International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Special Issue: 1 Volume No.7 April – 2023
DOI:10.46647/ijetms.2023.v07si01.019 ISSN: 2581-4621

Error Detection and Correction Using Minimal ParityBased on Matrices

[1]P.Anil Kumar, [2]R.Supriya, [3]M.Sravya, [4]G.Tejaswi, [5]M.Sharmila Bai
[1]Assistant Professor, Department of ECE, Annamacharya Institute of Technology and Sciences, Tirupati, A.P.
[2,3,4,5]B. Tech Student, Department of ECE, Annamacharya Institute of Technology and Sciences, Tirupati, A.P.

Abstract**—**Multiple cell upsets are caused by improvements in complementary metal oxide semiconductor (CMOS) technology (MCUs). MCUs had been a difficult problem for data storage in memory for numerous applications because of the radiation particles. Because of their simple encoding and decoding, error correction codes are one of the approaches that are employed more frequently to safeguard memories. Typically, MCUs have an impact on nearby memory bits. Consequently, a productive strategy would be one that would find and rectify as many neighboring bits as possible. Matrix-based code has only one drawback: it requires a large number of parity bits to facilitate error correction in memories. We reduced the number of parity bits in this document to address the flaw. When compared to other existing strategies, the suggested technique provides an equivalent error-correction capability with a lower parity bit count. Area, power, and delay time have all decreased, along with the total number of parity bits, by 30%, and by 1.12%, 33.59%, and 55.46%, respectively. These factors render the suggested strategy for memory protection effective and efficient. Applications where parity bits and speed are strictly regulated can employ this technique.
Keywords: Parity bits, Multiple Cells Upset (MCU), ErrorCorrection Codes (ECCs), Matrix Based Codes.

I. INTRODUCTION
Memory is a term used to describe a data storage area. Data in memory is encoded while being stored and decoded when being retrieved. Information can be stored by using distinct storage components and clusteharing them together. Each memory cell has the capacity to hold a single bit of data. The reliability of memory is seriously threatened during data storage by soft mistakes brought on by particle radiation [1][2]. Errors happen when radiation particles come in contact with the system's sensitive parts [3]. The result is a corruption of the data kept in memory. As a result, matrix-based codes are frequently utilized [4][5] to safeguard the data bits in memories. From technique to technique, the bit layout differs. Fig. 1 depicts our method of layout. Some parity bits will also be kept in memory along with the data bits. So, the task of the encoder is to change the p-bit data that is provided into r-bit data, where p-bits are data bits and r-p bits are parity bits. If the ECCs are able to repair the bits in the memory, the decoder takes on the task of doing so and outputs the original data. The complexity of the decoder and encoder circuits as well as the quantity of parity bits determine the overhead of the memory system that is impacted by radiation. For various purposes, different ECCs with overheads and varied correction abilities are chosen.
Based on their demands, such as performance, time, and area complexity. Because they are simple to use, Single Error Correction and Double Error Detection codes (SEC-DED) offer a different approach to correcting ECCs. SEC-DEC codes are unable to fix memory faults brought on by MCUs [4]. MCUs are common as a result of technology scalability. The MCUs appear to target nearby cells whose spacing are shorter than two to three bits, according to recent radiation experiments [6]. Certain types of MCUs, including OLS and DS codes, among others, are handled by ECCs with powerful correction capabilities. They are employed for memory-related purposes. Hence, while choosing ECCs to secure these memories, redundancy overheads or the correction capability are taken into account as key factors. So, in the current work, these aspects are optimized.

