International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Implementation of Decimal Matrix Code For Multiple Cell Upsets in Memory

Shwetha N [1], Shambhavi S[2]

[1, 2] Department of E&C, Kalpataru Institute of Technology, Tiptur, Karnataka, India

Abstract: ***Transient multiple cell upsets (MCUs) are becoming major issues in the reliability of memories exposed to radiation environment. To prevent MCUs from causing data corruption ,more complex error correction codes (ECCs) are widely used to protect memory, but the main problem is that they would Require higher delay overhead. Recently, matrix codes(MCs) based on Hamming codes have been proposed for memory protection. The main issue is that they are double error correction codes and the error correction capabilities are not improved in all cases. In this paper,novel decimal matrix code (DMC) based on divide-symbol is proposed to enhance memory reliability with lower delay overhead. The proposed DMC utilizes decimal algorithm to obtain the maximum error detection capability. Moreover, the encoder-reuse technique (ERT) is proposed to minimize the area overhead of extra circuits without disturbing the whole encoding and decoding processes. ERT uses DMC encoder itself to be part of the decoder. The proposed DMC is compared to well-known codes such as the existing Hamming, MCs, and punctured difference set (PDS) codes.***

Keywords: Decimal algorithm, error correction codes (ECCs), mean time to failure (MTTF), memory, multiple cells upsets (MCUs).

1. Introduction

AS CMOS technology scales down to nano scale 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[1].

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[2]. 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 [3]-[6]. For example, the Bose-Chaudhuri-Hocquenghem codes [7], Reed-Solomon codes[8] and punctured difference set (PDS) codes [9] 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 [10], which rearrage cells in the physical words. However, interleaving technique may not be practically used in content-addresable memory (CAM), because of the tight coupling of hardwear structures from both cells and comparison circuit structure [11], [12].

Built-in current sensors (BICS) are proposed to assist with-error correction and double-error detection codes to provide protection against MCUs [13], [14]. However, this technique can only correct two errors in a word. More recently, in[15], 2-D matrix codes(MCs) are proposed to efficiently correct MCUs per word with a low decoding delay, in which one word is divided into multiple rows and multipla columns in logical. The bits per row are protected by Hamming code, while parity code is added in each column. For the MC [15] 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 softwear level. It uses addition of iteger values to detect and correct soft errors. The results obtained have shown that this approach have a lower delay overhead over other codes. In this paper, novel decimal matrix code(DMC) based on divide-symbol is proposed to provide enhanced memory reliability. The proposed DMC utilizes decimal algorithm (decimal integer addition and decimal integer subtraction) to detect errors. The advantage of using decimal algorithm is that the error detection capability is maximized so that the reliability of memory is enhanced. Besides, the encoder-reuse technique (ERT) is proposed to minimize the area overhead of extra circuits (encoder and decoder) without disturbing the whole encoding and decoding process, because ERT uses DMC encoder itself to be part of the decoder. This paper is dividede into the folloeing sections. The proposed DMC is intruduced and its encoder and decoder circuits are present in Section II. This section also illustrates the limits of simple binary error detection and the advantage of decimal error detection with simple examples. The reliability and overheads analysis of the proposed code are analzed in Section III.

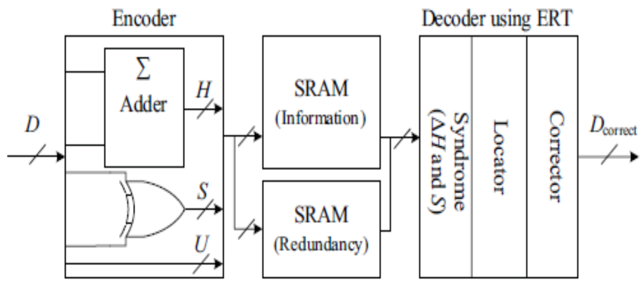


Figure 1: Proposed schematic of fault-tolerant memory protected with DMC.

