Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Novel Methodology for Detection and Correction of Errors in Memory using Parity Matrix Code (PMC)

[1]Neha Ibrahim, [2]Vinoj P.G., [3]Sunil Jacob
[1]PG Scholar, [2, 3]Professor
[1, 2, 3] VLSI and Embedded Systems, SSET,
Ernakulam, Kerala, India

Abstract*—Constant shrinkage in the device dimensions by scaling of CMOS technology to nano-scale level has resulted in very dense high speed memory cells, also resulting in increase of fault occurrence in the memory. Soft errors and Multiple Cell Upsets (MCUs) are becoming a major reliability issues in memory exposed to extraordinary conditions like radiation and ionization. Thus we need some method for fault-tolerant memory cells. Conventional Error Correcting Codes (ECC) cannot correct multiple errors in memories even though many of these are capable of detecting multiple errors. The existing Decimal Matrix Code (DMC) based method for memory protection require more redundant bits to maintain higher reliability of memory. This paper presents a novel Parity Matrix Code (PMC) based encoding and decoding method to assure reliability in presence of multiple bit flip. PMC reduces number of redundant bit and it corrects more errors compared to existing system. The existing DMC and proposed PMC is compared to well known codes such as Hamming Codes, Matrix Code and Punctured Difference Set Code in terms of area used, delay and power resulting in proving minimal overhead for proposed PMC.

Keywords*— Soft Errors, Multiple Cell Upsets (MCUs), Error Correction Codes, Error Detection and Correction (EDAC), Decimal Matrix Code (DMC), Parity Matrix Code (PMC).

I. INTRODUCTION

As CMOS technology scales down to nanoscale and memories are combined with an increasing number of electronic systems, the soft error rate in memory cells is rapidly increasing, especially when memories operate in space environments due to ionizing effects of atmospheric neutron, alpha-particle, and cosmic rays[2]. Although single bit upset is a major concern about memory reliability, multiple cell upsets (MCUs) have become a serious reliability concern in some memory applications. In order to make memory cells as fault-tolerant as possible, some error correction codes (ECCs) have been widely used to protect memories against soft errors for years [4],[13]. For example, the Bose–Chaudhuri–Hocquenghem codes [6], Reed–Solomon codes [3], and punctured difference set (PDS) codes [16] have been used to deal with MCUs in memories. But these codes require more area, power, and delay overheads since the encoding and decoding circuits are more complex in these complicated codes.

Interleaving technique has been used to restrain MCUs[7], which rearrange cells in the physical arrangement to separate the bits in the same logical word into different physical words. However, interleaving technique may not be practically used in content-addressable memory (CAM), because of the tight coupling of hardware structures from both cells and comparison circuit structures[1],[11]. Built-in current sensors (BICS) are proposed to assist with single-error correction and double-error detection codes to provide protection against MCUs. However, this technique can only correct two errors in a word.

More recently 2-D matrix codes (MCs)[14] are proposed to efficiently correct MCUs per word with a low decoding delay, in which one word is divided into multiple rows and multiple columns in logical. The bits per row are protected by Hamming code, while parity code is added in each column. For the MC [14] based on Hamming, when two errors are detected by Hamming, the vertical syndrome bits are activated so that these two errors can be corrected. As a result, MC is capable of correcting only two errors in all cases. An approach that combines decimal algorithm with Hamming code has been conceived to be applied at software level in [15]. It uses addition of integer values to detect and correct soft errors. The results obtained have shown that this approach have a lower delay overhead over other codes.

Existing decimal matrix code (DMC) based on divide-symbol has been proposed with enhanced error correction capability and has lower Area, Power and delay overheads. It uses decimal integer addition and subtraction for error detection and correction. It uses Encoder Reuse Technique to reduce the delay and has less encoder and decoder circuit complexity. But it needs more number of check bits for memory protection.

This paper gives a novel Parity Matrix Code (PMC) based method. A parity algorithm (matrix multiplication and matrix addition) is used to identify and fix many errors which need fewer number of check bits compared to decimal matrix codes (DMC). This fixes maximum number of slips in memories and has minimal performance overheads.

This paper is divided into the following sections. Background works is discussed in Section II. The existing DMC is presented in section III. The proposed PMC is discussed in section IV. Results of PMC and DMC are shown in section V. Finally, overhead analysis some decisions of this paper are conferred and united in Section V.

II. BACKGROUND WORKS

The CMOS technology scaling to nm, low cost, high density, high speed integrated circuits with low supply voltage has increased the probability of fault occurrence in the memories. This lead to the major reliability concern especially increases SRAM memory failure rate. Some commonly used mitigation techniques are triple modular redundancy, and error correction codes (ECCs).

Soft errors are the major issue in the reliability of memories. Soft error will not damage the hardware, they only damage the data that is being processed. If detected, soft errors are corrected by rewriting corrected data in the place of erroneous data. Highly reliable system uses error correction approach, however in many systems it is difficult to correct data, or even impossible to detect error. To prevent soft errors from causing corruption in the data stored error correction codes are used such as matrix code, hamming etc. when ECC is used, data are encoded when written in the memory and data are decoded when read from the memory. Thus the encoding and decoding process possess a vital impact on the memory access time and complexity.

Multiple cell upsets have become the reliability concern in some application apart from single cell upset. The BCH code, Reed Solomon code etc are used to deal with MCUs, but the area, power and delay overhead of these codes are high due to the complex encoding and decoding architecture. The decimal matrix code uses encoder reuse technique which uses encoder as apart of the decoder and thus reduces the area overhead and complexity.DMC enchances the reliability of the memory by improving the error correction capability.

