International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 9 Issue 3 ǁ 2021 ǁ PP. 58-63

High Performance Error Detection/Correction Codes for Multiple Bit Upsets in Memories

A.Alphiya[1]
*[*1] Assistant Professor, Department of ECE, Ponjesly College of Engineering, Nagercoil.*

Abstract
Multiple bit upsets are important problem in designing memories. This affects the reliability and in order to maintain reliability error detection and correction scheme is used. This work presents a method to mitigate multiple bit upsets in memories. The proposed detection/correction method is called High performance Error Detection/Correction codes and the protection bits are used in the form of a matrix. This code combines Hamming codes and Parity codes. Hamming codes are used to detect double bit errors and correct single bit errors. It uses extra check bits, that are added along each row in a matrix to check for errors and performs the checks using simple check equation, which covers a portion of bits. A parity bit is an extra bit attached to the word to detect and correct for errors. This can be evaluated using fault injection experiments. The check bits and parity bits are calculated for the faulty input and they are compared with the check bit and the parity bit obtained from the actual code word and the syndrome bit is generated. The actual code word is obtained using the error correction formula.

Keywords: *Multiple Bit Upsets, Hamming codes, Check Bits, Parity codes, memories.*

Date of Submission: 27-02-2021 Date of acceptance: 12-03-2021

I. INTRODUCTION

Single event upsets (SEU) are caused by ionizing radiation alpha particles and cosmic rays [1] in microcircuits including memory chips, microprocessors, etc. Alpha particles are emitted by contaminants in memory chip packages. When a single charged particle strikes the silicon, due to the production of electron hole pairs it loses its energy. If the electron hole pair is collected by the source or drain diffusion, it could change the voltage level of the node. Such a flip of one bit makes the electronic device to lockup, crash or unstable. Due to the recent increase of the soft error rate of the combinatorial logic circuit [2], this issue has drawn a growing attention from the fault tolerance community. Cosmic rays carry energetic particle that cause upsets in electronic device, even they are protected by packing and shielding [3][4]. An ionizing particle can induce more than one bit failure [5] called multiple bit upsets (MBU), which becomes important problem in designing memories because of the following:

1) As the amount of transistors in the device increase, the number of upsets increases [6][7].
2) Due to technology shrinkage [3] [8], the error rate of memories are increased.

Memory errors are transient, intermittent or permanent in nature. New ideas have been given by scientists, researchers and the memory chip designers over hundreds of years. The focus of all is to get the reliable systems either by eliminating the cause of errors or by eliminating the effects of errors in memories and logic circuits. A high level method for detection and correction of multiple faults is proposed in this work. This method is based on combining Hamming codes and Parity codes in a matrix format, so the detection and correction of multiple faults can be achieved.

The remainder of this work is organized as follows. Section II provides some background and related work. High performance Error Detection/Correction code (EDC) is introduced in section III. Experimental results based on fault injection of the proposed method are provided in section IV, and finally section V concludes this paper.

II. BACKGROUND AND RELATED WORK

Error detecting codes (EDC) and error correcting codes (ECC) are common techniques used to protect memory against errors. Hamming code has efficient ability to correct single upsets per coded words with reduced area overhead and performance [9]. Therefore it is used to protect data against SEU. However Hamming code is not suitable to cope with multiple errors. Reed Solomon (RS) code is a block based error correcting code, which corrects multiple bit upsets but it does not correct double faults located in two adjacent blocks. This requires the use of RS code with double block correction capability, and the cost of double block correction code is high compared to single block correction code [10].

Bit interleaving is one of the most commonly used techniques to minimize the probability of multiple bit upsets in a single word. In interleaving the cells that belong to the same logical word are separated. Moreover interleaving is not feasible for smaller memories such as content addressable memory or register files [11].

Matrix code proposed in [12] can correct MBUs when the parity bit is 1. The drawback with this method is when the parity bit corresponds to the particular column is 0; errors that occur along the particular column will not get corrected. If the parity bits obtained from the saved data bits are all 0, even the single bit upsets cannot get corrected. New mix codes [13] can deal with multiple errors in memories. But when more than two errors occur in the memory the correction coverage does not reach 100%.

