

Low Delay Single Symbol Error Correction Codes based on Reed Solomon Codes

Salvatore Pontarelli, Pedro Reviriego, Marco Ottavi and Juan Antonio Maestro

Abstract— To avoid data corruption, Error Correction Codes (ECCs) are widely used to protect memories. ECCs introduce a delay penalty in accessing the data as encoding or decoding has to be performed. This limits the use of ECCs in high-speed memories. This has led to the use of simple codes such as Single Error Correction Double Error Detection (SEC-DED) codes. However, as technology scales Multiple Cell Upsets (MCUs) become more common and limit the use of SEC-DED codes unless they are combined with interleaving. A similar issue occurs in some types of memories like DRAM that are typically grouped in modules composed of several devices. In those modules, the protection against a device failure rather than isolated bit errors is also desirable. In those cases, one option is to use more advanced ECCs that can correct multiple bit errors. The main challenge is that those codes should minimize the delay and area penalty. Among the codes that have been considered for memory protection are Reed-Solomon (RS) codes. These codes are based on non-binary symbols and therefore can correct multiple bit errors. In this paper, single symbol error correction codes based on Reed-Solomon (RS) codes that can be implemented with low delay are proposed and evaluated. The results show that they can be implemented with a substantially lower delay than traditional single error correction RS codes.

Index Terms— Error correction codes, Reed-Solomon codes, DRAM memory module, soft errors.

I. INTRODUCTION

DATA corruption caused by errors is a major issue in memories. Errors can be caused for example by radiation induced soft errors that affect one or more memory cells and change their values. Other types of failures cause permanent damage such that the device no longer provides correct data. To ensure that data is not corrupted when failures occur, Error Correction Codes (ECCs) are widely used in memories [1].

Manuscript received 26-September-2012 and in revised form 23-August-2013. This work was supported by the Spanish Ministry of Science and Education under Grant AYA2009-13300-C03. M. Ottavi is funded by the Italian Ministry for University and Research; Program "Incentivazione alla mobilità di studiosi stranieri e italiani residenti all'estero", D.M. n.96, 23.04.2001. This paper is part of a collaboration in the framework of COST ICT Action 1103 "Manufacturable and Dependable Multicore Architectures at Nanoscale".

S. Pontarelli and M. Ottavi are with University of Rome "Tor Vergata", Via del Politecnico 1 - 00133 Rome, Italy (phone +39-0672597344; email: {pontarelli, ottavi}@ing.uniroma2.it).

P. Reviriego and J.A. Maestro are with Universidad Antonio de Nebrija, C/ Pirineos, 55 E-28040 Madrid, Spain (phone: +34-914521100; fax: +34-914521110; email: {previrie, jmaestro}@nebrja.es).

ECCs add some additional parity check bits to each memory word such that errors can be detected and corrected. These additional bits reduce the effective capacity of the memory. Other overheads introduced by the ECC are the encoding and decoding circuitry. This circuitry has an impact also on the delay as the data has to be encoded when writing into the memory and decoded when reading from it. In most cases, the decoding is more complex than the encoding and limits the speed of the ECC [2].

Traditionally Single Error Correction Double Error Detection (SEC-DED) codes are used to protect memories [3]. These codes have a minimum Hamming distance of four such that single bit errors can be corrected while double errors are detected and not miscorrected. SEC-DED codes can be implemented with a relatively low area and delay overhead and some optimizations have been proposed in recent years [4],[5],[6]. Multiple bit errors are an issue when SEC-DED codes are used. When multiple errors affect cells that are physically close, as is the case of radiation induced Multiple Cell Upsets (MCUs) [7], SEC-DED codes can be combined with interleaving to ensure that the errors affect only one bit per logical word. That is also the case when an error causes the malfunction of a device in a memory module. In that case, the word is divided in sub-blocks and an ECC is used for each of them. Then the sub-blocks are interleaved in the devices such that in a device there is only one bit of a given sub-block [8]. However, the use of interleaving has an impact on the memory design and can increase area and power [9]. For a memory module, the use of interleaving increases the number of parity check bits required, as additional bits are required per each of the sub-blocks. Finally, when multiple errors are caused by independent error events, more powerful ECCs are needed to ensure the correction of errors [10].

