An Efficient Error Correction Approach by using Successive Parity Generation

Blinded for Review

Abstract— **Soft errors hamper reliability of modern electronic circuits. In critical application areas, demand of safety against soft errors is increasing. Several methods have been developed to provide such error free environment in modern systems. The use of Golay code, and BCH codes are some of the top most widely used techniques. Nonetheless, still no methods have claimed that it has 100% correction rate. In this paper, a new Successive Parity Generation (SPG) based approach is proposed to tolerate such soft errors. After occurrence of bit errors in data, the proposed method corrects all the bit errors completely. It calculates parity at receiver end and successively detects errors and correct the erroneous data by comparing sender and receiver parity bits. The proposed method can detect and correct 100% error for any size of codeword. Experimental studies show that it outperforms existing methods with respect to error correction rate, time and information overhead.**

Keywords—Successive Parity Generation; Soft Error Tolerance; Parity Matching; Error Correction Code; Data Block.

I. Introduction

As technology advances, the scaling of complex systems based on circuitry is increasing exponentially. Thus the complexity of reliability of a system arises [1]. While manufacturing memory devices, defects can hamper productions. After the products such as memory devices, CMOS circuits, chips etc. are in the market, soft errors can occur because of these defects. The amount of these kind of systems are rapidly growing as application of such systems are highly demanded.

In safety critical environments, such as satellite control system or nuclear power plants, a single bit error can cause catastrophic events [2], [3]. A complex system may suffer through erroneous state where the participant data bits are changed into erroneous data bits [1]. It is a matter of great concern where total reliability of the system fails upon such erroneous data bits. Mainly error occur when the data are being transferred from one unit to another, from one system to another, or even while the data are stored in a memory. These memory components are heavily linked in an intricately designed complex system.

Semiconductor devices such as CMOS chips, are becoming more complex as large scale integrations are performed. As more and more transistors are put on a single congested chip the complexity of the system rises exponentially. Comprehensive write and sense circuits can perform error rate analysis on Spin-transfer torque magnetic random access memory (STT-MRAM) which is a form of CMOS circuitry [4]. By following Nanoscale Complementary Metal Oxide Semiconductor Technology, Memory cells can provide fault tolerance upto a certain level which is efficient for memory application errors caused by radiation [5]. Radiation induced soft errors such as alpha particle strikes increases as scaling factors are increased [6]. It becomes more error prone by manufacturing variations [7].

Soft errors in systems poses as a threat to reliability. Thus, selective actions are needed to make the system more error free and increase the error tolerance of the system. Research upto now has focused on both circuit level and logic level solutions [2]. Still, the system remains error prone to soft errors. Duplication of hardware and software modules is another approach for soft error tolerance. The performance of the system suffers with duplicated modules for lack of synchronization between the modules. But with increased performance overhead, the system also suffers from high area, time and power overheads [8], [9].

The designing process of basic circuit blocks also reduces the soft error rate. To ensure the security and reliability of such circuit blocks, a wide variety of techniques are followed [10]. These includes superior manufacturing process for integrated circuits (IC). To properly solve this issue, the current technology suggests using of redundant components and use of error correcting codes (ECC). Redundant module techniques such as Triple modular redundancy (TMR) [11] come into consideration when system overhead is not an issue. TMR triples the modules and propose a voting technique which ensured detection and correction of errors. But it also increases the system overhead > 200%. The simple modules are replicated thrice to provide the error tolerance in this system. The overhead can be hard to implement for various applications.

On the other hand, ECC can be used to handle soft errors, but it also reduces reliability as a side effect. Different types of error recovery codes are used in modern days computing. Complex coding techniques are avoided due to time and space complexity of resources. Among error detection and correction codes, Golay codes [12], BCH codes [13], rectangular parity codes, horizontal vertical diagonal (HVD) parity codes can be mentioned. However, most of the codes suffer from low error detection and correction rate. As a result, the need for further research to develop an efficient technique for error detection and correction with minimal complexity is increasing day by day.

Thus, a new approach is needed to correct soft errors in memory bits which will uphold the correction process without reducing the degree of reliability. In this paper, a new approach for error detection and correction method is proposed to protect against soft errors. This method uses successive parity coding scheme to detect and correct error. The proposed method provides 100% error detection and correction in a data block.

The paper is organized as follows, section II some related works in this field of study is discussed upon the matter of error correction rate and overall system reliability. In section III, our proposed methodology is discussed briefly. In section IV we analyzed the performance of our system over other potential coding techniques. Section V contains concluding remarks.

II. Related Work

Several approaches are made to increase error detection and correction rate. The most common approach is use of parity for error-detection in memories of computer systems [11]. Such error correction methods have certain advantages & disadvantages.

In digital communication, Golay code [12] is one of the most popular error correction technique. It's strongly connected to finite sporadic groups in traditional mathematics. It encodes 11 bits of data into a 23 bits codeword. It has the ability to correct upto 3 bit of errors.

