Improved Error Correction Capability using Parity Matrix Code

X. Susan Christiana *, K. Javid, C. Mahalakshmi
Department of Electronics & Communication, MIET Engineering College, Trichy, Tamil Nadu, India

Article Info
Article history:
Received 5 February 2015
Received in revised form
20 February 2015
Accepted 28 February 2015
Available online 6 March 2015

Keywords
Parity Algorithm,
Error Correction Codes (ECC) ,
Memory,
Multiple Cells Upsets (MCUs)

Abstract
Multiple cell upsets (MCUs) are formed into a serious reliability problem in memory applications. To make fault-tolerant memory cells, Error correction codes (ECCs) are widely used against soft errors for years. In this paper, Parity matrix code is discussed to improve the error correction capability with reduced number of check bits. Maximum error correction can be achieved by parity algorithm. Parity Matrix Code is compared with the existing Decimal Matrix Code in terms of Power and Delay requirements.

1. Introduction

Multiple cell upsets are defined as instantaneous errors in more than one memory cell made by a single event. The idea of the MCU contains both upsets that can be fixed by error detection/correction code as well as those which cannot. MBU of memory cells may cause wrong functioning of computer systems. In addition, MCU can be a risk in mission-critical systems with an extreme number of logic devices that are mainly protected by spatial or time redundancies.

Though single bit upset is a major problem in memory reliability, multiple cell upsets (MCUs) have become a serious reliability issue in some memory applications. To avoid memory reliability issues, some error correction codes (ECCs) have been widely used. Conventional error correction codes explain various methods that relate soft error. For example, the Bose–Chaudhuri–Hocquenghem [4] codes, Reed–Solomon codes [8], and punctured difference set (PDS) [2] codes have been used to handle with MCUs in memories. But these codes need more area, power, and delay overheads subsequently the encoding and decoding circuits are more complex in these complicated codes. Interleaving method [3] has been used to solve MCUs, which reorder cells in the physical arrangement to isolate the bits in the equivalent logical word into changed physical words. [7] [9] Because of the tight connection of hardware arrangements from both cells and comparison circuit configurations interleaving method may not be practically used in content-addressable memory (CAM). Built-in current sensors (BICS) are suggested to assist with single-error rectification and double-error identification codes to provide protection against MCUs. [5] [6] But this method is capable of correcting two errors only in a word. In [4], matrix codes (MCs) are projected to proficiently correct MCUs for each word with a reduced delay overhead. In MC one word is separated into several rows and columns in logical. The bits in each row are secured by Hamming code and column bits are protected by parity code. But MC is capable of fixing two errors only. In [1], Decimal Matrix Code 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 method, Parity Matrix Code (PMC). 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 paper fixes maximum number of slips in memories.

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

2. Decimal Matrix Code

The diagram of fault-tolerant memory is represented in Fig.1. Firstly, throughout the encoding process, data bits D are given to the DMC encoder, and then the horizontal check bits H and vertical check bits V are found from the DMC encoder. When the encoding procedure is completed, the achieved DMC encryption term is stored in the memory. MCUs happen in the memory means these errors can be fixed by the decoding process [1].

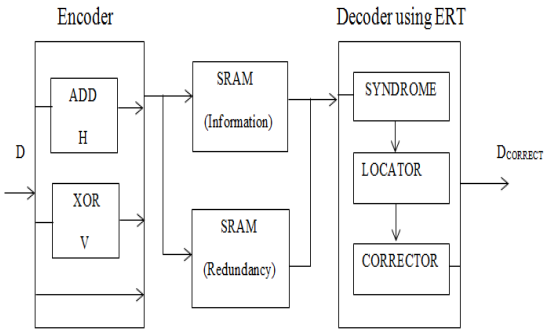

Fig: 1. Memory protected with DMC

Corresponding Author,
E-mail address: mahala.msec@gmail.com

A. DMC Encoder

First of all in the DMC encoder the divide-symbol and arrange-matrix concepts are accomplished, i.e., the N-bit word is separated into k symbols of m bits (N = k × m), and these symbols are set in a k1× k2 2-D matrix (k = k1× k2, where the values of k1 and k2 denote the numbers of rows and columns respectively). Instant, the horizontal check bits H are formed by doing decimal integer addition of designated symbols for each row. Finally, the vertical check bits V are formed by Exoring the bits for each column.

