Improving Memory Reliability Against Soft Errors Using Block Parity

P. Reviriego, C. Argyrides, J. A. Maestro, and D. K. Pradhan

***Abstract*—Memory reliability is an important issue. The continuous scaling of transistor technology enables the use of larger memories making soft errors more likely to occur. To ensure that those errors do not cause data corruption, error correcting codes (ECC) are commonly used. Single error correction-double error detection codes (SEC-DED) are typically implemented in each memory word, so that a single error in a word can be corrected and two errors can be detected. In this paper, a technique to improve the reliability of memories that use SEC-DED is studied. The proposed technique shows that it is possible to substantially improve the mean time to failure (MTTF) of the memory at the cost of increasing the access time for writing operations.**

***Index Terms*—Error correcting codes, MTTF, reliability.**

I. Introduction

RELIABILITY is a critical factor for memories [1] when they operate in environments where there are many sources of error [2], [3]. Radiation, for example, induces soft errors when a particle hits a memory cell and changes the stored value (what is called a bitflip). Traditionally, memories have been protected with single error correction (SEC) codes [4]–[7] that can correct up to one error per memory word. Per-word parity bits are also commonly used when the objective is only to detect errors. Considering SEC codes, and in order to avoid the accumulation of errors in the memory (which would eventually cause a failure when a second soft error affects a bit of a word already affected by a previous one), scrubbing is used [8]. The scrubbing process periodically reads the memory words and corrects the errors, so that only if two errors arrive in the same scrubbing period a failure occurs. This is usually a valid solution, but some times it may not be feasible or desirable. For example, considering a memory, if power saving is an issue, then scrubbing could not be the most convenient option, since it would increase power consumption.

Manuscript received September 06, 2010; revised November 22, 2010 and January 17, 2011; accepted January 25, 2011. Date of publication February 22, 2011; date of current version June 15, 2011. This work was supported in part by the Spanish Ministry of Science and Innovation under Grant AYA2009-13300-C03-01.

P. Reviriego and J. A. Maestro are with Universidad Antonio de Nebrija, 55 E-28040 Madrid, Spain (e-mail: previrie@nebrija.es; jmaestro@nebrija.es).

C. Argyrides is with C. A. evolvIT LTD, 3010, Limassol, Cyprus (e-mail: costas@computer.org).

D. K. Pradhan is with the University of Bristol, BS8 1UB, Bristol, U.K. (e-mail: pradhan@cs.bris.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2011.2109965

Another problem is that the constant technology shrinking enables for each technology node the use of larger memories. This means that errors are more likely to occur. This increased number of errors poses a threat to existing protection techniques [9] and highlights the need for more powerful protection.

Introducing codes able to correct a number of errors higher than one is an alternative [10], [11], but these codes need to be carefully chosen, as they usually introduce some drawbacks.

The most obvious is cost, since multi-bit protection implies additional redundancy. This cost includes the extra bits added to each word plus the correction and protection logic, which suffers an increment of complexity [12]. Another alternative is the use of smaller scrubbing periods but this also implies larger power consumption and reduced memory bandwidth for data operations.

In this paper, another alternative is studied, namely to increase the reliability of a memory by using some of its words to provide additional protection. The assumption is that the system designer is working with a memory that implements SEC and that does not meet the reliability requirements. The proposed technique shows how the reliability can be increased by dedicating some of the memory words for error detection and correction. The reliability achieved can be close to that of double error correction (DEC). One of the costs associated with this technique is that a write operation requires four memory accesses. However, in applications where writing operations are not limiting the memory bandwidth, this should not be an issue. The other overhead is caused by the number of words that are dedicated to error protection, which decreases the actual capacity of the memory for data. The overhead in terms of memory space is given by the number of words in each parity block used in the proposed method.

