Diagonal Parity Check Code for Three-Dimensional Error Detection and Correction of Multiple Errors

Adarsh Ranjan, Rian Shane Pinto, Siddharth Gupta
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore, India
9972854652, 9113041845, 9165166290
adarshranjan.221cs103@nitk.edu.in, rian.221cs144@nitk.edu.in,
siddharthgupta.221cs153@nitk.edu.in

No Institute Given

Abstract. In a data computer network, data transmission efficiency and reliability are crucial. Since communication systems are naturally nondeterministic, the likelihood of a mistake affects data transmission. During data transmission, two methods are typically used to manage errors in communication systems: channel coding and forward error correction (FEC). One technique to address this problem is to arrange data into several two-dimensional layers and calculate parity bits along different axes. Further enhancements were made by incorporating Hamming code for error control. Hamming code is utilized to identify and fix errors in redundant or generated parity check bits in received code words. This paper proposes a new error detection and correction method by employing parity check and Hamming codes on data bits arranged in a three-dimensional cube. The proposed method uses parity codes in cube layers' horizontal and vertical directions. Parity codes are also generated along the cube's height, along each plane, and across the three-dimensional diagonal. The proposed method effectively detects and corrects errors due to its low computational complexity.

1 Introduction

Due to the unreliability of communication channels, such as electromagnetic interference, noise, and signal attenuation, transmitted data over a network is susceptible to errors. Error correction becomes especially vital in long-distance communication. Implementing an efficient error correction mechanism contributes to cost efficiency as it minimizes the need for complete data re-transmission. Error correction methods are primarily segregated into linear and convolutional blocks [1]. To uphold consistency in transmitting messages, the fundamental requirement of all error-correcting algorithms is the addition of extra bits to the originally sent data. The presence of the redundant bits, along with the original message, form a new pattern of bits referred to as a code word. Suppose these extra or redundant bits are physically located at one of the extreme ends of a sequence of bits. In that case, such an error correction is a systematic error correction method. On the other hand, in a non-systematic method, the original message cannot be directly obtained from the code word. The extra bits are mixed in the code word. The algorithm is reapplied to the message bits on the receiving end, and a syndrome bit is generated. This syndrome gives sufficient information regarding the position of the erroneous bits. Since bits can only take values either 0 or 1, the presence of an error at a position can be rectified by simply toggling the bit.

A simple parity check refers to adding a bit to the original message, representing the modulo-2 sum or the logical XOR of the message bits. Generating parity bits and their subsequent location can help correct message bits accordingly. One such method is organizing data into multiple two-dimensional layers and generating horizontal vertical-diagonal (HVD) parity bits [1]. Because the HVD parity code has an extra diagonal verification element [2, 3], it secures the data section more precisely. A three-dimensional parity-checking system that uses parity planes can extend the security over a larger cluster of data layered in a three-dimensional structure. Further, the Hamming code secures against redundant bits resulting from burst mistakes in a layered data set. One challenge in three-dimensional data arrangement is increased complexity and computational requirements. Since this involves the application of multiple data dimensions, it creates the availability of more parity bits, which are also exposed to noise on their own. Combining horizontal, vertical, and diagonal error detection and correction might be computationally expensive, leading to slower execution. Another major problem is burst error.

This paper intends to implement a Diagonal Parity Check code on three-dimensionally arranged data bits with the expectation of having a reduced overhead and an increased code rate in the code word. The data bits are arranged in the form of a cube, from which parity bits are generated. The choice of parity bits is specific to the requirement of error correction in the main message bits, as discussed in Section 3. Further, redundant bits are generated on these parity bits using hamming code. The proposed method was tested on data in which multiple faults were injected, and the technique shows that almost all the injected errors could be detected. This method can correct six faulty bits, two data-bit, and four parity-bit errors.

The rest of this paper is arranged as follows. Section 2 reviews a few related works and publications in this domain and explains our motivation for producing this paper. Subsequently, Section 3 introduces the design and implementation of our proposed solution by laying the groundwork for our model's visualization and identifying different parity check bits within the model, followed by Section 4, where the result and analysis of previous works and the proposed solution are discussed. Finally, the paper concludes with Section 5 the future work and scope of this scheme.

2 Related works

This section compares a few papers that have implemented error correction algorithms that align with our model. The existing approaches check for duplicated or erroneous bits created by the data and use multi-directional parity and Hamming codes on communication systems to identify and fix errors in the data. The suggested error detection and correction approach uses the multi-directional parity code to identify and correct errors in the data portion.

2.1 Three-dimensional Parity-Check Codes for Correction and Detection of Multiple Errors

N. Babu Anne et al. [1] proposed a method of Three-dimensional parity check [1]. This is done by setting the message bits sequentially of a fixed length. Further, each sequence is placed over another, creating a plane of message bits of height and width equal to n. Making multiple such sequences and layering them, one over another, produces a three-dimensional visualization of the inputted message bits. To this, creating three-dimensional parity checking codes is possible. This method arranges information and parity bits into two-dimensional planes to form a cube shape, so the parity planes make a three-dimensional layer covering the data bit cube from 3 sides.

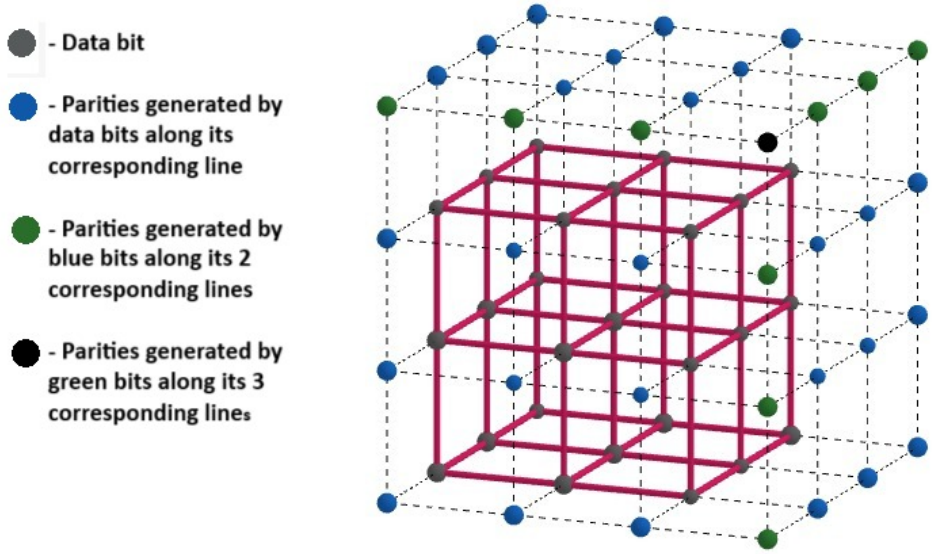

Fig. 1. The arrangement of the data bits and parity bits in Three-dimensional parity check [1]