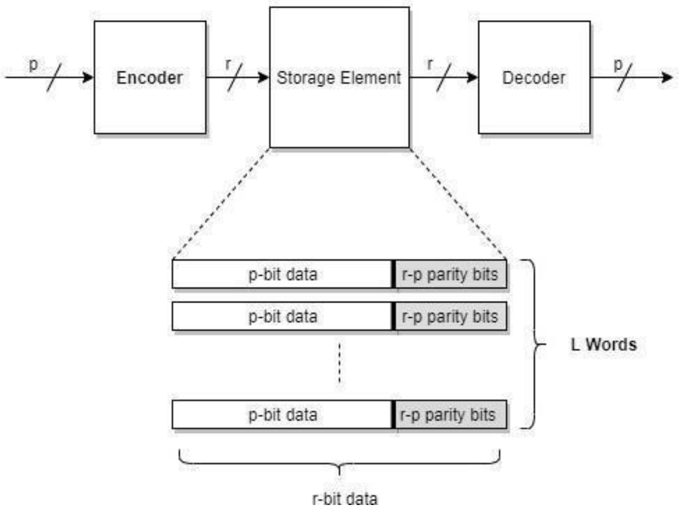

Fig. 1. Data Organization in memory

The data acquired from various survey papers has been mentioned in Section II and will be discussed in the following parts. The new matrix-based code and its decoding and encoding procedures are described in Part III. The various cases that the suggested technique can handle and its capacity for rectification are discussed in Section IV. Bitrate, as well asPart V presents the parametric results, and Section VI concludes the discussion.

II. RELATED WORKS

One of the most used strategies in memory applications is matrix-based coding. The single and multiple error detection and correction, as well as the reduction of area, power, and bit overhead, are all goals of the coding schemes.
The combination of parity codes and hamming codes contains more redundant bits, claims the work [7]. These data bits are stored in the encoder as a matrix of order (k x z), and order (11,8,4) diagonal, vertical, and horizontal parity sharing bits are employed to encode the data. Whereas the HVHC (Horizontal Vertical Hamming Code) block corrects errorsthat happened in parity bits, the 3PC (3-Parity Check) block identifies and corrects faults that occurred in data bits. This technique finds and fixes three-bit errors in both parity and data bits. In the paper Adjacent Error Correction using Matrix Based Codes [8], a word is stored in a matrix configuration that is 4 x 8 and includes the extra parity bits generated horizontally and vertically.
In addition to the 32 data bits illustrated in Fig. 2 with this method, there are an additional 8 Hamming bits and 12 vertical parity bits. This technique is typically used for the identification and correction of adjacent 4-bit errors. To eliminate clustering mistakes brought on by MCUs, technique uses up more space and money.
This method allows for the detection and correction of all single bit faults as well as the protection of memories from MCUsand SEUs. With the addition of 8 parity bits and 20 Hamming bits, a 32-bit data set is organized in matrix format of order 4 x 8 in the work [9]. It can correct any subset of eight faults in a row as long as the other rows have only one bit error, and it can repair one-bit errors as well as numerous errors in a row. The only issue is that when several problems do occur in multiple rows, some of the errors are fixed but others remain corrupted. The 32-bit data isgrouped in the [10] approach, 28 parity bits are calculated in the encoder, and 8-bit faults can be fixed using this methodology. Here, a 1-bit error can also be fixed.

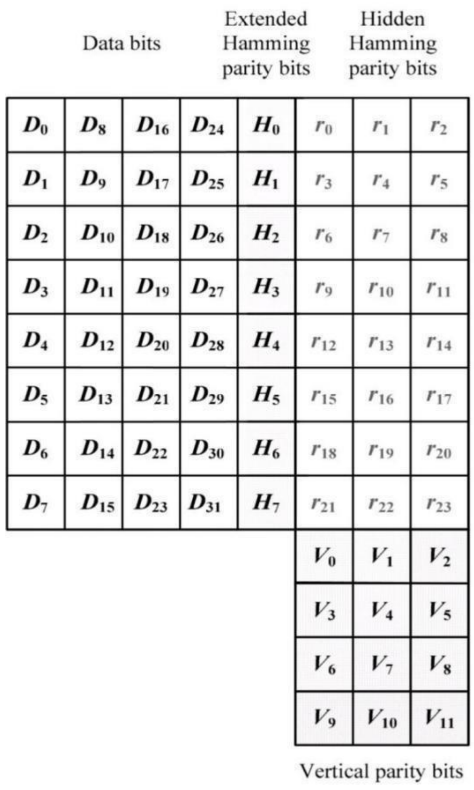

Fig. 2. Logic diagram for Low Redundancy Matrix-Based codes for Adjacent Error Correction with Parity Sharing

This method's disadvantage is that it cannot fix parity bit issues. Although it is impossible, the parity bits are thought to be error-free. Data bits will be stored in the way described in [11][12] as a matrix of order (8 x 4). In which the extended Hamming coding method is employed to produce extended parity bits, which are then utilized to detect flaws. The hidden faults are corrected using both hidden bits and regular parity bits. The fundamental flaw in this method is the inability to recognize and correct errors in data bits and associated parity bits that are caused by MCUs.

