A New Error Correction Coding Approach

Muhammad Sheikh Sadi
Department of Computer Science & Engineering,
Khulna University of Engineering & Technology, Khulna, Bangladesh
Email: sheikhsadi@gmail.com

Palash Kanti Bachar, Palash Ghosh and Muhammed Saifur Rahman
Department of Computer Science & Engineering,
Khulna University of Engineering & Technology, Khulna, Bangladesh
Email: {palash.kuet, palash0707055, saifur.nishat}@gmail.com

***Abstract*—The wide variety of applications of embedded systems has resulted in these systems permeating human life as well as where their proper functioning is crucial. The computing tasks could be faulty due to various causes. Faulty tasks may compromise the safety and performance of the system and even cause disastrous consequences. So the error detection and correction is becoming important day by day. This paper proposes a new error detection and correction method. Experimental studies show that the proposed method outperforms existing dominant approaches.**

***Index Terms*—Error Correction Coding, Generator Matrix, Parity Matrix, Fault Tolerance.**

I. Introduction

The high level of complexity and the fact that the software and hardware are so intricately linked, means that the system may be very sensitive to errors [1], [2], [3]. Specifically errors are a matter of great concern when designing high availability systems or systems used in electronic-hostile environment. Space programs, where a system cannot afford a malfunction while in flight, are vulnerable to soft errors. Nuclear power monitoring systems, where a single failure may cause severe destruction and real-time systems, where a missed deadline can constitute an erroneous action and a possible system failure, are a few other examples where soft errors are a critical issue [4], [5]. A momentary failure in banking transactions due to soft errors may cause a huge difference in balances. If an error causes 1→0 bit flips in the most significant bit of the register storing the amount of money deposited in a bank account, then the effect might be an unexpected change in balance [6]. A corrupted intermediate value can corrupt all subsequent computations and the executions of the whole program. Real-time and embedded systems are now a central part of our lives. Reliable functioning of real-time systems is of paramount concern to the millions of users that depend on these systems every day [7]. Hence fault tolerance is essential for real-time systems [8]. To design a fault tolerant system, error detection and correction is very important. Software error occurs when information is transmitted from one node to another node. Detection and correction of error at this particular moment is a must. Existing methods to detect and correct error incurs high information redundancy and/or high information redundancy. For this reason an efficient approach is needed to detect and correct error.

This paper proposes a new coding scheme which can detect and correct error in an efficient way. The proposed method uses comparatively lower number of redundant bits to detect and correct error. For example, to process seven bits information Hamming code takes four redundant bits whereas the proposed method can process these using three redundant bits. The proposed method takes comparatively lesser time to correct error than the other method. The paper is organized as follows. Related works is explained in Section II. The proposed methodology is presented in Section III. Experimental results are shown in Section IV. Conclusions are drawn in Section V.

II. Related Work

Several methodologies have already been proposed by many researchers that deal with analysis and optimization about the error detection and correction. Low density parity-check (LDPC) codes were first presented by Gallager [9]. A regular Gallager code has a parity check matrix with uniform column weight j and uniform row weight k [10]. In the past few years, several low-density parity-check (LDPC) codes were designed with performances very close to Shannon limit [11].

Sankaranarayanan and Vasic compared parity matrix with several error detection and correction methods [2]. They have described about parity-check orthogonalization approach. The parity of the word is checked after reading the word from memory. The word is accepted if the parity bits read out are correct. If the parity bits are incorrect, an error is detected but it cannot be corrected. An error-correcting code uses multiple parity check bits that are stored with the data word in memory. Each check bit is a parity bit for a group of bits in the data word. When the word is read back from memory, the parity of each group, including the check bit is evaluated. If the parity is correct for all groups, it signifies that no detectable error has

doi:10.4304/jait.4.3.142-147

occurred. Two dimensional parity can also detect (but not correct!) any combination of two errors in a packet.

Ibraheem described Hamming code and cyclic code [3]. The theory is generalized by mathematician Hocquenghem and gives rise to the family of codes BCH. The Hamming codes the binary ones seem codes BCH immediately. Hamming codes can detect and correct single-bit errors. In other words, the Hamming distance between the transmitted and received code-words must be zero or one for reliable communication. Alternatively, it can detect (but not correct) up to two simultaneous bit errors. He combined hamming code and cyclic code to detect and correct error efficiently.

