ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Improving Reliability of Memory against Multiple Cell Upsets Using Decimal Matrix Code

Vonteru Neelima
PG Scholar,
Department of ECE,
S.O.E.T,
Sri Padmavathi Mahila Viswa Vidyalayam,
Tirupati.

Mrs. B.Yamini Pushpa, M.Tech
Faculty
Department of ECE,
S.O.E.T,
Sri Padmavathi Mahila Viswa Vidyalayam,
Tirupati.

Abstract:
Transient multiple cell upsets (MCUs) are getting to be real issues in the unwavering quality of recollections presented to radiation environment. To keep MCUs from bringing about information defilement, more perplexing error correction codes (ECCs) are generally used to secure memory, yet the principle issue is that they would require higher defer overhead. As of late, network codes (MCs) in light of Hamming codes have been proposed for memory assurance. The principle issue is that they are twofold mistake rectification codes and the blunder revision capacities are not enhanced in all cases. In this paper, novel decimal lattice code (DMC) in view of partition image is proposed to improve memory unwavering quality with lower defer overhead. The proposed DMC uses decimal calculation to get the most extreme mistake discovery ability. In addition, the encoder-reuse strategy (ERS) is proposed to minimize the zone overhead of additional circuits without aggravating the entire encoding and deciphering forms. ERT utilizes DMC encoder itself to be a piece of the decoder. The proposed DMC is contrasted with understood codes, for example, the current Hamming, MCs, and punctured distinction set (PDS) codes. The acquired results demonstrate that the interim to disappointment of the proposed plan is 452.9%, 154.6%, and 122.6% of Hamming, MC, and PDS, individually. In the meantime, the defer overhead of the proposed plan is 73.1%, 69.0%, and 26.2% of Hamming, MC, and PDS, separately. The main disadvantage to the proposed plan is that it requires more repetitive bits for memory insurance. At present the proposed system reduces the redundant bits from 32 bits to 24 bits.

*Index Terms***—** ***Decimal algorithm, error correction codes(ECCs), mean time to failure (MTTF), memory, multiple cells upsets (MCUs).***

I. INTRODUCTION

As CMOS innovation downsizes to nano-scale and recollections are joined with an expanding number of electronic frameworks, the delicate blunder rate in memory cells is quickly expanding, particularly when recollections work in space situations because of ionizing impacts of environmental neutron, alpha-molecule, and astronomical beams [1]. Albeit single piece bombshell is a noteworthy worry about memory unwavering quality, numerous phone upsets (MCUs) have turned into a genuine dependability worry in some memory applications [2]. So as to make memory cells as blame tolerant as could be expected under the circumstances, some blunder rectification codes (ECCs) have been generally used to ensure recollections against delicate mistakes for a considerable length of time [3]–[6]. For instance, the Bose–Chaudhuri–Hocquenghem codes [7], Reed–Solomoncodes [8], and punctured difference set (PDS) codes [9] have been utilized to manage MCUs in recollections. Be that as it may, these codes require more zone, power, and postpone overheads since the encoding and translating circuits are more intricate in these confused codes. Interleaving strategy has been utilized to limit MCUs [10], which revise cells in the physical game plan to isolate the bits in the same sensible word into various physical words. Be that as it may, interleaving system may not be for all intents and purposes utilized as a part of substance addressable memory (CAM), on account of the tight coupling of equipment structures from both cells and correlation circuit structures [11], [12].

Worked in current sensors (BICS) are proposed to help with single-mistake amendment and twofold blunder discovery codes to give insurance against MCUs [13], [14]. Be that as it may, this procedure can just right two mistakes in a word.More as of late, in [15], 2-D grid codes (MCs) are proposed to proficiently adjust MCUs per word with a low unraveling delay, in which single word is separated into numerous lines and different sections in sensible. The bits per line are ensured by Hamming code, while equality code is included every segment. For the MC [15] in light of Hamming, when two blunders are distinguished by Hamming, the vertical disorder bits are actuated so that these two mistakes can be amended. Therefore, MC is equipped for redressing just two blunders in all cases. In [16], an approach that joins decimal calculation with Hamming code has been imagined to be connected at programming level. It utilizes expansion of number qualities to distinguish and redress delicate blunders. The outcomes got have demonstrated that this approach have a lower postpone overhead over different codes.