III. PROPOSED TECHNIQUE

The k-bit data in the proposed technique is organized as a m x n matrix, where the rows and columns are denoted by m and n, respectively (i.e., k = m x n). The data is organized in a matrix row by row, in contrast to the previous work Adjacent Error Correction using Matrix Based Codes. From all the data bits, four Vertical Hamming bits and ten Parity Sharing bits are derived. The bits M8 and M9 in our study are calculated differently using bits that are present in the matrix's DNA (Deoxyribonucleic Acid)-shaped curve, as illustrated in Fig. 3. Also, those portions are split into two parts, which will be discussed further. Consider a 32-bit data set as an example to help clarify our work. The data bits are arranged in a matrix form with m = 8 and n = 4 and are labelled D0 through D31. When the column bits are calculated as shown in equation 1, the extended Hamming parity bits (V0 - V3) are useful for finding faults in the column bits. Equation 1 describes how V0 is computed, as well as how V1 - V3 is determined. Equations 2 to 5 are used to determine how to calculate the horizontal parity sharing bits (M0 to M9). Equation 2 shows how to calculate M2, M4, M6, in the same way that M0 is calculated, and Equation 3 explains how to calculate M3, M5, and M7 in the same manner that M1 is calculated. M8 (shown in Figure 3 as Blue) and M9 (shown in Figure 3 as Yellow) are determined in a special fashion that is reflected in Equations 4 and 5. Equations 1 through 5 contain the Hamming Equation. As a result, the fundamental operation XOR is used to create all of the equations. The overall number of redundant bits calculated has been decreased and the use of imaginary bits has been eliminated in this technique when compared to previous work on adjacent error correction utilizing matrix- based codes. For extended, syndrome bits are calculated.

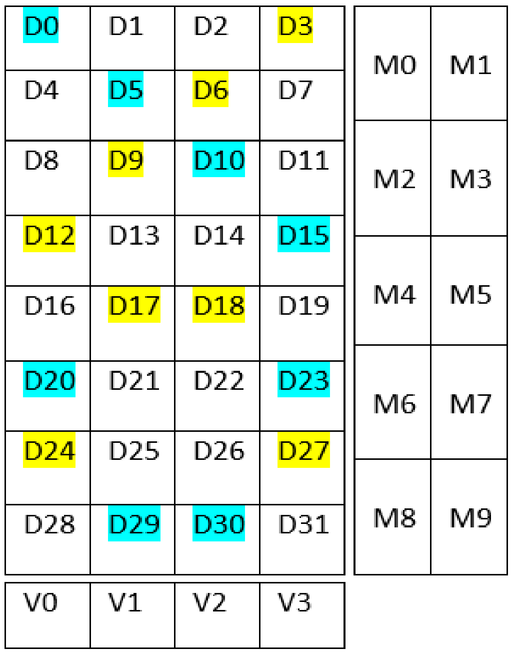

Fig. 3. Logic diagram for Proposed Technique

Bits of hamming parity (V). The parity bit of that particularcolumn will aid in identifying any bits that are affected by errors.

$$V0 = D0 \oplus D4 \oplus D8 \oplus D12 \oplus D16 \oplus D20 \oplus D24 \oplus D28 \quad (1)$$

$$M0 = D0 \oplus D1 \oplus D2 \oplus D16 \oplus D17 \oplus D18 \quad (2)$$

$$M1 = D1 \oplus D3 \oplus D17 \oplus D19 \quad (3)$$

$$M8 = D0 \oplus D5 \oplus D10 \oplus D15 \oplus D20 \oplus D23 \oplus D29 \oplus D30 \quad (4)$$

$$M9 = D3 \oplus D6 \oplus D9 \oplus D12 \oplus D17 \oplus D18 \oplus D24 \oplus D27 \quad (5)$$