Cyclic Redundancy Check (CRC) codes are a special subset of linear block codes that are very popular in digital communications. CRC codes have the cyclic shift property; when any code word is rotated left or right by any number of bit digits, the resulting string is still a word in the code space [2]. A parallel CRC circuit simultaneously processes multiple data bits [12]. Theoretical aspects of encoding cyclic redundant codes (CRCs) are reviewed [13]. Sprachmann has described about the CRC circuit generation to detect and correct error [3]. This property makes encoding and decoding very easy and efficient to implement by using simple shift registers. More complex error detection (and correction) methods make use of the properties of finite fields and polynomials over such fields [14]. The cyclic redundancy check considers a block of data as the coefficients to a polynomial and then divides by a fixed, predetermined polynomial. The coefficients of the result of the division are taken as the redundant data bits, the CRC. On reception, one can recompute the CRC from the payload bits and compare this with the CRC that was received. A mismatch indicates that an error has occurred. The main drawback of using CRC codes is that it has only error detecting capabilities. They cannot correct any errors in the data once detected at the destination and the data must be transmitted again to receive the message. For this reason, CRC codes are usually used in conjunction with another code that provides error correction. If CRC codes are the only ones used for an application, the raw BER of the channel is usually extremely low and data is not time-critical. They are good for magnetic and optical storage, where a simple retransmit request to correct bit errors is feasible. Hudak et al., compared several fault-tolerant software techniques including Algorithm Based Fault Tolerance based on error rate [15]. The techniques were ranked based on reliability and cost. Here he described different types of error detection and correction scheme.

Beaudry defined computation capacity as a performance metric in order to study the interaction between performance and reliability [16]. The author incorporated computation capacity in reliability modelling of redundant systems. The author defined computation capacity as the amount of useful computation per unit time available on the system. Since the author was evaluating gracefully degrading systems, the author defined computation capacity as a function of the system state. To increase the capacity of computing, system needs error detection and correction schemes as first as possible. The proposed method is fast and has automatic error detection and correction property.

III. The Proposed Methodology To Error Detection And Correction

The proposed method can detect and correct error using matrices multiplication. When a file is loaded it takes the binary value of this input file and from this binary value it generate input matrix. In Fig1 sender generate input matrix and using matrix multiplication generate codeword. Sender sends the codeword to the receiver. Receiver receives the codeword check the result. Then receiver uses matrix multiplication between receiving codeword and parity matrix. After multiplication receiver will check the result. If result is zero then there is no error. If the result of multiplication is not zero then the receiver will return a value that is the position where error has occurred. Going the exact position, invert the bit, if 0 is present then replace it by1. If 1 is present then replace it by 0.

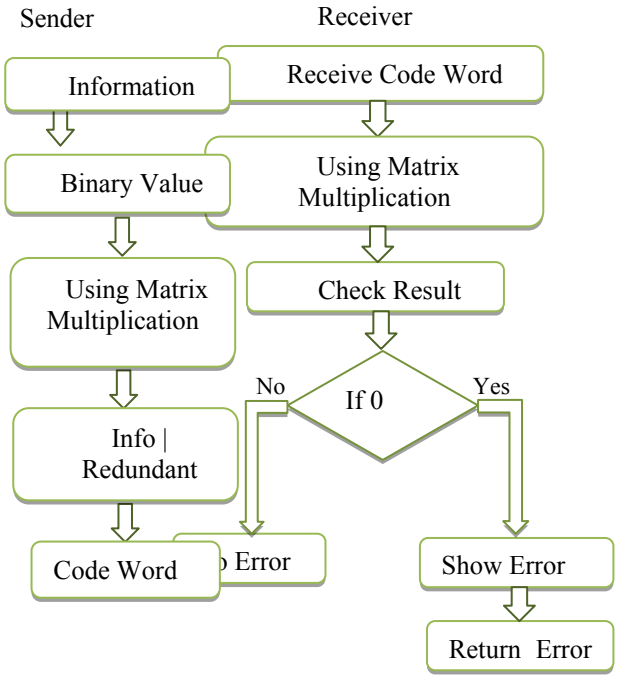

Fig. 1: Block Diagram for Sender and Receiver.