Finally, some conclusions of this paper are discussed and shared in section IV.

2. Proposed DMC

In this section, DMC is proposed to assure reliability in the presence overheads, and a 32-bit word is encoded and decoded as an example based on the proposed techniques.

A. Proposed Schematic of Fault-Tolerant Memory

The proposed 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 redundants 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 proposed to reduce the area overhead of extra circuits and will be introduced in the following sections.

B. Proposed DMC Encoder

In the proposed 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×m), and these symbols are arranged in a k1×k2 2-D matrix (k=k1×k2, where the values oa k1 and k2 represent the numbers of rows and columns in the logical matrix respectively). Second the horizontal 8, only 1-bit error can be corrected and the redundant bits H are priduced by performing 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 proposed DMC scheme, we take a 32-bit word as an example, as shown in Fig.2. The cells from D_0TO D_{31} 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. H_0-H_{19} are horizontal check bits; V_0 through V_{15} 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=2×2 and m=8, only 1-bit error can be corrected and the number of redundant bits is 40. When k=4×4 and m=2, 3-bit errors can be corrected and the number of redundant bits is reduced to 32. However, when k=2×4, and m=4, the maximum correction capability is upto 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=2×4, and m=4, are utilized to construct DMC.

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

$$H_4H_3H_2H_1H_0=D_3D_2D_1D_0+D_{11}D_{10}D_9D_8 \quad (1)$$

$$H_9H_8H_7H_6H_5=D_7D_6D_5D_4+D_{15}D_{14}D_{13}D_{12} \quad (2)$$

And similarly for the horizontal redundant bits $H_{14}H_{13}H_{12}H_{11}H_{10}$ and $H_{19}H_{18}H_{17}H_{16}H_{15}$ Where "+"represents decimal integer addition

For the vertical redundant bits V, we have

$$V_0=D_0{'}\oplus D_{16} \quad (3)$$

$$V_1=D_1{'}\oplus D_{17} \quad (4)$$

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, H_{19}-H_0 are horizontal redundant bits, V_{15}-V_0 are vertical redundant bits, and the remaining bits U_{31}-U_0 are the information bits which are directly copied from D_{31}-D_0. The enable signal En will be expalined in the next section.

C. Proposed DMC decoder

To obtain a word being corrected, the decoding process is required. For example, first the received redundant bits $H_4H_3H_2H_1H_0{'}$ and $V_0{'}$-$V_3{'}$ are generated by the received information bits. The horizontal syndrome bits $\Delta H_4H_3H_2H_1H_0$ and the vertical syndrome bits S_3-S_0 can be calculated as follows:

$$\Delta H_4H_3H_2H_1H_0= H_4H_3H_2H_1H_0{'} - H_4H_3H_2H_1H_0 \quad (5)$$

$$S_0=V_0{'}\oplus V_0$$

And similarly for the rest vertical syndrome bits, where "-" represents decimal integer subtraction. When $\Delta H_4H_3H_2H_1H_0$ and S_3-S_0 are equal to zero, the stored code word has original information bits in symbol 0 where no errors correct. When $\Delta H_4H_3H_2H_1H_0$ and S_3-S_0 are non zero, the induced errors (the number of errors is 4 in

