International Journal of Electrical and Electronic Engineering & Telecommunications

ISSN 2319 – 2518 www.ijeetc.com
Vol. 4, No. 4, October 2015

Research Paper

REALIZATION OF POWER OPTIMIZED FAULT TOLERANT MEMORY PROTECTED ECC STRUCTURE

K Sarala[1]* and N Suresh Babu[2]

*Corresponding Author: K Sarala, ✉saralakale65@gmail.com

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. Although single bit upset is a major concern about memory reliability, Multiple Cell Upsets (MCUs) have become a serious reliability concern in some memory applications. In order to make memory cells as fault-tolerant as possible, some Error Correction Codes (ECCs) have been widely used to protect memories against soft errors for years. For example, the Bose Chaudhuri Hocquenghem codes, Reed-Solomon codes, and Punctured Difference Set (PDS) codes 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. In this project due to introduction of parallel corrector block to enhance the performance of the corrector with less power consumption is proposed.

Keywords: Fault-tolerant, Power consumption, Memory, Parallel corrector

INTRODUCTION

Electronic space provided by silicon chips (semiconductor memory chips) or magnetic/ optical media as temporary or permanent storage for data and/or instructions to control a computer or execute one or more programs. Two main types of computer memory are: (1) Read only memory (ROM), smaller part of a computer's silicon (solid state) memory that is fixed in size and permanently stores manufacturer's instructions to run the computer when it is switched on. (2) Random access memory (RAM), larger part of a computer's memory comprising of hard disk, CD, DVD, floppies, etc. (together called secondary storage) and employed in running programs and in archiving of data. Memory chips provide access to stored data or instructions that is

[1] M.Tech Student, Department of ECE, Chirala Engineering College, Chirala, Andhra Pradesh 523157, India.

[2] Associate Professor, Department of ECE, Chirala Engineering College, Chirala, Andhra Pradesh 523157, India.

hundreds of times faster than that provided by secondary storage.

Error detection and correction or error control is techniques that enable reliable delivery of digital data over unreliable communication channels (or storage medium. Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver. Error correction is the detection of errors and reconstruction of the original, error-free data. The goal of error control coding is to encode information in such a way that even if the channel (or storage medium) introduces errors, the receiver can correct the errors and recover the original transmitted information.

ECC stands for "Error Correction Codes" [1] and is a method used to detect and correct errors introduced during storage or transmission of data. Certain kinds of RAM chips inside a computer implement this technique to correct data errors and are known as ECC Memory. ECC Memory chips are predominantly used in servers rather than in client computers. Memory errors are proportional to the amount of RAM in a computer as well as the duration of operation. Since servers typically contain several Gigabytes of RAM and are in operation 24 hours a day, the likelihood of errors cropping up in their memory chips is comparatively high and hence they require ECC Memory.

Memory errors that are not corrected immediately can eventually crash a computer. This again has more relevance to a server than a client computer in an office or home environment. When a client crashes, it normally does not affect other computers even when it is connected to a network, but when a server crashes it brings the entire network down with it. Hence ECC memory is mandatory for servers but optional for clients unless they are used for mission critical applications. An Error-Correcting Code (ECC) [15] or Forward Error Correction (FEC) code is a system of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a back-channel is not required in forward error correction, and it is therefore suitable for simplex communication such as broadcasting. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks, and RAM.

ERROR-CORRECTING CODES

Particularly, we identify a class of Error-Correcting Codes (ECCs) that guarantees the existence of a simple fault-tolerant detector design. This class satisfies a new, restricted definition for ECCs which guarantees that the ECC codeword has an appropriate redundancy structure such that it can detect multiple errors occurring in both the stored codeword in memory and the surrounding circuitries. We call this type of error-correcting codes, are Fault-Secure Detector capable ECCs (FSD-ECC). The parity-check Matrix of an FSD-ECC has a particular structure that the decoder circuit, generated from the parity-check Matrix, is Fault-Secure.