A. Error Detection and Correction

EDAC methods are used to find that the data is error free or is not corrupted, either by noisy channel, by hardware failure or during read-write operation in the memory segment. Various error detection methods exist in the communication system. One method currently utilized to produce reliable memory is the use of Error Correction Codes (ECC) to encode data before it is stored in the memory. Error correction codes take a set of information bits at the producer of the information and create a set of redundant bits based on the information bits. These redundant bits are sent or stored with the original set of information bits. The consumer of the information then uses the redundant bits to determine if any errors have occurred in transmission or storage. In the case of memory, the redundant bits are calculated and stored along with the original bits and then when they are read from the memory they are examined to determine if any errors have occurred between the time the information was stored and the time it was retrieved.

The most common error detecting and correcting scheme being employed are parity bit, CRC, HVD and Hamming codes. All these methods are implemented on the second layer of OSI model at Data link layer. The upper layers work on some generalized view of network architecture and are not aware of actual hardware data processing. Therefore, the upper layers require error-free transmission between two systems. Almost every application did not work if it receiver data with errors. Applications like voice and video may not get that much affected and may still function well with some error. Data-link layer uses some error control mechanism to ensure that data bit streams are transmitted with certain level of accuracy. But to recognize how errors can be controlled, it is important to know what types of errors may occur.

B. Hardware Redundancy Vs Software EDAC

In order to protect semiconductor memories, software EDAC or redundancy can be used. Redundancy can either be hardware redundancy that is provided by extra components or time redundancy that is provided by extra execution time or by different moment of storage or can be a combination of both the hardware and time redundancies. To allow redundancy to detect permanent faults, the repeated computations are performed differently. TMR (Triple Modular Redundancy) is a suitable technique for SRAM-based FPGAs because of its full hardware redundancy property in the combinational and sequential logic. One solution for the protection of memories is use of hardware redundancy techniques, but they are too costly.

When hardware redundancy is not possible, we have to go for software solutions. By using software Error Detection and Correction, transient faults in the combinational logic will never be stored in the storage cells, and bit flips in the storage cells will never occur or will be immediately corrected. For applications where read and write operations are done in blocks of words, such as secondary storage systems made of solid-state memories (RAM discs), software-implemented EDAC could be a better choice than hardware EDAC, because it can be used with a simple memory system and it provides the flexibility of implementing more complex coding schemes. With software EDAC, the data that is read from main memory may be erroneous, if the error occurs after the last scrub operation and before the time of reading. In other words, single-bit errors may cause failures. In contrast, hardware EDAC checks all the data that is read from memory, and corrects single-bit errors.

Therefore, hardware EDAC provides improved reliability and when feasible should be the first choice for protecting the main memory. When hardware EDAC is not available or affordable, software EDAC can be used as a low cost solution for enhancing the reliability of systems. For cases where data is read and written in blocks of words rather than individual words, software EDAC may be a better choice than hardware EDAC.

C. Types of Errors

There may be three types of errors:

1) Single bit error: In this a frame consists of only one bit corrupted anywhere throughout.

2) Multiple bit errors: In this a frame is received with more than one bits in corrupted state.

3) Burst bit errors: In this a frame contains more than one consecutive bits corrupted or more than one bit flips.

III. EXISTING DECIMAL MATRIX CODING (DMC) METHOD

DMC assures reliability in the presence of MCUs with reduced performance overheads, and a 32-bit word is encoded and decoded as an example based on the proposed techniques.

A. Schematic of Fault-Tolerant Memory

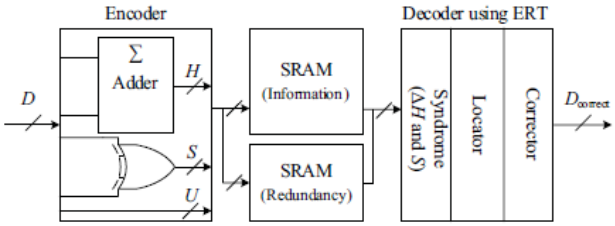


Fig.1. Schematic of fault-tolerant memory protected with DMC.

The schematic of fault-tolerant memory is depicted in Fig.1. First, during the encoding (write) process, information bits *D* are fed to the DMC encoder, and then the horizontal redundant bits *H* and vertical redundant bits *V* are obtained from the DMC encoder. When the encoding process is completed, the obtained DMC codeword is stored in the memory. If MCUs occur in the memory, these errors can be corrected in the decoding (read) process. Due to the advantage of decimal algorithm, the proposed DMC has higher fault-tolerant capability with lower performance overheads. In the fault-tolerant memory, the ERT technique is proposed to reduce the area overhead of extra circuits and will be discussed in the following sections.

B. DMC Encoder

In the DMC, first, the divide-symbol and arrange-matrix ideas are performed, i.e., the *N*-bit word is divided into *k* symbols of *m* bits ($N = k \times m$), and these symbols are arranged in a $k1 \times k2$ 2-D matrix ($k = k1 \times k2$, where the values of $k1$ and $k2$ represent the numbers of rows and columns in the logical matrix respectively). Second, the horizontal redundant bits *H* are produced by performing decimal integer addition of selected symbols per row. Here, each symbol is regarded as a decimal integer. Third, the vertical redundant bits *V* are obtained by binary operation among the bits per column. It should be noted that both divide-symbol and arrange-matrix are implemented in logical instead of in physical. Therefore, the proposed DMC does not require changing the physical structure of the memory.