In a cube with parity planes positioned at the edges, the study discovered the capability to rectify two errors and detect up to seven errors. Any single-bit error (x, y, z) can be detected using corresponding parity equations, but sometimes, these errors may go undetected.Fig. 2 shows two cases where the three-dimensional parity check fails to detect the 8-bit and 12-bit errors. Numerous other error patterns may need to be noticed, yet each pattern comprises more than eight bits. The unnoticed error pattern exists since parity codes cannot detect an even number of error patterns in data bits generating a single parity bit.

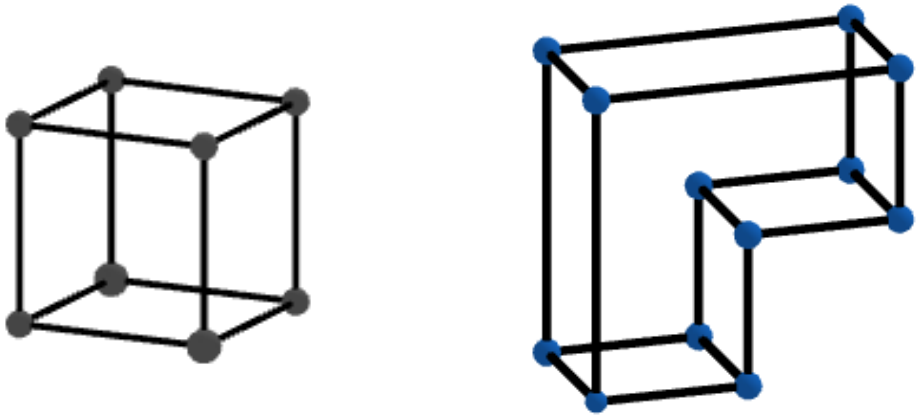

Fig. 2. The undetectable 8-bit and 12-bit error patterns [1]

2.2 Horizontal-vertical-diagonal error detecting and correcting code

M. Kishani et al. [4] proposed an error detection scheme, HVD code [4], to address the shortcomings of the previously suggested method. The data bits are two-dimensional, then parity bits are generated using columns, rows, and diagonals. Parity bits for the diagonal of two-dimensionally arranged data are generated in two directions, as shown in Figs. 3 and 4. A data block is divided into several smaller data segments to display trade-offs between parameters. Next, a few tests were conducted to examine the impact of the granularity of the data segments inside a data block. The outcomes demonstrate that the HVD approach can identify 100 percent of injected faults in the protected block and can be shown to fix up to three. The suggested approach uses parity codes with low processing latency and implementation complexity to increase fault detection and rectification coverage even though it has more excellent memory overhead than earlier approaches.

When the receiver captures the data for error detection, the receiver must calculate the parity bits for the row, column, and two diagonals. Since the computation of parity bits along different axes is an independent operation, the computation speed could be increased using parallel computing; this property significantly improves the speed of real-time and high-speed applications. Then, the receiver could match the calculated parity bits with the received parity bits; any mismatch in parity bits indicates a possible error in the data bits received.

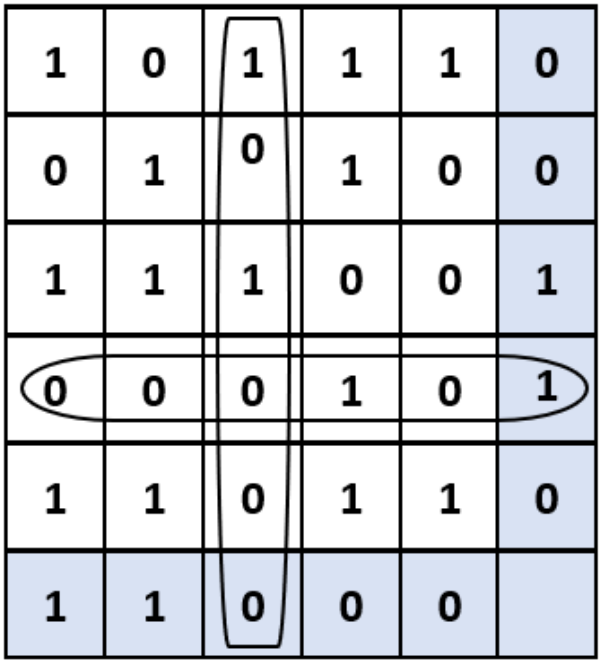

Fig. 3. Horizontal and Vertical parity [4]

For error correction, after comparing calculated parity bits with received parity bits, any differences in the horizontal, vertical, and two diagonals could be easily detected as the corresponding bit is set to one. It produces four different arrays, and all ordered pairs possible are made. A maximum of one candidate bit may be found using every ordered pair. In a few cases, no candidate bit is found among these collected bits.

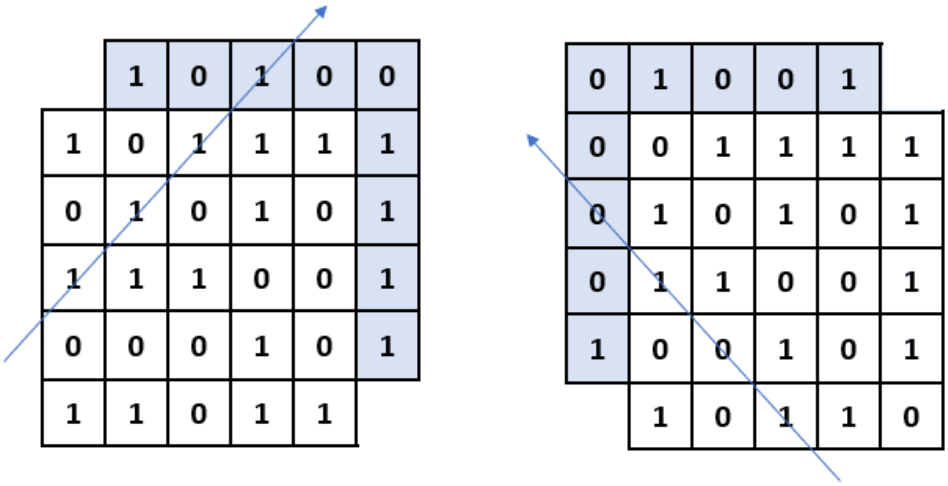

Fig. 4. Slash and Backslash diagonal parity [4]

