D. C. Bossen
M. Y. Hsiao

A System Solution to the Memory Soft Error Problem

High-density and/or high-performance memory chip designs often create new reliability problems; one good example is the alpha-particle problem for high-density RAM and CCD chips, the problem being that soft errors may "line up" with existing hard errors, giving rise to double errors which are not correctable with conventionally implemented single-error-correcting double-error-detecting codes. In this paper it is shown that an overall system approach based on error-correcting codes and system maintenance strategy will reduce the main memory failure rate at the system level as if the alpha-particle problem had not occurred. This system solution is designed to be compatible with most existing memory designs so that there should be minimal additional cost for implementing it. The procedure described herein uses the capability of a single-error-correcting and double-error-detecting code to detect one hard and one soft error; then a microcode and hardware algorithm performs the correction of both errors. Results of both analytical and simulation modeling of the method and its comparison with other techniques are also included.

Introduction

Progress in memory chip technology has been significant with regard to cost, performance, and reliability improvements. Cost reduction has come mostly from chip density increases [1-3]. For example, FET memory chip density has advanced from 256 bits per chip to 64K bits per chip in the last decade, with the major advantage of this density improvement being cost reduction. Another advantage of higher bit densities on a chip is the reduction in the number of packaging levels and the interconnections between them. This also results in a reliability improvement in the intrinsic failure rate on a per-bit basis [3].

To achieve the high density, the memory cell area has to be extremely small. At present, most LSI and VLSI RAM memories are of the dynamic MOSFET type using the one-device-cell design [4]. In recent laboratory tests [5, 6], this kind of dynamic MOSFET memory has suffered a new kind of failure caused by alpha-particle radiation, and has a failure rate one to two orders of magnitude higher than the basic intrinsic failure rate. It has also been discovered that the primary source of alpha particles is in the package and solder. Therefore, much work has been devoted to fixing this problem by changing the package material or the cell design, or by adding a shielding layer between package, solder, and the memory circuit. Although some progress has been made, the problem has not been completely eliminated.

In this paper, instead of a device or package fix, a system solution is presented which uses error-correcting codes in combination with system maintenance strategy. There is virtually no additional cost in this solution to the alpha-particle problem and yet it achieves the desired result.

The next section of this paper describes the basic failure definitions and the memory cell failure mode caused by alpha-particle and other radiation sources. The section following that reviews the base error-correcting-code (ECC) system with respect to a specific single-error-correcting and double-error-detecting code example. Subsequent sections then describe what options are available if we need to correct a high rate of errors in a memory system; the system solution that was arrived at after examining the options; a microcode implementation approach; presentation of resulting data from an analytical model and a simulation run; and a comparison of this technique with the state of the art. A summary and conclusions are presented in the final section.

Overview of alpha-particle phenomenon and basic failure modes

Alpha particles are doubly charged helium nuclei ejected from the nucleus of a high atomic number atom during radioactive decay. The mass of an alpha particle is about 7500 times that of an electron, and its kinetic energy is of the order of several million electron volts, *e.g.*, up to 10.5 MeV. Electron-hole pairs are generated by the alpha particles as they lose energy in the silicon and resulting charges can be collected in the memory storage capacitors within a cell.

For high-density dynamic memory chips, the presence or absence of minority carrier charge in the cell is the basic principle of data storage. In the case of CCDs and FET RAMs, the charges are electrons and are typically in the range of 3×10^5 to 3×10^6 electrons. Although there can be a relatively large difference in the charge between *one* and *zero* states, many factors will reduce this difference, for example, noise tolerance, sense amplifier, etc. The net effective difference in the number of electrons between the *one* and *zero* states is called a "critical charge," Q_{crit}. If the alpha-particle-generated electrons exceed Q_{crit}, an error will result.

In the one-device dynamic MOSFET memory design, the information is stored as the voltage state of a floating storage node which serves as one plate of an MOS capacitor. Collection of alpha-particle-generated charge by the floating storage node alters its voltage and can result in loss of the stored information. The information is lost by an empty weli becoming filled, *i.e.*, a *one* becoming a *zero*. However, the opposite direction, *i.e.*, *zero* to *one*, cannot occur in the cell. This type of failure is not permanent and can be erased during the next memory write operation; this is why the term *soft* error is used, since there is no permanent physical damage to the memory cell.