Another technique to follow for cyclic error correction is BCH code [13]. The main advantage of BCH code is it can control the number of errors correctable over the finite field.it generates a 31 bits codeword from 15 bits of data which can correct errors upto 5 bits.

Matrix code [14] divides the n-bit data into a fixed formation of l_1 sub words with width l_2 where $n = l_1 \times l_2$. A matrix is formed where rows are added for single error correction/double error detection. The columns are added for generating vertical parity bits.

For mitigating Multiple Cell Upsets (MCU), Decimal Matrix code [15] are another well-known approach. Though it reduces performance overhead, it corrects upto 2 bits of errors completely. IT relies upon an encoder which generates horizontal & vertical redundant bits and stores in the main memory.

In Horizontal Vertical (HV)[11],[16] parity code technique, data bits are first formed into a **n x m** matrix. Parity of each row is calculated and stored at the end in a new column. For column parity, data bits along columns are calculated.

Detection and correction of multi-bit soft error by using Horizontal - Vertical - Double - Bit -Diagonal (HVDD) [17] parity bits can detect all combinations of errors and correct up to 3 bit errors with a comparatively low overhead. Queen parity is added to HVD code to produce HVDQ [18] code. Errors at queen positions can be corrected by this coding technique.

Argyrides and Reviriego et al. [19] researched upon the impact of error correction and detection techniques on reliability of a system. They analyzed such approaches that use error correction codes, which in addition to soft errors can resolve defects, at the cost of reduced ability to correct soft errors. The results showed that low defect rates or small memory sizes are required to have a low impact on reliability. If correction rate is reduced, some systems tend to become unreliable.

Tay and Chang et al. [20] proposed a non-iterative multiple residue digit error detection and correction algorithm in redundant residue number system (RRNS). The received residue digits are divided into 3 groups from which 7 error location categories are defined for all combinations of residue digit errors of any legitimate moduli set.

Vasyl and Taras et al. [21] worked upon multiple error detection and correction based on modular arithmetic operations where they used residue number system (RNS) to correct single errors.

In semiconductor memories, soft errors can be detected and corrected up to 5 bits by the proposed method of Sharma and Vijayakumar [22]. The method uses horizontal, vertical and diagonal parity codes. The parity bits are calculated at the receiver end for each row, column and diagonal in slash and backslash directions in a memory array. The parities are regenerated at the receiver end; the comparison of transmitted and received parity bits detects the error. As soon as the error is detected, the code corrects the detected error. Hamming code is used for error detection and correction. Multiple upsets tolerance in SRAM memory was also researched upon by

Argyrides & Zarandi et al. in [8].

Use of diagonal parity in both directions can be effective against soft errors. Kishani et al. [23] used these techniques with horizontal and vertical parity codes which can correct up to 3 bit errors in 64 bits data sets. 60 extra bits are required to perform such corrections with a bit overhead of 73.12%. Structuring data sets as cubes provide error correction benefits which can detect up to 15 bits of errors and correct up to 4 bit errors. The method was proposed by Aflakian et al. [24] which had greater complexity than most other systems. They focused on parity-check code for optimal error detection and correction utilizing check bits without degrading the data rate too much.

Problems of cube pattern still persists which was observed by Anne et al. [25]. The author organized the data sets in different layers where forms a cube. The parity bits are in the outer layer whereas data bits form the inner layers. A certain number of undetectable error patterns were recognized while performing the research.

Pflanz et al. [26] proposed a method which can detect and correct 5-bit error using 3 dimensional parity codes. This method cannot detect and correct all combination of 5-bit error in the data bits and it ignored the possibility of error occurrence in parity bits.

III. Tolerating Error by **SPG**

Erroneous bits occurring in memory cells can be divided into two terms.

- The actual data in memory cells = Sent Data Word
- The erroneous data in same memory cells = Received Data Word.

The proposed method can detect and correct errors in received data word using the principle of successive parity generation. This method processes erroneous bits by successively calculating the check bits and comparing the received data word with the calculated check bits.

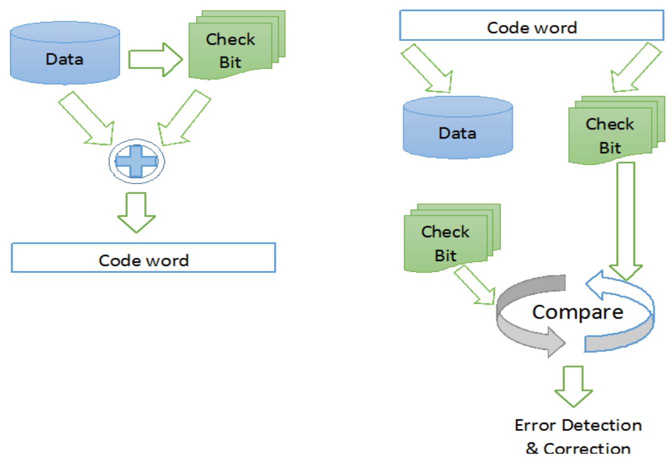


Fig 1. Block diagram of SPG method

n bit errors occurrence were considered in a code word. This section shows how the proposed methodology corrects the erroneous bits. The detection and correction procedure of the proposed method are described as follows.