Efficient two dimensional error codes [14] are proposed to assure the reliability of memory. The general drawback with this method is if the interval of errors equals to maximum number of errors produced by MBU, these errors are uncorrectable. A Bose-Chaudhuri-Hocquenghem (BCH) [15] code, Golay [16] codes is another protection code that is able to detect and correct multiple errors. The limitations of these methods are latency and power consumption. Moreover the encoding and decoding are more complex and require several lookup tables for multiplication in higher order fields.

In order to replace the defective ones, there are certain techniques that are based on the use of redundant elements. In one dimensional redundancy [17]-[19] only redundant rows (or redundant columns) are included in the memory array, and used to replace defective rows (or defective columns). A defective column (row) containing multiple defective cells cannot be replaced by a single redundant row (column), and therefore its repair efficiency is low.

In two dimensional redundancy [20],[21] both redundant rows and columns are added to the memory array, when multiple defective cells exist in the same row or column of the array, it provides more efficient repair. When multiple faulty cells are detected the use of redundant row or redundant column to replace them is made based on the maximum repair capability. The main drawback of this approach is when the number of defective cells exceeds the redundant element; the chip is to be discarded.

The last alternative before discarding the chip is to use it as a downgraded version of memory [22]. The drawback with this technique is that they do not provide soft error tolerance. Another method is called triple modular redundancy (TMR) in which the memory elements triplicates [9]. The drawback with this approach is TMR increases the area of storage cells. The existing techniques do not correct multiple bit errors and in order to correct multiple bit errors high performance error detection/correction code is used.

III. PROPOSED WORK

The detection/correction method that is used to mitigate multiple bit upsets in memories is called high performance error detection/correction codes in which the message bits are arranged in the form of a matrix. A (r, c) matrix is formed which represents the number of rows and columns in the memory.

Table 1: 40 bit message with check and parity bits

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	n_1	n_2	n_3	n_4	n_5
m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}	n_6	n_7	n_8	n_9	n_{10}
m_{17}	m_{18}	m_{19}	m_{20}	m_{21}	m_{22}	m_{23}	m_{24}	n_{11}	n_{12}	n_{13}	n_{14}	n_{15}
m_{25}	m_{26}	m_{27}	m_{28}	m_{29}	m_{30}	m_{31}	m_{32}	n_{16}	n_{17}	n_{18}	n_{19}	n_{20}
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8					

The proposed method uses Hamming codes and Parity codes to detect and correct for errors. Hamming code uses extra check bits that are added along each row in a matrix to check for errors and this performs the checks using simple check equation which covers a portion of bits. C parity bits are added along each column to check for errors. If m is the number of message bits and n the number of check bits, then the Hamming bound to be satisfied is:

$$2^n \geq m+n+1 \tag{1}$$

In order to identify the presence of error in the bit stream, a check code is generated which should have *n* number check bit equations, plus one extra combination to indicate that no error has occurred. The proposed method can be explained by means of considering 32 bit word length memory, which is divided into 4 rows and 8 columns as shown in Fig.1. In this example *m*=8, by rearranging the earlier equation:

$$2^n - n - 1 \geq 8 \tag{2}$$

One way to solve for k is to just select values of k starting at 1 and evaluating it until the bound is reached. For m=8, the solution is n=4 which exceeds the Hamming bound. The general equations for check bits are calculated for any number of bits. In order to frame the check bit equations number the bits starting from 1,

2, 3, etc. and the bit numbers in binary. All bit positions that are powers of two are check bits and other bit positions are message bits. Each message bit is included in a unique set of 2 or more check bits, as determined by the binary form of its bit position.
The check bits are calculated by:

$$n_1 = m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 \tag{3}$$

$$n_2 = m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 \tag{4}$$

$$n_3 = m_2 \oplus m_3 \oplus m_4 \oplus m_8 \tag{5}$$

$$n_4 = m_5 \oplus m_6 \oplus m_7 \oplus m_8 \tag{6}$$

By modulo-2 addition of all data bits extra parity bit can be formed.

$$n_5 = m_1 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 \tag{7}$$

The check bits of the remaining rows are calculated by using:

$$n_{new} = n_{x+(y*z)} \tag{8}$$

$$m_{new} = m_{o+(c*z)} \tag{9}$$

where *x* is the corresponding check bit's position in the first row, *y* is the number of check bits in a row, *z* is the row number, *o* is the corresponding data bits position in the first row and *c* represents the number of columns. The syndrome for check bits are calculated by means of xoring the saved check bits and the erroneous bits.