Very dense RAM (≥4K) and CCD (≥16K) chips, which are sensitive to alpha-radiation errors, are easily contaminated by exposure to high flux rates. In general, a CCD cell has a lower Q_{crit} than a RAM cell, hence a CCD cell is more susceptible to alpha-induced failures.

With very few exceptions, alpha-induced failure rates usually increase as cell size is reduced with high-density chip designs. This poses a serious obstacle to the progress of memory technology as higher and higher densities are sought.

At present most errors caused by alpha-particle radiation are single-bit errors. This may not necessarily be the case with future ultra-high-density RAM memory chips.

It should be pointed out here that in a one-device RAM memory the storage node may pick up alpha-generated electrons at any time; thus the bit failure rate will be independent of system operation or cycle time. Also, alpha emissions are nuclear events unaffected by temperature, pressure, etc.

Bit lines and sense amplifiers are also sensitive to alpha-particle charge because the sensing process involves sharing charge between the bit line and the memory cell. It makes no difference whether the charge generated by alpha particles is collected by the memory cell or by the bit line. However, a sense amplifier can fail in both the *one* and *zero* directions; therefore, we cannot take advantage of the memory failure mode of *one* to *zero* only, for which more effective codes possible with binary asymmetric channels [7] can be used.

Besides alpha-particle-induced errors, cosmic rays are another source that can cause soft failures in memory. The cosmic ray contains several kinds of high-energy particles, *e.g.*, protons, neutrons, and muons. The density of these particles is low at sea level but increases at higher altitudes [8].

LSI memory system and error-correcting codes

In the semiconductor memory area, the most widely used error-correcting codes are the class of single-error-correcting and double-error-detecting (SEC-DED) codes [9]. This class of codes is most effective for a memory system organized on a one-bit-per-card basis.

As the memory chip density goes up, the card may become a module, and the bit-per-card organization may become a bit-per-chip situation. In a bit-per-chip per ECC word-organized memory, it is true that any category of failure associated with one chip is correctable; typical categories include single array cell, single bit line, single word line, and a broad category referred to as "chip kill."

The significance of such a partition of failures into categories is that the number of erroneous bits produced by a particular failure is different for each category. However, in the presence of the SEC-DED code, the maintenance strategy of a system may allow failures to accumulate. The particular categories of failure which are allowed to accumulate will strongly affect the probability that future random errors (hard or soft) will "line up" with previous failures and cause a double error to occur. This is a serious concern when high levels of soft errors such as those caused by alpha-particle radiation can be expected.

Before we describe the solution to this problem, we shall review the base ECC using a (72, 64) SEC-DED code as an example.

Figure 1 A (72, 64) SEC-DED code matrix.

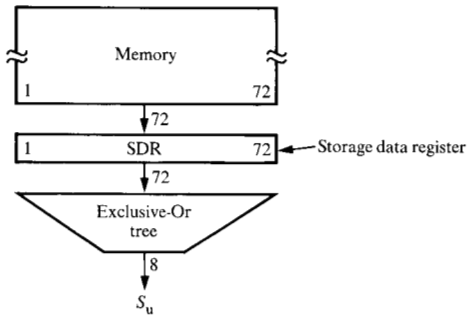

Figure 2 Simplified ECC implementation.

The (72, 64) code shown in Fig. 1 is an SEC-DED code. Its basic properties and implementation have previously been described [9]. Figure 2 illustrates a simplified implementation of an ECC procedure based on (72, 64) code.