A wide number of multiple bit ECCs have been proposed to protect memories. These include Bose-Chaudhuri-Hocquenghem (BCH) [11], Euclidean Geometry [12], Difference Set [13], Orthogonal Latin Squares [14] and Reed-Solomon codes [15]. Reed-Solomon (RS) codes have a distinct feature when compared with the other codes: they are not binary. They use symbols from a Galois Field such that each symbol is represented by multiple bits. Therefore a SEC RS code can correct multiple bit errors as long as they affect a single symbol [16]. This is very attractive for memory modules as when the number of bits in the devices matches those of the symbols in the RS code, failures in one device can

be corrected. In fact, for this reason RS codes are commonly used to protect main memories in computer systems for space applications, such as those described in [17],[18].

In general, the data that forms an RS codeword are considered as polynomial coefficients with values belonging to the Galois Field. The polynomial corresponding to a codeword is a multiple of a specific polynomial, called generator polynomial $g(x)$. Interested readers can refer to a classical textbook on error control codes (e.g. [19]) for further details.

In the case of SEC RS codes, the codeword is composed by appending two check symbols to a dataword of k symbols. To define the SEC RS codes, usually the matrix representation is preferred, thus the symbols composing the codeword are vectors of Galois Field elements. As for binary codes, a code is defined using a matrix H , called parity check matrix. A codeword is a vector v , such that $Hv=0$. The encoding process of a data vector d is performed starting from a generator matrix G , by computing $Gd = v$, which ensures that $Hv=0$. Since the SEC RS code is usually separable, the G matrix assumes the form $G = \begin{bmatrix} I_k \\ P \end{bmatrix}$, where I_k is a $k \times k$ identity matrix and P is of dimension $k \times 2$.

As with other advanced ECCs one issue for the use of RS codes in memories is the delay introduced by the decoding. In the case of RS codes, several arithmetic operations in the Galois Field are needed to encode or decode a block. This results in a much larger delay than that of traditional SECDED codes. To mitigate the impact on delay when using advanced ECCs, one option is to perform error detection first and only when errors are detected proceed to the correction phase [19]. As errors are rare, the average delay will be close to that of the error free case, which is given by the time required to perform error detection only. This is much lower than the time needed for correction. However, even with this modification delay can be large as the error detection time for RS codes can be significant.

In this paper, modifications to traditional RS codes to reduce the error detection delay are presented. This reduction would effectively reduce the delay to access the data when the mentioned scheme of performing error detection first and proceeding to the next phases of decoding only if there are errors is used. The proposed schemes are also implemented and compared with traditional RS codes. The results show that significant reductions in delay can be achieved with the proposed modifications. The rest of the paper is organized as follows. Section II describes SEC RS codes analyzing in detail the decoder. In section III, the proposed SEC codes based on RS codes are presented. Then in section IV, the new codes are evaluated in terms of area and delay and compared with existing SEC RS codes. Finally, the conclusions of this work are summarized in section V.

II. SINGLE ERROR CORRECTION (SEC) REED SOLOMON (RS) CODES

Reed Solomon codes are a subclass of non binary BCH codes constructed with symbols from a Galois Field $GF(q)$

where q is typically a power of two. For $q = 2^m$, m bit symbols are used to construct the code. An RS code has the following parameters: maximum block length $n = q-1$, number of parity check symbols $n-k = 2t$ and minimum distance $d_{min} = 2t+1$. All those parameters are expressed in terms of q -ary symbols. As a symbol has m bits, the maximum block length in bits is $m(q-1)$ and the number of parity check bits $2mt$. When $t = 1$, the minimum distance is three and therefore the code can correct single symbol errors. These errors can affect multiple bits as long as all of them belong to the same symbol. RS codes are commonly expressed as $RS(n,k,m)$. The parity check matrix for an RS code is constructed as shown in equation 1 where α is a primitive element in $GF(q)$.

$$H = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \dots & (\alpha^2)^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & \dots & (\alpha^{2t})^{n-1} \end{bmatrix} \quad (1)$$

For a Single Error Correction (SEC) code only two parity check symbols are needed and therefore the matrix is simply:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} \end{bmatrix} \quad (2)$$

The decoding of a SEC RS code starts with the computation of the syndrome vector, which for a block r is simply:

$$s = H \cdot r \quad (3)$$

The syndrome vector can be used to detect errors as follows. When all the bits in the vector are zero, no error is detected. When at least one bit is one, an error is detected and therefore error correction has to be performed.