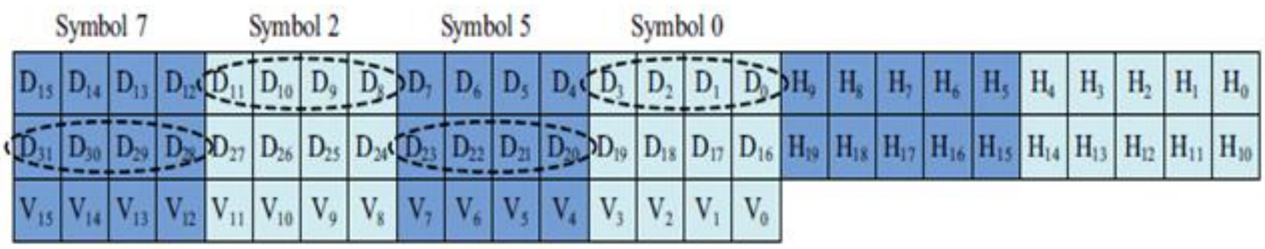

Symbol 7
Symbol 2
Symbol 5
Symbol 0
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 H9 H8 H7 H6 H5 H4 H3 H2 H1 H0
D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 H19 H18 H17 H16 H15 H14 H13 H12 H11 H10
V15 V14 V13 V12 V11 V10 V9 V8 V7 V6 V5 V4 V3 V2 V1 V0

Figure 2: 32-bits DMC logical organization (k=2×4 and m=4), here, each symbol is regarded as a decimal integer

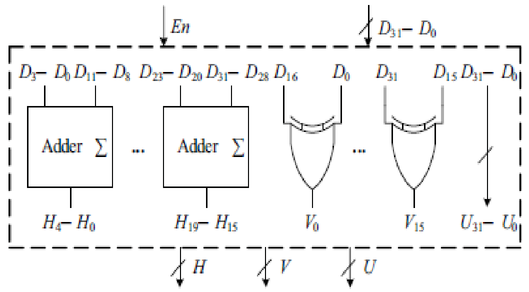
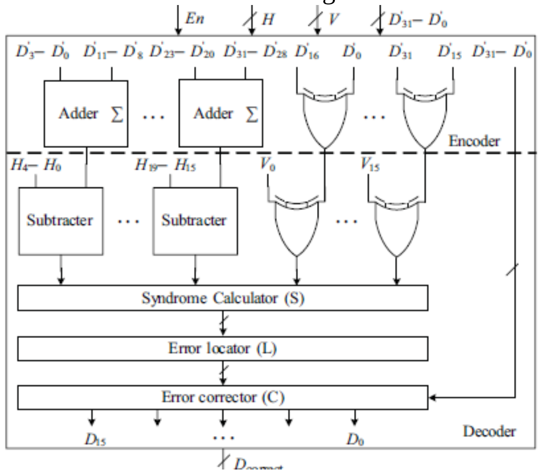


Figure 3: 32-bit DMC encoder structure using multi bit adders and XOR gates



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

Figure 4: 32-bit DMC decoder structure using ERT.

This case) are detected and located in symbol 0, and then these errors can be corrected by

$$D_{0correct}=D_0 \oplus S_0 \,. \quad (7)$$

The proposed DMC decoder is depicted in Fig.4. which is made up of the following submodules, and each excutes 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

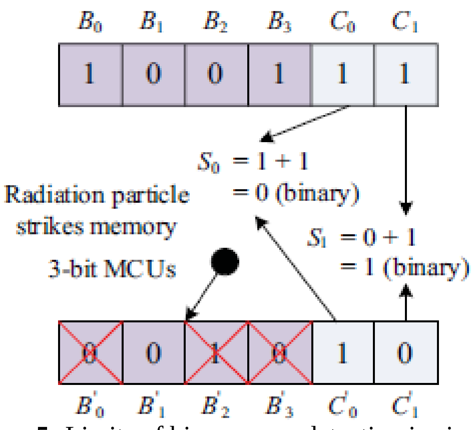


Figure 5: Limits of binary error detection in simple binary operations

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 error can be corrected by inverting the values of error bits. In the proposed scheme, 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. Thereore, 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.