A. Matrix Representation

Matrix representation is most important feature for proposed method. Parity check matrix is generated from the Generator matrix. Matrix representation is done in two steps using two different matrices.

1) Generating Generator Matrix

Suppose the input data is a file. Then the file is converted into binary data. A matrix is formed by using this binary data, which is called input matrix. Let $X_{m1}, X_{m2}, X_{m3} \ldots, X_{mk}$ denoted the k information bits and the information matrix is $X_m = [X_{m1}, X_{m2}, X_{m3} \ldots, X_{mk}]$.

Generator matrix is formed using two matrices one is identity matrix and another is parity matrix. Here the Generator matrix form is given below:

$$G = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{11} & p_{12} & \cdots & p_{1(n-k)} \\ 0 & 1 & \cdots & 0 & p_{21} & p_{22} & \cdots & p_{2(n-k)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{k1} & p_{k2} & \cdots & p_{k(n-k)} \end{bmatrix} \quad (1)$$

The generator matrix has the form $[I_k, P]$ where I_k is the identity matrix (note: an identity matrix is a matrix which has a diagonal made up of 1s and has 0s everywhere else) and P is the $k \times (n-k)$ parity check matrix. Identity matrix depends on the input matrix. If there are seven bits in input matrix then the generator matrix will be 7×10 matrix.

Since G is a $k \times n$ matrix with all k rows being linearly independent where the rows of G form a set of basis vectors in a k-dimensional subspace of the n-dimensional space defined by considering all valid and invalid code words.

Code word is generated by multiplying the input matrix with the generator matrix. Which have input value and according the input value generates redundant bits. Redundant bits are important to detect and correct the information. The proposed method required n number of redundant bits to process 2^n number of information bits. Then linear equations that generate each code word may be written in matrix form as follows in equation (2) where information bits and redundant bits are present. After processing this code word, the sender must send the code word to the receiver.

$$C_{mj} = (X_{m1} G_{1j} \oplus X_{m2} G_{2j} \oplus \dots\dots\dots \oplus X_{mk} G_{mj}) \quad (2)$$

for $j = 1,2,3 \dots\dots\dots\dots, n$.

The code word is represented by a $1 \times n$ matrix as shown below.

$$C_m = [c_{m1}, c_{m2}, \dots\dots\dots, c_{mn}]$$

2) Generating Parity Matrix from Generator Matrix

A parity matrix is essential to check the error detection and to find out the error position. The first step of generating a parity matrix from the generator matrix is in equation (3). Where firstly form the transpose of the parity matrix which contain generator matrix. Then the other bits will be fulfilled with the identity matrix. The parity matrix is the combination of P_k and I_k.

$$H \begin{bmatrix} P_{11} & P_{21} & \cdots & \cdots & P_{k1} & 1 & 0 & 0 & \cdots \\ P_{12} & P_{22} & \cdots & \cdots & P_{k2} & 0 & 1 & 0 & \cdots \\ P_{13} & P_{23} & \cdots & \cdots & P_{k3} & 0 & 0 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ P_{1(n-k)} & P_{2(n-k)} & \cdots & \cdots & P_{k(n-k)} & 0 & 0 & 0 & \cdots \end{bmatrix} \quad (3)$$

Transpose of the matrix (3) has done to get the final parity matrix (4) which is needed to detect and correct error. To check error in information bits the parity check matrix is needed in receiver end. The receiver will generate this matrix according to the input matrix.

$$H' = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \cdots & P_{1(n-k)} \\ P_{21} & P_{22} & P_{23} & \cdots & P_{2(n-k)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ P_{k1} & P_{k2} & P_{k3} & \cdots & P_{k(n-k)} \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \quad (4)$$

Therefore must exist another matrix, H', consisting of $n-k$ linearly independent rows that forms a set of basis vectors in the $n-k$ subspace which is unreachable from G. It follows that each valid codeword C_m will be orthogonal to the basis vector set defined by H', i.e

$$C_m H' = 0 \quad (5)$$

The matrix multiplication result of the $C_m H'$ will be zero if there is no error in codeword. But if there is present error in information bit then the multiplication return the result which indicates the position of the erroneous information.