In this paper, novel decimal framework code (DMC) in view of separation image is proposed to give upgraded memory dependability. The proposed DMC uses decimal calculation (decimal whole number expansion and decimal whole number subtraction) to identify blunders. The upside of utilizing decimal calculation is that the blunder discovery ability is boosted so that the dependability of memory is improved. Furthermore, the encoder-reuse system (ERT) is proposed to minimize the region overhead of additional circuits (encoder and decoder) without irritating the entire encoding and unraveling forms, since ERT utilizes DMC encoder itself to be a piece of the decoder.

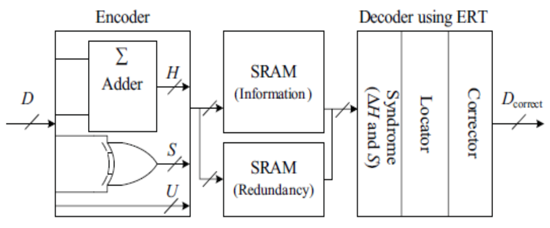


Fig. 1. Proposed schematic of fault-tolerant memory protected with DMC.

II. PROPOSED DMC

In this section, DMC is proposed to assure 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. 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 redundant bits *V* are obtained from the DMC encoder. When the encodingprocess 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 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 ($N=k\times m$), and these symbols are arranged in a $k_1\times k_2$ 2-D matrix ($k=k_1\times k_2$, where the values of k_1 and k_2 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 proposed DMC scheme, we take a 32-bit word as an example, as shown in Fig. 2. The cells from $D_0$to $D_{31}$are information bits. This 32-bit word has beendivided into eight symbols of 4-bit. k_1= 2 and k_2= 4 have been chosen simultaneously. H_0–$H_{19}$are horizontal check bits; V_0through $V_{15}$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 carefullyadjusted to maximize the correction capability and minimize the number of redundant bits. For example, in this case, when $k = 2\times2$ and $m = 8$, only 1-bit error can be corrected and thenumber 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. But now the reliability of memory is improved theerror correction capability is increased,so k=2x4 and m=4 are used to reduce the redundant bits upto 24 bits.

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

$$H_4H_3H_2H_1H_0 = D_3D_2D_1D_0 + D_{11}D_{10}D_9D_8 \qquad (1)$$
$$H_9H_8H_7H_6H_5 = D_7D_6D_5D_4 + D_{15}D_{14}D_{13}D_{12} \qquad (2)$$

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

$$V_0 = D_0 \oplus D_{16} \qquad (3)$$
$$V_1 = D_1 \oplus D_{17} \qquad (4)$$

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 multi-bit 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} to D_0. The enable signal *En* will be explained in the next section.

C. ProposedDMCDecoder

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 *D* . Second, the horizontal syndrome bits H_4 H_3 H_2 H_1 $H_0$and the vertical syndrome bits S_3− S_0canbe calculated as follows:

$$H_4 H_3 H_2 H_1 H_0 = H_4H_3H_2H_1H_0 - H_4H_3H_2H_1H_0 \qquad (5)$$

$$^{S}0 = V_0 \oplus V_0 \qquad (6)$$

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

When $H_4H_3H_2H_1H_0$ and S_3−S_0 are equal to zero, the stored codeword has original information bits in symbol 0where no errors occur. When $H_4H_3H_2H_1H_0$ and S_3–S_0 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

$$^{D}0_{\text{correct}} = {}^{D}0 \oplus {}^{S}0. \qquad (7)$$

The proposed DMC decoder is depicted in Fig. 4, which is made up of the following sub-modules, 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 originalset of excess bits to get the disorder bits H and S. At that point mistake locator utilizes H and S to recognize and find which bits a few blunders happen in.