During the fetch operation, a codeword containing 72 bits, *i.e.*, 64 data bits and 8 check bits, is fed into the Exclusive-Or tree to generate the 8 syndrome bits (S_u). Since the code matrix [**H**] in Fig. 1 has columns containing an odd number of *ones*, the syndrome S_u should have one of the following:

1. All 8 bits = 0 $\Rightarrow$ no error.
2. Odd number of *ones* in the 8 bits $\Rightarrow$ single or multiple errors.
   (a) Match one of the code matrix columns $\Rightarrow$ single error assumed and thus the error is corrected at the matched column bit position [10].
   (b) No match in any column position $\Rightarrow$ an *uncorrectable error* (UE) has been detected.
3. Even number of *ones* in the 8 bits $\Rightarrow$ even number of errors. Again, this is a UE situation.

In the three cases listed, only cases 2(b) and 3 could cause a UE. Since errors are assumed to be statistically independent, it is assumed here that double errors are much more likely to happen than three-, four-, and higher multiple-bit errors within a codeword. In the following sections, we shall concentrate our discussion on the case of a double error caused by single solid [11] and soft errors. This situation is a predominant case because if the solid error is a chip kill or word-line kill, the probability of having another single random soft error caused by alpha-particle radiation within the same address block will be high. In this case, the memory system will require a double-error-correction scheme or other alternatives.

Error-correction options and memory system maintenance strategies

One possible solution would be to use a random double-error-correcting code. In order to correct double errors and detect triple errors (DEC-TED), a Bose-Chaudhuri-Hocquenghem (BCH) code of Hamming distance 6 is required [7]. This class of codes requires more check bits and a complicated decoding circuit, as well as a longer decoding time than the SEC-DED code. For example, a DEC-TED code requires 14 check bits instead of 8 in the case of 64 data bits. Adding the 6 extra check bits to every address increases the cost proportionately. In order to stay with the SEC-DED code, we have to take advantage of the erasure-correction capability of a distance 4 code. The double-bit error mode caused by a hard error and an alpha-induced error can be treated as equivalent to an erasure error correction, *i.e.*, a Hamming distance 4 code can correct one solid and one soft error. A method for exploiting this capability will be given in the following section.

In a system environment, another option for failure recovery is the system maintenance strategy [12]. Instead of having service personnel sent immediately to a memory system when a solid error occurs, error-correcting codes would allow physical replacement of solid failures to be deferred for a period of time. A storage maintenance strategy which allowed "solid" correctable failures to accumulate to some threshold before physical replacements were made would be exposed to a higher UE rate when the soft error rate was high. To control this higher UE rate caused by alpha particles, maintenance strategies could be altered so that fewer solid fails would be allowed to accumulate, but this would cause higher removal rates for storage components and thus higher parts costs.

It is possible, however, to take a new approach to the alpha-particle-induced UE which avoids unnecessary extra replacements. This is accomplished by correcting the alpha-particle-induced UE using the existing (72, 64) SEC-DED code along with the decoding logic described in the following section. For this discussion, a memory system UE is equal to one solid error plus one alpha-particle error.

A solid and soft error-correction scheme using an SEC-DED code

The simplified implementation scheme shown in Fig. 2 can be extended as shown in Fig. 3 to correct one solid and one soft error. By dynamically locating the solid position, advantage is taken of the erasure correction capability of a distance 4 code. This is different from the technique by Walker *et al.* [13] where an erasure position is established through historic mark-up and an extra auxiliary memory is thus required. This difference will be further discussed in a subsequent section.

In the algorithm used here, the solid error is dynamically located during the correction process and an algorithmic modification of the original syndrome allows correction of the soft random error. Corrected information is then rewritten into memory. Since the alpha error is transient, subsequent read operations will have only the single solid error to correct. In this way, no extra auxiliary storage is required. As shown in Fig. 1, an ECC codeword consists of 72 bits. Let h ($1 \leq h \leq 72$) be the solid-error bit position and α ($1 \leq \alpha \leq 72$ and $\alpha \neq h$) be the soft-error bit position; then, according to Fig. 3, the resultant syndrome S_u is the Exclusive-Or of the syndromes due to erroneous bits α and h, *i.e.*,

$$S_u = S_\alpha \oplus S_h.$$

The decoding algorithm is summarized as follows:

1. Detect the UE, represented by S_u, a double-bit error syndrome. Save the codeword with errors, as well as the syndrome.
2. Using the exerciser diagnostic patterns, locate position h of the solid error. Knowing the index h, generate S_h using check-bit-generation logic.
3. Determine $S_\alpha = S_u \oplus S_h$.
4. Decode S_α to correct bit α in the data. Invert bit h, determined in step 2, to correct the solid error.

