*Step 3* Find at the most unreliable message node connected to each check node.

$$r_m^{\min} = \min_{n \in N(m)} |y_n| \quad (5)$$

For each message node, compute the error metric

$$E_n = \min_{m \in M(n)} (2S_m - 1) \times t_n \quad (6)$$

*Step 4* Flip the bit with the highest  $E_n$  by performing the Maxlikelihood logic operation and update the syndrome vector  $S = R.H^T$  and the Maxlikelihood logic operation is given by

$$S_n \leftarrow S_n \cup (E_n \text{ xor } S_n) \quad (7)$$

*Step 5* If the maximum number of iterations is reached by the algorithm or all the parity check equations are satisfied, the algorithm is terminated; otherwise, the algorithm is repeated from step 1 to step 3.

In this algorithm, in determining the error term for each bit, two kinds of information need to be considered. These are the control node data and the basic data. In fact, without passing a message, the information obtained from their neighbours in each variable node is very small. The  $2S_m - 1$  term should carry enough information from the check nodes, it is noted. In comparison to the reliability of the neighbour variable nodes participating in the same control nodes, Self-Reliability  $|y_n|$  should therefore be considered more. The latest word used for the error is

$$E_n = \sum_{m \in M(n)} (2S_m - 1) \times t_n \quad (8)$$

Ignorance of the reliability of neighbouring variable nodes can significantly reduce the difficulty of decoding.

### 3.3 Pseudo Code for HSBF Decoder

```

HSBF DECODE (y,z)
1      i ← 1 & j ← 1      ▷ Initialization
2      loop ← ∞
3      do
4      {
4      for i ← 1:n do
5      yi ← yi ∪ y
6      end for
7      for j ← 1:m do
8      zj ← zj ∪ z
9      end for
    ▷ Compute Syndrome vector
10     for j ← 1:m do
11     for i ← 1:n do
12     Sm ← ∑n=1N Zm × Hmn
13     end for
14     end for
    ▷ Compute yn
15     for i ← 1:n do
16     for j ← 1:m do
17     ymmin ← |yn|
18     end for
19     end for
    ▷ Compute En
20     do
21     {
22     for i ← 1:n do
23     for j ← 1:m do
24     En ← ∑m ∈ M(n) (2Sm - 1) × tn
    ▷ Flip the bit with biggest En and update the Syndrome vector
    ▷ Perform XOR operation
25     Sn ← Sn ∪ (En xor Sn)
26     } While (Sn ← 0)
27     for i ← 1:n do
28     for j ← 1:m do
    ▷ Test: the parity check equations are satisfied or not
29     Lj ← ∑i' ∈ N(m) (Si' mod 2)
30     end for
31     if all Lj ← 0 or i ← imax then
32     end of operation
33     else
34     i ← i + 1
35     endif
36     } While (loop ← ∞)
end PROCEDURE

```

The lines 1 and 2 initialize the block, which receive the set of bits and hard decision sequence. The do-while loop that spans from lines 3 to 36 is executed until the end of system operations. The lines 4 to 9 initialize the real sequence values and corresponding binary hard decision sequence. The lines 10 to 19 are used for computing syndrome vector and the lowest magnitude of all messages nodes participating in the  $m^{th}$  check node. The do-while loop that spans from lines 20 to 26 is executed until the syndrome vector becomes zero. At the same time, it determines the flip bit with the most significant value of  $E_n$ . The lines 27 to 35 are used to test the parity check, whether the parity check equations are satisfied or not.

### 3.3.1 Computational Analysis

The proposed self-reliability based weighted bit flipping decoding algorithm is more hardware friendly, compared with the other bit flipping decoding algorithms.

Initially, the number of wires required in the Self-reliability Weighted Bit Flipping algorithm is to be examined.

$$E_n = \frac{\sum_{m \in M(n)} (2S_m - 1)}{|y_n|} \tag{9}$$

From Eq. (8), to compute the error term  $E_n$ , each variable node required two inputs:  $S_m$  and  $|y_n|$ . The  $|y_n|$  is the magnitude of the received value, which is stored inside of the each variable node. The input bit  $S_m$  is received from its neighbor check and executes one bit at a time. Similarly, each check node required the sign of every variable node to compute  $S_m$ , and it can be processed one bit at a time. Hence, each edge in the Tanner graph maps into one pair of wires only.