So by using this methodology, in real time and embedded system, it will automatically detect and correct the error.

IV. Experimental Analysis

The methodology is useful for soft and hard error. A simulator is designed by using this methodology and discussed about the experimental result. It has been clarified that the method can be useful for any type of computing system. In safety critical system any error may result in loss of life. So to avoid this situation, automatic error detection and correction method is used. The proposed approach is tested taking different types information code word.

Several tools and library is used for this purpose. List of these tools and library are given below-

1. Microsoft Visual Studio 2010 is used as IDE for its extensive support for debugging and user interface design.
2. .NET Framework 4.0 as software framework is used because of its large library.
3. C# 4.0 is used for implementation process.

The simulator has some special features such as it takes an input file and converts it into an input matrix. Then multiplying the input matrix with the generator matrix, it generates a matrix which is called code word. Code word contains input data and redundant bit. The simulator can upload a file and generate a code word according to the input file. It also calculates the time which is needed to generate the code word from the input file. When the data is transmitted from one node to another node, error may occur randomly on the information bits. The receiver will receive the information that contains with erroneous bit. For example, if the sender wants to send "A" to the receiver than the sender will form an input matrix which is the binary

value of input "A". The input matrix is "1000001" and multiplying input matrix, the generated code word will be "1000001110" where last three bits are redundant bits. Finally the sender will send this code word to the receiver. During transmission of the code word, randomly an error may occur and the receiver receives the code word which contains the value "1010001110"

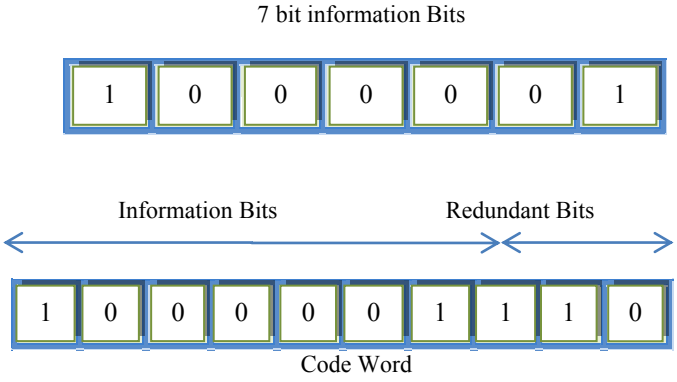

Fig. 2: Pictorial View of Code word in Sender End

The receiver receives information data or code word and calculates the error. If there are error/errors present in code word then the matrix multiplication of code word and parity matrix will return nonzero value. Finally, it returns the error position of each data. For example the receiver receives data "1010001110" which is erroneous data. After multiplying the receiver will return the bit position 3. So it implies that in bit position three an error has occurred.

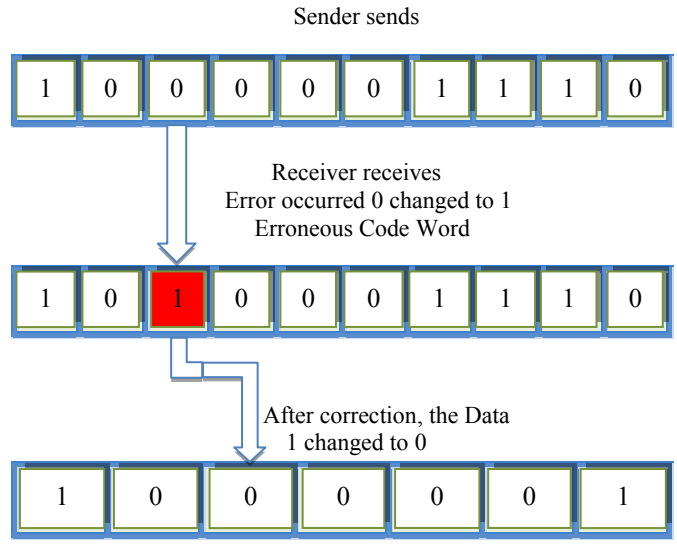

Fig. 3: Pictorial View of Error Correction in Receiver End

It selects the error position and goes to the position and invert the value which is present in that the particular position. Finally it gets the actual data. This simulator can detect and correct 1 bit error within a code word.