D. Limits Of Binary Error Detection

For the proposed binart error detection technique in [13], aithough it requires low redundant bits, its error detection capability is limited. The main reason for this is that its error detection mechanism is based on binary. We illustrate the limits of this simple binary error detection [13] using a simple example. Let us suppose that the bits B_3, B_2 , B_1, and B_0 are original information bits and the bits C_0 and C_1 are redundant bits shown in Fig.5. The bits C_0 and C_1 are obtained using the binary algorithm

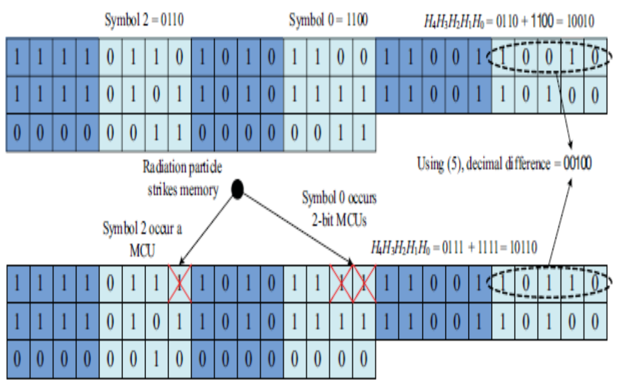

Symbol 2 = 0110
Symbol 0 = 1100
H4H3H2H1H0 = 0110 + 1100 = 10010
Radiation particle strikes memory
Using (5), decimal difference = 00100
Symbol 0 occurs 2-bit MCUs
Symbol 2 occur a MCU
H4H3H2H1H0 = 0111 + 1111 = 10110

Figure 6: Advantage of decimal error detection. Using decimal algorithm. $H_4H_3H_2H_1H_0$ will not be “0” (decimal). This represents that MCUs can be detected and corrected so that the decoding error can be avoided.

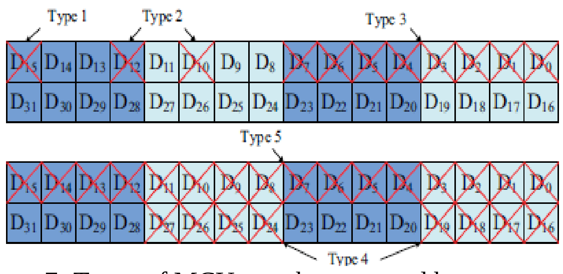

Type 1
Type 2
Type 3
Type 5
Type 4
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16

Figure 7: Types of MCUs can be corrected by our proposed DMC. Type 1 is a single error. Type 2 is an inconsecutive symbols. Type 3 is a consecutive error in two consecutive symbols, type 4 is an inconsecutive error in two

inconsecutive symbols, and type 5 is a consecutive error in four consecutive symbols.

(XOR)

$$C_0=B_0 \oplus B_2=1 \oplus 0=1 \quad (8)$$
$$C_1=B_1 \oplus B_3=0 \oplus 1=1. \quad (9)$$

Then assume now that MCUs occur in bits B_3, B_2, and B_0 (i.e., $B_3^{'}=0$, $B_2^{'}=1$, and $B_0^{'}=0$). The received redundant bits $C_0^{'}$ and $C_1^{'}$ are computed

$$C_0^{'}=B_0^{'} \oplus B_2^{'}=0 \oplus 1=1 \quad (10)$$
$$C_1^{'}=B_1^{'} \oplus B_3^{'}=0 \oplus 0=0 \quad (11)$$

In order to detect these errors, the syndrome bits S_0 and S_1 are obtained

$$S_0= C_0^{'} \oplus C_0=1 \oplus 1=0 \quad (12)$$
$$S_1= C_1^{'} \oplus C_1=0 \oplus 1=1. \quad (13)$$

These results mean that error bits B_2 and B_0 are wrongly regarded as the original bits so that these two error bits are not corrected. This example illustrates that for this simple binary operation [13], the number of even bit errors cannot be detected.