Next, the (M0–M9) horizontal parity sharing bits are computed. The corresponding parity bit of that particular row will aid in the detection of any error-affected bits if any are present. The error bit is now fixed. Let's use an illustration to comprehend the procedure. If the fault affects the D1 bit, the syndrome (V0 - V3) will equal 0100, which denotes an error in the second column. Then M0 to M9 horizontal parity sharing bits will aid in identifying that specific row of errors (1st row). Finally, an error-containing bit is found and fixed. Itis vital to note that because there are no distinct parity bits for error detection in the parity bits themselves, data bits cannotbe identified when there are mistakes in both the parity and data bits that are caused by MCUs. In order to prevent this kind of Parity bits may be spaced apart by an interval more than 3 bits while saving data bits to memory.
These kinds of setups typically prevent parity bit mistakes. On the other side, if there are no errors, additional decoding is omitted, which cuts down on latency. The suggested technique's primary superiority is that it corrects all neighboring horizontal mistakes in rows.

IV. EVALUATION

When the encoder receives 32-bit data, it outputs 46 bits, which are made up of data bits and parity bits. Then, the decoderreceives these 46 bits as input and checks them for data bit errors. In this section, we discussed many likely MCUs, along with instances of possible memory events and the decoding procedure used to correct the data.

A. *One Bit Error*

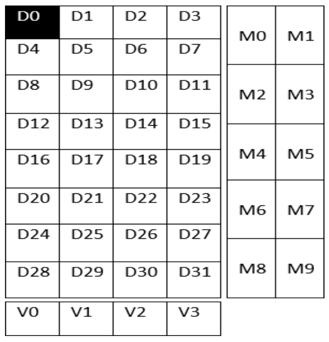

Fig. 4. 1-bit error

For instance, if D0 in Fig. 4 is a damaged bit, errors will appear in M0, V0, and M8. Decoder will therefore recognize D0 bit, and data bit is reversed for restoration.

B. *Two Bit Error*

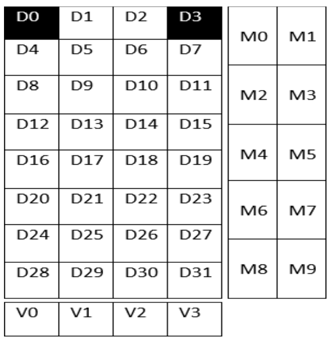

Fig. 5. 2-bit error

If D0 and D3 are assumed to be corrupted bits, asshown in Fig. 5, errors will appear in M0, V0, M1, V3, M9,and M8. Decoder will therefore recognize D0 and D3 bits, and data bits are reversed for restoration.

C. *Three Bit Error*

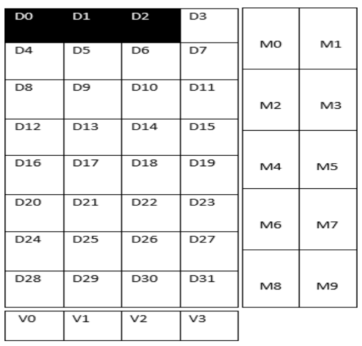

Fig. 6. 3-bit error

F. Parametric Results

Verilog HDL is used to create both the suggested code and the concealed parity code, which is then simulated in Xilinx Vivado. Using Xilinx ISE, both codes were implemented in Vertex-6. Table II shows that, in comparison to adjacent error correction utilizing matrix-based codes, significantly reduced delay, area, and power have been obtained. The memory's bit storage capacity has been decreased from 52 to 46 bits, and the decoding procedure is carried out without the usage of imaginary bits, as in the work Adjacent Error Correction using Matrix Based codes. According to Table III, consumption of power, area, and delay is decreased by 1.26%, 5.55%, and 21.17%, respectively.