The delay and complexity of syndrome computation depends on the values of the parity check matrix. For example, for the first row of (1) the calculation is simple as all values are ones while for the second several multiplications are needed.

Assuming a single symbol error e in position i in the block, the syndrome would be:

$$s = \begin{bmatrix} e \\ e \cdot \alpha^i \end{bmatrix} \quad (4)$$

therefore for a single symbol error if the second symbol of the syndrome is divided by the first, the value α^i is obtained. The error is then located by finding the exponent of the quotient and can be finally corrected by subtracting e from the i^{th} symbol. Correction therefore requires one division and one logarithm operation. As the symbols are from $GF(q)$, all operations are done over that field. This means that the decoding becomes more complex as $q = 2^m$ grows.

From equations (2) and (4) it is easy to understand which is the maximum block length of a SEC RS code. Suppose that we want to add a further column in the form $\begin{bmatrix} 1 \\ \alpha^n \end{bmatrix}$ to the matrix in equation (2). Since in the Galois Field $GF(q)$ $\alpha^n = \alpha^{q-1} = 1$, an error in this column would produce the same syndrome of

an error in the first column of the parity check matrix. This aliasing means that there are uncorrectable errors when the codeword length exceeds n .

However, there is a well known extension of the SEC RS code [19], to allow extending the maximum block length up to $q+1$ symbols. This can be done adding the columns $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ to the H matrix, obtaining the following matrix:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 & 1 & 0 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} & 0 & 1 \end{bmatrix} \quad (5)$$

The columns of the H matrix are still linearly independent and the decoding procedure must be modified to manage the cases in which one of the syndromes is zero, that correspond to an error in the last two codeword symbols.

In summary, the decoding of SEC RS codes requires a number of complex operations over $GF(q)$ especially for error correction where division and logarithm need to be implemented. Syndrome computation only requires multiplications and additions and its delay depends on the values of each of the rows in the H matrix.

As discussed before, it is possible to reduce data access delay by performing error detection first and proceed to correction only when there are errors. In this case, data access delay will be limited by the time needed to compute the syndrome.

III. PROPOSED CODES

In this section, two modified SEC RS codes are presented to reduce the delay for encoding and for error detection. The first modification tries to optimize the parity check matrix to reduce the delay for encoding and syndrome computation. The second combines the first with an extension of the use of RS codes for a given a value of q to enable longer block lengths. This reduces the encoding and decoding delay further as the multiplications and other arithmetic operations are done over a simpler Galois Field.

a) Optimizing parity check matrix

As explained in the previous section, for a SEC-RS code, the computation of the parity check matrix is unbalanced. The multiplication of the first row of the matrix for a block r requires no multiplications on the Galois Field while the second requires $n-1$ multiplications on the Galois Field. The first proposed modification to RS codes tries to balance the complexity of both calculations. This is done by using the following parity check matrix:

$$H = \begin{bmatrix} 1 & 1 & \alpha^{-2} & 1 & \dots & \alpha^{-(n-1)} & 1 & 0 \\ 1 & \alpha & 1 & \alpha^3 & \dots & 1 & 0 & 1 \end{bmatrix} \quad (6)$$

and the corresponding parity generator matrix:

$$G = \begin{bmatrix} 1 & 1 & \alpha^{-2} & \dots & \alpha^{-(n-2)} \\ 1 & \alpha & 1 & \dots & 1 \end{bmatrix} \quad (7)$$

The matrix of equation (6) can be directly obtained from equation (5) multiplying the i -th columns, when i is even, for

the value α^{-i} . The modified code is still a SEC code as the columns are linearly independent. With this modifications the computation of each of the check symbols during the encoding process, and of the syndrome symbols during the decoding process, requires the same number of multiplications. This reduces the length of the critical path and therefore lowers the delay.