Comparisons between existing and proposed method is described based on the experimental result. Some advantages and disadvantages of both methods will be also discussed. There are many methods existing for error detection and error correction. But here we have compared between hamming code and proposed method. There are many methods present for error detection and error correction. But one of the methods is selected to compare with the proposed method. That method is hamming code, which can detect two bits error and correct 1 bit data error. Using hamming code, it can detect two bits and correct one bit data within 7 bit code word. Redundant bit in most important to developed a coding scheme, hamming code takes $log_2(n)+1$ bits to error detection and correction, where proposed method needs $log_2 n$ redundant bits. For example to process 7 bit data hamming code needs 4 redundant bits and proposed method needs 3 redundant bits. To process large amount of data proposed method gives better result than the other method.

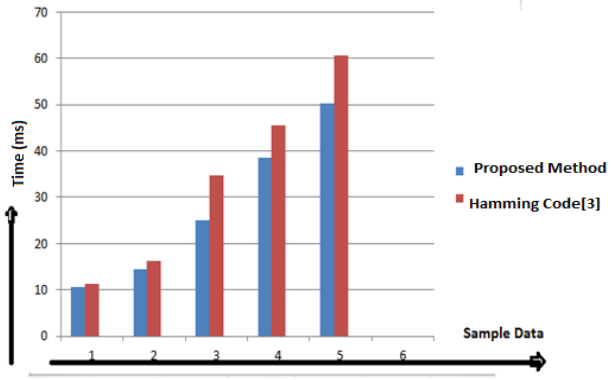

Fig. 4: Comparisons between Proposed Method and Hamming Method

Comparison between proposed method and hamming method with respect to the time and code word is given in figure 4. For the large amount of data, proposed method is better than the other method. For the fixed time proposed method process more information than the hamming method. So it can process more information taking lowest time.

Another comparison between Berger code and proposed method is shown in fig 5. With the increasing number of code word proposed method takes comparatively low time to detect error than the Berger code.

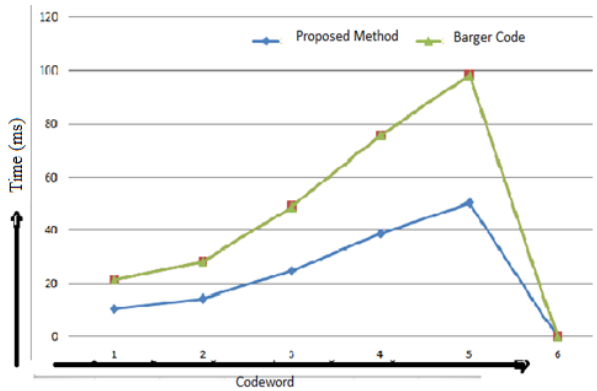

Fig. 5: Comparisons between Proposed Method and Berger Code to Detect Error

Comparison between experimental results always shows that proposed method gives better result than the other method.

V. Conclusions

In this paper, a new methodology of error detection and correction has been proposed for detecting and correcting error efficiently. For large number of data proposed method gives better performance than the small number of data comparatively with other methods. Now-a-days computer are using large number of data and transferring more data from one place to another place. So the need to check accuracy of this huge number of data is much higher than ever. To check and correct error proposed method takes less time comparatively than some methods such as Hamming Code, Berger Code. The proposed method uses less number of redundant bits comparatively with the other methods. The proposed method will help in error detection and correction of huge number of data during transmission as fast as possible.

The proposed method based on error detection and correction can detect two bits error within a code word. But correct only one bit error within one code word. It is important to implement this method in computing system to prevent fault.

References

[1] Muhammad Sheikh Sadi, D.G. Myers, and Cesar Ortega Sanchez, "Component Criticality Analysis to Minimizing Soft Errors Risk*," In International* Journal *of Computer Systems Science and Engineering,* CRL Publishing, vol. 25, No. 5, 2010.

[2] S. Sankaranarayanan and B. Vasic, "Iterative decoding of linear block codes: A parity-check orthogonalization approach," IEEE Transactions on Information Theory, vol. 51, no. 9, pp. 3347–3353, September 2005.

[3] V. Kumar and O. Milenkovic, "On graphical representations of algebraic codes suitable for iterative decoding," IEEE Communications Letters, vol. 9, no. 8, pp. 729–731, August 2005.