A. *Encoding*

All successive parity bits are calculated by using the algorithm as shown in Fig. 2. **Step 1** includes initializing of a data block D of n bit, and a parity block P is organized as one-dimension array. Counter ***i*** is set to 0. In order to detect bit upsets in the code word, all check bits in the receiver end are needed to be calculated again. All the mentioned check bits can be calculated sequentially in serial to each other by **step 2** till i ≠ n. While computing the 1st check bit, other check bits can be computed later on. This property has a major impact in real-time applications and effective for high speed computing with reduced erroneous bits. After calculating the check bits for each data bits by **step 3**, i is incremented by 1 in **step 4.** The encoding process will terminate by **step 5**.

```
Step 1: Initialize data block D & parity block P.
        Initialize counter i to 0.

Step 2: Go to Step 3 if I ≠ n
        Else go to Step 5.

Step 3: $P_i = P_{i-1} \oplus D_i$.

Step 4: Increment i by 1.
        Go to Step 2.

Step 5: Exit.
```

Fig 2. Algorithm for Parity Generation by SPG.

The generation of parity bits of n bits data word with simple logical XOR operation is shown in Fig. 3.

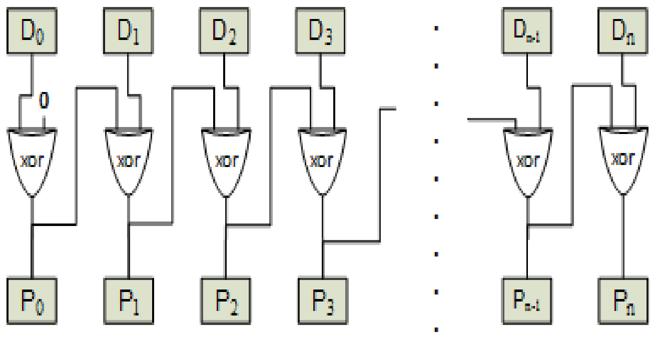


Fig 3. Check Bit Generation.

As Fig. 3 refers, the generation of 1st parity bit is originated from XOR of D_0 & 0. Consecutively, the 2nd parity bit P_1 originates from XOR of 1st parity and D_1. This operation is performed continuously to generate all successive parity bits. Along with n-bit check bits total 2n-bit code word is generated at the end of the process.

B. Detection Process

After any erroneous events such as SEU or MCU, we again calculate received check bits of n-bit from received data word by following the same encoding method.

The generated check bits are checked bit by bit with the received check bits. Each nth (generated) check bit is compared with nth (received) check bit. If they mismatch, the nth data bit is erroneous. The working procedure of the detection & correction scheme is shown in Fig. 4.

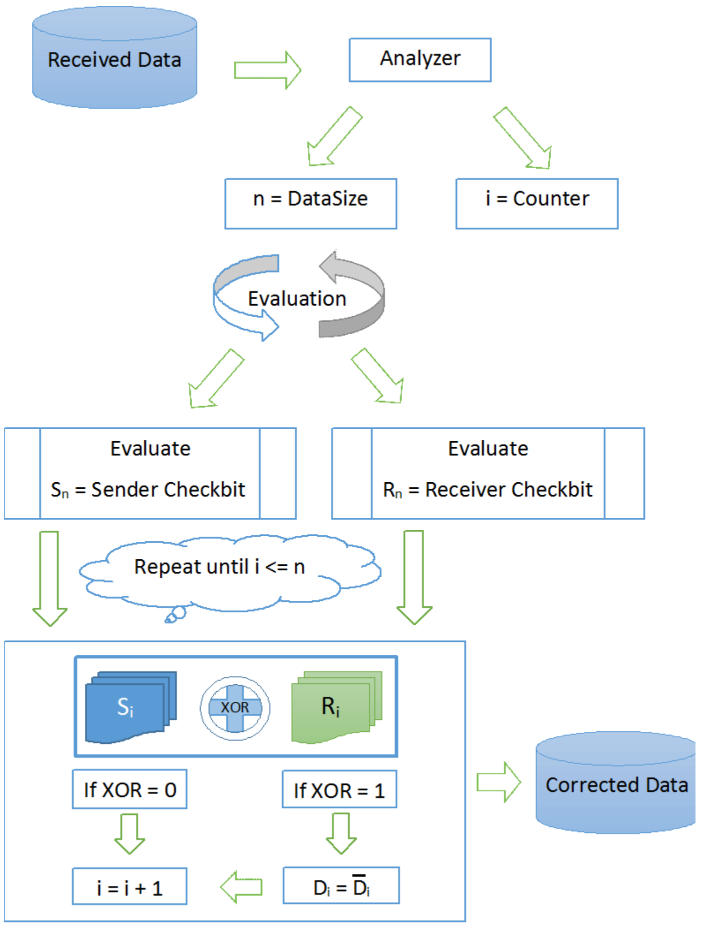


Fig 4. Error Detection & Correction by SPG.