The decoding for the proposed codes is similar to that of SEC RS codes. Suppose to have an error of magnitude e in the i^{th} symbol. If the syndrome vector $[S_0, S_1] = [S_0, 0]$, then $i=n-1$ and $e=S_0$. If $[S_0, S_1] = [0, S_1]$, then $i=n$ and $e=S_1$. Otherwise, the syndrome vector will be $[S_0, S_1] = [e, e\alpha^i]$ when i is even and $[S_0, S_1] = [e\alpha^{-i}, e]$ when i is odd. Therefore, $S_1/S_0 = \alpha^i$ for all $i < n-1$ and we are able to compute the error location. Then we can retrieve $e=S_0$ for even values of i , and $e=S_1$ for odd values of i and correct the error. This process requires one division and one logarithm, the same as the decoding of traditional SEC RS codes.

b) Extension to longer block lengths

Another factor that impacts the delay of RS encoders and decoders is the size of the Galois Field that is used to construct the code. This is due to the increase of the complexity of the arithmetic operations for larger Galois Fields. As explained in the previous section, for a given $GF(q)$ the maximum block size of a traditional RS code is $q-1$ symbols. This limitation forces the use of larger Galois Fields for large block sizes thus increasing the decoding delay. This issue can be solved by using a modified SEC RS code with the following H matrix:

$$H = \begin{bmatrix} \alpha & 1 & 1 & \alpha^2 & 1 & 1 & \dots & 1 & 0 & 0 \\ 1 & \alpha & 1 & 1 & \alpha^2 & 1 & \dots & 0 & 1 & 0 \\ 1 & 1 & \alpha & 1 & 1 & \alpha^2 & \dots & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

With this modification, the block length can be up to $3(q-1)$ symbols. The additional row allows to discriminate between syndrome vectors that otherwise could have the same syndrome value, as is shown in the following example:

Example: Suppose we have $m=3$ bits, and thus $q=8$ and $n=7$, and suppose we have an error of magnitude $e=\alpha^6$ in the first symbol (the one corresponding to the H column $\begin{bmatrix} \alpha \\ 1 \\ 1 \end{bmatrix}$). The

corresponding syndrome would be $s_1 = \begin{bmatrix} 1 \\ \alpha^6 \\ \alpha^6 \end{bmatrix}$. Now suppose to

have an error of magnitude $e=1$ on the 17^{th} symbol, that corresponds to the H column $\begin{bmatrix} 1 \\ \alpha^6 \\ 1 \end{bmatrix}$. In this case the syndrome

vector would be $s_2 = \begin{bmatrix} 1 \\ \alpha^6 \\ 1 \end{bmatrix}$. We can see that the two

syndromes differ only on the third vector coordinate. This additional information allows to discriminate between syndromes that otherwise would be identical.

The decoding is similar to the previous case. Suppose we have an error of magnitude e in the i^{th} symbol. If the syndrome vector $[S_0, S_1, S_2]$ contains two zeros, the error is in the

corresponding check symbol. Otherwise, two syndrome symbols will have the same value, corresponding to the error magnitude e . The third syndrome symbol (the one with a value different from the other two) will have a value of $e \cdot \alpha^{ceil(i/3)}$ and therefore the error location can be obtained. More precisely, if we denote S_a (with $a = 0, 1$ or 2) this syndrome symbol, the location i is $i = 3 * (\log(S_a / e) - 1) + a$. The decoding takes again some comparisons, a division and a logarithm. The proposed codes have been validated using random errors to ensure that all single symbol errors are corrected. The encoder and decoder corresponding to the above presented codes has been designed in VHDL and synthesized, and the results are presented in the next section.