To clarify the DMC structure, we take a 32-bit word as presented in Fig. 2. symbol 7 symbol 2 symbol 5 symbol 0

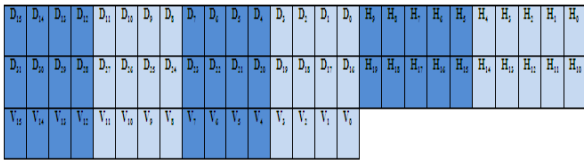

Fig: 2. 32 bit word arrangement in DMC

The cells from D0 to D31 are data bits. This 32-bit word has been separated into eight symbols of 4-bit. k1= 2 and k2 = 4 have been taken instantaneously. H0–H19 are horizontal check bits; V0 through V15 are vertical check bits. But the problem is maximum correction capability and the number of redundant bits is dependent on the values of k and m.

The horizontal check bits H can be found as,

H4H3H2H1H0 = D3D2D1D0 + D11D10D9D8 (1)

H9H8H7H6H5 = D7D6D5D4 + D15D14D13D12 (2)

And likewise for the remaining horizontal check bits H14H13H12H11H10 and H19H18H17H16H15,

The vertical check bits V can be found as,

V0 = D0 ⊕ D16 (3)

V1 = D1 ⊕ D17 (4)

And likewise for the remaining vertical check bits V2 to V15.

B. DMC Decoder

To achieve a term being fixed, the decoding process is necessary. Firstly, the received check bits H4H3H2H1H0' and V0'− V3' are produced by the received data bits D'. Instant, the horizontal syndrome bits ΔH4H3H2H1H0 and the vertical syndrome bits S3 − S0 can be obtained as in [1]

ΔH4H3H2H1H0 = H4H3H2H1H0'− H4H3H2H1H0 (5)

S0 = V0'⊕ V0 (6)

And likewise for the remaining syndrome bits ΔH9H8H7H6H5, ΔH14H13H12H11H10, ΔH19H18H17H16H15 and S1 to S15

When ΔH4H3H2H1H0 and S3 − S0 are equivalent to zero, the kept encryption term has original data bits in symbol0 where no errors happen. When ΔH4H3H2H1H0 and S3 − S0 are nonzero, the made errors are identified and found in symbol0, and then these errors can be fixed by

D0 correct = D0 ⊕ S0. (7)

Even though DMC has lower area, power and delay overheads with extreme error rectification ability, it requires more number of check bits for memory security. The suggested PMC will overcome this problem.

3. Parity Matrix Code

Parity matrix codes are calculated by creating a sparse parity-check matrix and then shaping a generator matrix for the code subsequently. A bit storage type of Content Addressable Memory is used for storage.

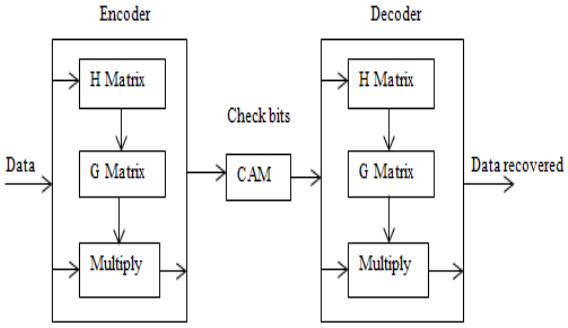

Fig: 1: Memory protected with PMC

A. PMC Encoder

In PMC encoder, firstly Parity Matrix H is generated as combination of data bits and identity matrix (data bits are shifted right)

$$H = \begin{pmatrix} 0\,1\,1\,0 & 1\,0\,0\,0 \\ 0\,0\,1\,1 & 0\,1\,0\,0 \\ 1\,0\,0\,1 & 0\,0\,1\,0 \\ 1\,1\,0\,0 & 0\,0\,0\,1 \end{pmatrix}$$

Then matrix G is generated from H matrix by row column operations (transposing)

$$G = \begin{pmatrix} 1\,0\,0\,0 & 0\,0\,1\,1 \\ 0\,1\,0\,0 & 1\,0\,0\,1 \\ 0\,0\,1\,0 & 1\,1\,0\,0 \\ 0\,0\,0\,1 & 0\,1\,1\,0 \end{pmatrix}$$

Then check bits are generated by matrix addition and multiplication of G matrix with data bits

$$C = \begin{pmatrix} 0\,1\,0\,1 \end{pmatrix} \begin{pmatrix} 1\,0\,0\,0 & 0\,0\,1\,1 \\ 0\,1\,0\,0 & 1\,0\,0\,1 \\ 0\,0\,1\,0 & 1\,1\,0\,0 \\ 0\,0\,0\,1 & 0\,1\,1\,0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0\,1\,0\,1 & 1\,1\,1\,1 \end{pmatrix}$$

The obtained code word stored in CAM. If MCU's happened in memory it will be handled by the decoder.

B. PMC Decoder

The decoder just undoes the encoding process so that the original data can be recovered. To analyze the error detection and correction capability of the projected method, binary operation on check bits has been used.