The proposed technique compares favorably with DEC in terms of decoding complexity and number of parity bits per word [12]. The reduced decoding complexity means that the access time could be reduced. The number of parity bits is significantly lower in SEC-DED codes than in DEC [12] as shown in Table I. This means that the area can be lower when using the proposed technique. However the technique incurs in overhead in terms of access time and area as mentioned before. Therefore, a detailed comparison between DEC and the proposed approach should be done for each specific design.

Finally, it is worth mentioning that the proposed technique is similar to the concatenated codes used in communications for many years [13] in which a first code corrects some of the errors and the second corrects the remaining ones. In our case the SEC-DED code corrects the single errors and the added block parity the double errors.

TABLE I
PARITY BITS FOR SEC-DED AND DEC CODES FOR DIFFERENT WORD LENGTHS

Data bits	SEC-DED parity bits	DEC parity bits
16	6	10
32	7	12
64	8	14
128	9	16

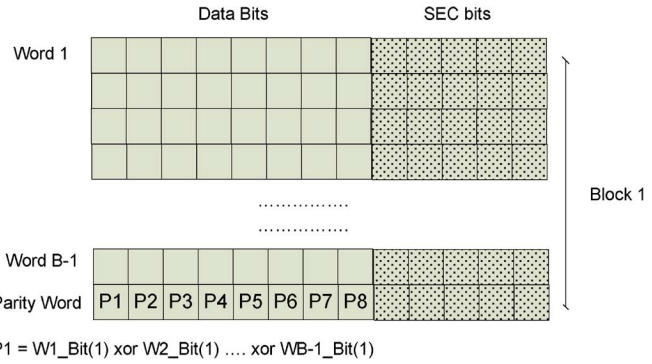

Fig. 1. Proposed block memory organization to improve the reliability of a memory that implements SEC.

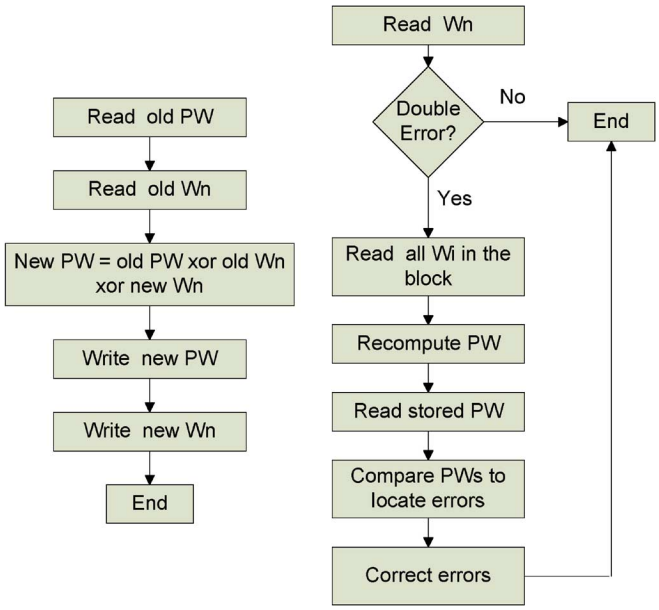

Fig. 2. Illustration of the write (left) and read (right) operations when implementing the proposed technique in a memory protected with SEC-DED codes.

The rest of the paper is organized as follows. The proposed technique is introduced in Section II; a reliability analysis of the proposed technique is presented in Section III; then, in Section IV, simulation results are reported showing the validity of the analysis. In Section V, the design tradeoffs in terms of the block size are discussed, and finally, some conclusions will be shared in Section VI.

II. PROPOSED TECHNIQUE

In this section, the proposed technique to improve reliability is presented for a memory that implements single error correction and double error detection (SEC-DED) [14], [15] in each memory word.