IV. EVALUATION

In this section, the proposed modified RS codes are evaluated and compared with existing RS codes. The application considered, similarly to the scenario of [17],[18] is the protection of memory modules with data widths of 64 or 128 bits commonly used in computing applications. It is reasonable to assume that the modules are built using 8 bit memory devices and the protection requirements are to be able to correct any error that affects a single device [8]. For those configurations, a straightforward solution would be to use a SEC RS code with eight bit symbols (i.e. $m = 8$). For 64 bits, the code would be a SEC RS(10,8,8) code and for 128 bits a SEC RS(18,16,8) code. In the first case to reduce the decoding delay, one option is to use two interleaved SEC RS(10,8,4) codes such that each device stores one symbol from each of the codes. This ensures that errors affecting a single device would cause at most a single symbol error in each code and therefore the errors can be corrected. As discussed before using 4 bit symbols simplifies the arithmetic operations and therefore lowers the delay.

For the modified codes, the first modification described in section III.a will be denoted as SEC RS_{mod1} and the second modification as SEC RS_{mod2}. In the case of the first modification, the same code parameters as for the traditional SEC codes can be used. Those are, for the 64 data bit module, a SEC RS_{mod1}(10,8,8) code or two interleaved SEC RS_{mod1}(10,8,4) codes and for the 128 data bit module a SEC RS_{mod1}(18,16,8) code. The second modification enables the use of two interleaved 4 bit symbols SEC RS_{mod2}(19,16,4) codes for the 128 bits module. The parameters for the scenarios considered are summarized in Table I. It can be observed that for a given number of data bits the number of parity check bits is the same except for the 2x interleaved SEC-RS_{mod2}(19,16,4) code. This means that only the encoder and decoder need to be changed to use the proposed modifications.

TABLE I
PARAMETERS OF THE CODES EVALUATED

Code	Data bits	Parity Check Bits	GF(q)
SEC-RS(10,8,8)	64	16	2^8
SEC-RS _{mod1} (10,8,8)	64	16	2^8
2xSEC-RS(10,8,4)	64	16	2^4
2xSEC-RS _{mod1} (10,8,4)	64	16	2^4
SEC-RS(18,16,8)	128	16	2^8
SEC-RS _{mod1} (18,16,8)	128	16	2^8
2xSEC-RS _{mod2} (19,16,4)	128	24	2^4

To evaluate the delay and area of the different codes, they were implemented in HDL and synthesized for a 45nm library [20]. As discussed before, error correction is more complex than error detection and therefore delay can be reduced by performing error detection first so that correction is done only when errors are detected. This was the scheme used in the implementation. For this implementation the delay in the error free case is given by that of syndrome computation. The clock cycle duration is adjusted to that speed. In case of error, the delay will be larger and is given in the number of clock cycles required to correct the error. In some cases, the error correction may also limit the speed even when implemented in multiple clock cycles.

The delay and area results for the encoder and for the syndrome computation block are presented in Tables II and III. The delay is given in nanoseconds and the area in μm^2 . For the encoder it can be observed that for 64 data bits the proposed modification provides savings of 6.25% when 8 bit symbols are used and of 16.13% for 4 bit symbols. For 128 bits, the reduction is 12.77% when the first modification is used and 31.91% when the second one is used. The area of the 64 bits encoder is slightly increased in case of $m=8$ (+9%) and slightly decreased when $m=4$,(-4.5%). For the 128 bits encoder, the area saving is 36% for 8 bits symbols, and 4% when the RS(19,16,4) code is used.

For the 64 bits syndrome computation block, the delay saving is 12% when 8 bit symbols are used and of 10% for 4 bit symbols. For the 128 bits syndrome calculation, the reduction is 12.5% when the modified RS_{mod1}(18,16,8) is used and 33% when RS_{mod2}(19,16,4) is used.

The area of the 64 bits syndrome computation increases by 11% and 2% for $m=8$ and $m=4$ respectively. Finally, the area of the 128 bits syndrome computation block shows a 18% reduction when the modified RS_{mod1}(18,16,8) is used and 2.5% reduction when RS_{mod2}(19,16,4) is used.