The received data is first passed through the analyzer. The function of the analyzer is to calculate the size of the data. It assigns n and a counter i to total no of bits in data. The evaluation process begins to differentiate between the data bits and checkbits. It assign Sn to Sender checkbits. New checkbits are calculated from received data bits and assigns to Rn. For each bit position, XOR operation is performed between Si and Ri . If the result is 0, then i is incremented by 1 and next bit positons are checked. If the result is 1, Di is flipped and i is incremented by 1 also. Thus, data is partially corrected in each iteration. The process is repeated until i <= n .At the end of n no. of iterations, the process ends and n bit corrected data is available at the receiver end.

C. Error Correction

By following the algorithm to detect & correct error shown in Fig. 4, the parity bits are checked. In Fig. 5, error detection and correction sequence can be seen for the Data bit 0, where received check bit 0(S0) and calculated check bit 0(R0) is compared. If the check bits mismatch, then Data bit 0 is flipped and a new set of check bits are calculated and again compared with the received check bits. The operation is performed repeatedly till the received and generated check bits matches completely, thus making sure that n-1 bits of data are correct while checking the nth data bit. Total n bits are compared and corrected to produce a total n bits of error free data.

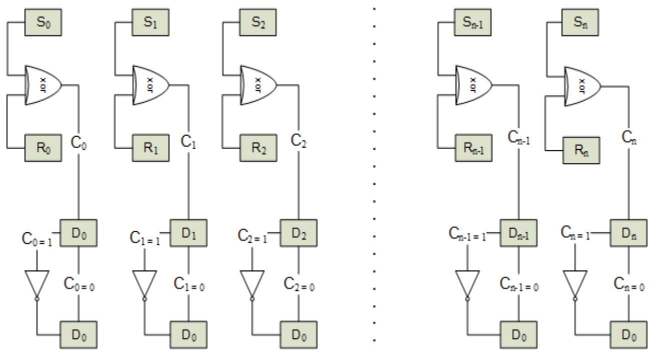
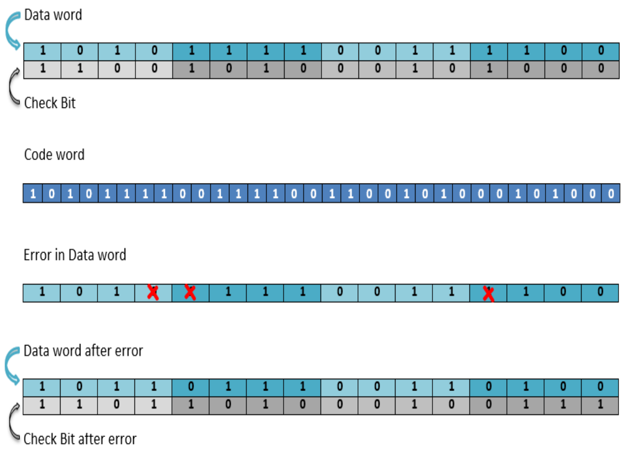


Fig 5. Error Correction Scheme.

D. A simplified Example

To clarify the SPG method, a 16-bit word for instance is considered in Fig. 6. The cells from D_0 to D_{15} are data bits. S_0 to S_{15} are check bits. For example, in this situation, 3-bit of errors (3rd, 4th & 12th bit positions) are injected and corrected using SPG.


Data word
Check Bit
Code word
Error in Data word
Data word after error
Check Bit after error

1	0	1	0	1	1	1	1	0	0	1	1	1	1	0	0
1	1	0	0	1	0	1	0	0	0	1	0	1	0	0	0

1	0	1	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	0	0	1	0	1	0	0	0	1	0	1	0	0	0

1	0	1	✗	✗	1	1	1	0	0	1	1	✗	1	0	0

1	0	1	1	0	1	1	1	0	0	1	1	0	1	0	0
1	1	0	1	1	0	1	0	0	0	1	0	0	1	1	1

Fig 6. Error in Data word and Checkbit Generation.

$S_0 = D_0 \oplus 0$

$S_0 = 1 \oplus 0 = 1$

$S_1 = S_0 \oplus D_1$

$S_1 = 1 \oplus 0 = 1$

- - - - - - - - -

$S_3 = S_2 \oplus D_2$

- - - - - - - -

$S_{15} = S_{14} \oplus D_{15}$

After injection of errors as Fig. 6 on bit word, we again calculate check bits R_0 to R_{15}.