To explain the DMC scheme, we take a 32-bit word as an example, as shown in Fig.2. The cells from $D0$ to $D31$ are information bits. This 32-bit word has been divided into eight symbols of 4-bit. $k1 = 2$ and $k2 = 4$ have been chosen simultaneously. $H0$–$H19$ are horizontal check bits; $V0$ through $V15$ are vertical check bits. However, it should be mentioned that the maximum correction capability (i.e., the maximum size of MCUs can be corrected) and the number of redundant bits are different when the different values for *k* and *m* are chosen. Therefore, *k* and *m* should be carefully adjusted to maximize the correction capability and minimize the number of redundant bits. For example, in this case, when $k = 2{\times}2$ and $m = 8$, only 1-bit error can be corrected and the number of redundant bits is 40. When $k = 4 \times 4$ and $m = 2$, 3-bit errors can be corrected and the number of redundant bits is reduced to 32. However, when $k = 2 \times 4$ and $m = 4$, the maximum correction capability is up to 5 bits and the number of redundant bits is 36. In this paper, in order to enhance the reliability of memory, the error correction capability is first considered, so $k = 2 \times 4$ and $m = 4$ are utilized to construct DMC.

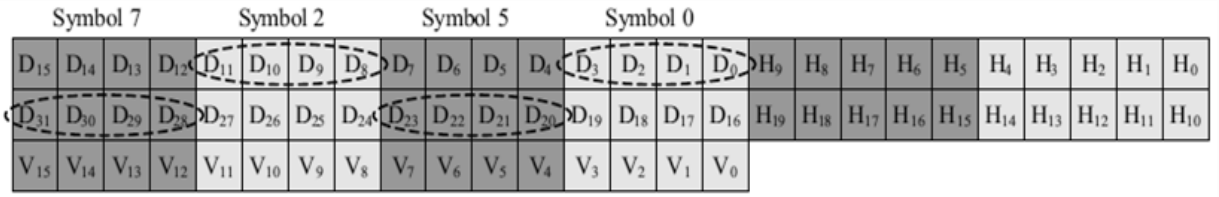


Fig.2. 32-bits DMC logical organization ($k = 2 \times 4$ and $m = 4$).
Here, each symbol is regarded as a decimal integer.

The horizontal redundant bits *H* can be obtained by decimal integer addition as follows:

$H4H3H2H1H0 = D3D2D1D0 + D11D10D9D8$ (1)

$H9H8H7H6H5 = D7D6D5D4 + D15D14D13D12$ (2)

Similarly for the horizontal redundant bits $H14H13H12H11H1$ and $H19H18H17H16H15$, where "+" represents decimal integer addition.

For the vertical redundant bits V, we have

$V0 = D0 \oplus D16$ (3)

$V1 = D1 \oplus D17$ (4)

and similarly for the rest vertical redundant bits.

The encoding can be performed by decimal and binary addition operations from (1) to (4). The encoder that computes the redundant bits using multibit adders and XOR gates is shown in Fig.3. In this figure, $H19 - H0$ are horizontal redundant bits, $V15 - V0$ are vertical redundant bits, and the remaining bits $U31 - U0$ are the information bits which are directly copied from $D31$ to $D0$. The enable signal En will be explained in the next section.

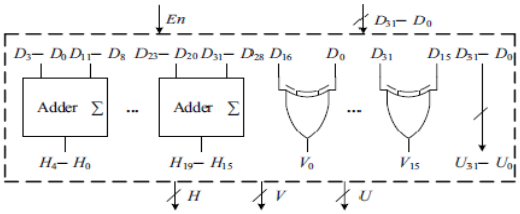


Fig. 3. 32-bit DMC encoder structure using multibit adders and XOR gates.

C. DMC Decoder

To obtain a word being corrected, the decoding process is required. For example, first, the received redundant bits $H4H3H2H1H0'$ and $V0'$-$V3'$ are generated by the received information bits D'. Second, the horizontal syndrome bits $\Delta H4H3H2H1H0$ and the vertical syndrome bits $S3 - S0$ can be calculated as follows:

$\Delta H4H3H2H1H0 = H4H3H2H1H0' - H4H3H2H1H0$ (5)

$S0 = V0 \oplus V0$ (6)

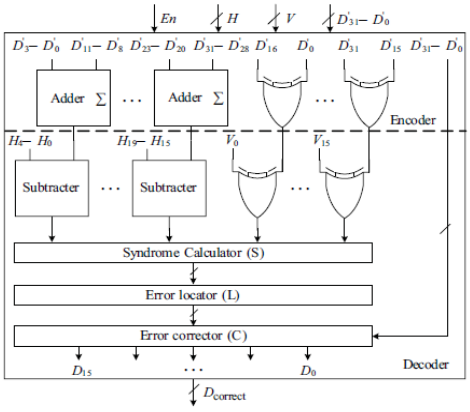
and similarly for the rest vertical syndrome bits, where "–" represents decimal integer subtraction.

When $\Delta H4H3H2H1H0$ and $S3 - S0$ are equal to zero, the stored codeword has original information bits in symbol 0 where no errors occur. When $\Delta H4H3H2H1H0$ and $S3 - S0$ are nonzero, the induced errors (the number of errors is 4 in this case) are detected and located in symbol 0, and then these errors can be corrected by

$$D0correct = D0 \oplus S0 \quad (7)$$

The DMC decoder is depicted in Fig. 4, which is made up of the following submodules, and each executes a specific task in the decoding process: syndrome calculator, error locator, and error corrector. It can be observed from this figure that the redundant bits must be recomputed from the received information bits D' and compared to the original set of redundant bits in order to obtain the syndrome bits ΔH and S. Then error locator uses ΔH and S to detect and locate which bits some errors occur in. Finally, in the error corrector, these errors can be corrected by inverting the values of error bits.

In this the circuit area of DMC is minimized by reusing its encoder. This is called the ERT. The ERT can reduce the area overhead of DMC without disturbing the whole encoding and decoding processes. From Fig. 4, it can be observed that the DMC encoder is also reused for obtaining the syndrome bits in DMC decoder. Therefore, the whole circuit area of DMC can be minimized as a result of using the existent circuits of encoder. Besides, this figure also shows the proposed decoder with an enable signal En for deciding whether the encoder needs to be a part of the decoder. In other words, the En signal is used for distinguishing the encoder from the decoder, and it is under the control of the write and read signals in memory. Therefore, in the encoding (write) process, the DMC encoder is only an encoder to execute the encoding operations. However, in the decoding (read) process, this encoder is employed for computing the syndrome bits in the decoder. These clearly show how the area overhead of extra circuits can be substantially reduced.