At last, in the mistake corrector, these blunders can be adjusted by modifying the estimations of blunder bits. In the proposed conspire, the circuit range of DMC is minimized by reusing its encoder. This is known as the ERT. The ERT can diminish the region overhead of DMC without aggravating the entire encoding and disentangling forms. From Fig. 4, it can be watched that the DMC encoder is additionally reused for acquiring the disorder bits in DMC decoder. Along these lines, the entire circuit zone of DMC can be minimized as an aftereffect of utilizing the existent circuits of encoder. Additionally, this figure likewise demonstrates the proposed decoder with an empower flag E n for choosing whether the encoder should be a part of the decoder. As it were, the E n flag is utilized for recognizing the encoder from the decoder, and it is under the control of the compose and read motions in memory. Hence, in the encoding (compose) prepare, the DMC encoder is just an encoder to execute the encoding operations. Notwithstanding, in the translating (read) prepare, this encoder is utilized for registering the disorder bits in the decoder. These plainly indicate how the territory overhead of additional circuits can be generously decreased.

D. Limits of Simple Binary Error Detection

For the proposed binary error detection technique in [13], although 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.

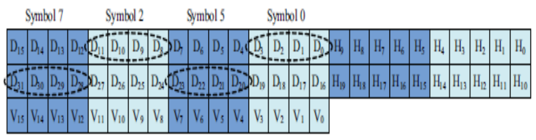


Fig. 2. 32-bits DMC logical organization (k= 2 × 4 and m= 4). Here, each symbol is regarded as a decimal integer.

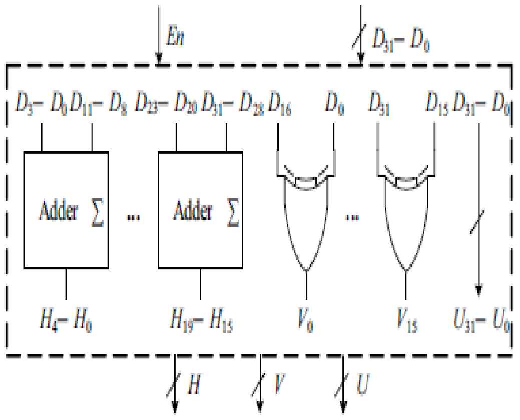
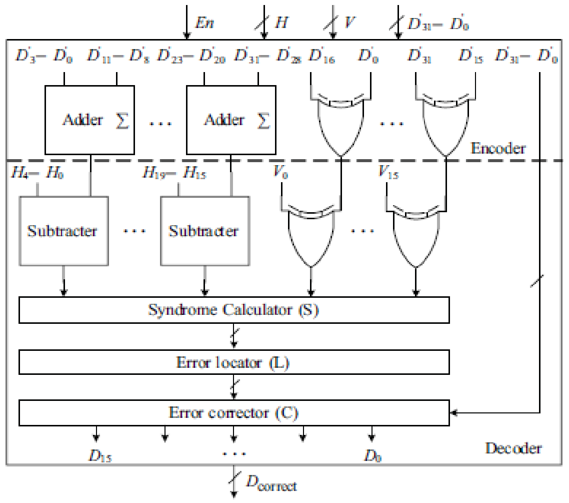

En
$D_{31}-D_0$
D_3-D_0 $D_{11}-D_8$ $D_{23}-D_{20}$ $D_{31}-D_{28}$ D_{16} D_0 D_{31} D_{15} $D_{31}-D_0$
Adder ∑
Adder ∑
H_4-H_0
$H_{19}-H_{15}$
V_0
V_{15}
$U_{31}-U_0$
H
V
U

Fig. 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

Fig. 4(a).32-bit DMC decoder structure using ERT.