E. Advantage of Decimal Error Detection

In the previous discussion, it has been shown that error detection [13] based on binary algorithm can only detect a finite number of error. However, when the decimal algorithm is used to detect error, these errors can be detected so that the decoding error can be avoided. The reason is that the operation mechanism of decimal algorithm is different from that of binary. The detection procedure of decimal error detection using the proposed structure shown in Fig. 2 is fully described in Fig. 6. First of all, the horizontal redundant bits $H_4H_3H_2H_1H_0$ are obtained from the original information bits in symbol 0 and 2 according to (1)

$$\begin{aligned} H_4H_3H_2H_1H_0 &= D_3D_2D_1D_0+D_{11}D_{10}D_9D_8 \\ &=1100+0110 \\ &=10010. \end{aligned} \quad (14)$$

When MCUs occur in symbol 0 and symbol 2, i.e., the bits in symbol 0 are upset to “1111” from “1100” ($D_3D_2D_1D_0^{'}$=1111) and the bits in symbol 2 are upset to “0111” from “0110” ($D_{11}D_{10}D_9D_8^{'}$=0111). During the decoding process, the received horizontal redundant bits $H_4H_3H_2H_1H_0$ are first computed. As follows:

$$\begin{aligned} H_4H_3H_2H_1H_0^{'} &= D_3D_2D_1D_0^{'}+D_{11}D_{10}D_9D_8^{'} \\ &=0111+1111 \\ &=10110 \end{aligned} \quad (15)$$

Then, the horizontal syndrome bits $\Delta H_4H_3H_2H_1H_0$ can be obtained using decimal integer subtraction

$$\begin{aligned} \Delta H_4H_3H_2H_1H_0 &= H_4H_3H_2H_1H_0^{'} - H_4H_3H_2H_1H_0 \\ &=10110 - 10010 \\ &=00100. \end{aligned} \quad (16)$$

The decimal value of $\Delta H_4H_3H_2H_1H_0$ is not “0” which represents that errors are detected and located in symbol 0 or symbol 2. Subsequently, the precise location of the bits that were flipped can be located by using the vertical syndrome bits S_3-S_0 and S_{11}-S_8. Finally, all these errors can be corrected by (7). Therefore, based on decimal algorithm, the proposed technique has higher tolerance capability for protecting memory against MCUs.

As a result, it is posssible that all single and double errors and any types of multiple errors per row can be corrected by the proposed technique no matter whether these errors are consecutive or inconsecutive in Fig. 7. The proposed DMC

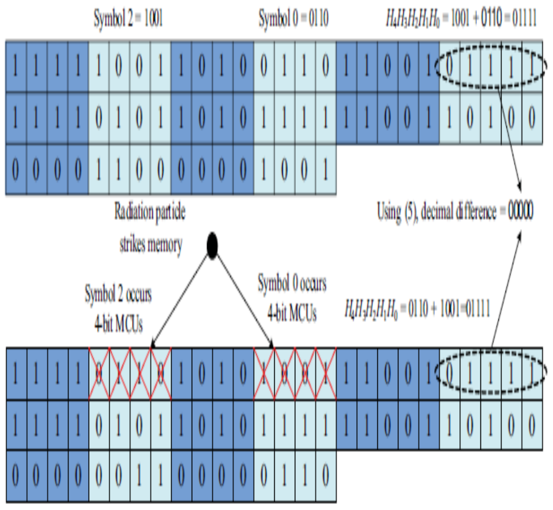

Symbol 2 = 1001
Symbol 0 = 0110
H4H3H2H1H0 = 1001 + 0110 = 01111
Radiation particle strikes memory
Using (5), decimal difference = 00000
Symbol 2 occurs 4-bit MCUs
Symbol 0 occurs 4-bit MCUs
H4'H3'H2'H1'H0' = 0110 + 1001 = 01111

Figure 8: Error type cannot be corrected by our proposed DMC. The main reason is that $H_4 H_3H_2H_1H_0$ will be “0” (decimal). Note that even though 7-bit errors occur in symbols 0 and 2 simultaneously, the decoding error can be refused

Can easily correct upsets of type 1, 2, and 3, because these are the essential property of DMC: any types of single error and multiple error corrections in two consecutive symbols. Upsets of types 4 and 5 introduced in Fig. 7 are also corrected because the multiple errors per row can be detected by the horizontal syndrome bits (see Fig. 6). These show that the proposed technique is an attactive option to protect memories from large MCUs. However, for the upsets of type 4 and 5, it is important to recognize that it can result in decoding error when the fpllowing prerequisite factors are achieved simultaneously (this error is typical of its kind).