The proposed decoding algorithm used  $\frac{(q-1)}{q}$  ( $q$  is the quantization bits) wires only, whenever comparing with the other traditional bit Flipping algorithms. The proposed decoding algorithm significantly eliminates the routing problem for implementing with considerable codeword length.

In existing decoding algorithms, the divider imposed in each variable node, which makes the decoder design a little complex. By applying the modifications on Eq. (9) and re-written the statements, as given below:

$$t_n = \frac{1}{y_n} \tag{10}$$

$$E_n = \sum_{m \in M(n)} (2S_m - 1)Xt_n \tag{11}$$

Note that the division operation in Eq. (8) is performed based on the received value for each bit. Furthermore, the received sequence  $y$  comes into the decoder successively bit by bit. Therefore, only one simple multiplier is needed at the first receiving of  $y$ . Where  $t_n$  is stored in every variable node instead of  $y_n$ . This multiplier can be implemented by using a LUT operation, which is of very high speed and area-saving for small input word length.

The proposed decoding algorithm is evaluated through simulations. A four, eight, sixteen, and thirty-two bits quantization are considered to assess the performance of the algorithm. For example, a four-bit quantization is viewed as the trade-off between

**Table 1** Logic function usage per iteration in each variable node ( $y_n$ ) and each check node ( $C_n$ ) for various decoding algorithms (as per pseudo code)

Algorithm	WBF	MLDD	MWBF	BPD	RRWBF	HSBF (SRWSBF)
ADD	–	1	-	1	1	–
MUL	1	1	1	1	1	1(simple)
DIV	1	-	1	–	1	–
NOT	1	1	1	1	1	–
XOR	1	1	1	1	1	1

**Table 2** Latency comparison (per iteration) of various decoding algorithms

S.No	Name of the Algorithm	Latency per iteration
1	WBF Algorithm	38 Cycles
2	MLDD Algorithm	09 Cycles
3	MWBF Algorithm	38 Cycles
4	BPD Algorithm	09 Cycles
5	RRWBF Algorithm	39 Cycles
6	HSBF (SRWSBF) Algorithm	07 Cycles

complexity and decoding performance, one bit for the sign, and one more bit for the integer part and two for the fraction part.

$$E_n = \sum_{m \in M(n)} (2S_m - 1)Xt_n \tag{12}$$

The multiplication term specified in Eq. (11) has minimal possibilities. By comparing proposed algorithm with other decoding algorithms, existing decoding algorithms usually need more operations in calculating  $E_n$ . Due to this complex operations, existing decoding algorithms are hardware hungry and time-consuming.

Table 1 shows the logic function usage per iteration in each variable node and check node (as per pseudo code) of the bit Flipping-based and peeling decoding algorithms. It indicates that HSBF (SRWSBF) has significantly low complexity. If the computational complexity is less, then hardware complexity can also be minimized. It results that power consumption can be reduced with less hardware utilization Table 2.

The most modern processors perform the standard integer operations in one clock cycle, except in case of multiplication and division (if available). During the processing, the Multiplication may consume around six cycles, and the division may consume 30 to 60 cycles. It all depends entirely on the specific CPU. However, the simple bitwise operations may consume typically one cycle, regardless of operand width by using modern processors. Most of the Modern processors can do many bitwise and/or/xor operations in a single cycle. From the above Table 1, weighted bit flipping algorithm (WBF) and Modified WBF algorithms consume 38 cycles, MLDD consumes 9 cycles, BPD consumes 9 cycles, Reliability Ratio Weighted Bit Flipping (RRWBF) consumes 39 cycles, and SRWSBF consumes only 7 cycles. With this, the proposed HSBF (SRWSBF) consumes a minimum number of cycles to perform one iteration. From this, this work concludes that HSBF has less latency compared to other decoding algorithms. If the computational complexity and

latency are less, then hardware complexity can also be minimized. From this, it can conclude that power consumption is also minimized with the utilization of less hardware.