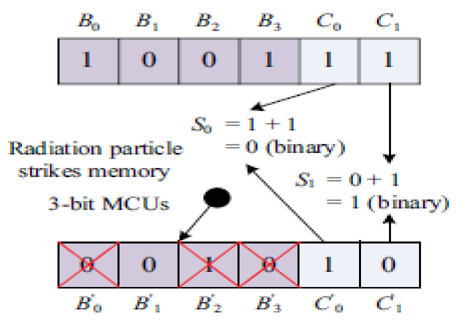

B_0 B_1 B_2 B_3 C_0 C_1
1 0 0 1 1 1
$S_0 = 1 + 1$
$= 0$ (binary)
Radiation particle strikes memory
3-bit MCUs
$S_1 = 0 + 1$
$= 1$ (binary)
0 0 1 0 1 0
B'_0 B'_1 B'_2 B'_3 C'_0 C'_1

Fig.4(b).Limits of binary error detection in simple binary operations.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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

(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

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

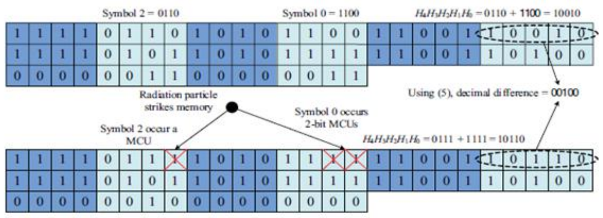

Symbol 2 = 0110
Symbol 0 = 1100
$H_4H_3H_2H_1H_0$ = 0110 + 1100 = 10010
Radiation particle strikes memory
Using (5), decimal difference = 00100
Symbol 0 occurs 2-bit MCUs
Symbol 2 occur a MCU
$H_4H_3H_2H_1H_0$ = 0111 + 1111 = 10110

Fig. 6.Advantage of decimal error detection. Using decimal algorithm, H4H3H2H1H0 will not be "0" (decimal).

This represents that MCUs can be detected and corrected so that the decoding error can be avoided.

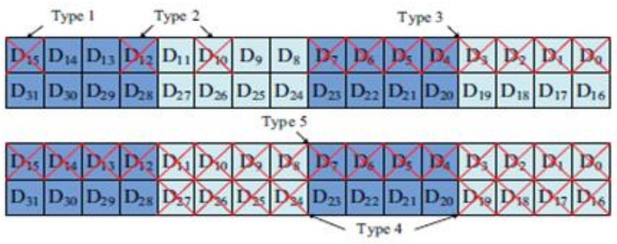


Fig. 7. Types of MCUs can be corrected by our proposed DMC. Type 1 is a single error, type 2 is an inconsecutive error in two consecutive symbols, type 3 is a consecutive error in two consecutive symbols, type4 is an inconsecutive error in two inconsecutive symbols, and type 5 is a consecutive error in four consecutive symbols.

These results mean that error bits $B2$ and $B0$ 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 errors. However, when the decimal algorithm is used to detect errors, 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_4 H_3 H_2 H_1 $H_0$are obtained from the original information bitsin symbols 0 and 2 according to (1)

$$H_4\,H_3\,H_2\,H_1\,H_0 = D_3\,D_2\,D_1\,D_0 + D_{11}\,D_{10}\,D_9\,D_8$$
$$= 1100 + 0110$$
$$= 10010. \quad (14)$$

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 bitsH_4 H_3 H_2 H_1 $H_0$are first computed, as follows:

$$H_4\,H_3\,H_2\,H_1\,H_0 = D_{11}\,D_{10}\,D_9\,D_8 + D_3\,D_2\,D_1\,D_0$$
$$= 0111 + 1111$$
$$= 10110. \quad (15)$$

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

$$H_4\,H_3\,H_2\,H_1\,H_0 = H_4\,H_3\,H_2\,H_1\,H_0 - H_4H_3H_2H_1H_0$$
$$= 10110 - 10010$$
$$= 00100. \quad (16)$$

The decimal value of $H_4H_3H_2H_1H_0$ 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 the decimal value of$H_4H_3H_2H_1H_0$ 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, baseddecimal algorithm, the proposed technique has higher tolerance capability for protecting memory against MCUs. As a result, it is possible 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.