This algorithm can be implemented completely by hardware, as shown in Fig. 3, or by a microcode approach which will be described in the next section.

Implementation in microcode

For many computer systems using the current memory chips and SEC-DED codes, it may be important to have

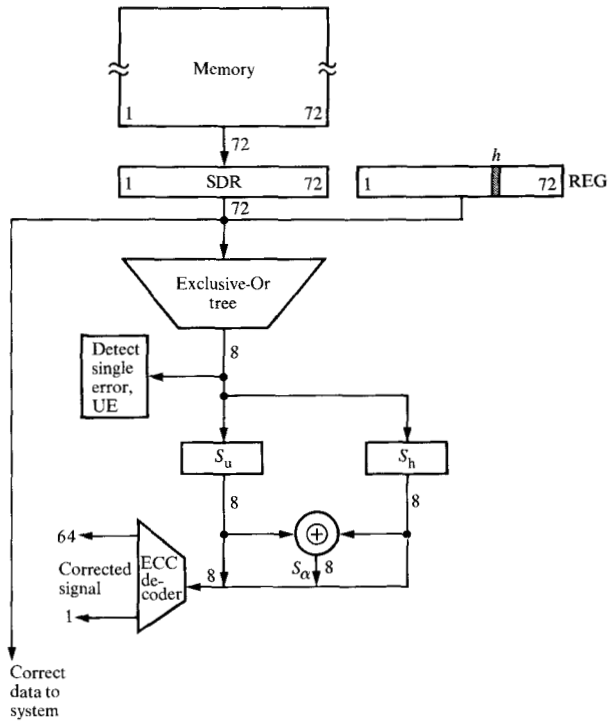

Figure 3 Schematic implementation for correcting alpha-particle-induced UE. (Note: ⊕ indicates Exclusive-Or.)

the flexibility to add/delete the additional double-error-correction capability due to soft error plus hard error. In these cases the capability of microcode implementation would be very attractive.

In order to implement erasure correction, the decoding process must know the position of the permanently stuck bits. The approach taken in this paper is to accomplish this using a small number of tests. In order to apply these tests and to observe the results using a microcode algorithm, the system was designed so that the microcode has access to the syndrome on a fetch operation. This could also be accomplished by the capability to read check bits.

The general flow diagram of the algorithm, illustrated in Fig. 4, follows the steps of the previously described hardware implementation.

The code matrix [H] is given in Fig. 1. The three test patterns P_1, P_2, P_3 are defined in Fig. 5, and the applicable valid syndrome sets are given in Table 1.

It will be noted that this set of three test patterns has the feature that each is a valid codeword, and *one* and *zero* are presented to each bit position when the set is