1) The decimal integer sum of information bits in symbols 0 and 2 is eual to 2^m-1.
2) All the bits in symbol 0 and 2 are upset.

The more detailed explanation is shown in Fig. 8. Assuming that these two factors have been achieved, according to the encoding and decoding processes of DMC, $H_4H_3H_2H_1H_0$, and $H_4H_3H_2H_1H_0^{'}$ are computed, as follows:

$$\begin{aligned} H_4H_3H_2H_1H_0 &= D_3D_2D_1D_0+D_{11}D_{10}D_9D_8 \\ &=0110+1001 \\ &=01111 \end{aligned} \quad (17)$$

$$\begin{aligned} H_4H_3H_2H_1H_0^{'} &= D_3D_2D_1D_0^{'}+D_{11}D_{10}D_9D_8^{'} \\ &=1001+0110 \\ &=01111 \end{aligned} \quad (18)$$

Then the horizontal syndrome bits $\Delta H_4H_3H_2H_1H_0$ can be obtained

$$\Delta H_4H_3H_2H_1H_0 = H_4H_3H_2H_1H_0' - H_4H_3H_2H_1H_0 = 01111-01111 = 00000 \quad (19)$$

This result means that no errors occur in symbols 0 and 2 and memory will suffer a failure. However, this case is rare. For example, when m=4, the probability of decoding errors is

$$P\Delta H=0=4\times (1/2^4)^2\times P_{MCU8}\approx 0001. \quad (20)$$

If m=8

$$P\Delta H=0=4\times (1/2^4)^2\times P_{MCU8}\approx 0.000001 \quad (21)$$

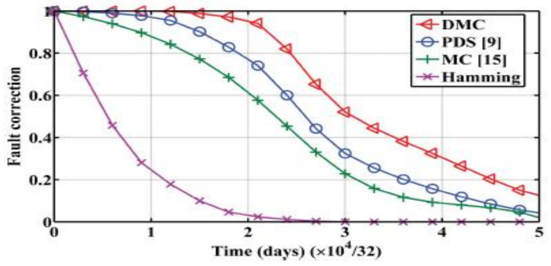


Figure 9: J(S)s versus time of different protection codes (M=32)

P_{MCU8} represents the probability of eight upsets in a given word, and similarly for $P_{MCU16.}$ Moreover, according to the radiation experiments in [1], [2], it can be obtained that the word in a memory usually has a limited number of consecutive errors and the interval of these errors is not more than three bits. Therefore, this should not be an issue.

3. Reliability and Overheads Analysis

In this section, the proposed DMC has been implemented in HDL, simulated with modelsim and tested for functionality by given various inputs. The encoder and decoder have been synthesized by the synopsys design compiler in the SMIC 0.18μm technology. The area, power, and critical path delay of extra circuits have been obtained. For fair comparisions, Hamming, PDS [9], and MC [15] are used for references. Here. The usage of (64, 45) PDS is a triple-error correction code [9] and its information bits is shorted to 32 bits from 45 bits.

A. Fault Injection

The correction coverage of PDS [9], MC [15], Hamming, and the proposed DMC codes are obtained from one million experiments. The results of coverage are shown in table I. It can be seen that proposed DMC have superior protection level compared with other codes. These results show how our proposed technique provides single- and double-error correction, but can also provide effective tolerance capabilities against large MCUs that exceed the performance of other codes.