## 4 Simulation Results

In this region, modeling process and outcomes are presented. Simulation is performed by using Xilinx Model Sim simulator. Experimental setup done through Test Bench which validates complete data, thus it reduces the test times and data footprints for every validation. This Test Bench process improves the testing productivity and accuracy of data. Experimentation done through Xilinx Model Sim by employing HSBF algorithm. Basing on the results through simulation, the proposed algorithm is evaluated and compared with the decoding algorithms such as MLDD algorithm, SBF Algorithm, BPD [13], SPD algorithm and PPD algorithm.

### 4.1 Simulation Results

In this unit, the modeling and synthesis outcomes for HSBF algorithm are documented in the following. In this, we patterned the decoding latency shown in Fig. 4. The comparative results for various parameters for these algorithms are also provided in this chapter. The self-reliability based biased soft-bit-flipping algorithm for EG-LDPC codes were defined by using Verilog code. Xilinx ISE Simulator is used to observe the results.

### 4.2 Discussions

This Technique and Simulation section discusses the findings of decoding algorithms. The proposed research study contrasts HSBF decoding algorithm results with current decoding algorithms in terms of latency, hardware usage or complexity and power consumption. The following three metrics are used to calculate the latency of decoding. Simulation values are derived from the Xilinx synthesis Analyzation summary and the SPARTAN 3e. In Table 1, the resulting values are summarised.

The latency metric performance is computed using Eq. (13) [16].

$$\text{Latency} = CC * CP \quad (13)$$

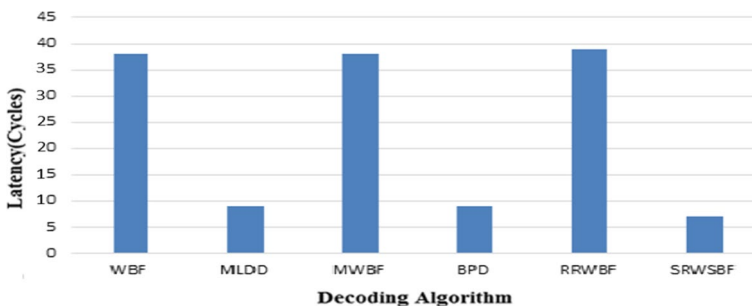


Fig. 4 Latency comparison of various decoding algorithms

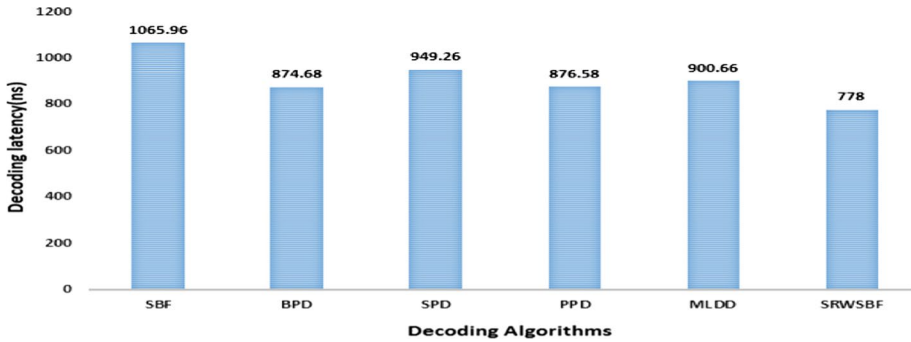


Fig. 5 Decoding latency comparison over various algorithms

where  $CC = \text{Total number of Clock Cycles to obtain output}$

$CP = \text{Minimum required Clock Period}$

The total number of clock cycles referred to as  $CC$  is computed from Eq. (14) [17].

$$CC = \frac{\text{Time period needed for decoding the output}}{\text{Time period Interval}} \tag{14}$$

where HSBF also known as SRWSBF Algorithm. Figure 5 shows the Decoding Latency comparison over various algorithms. From this figure, it is concluded that HSBF took less latency compared to other conventional algorithms. Figures 6 and 7 shows the consumption rate, which clearly shows that the use/complexity of the hardware is reduced when comparing HSBF, a proposed research with decoding algorithms SBF, SPD, PPD, BPD and MLDD. It is also found that slightly maximum number of slice Flip-Flops were used by the proposed research study HSBF decoding algorithms.