In the event that D0, D1 and D2 are corrupted bits, as in Fig. 6, errors will appear in M0, M1, V0, V1, V2 and M8. Decoder will therefore recognize D0, D1, and D2 bits, and data bits are reversed for restoration.

D. Four Bit Error

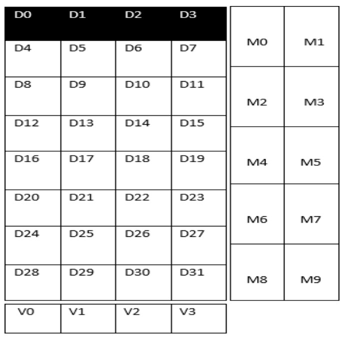

Fig. 7. 4-bit error

According to Fig. 7, if D0, D1, D2 and D3 are corrupted bits, faults will appear in M0, V0, V1, V2, V3, M9, and M8. As a result, the decoder will recognize the D0, D1, D2, and D3bits and flip the data bits for restoration. The key characteristicof our suggested method, "Adjacent Horizontal Four-Bit ErrorCorrection," is this.

E. Bit Rate

The Bit Rate is calculated using,

Table II: Comparison of Area, Power and Delay

Codec	Area	Total Power(μW)	Delay(ns)
Matrix based Codec	72	3.649	2.153
Proposed Codec	68	3.603	1.697

Table III: Percentage Reduction of Area, Power and Delay

Technique	Area %	Power%	Delay %
Encoder	15%	0%	26.96%
Decoder	1.92%	1.95%	57.48%
Proposed Codec	5.55%	1.26%	21.18%

IV. CONCLUSION

The suggested method requires just 14 parity bits and can quickly identify and fix neighboring

problems. Power, area, and latency have all been significantly reduced using this technology. These calculations for Vertex-6 reveal that the power, area, and time delay consumption are lowered by a combined maximum of 1.26%, 5.55%, and 21.17%. The proposed code fits the memory system's rigorous requirements for area and time delay better, according to the results. This approach effectively corrects all single bit errors, a few two- bit errors, and up to four-bit adjacent faults.
In terms of future application, the technique can be extended to fix data bits when there are an even number of column bit mistakes. a method that can fix mistakes even in parity bits.

REFERENCES

[1] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo and T. Toba, "Impact of scaling on neutron-induced soft error rate in SRAMs From a 250 nm to a 22 nm Design Rule", IEEE Trans. on Electron Devices, Vol. 57 , No. 7, pp. 1527-1538, July. 2010.
[2] Shivani Tambatkar, Siddharth Narayana Menon, Sudarshan.V, M.Vinodhini and N.S.Murty , "Error Detection and Correction in Semiconductor Memories using 3D Parity Check Code with Hamming Code ," International Conference on Communication and Signal Processing, April 6-8, 2017, India.
[3] Shanshan Liu, Liyi Xiao, Jie Li, Yihan Zhou, and Zhgang Mao, "Low Redundancy Matrix-Based codes for Adjacent Error Correction with Parity ," 18th International Symposium on Quality Electronic Design (ISQED),2017
[4] Sunita M.S and Kanchana Bhaaskaran , " Matrix code based Multiple Error Correction Technique for N-bit memory data" , International Journal of VLSI design Communication Systems (VLSICS) Vol.4, No.1, February. 2013.
[5] Paromita Raha, M Vinodhini, N. S. Murty "Horizontal-Vertical Parity and Diagonal Hamming Based Soft Error Detection and Correction for Memories," Jan. 05 – 07, 2017, Coimbatore, INDIA.
[6] Costas A. Argyrides, Pedro Reviriego, Dhiraj K. Pradhan, and Juan Antonio Maestro ,"Matrix-Based Codes for Adjacent Error Correction in storing data," IEEE Transactions on Nuclear Science, Vol. 57, No. 4, August. 2010.
[7] N. N. Mahatme, B. L. Bhuva, Y. P. Fang, and A. S. Oates, "Impact of strained-Si PMOS transistors on SRAM soft error rates," IEEE Trans. Nucl. Sci. , Vol. 59, No. 4, pp. 845–850, August. 2012.