In summary, the proposed codes consistently reduce the encoding and syndrome computation delay. The larger reductions (over 30%) are obtained when the second modification is used as for 128 bits the arithmetic operations are done over $GF(2^4)$ rather than over $GF(2^8)$.

TABLE II
ENCODER AND SYNDROME COMPUTATION DELAY

Code	Data bits	Encoder	Syndrome
SEC-RS(10,8,8)	64	0.32	0.32
SEC-RS _{mod1} (10,8,8)	64	0.30	0.28
2xSEC-RS(10,8,4)	64	0.31	0.31
2xSEC-RS _{mod1} (10,8,4)	64	0.26	0.28
SEC-RS(18,16,8)	128	0.47	0.48
SEC-RS _{mod1} (18,16,8)	128	0.41	0.42
2xSEC-RS _{mod2} (19,16,4)	128	0.32	0.32

TABLE III
ENCODER AND SYNDROME COMPUTATION AREA

Code	Data bits	Encoder	syndrome
SEC-RS(10,8,8)	64	769	876
SEC-RS _{mod1} (10,8,8)	64	840	978
2xSEC-RS(10,8,4)	64	704	746
2xSEC-RS _{mod1} (10,8,4)	64	672	762
SEC-RS(18,16,8)	128	1902	2112
SEC-RS _{mod1} (18,16,8)	128	1210	1731

2xSEC-RS _{mod2} (19,16,4)	128	1830	2062
------------------------------------	-----	------	------

REFERENCES

For completeness, Tables IV and V show the area and delay parameters for the error correction block of the decoders. Since an aggressive pipeline is used to maintain high speed, the area of both the combinatorial and sequential parts of the circuit has been reported. It can be seen that the use of 4 bits symbols, allows a big area saving, since less pipeline stages are required. As for the speed of the error correction block, the decoder with 4 bits symbols can be always clocked at the maximum frequency given by the syndrome computation block, and can provide the result of the error correction procedure after 4 clock cycles. In particular, the proposed RS_{mod1}(10,8,4) achieves the best timing results for the 64 bits codeword, while the RS_{mod2}(19,16,4) outperforms the 8 bits symbol based codes, both in terms of clock cycle delay and number of cycles. In fact, all the 8 bits symbol codes require 10 clock cycles to achieve high operating frequencies, and in the case of the SEC-RS(10,8,8) and SEC-RS_{mod1}(10,8,8), they were unable to reach the operating frequency of the syndrome computation block, thus limiting the maximum operating frequency of the whole decoder. In any case, the SEC-RS_{mod1}(10,8,8) code outperforms the standard RS(10,8,8) code with a delay of 0.39ns compared to 0.43ns.

TABLE IV
ERROR CORRECTION BLOCK AREA

Code	Combinational	Flip-flops	Total
SEC-RS(10,8,8)	7379	8448	15827
SEC-RS _{mod1} (10,8,8)	7506	8472	15978
2xSEC-RS(10,8,4)	3762	1754	5516
2xSEC-RS _{mod1} (10,8,4)	3836	1786	5622
SEC-RS(18,16,8)	8645	12134	20779
SEC-RS _{mod1} (18,16,8)	8709	12908	21617
2xSEC-RS _{mod2} (19,16,4)	6650	2904	9554

TABLE IV
ERROR CORRECTION BLOCK DELAY

Code	Cycle delay	Number of cycles
SEC-RS(10,8,8)	0.43 (*)	10
SEC-RS _{mod1} (10,8,8)	0.39 (*)	10
2xSEC-RS(10,8,4)	0.31	4
2xSEC-RS _{mod1} (10,8,4)	0.28	4
SEC-RS(18,16,8)	0.45	10
SEC-RS _{mod1} (18,16,8)	0.41	10
2xSEC-RS _{mod2} (19,16,4)	0.32	4

(*) maximum operating frequency of the decoder depends on this delay

V. CONCLUSIONS

In this paper, new codes based on modifications of Single Error Correction Reed Solomon (SEC RS) codes have been proposed with the objective of reducing delay. The codes have been implemented and evaluated. The examples used for evaluation correspond to real configurations commonly used in memory modules. For those, the proposed codes enable significant delay reductions in encoding and decoding delay. This makes the modified codes attractive for high-speed memories. Future work will consider the evaluation of the proposed codes to protect other types of memory like for examples caches.