$$sn_i = n_i \oplus n_i^{'} \tag{10}$$

where *sn* is the syndrome for check bit, *n* is the check bit obtained for saved data bit, n' is the check bit for erroneous data bit. If the syndrome bit resulting from the check bit is 1, the presence of error can be detected.
The parity bits are calculated by using:

$$x_i = m_i \oplus m_{i+8} \oplus m_{i+16} \oplus m_{i+24} \tag{11}$$

where *i* is the number of columns from 1 to 8.
If the parity bit obtained is '0' the result is complemented.

$$y_i = x_i \oplus 1 \tag{12}$$

The syndrome bit of parity bit is calculated by

$$sy_i = y_i \oplus y_i^{'} \tag{13}$$

where *sy* is the syndrome for parity bit, y is the parity bit obtained for saved data bit, y' is the parity bit obtained for erroneous data bit.

If any error occurs in the saved data bits this can be corrected by means of error correcting formula:

$$m_{icorrect} = m_{ierror} \oplus y_i \oplus sy_i \tag{14}$$

m_{ierror} is the erroneous bit, y_i' is the parity bit obtained for the erroneous data bit and sy_i is the corresponding syndrome for parity bit.

The error detection/correction can be explained by means of considering 32 bit word length memory, which is divided into 4 rows and 8 columns. Let us consider the message bit "10101110 00101011 01110010 00101110" is saved onto the memory. Consider that while reading the message bits from the memory some bits are erroneous. Let the erroneous data bits be "00111001 01011010 10110110 01001011". This can be corrected by means of Hamming codes and Parity codes.
The steps that are involved in the error detection/correction are:

1. Calculate the check bits for saved data bits (n_i).
   n_i = 10010 01011 01101 01110
2. Calculate the check bits for erroneous data bits (n_i').
   n_i' = 00111 00000 11100 01010
3. Calculate the syndrome bits for the check bits (sn_i).
   sn_i = 10101 01011 10001 00100
4. Calculate the parity bits for the saved data bits (.x_i)
   x_i = 11011001
5. If the parity bit is '0' the result is complemented.
   y_i = 11111111

6. Calculate the parity bit for the erroneous data bits.
   $y_i' = 10011110$
7. Calculate the syndrome for parity bits.
   sy_i = 01100001
8. Correct the message bits if it is erroneous and write back the data onto the memory.

Inorder to perform MEMORY READ and MEMORY WRITE operation a memory array is to be designed. The steps that are involved in the memory read and memory write operation are:

1. Read the message bits from the memory array.
2. Check for errors.
3. If any error occurs, error detection/correction algorithm is applied and the corrected message bits are written to the memory array.
4. Else the message bits are written back to the memory array.

IV. EXPERIMENTAL RESULTS

For each message bit the proposed Error Detection/Correction (EDC) code can be divided into two different matrices such as "EDC A" and "EDC B". For the 16-bit code word, *r*=2 and *c*=8 for "EDC A" and *r*=4 and *c*=4 for "EDC B". For the 32-bit code word, *r*=4 and *c*=8 for "EDC A" and *r*=8 and *c*=4 for "EDC B". For the 64-bit code word, *r*=4 and *c*=16 for "EDC A" and r=8 and *c*=8 for "EDC B". The check bits and the parity bits that are required for both detection and correction techniques are portrayed in Table-II.

TABLE- II : REQUIRED CHECK BITS AND PARITY BITS

Type of Protection	Word Size		
	16 bits	32 bits	64 bits
EDC A	18	28	40
EDC B	20	36	48
Matrix Codes	18	28	40
Hamming Codes	6	7	8

Hamming codes requires less number of check bits when compared to proposed error detection/correction code and matrix code. The method described in the previous section are coded in VHDL and synthesized in two different devices Spartan3E and Virtex5 from Xilinx and simulated for various inputs. Using fault injection experiment random faults are thrown into the message bits and the error detection/correction is verified. Table III shows the critical path delay of proposed error detection/correction code simulated by means of Virtex5 and table IV shows the critical path delay of proposed error detection/correction code simulated by means of Spartan 3E. The path delay obtained for the proposed code is compared with Hamming codes and Matrix codes.