A. Introduction of the Parity Word

In this case, the reliability can be increased by introducing additional redundancy so that double errors can be corrected. To this end, the memory can be divided in blocks of size B words, where the first B-1 words are used for data storage and the last one contains a parity of the others, as shown in Fig. 1. More precisely, each data bit in the parity word, PW, contains the parity of the same bit for the previous B-1 words:

$$\mathrm{PW_i} = \mathrm{W1_i} \oplus \mathrm{W2_i} \oplus \ldots \oplus \mathrm{W(B-1)_i}.$$

This parity word would also be protected against single errors, as the rest of the memory, with SEC-DED codes. Each time a write operation is performed, the parity word must also be recalculated, in order to keep it up to date. The exact sequence for a write operation of word Wn is as follows.

- Read the old value of the word that is to be written, Old_Wn, and do an XOR with the new data, $\mathrm{New\_Wn}$: $\mathrm{Parity\_change} := \mathrm{Old\_Wn} \oplus \mathrm{New\_Wn}$.
- Read the old parity word and do an XOR of its value with the result of the first step. Write back the obtained value to the parity word: $\mathrm{New\_PW} := \mathrm{Old\_PW} \oplus \mathrm{Parity\_change}$.
- Update the parity PW and word Wn, with the new data.

The overall process is illustrated in the left part of Fig. 2.

B. Overhead in the Read and Write Operations

Each write operation takes two reads and two writes into the memory. On the other hand, for reading, there are two cases:

1) If the word has no error or a single error, then only one read is needed, as the SEC-DED code can make the correction.
2) If the word has two errors, then all B-1 words in the block are read to check the parity and detect which bits are in error. The error is then corrected by inverting those bits, as shown in the right part of Fig. 2.

Therefore, in most cases, only a word is read, except in the case when there are two errors (which require additional operations). In summary, the proposed technique incurs a significant overhead for writing and a negligible one for reading, since double errors are rare.

C. Correction Capability

To analyze the improvement of the proposed technique in memory reliability it should be noted that as long as there is only a word with two errors per block, there is no failure. The reason for this is that the parity word can be used to correct the double error, while single errors are corrected by the SEC-DED code.

As an example, a situation is considered in Fig. 3, in which a double error is present in word 1 and a single error in word 4.

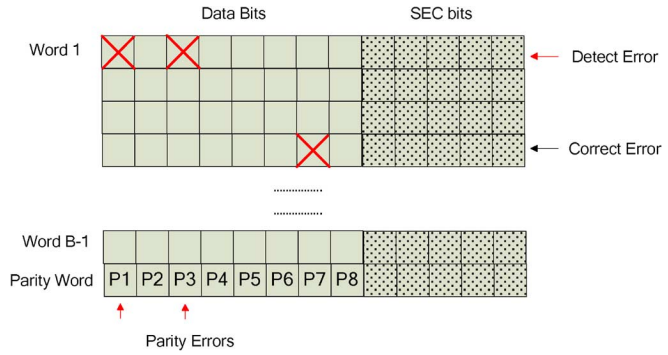

Fig. 3. Example of errors corrected by the proposed technique.

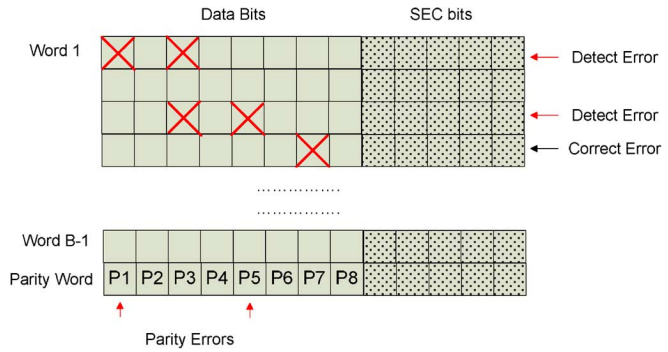

Fig. 4. Example of memory failure for the proposed technique.

In this case, the single error is corrected by the SEC code, while the double error is detected by the SEC-DED code and corrected using the parity word to locate the error positions. However, when there are two words with two errors each, the parity word cannot be used to correct these errors. This is illustrated in Fig. 4, where double errors are present in words 1 and 3. This causes parity errors on two bits, so that it is not possible to locate and correct them. In fact, for the errors in bit 3 the parity word does not even detect an error, since there are two of them in the same column. There is also a single error in word 4 that is corrected by the SEC-DED code, which does not cause errors on the parity bits. Therefore, in all cases where two words in a block have double errors, there will be a failure.