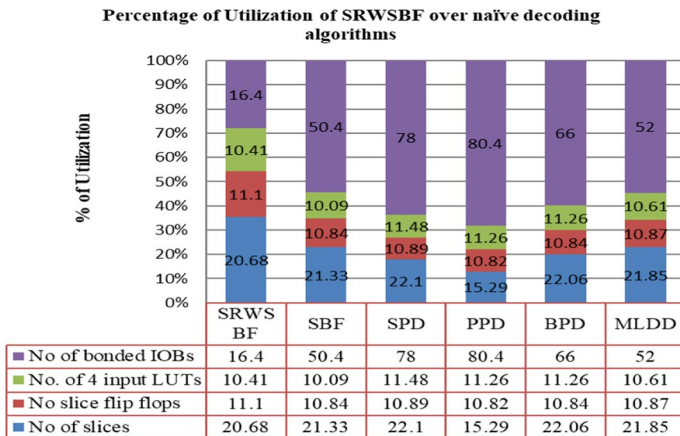


Fig. 6 Hardware utilization of HSBF algorithm over other decoding algorithms

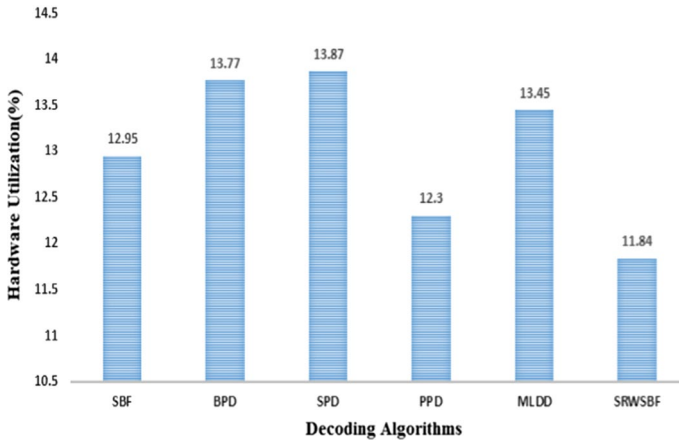


Fig. 7 Hardware Utilization of various algorithms

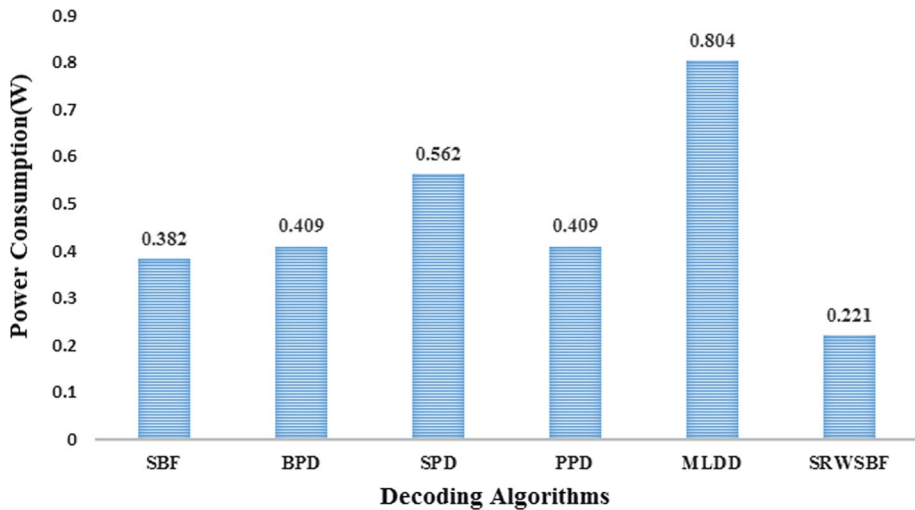


Fig. 8 Power consumption of HSBF algorithm over naïve decoding algorithms

It is observed that the various synthesis parameters minimize the use of hardware and few synthesis parameters increase the use of hardware compared to the proposed HSBF decoding algorithm research study. In this section we introduce another performance metric called power consumption, which estimates the power requirement for signal decoding in communication systems. The following parameters are used for the measurement of power consumption. Simulation values of the Xilinx Synthesis Study and SPARTAN 3e Gain. The resulting values are shown in Fig. 8. HSBF consumption is reduced by 42.15 percent compared to the SBF decoding algorithm. 45.90 percent less power consumption in HSBF compared to the algorithm for BPD decoding. The HSBF algorithm reduces power consumption by up to 59.52 percent over the decoding algorithm for PPD. Nearly 60.68 percent of power consumption is lower in the HSBF algorithm as compared to the SPD decoding algorithm in the research study. Consequently, the proposed HSBF research study



consumes less power consumption compared to existing decoding algorithms and enhances the efficiency of the data transmission communication system without interruptions.

## 5 Conclusion

In this paper, the HSBF decoding algorithm is proposed for Euclidean Geometric-LDPC decoding used in multimedia, Security and Information processing systems for minimizing the decoding complexity and maximizing Information transmission and reception. A simulation model is presented to evaluate performance of the proposed algorithm along with SBF, BPD, SPD, MLDD and PPD algorithms by employing Xilinx Model Sim 14.7 ISE simulator considering dynamic power and logic utilization factors for 4-bit, 8-bit, 12-bit and 16-bit soft and burst