TABLE- III : DELAY ANALYSIS USING VIRTEX5

Type of protection	Word size		
	16 bits	32 bits	64 bits
EDC A	4.668ns	4.560ns	5.502ns
EDC B	3.884ns	4.239ns	4.560ns
Matrix codes	3.915ns	4.314ns	5.506ns
Hamming Codes	5.732ns	6.359ns	9.331ns

TABLE- IV : DELAY ANALYSIS USING SPARTAN3E

Type of protection	Word size		
	16 bits	32 bits	64 bits
EDC A	8.316ns	8.222ns	10.330ns
EDC B	7.237ns	7.183ns	8.222ns
Matrix codes	8.369ns	7.237ns	10.444ns
Hamming Codes	13.232ns	13.092ns	17.653ns

The delays obtained by means of proposed code and matrix codes are quite similar but the error correction capability of proposed code is high when compared to matrix codes. The design area occupied is measured in terms of look-up table (LUT). Table V shows the number of LUTs that are required to implement the logic using Virtex5.

TABLE- V : NUMBER OF LUTS

LUTs	16 bit		32 bit		64 bit	
	EDCA	EDCB	EDCA	EDCB	EDCA	EDCB
LUT4	9	20	40	24	56	41
LUT5	14	4	24	20	15	73
LUT6	7	12	24	24	75	46

Number of xor gates and the input output buffers (IO buffers) that are used to implement the logic is portrayed in table VI.

TABLE- VI : NUMBER OF XORS AND IO BUFFERS

Data Bits	16 bit		32 bit		64 bit	
	EDCA	EDCB	EDCA	EDCB	EDCA	EDCB
Xors	86	67	163	155	312	314
IOs	110	107	188	208	307	344

V. CONCLUSION

This work describes a high level error detection and correction method called High performance error detection/correction codes which combines Hamming code and Parity code. The proposed code can detect and correct all errors that occurs in the message bits. This can be evaluated using fault injection experiments. The design was tested for various inputs and the fault that occurs in the code word can be detected and corrected. This method provides maximum fault tolerance even the error rate is high. Also the delay obtained is less when compared to the existing methods.

REFERENCES

[1] Robert C. Baumann," Soft Errors in Advanced Semiconductor Devices— Part I: The Three Radiation Sources", IEEE Transactions on device and materials reliability, VOL. 1, NO. 1, March 2001,pp. 17-22.

[2] Premkishore Shivakumar, Michael Kistlery,Stephen W.Keckle, Doug Burger ,Lorenzo Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic", Proceedings of the International Conference on Dependable Systems and Networks (DSN'02), 2002.

[3] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," IEEE Trans. Nucl. Sci., vol. 47, no. 6, pp. 2586–2594, Dec. 2000.

[4] N. Seifert, D. Moyer, N. Leland, and R. Hokinson, "Historical trend in alpha-particle induced soft error rates of the Alpha microprocessor," in Proc. 39th Annu. IEEE Int. Reliab. Phys. Symp., 2001, pp. 259–265.

[5] F. Wrobel, J.-M. Palau, M.-C. Calvet, O. Bersillon, and H. Duarte "Simulation of Nucleon-Induced Nuclear Reactions in a Simplified SRAM Structure: Scaling Effects on SEU and MBU Cross Sections" IEEE Transactions on nuclear science, vol. 48, no. 6, December 2001, pp.1946-1952.

[6] J. Karlsson, P. Liden, P. Dahlgren, R. Johansson, and U. Gunneflo, "Using heavy-ion radiation to validate fault-handling mechanisms," IEEE Trans. Microelectron., vol. 14, pp. 8–23, 1994.

[7] R. Reed, M. Carts, P. Marshall, C. J. Marshall, O. Musseau, P. McNulty, D. Roth, S. Buchner, J. Melinger, and T. Corbiere, "Heavy ion and proton-induced single event multiple upset," IEEE Trans. Nucl. Sci., vol. 44, no. 6, pp. 2224–2229, Dec. 1997.

[8] P. A. Ferreyra, C. A. Marques, R. T. Ferreyra, and J. P. Gaspar, "Failure map functions and accelerated mean time to failure tests: New approaches for improving the reliability estimation in systems exposed to single event upsets," IEEE Trans. Nucl. Sci., vol. 52, no. 1, pp. 494–500, Jan. 2005.