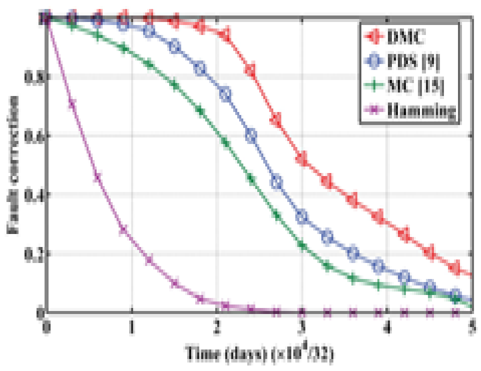


Fig. 9.J(S)s versus time of different protection codes (M= 32).

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

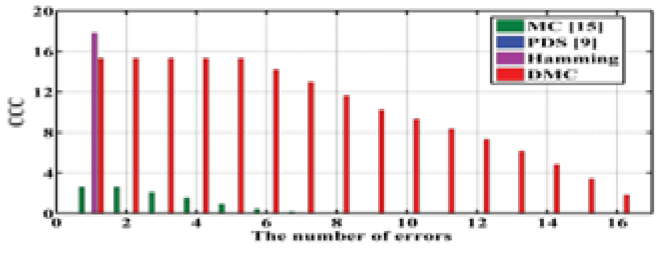


Fig. 10. CCCs of different protection codes.

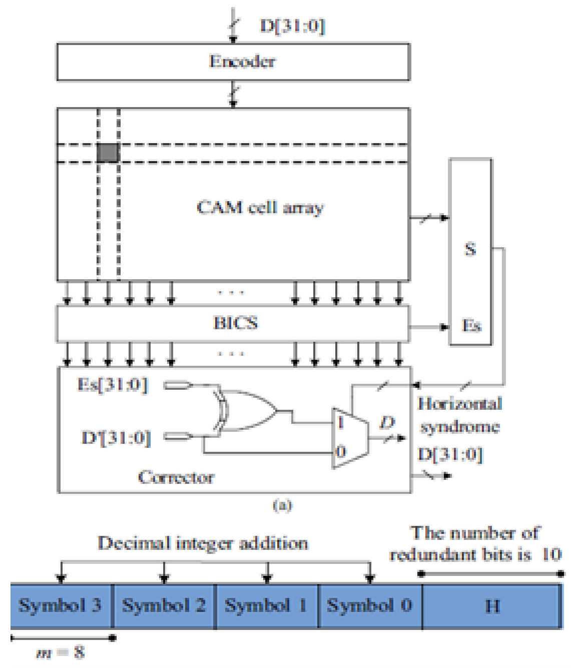


Fig. 11. (a) Proposed fault-tolerant CAM using decimal error detection technique together with BICS. Note that when errors do not exist in CAM, the stored codeword is directly output without though error detection and correction circuits. (b) 32-bit word organization in CAM (k= 1 × 4 and m =8).

DECIMAL ERROR DETECTION IN CAM

ECC code is an effective method to remedy MCUs in memory, as specified some time recently. In any case, ECC usage in CAM is fundamentally not the same as it execution in SRAM because of synchronous access to every one of the words in CAM, with the goal that ECC code is not appropriate to specifically secure CAM [12]. In [14], BICS together with Hamming code is utilized to ensure SRAM. Since BICS has zero blame identification idleness for numerous blunder discovery, it is reasonable for location mistakes in CAM also. For the decimal mistake identification, this capacity to recognize any sort of blunder can be helpful in CAM. Give us a chance to consider that the decimal blunder location procedure joins with BICS to secure CAM. The blame tolerant CAM structure is portrayed in Fig. 11(a) and a 32-bit word association in CAM is proposed in Fig. 11(b). For every section of

CAM, BICS is added to recognize the blunder (the fundamental guideline and circuit of BICS are appeared in [13] and [14]). At the point when MCUs happen in an expression of CAM, for every blunder section, a flitting current heartbeat is created between power supply and ground. BICS can recognize this present heartbeat and produce a mistake flag E s, i.e., this E s flag distinguishes and finds segments which the blunders happen in. In the meantime, the disorder count is dynamic to distinguish the mistake push, i.e., (5) is performed line and line. At that point in the blunder corrector these mistakes can be rectified. At long last, the correctable word is composed back in CAM.