Extra circuit	*En* signal		Function
	Read signal	Write signal	
Encoder	0	1	Encoding
	1	0	Compute syndrome bits

Fig. 4. 32-bit DMC decoder structure using ERT.

D. Limitations of DMC

Thus decimal matrix code maximizes the error detection and correction capability. But it results in decoding error when the following factors occur:

- When upsets occurs in both the rows.
- When the decimal integer sum value of both the original data and that of the flipped data is the same.
- When the bits in symbol 0 and symbol 2 are upset such that their decimal integer sum of information bit is equal to 2m- 1.

Moreover DMC requires large number of redundant bits to detect and correct errors. to secure a 32- bit information it requires 36 redundant bits namely 20 horizontal syndrome bit and 16 vertical syndrome bits. Thus the only drawback is the increase in bandwidth. To overcome this, the number of redundant bits used must be reduced. This redundant bit reduction should not affect the security of the data as well. To achieve these criteria a new code called the Party Matrix Code is used in which the number of redundant bit is reduced, meanwhile the security is also maintained.

IV. PROPOSED PARITY MATRIX CODING (PMC) METHOD

As their name suggest parity matrix codes are block code with parity check matrices that contains only a very small number of non—zero entries. It is sparseness of H which guarantees both a decoding complexity which increases only linearity with the code length and minimum distance which is also increases linearly with the code length. Aside from the requirement that H sparse, an parity matrix codes code itself is no different to any other block code. Indeed existing block codes can be successfully used with the parity matrix codes iterative decoding algorithm if they can be represented by a sparse parity-check matrix. Generally however, finding a sparse parity check matrix for an existing code is not practical. Instead parity matrix codes are designed by constructing a sparse parity-check matrix first and the determining a generator matrix for the code afterwards. The biggest difference between parity matrix codes and classical block code is how they are decoded.

A. Schematic of Fault-Tolerant Memory

Parity Matrix Codes (PMC) are used to reduce the large number of redundant bits for error detection and correction. This new technique uses parity algorithm (matrix multiplication and matrix addition) which is used to detect and correct multiple errors with less number of redundant bits compared to Decimal Matrix Codes (DMC). The PMC corrects maximum number of errors in the memories compared to DMC and also consumes less power, delay and area than the Decimal Matrix Code. Consider a 32 – bit word to which the Parity Matrix Code is applied and verified with the results of the Decimal Matrix Code.

The block diagram for a fault tolerant memory protected with PMC is as shown in Fig.5.

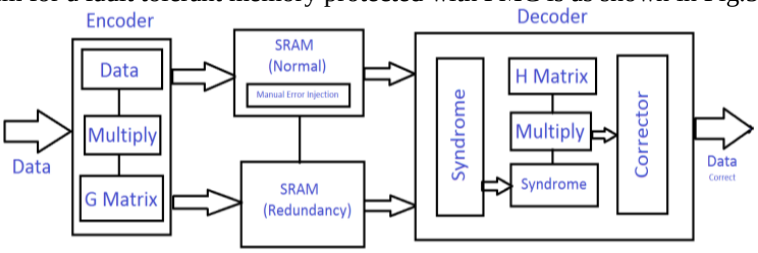


Fig.5. Fault tolerant memory protected with PMC

B. Proposed PMC encoder

Consider an n-bit information bit and similar to the decimal matrix code the 32-bit information is divided into k symbols of m bits each, i.e. if a 32-bit information is considered then divide the word into 16 symbols of 2 bits each. Then represent each data symbols in column vectors. Represent each parity bit with a column vector containing a 1 in the row corresponding to each data bit included in the computation and a zero in all other rows. Next create a generator matrix [G], by arranging the column vectors into a l×m matrix such that the columns are ordered to match their corresponding bits in a code word. Arranging the columns in any other order will just change the positions of bits in the code word. Finally the data is encoded by multiplying it with the generator matrix. This encoded data is stored in the memory. The stored data/information may be corrupted when exposed to harmful radiations like gamma rays, x-rays etc. Example of G-Matrix:

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

C. Proposed PMC Decoder

A parity check matrix [H] is constructed such that row 1 contains 1s in the position of the first parity bit and all of the data bits that are included in its parity calculation. Row 2 contains 1s in the position of the second parity bit and all of the data bits that are included in its parity calculation. Row 3 contains 1s in the position of the third parity bit and all of the data bits that are included in its parity calculation and so on. Multiplying the encoded data with the H will result in a syndrome generation. There are two useful proprieties of the syndrome. If the syndrome is all zeros, the encoded data is

error free. If the syndrome has a non-zero value, flipping the encoded bit that is in the position of the column in [H] that matches the syndrome will result in a valid code word. Thus the errors are detected and corrected using parity matrix code.

The H-Matrix corresponding to the G –Matrix given is:

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

D. Advantages of Proposed PMC

Number of redundant bits reduced when compared to DMC. Detection & correction of maximum number of errors in the memory resulting in enhanced memory reliability. Minimal area, power and delay overheads are achieved.

Table I Error Correction Capability Comparison

ECC codes	Data bits	No. of check bits	No. of errors corrected
PMC (Proposed)	32	32	16
DMC (Existing)	32	36	11

V. SIMULATION AND RESULTS

The modules are modelled using VHDL in Xilinx ISE Design Suite 8.1i and the simulation of the design is performed using Modelsim SE 6.3f to verify the functionality of the design. Here a structural model of fault tolerant memory protected with DMC and PMC are developed. The fault tolerant memory protected with DMC and PMC contains modules such as an encoder, memory information block, memory redundant block, reuse encoder, syndrome calculator and an error corrector.