To summarize, the proposed technique will be able to correct double errors as long as there is only one per memory block.

III. Reliability Analysis

In this section, the reliability of the proposed technique is analyzed in terms of the mean time to failure (MTTF). This is done for the case in which scrubbing is not used. The analysis could be easily extended for the scrubbing case. In fact in that case the proposed technique will be closer to DEC in terms of MTTF as most failures would occur when there are few errors in the memory.

The following assumptions will be used in the following:

- memories do not implement scrubbing; errors accumulate in the memory until a failure occurs;
- error events are supposed to arrive following a Poisson distribution.

In order to characterize the quality of the protection codes, these parameters will be used:

- M: the number of words per memory;
- N: the number of data bits per word;
- B: the number of words in a block;
- c: the number of redundant bits per word;
- mean time to failure (MTTF): it represents how much time the memory should work on average until it fails;
- event arrival rate per bit (λ_{bit}): number of error events that are produced on average in the memory per bit and unit of time; the event arrival rate per memory will depend both on the previous parameter, the number of words in the memory and the number of bits in each word: $\lambda_{\text{memory}} = \lambda_{\text{bit}} \cdot M \cdot (\text{N} + \text{c})$;
- mean number of events to failure (METF): this represents how many error events the memory has suffered on average until reaching failure. Following a well-known relation for Poisson processes (see [16]):

$$\text{MTTF} = \text{METF}/\lambda_{\text{memory}} = \text{METF}/\left(\lambda_{\text{bit}} \cdot M \cdot (\text{N} + \text{c})\right). \tag{1}$$

The objective of the following derivation is to study the level of reliability of the proposed protection technique. More precisely, the goal is to compare it with a system protected with Double Error Correction codes and verify that both approaches are similar, for certain values of block size (B) and memory size (M).

Under the mentioned circumstances, a failure in the memory can be produced by the following situations:

1) two words (or more) with two errors on a block;
2) three (or more) errors on a word.

The probability of failure in a block caused by two words with a double error (given k errors in the entire memory) can be computed by first estimating the probability of a word having two errors

$$P_{\text{one\_word}}^{(2\_\text{errors})} = \binom{N}{2} \cdot \left(\frac{k}{M \cdot N}\right)^2 \cdot \left(1 - \frac{k}{M \cdot N}\right)^{N-2} \cong \frac{1}{2} \cdot \left(\frac{k}{M}\right)^2. \tag{2}$$

The last two terms of the expression are the probability that 2 of the k errors have affected a given word, and that the rest of the bits in that word have not been affected. The first term is the number of possible combinations of N bits with 2 errors.

Then, the probability of failure in a block caused by two words with double errors can be obtained as

$$P_B|_{\text{two\_words}}^{(2\_\text{errors})} \cong \binom{B}{2} \cdot \left[P_{\text{one\_word}}^{(2\_\text{errors})}\right]^2 = \frac{B \cdot (B-1)}{8} \cdot \left(\frac{k}{M}\right)^4. \tag{3}$$

The second term is the probability of having two words (that is why the square of P is considered) with double errors, based on the previous expression. The first term is the number of possible combinations of having 2 out of B words in a single block with the mentioned characteristics.