$R_1 = R_0 \oplus D_1$

$R_2 = R_1 \oplus D_2$

- - - - - - - -

$R_{15} = R_{14} \oplus D_{15}$

S and R are compared in Fig. 7. They are first mismatched on bit 3, so D_3 is changed due to error. Now we flip D_3 to get the correct data bit.

So, $D_3 = \overline{D_3}$

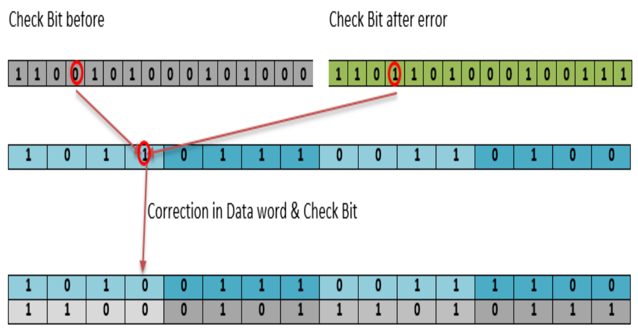


Fig 7. Correction of First Error Bit

After this correction, we again calculate check bits on this partially corrected data word in Fig. 8. Now it can be seen that S and R are mismatched on bit 4, we flip bit 4 to get correct data bit.

So, $D_4 = \overline{D_4}$

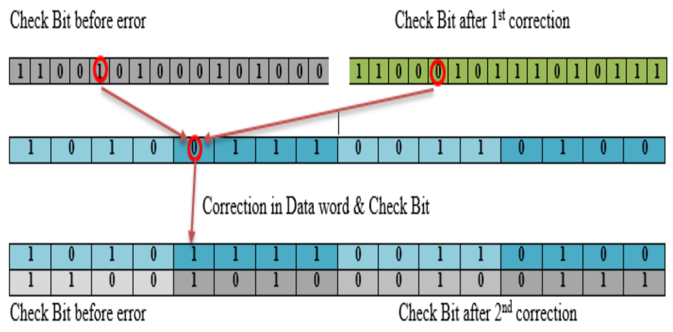

Check Bit before error
Check Bit after 1st correction
Correction in Data word & Check Bit
Check Bit before error
Check Bit after 2nd correction

Fig 8. Correction of Second Error Bit

Using same procedure, we corrected bit 12 in Fig. 9. And finally S and R fully matches thus we have corrected all the errors.

So, $D_{12} = \overline{D_{12}}$

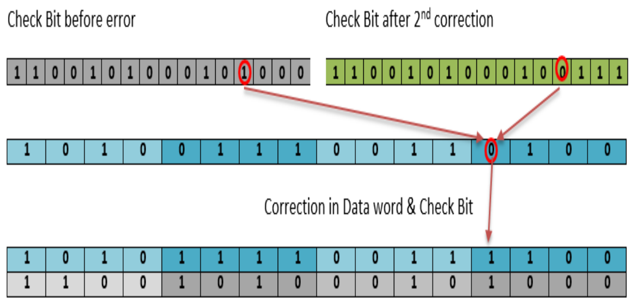

Check Bit before error
Check Bit after 2nd correction
Correction in Data word & Check Bit

Fig 9. Correction of Third Error Bit

IV. Experimental Analysis

In this chapter, we consider any errors occurrence in a code word and show how the proposed methodology corrects it. The detection and correction method of SPG are compared with other ECC techniques. The efficiency of the proposed method is validated in this chapter.

A. Experimental Setup

Intel(R) Core™ i5-4130 CPU @ 3.40 GHz

- RAM 8 GB
- Language Python 2.7
- IDE Python IDLE

B. Experimental Results

Fault injection is one of the important method to estimate the fault detection coverage of different error correcting codes. In order to estimate the fault detection coverage of the proposed the method, we used the fault injection method on software simulation level.

The conventional error checking and correction methods deals with data words in different manners. Table 1 depicts the performance comparison between different methods, where first column represents the no. of data bits, second column represents no. of parity bits needed for each ECC method. In the third column, the no. of erroneous bits which can be corrected by a method is tabulated. Fourth column represents total code word bits. Fifth and sixth column tabulates the bit overhead and code rate of each method. For example, Golay (23,1,2,7) codes [12] can correct up to 3 bits in a 11 bits data word. Whereas, HVD (64) can correct 3 bits error in 64 bits data. The proposed successive code can correct 64 bits out of 64 bits of data word.

Table 1: Performance Comparison of Different Codes.