Since the proposed decimal mistake recognition system can identify any number of blunders in a word, the dependability of CAM has a satisfactory level of insusceptibility to MCUs in a word. For instance, when 32-bit mistakes happen in an expression of CAM, the disorder bits H can distinguish these blunders (H can identify blunders yet can't find the exact bombshell areas; this is sufficient) and initiate the disorder count so that all mistakes can be redressed to the detriment of slightest time utilization.

RESULTS

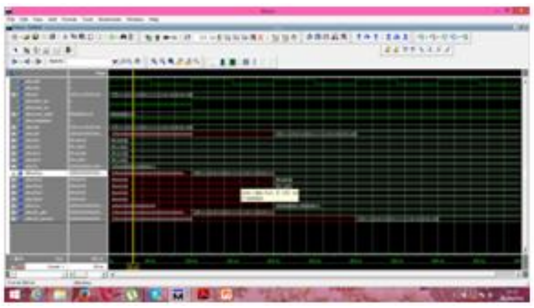


Fig. 12(a). 24-redundant bits

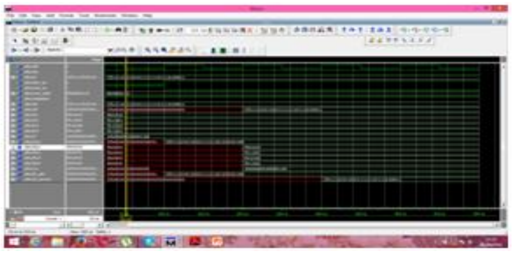


Fig. 12(b). 24-redundant bit-execution

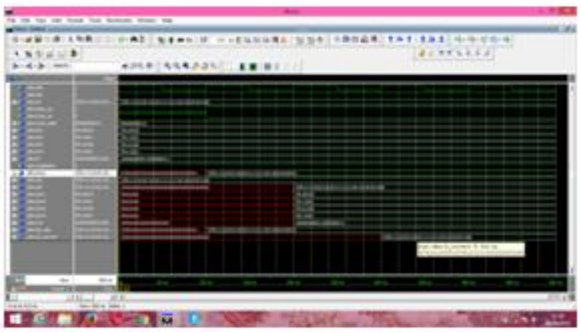


Fig. 12(c). 24-redundant bit-execution

CONCLUSION

In this paper, novel per-word DMC was proposed to guarantee the unwavering quality of memory and reduces the redundant bits to lower level. The proposed assurance code used decimal calculation to identify blunders, so that more mistakes were recognized and redressed. The acquired results demonstrated that the proposed plot has an unrivaled assurance level against expansive MCUs in memory. Also, the proposed decimal mistake discovery strategy is an appealing conclusion to distinguish MCUs in CAM since it can be joined with BICS to give a sufficient level of resistance.

The main advantageof the proposed DMC is that less repetitive bits are required to keep unwavering quality of memory, so that a sensible blend of k and m ought to be amplified memory dependability and minimize the quantity of excess bits in light of radiation examinations in genuine execution. Hence, future work will be directed for the lessening of the excess bits and the upkeep of the dependability of the proposed procedure.

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 a 22 nm design rule,”

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," IEEETrans. 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-StateCircuits, 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 soft error failure model," IEEE Trans. Nucl. Sci., vol. 56, no. 4, 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. DesignAutom. 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," IEEETrans. 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. 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.

17. 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, 1030–1036, Aug. 2007.

18. 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.

19. 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.

20. F. Alzahrani, and T. Chen, "On-chip TEC-QED ECC for ultra-large, single-chip memory systems," in Proc. IEEE Int. Conf. Comput.Design Design, Very-Large-Scale Integr. (VLSI) Syst. Comput. Process.,Oct. 1994, pp. 132–137