Following a similar reasoning, the probability of failure in a block caused by a single word having three errors can be approximated by computing first the probability of a word having three errors:

$$P_{\text{one\_word}}^{(3\_\text{errors})} = \binom{N}{3} \cdot \left(\frac{k}{M \cdot N}\right)^3 \cdot \left(1 - \frac{k}{M \cdot N}\right)^{N-3} \cong \frac{1}{6} \cdot \left(\frac{k}{M}\right)^3. \quad (4)$$

Then, computing the probability that such a word occurs in a block:

$$P_B|_{\text{one\_word}}^{(3\_\text{errors})} \cong \binom{B}{1} \cdot P_{\text{one\_word}}^{(3\_\text{errors})} = \frac{B}{6} \cdot \left(\frac{k}{M}\right)^3. \quad (5)$$

As explained before, the failure scenarios of the presented techniques happen when two or more words in a block suffer double errors, or when a single word suffers three errors. That is, the probability of failure for the block parity technique is

$$P_{\text{Block\_Parity}} \cong P_B|_{\text{two\_words}}^{(2\_\text{errors})} + P_B|_{\text{one\_word}}^{(3\_\text{errors})}. \quad (6)$$

This holds if both probabilities are low, which is true in this case. However, the probability of failure of a memory protected with DEC codes is

$$P_{\text{DEC}} \cong P_B|_{\text{one\_word}}^{(3\_\text{errors})}. \quad (7)$$

In other words, the DEC codes will only fail with three or more errors per word.

Therefore, if we want that the presented technique based on Block parity offers a reliability similar to the one obtained with DEC codes, the following relation has to be met:

$$P_B|_{\text{one\_word}}^{(3\_\text{errors})} \gg P_B|_{\text{two\_words}}^{(2\_\text{errors})}. \quad (8)$$

If the expressions previously derived for each probability are used, then this relation becomes

$$1 + \frac{4 \cdot M}{3 \cdot k} \gg B. \quad (9)$$

This expression (9) is true for small values of k. Therefore, a limit to this value should be calculated in order to make the expression applicable. When scrubbing is used, this value could be determined based on the scrubbing period and the error arrival rate. These will be generally small enough such that the condition is easily met.

In order to make this expression more practical for the non scrubbing case, a bound to the number of errors that have arrived in the memory should be used. It is intuitive that an appropriate decision would be to use the METF, since this value will likely produce a failure in the system.

There is an approximation in [11] that predicts the METF for memories with codes able to correct up to L bits:

$$\text{METF}|_L \cong \sqrt[L+1]{(L+1)!} \cdot \Gamma\left(1 + \frac{1}{L+1}\right) \cdot M^{\frac{L}{L+1}} \quad (10)$$

where $\Gamma(x)$ is the gamma function. This expression can be used when M is much larger than one (as it normally happens in memories). In the case under study, DEC codes would correct up to $L = 2$ errors. Therefore, the approximation becomes

$$\text{METF}|_{L=2} \cong \sqrt[3]{3!} \cdot \Gamma\left(1 + \frac{1}{3}\right) \cdot M^{\frac{2}{3}}. \quad (11)$$

If this value replaces k in (9), then the following is obtained:

$$1 + \frac{4 \cdot M^{\frac{1}{3}}}{3 \cdot \sqrt[3]{3!} \cdot \Gamma\left(1 + \frac{1}{3}\right)} \gg B. \quad (12)$$

Simplifying this expression, and assuming that $M \gg 1$:

$$0.82 \cdot M^{\frac{1}{3}} \gg B. \quad (13)$$

When this inequality is met, the probability of failure due to two words with two errors each is negligible compared to failures due to a word with three errors.

This is an interesting result, as it means that for large memories the proposed scheme will approach the METF of a DEC code.

In fact, in terms of MTTF, the reliability of the proposed approach can be even larger than that of a memory protected by a DEC code.

As an example, if we consider a word size of $N = 16$ bits, then from [12] the number of bits for a SEC-DED code would be $c = 6$ while for a DEC is $c = 10$. Therefore, the arrival rate per block would be for a memory protected by DEC:

$$\lambda_{\text{block}}|_{\text{DEC}} = \lambda_{\text{bit}} \cdot (16 + 10) \cdot (B - 1) \quad (14)$$

where the second factor is the word width considering the redundant bits for protection, and the third factor is the number of words per block.