- [1] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-the-art review", *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 124-134, 1984.
- [2] S. Lin and D. J. Costello, "Error Control Coding, 2nd Ed.", Englewood Cliffs, NJ: Prentice-Hall, 2004.
- [3] M.Y. Hsiao "A Class of Optimal Minimum Odd-weight column SEC-DED codes", *IBM Journal of Research and Development*, vol. 14, pp.395-301, Jul. 1970.
- [4] M. Richter, K. Oberlaender, and M. Goessel, "New linear SEC-DED codes with reduced triple bit error miscorrection probability", in *IEEE On-Line Testing Symposium*, pp. 37-42, 2008.
- [5] V. Gherman, S. Evain, N. Seymour and Y. Bonhomme, "Generalized parity-check matrices for SEC-DED codes with fixed parity", in *IEEE On-Line Testing Symposium*, pp. 198 - 20, 2011.
- [6] P. Reviriego, S. Pontarelli, J. A. Maestro and M. Ottavi, "A Method to Construct Low Delay Single Error Correction Codes for Protecting Data Bits Only", *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 3, pp. 479 - 483, 2013,
- [7] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo and T. Toba, "Impact of Scaling on Neutron-Induced Soft Error in SRAMs From a 250 nm to a 22 nm Design Rule", *IEEE Trans. on Electron Devices*, vol. 57, no. 7, pp. 1527 - 1538, 2010.
- [8] T. J. Dell, "A white paper on the benefits of chipkill-correct ECC for PC server main memory," *IBM Microelectronics Division*, July 1997.
- [9] A. Dutta and N.A. Touba, "Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC ode", 25th *IEEE VLSI Test Symposium*, 2007, pp. 349 - 354.
- [10] M. A. Bajura et al., "Models and Algorithmic Limits for an ECC-Based Approach to Hardening Sub-100-nm SRAMs", *IEEE Trans. On Nuclear Science*, vol. 54, no. 4, pp. 935-945, 2007.
- [11] P. Ankolekar, S. Rosner, R. Isaac and J. Bredow, "Multi-bit error correction methods for latency-constrained flash memory systems," *IEEE Trans. on Device and Materials Reliability*, vol. 10, no. 1, pp. 33-39, 2010.
- [12] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault-tolerant nano-scale memory", in *Proc. of Foundations of Nanoscience (FNANO07)*, Snowbird, Utah, 2007.
- [13] S. Liu, P. Reviriego and J. A. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 20, no. 1, pp 148-156, 2012.
- [14] M.Y. Hsiao, D.C. Bossen, and R.T. Chien, "Orthogonal Latin Square Codes," *IBM J. Research and Development*, vol. 14, no. 4, pp. 390-394, 1970.
- [15] G.C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Data integrity evaluations of Reed Solomon codes for storage systems," Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004.
- [16] G.C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Fault tolerant solid state mass memory for space applications", *IEEE Trans. on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1353-1372, Oct. 2005.
- [17] F. Koebel and J.-F. Coldefy, "SCOC3: a space computer on a chip: an example of successful development of a highly integrated innovative ASIC", in *Proc. of the Conference on Design, Automation and Test in Europe*, (DATE), 2010.
- [18] Maxwell SCS750 data sheet, available online at: http://www.maxwell.com/products/microelectronics/docs/scs750_rev7.pdf
- [19] J. Castiñeira Moreira, P. Guy Farrell, *Essentials of Error-Control Coding*, Wiley, 2006.
- [20] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu., D. Somasekhar and S. Lu, "Reducing Cache Power with Low Cost, Multi-bit Error-Correcting Codes", *International Symposium on Computer Architecture*, pp. 83-93, Jun. 2010.
- [21] J. E. Stine et al., "FreePDK: An open-source variation-aware design kit," in *IEEE Int. Conf. Microelectronic Systems Education, (MSE'07)*, Jun. 2007, pp. 173-174.