This method proves to be advantageous over other error-correcting schemes. First, all multiple faulty bits can be detected, and up to 3-bit errors can be corrected. Second, other error-detecting and correcting schemes rely heavily on complex calculations, but HVD code requires very little calculation, which can be computed parallelly. Though this method provides very high error detection, it has minimal error correction capability. This method cannot be used as a standalone error-detecting and correcting scheme in any application. Still, it can be combined with code schemes that provide limited error detection.

2.3 Implementation of Multi-directional Parity Check code Using Hamming Code for Error Detection and Correction

V. Badole et al. [2] improved the previous method by including Hamming code by doing parity checks along the horizontal, vertical, and diagonal [2]. Furthermore, the authenticity of the parity check bits is verified using a Hamming Code System. A hamming code decoder is used at the receiving end to withdraw error-free parity check bits. This is followed by a multi-directional parity code decoder, where the verified parity check bits are used to identify the erroneous bits in the message bit matrix. This ensures that the bits addressed by the parity in the main message data matrix are authentically corrected. It reduces the false modification of non-erroneous messages.

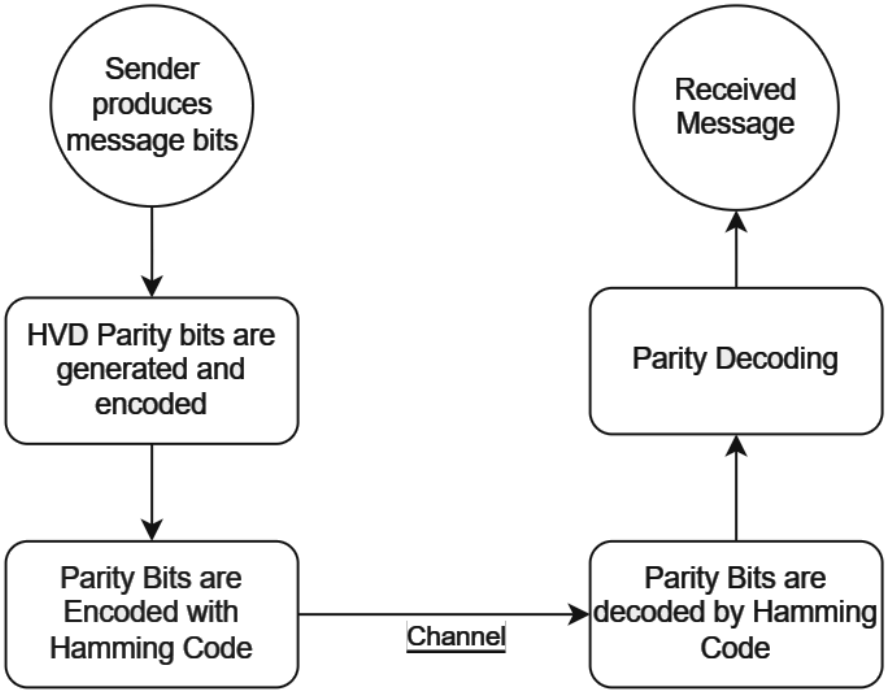

Fig. 5. Implementation of multi-directional parity code and hamming code on parity bits [2]

The code rate and the overhead are almost similar to the HVD parity check technique [4]. It had an extra $log(n)$ terms of parity bits that can be ignored for a significantly larger value of n. A disadvantage of this method is that the complexity of implementation is slightly increased due to the increased stages in the process, as displayed in the flowchart above. Moreover, the technique can increase dimension and reduce the moderate. Third, the delay overhead in the case of HVD code [4] is much less compared to other error-detecting and correcting schemes. The above two methods are currently used only for two-dimensional data sets. If these methods are applied in a three-dimensional data set considering each horizontal plane, it produces a considerable overhead.

3 Design and Implementation of Diagonal Parity Check Code

This section demonstrates the setup of our message bits and the methodology of taking parity check bits. Further, this paper demonstrates using Hamming Code on the model to perform error correction on the generated parity bits.