On the other hand, the arrival rate per block for the proposed scheme would be

$$\lambda_{\text{block}}|_{\text{block\_parity}} = \lambda_{\text{bit}} \cdot (16 + 6) \cdot B. \quad (15)$$

Notice that, in this case, there is one more word per block than in the previous scenario, which in fact is the parity word itself.

Assuming similar METFs, the ratio of MTTFs can then be approximated by

$$\frac{\text{MTTF}_{\text{block\_parity}}}{\text{MTTF}_{\text{DEC}}} \cong \frac{\lambda_{\text{block}}|_{\text{DEC}}}{\lambda_{\text{block}}|_{\text{block\_parity}}} = \frac{(16 + 10) \cdot (B - 1)}{(16 + 6) \cdot B}. \quad (16)$$

The result of this ratio, for different values of block size (B), is offered in Table II which shows a significant MTTF increase for values of B larger than 8.

IV. Simulation Results

In this section, the reliability of the proposed techniques is validated by simulation. The experiments illustrate the METF [from which the MTTF can be derived using (1)] for different memory configurations and validate the theoretical results presented in the previous section.

TABLE II
RATIO OF MTTFS FOR THE PROPOSED TECHNIQUE VERSUS DEC FOR DIFFERENT BLOCK SIZES

B	MTTF ratio
8	1.0341
16	1.1080
32	1.1449
64	1.1634
128	1.1725
256	1.1772

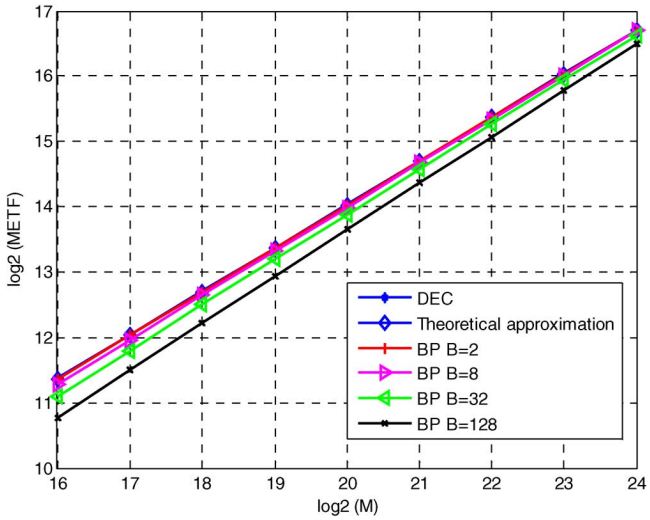

Fig. 5. METF for Block Parity (BP) for different memory and block sizes.

A number of simulations have been performed, for different memory sizes (M) and block sizes (B). The results obtained have been analyzed and are shown in Fig. 5. The first conclusion is that the METF for the proposed technique is close to that of a DEC code, as suggested before. In this way, a memory protected with SEC can be easily upgraded to provide DEC reliability performance by using the block parity presented here.

Another conclusion is that both behaviors are closer when blocks are small (low B). The reason of this is that smaller blocks will have fewer combinations to produce a failure. On the other hand, smaller blocks will introduce a higher area overhead, since there is a Parity Word per block. Therefore, the most appropriate block size should be chosen in advance, in order to get an appropriate reliability at the minimum cost.

These results are better seen in Fig. 6. In this graph, the reliability ratio of the proposed technique versus DEC is depicted. It can be seen as before, that smaller blocks produce a ratio closer to 1. Besides, regardless the block size, the behavior is better when the memory size M grows. The ratio for DEC slightly deviates from 1 due to the finite number of experiments used in the simulations.

In Fig. 7, the block size given by the condition shown in (13) is depicted, for different memory sizes. That represents the maximum block size that must be chosen to make the memory behave as protected with DEC codes. As said before, the lower the block size, the better the reliability but the higher the area penalty.