A. DMC Simulation Results

1) Simulation of Encoder

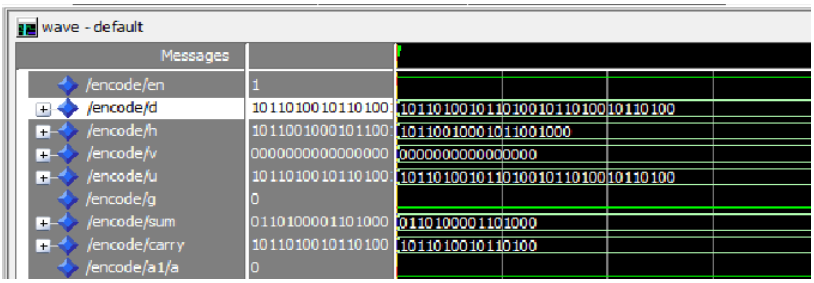


Fig.6. Encoder

For example if the 32 bit information bit is:

d= 10110100101101001011010010110100

Then the DMC logical organization is as follows:

1011 0100 1011 0100

1011 0100 1011 0100

At the encoder when the information bit (d) is applied, horizontal redundant bit (h), vertical redundant bit (v) and information bits (u) is obtained. The simulated waveform is as shown in fig.6. Hence,

h= 10110 01000 10110 01000

v= 0000 0000 0000 0000

u=10110100101101001011010010110100

2) Simulation of SRAM Information Block

At the SRAM information block, the information bit (d) is applied and the flipped information bit (dd) is obtained. The simulated waveform is as shown in fig.7. Hence,

dd = 10110100101101001011011100001101

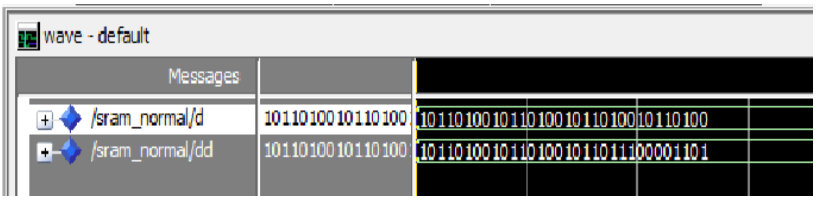


Fig. 7.SRAM Information

3) Simulation of SRAM redundancy block

At the SRAM redundant block, both horizontal redundant bit(h) and vertical redundant bit (v) are applied and SRAM horizontal redundant bit(mem h) and vertical redundant bit (mem v) are obtained. The simulated waveform is as shown in fig.8. Hence,

mem h=10110 01000 10110 01000

mem v= 0000 0000 0000 0000

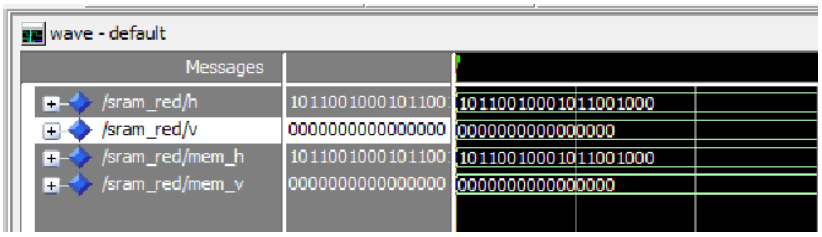


Fig.8. SRAM redundancy

4) Simulation of Reuse Encoder

At the reuse encoder part of the decoder the received horizontal redundant bit (hnot) and received vertical redundant bit (vnot) is calculated from the flipped information bit (dd). The simulated waveform is as shown in fig.9. Hence,

hnot= 10110 01000 01011 10100

vnot= 0000 0011 1011 1001

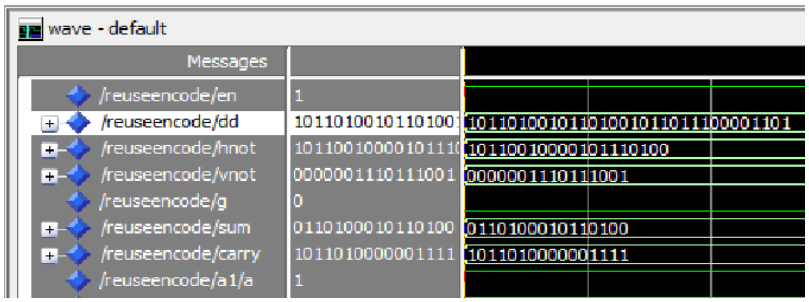


Fig.9. Reuse Encoder

5) Simulation of Syndrome Calculator

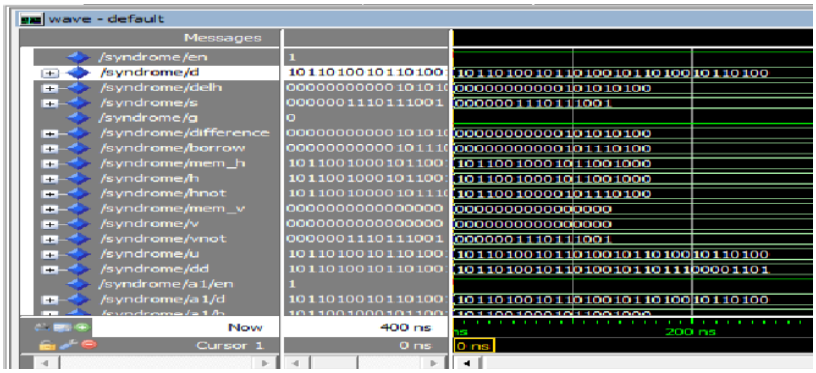


Fig.10. Syndrome Calculator

At the syndrome calculator section, the horizontal syndrome bits (delh) and the vertical syndrome bits (s) are calculated from mem h, mem v, hnot and vnot. The simulated waveform is as shown in fig.10. Hence,