3.1 Encoding

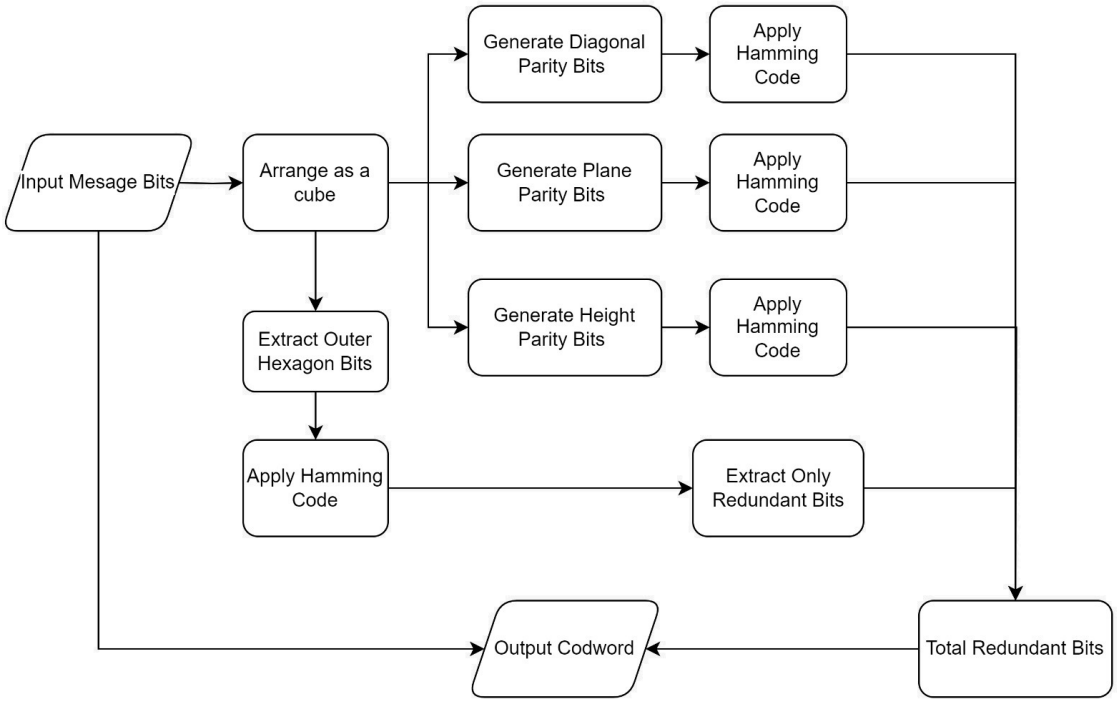

Fig. 6. Encoding of data-word in Diagonal Parity Check Code

Regarding the literature review, this paper proposes a solution, which is an extension to the dimension of the HVD parity implementation, along with novelty in generating the parity bits. The method of increasing dimensions from two to three and subsequently generating the parity bits is discussed further. In the proposed solution, the message bits are organized into a three-dimensional cube structure, each dimension having a size of $n \times n \times n$. Arranging bits into a cube helps exploit the beauty of its symmetry. Now, a singular plane of bits, representing our parity plane, is placed over the message cube. Each element in the parity plane represents the modulo-2 sum of the column of bits exactly below the specified element concerning the entirety of the cube. Let these parity bits belong to the set $H = (h_1, h_2, h_3, ..., h_{n^2})$.

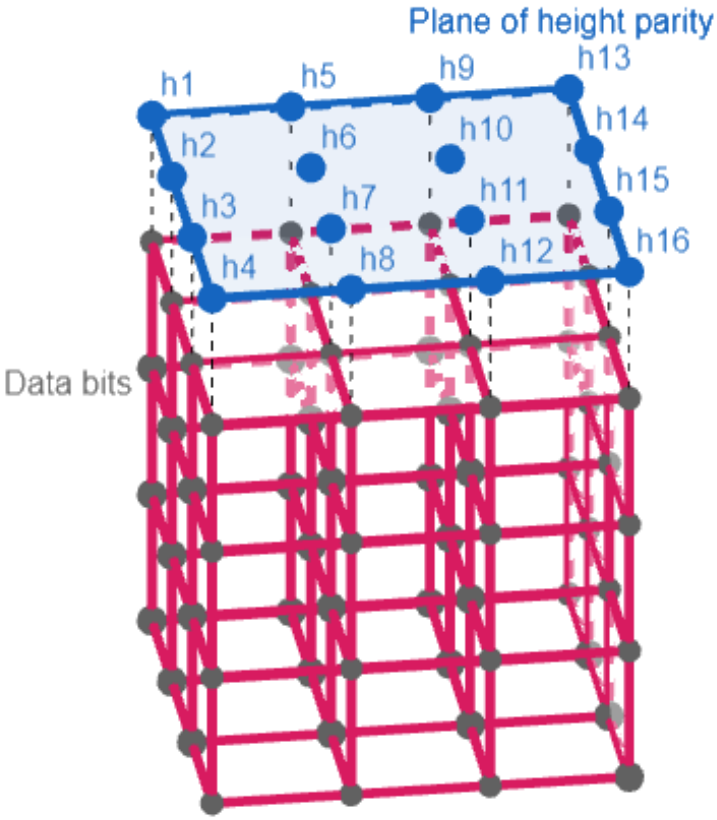

Fig. 7. Parity plane above the data bits

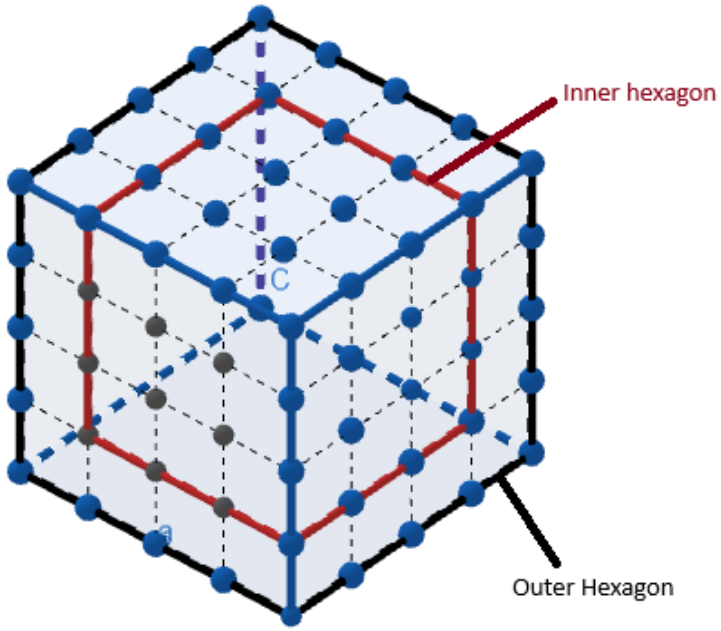

Fig. 8. Inner and outer hexagon bits from the isometric view of the cube

Furthermore, a non-conventional approach is taken to obtain the diagonal parity bits. Consider the isometric view of the message bit, as shown above. The cube has two views: the outer hexagon and the inner hexagon. The outer hexagon trivially consists of only one bit. It does not superposition any bits beneath it. On the contrary, in the inner hexagon, every data bit overlaps one or more data bits beneath it. Without loss of generality, consider one bit in the internal hexagon as X. Let $(s_1, s_2, \ldots, s_k)$ be the set of bits overlapped by the bit X. A new parity bit P can be generated, which represents the modulo two sums of X and all of its overlapping data bits, in its corresponding set S. Similarly, consider a new set of parity bits $D = (d_1, d_2, d_3, \ldots, d_t)$, where $t = 3n^2 - 9n + 7$.