Let $i = (i_0, i_1, i_2, ...i_{k-1})$ be the k-bit information vector that will be encoded into an n-bit codeword, $c = (c_0, c_1, ...c_{n-1})$. For linear codes, the encoding operation essentially performs the following vector-matrix multiplication:

$$c = i.G \quad \ldots(1)$$

where G is a $k \times n$ generator matrix? The validity of a received encoded vector can be checked with the Parity-Check matrix, which is a $(n-k) \times n$ binary matrix named H. The checking or detecting operation is basically summarized as the following vector-matrix multiplication:

$$s = c.H^T \quad \ldots(2)$$

The $(n-k)$-bit vector s is called the syndrome vector. A syndrome vector is zero if c is a valid codeword, and nonzero if c is an erroneous codeword. Each code is uniquely specified by its generator matrix or parity-check matrix. A code is a systematic code if every codeword consists of the original k-bit information vector followed by $n-k$ parity bits. With this definition, the generator matrix of a systematic code must have the following structure:

$$G = [I : X] \quad \ldots(3)$$

where I is a $k \times k$ identity matrix and X is a $k \times (n-k)$ matrix that generates the parity-bits. The advantage of using systematic codes is that there is no need for a decoder circuit to extract the information bits. The information bits are simply available in the first k bits of any encoded vector. A code is said to be a cyclic code if for any codeword c, all the cyclic shifts of the codeword are still valid code words. A code is cyclic if the rows of its parity-check matrix and generator matrix are the cyclic shifts of their first rows.

The minimum distance of an ECC, d, is the minimum number of code bits that are different between any two code words. The maximum number of errors that an ECC can detect is $d-1$, and the maximum number that it corrects is $d/2$. Any ECC is represented with a triple (n, k, d), representing code length, information bit length, and minimum distance, respectively.

IMPLEMENTATION OF FAULT-TOLERENT MEMORY SYSTEM

We outline our memory system design that can tolerate errors in any part of the system, including the storage unit and encoder and corrector circuits using the fault-secure detector. For a particular ECC used for memory protection, let E be the maximum number of error bits that the code can correct and D be the maximum number of error bits that it can detect, and in one error combination that strikes the system, let e_e, e_m, and e_c be the number of errors in encoder, a memory word, and corrector, and let e_{de} and e_{dc} be the number of errors in the two separate detectors monitoring the encoder and corrector units. In conventional designs, the system would guarantee error correction as long as $e_m \le E$ and $e_e = e_c = 0$. In contrast, here we guarantee that the system can correct any error combination as long as $e_m \le E$, $e_c + e_{dc} \le D$, and $e_m + e_c + e_{dc} \le D$. This design is feasible when the following two fundamental properties are satisfied:

1. Any single error in the encoder or corrector circuitry can at most corrupt a single

codeword bit (i.e., no single error can propagate to multiple codeword bits);

2. There is a fault secure detector that can detect any combination of errors in the received codeword along with errors in the detector circuit. This fault-secure detector can verify the correctness of the encoder and corrector operation.

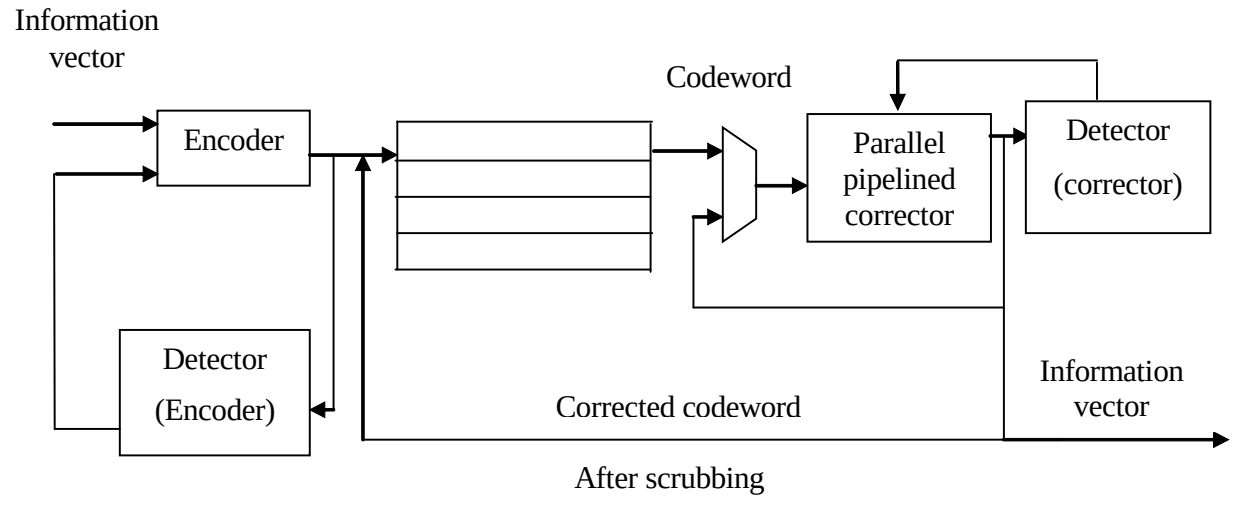
The first property is easily satisfied by preventing logic sharing between the circuits producing each codeword bit or information bit in the encoder and the corrector respectively. We define the requirements for a code to satisfy the second property.