delh = 00000 00000 01010 10100; s = 0000 0011 1011 1001

6) Simulation of Error Corrector

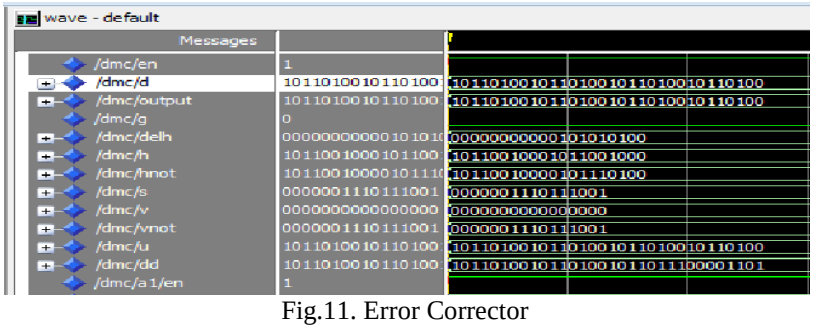


Fig.11. Error Corrector

At the error corrector section the output information bit (output) is obtained as equivalent to the information bit (d) by the XOR operation of flipped information bit (dd) and that of the vertical syndrome bit(s).The simulated waveform is as shown in fig.11. Hence,

Output= 10110100101101001011010010110100

B. PMC Simulation Results

1) Simulation of Encoder

32 bit information is applied to the encoder. This information bit is divided into 2 bits of 16 blocks. These 2 bit of 16 blocks are encoded with G-matrix. The simulated waveform is as shown in fig.12.

input= 1011 0100 1011 0100 1011 0100 1011 0100

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

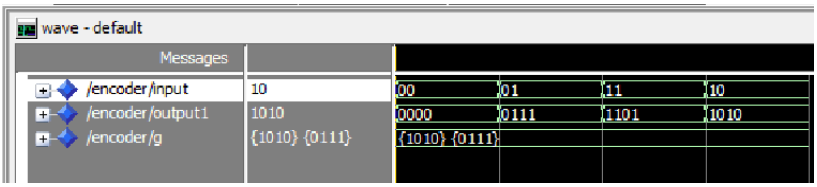


Fig.12. Encoder

2) Simulation of Memory Normal

At the memory normal block, the information bit (input) is applied and the flipped information bit (mem_d) is obtained. The simulated waveform is as shown in fig.13. Hence,

input= 1011 0100 1011 0100 1011 0100 1011 0100

mem_d=0001 1110 1101 1110 0001 1110 0001 1110

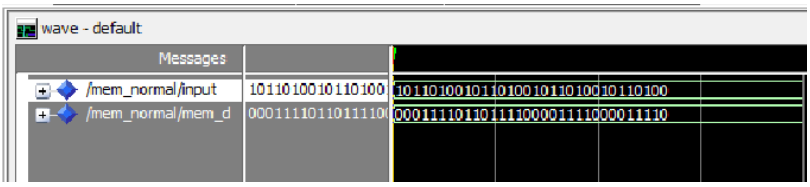


Fig.13. Memory Normal

3) Simulation of Memory Redundancy

At the memory redundant block, the information bits (input) multiplied with G-Matrix is stored. The simulated waveform is as shown in fig.14. Hence,

output2=1001 1100 1001 1100 1001 1100 1001 1100

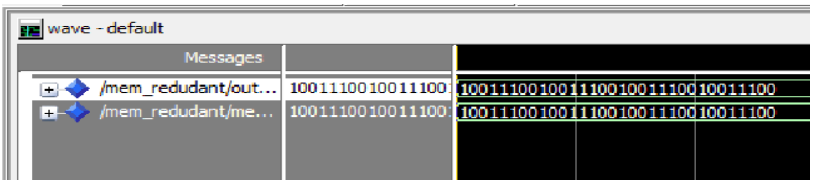


Fig.14. Memory Redundancy

4) Simulation of Syndrome Calculator

Here the information bits from memory normal(mem_d) and memory redundancy (output2) are concatenated together to form 64 bit data. The simulated waveform is as shown in fig.15. Hence,

Syndrome=0010 0101 1111 1000 1110 0101 1111 1000

0010 0101 1111 1000 0010 0101 1111 1000

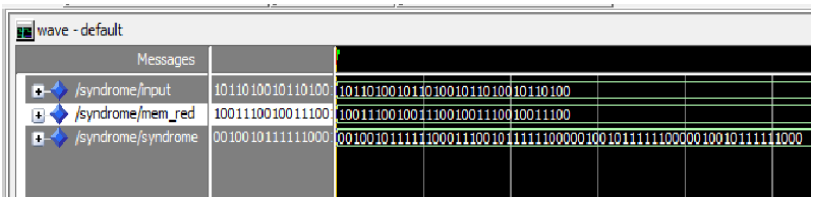


Fig.15. Syndrome Calculator

5) Simulation of Decoder and Final PMC error corrector

At the decoder the syndrome bits are multiplied with H-matrix to get the corrected output. a1 & a2 are logic signals, when the corresponding syndrome bit is corrupt it turns high. Thus a XOR operation gives the correct data as output from memory. The simulated waveform is as shown in fig.16 and fig.17. Hence,

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

output= 1011 0100 1011 0100 1011 0100 1011 0100
output3= 1011 0100 1011 0100 1011 0100 1011 0100