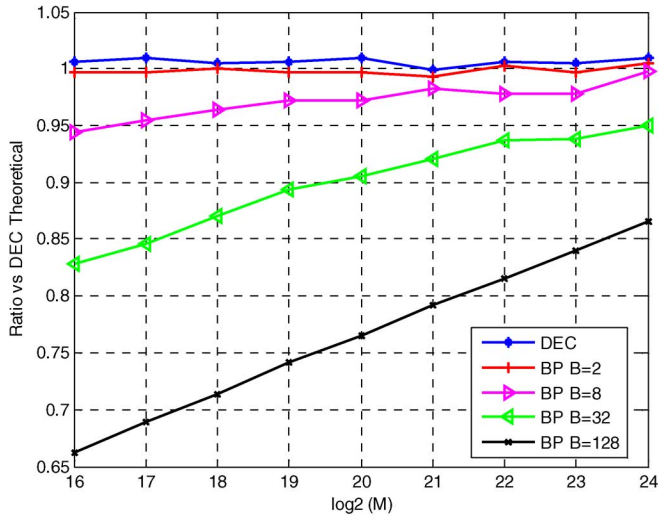

Fig. 6. Ratio METF for block parity (BP) versus DEC for different memory and block sizes.

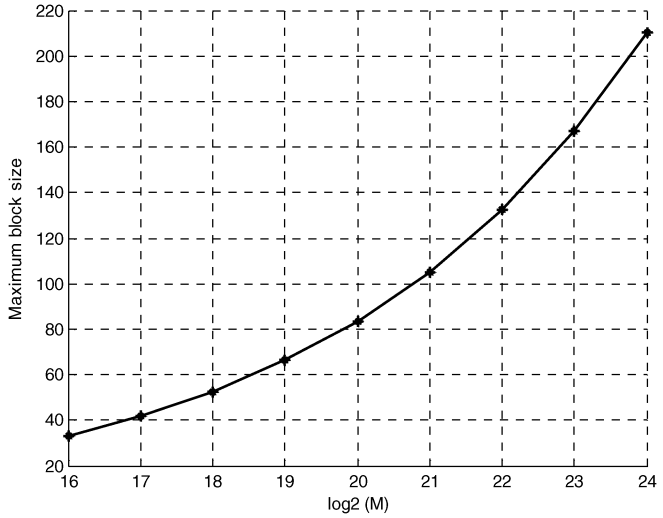

Fig. 7. Block size (B) given by satisfying condition in equation (13) for different memory sizes.

For a memory size of 2^{24} words, as illustrated in Fig. 7, a block size (B) of 210 or less should be used. Following this assessment, in Fig. 6, that for block sizes much smaller than 210 the reliability performance of the memory approaches that of DEC protected memory. At 128 the ratio has been reduced to 0.85.

V. DESIGN TRADEOFFS

In the previous sections, the block parity technique has been introduced and its reliability has been analyzed. In this section, the different trade-offs involved in its use are discussed.

The first implication of block parity is that some words are dedicated to error protection, and therefore the number of words that can be used for data storage is reduced. This decreases the effective memory size (i.e., the actual capacity available for data, excluding parity words). This effective memory size

TABLE III
FRACTION OF MEMORY AVAILABLE FOR DIFFERENT VALUES OF B

B	Percentage of Original Size
2	50.0%
4	75.0%
8	87.5%
16	93.7%
32	96.9%
64	98.4%
128	99.2%
256	99.6%

is shown in Table III, for different values of block size (B), as a percentage of the original size.

The table clearly shows that large values of B reduce the overhead associated with the technique. On the other hand, small values of B introduce a significant overhead in the memory capacity.

For practical reasons, the values of B to be considered should be equal or larger than 8, in order to produce an acceptable size reduction.

From the cost analysis it becomes clear that larger values of B are preferable in terms of memory effectiveness. But larger values of B reduce reliability, as discussed before. As a conclusion of the previous discussion, the value chosen for B should be the maximum that meets condition in (13). This seems reasonable from the results in Fig. 6, which show that a value of $B = 8$ can provide a reliability close to that of a memory protected with DEC for memories whose size is larger than 64 K. As a summary from the discussions above, values of $B = 8$ or 16 seem to provide a good tradeoff between cost and reliability in many cases.