errors. Owing to proposed algorithm, performance is increased whenever the signals are transferring between transmitter and receiver and reduces the number of resources usage; reduce hardware complexity, decoding latency and also power consumption. Henceforth, the proposed HSBF algorithm is better than the other existing decoding algorithms using in Euclidean Geometric -LDPC Codes. It is worth to undertake additional experimentation to test for 32-bit, 64-bit and 128-bit soft and burst errors and improve the Information transmission rate and complexity of EG-LDPC decoding.

**Author contributions** J. ChinnaBabu: Conceptualization, Methodology, Software. Writing-original draftN. Mallikharjuna Rao: Data curation, Writing-original draft, Visualization, Investigation. Kadiyala Ramana: Software, Validation, Editing, Writing-review & editing. Vidya CharanBhaskar:Supervision,Writing—review & editing.

**Funding** None.

**Availability of Data and Material (Data Transparency)** Data sharing is not applicable to this article as no new data were created or analyzed in this study.

**Code Availability (Software Application or Custom Code)** The code that support the findings of this study are available on request from the corresponding author. The codeare not publicly available due to containing information that could compromise the privacy of research participants.

## Declarations

**Conflict of interest** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

1. Kou, Y., Lin, S., & Fossorier, M. P. C. (2001). Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Transactions on Information theory*, 47(7), 2711–2736.
2. MacKay, D. J. C., & Neal, R. M. (1996). Near Shannon limit performance of low density parity patterned codes. *Electronics Letters*, 32(18), 1645–1646.
3. MacKay, D. J. C. (1999). Good error-correcting codes based on very sparse matrices. *IEEE transactions on Information Theory*, 45(2), 399–431.
4. MacKay, D. J. C. (1998). Gallager codes that are better than turbo codes. *Proc. 36th Allerton Conf. Communication, Control, and Computing*.