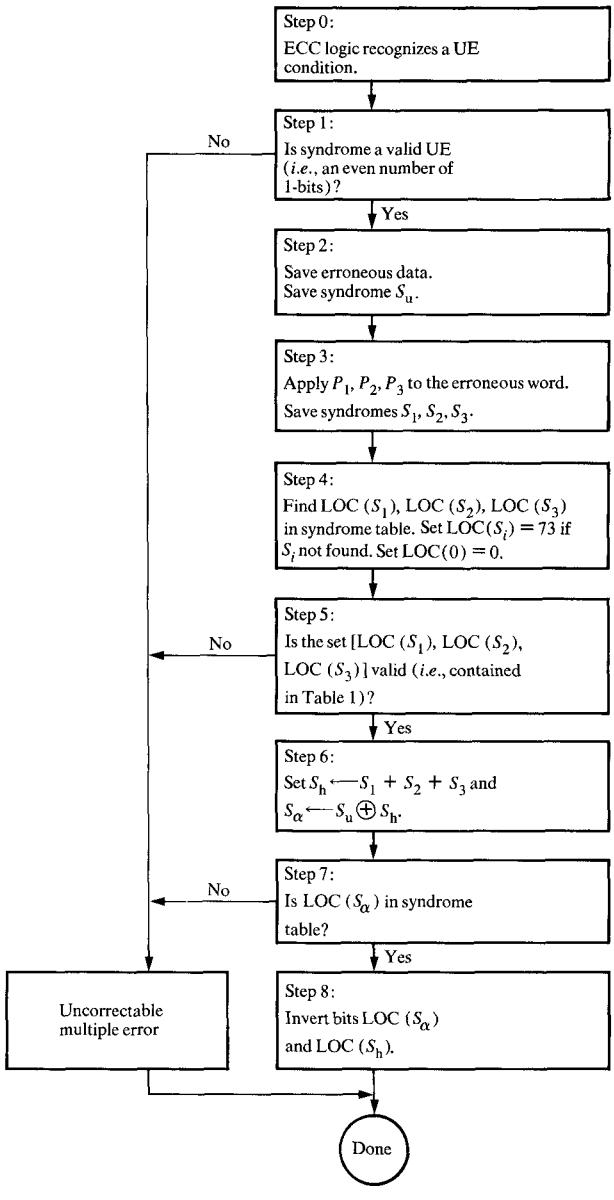

Figure 4 Microcode implementation algorithm. (Note: + indicates Or.)

Table 1 Valid syndrome sets ($1 \leq A \leq 72$).

$LOC(S_1)$	$LOC(S_2)$	$LOC(S_3)$
A	0	0
0	A	0
0	0	A
A	A	0
A	0	A
0	A	A

applied. In addition, it is not necessary to perform a data compare in order to locate the erroneous location. The syndromes from the three tests are sufficient, using the algorithm. An alternate approach would be to provide memory capability to "bypass ECC" and thereby write and read arbitrary 72-bit test patterns; in this case two patterns, one consisting of 72 zeros and the other consisting of 72 *ones*, would be sufficient.

Modeling of reliability improvement

Since a memory with error correction can operate correctly in the presence of permanent physical failures, criteria for replacing physical failures must be established. The criteria for replacement will generally consist of 1) a definition of the event which triggers replacement, and 2) a definition of the failure characteristics of the parts which qualify for replacement. Events which trigger replacement could be, for example, an uncorrectable error or the occurrence of a total number of failed bits exceeding a predetermined threshold; and the specification of failure characteristics of parts which qualify for replacement could be the threshold number of erroneous bits, or chips, etc., per part. Any particular definition of such replacement criteria is called a *replacement strategy*.

A number of strategies for memories with error correction have been described and analyzed with a Monte Carlo simulation model [12]. Key output parameters of such a model are the parts removal rate and the rate of uncorrectable errors. A general conclusion obtained from such modeling is that the parts removal rate and the uncorrectable error rate can be greatly reduced any time that error correction is employed, and a desirable strategy is to minimize the parts removed subject to an acceptable uncorrectable error rate.

The following data will be used to define the failure rate characteristics of a memory system for analytical modeling.

Soft error rates in the range 0.1 to 150% per 1000 hours per chip have been reported [5, 6, 8]. Hard failure rates and failure mode distribution in memory technology have been published [14, 15]. By using these hard and soft error rates and the failure mode percentage distribution in a chip, the following analytical results and a simulation analysis were obtained.

Example memory:

Organization:	72 cards, (72, 64) SEC-DED code, one bit per card per codeword.
Card:	64 chips, one chip selected per card.

P_1	10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010	11111111
P_2	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	11111111
P_3	00000000	00000000	00000000	00000000	00000000	10000000	00000000	00000001	00000000

Figure 5 Test patterns for the (72, 64) code of Fig. 1.

Chip: 16K bits per chip, 128 word lines by 128 bit lines.

Hard failure rate: chip: 0.005% per 1000 power-on hours.
chip piece part category failure rates:
128 word lines: 25% of total.
128 bit lines: 25% of total.
16K bits: 25% of total.
chip kill: 25% of total.