An overview of our proposed reliable memory system is shown in Figure 1 and is described in the following. The information bits are fed into the encoder to encode the information vector, and the fault secure detector of the encoder verifies the validity of the encoded vector. If the detector detects any error, the encoding operation must be redone to generate the correct codeword. The codeword is then stored in the memory. During memory access operation, the stored code words will be accessed from the memory unit. Code words are susceptible to transient faults while they are stored in the memory; therefore a corrector unit is designed to correct potential errors in the retrieved code words. In our design (see Figure 1) all the memory words pass through the corrector and any potential error in the memory words will be corrected. Similar to the encoder unit, a fault-secure detector monitors the operation of the corrector unit. All the units shown in Figure 1 are implemented in fault-prone the only component which must be implemented in reliable circuitry are two OR gates that accumulate the syndrome bits for the detectors shown in Figure 4.

Encoder

An -bit codeword c, which encodes a k-bit information vector i is generated by multiplying the k-bit information vector with a $k \times n$ bit generator matrix G; i.e., $c = i.G$. EG-LDPC codes are not systematic and the information bits must be decoded from the encoded vector, which is not desirable for our fault-

Figure 1: Fault-Tolerant Memory Architecture, with Pipelined Corrector



tolerant approach due to the further complication and delay that it adds to the operation. However, these codes are cyclic codes [1]. We used the procedure presented in [1] and [4] to convert the cyclic generator matrices to systematic generator matrices for all the EG-LDPC codes under consideration.

Figure 2 shows the systematic generator matrix to generate (15, 7, 5) EG-LDPC code. The encoded vector consists of information bits followed by parity bits, where each parity bit is simply an inner product of information vector and a column of X, from $G = [I : X]$. Figure 3 shows the encoder circuit to compute the parity bits of the (15, 7, 5) EG-LDPC code. In this figure $i = (i_0, i_1, i_2, ...i_6)$ is the information vector and will be copied to $(c_0, ...c_6)$ bits of the encoded vector, c, and the rest of encoded vector, the parity bits, are linear sums (XOR) of the information bits. If the building block is two-inputs gates then the encoder circuitry takes 22 two-input XOR gates. The area of the encoder circuits for each EG-LDPC codes under consideration based on their generator matrices. Once the XOR functions are known, the encoder structure is very similar to the detector structure shown in Figure 3.

Figure 2: Generator Matrix for the (15, 7, 5) EG-LDPC in Systematic Format

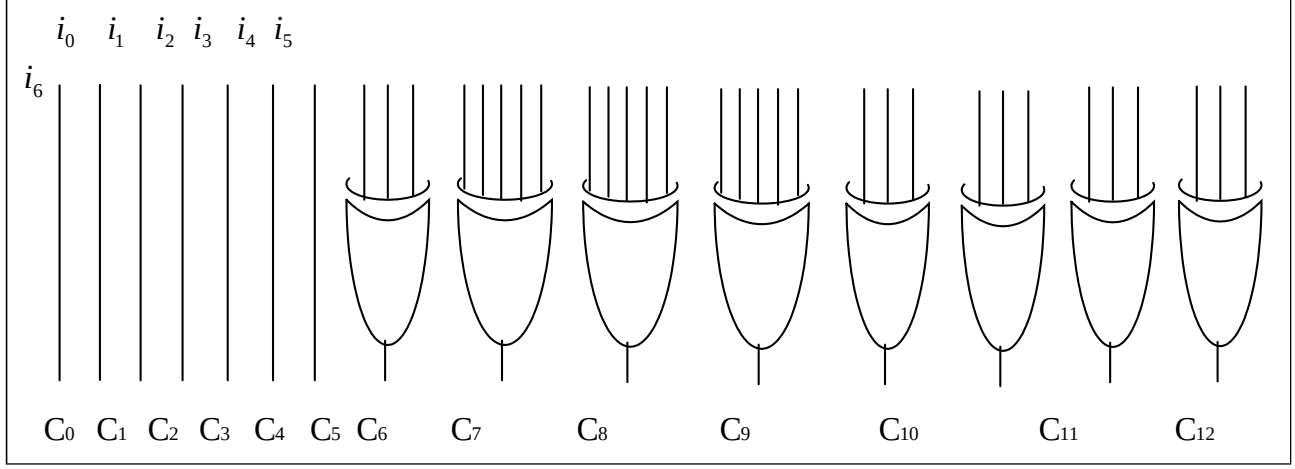
$$\begin{bmatrix} 1&0&0&0&0&0&0&1&0&0&1&1&1&0&1 \\ 0&1&0&0&0&0&0&1&1&0&0&1&1&1&0 \\ 0&0&1&0&0&0&0&0&1&1&1&0&0&0&1 \\ 0&0&0&1&0&0&0&1&0&1&1&1&0&0&0 \\ 0&0&0&0&1&0&0&0&1&0&1&1&1&0&0 \\ 0&0&0&0&0&1&0&0&0&1&0&1&1&1&0 \\ 0&0&0&0&0&0&1&0&0&0&1&0&1&1&1 \end{bmatrix}$$