[4] Muhammad Sheikh Sadi, Mizanur Rahman Khan, Nazim Uddin, and Jan Jürjens, "An Efficient Approach towards Mitigating Soft Errors Risks," *Signal & Image Processing: An International Journal (SIPIJ),* vol. 2, No. 3, September 2011.

[5] Ne'am Hashem Ibraheem "Error-Detecting and Error-Correcting Using Hamming and Cyclic Codes," IEEE Transactions on Information Theory, vol. 51, no. 9, pp. 3347–3353, September 2005.

[6] R. W. HAMMING, "Error Detecting and Error Correcting Codes," Bell System Tech. Jour., 29 (1950): 147–160.

[7] Xue Liu, Hui Ding, Kihwal Lee, Lui Sha, Marco Caccamo "Feedback Fault Tolerance of Real-Time Embedded Systems – Issues and Possible Solutions"

[8] D. Rubenstein, J. Kurose, D. Towsley ``Real-Time Reliable Multicast Using Proactive Forward Error Correction,'' Proceedings of NOSSDAV *'98*, (Cambridge, UK, July 1998).

[9] 9. S. J. Johnson and S. R. Weller, "A family of irregular LDPC codes with low encoding complexity," IEEE Communications Letters, vol. 7, no. 2,pp. 79–81, February 2003.

[10] D. J. C. MacKay and M. C. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in Codes, Systems and Graphical Models, ser. IMA Volumes in Mathematics and its Applications, B. Marcus and J. Rosenthal, Eds. New York: Springer, 2000, vol.123, pp.113–130. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/CodesRegular.html

[11] B. Vasic and O. Milenkovic, "Combinatorial constructions of low-density parity-check codes for iterative decoding," IEEE Transactions on Information Theory, vol. 50, no. 6, pp. 1156–1176, June 2004.

[12] M. Sprachmann, "Automatic generation of parallel CRC circuits," IEEE Design & Test of Computers, vol. 18, no. 3, pp. 108–114, May-June 2001.

[13] G. Albertengo and R. Sisto, "Parallel CRC generation," IEEE Micro, vol. 10, no. 5, pp. 63–71, 1990.

[14] Wells, Richard B., "Applied Coding and Information Theory for Engineers." Upper Saddle River, NJ: Prentice-Hall, 1999.

[15] Ahmad A. Al-Yamani, Nahmsuk Oh and Edward J. McCluskey "Algorithm-based fault tolerance: a performance perspective based on error rate".

[16] Q. H. Spencer, "Short-block LDPC codes for a low-frequency power-line communications system," in IEEE International Symposium on Power Line Communications and Its Applications. IEEE, April 6-8 2005, pp. 95–99.

Muhammad Sheikh Sadi received B.Sc. Eng. in Electrical and Electronic Engineering from Khulna University of Engineering and Technology, Bangladesh in 2000, M.Sc. Eng. in Computer Science and Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 2004, and completed PhD (Area: Dependable Embedded Systems) from Curtin University of Technology, Australia in 2010. He is currently Professor at the Department of Computer Science and Engineering, Khulna University of Engineering and Technology, Bangladesh. He teaches and supervises undergraduate and postgraduate theses in topics related to Fault Tolerate Computing, Embedded Systems, Digital System Design, Soft Errors Tolerance etc. He has published 32 papers and book chapters in his area of expertise. Muhammad Sheikh Sadi is a member of the IEEE since 2004.

Palash Kanti Bachar received B.Sc. Eng. in Computer Science and Engineering from Khulna University of Engineering and Technology, Bangladesh in 2012. He is currently working as a Software Engineer in Hi-Tech Bangla. His research interests are Fault Tolerate Computing, Network Security, and Software Engineering.

Palash Ghosh received B.Sc. Eng. in Computer Science and Engineering from Khulna University of Engineering and Technology, Bangladesh in 2012. He is currently working as a Software Engineer in Compaq Solution. His research interests are Fault Tolerate Computing and Software Engineering.

Muhammed Saifur Rahman is currently a student of Khulna University of Engineering and Technology under Department of Computer Science and Engineering. His research interests are Fault Tolerant System Development, Network Security, Machine Learning and Artificial Intelligence.