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 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

1 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].

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 1 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 1 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 $\mathrm{G} = [{}^{I_k}_{P}]$, 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 SEC-DED 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 2 describes SEC RS codes analyzing in detail the decoder. In Section 3, the proposed SEC codes based on RS codes are presented. Then in Section 4, the new

- *S. Pontarelli and M. Ottavi are with the Department of Electronic Engineering, University of Rome Tor Vergata, Via del Politecnico 1, 00133 Rome, Italy. E-mail: {pontarelli, ottavi}@ing.uniroma2.it.*
- *P. Reviriego and J.A. Maestro are with the Department of Ing INformatica, Universidad Antonio de Nebrija, C/ Pirineos, 55 E-28040 Madrid, Spain. E-mail: {previrie, jmaestro}@nebrija.es.*

Manuscript received 26 Sept. 2012; revised 23 Aug. 2013; accepted 17 Apr. 2014. Date of publication 7 May 2014; date of current version 8 Apr. 2015.
Recommended for acceptance by M. Fujita.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2322599

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 5.

2 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 $GF(q)$:

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

For a single error correction code only two parity check symbols are needed and therefore the matrix is simply:

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

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 1 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 ith 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 $\left[\begin{smallmatrix} 1 \\ \alpha^n \end{smallmatrix}\right]$ 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 $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ to the H matrix, obtaining the following matrix:

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

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.

3 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.

3.1 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:

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

and the corresponding parity generator matrix:

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

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 ith 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,

TABLE 1
Parameters of the Codes Evaluated

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

TABLE 2
Encoder and Syndrome Computation Delay

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

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.

3.2 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:

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

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 $[\begin{smallmatrix}\alpha\\1\\1\end{smallmatrix}]$). The corresponding syndrome would be $s_1 = [\begin{smallmatrix}1\\\alpha^6\\1\end{smallmatrix}]$. Now suppose to have an error of magnitude $e = 1$ on the $17th$ symbol, that corresponds to the H column $[\begin{smallmatrix}1\\\alpha^6\\1\end{smallmatrix}]$. In this case the syndrome vector would be $s_2 = [\begin{smallmatrix}1\\\alpha^6\\1\end{smallmatrix}]$. 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 ith 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.

4 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 to assume that the modules are built using 8bit 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 8 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 3.1 will be denoted as SEC RS$_{\text{mod1}}$ and the second modification as SEC RS$_{\text{mod2}}$. 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$_{\text{mod1}}$(10, 8, 8) code or two interleaved SEC RS$_{\text{mod1}}$(10, 8, 4) codes and for the 128 data bit module a SEC RS$_{\text{mod1}}$(18, 16, 8) code. The second modification enables the use of two interleaved 4 bit symbols SEC RS$_{\text{mod2}}$(19, 16, 4) codes for the 128 bits module. The parameters for the scenarios considered are summarized in Table 1. It can be observed that for a given number of data bits the number of parity check bits is the same except for the $2\times$ interleaved SEC-RS$_{\text{mod2}}$(19, 16, 4) code. This means that only the encoder and decoder need to be changed to use the proposed modifications.

To evaluate the delay and area of the different codes, they were implemented in HDL and synthesized for a 45 nm 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 2 and 3. 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 percent when 8 bit symbols are used and of 16.13 percent for 4 bit symbols. For 128 bits, the reduction is 12.77 percent when the first modification is used and 31.91 percent 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

TABLE 3
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

36 percent for 8 bits symbols, and 4 percent when the RS (19, 16, 4) code is used.

For the 64 bits syndrome computation block, the delay saving is 12 percent when 8 bit symbols are used and of 10 percent for 4 bit symbols. For the 128 bits syndrome calculation, the reduction is 12.5 percent when the modified $\mathrm{RS}_{\mathrm{mod1}}(18, 16, 8)$ is used and 3 percent when $\mathrm{RS}_{\mathrm{mod2}}(19, 16, 4)$ is used.

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

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

For completeness, Tables 4 and 5 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 $\mathrm{RS}_{\mathrm{mod1}}(10, 8, 4)$ achieves the best timing results for the 64 bits codeword, while the $\mathrm{RS}_{\mathrm{mod2}}(19, 16, 4)$ 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-$\mathrm{RS}_{\mathrm{mod1}}(10, 8, 8)$, 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-$\mathrm{RS}_{\mathrm{mod1}}(10, 8, 8)$ code outperforms the standard RS (10, 8, 8) code with a delay of 0.39 ns compared to 0.43 ns.

5 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.

TABLE 4
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 5
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		

Acknowledgments

This work was supported by the Spanish Ministry of Science and Education under Grant AYA2009-13300-C03. Marco 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".

References

[1] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM J. Res. Develop.*, vol. 28, no. 2, pp. 124–134, 1984.
[2] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.
[3] M. Y. Hsiao, "A class of optimal minimum odd-weight column SEC-DED codes," *IBM J. Res. Develop.*, 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 *Proc. IEEE 14th Int. On-Line Testing Symp.*, 2008, pp. 37–42.
[5] V. Gherman, S. Evain, N. Seymour, and Y. Bonhomme, "Generalized parity-check matrices for SEC-DED codes with fixed parity," in *Proc. IEEE 17th Int. On-Line Testing Symp.*, 2011, pp. 198–201.
[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. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 32, no. 3, pp. 479–483, Mar. 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. Electron. Devices*, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.
[8] T. J. Dell, "A white paper on the benefits of chipkill-correct ECC for PC server main memory," IBM Microelectron. Division, Jul. 1997.
[9] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in *Proc. 25th IEEE VLSI Test Symp.*, 2007, pp. 349–354.
[10] M. A. Bajura, Y. Boulghassoul, R. Naseer, S. DasGupta, A. F. Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill, and J. N. Damoulakis, "Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs," *IEEE Trans. Nuclear Sci.*, vol. 54, no. 4, pp. 935–945, Aug. 2007.
[11] P. Ankolekar, S. Rosner, R. Isaac, and J. Bredow, "Multi-bit error correction methods for latency-constrained flash memory systems," *IEEE Trans. Device Mater. Rel.*, vol. 10, no. 1, pp. 33–39, Mar. 2010.
[12] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault-tolerant nano-scale memory," *presented at the Foundations of Nanoscience*, Snowbird, Utah, USA, 2007.
[13] S. Liu, P. Reviriego, and J. A. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 1, pp. 148–156, Jan. 2012.
[14] M. Y. Hsiao, D. C. Bossen, and R. T. Chien, "Orthogonal Latin square codes," *IBM J. Res. Develop.*, 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," in *Proc. IEEE 19th Int. Defect Fault Tolerance VLSI Syst.*, 2004, pp. 158–164.

[16] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Fault tolerant solid state mass memory for space applications," *IEEE Trans. Aerosp. Electron. Syst.*, 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. Conf. Des., Autom. Test Eur.*, 2010, pp. 1345–1348.

[18] R. Hillmand, G. Swift, et al., "Space processor radiation mitigation and validation techniques for an 1800 MIPS processor board," In *Proc. 7th IEEE Eur. Conf. Radiat. Effects Components Syst. (RADECS)*, 2003, pp. 347–352.

[19] J. Castiñeira Moreira and P. Guy Farrell, *Essentials of Error-Control Coding*. Hoboken, NJ, USA:: 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," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, Jun. 2010, pp. 83–93.

[21] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An open-source variation-aware design kit," in *Proc. IEEE Int. Conf. Microelectron. Syst. Edu.*, Jun. 2007, pp. 173–174.