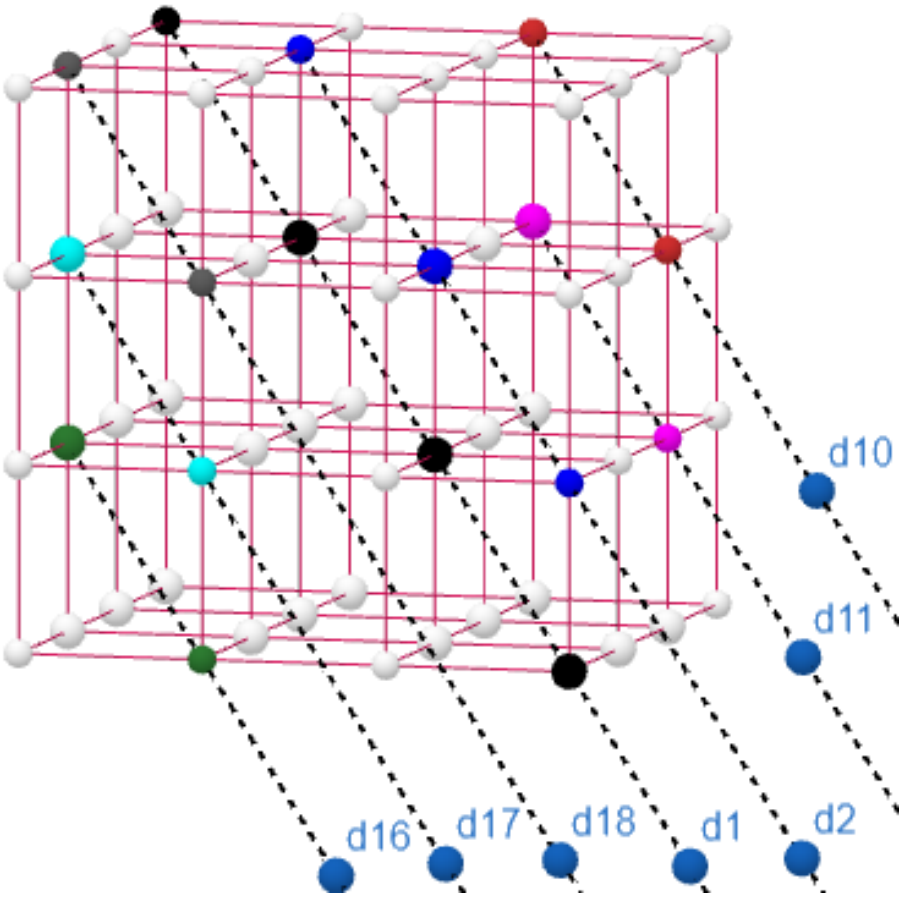

Fig. 9. Diagonal Parity bits

Fig. 9 provides a skeletal view of the cube. Each parity bit belonging to the set D is generated by calculating the modulo-2 sum of all bits belonging to one color group. It is to be noted that the figure only shows a few selective diagonal parities for ease of visualization. Note that the data bits comprising the perimeter of the outer hexagon don't have any overlapping bits beneath it. Consider the set of all of these bits as $B = (b_1, b_2, b_3, \ldots, b_t)$ where, $t = 6n$. They are not sent separately over the channel to reduce redundancy. The required bits in the set B can easily be achieved from the main message cube. Additionally, consider new parity bits, which are the total modulo of two sums of the bits in every horizontal plane. consider a new set of parity bits $P = (p_1, p_2, p_3, \ldots, p_n)$. Hence, the set of all the parity check bits cumulatively is given by the set $H \cup P \cup D$.

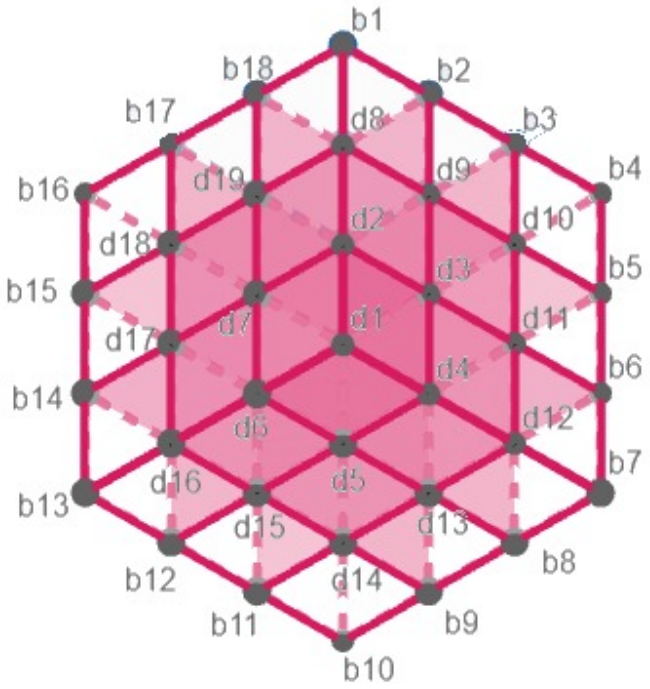

Fig. 10. Parity bits D and data bits B

Fig. 10 is the isometric view of the cube from the diagonal vertex opposite to the first message bit in the topmost plane of the message block [5]. Consider the three sets of parity bits D, H, and P. To secure these parity bits, apply the hamming code for all three sets, say D', H', and P'. These bits are helpful for the detection and correction of errors in these parity bits. These additional bits contribute an extra log factor to the redundancy of our coding scheme(approx). Additionally, apply hamming code to Set B to produce B'. However, as mentioned earlier, only its redundant bits are sent over the channel. Moreover, this ensures one-bit correction and two-bit error detection in each set. This gives a cumulative of four bits error correction capabilities in all parity sets and B from each of the three sets.