i_0 to i_6 are 7-bit information vector. Each of the XOR gates generates one parity bit of the encoded vector. The codeword consists of seven information bits followed by eight parity bits.

Fault Secure Detector

The core of the detector operation is to generate the syndrome vector, which is basically implementing the following vector-

Figure 3: Structure of an Encoder Circuit for the (15, 7, 5) EG-LDPC Code

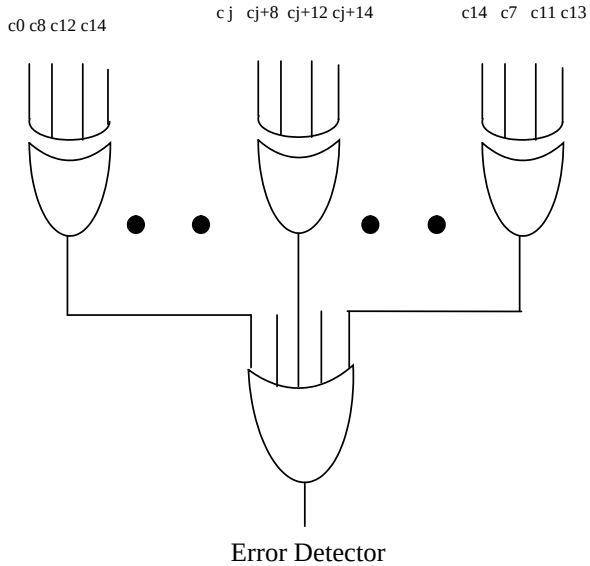


matrix multiplication on the received encoded vector c and parity-check matrix H:

$$s = c.H^{T} \quad \text{...(4).}$$

Therefore each bit of the syndrome vector is the product of c with one row of the parity-check matrix. This product is a linear binary sum over digits of c where the corresponding digit in the matrix row is 1. This binary sum is implemented with an XOR gate. Figure 4 shows the detector circuit for the (15, 7, 5) EG-LDPC code. Since the row weight of the parity-check matrix is ° , to generate one digit of the syndrome vector we need a ° -input XOR gate, or (° −1) 2-input XOR gates. For the whole detector, it takes n(° −1) 2-input XOR gates. Note that implementing each syndrome bit with a separate XOR gate satisfies the assumption of Theorem I of no logic sharing in detector circuit implementation. An error is detected if any of the syndrome bits has a nonzero value. The final error detection signal is implemented by an OR function of all the syndrome bits. The output of this-input OR gate is the error detector signal.