Error Correcting Method	# of Data Bits	# of parity bits	Error Bits Correction	# of codeword bits	Bit Overhead (%)	Coderate (%)
Golay(23,12,7)	11	12	3	23	91.67	52.17
BCH(31,16, 7)	15	16	3	31	93.75	51.61
HVD(64)	64	50	3	114	78.12	56.14
HVDQ(64)	64	60	5	124	93.75	51.63
Successive Code	64	64	64	128	100	50

In Fig. 10, differences with respect to overheads and code rates can be seen graphically. Golay code [12] has an overhead of 90% and a coderate of 49%. BCH [13] follows closely to Golay with 92% & 50% respectively. HVD has reduced overhead to 73% and code rate remains highest among the ECC methods. DMC [15] and MC [14] both have higher overhead than 100%. SPG has moderate overhead in comparison with other methods.

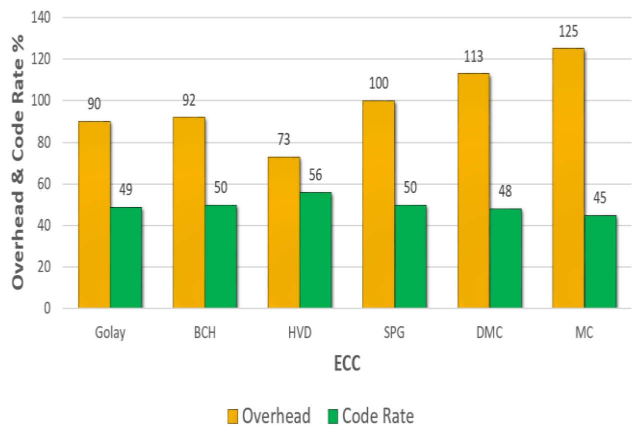


Fig 10. Comparison of Overhead and Code Rate

The proposed SPG has 100% overhead and 50% coderate. But in comparison with MC [14] technique, which has 125% overhead and 45% coderate, our method performs relatively well. Though DMC [15] has 113% overhead, it's still 13% expensive in overhead size than SPG.

The correction rate of any Error detection and correction method is the most significant feature to be noted. SPG performs remarkably with respect to conventional methods as can be seen on Fig. 11. It has a correction rate of 100, where BCH [13] and Golay [12] corrects 33.33% & 27.27% bits. The HVD and HVDQ techniques falls below 10% correction rate with 4.68% & 7.1% respectively.

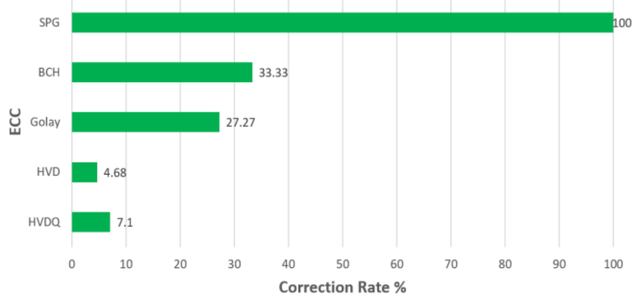


Fig 11. Comparison of Correction Rate.

The comparison between overhead and data size for different coding techniques is shown in Fig. 12. The proposed method performs better than other methods. For 1Mb to 2Mb, the overhead size remains almost similar for different methods. The differences in overhead can be clearly seen if the data word size increases further.

For 8Mb data word, BCH [13] has 2.64Mb and Golay [12] has 2.16 Mb overhead. MC [14] & DMC [15] has 10 Mb and 9.04 Mb respectively. SPG remains same to 8Mb. For 16Mb data size, Both BCH and Golay have 5.28Mb & 4.32 Mb overheads respectively. Whereas, SPG has 16Mb overhead, it performs quite well with comparison to MC & DMC. They have 20Mb and 18.08 Mb overhead for 16Mb data, relatively higher than SPG. Thus, SPG performs moderately better with respect to overhead & data size.

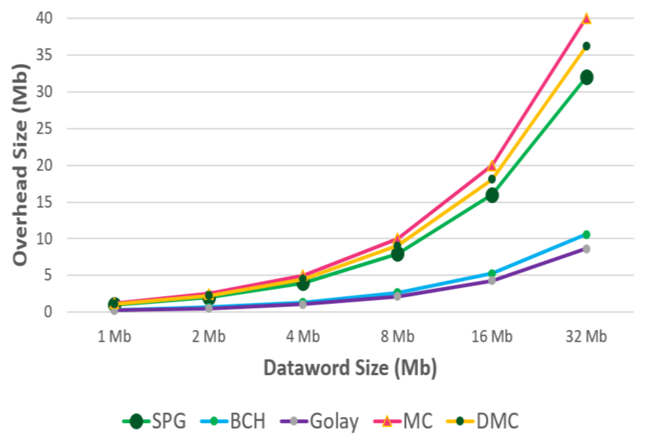


Fig 12. Comparison of Overhead vs Data Size.