Soft failure rate: varies from 0.1 to 10% per 1000 power-on hours per chip.

This memory has been modeled with an analytic model to determine the uncorrectable error rate as a function of various alpha-particle failure rates. The assumed replacement strategy is no preventive maintenance; *i.e.*, hard fails are allowed to accumulate until a hard uncorrectable error (double bit in some word) occurs, then all failures are replaced. This is referred to as a "clean" maintenance strategy. The system uncorrectable error (UE) rate is given in Fig. 6 for the following code capabilities:

1. No error correction (no ECC).
2. With single error correction only (SEC).
3. With extended error correction (ESEC).

It can be seen that for soft rates in the neighborhood of 1% per 1000 hours per chip and above, the improvement in the UE rate for extended ECC compared to SEC is 100 times.

Modeling of strategies which do not completely clean the memory when a UE occurs requires simulation [14]. These strategies are desirable because they have a lower removal rate of cards than the "clean" strategy.

When repair is necessary, an effective strategy is to replace only the card with the most erroneous bits, with no further action taken. This is called a "minimum maintenance" strategy and is one of the five maintenance strategies discussed by Kwon and Harvey [12]. Under this strategy, the memory card replacement rate is the lowest and UE rate is the highest when using SEC-DED code, since the memory card contains all the errors that are

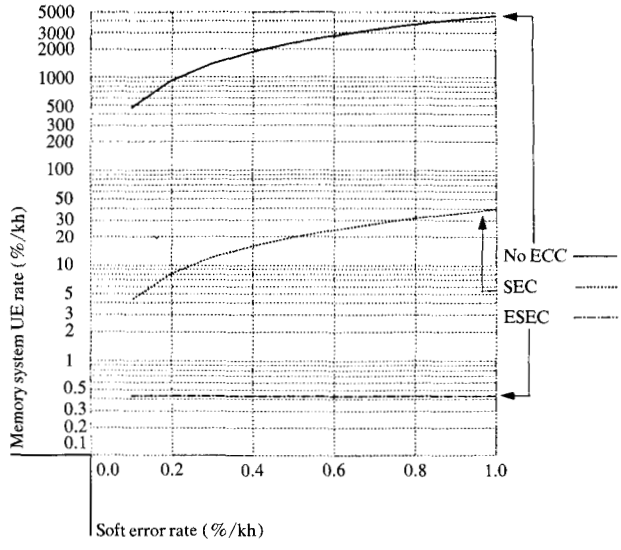

Figure 6 UE rate with no preventive maintenance. (Note: kh = 1000 power-on hours.)

single and can be corrected, and yet it is still functioning. Therefore, the probability that a UE will occur at the next time-to-fail is high. Also, a moderately intelligent memory diagnostic package is required to isolate solid failures. Clearly, this strategy appears not to be the best choice in the case of high alpha-particle-induced errors when the system only has the capability to correct single errors. One example is that a solid chip-kill failure in combination with any alpha-particle-induced soft error within the same address group will cause a UE. By adopting the strategy of replacing any card which contains a chip kill, the UE rate is substantially reduced. This results, however, in a higher card removal rate than the minimum maintenance strategy. Using the Monte Carlo model described in [12], we have run some data on the cases of SEC-DED and extended SEC-DED, *i.e.*, correction of one solid and one soft error, under the two different maintenance strategies. Results are shown in Fig. 7.

From Fig. 7 it is clear that with extended two-error ECC, the UE and card removal rate can be effectively reduced to the rate which existed with solid errors only and one-error ECC; *i.e.*, we have virtually eliminated the