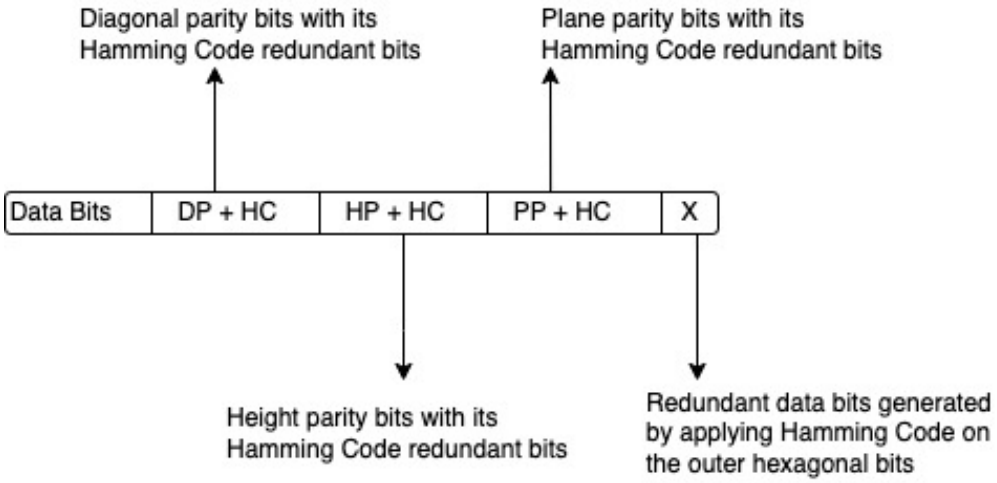

Fig. 11. Generated codeword

3.2 Error Detection and Correction

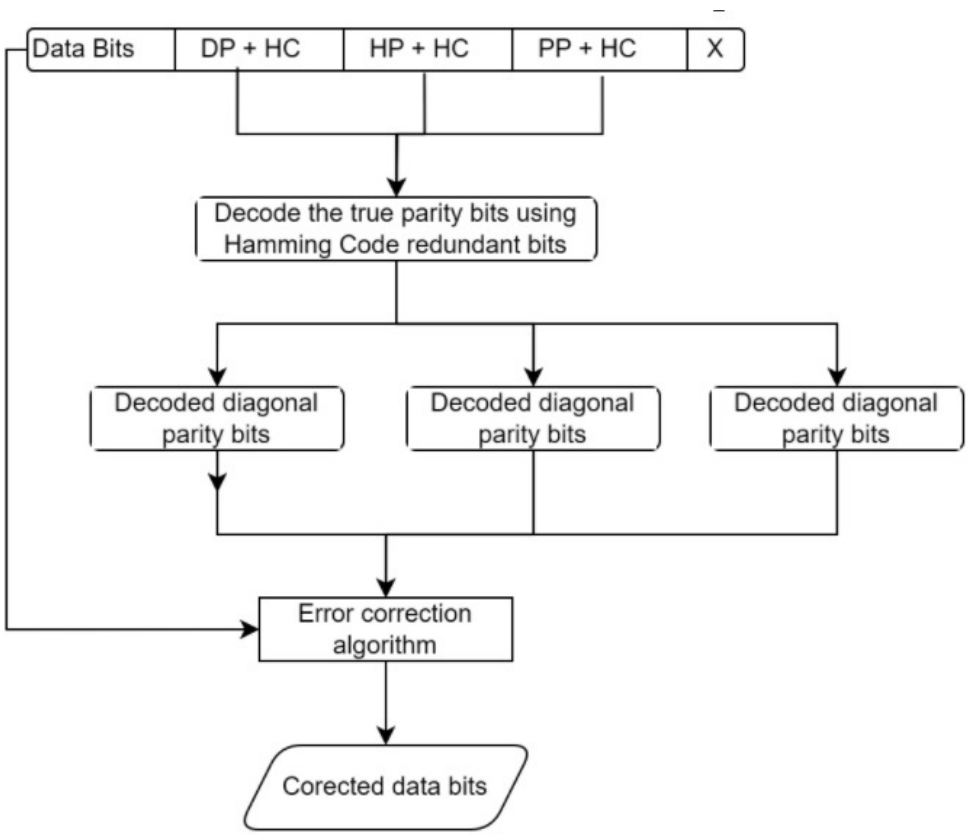

Fig. 12. The decoding algorithm

This section discusses the crux of the paper, error detection and correction. The entire message bits cube and the parity bits generated at the source are available on the receiving end. Furthermore, the parity bits are accompanied by redundant bits due to the Hamming code applied to them. Once the array of parity bits and their Hamming code redundancies are received, they are initially decoded to find the original parity bits' values following the Hamming Code decoding algorithm. The beauty of hamming code is that the syndrome bits it generates directly indicate the relative position of the error. Hence, the erroneous bits can be toggled. However, the parity check codes are limited to errors of up to only two bits. Thus, the parity bits are obtained by applying hamming code to each set of parity bits. This paper will continue our further discussion with the expectation that our parity bits are as sent initially after decoding. Now, with the available parity bits, correction of data bits of the message block is possible in the following way. Correcting an error of one bit is trivial. Since 1 is an odd number, the bit will flip the parity bit corresponding to its height. The

bit will also flip its planar parity check bit. Since all bits in a plane are associated with different height parity check bits, the position of the erroneous bit can be narrowed down strictly.

Up to 6 bits of errors could be corrected, four parity bits and two data bits. Now, on expanding the scope to 2-bit error correction, consider the claim that there is enough information to correct up to 2-bit errors in the data bits. Progressing with our proof algorithmically. Consider two points, X and Y, with error. Consider the following possibilities in the position of X and Y. The position is discussed concerning its involvement with the type of parity bits.

- **Case 1:** X and Y are not simultaneously involved in generating parity bits. In this case, sufficient parities are available to narrow the exact position of the bit in the block due to their difference in plane parity and height parity.
- **Case 2:** X and Y are on the same diagonal, generating the same parity bit in set D. This ensures the two are in different planes and heights. Hence, its location is detectable and correctable for the reason mentioned in Case 1.
- **Case 3:** X and Y are on the same vertical line. They are involved in generating the same parity bit in set H. Both of these points are in two different planes. If these two bits do not toggle any of the diagonal parity bits, it implies that they are in the outer hexagon. It is to be noted that there are precisely 4 bits (2 pairs) that simultaneously pass through both planes and are part of the outer hexagon. Both bits in each pair lie on the same height, and the two pairs lie on opposite edges of the cube. The two error bits are part of these 4 bits. Since the location of the four bits is precisely known, all possibilities of occurrence of the 4 bits can be generated, and its validity against the syndrome using the redundant hamming code bits (which was sent additionally along with the message as a part of the redundant bits). This is possible due to the location of the 4 bits that have the possibility of having an error.