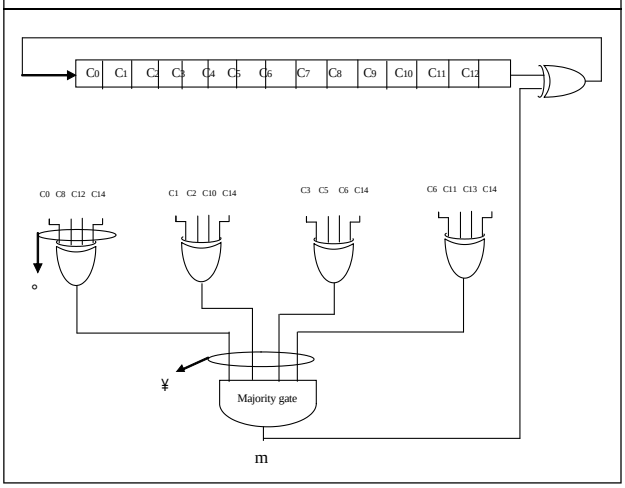
Figure 4: Fault-Secure Detector for (15, 7, 5) EG-LDPC Code



Corrector

One-step majority-logic correction is a fast and relatively compact error-correcting technique [1]. There is a limited class of ECCs that are one-step-majority correctable which include type-I two-dimensional EG-LDPC. In this section, we present a brief review of this correcting technique. Then we show the one-step majority-logic corrector for EG-LDPC codes.

Figure 5: Serial One-Step Majority Logic Corrector Structure to Correct Last Bit (bit14th) of 15-bit (15, 7, 5) EG-LDPC Code



Parallel Corrector

For high error rates [e.g., when tolerating permanent defects in memory words as well, the corrector is used more frequently and its latency can impact the system performance. Therefore we can implement a parallel one-step majority corrector which is essentially n copies of the single one-step majority-logic corrector. Figure 1 shows a system integration using the parallel corrector. All the memory words are pipelined through the parallel

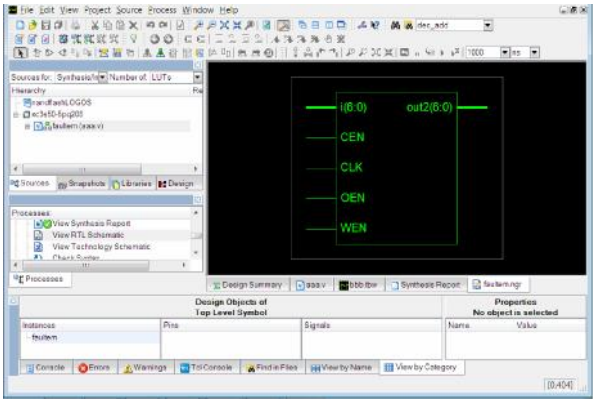
corrector. This way the corrected memory words are generated every cycle. The detector in the parallel case monitors the operation of the corrector, if the output of the corrector is erroneous; the detector signals the corrector to repeat the operation. Note that faults detected in a nominally corrected memory word arise solely from faults in the detector and corrector circuitry and not from faults in the memory word. Since detector and corrector circuitry are relatively small compared to the memory system, the failure rate of these units is relatively low.

RESULTS

RTL Schematic

The RTL SCHEMATIC gives the information about the user view of the design.The internal blocks contains the basic gate representaion of the logic. These basic gate realization is purely depend upon the corresponding FPGA selection and the internal database information.

Figure 6: RTL Schematic



In the waveform which is shown above, clk signal represents clock represents the input which we are applying to the design. Similarly out2 is the output signal for the design. Here

Figure 7: Internal View of RTL Schematic

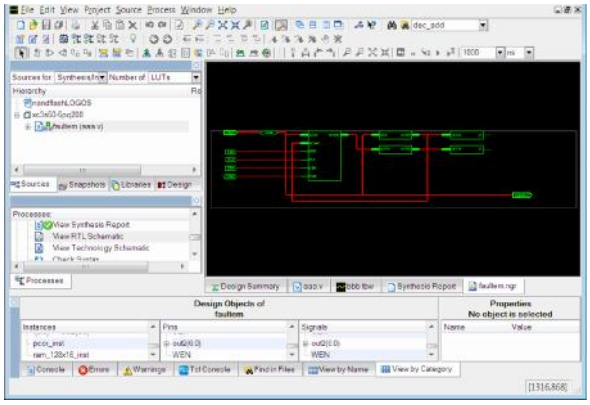
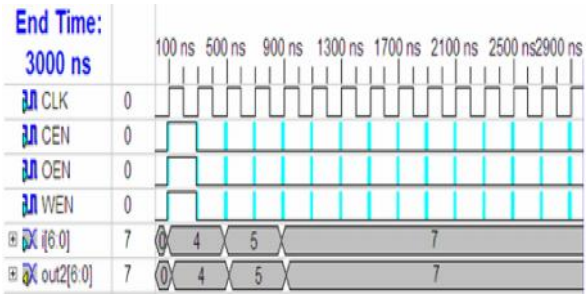


Figure 8: Simulation Waveform of the Top Module



clock signal is generated for the positive edge. Initially the CEN, OEN, WEN signals should be force to logic 1 and after one clock cycle made it to logic 0 for performing the corresponding functional operation. To obtain the required outputs force the inputs logic with the required values. The out2 is following the input i sequence which has to be done in the project to realize the ECC action.