TABLE I

CORRECTION FOR COVERAGE (32-bit)

ECC Codes	The Number of Errors in a Word															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DMC (%)	100	100	100	100	100	92.6	84.7	76.0	66.7	60.9	54.5	47.7	40.0	31.6	22.3	11.8
PDS [9] (%)	100	100	100	0.8	0	0	0	0	0	0	0	0	0	0	0	0
MC [15] (%)	100	100	76.4	54.3	35.1	14.2	6.7	0.6	0	0	0	0	0	0	0	0
Hamming (%)	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

B. Reliability Estimation

The reliability of our proposed code can be analyzed in terms of the mean time to failure (MTTF). It is asuumed that MCUs arrive at memories following a poisson distribution [19]. For one word, the correctable probability R(S) after S radiation events can be given by [14], [15]

$$R(S)= \sum_{i+j+\ldots+z\leq T} P_i^1 P_j^2 \ldots P_Z^S \quad (22)$$

Where T is the maximum number of errors and P_Z^S is the correctable probability upon the reception of radiation event S which causes z errors. For a memory with M words, the correctable probability J(S) after S radiation events can be given by

$$J(S) = \sum_{a+b+\cdots+e=S} \frac{C_M^x}{M^x} R_a^1 R_b^2 \cdots R_e^x \quad (23)$$

Where x(x≤S) is the number of words affected by radiation events. C_M^x is the selection of x from M words in memory. And R_e^x represents the correctable probability when e radiation events affect x words.

If we assume that the word number M is 32 and the correctable probability P_z^s can be obtained from Table I. the correctable probabilities J(S) s of different protection codes have been shown in Fig. 9. It can be seen that the correctable probability J(S) of the proposed scheme is larger than other codes.

Then the MTTF can be given by (24), which is the integral of function (23)

$$\mathrm{MTTF} = \int_0^{\infty} J(t)dt. \quad (24)$$

Table II shows MTTFs of different event arrival rate γ. I n this table, we can see that the proposed scheme has higher MTTF by more than 122.6% 154.6%, and 452.9% compared to PDS [9], MC [15], and Hamming, respectively.

In general cases, for the proposed technique, it can be inferred that the larger the word widths, the higher the tolerance capability and the better the reliability. For example, for a 64-bit word, when k=2×4 and m=8 the correction capability of the proposed technique is up to 9 bits.

TABLE II
MTTF ($M = 32$)

λ (Upsets/bit per Day)	DMC	PDS [9]	MC [15]	Hamming
10^{-4}	1121.9	915.0	725.6	247.7
10^{-5}	11218.8	9150.3	7256.5	2477.4

TABLE III
AREA, POWER, AND DELAY ANALYSIS OF ENCODER AND DECODER

ECC Codes	Area		Power		Delay	
	μm^2	%	mw	%	ns	%
DMC	41572.6	100	10.8	100	4.9	100
PDS* [9]	486778.1	1170.9	221.1	2047.2	18.7	381.6
MC [15]	77933.7	187.5	24.7	228.7	7.1	144.9
Hamming	58409.4	140.5	20.5	189.8	6.7	136.7

*Using parallel decoder instead of serial decoder for fair comparison

C. Overhead Analysis

For each protection code, area, power, and delay overheads of encoder and decoder have been shown in table III. From table III, we can observe that the proposed MC has a significant reduction compared with other codes. The area and power overheads of PDS are 1170.9%, 2047.2% of the proposed scheme, respectively. The delay overhead of DMC is 26.2%, 69.0%, and 73.1% of PDS [9], MC [15], and Hamming respectively. This indicates that the memory with the proposed scheme performs faster than other codes. Different decoding algorithms could result in different overheads. The decoding algorithms PDS [9] is more complex than that of other codes; thus, it has maximum area, power, and delay overheads. However, for the proposed DMC, its decoding algorithm is quite simple so that the overheads are minimal.