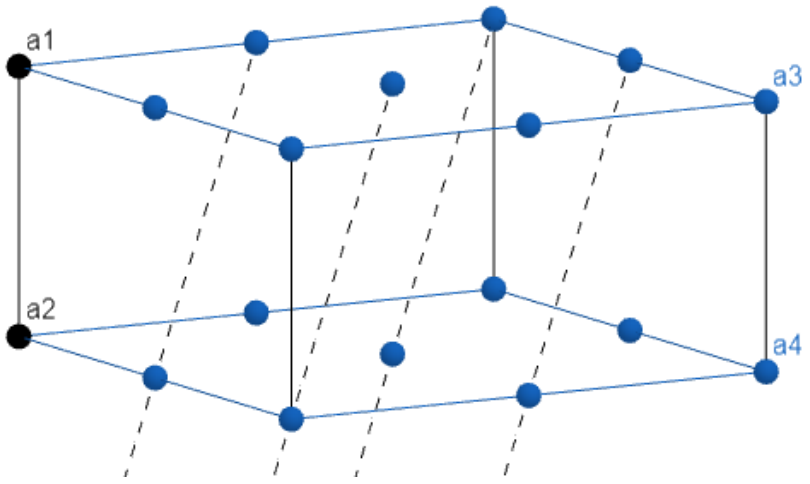

Fig. 13. A possibility of the occurrence of case 3 when the erroneous bits are on the outer hexagon

The Fig.13 shows a specific occurrence of case 3. In the case where diagonal parity bits indicate no error, and plane 2 and plane 3 indicate the presence of an error; this narrows down the possibility of the error in the bits a1,a2,a3, and a4 only. Brute forces all possibilities of the four bits until the redundant hamming code bits sent for the outer hexagon detect no errors. In Fig.13, a1 and a2 are represented as the erroneous bits. Otherwise, if the diagonal parity check bits are toggled, then the erroneous bits are not in the outer hexagon. These bits will have different diagonal and plane parity check bits, giving sufficient information to correct the error.

– **Case 4:** X and Y are on the same plane. There will be two different diagonal parity bits passing through X and Y. This is because each diagonal considered penetrates through two other planes as shown in Fig.9. Additionally, the two bits will have different height parity bits. This, again, gives sufficient information on detecting and correcting the bits.

3.3 Mathematical Analysis of Overhead and Code rate

This section discusses the number of data bits and their comparison with redundant bits. Let the data word's height, width, and base be n each. Thus, there are n^3 data bits, n^2 height parity bits $n(H)$, n plane parity bits $n(P)$, $3n^2 - 9n + 7$ Diagonal parity bits $n(D)$. Each parity set also adds an extra log factor in the calculation of redundant bits.

$$\begin{aligned}\text{Total redundant bits} &= n(H) + n(D) + n(P) + log(n(H)) + log(n(D)) + log(n(P)) \\ &= 3n^2 - 9n + 7 + n^2 + n + log(6n - 6) + log(3n^2 - 6n + 1) + log(n^2) + log(n) \\ &= 4n^2 - 8n + 10 + 4log(n) + log(3n^2 - 9n + 7)\end{aligned} \tag{1}$$

The overhead and code rate of this scheme is given by,

$$\text{Overhead} = \frac{\text{Number of parity bits}}{\text{Number of data bits}} = \frac{4n^2 - 8n + 10 + 4log(n) + log(3n^2 - 9n + 7)}{n^3} \tag{2}$$

$$\text{Code rate} = \frac{\text{Information bits}}{\text{Total bits}} = \frac{n^3}{n^3 + 4n^2 - 8n + 10 + 4log(n) + log(3n^2 - 9n + 7)} \tag{3}$$

For higher values of n, the log factors can be ignored; however, they significantly come into the picture for smaller n. For $n = 10$, There is an overhead of 35.1%, and the code rate comes out to be 74.02%. For other values of n, a comparison has been made in Fig. 14.

4 Result and Analysis

This section performs a comparative analysis between different error detection and correction algorithms based on parameters such as number of redundancy bits, codeword length, overhead, and code rate. Consider the analysis of error correction parameters of the algorithms presented in the literature review.

4.1 Three-dimensional Parity-Check Codes for Correction and Detection of Multiple Errors

Let the data word's height, width, and base be h, l, and b, respectively. Thus, the data bits $l \times b \times h$ and $l \times b$ (x-axis), $b \times h$ (y-axis), $l \times h$ (z-axis) and additional $l + b + h + 1$ on the edge and corner parity bits.

$$\text{Total parity bits} = lb + bh + hl + b + h + l + 1 \tag{4}$$

The overhead and code rate of the three-dimensional parity check scheme are given by

$$\text{Overhead} = \frac{\text{Number of parity bits}}{\text{Number of data bits}} = \frac{lb + bh + hl + l + b + h + 1}{lbh} = \frac{3l^2 + 3l + 1}{l^3} \ (l = b = h) \tag{5}$$

$$\text{Code rate} = \frac{\text{Information bits}}{\text{Total bits}} = \frac{lbh}{(l+1)(b+1)(h+1)} = \frac{l^3}{(l+1)^3} \ (l = b = h) \tag{6}$$

4.2 Horizontal-vertical-diagonal error detecting and correcting code

Let the height and width of the data word be n each. Thus, producing n^2 data bits and n (horizontal), n (vertical) and $2n - 1$ diagonal parity bits.

$$\text{Total parity bits} = n + n + 2n - 1 = 4n - 1 \tag{7}$$

The overhead and code rate of the HVD parity check scheme are given by

$$\text{Overhead} = \frac{\text{Number of parity bits}}{\text{Number of data bits}} = \frac{4n - 1}{n^2} \tag{8}$$

$$\text{Code rate} = \frac{\text{Information bits}}{\text{Total bits}} = \frac{n^2}{n^2 + 4n - 1} \tag{9}$$

Table 1. Comparison of code rate and overhead between various methods

Comparison between various methods					
Error Correcting Method	Number of data bits	Number of redundant bits	Number of Codeword bits	Overhead (%)	Code rate (%)
HVD	1000	390	1390	39	71.9
Three-dimensional	1000	331	1331	33.1	75
HVD+Hamming	1000	450	1450	45	68.9
DPC Code	1000	351	1351	35.1	74