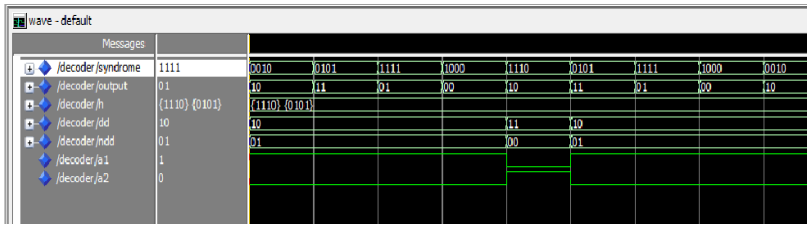


Fig.16.Decoder

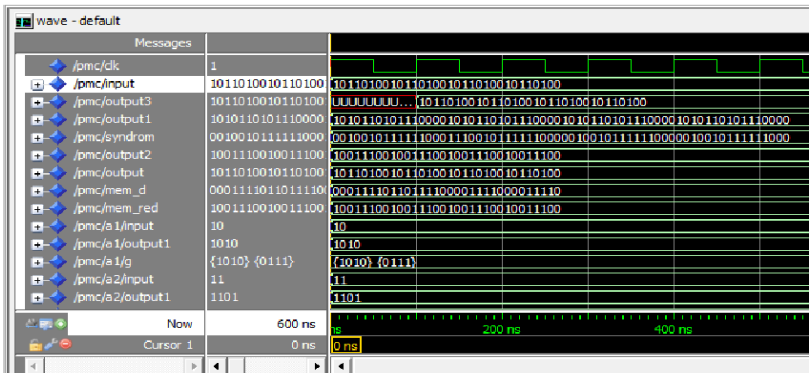


Fig.17. Corrected output using PMC

VI. OVERHEAD ANALYSIS

The proposed PMC and existing DMC is compared to well-known codes such as the Hamming Code, MCs, and punctured difference set (PDS) codes in terms of Area, Delay and Power.

A) Area Analysis

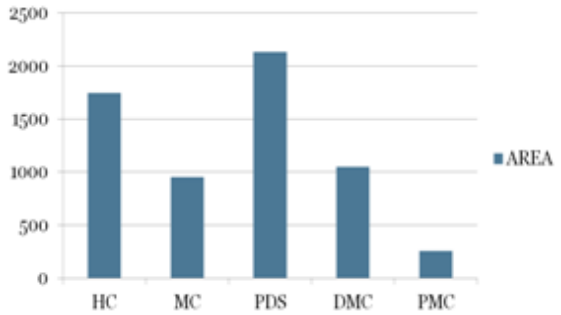


Fig.18.Area Comparison

Area is calculated by using equivalent gate count and the comparison result in shown in fig.18. The table II gives the area of various codes. It shows that PMC requires minimal area. There is 76% area reduction attained by using proposed PMC when compared to DMC.

Table II Area Analysis of different ECC

ERROR CORRECTING CODE	AREA (EQUIVALENT GATE COUNT)
Hamming Code (HC)	1745
Matrix Code (MC)	955
Punctured Difference Set (PDS)	2133
Decimal Matrix Code (DMC)	1052
Parity Matrix Code (PMC)	256

B) Delay Analysis

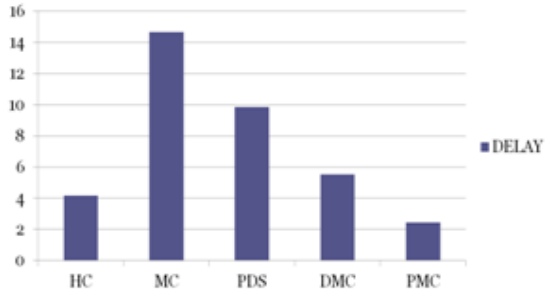


Fig.19.Delay Comparison

Delay is calculated in nanoseconds(ns) and the comparison result in shown in fig.19. The table III gives the delay of various codes. It shows that PMC requires minimal delay. We see that 56% faster process acquired by using proposed PMC when compared to DMC.

Table III Delay Analysis of different ECC

ERROR CORRECTING CODE	DELAY (ns)
Hamming Code (HC)	4.156
Matrix Code (MC)	14.657
Punctured Difference Set (PDS)	9.838
Decimal Matrix Code (DMC)	5.522
Parity Matrix Code (PMC)	2.441

C) Power Analysis

Power is calculated in milliWatt (mW) and the comparison result in shown in fig.20. The table IV gives the power of various codes. It shows that PMC requires minimal power. Here power is reduced by several mW using proposed PMC when compared to DMC.

Table IV Power Analysis of different ECC

ERROR CORRECTING CODE	POWER (mW)
Hamming Code (HC)	44
Matrix Code (MC)	45
Punctured Difference Set (PDS)	43
Decimal Matrix Code (DMC)	32
Parity Matrix Code (PMC)	28

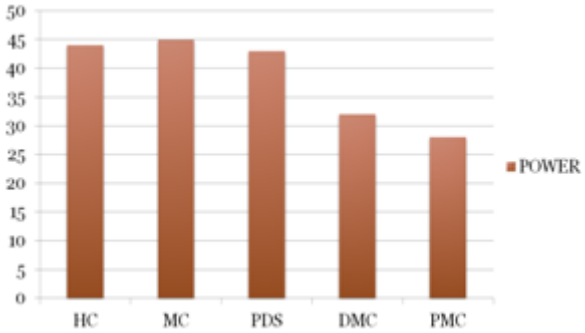


Fig.20. Power Comparison

The graphs show that PMC utilizes less area, power and delay when compared to existing DMC as well as previous methods. This indicates that the memory with the proposed scheme performs faster than other codes. Thus enhanced memory reliability at minimal overheads is attained by this new scheme.