In Fig. 13, the correction rate of our proposed SPG method is compared with closely related ECC techniques such as MC [14] and DMC [15]. Correction rate of MC, DMC and SPG is 100% for 1-bit error. It remains same for 2 bit errors for MC & SPG, but DMC fails to 96%. For 3 bit errors, DMC and SPG remains at 100% but MC falls short to 76.4%.MC technique works perfect for 2 erroneous bits, as the number of erroneous bits increases, its correction rate falls drastically. DMC well for close to 7 bit errors with a correction rate above 90%. It falls from that region as the erroneous bits increases.

For 16 bit errors, its correction rate falls to 9.8%. Whereas, the proposed system performs remarkably with 100% correction rate for any number of erroneous bits.

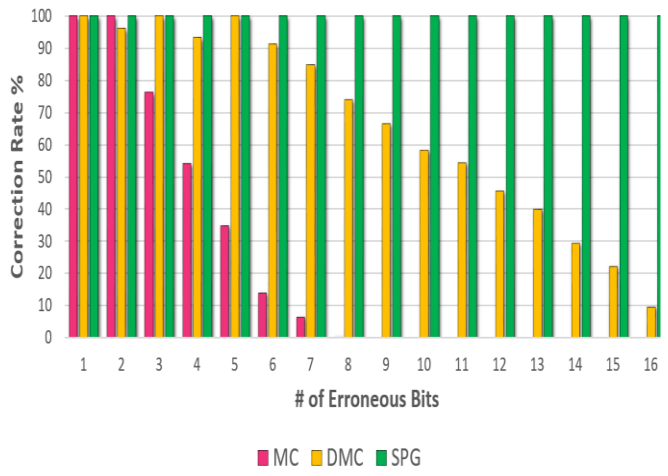


Fig 13. Correction Rate Over Different Bit.

To check the number of erroneous bits in a code word, different methods take different times. To test out the robustness of SPG, it was compared with MC [14] & DMC[15]. 32 bits data word is considered for comparing the correction time if any error occurs in that data word. SPG takes only 0.0137 seconds to actually detect and correct a bit of error from the 32 bit data word as can be seen in Fig. 14.

Whereas MC & DMC takes longer times to correct the error such as 0.0178 & 0.0183 seconds respectively. The time differences increase when the data word size increases along. SPG performs well with respect to the MC and DMC methods in the test setup which was organized as described in the experimental setup section. Systems with high performance capability will further help to detect the actual time differences if the methods with better precision.

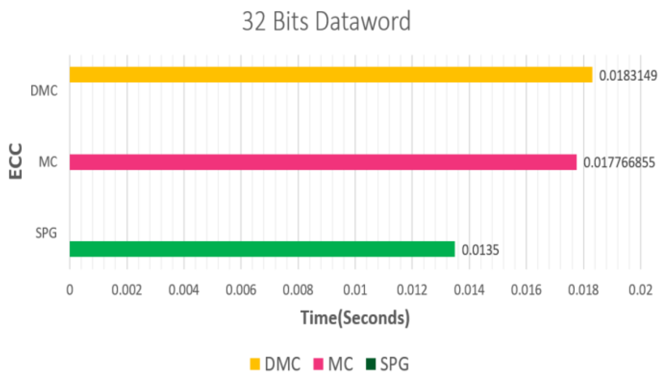


Fig 14. Run Time Comparison

V. CONCLUSIONS

The conventional error detection and correction methods are still being used is general application. Today's systems demand new and intuitive approach for error correction, instead of conventional methods. However, new methods have been discovered to increase the data correction rate on some applications. The thesis proposes a novel error correction coding scheme with a high error correction rate. The proposed method works well for large number of data. While sending a large number of data, the possibility of error occurrence increases. However, the existing dominant error correcting coding approaches could detect and correct up to a limited no of bit of errors. The proposed method shows the ability to detect and correct up to n bit errors by incurring a moderate overhead. For the systems where reliability is a matter of concern, the proposed method is effective. Further studies can be performed to find the scope to minimize bit overhead by some compression techniques for check bits.

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] 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.

[3] Ne’amHashemIbraheem“Error-Detecting and Error-Correcting Using Hamming and Cyclic Codes,” IEEE Transactions on Information Theory, vol. 51, no. 9, pp. 3347–3353, September 2005.

[4] J. Yang et al., "Radiation-Induced Soft Error Analysis of STT-MRAM: A Device to Circuit Approach," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 3, pp. 380-393, Mar. 2016.

[5] Jing Guo; Liyi Xiao; Tianqi Wang; Shanshan Liu; Xu Wang; Zhigang Mao, "Soft Error Hardened Memory Design for Nanoscale Complementary Metal Oxide Semiconductor Technology," Reliability, IEEE Transactions on , vol.64, no.2,pp.596,602, Jun. 2015

[6] R. Baumann, "Soft errors in advanced computer systems," IEEE Des. Test Comput., vol. 22, no. 3, pp. 258266, May/Jun. 2005.