4. Results

In this section, DMC and PMC codes are implemented in HDL, simulated with ModelSim and tested for functionality by given various inputs. The encoder and decoder circuits are synthesized using Xilinx.

Fig.3 depicts the simulation result of existing Decimal Matrix Code.

Simulation result of Parity Matrix Code implemented in CAM is depicted in Fig.4.

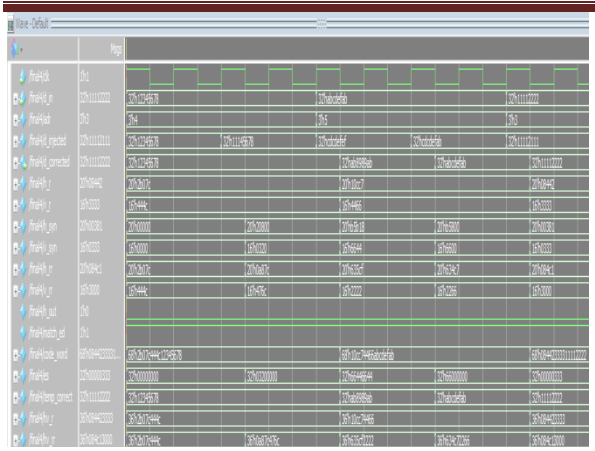

Fig: 3. Simulation Result of DMC in CAM

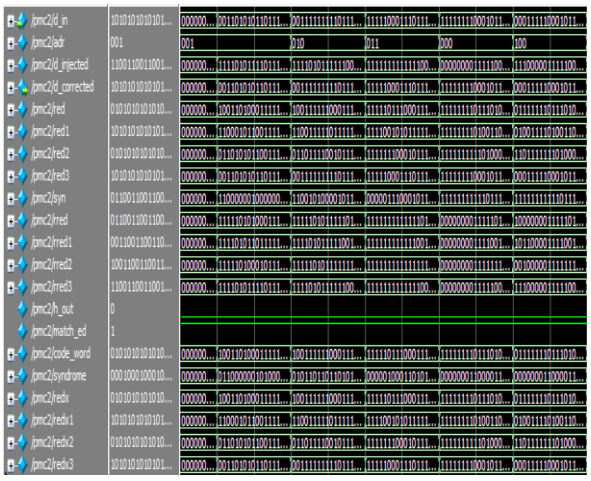

Fig: 4. Simulation result of PMC in CAM

Table: 1: Error Correction capability Compsrison

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

Table 1 compares the error correction capability of the existing DMC with the proposed PMC. From that it is noted that proposed method has the maximum error correction capability with reduced number of check bits compared to the existing method.

Table: 2. Power and Delay Estimation

ECC codes	Power (mw)	Delay (ns)
PMC (Proposed)	492	15.5
DMC (Existing)	494	15.5

Table 2 compares the Power and delay overheads of DMC with PMC. From that it is noted that PMC has equal delay and has a small decrease in power with respect to DMC.

5. Conclusion

Parity based matrix codes is proposed to increase the error handling capability. Even though Decimal Matrix code 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. The proposed method will increase number of detectable and correctable errors and will decrease the total number of extra bits need to be stored to detect the errors. Drawback of the existing system is solved in this method.

References

[1] Jing Guo, Liyi Xiao, Member, IEEE, Zhigang Mao, Member, IEEE, and Qiang Zhao "Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code" IEEE Transactions on VLSI systems, 22(1), 2014

[2] S. Liu, P. Reviriego, J. A. Maestro, Efficient majority logic fault detection with difference-set codes for memory applications, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 20(1), 2012, 148–156

[3] P. Reviriego, M. Flanagan, J. A. Maestro, A (64, 45) triple error correction code for memory applications, IEEE Trans. Device Mater. Rel., 12(1), 2012, 101–106

[4] M. Zhu, L. Y. Xiao, L. L. Song, Y. J. Zhang, H. W. Luo, New mix codes for multiple bit upsets mitigation in fault-secure memories, Microelectron. J., 42(3), 2011, 553–561

[5] C. Argyrides, R. Chipana, F. Vargas, 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., 60(3), 2011, 528–537

[6] S. Baeg, S. Wen, R. Wong, Minimizing soft errors in TCAM devices: A probabilistic approach to determining scrubbing intervals, IEEE Trans. Circuits Syst. I, Reg. Papers, 57(4), 2010, 814–822

[7] S. Baeg, S. Wen, R. Wong, Interleaving distance selection with a soft error failure model, IEEE Trans.Nucl. Sci., 56(4), 2009, 2111–2118

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

[9] K. Pagiamtzis, A. Sheikholeslami, Content addressable memory (CAM) circuits and architectures: A tutorial and survey IEEE J. Solid-State Circuits, 41(3), 2003, 712–727