VII. CONCLUSION

A novel per-word DMC exists to assure the reliability of memory. The DMC protection code utilized decimal algorithm to detect errors, so that more errors were detected and corrected. Though DMC has less complexity of encoding and decoding circuits and less area, power, delay overheads, its error correction capability is limited and it requires more number of check bits for memory protection. Thus parity based matrix codes is proposed to increase the

error handling capability with less redundant bits.It uses Matrix Multiplication and Addition in encoding and Decoding using Parity Check Matrix (H) and its Generator Matrix (G). Thus the drawback of the existing system is solved in this method. With significant reductions in both area and delay the proposed codes find its application in protection of registers in circuits, high speed memories/ caches, where the area and delay of the encoder and decoder can be a more important issue. As a future work, application based design can be done.

ACKNOWLEDGEMENT

The authors acknowledge SCMS School of Engineering and Technology for providing platform to discuss the paper and its library resources for providing reference books and journals to conduct research in the aforementioned paper.

REFERENCES

[1] K. Pagiamtzis and A. Sheikholeslami, "Content addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, Mar. 2003.

[2] D. Radaelli, H. Puchner, S. Wong, and S. Daniel, "Investigation of multi-bit upsets in a 150 nm technology SRAM device," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2433–2437, Dec. 2005.

[3] G. Neuberger, D. L. Kastensmidt, and R. Reis, "An automatic technique for optimizing Reed-Solomon codes to improve fault tolerance in memories," *IEEE Design Test Comput.*, vol. 22, no. 1, pp. 50–58, Jan.–Feb. 2005.

[4] C. Argyrides and D. K. Pradhan, "Improved decoding algorithm for high reliable reed muller coding," in *Proc. IEEE Int. Syst. On Chip Conf.*,Sep. 2007, pp. 95–98.

[5] Y. Yahagi, H. Yamaguchi, E. Ibe, H. Kameyama, M. Sato, T. Akioka, and S. Yamamoto, "A novel feature of neutron-induced multi-cell upsets in 130 and 180 nm SRAMs," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 4, pp. 1030–1036, Aug. 2007.

[6] R. Naseer and J. Draper, "Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs," in *Proc. 34th Eur. Solid-State Circuits*, Sep. 2008, pp. 222–225.

[7] S. Baeg, S. Wen, and R. Wong, "Interleaving distance selection with a soft error failure model," *IEEE Trans. Nucl. Sci.*, vol. 56, no. 4, pp. 2111–2118, Aug. 2009.

[8] P. Reviriego and J. A. Maestro, "Efficient error detection codes for multiple-bit upset correction in SRAMs with BICS," *ACM Trans. Design Autom. Electron. Syst.*, vol. 14, no. 1, pp. 18:1–18:10, Jan. 2009.

[9] C. A. Argyrides, C. A. Lisboa, D. K. Pradhan, and L. Carro, "Single element correction in sorting algorithms with minimum delay overhead," in *Proc. IEEE Latin Amer. Test Workshop*, Mar. 2009, pp. 652–657.

[10] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron induced soft error in SRAMs from an 250 nm to a 22 nm design rule," *IEEE Trans. Electron Devices*, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[11] S. Baeg, S. Wen, and R. Wong, "Minimizing soft errors in TCAM devices: A probabilistic approach to determining scrubbing intervals," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 814–822, Apr. 2010.

[12] C. A. Argyrides, P. Reviriego, D. K. Pradhan, and J. A. Maestro, "Matrix-based codes for adjacent error correction," *IEEE Trans. Nucl. Sci.*, vol. 57, no. 4, pp. 2106–2111, Aug. 2010.

[13] M. Zhu, L. Y. Xiao, L. L. Song, Y. J. Zhang, and H. W. Luo, "New mix codes for multiple bit upsets mitigation in fault-secure memories," *Microelectron. J.*, vol. 42, no. 3, pp. 553–561, Mar. 2011.

[14] C. Argyrides, R. Chipana, F. Vargas, and D. K. Pradhan, "Reliability analysis of H-tree random access memories implemented with built in current sensors and parity codes for multiple bit upset correction," *IEEE Trans. Rel.*, vol. 60, no. 3, pp. 528–537, Sep. 2011.

[15] C. Argyrides, D. K. Pradhan, and T. Kocak, "Matrix codes for reliable and cost efficient memory chips," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 3, pp. 420–428, Mar. 2011.

[16] P. Reviriego, M. Flanagan, and J. A. Maestro, "A (64, 45) triple error correction code for memory applications," *IEEE Trans. Device Mater. Rel.*, vol. 12, no. 1, pp. 101–106, Mar. 2012.

[17] 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, Aug. 2012.

[18] M. Imran, Z. Al-Ars, G. N. Gaydadjiev, "Improving Soft Error Correction Capability of 4-D Parity Codes", *IEEE transaction*, Vol. 4, Issue 2, January 2012, pp 233-238.

[19] Y. Bentoutou, "Program Memories Error Detection and Correction On- Board Earth Observation Satellites", *World Academy of science, Engineering and Technology* 66 2012.

[20] Heesung Lee, Joonkyung Sung, and Euntai Kim, "Reducing Power in Error Correcting Code using Genetic Algorithm", *World Academy of Science, Engineering and Technology*, 2013.

[21] M.Kishani, H. R. Zarandi, H. Pedram, A Tajary, M. Raji, B. Ghavami, "Horizontal-Vertical Diagonal Error Detecting And Correcting Code To Protect Against With Soft Errors" *Springer Science*, Vol. 15, Issue No. 3-4, May 2013, Pp 289-310.

[22] S. Sharma, Vijay Kumar, "An HVD Based Error Detection and Correction of Soft Errors in Semiconductor Memories Used for Space Application", *International conference on devices, circuits and systems (ICDCS)*, March 2014, pp. 563-56.