[9] R. Hentschke, R. Marques, F. Lima, L. Carro, A. Susin, and R. Reis, "Analyzing area and performance penalty of protecting different digital modules with hamming code and triple modular redundancy," in Proc. Symp. Integr. Circuits Syst. Des., 2002, pp. 95–100.

[10] G. Neuberger, F. D. Lima, L. Carro, and R. Reis, "A multiple bit upset tolerant SRAM memory," ACMTrans. Des. Autom. Electron. Syst. (TODAES), vol. 8, no. 4, pp. 577–590, 2003.

[11] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in Proc. IEEE VLSI Test Symp. (VTS), 2007, pp. 349–354.

[12] C.Argyrides , D.K.Pradhan, T.Kocak, "Matrix codes for reliable and cost efficient memory chips", IEEE Trans, VLSI Syst, 2011, pp. 420-428.

[13] M.Zhu, Li Yi Xiao, Li Li Song, Yan Jing zhang, Hong Wei Luo, " New mix codes for multiple bit upsets mitigation in fault-secure memories", in microelectronics journal 42, 2011, pp 553-561.

[14] M.Zhu, L.Y. Xiao, S.H.Li, Y.J.Zhang, "Efficient two-dimensional error codes for multiple bit upsets mitigation in memory, in proceedings of the 25th IEEE International Symposium on Defect and Fault-Tolerance in VLSI systems, DFT2010,2010, pp 129-135.

[15] R.Naseer, J.Draper, "Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs", 34th European Solid-state circuits conference, ESSCIR2008, Sept.2008, pp. 222-225.

[16] M.A.Bajura, Y.Boulghassoul, R.Naseer, S.Das Gupta, a.f.Witulski, J.Sondeen, S.D. Stansberry, J.Draper, L.W.Massengill, J.N.Damoulakis, "Models and algorithmic limits for an ECC based approach hardening sub 100 nm SRAMs", IEEE Trans, Nucl. Sci.,vol.54, no.4, Aug 2007, pp. 935-945

[17] D. K. Bhavsar, "An algorithm for row-column self-repair of RAM's and its implementation in the ALPHA 21264," in Proc. Int. Test Conf., 1999, pp. 311–318.

[18] K. Kim, C. Kim, and K. Roy, "TFT-LCD application specific low power SRAM using charge-recycling technique," in Proc. 6th Int. Symp. Quality Electron. Des., 2005, pp. 59–64, 21–23.

[19] C. A. Lisboa, M. I. Erigson, and L. Carro, "System level approaches for mitigation of long duration transient faults in future technologies," in Proc. 12th Eur. Test Symp. (ETS), May 2007, pp. 165–170.

[20] S. K. Lu, "Efficient built-in redundancy analysis for embedded memories with 2-D redundancy," IEEE Trans. Very Large Scale Integr. (VLSI) Systems, vol. 14, no. 1, pp. 34–42, Jan. 2006.

[21] S. K. Lu and S.-C. Huang, "Built-in self-test and repair (BISTR) techniques for embedded RAMs," in Proc. Int.Workshop, Memory Technol. Des. Test., Aug. 2004, pp. 60–64.

[22] C. Argyrides, A. A. Al-Yamani, C. Lisboa, and L. C. D. K. Pradhan, "Increasing memory yield in future technologies through innovative design," in Proc. 8th Int. Symp. Quality Electron. Des. (ISQED), Mar. 2009, pp. 622–626.

[23] Richard Soja, "MC68HC11 EEPROM Error Correction Algorithms in C", Freescale Semiconductor, Inc, pp. 1-15.

[24] C. Argyrides, H. Zarandi, and D. K. Pradhan, "Matrix codes: Multiple bit upsets tolerant method for SRAM memories," in Proc. 22nd IEEE Int. Symp. Defect Fault Toler. VLSI Syst. (DFT), Sep. 2007, pp. 340–348.Shih-Fu Liu, Pedro Reviriego, Juan Antonio Maestro, "Enhanced Implementations of Hamming Codes to Protect FIR Filters," IEEE Transactions on nuclear science, vol. 57, no. 4, Aug 2010