CONCLUSION

The Enhanced Memory Reliability against Multiple Cell Upsets concept is realized. The ECC is implemented by introducing the parallel corrector block to enhance the performance of the corrector with less power consumption. The proposed design consumes 2.709 mw

power which is less when compare to the existed design which consumes 10.8 mw. The functionality is verified through XILINX ISE Simulator and synthesized using XILINX XST. The RTL is developed based on the VERILOG HDL language.

REFERENCES

1. Argyrides C A, Lisboa C A, Pradhan D K and Carro L (2009), "Single Element Correction in Sorting Algorithms with Minimum Delay Overhead", in *Proc. IEEE Latin Amer. Test Workshop*, March, pp. 652-657.
2. Argyrides C and Pradhan D K (2007), "Improved Decoding Algorithm for High Reliable Reed Muller Coding", in Proc. IEEE Int. Syst. on Chip Conf., September, pp. 95-98.
3. Argyrides C, Chipana R, Vargas F and Pradhan D K (2011), "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.
4. Argyrides C, Pradhan D K and Kocak T (2011), "Matrix Codes for Reliable and Cost Efficient Memory Chips", *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, Vol. 19, No. 3, pp. 420-428.
5. Baeg S, Wen S and Wong R (2009), "Interleaving Distance Selection with a Soft Error Failure Model", *IEEE Trans. Nucl. Sci.*, Vol. 56, No. 4, pp. 2111-2118.
6. Baeg S, Wen S and Wong R (2010), "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.
7. Ibe E, Taniguchi H, Yahagi Y, Shimbo K and Toba T (2010), "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.
8. Liu S, Reviriego P and Maestro J A (2012), "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.
9. Mahatme N N, Bhuva B L, Fang Y P and Oates A S (2012), "Impact of Strained-Si PMOS Transistors on SRAM Soft Error Rates", *IEEE Trans. Nucl. Sci.*, Vol. 59, No. 4, pp. 845-850.
10. Naseer R and Draper J (2008), "Parallel Double Error Correcting Code Design to Mitigate Multi-Bit Upsets in SRAMs", in *Proc. 34th Eur. Solid-State Circuits*, September, pp. 222-225.
11. Neuberger G, Kastensmidt D L and Reis R (2005), "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.
12. Pagiamtzis K and Sheikholeslami A (2003), "Content Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", *IEEE J. Solid-State Circuits*, Vol. 41, No. 3, pp. 712-727.

13. Radaelli D, Puchner H, Wong S and Daniel S (2005), "Investigation of Multi-Bit Upsets in a 150 nm Technology SRAM Device", *IEEE Trans. Nucl. Sci.*, Vol. 52, No. 6, pp. 2433-2437.

14. Reviriego P and Maestro J A (2009), "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.

15. Reviriego P, Flanagan M and Maestro J A (2012), "A (64, 45) Triple Error Correction Code for Memory Applications", *IEEE Trans. Device Mater. Rel.*, Vol. 12, No. 1, pp. 101-106.

16. Sanchez-Macian A, Reviriego P and Maestro J A (2014), "Hamming SEC-DAED and Extended Hamming SEC-DED-TAED Codes Through Selective Shortening and Bit Placement", *IEEE Trans. Device Mater. Rel.*, to be Published.

17. Yahagi Y, Yamaguchi H, Ibe E, Kameyama H, Sato M, Akioka T and Yamamoto S (2007), "A Novel Feature of Neutron-Induced Multi-Cell Upsets in 130 and 180 nm SRAMs", *IEEE Trans. Nucl. Sci.*, Vol. 54, No. 4, pp. 1030-1036.

18. Zhu M, Xiao L Y, Song L L, Zhang Y J and Luo H W (2011), "New Mix Codes for Multiple Bit Upsets Mitigation in Fault-Secure Memories", *Microelectron. J.*, Vol. 42, No. 3, pp. 553-561.