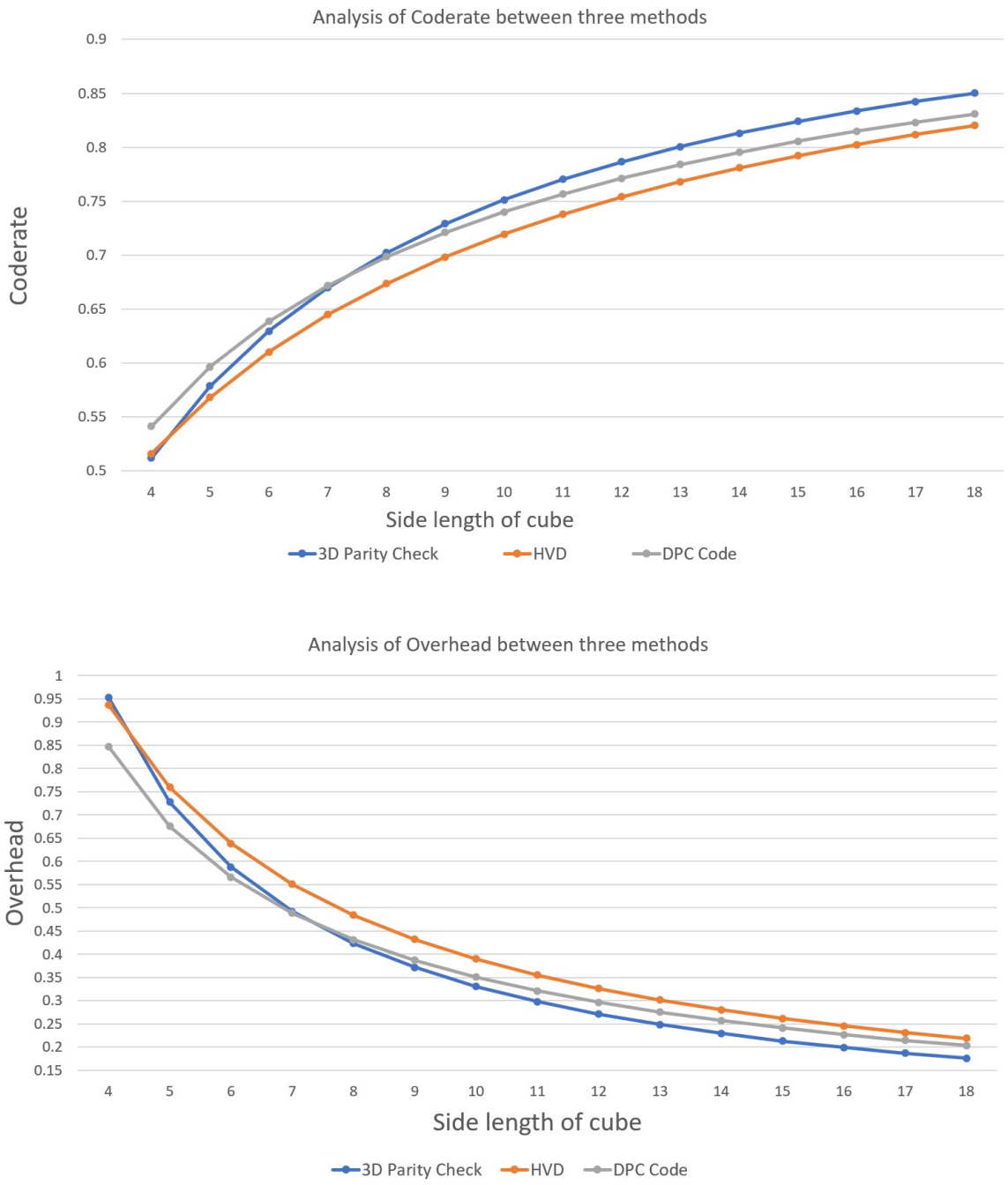

Fig. 14. Analysis of Coderate and Overhead between the three methods

The graph shows that our proposed model outperforms the HVD implementation in terms of code rate and bit overhead. This could be accounted for by the excessive diagonal parity bits generated for every plane in the HVD implementation. Moreover, the simple 3D parity check has a better code rate and overhead. Still, the algorithm has flaws as it fails in multiple error detection, as discussed in the literature review. DPC provides high detection and correction of errors generated during transmission. It can detect burst errors almost 100%. It is clear that the proposed method can correct up to 6 bits of error, 4-bit errors in the data bits, and 2-bit errors in the redundant bits and can detect almost 100% of errors. HVD [4] method can correct up to 3 bits. Three-dimensional parity-check codes [1] can detect up to 5 errors and correct up to 2 errors. Multidimensional parity check codes with hamming code [2] can correct up to 4 bits of error, a 3-bit error in the data part, and a 1-bit error in the redundant bits.

5 Conclusion and Future Work

A Diagonal Parity Check can undoubtedly correct up to 2 data bits regarding the correction of bits. It can also correct 3-bit errors in specific scenarios depending on the relative position of the three bits, but only for some positions. A scenario was found in the case of error detection where the error detection of a particular pattern is not possible. Experimentally, it was found that the possibility of this happening is very low (1 out of 7.5 million). The Diagonal Parity Check Code for Three Dimensional is a novel approach to taking parity checks and applying the Hamming Code. This error detection and correction model corrects 6-bit errors, 4 of which come from the application of Hamming Code, and the remaining 2 from the message bits, using the newly corrected parity check bits. This model has an overhead of 35.1% and a code rate of 74%.

References

1. N. B. Anne, U. Thirunavukkarasu, S. Latifi, Three and four-dimensional parity-check codes for corrections and detection of multiple errors, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1286763, [Accessed: 19-01-2024] (2004).
2. V. Badole, A. Udwat, Implementation of multidirectional parity check code using hamming code for error dectection and correction, https://ijrat.org/downloads/Vol-2/may-2014/paper%20ID-2520141232.pdf, [Accessed: 17-01-2024] (2014).
3. S. G, R. N, VLSI design of Parity check Code with Hamming Code for Error Detection and Correction, https://ieeexplore.ieee.org/document/9065790, [Accessed: 19-01-2024] (2019).
4. M. Kisahni, H. R. Zarandi, H. Pedram, A. Tajary, M. Raji, B. Ghavami, HVD: horizontal-vertical-diagonal error detecting and correcting code to protect against with soft errors, https://link.springer.com/article/10.1007/s10617-011-9078-2, [Accessed: 19-01-2024] (2011).
5. Arrangement of data bits and axis D and H, https://www.geogebra.org/m/tjtufxd9.