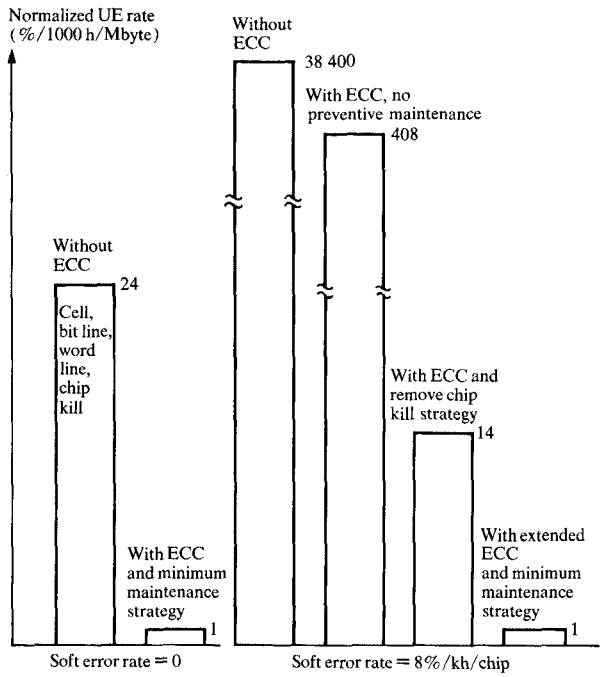

Figure 7 Monte Carlo simulation run results.

Table 2 Example of limitation of read-invert-write-read-invert process [based on (8, 4) $d = 4$ code].

	Data	*Check bits*
Original data	0 0 0 0	0 0 0 0
	"erasures"	random error
Fetched data	0 0 0 1	1 0 0 0
Stuck positions	0 X X X	0 0 0 0
Data after invert-read -invert*	0 0 1 0	1 0 0 0

*Now contains three errors, leading to miscorrection.

impact of alpha-particle-induced errors through extended ECC capability while still using a minimum maintenance strategy.

Comparing procedures

The erasure-correction capability of a distance d code is well known. Therefore, error-correction procedures for memory systems which exploit the erasure-correction capability of a particular code will differ significantly only in the characteristics of their implementations. Issues of hardware cost and speed of the check-bit-generation and error-correction processes are thus the major concerns in comparing procedures.

There are, however, differences in the capabilities of different procedures, even though the same code is used, due to the particular implementation algorithm utilized. For example, a $d = 4$ code has a theoretical capability to correct one hard (erasure) plus a single soft random error. A scheme based on the read-invert-write-read-invert process [15] will not be able to correct this type of error when the number of stuck positions (say three) is greater than the number of erasure errors (say one) which are actually present in the word. In other words, three stuck positions plus one random error can result in two bits in error which are miscorrected by the above process. (See Table 2.) On the other hand, an algorithm based on diagnostic testing of the location, such as that described in this paper, can be designed to properly detect this error.

Another consideration for discovering erasure information is the storage requirement. Erasure-decoding schemes for memory which depend upon recording of information about previous failures for later use when a double error is detected must consider the size and complexity of this special store. Large-capacity storage with a high soft error rate would require a prohibitively large auxiliary store for recording past error data.

In this application area, a more suitable approach to erasure correction is to derive the necessary erasure information at the time the double error occurs and is detected. This approach has the additional advantage of minimizing the probability of miscorrection if more errors than the code is capable of handling are present.

Such a scheme handles combinations of two types of errors, both permanent and soft. Methods such as the one described by Walker *et al.* [13] explicitly assume an accumulation of hard errors, so that the first single error causes storage of the syndrome of the single error. In an environment where soft errors are more predominant, such a procedure will not work, since the stored syndrome would in general represent a soft error which may not be present when a double error to be corrected occurs. A modification of this procedure to store the most recent hard error may relieve this difficulty, but would require a test-based decision to store syndromes.

Also the technique for the correction of erasures based on the read-invert-write-read-invert process has the capability, when combined with a distance 4 code, of correcting one hard and one soft error. An implementation requirement of this scheme is the ability to write arbitrary data patterns into a data word, including the check bit positions. In practical applications of erasure correction, where this capability does not exist, a more suitable ap-

proach is to use specifically designed data test patterns which are valid codewords as far as distance 4 code is concerned.