[7] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, "Dependability in Electronic Systems: Mitigation of Hardware Failures," Soft Errors, and Electro-Magnetic Disturbances. New York, NY, USA: Springer-Verlag, 2010.

[8] Argyrides C, Zarandi HR, Pradhan DK, "Multiple upsets tolerance in SRAM memory," International symposium on circuits and system, New Orleans, LA, May,2007

[9] Ferreyra PA, Marques CA, Ferreyra RT, Gaspar JP,“Failure map functions and accelerated meantime to failure tests: new approaches for improving the reliability estimation in systems exposed to single event upsets,” IEEE Trans NuclSci 52(1), pp. 494–500.

[10] M. Nicolaidis, "Design for soft error mitigation," IEEE Trans. Device Mater. Rel.,vol. 5, no. 3, pp. 405418, Sep. 2005.

[11] Dubrova, E. 2012. "Fault Tolerant Design: An Introduction," 1st ed. Springer New York Heidelberg Dordrecht London. p. 87-131

[12] Golay Code from wolfram mathworld http://mathworld.wolfram.com/GolayCode.html last access on 22/03/2017

[13] Muhammad Imran, Zaid Al-Ars, Georgi N. Gaydadjiev "Improving Soft Error Correction Capability of 4-D Parity Codes," 14th IEEE European Test Symposium,May. 2009.

[14] Costas Argyrides, Dhiraj K. Pradhan, and Taskin Kocak, "Matrix Codes for Reliable and Cost Efficient Memory Chips," IEEE Transaction on

Very Large Scale Integration (VLSI) Systems, VOL. 19, NO. 3, Mar. 2011.

[15] Jing Guo, Liyi Xiao, Zhigang Mao and Qiang Zhao, "Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code," IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol: 22, Issue: 1, pp. 1-9, Jan. 2014.

[16] Koren, I. and Krishna, C. 2009. "Fault Tolerant Systems". 1st ed. Morgan Kaufmann Publishers. p. 55-74

[17] Md. Shamimur Rahman, Muhammad Sheikh Sadi and Sakib Ahammed ,Jan Jurjens, "Soft error tolerance using Horizontal-Vertical-Double-Bit Diagonal parity method," in 2nd International Conference on Electrical Engineering and Information and Communication Technology (ICEEICT) 2015 Iahangirnagar University,Dhaka-I 342, Bangladesh , 21-23 May 2015

[18] Md. Shamimur Rahman, Sakib Ahammed (2015). "Soft Error Tolerance using HVDQ:Horizontal-Vertical-Diagonal-Queen Parity Method," (Undergraduate Thesis). Retrieved from Rental Library, Dept. of CSE, KUET. (Accession No. CSER-15-01)

[19] C. Argyrides, P. Reviriego and J. Maestro, "Using Single Error Correction Codes to Protect Against Isolated Defects and Soft Errors," IEEE Transactions on Reliability, vol. 62, no. 1, pp. 238-243, Mar. 2013.

[20] T. F. Tay and C. Chang, "A Non-iterative Multiple Residue Digit Error Detection and Correction Algorithm in RRNS," IEEE Transactions on Computers, vol. 65, no. 2, pp. 396-408, Feb. 2016.

[21] Vasyl Yatskiv 1, Taras Tsavolyk 1, Hu Zhengbing, "Multiple Error Detection and Correction Based on Modular Arithmetic Correcting Codes," 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw, Poland., pp. 850-854, Sep. 2015.

[22] P. Vijayakumar, Shalini Sharma "An HVD based error detection and correction of soft errors in semiconductor memories used for space applications," IEEE International Conference on Devices, Circuits and Systems (ICDCS), pp. 563 - 567,15-16 Mar. 2012.

[23] M. Kishani 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," Automation for Embedded Systems, vol. 15, no. 3-4, pp. 289-310, Dec. 2011.

[24] Danial Aflakian , Dr. Tamanna Siddiqui , Najeeb Ahmad Khan ,Davoud Aflakian, "Error Detection and Correction over Two-Dimensional and Two-Diagonal Model and Five-Dimensional Model," International Journal of Advanced Computer Science and Applications, vol. 2, no. 7, pp. 16-19, 2011.

[25] N. B. Anne, S. Latifi, U. Thirunavukkarasu "Three and Four-dimensional Parity-check Codes for Correction and Detection of Multiple Errors," Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04), vol. 2, pp. 840, 2004.

[26] M. Pflanz, K.Walther, C. Galke, H.T. Vierhaus, "On-line error detection and correction in storage elements with cross-parity check," On-Line Testing Workshop,2002. Proceedings of the Eighth IEEE International, Jul. 2002.