The issue is that the proposed technique requires more redundant bits compared with other codes. The redundant bits of these protection codes are shown in Table IV. Where a coding efficiency β is used to evaluate the area overhead of memory cell

$$\beta = \frac{\text{Redundant bits}}{\text{Redundant bits} + \text{Information bits}}. \qquad (25)$$

If the valueof β is small, the code needs lower memory cell overheads. From this table, we can see that hamming

Table 4: Redundant Bits(32-bit)

ECC	Information Bits	Redundant Bits	β	Note
DMC	32	36	52.9%	$k = 2 \times 4, m = 4$
DMC	32	32	50.0%	$k = 4 \times 4, m = 2$
PDS [9]	32	19	37.3%	Shorting and puncturing
MC [15]	32	28	46.7%	Correction capability is 2
Hamming	32	7	17.9%	Correction capability is 1

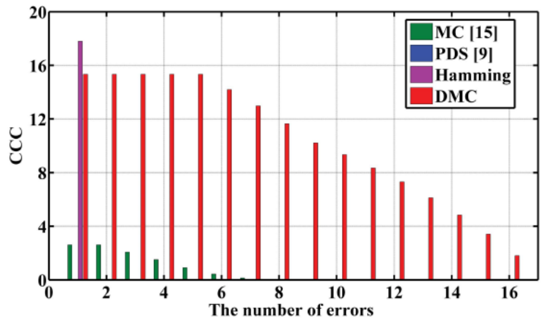


Figure 10: CCCs of different protection codes.

code has the least β value but its correction capability is a constant (1). For the MC [15], its correction capability is also a constant (2) due to the limits of error correction capability of Hamming code. PDS [9] has a lower β value compared with the proposed scheme, but it requires higher delay overhead which would severely affect the access time of memory. The scaling down of CMOS technology has resulted in a dramatic increase in the number of MCUs. In 90-nm technology, more than three errors have been observed in radiation test [1], [2], so Hamming, MC, and PDS are not good choices for protecting memory. The proposed scheme needs higher β value than other codes but it has higher correction capability. Therefore, designers should choose the optimal combination of k and m based on the radiation test to provide a good tradeoff between reliability and redundant bits. We have also used a metric to assess the efficiency of our proposed scheme compared to PDS, MC, and Hamming, which is called correction coverage per cost (CCC) and can be calculated as follow [15]:

$$\text{CCC} = \frac{\text{Correction Coverage}}{\text{Cost}} \qquad (26)$$

where Cost is obtained by

Cost = Area ·Power · Delay ·Redundantbits. (27)

Fig. 10 shows the values of this metric for different protection codes. From this figure, we can see that the proposed scheme has a higher CCC value than other codes except that only one MCU occurs. Therefore, based on the results in Table III and Fig. 10, the proposed scheme is quite suitable for high-speed memory applications. It should be mentioned that when the number of errors is more than two per word, Hamming and MC codes cannot correct any errors.

4. Conclusion

In this paper novel per-word DMC was proposed to assure the reliability of memory. The proposed protection code utilized decimal algorithm to detect errors, so that more errors were detected and corrected. The obtained results showed that the proposed scheme has a superior protection level against large MCUs in memory. Besides, the proposed decimal error detection technique is an attractive opinion to detect MCUs in CAM because it can be combined with BICS to provide an adequate level of immunity.

The only drawback of the proposed DMC is that more redundant bits are required to maintain higher reliability of memory. So that a reasonable combination of k and m should be chosen to maximize memory reliability and minimize the number of redundant bits based on radiation experiments in actual implementation. Therefore, future work will be conducted for the reduction of the redundant bits and the maintenance of the reliability of the proposed technique.

References

[1] 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.

[2] 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 a22nm designrule," *IEEE Trans. Electron Devices*, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[3] 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.

[4] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro, "Hamming SEC-DAED and extended hamming SEC-DED-TAED codes through selective shortening and bit placement," *IEEE Trans. Device Mater. Rel.*, to be published.

[5] S. Liu, P. Reviriego, and J. A. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 148–156, Jan. 2012.

[6] 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.

[7] 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.

[8] 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.

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

[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.