5. Fossorier, M. P. C., Mihaljevic, M., & Imai, H. (1999). Reduced complexity iterative decoding of low-density parity patterned codes based on belief propagation. *IEEE Transactions on Communications*, 47(5), 673–680.
6. Zhang, J., & Fossorier, M. P. C. (2004). A modified weighted bit-flipping decoding of low-density parity-patterned codes. *IEEE Communications Letters*, 8(3), 165–167.
7. Cho, J., & Sung, W. (2009). High-performance and low-complexity decoding of high-weight LDPC codes. *J. Korea Inf. Commun. Soc.*, 34(5), 498–504.
8. Chinna Babu, J., Chinnapu Reddy, C., & Giri Prasad, M. N. (2016). Comparison of technologies for the implementation of SBF decoder for geometric LDPC codes. *INDJST Journal*, 9(30), 1–9.
9. Chinna Babu, J. (2017). VLSI Implementation of Decoding algorithms for EG-LDPC Codes. *Elsevier Procedia Computer Science*, 115(3), 143–150.
10. Giri Prasad, M. N., Chinnapu Reddy, C., & Chinna Babu, J. (2017). Generation and Decoding of Non-Binary LDPC Codes using MSA Decoding algorithm. *Lecture Notes in Electrical Engineering*, 434, 583–591.
11. Chen, J., Dholakia, A., Eleftheriou, E., Fossorier, M. P. C., & Hu, X.-Y. (2005). Reduced-complexity decoding of LDPC codes. *IEEE Transactions on Communications*, 53(8), 1288–1299.
12. Palanki, R., Fossorier, M. P. C., & Yedidia, J. S. (2007). Iterative decoding of multiple-step majority logic decodable codes. *IEEE Transactions on Communications*, 55(6), 1099–1102.
13. Kudekar, S., Richardson, T., & Urbanke, R. (2013). Spatially coupled ensembles universally achieve capacity under belief propagation. *IEEE Transactions on Information Theory*, 59(12), 7761–7813.
14. Kumar, S., Young, A. J., Macris, N., & Pfister, H. D. (2014). Threshold saturation for spatially coupled LDPC and LDGM codes on BMS channels. *IEEE Transactions on Information Theory*, 60(12), 7389–7415.
15. Chinna Babu, J., Chinnapu Reddy, C., & Giri Prasad, M. N. (2017). Comparison of Various decoding algorithms for EG-low density parity check codes. *Lecture Notes in Electrical Engineering*, 442, 605–613.
16. Giri Prasad, M. N., Chinnapu Reddy, C., & Chinna Babu, J. (2020). Error Performance of Hybrid Weighted Bit Flipping decoder for Security Applications using EG-LDPC codes. *International Journal of Advanced Science and Technology*, 29(4), 669–678.
17. Sha, J., Gao, M., Zhang, Z., Li, L. (2006). Self reliability based weighted bit-flipping decoding for low-density parity-check codes”. *15th WSEAS international conference on instrumentation*
18. Chen, J., Dholakia, A., Eleftheriou, E., Fossorier, M. P. C., & Hu, X.-Y. (2006). Reduced-complexity decoding of LDPC codes. *IEEE Transactions on Communications*, 53(8), 1288–1299.
19. Palanki, R., Fossorier, M. P. C., & Yedidia, J. S. (2007). Iterative decoding of multiple-step majority logic decodable codes. *IEEE Transactions on Communications*, 55(6), 1099–1102.
20. Kudekar, S., Richardson, T., & Urbanke, R. (2013). Spatially coupled ensembles universally achieve capacity under belief propagation. *IEEE Trans on Inf Theory*, 59(12), 7761–7813.
21. Kumar, S., Young, A. J., Macris, N., & Pfister, H. D. (2014). Threshold saturation for spatially coupled LDPC and LDGM codes on BMS channels. *IEEE Transactions on Information Theory*, 60(12), 7389–7415.
22. Liu, S., Reviriego, P., & Maestro, J. A. (2012). Efficient majority logic fault detection with difference-set codes for memory applications. *IEEE Transactions Very Large Scale Integrations VLSI Systems*, 20(1), 148–156.
23. Chinna Babu, J., Prathyusha, S., & Usha Sree, V. (2014). Simulation and synthesis of majority logic decoder/detector for EG-LDPC codes. *Int J Comput Applications*, 104(8), 32–35.
24. Babu, J. C., & Prathyusha, S. (2014). Hard Decision and Soft Decoding Algorithms of LDPC and Comparison of LDPC With Turbo Codes RS Codes and BCH Codes. In *IRF International Conference*.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Dr. J. Chinna Babu** has received his B. Tech. degree from JNTU, Hyderabad. M. Tech. degree in VLSI System Design and Ph.D. in VLSI signal processing degrees from JNTUA, Ananthapuramu. Currently, he is working as Associate Professor in the Department of Electronics & Communication Engineering, AITS, Rajampet, Kadapa, A.P, and India. He has published a number of research papers in various National and International Journals and Conferences. His areas of interests are VLSI, Micro processor, Embedded Systems and Signal Processing.