Summary and conclusions

The progress of semiconductor memory technology has advanced in the industry from LSI to VLSI and will continue to advance to ULSI in the future. With accelerated progress in memory chip density, chip reliability is impacted by alpha-particle radiation errors at present, and is expected to be increasingly impacted in the future, possibly by other radiation sources as well. It is, therefore, of extreme importance to find practical solutions enabling us to reduce, eliminate, or to live with this type of failure. In this paper we have presented a system solution to this problem by extending the SEC-DED code capability in combination with a minimum maintenance strategy; and we have shown that this solution offers the best cost and reliability tradeoff with no reduction in performance, as compared with the double random error-correction code scheme. In future memory technology development the system solution is expected to continue to play an important role.

Innovative solutions will always be needed; probably a more powerful error-correcting code is also the way to go. This is similar to the path which the industry has taken in the magnetic media storage area; for example, error-correcting codes have been very important in increasing the area density of magnetic tape systems. In the semiconductor memory area, error-correcting codes can also be used to enhance yield [16] and can eventually be integrated on the chip for both yield and reliability improvement.

Acknowledgments

The authors thank R. A. Rutledge for his analytical model run, C. L. Chen for his test pattern generation, S. K. Kwon for his simulation run, and W. E. Harding for his comments.

References and notes

1. B. J. Greenblott and M. Y. Hsiao, "Where is Technology Taking Us in Data Processing Systems," *Proceedings*, National Computer Conference, Los Angeles, CA, 1975, pp. 623-628.
2. E. Bloch and D. J. Galage, "Component Progress: Its Effect on High Speed Computer Architecture and Machine Organization," *Proceedings*, Symposium on High Speed Computers and Algorithm Organization, Urbana, IL, April 13-15, 1977; also in *Computer* **11**, 64-76 (1978).
3. M. Y. Hsiao, "The Impact of LSI on Computer Reliability, Availability and Serviceability," *Proceedings of International Computer Symposium*, Taipei, Taiwan, 1978.
4. R. H. Dennard, "Field Effect Transistor Memory," U.S. Patent 3,387,286, 1968.
5. T. C. May and M. H. Woods, "Alpha-Particle-Induced Soft Errors in Dynamic Memories," *IEEE Trans. Electron Devices* **ED-26**, 2-9 (1979).
6. D. S. Yaney, J. T. Nelson, and L. L. Vanskike, "Alpha-Particle Tracks in Silicon and Their Effect on Dynamic MOS RAM Reliability," *IEEE Trans. Electron Devices* **ED-26**, 10-15 (1979).
7. W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes*, MIT Press, Cambridge, MA, 1972.
8. T. C. May, "Soft Errors in VLSI—Present and Future," *Proceedings*, 29th Electronic Components Conference, Cherry Hill, NJ, May 14-16, 1979.
9. M. Y. Hsiao, "A Class of Optimal Minimum Odd-weight-column SEC-DED Codes," *IBM J. Res. Develop.* **14**, 395-401 (1970).
10. A multiple number of odd errors could cause a miscorrection.
11. Solid errors and hard errors are equivalent in this paper.
12. S. K. Kwon and H. E. Harvey, "A Simulation Approach to the Reliability Analysis of Main Storage Systems," *Proceedings*, IEEE 12th Annual Simulation Symposium, Tampa, FL, 1979, pp. 257-272.
13. W. K. S. Walker, C. E. Sundberg, and C. J. Black, "A Reliable Spaceborne Memory with a Single Error Correction and Erasure Correction Scheme," *IEEE Trans. Computers* **C-28**, 493-500 (1979).
14. H. C. Rickers, "Microcircuit Device Reliability Memory/LSI Data," Reliability Analysis Center, *RADC/RBRAC, MDR-3*, Winter 1975-76, U.S. Air Force Rome Air Development Center, Rome, NY.
15. W. C. Carter and C. E. McCarthy, "Implementation of an Experimental Fault-Tolerant Memory System," *IEEE Trans. Computers* **C-25**, 557-568 (1976).
16. D. C. Bossen, C. F. Haugh, and M. Y. Hsiao, "Dynamic Address Translation Scheme Using Orthogonal Squares," U.S. Patent 3,812,236, 1974.

Received May 16, 1979; revised December 3, 1979

The authors are located at the IBM Data Systems Division laboratory, Poughkeepsie, New York 12602.