VI. CONCLUSION

In this paper, a technique to increase memory reliability by dedicating some of the memory words to error detection and correction has been presented. This technique achieves a substantial increase in the MTTF with a limited cost in terms of area, when larger block sizes are used. In fact, for a memory protected with SEC-DED, the proposed technique can provide a better MTTF than that of a memory with DEC, at a lower cost. The main drawbacks of the proposed technique is that it requires four memory accesses for writing operations and some area overhead to store the block parity.

Therefore, the technique presented in this work can be useful for designers that want to provide additional reliability for existing memories in an easy way.

Although in this paper the block parity technique has been used in a memory that implements SEC-DED, it could also be applied when the memory implements a parity bit or more advanced codes such as double error correction (DEC). In the case of parity bit protection reliability similar to that of SEC protection can be achieved with block parity. For DEC protection the reliability can get close to that of triple error correction (TEC) when block parity is used. Finally, it is worth mentioning that the benefits of block parity also apply when scrubbing is used.

REFERENCES

[1] R. C. Baumann, "Soft errors in advanced computer systems," *IEEE Des. Test. Comput.*, vol. 22, no. 3, pp. 258–266, May/Jun. 2005.
[2] R. D. Schrimpf and D. M. Fleetwood, *Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices*. Singapore: World Scientific, 2004.
[3] M. Nicolaidis, "Design for soft error mitigation," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 405–418, Sep. 2005.
[4] G. C. Cardarilli, A. Leandri, P. Marinucci, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Design of a fault tolerant solid state mass memory," *IEEE Trans. Rel.*, vol. 52, no. 4, pp. 476–491, Dec. 2003.
[5] A. M. Saleh, J. J. Serrano, and J. H. Patel, "Reliability of scrubbing recovery-techniques for memory systems," *IEEE Trans. Rel.*, vol. 39, no. 1, pp. 114–122, Apr. 1990.
[6] R. M. Goodman and M. Sayano, "The reliability of semiconductor RAM memories with on-chip error-correction coding," *IEEE Trans. Inf. Theory*, vol. 37, no. 3, pp. 884–896, 1991.
[7] M. Blaum, R. Goodman, and R. McEliece, "The reliability of single-error protected computer memories," *IEEE Trans. Comput.*, vol. 37, no. 1, pp. 114–119, 1988.
[8] G. C. Yang, "Reliability of semiconductor RAMs with soft-error scrubbing techniques," *Proc. Inst. Electr. Eng.—Comput. Digit. Tech.*, vol. 142, no. 5, pp. 337–344, Sep. 1995.
[9] 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. Nucl. Sci.*, vol. 54, no. 4, pt. 2, pp. 935–945, Aug. 2007.
[10] 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.
[11] J. A. Maestro and P. Reviriego, "Selection of the optimal memory configuration in a system affected by soft errors," *IEEE Trans. Device Mater. Rel.*, vol. 9, no. 3, pp. 403–411, Sep. 2009.
[12] R. Naseer and J. Draper, "DEC ECC design to improve memory reliability in sub-100 nm technologies," in *Proc. 15th IEEE Int. Conf. Electron., Circuits, Syst. (ICECS)*, 2008, pp. 586–589.
[13] S. Lin and D. J. Costello, *Error Control Coding, 2nd Ed.*. Englewood Cliffs, NJ: Prentice-Hall, 2004.
[14] R. W. Hamming, "Error correcting and error detecting codes," *Bell Sys. Tech. J.*, vol. 29, pp. 147–160, Apr. 1950.
[15] M. Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM J. R & D*, vol. 14, pp. 395–401, Jul. 1970.
[16] W. Feller, *An Introduction to Probability Theory and Its Applications*. New York: Wiley, 1968.