**Dr. Nuka Mallikharjuna Rao** is the Dean of Academics, Annamacharya Institute of Technology and Sciences, Rajampet. He moved up to the position of the Dean, Academics after serving the college for twenty years in diverse positions, continuing in Administrative positions as the Chairman Board of Studies of Master of Computer Applications and the member of Academic Council. He has joined at AITS, Rajampet as an Assistant Professor in the Department of Computer Science and Engineering in the year 1999. Consequently, he moved up to the position of Head of the Department of Master of Computer Applications in the year 2001.



**Dr. Kadiyala Ramana** is currently working as Associate Professor in Department of Artificial Intelligence & Data Science, AITS, Rajampet, India. He obtained his Bachelor of Technology degree in Information Technology from JNTUH, Hyderabad, India, M. Tech. from Satyabhama University, Chennai, India and completed his Ph.D. from SRM University, Chennai, India. He has 14 years of experience in teaching and research. He authored more than 30 international publications in Hindawi, Springer, IEEE etc. He is editorial board member and reviewer of several journals of international repute. His research interests Wireless Sensor Networks, Software-Defined Networking with Machine Learning and Data Analytics.




**Dr. Vidhyacharan Bhaskar** received the B.Sc. degree in Mathematics from the University of Madras, Chennai, India in 1992, M.E. degree in Electrical & Communication Engineering from the Indian Institute of Science, Bangalore in 1997, and the M.S.E. and Ph.D. degrees in Electrical Engineering from the University of Alabama in Huntsville in 2001 and 2002, respectively. During 2002-2003, he was a Post Doctoral fellow with the Communications research group at the University of Toronto, Canada, where he worked on the applications of space-time coding for wireless communication systems.

During 2003-2006, he was an Associate Professor in the Department of Information Systems and Telecommunications at the University of Technology of Troyes, France. From Jan. 2007 to May 2014, he was a Full Professor in the Department of Electronics and Communication Engineering at S.R.M. University, Kattankulathur, India. Since 2015, he is a Professor in the Department of Electrical and Computer Engineering at San Francisco State University, San Francisco, California, USA.

His research interests include MIMO wireless communications, signal processing, error control coding and queuing theory. He has published 133 Refereed Journal papers, presented around 72 Conference papers in various International Conferences over the past 20 years, published a book on “Adaptive Rate Coding for A-CDMA Systems” in Jan 2011, a book on “Higher-Order s-to-z mapping functions for digital filters,” in March 2013, and has also co-authored a book on MATLAB in 1998. He has 930 Google scholar citations. He has advised 50 Masters students, 11 Doctoral students, and 3 Senior design projects.

He is an IEEE Senior member (SM-IEEE) and is a member of IET (M-IET, UK). He is a Fellow of Institute of Electronics and Telecommunication Engineers (F-IETE), and a Fellow of Institute of Engineers (F-IE), Kolkata, India. He is also a Life member of the Indian Society of Technical Education (LM-ISTE) and a member of the Indian Science Congress (M-ISC).

## Authors and Affiliations

J. Chinna Babu<sup>1</sup> · Nuka Mallikharjuna Rao<sup>2</sup> · Kadiyala Ramana<sup>2</sup> · Vidhyacharan Bhaskar<sup>3</sup> 

J. Chinna Babu  
jchinnababu@gmail.com

Nuka Mallikharjuna Rao  
drmallik2009@gmail.com

Kadiyala Ramana  
ramana.it01@gmail.com

<sup>1</sup> Department of Electronics and Communication Engineering, Annamacharya Institute of Technology and Sciences, Rajampet, Andhra Pradesh, India

<sup>2</sup> Department of Computer Science and Engineering, Annamacharya Institute of Technology and Sciences, Rajampet, Andhra Pradesh, India

<sup>3</sup> Department of Electrical and Computer Engineering, San Francisco State University, San Francisco, CA 94132, USA