Lemma 4: Let I points $x_1, \cdots, x_I$ lie on some sphere S_r of radius r in $\{0,1\}^n$. Then

$$\frac{1}{n} d(x_i, x_j) < \frac{I}{I-1} \frac{2r}{n} \left(1 - \frac{r}{n}\right) \tag{6}$$

for at least one pair $i \neq j$.

To apply this lemma in our situation, we recall that $I \geq 2^{n\epsilon/4}$, $r \leq l$ and $\lambda(1-\lambda) \leq \tau$. Therefore, the inequality (6) implies that among the codewords $\varphi(m_k, E_k)$ there exist at least two, $\varphi(m_i, E_i), \varphi(m_j, E_j)$, with the distance between them not exceeding $2t$. Both codewords $\varphi(m_i, E_i), \varphi(m_j, E_j)$ lie at distance exactly r from x, and $x \oplus \varphi(m_i, E_i) \in \mathcal{V}(E_i)$, $x \oplus \varphi(m_j, E_j) \in \mathcal{V}(E_j)$. In other words, denoting $x_i = x \oplus \varphi(m_i, E_i)$, $x_j = x \oplus \varphi(m_j, E_j)$, we see that x_i and x_j have the following properties:

a) $x_i \in \mathcal{V}(E_i)$, $x_j \in \mathcal{V}(E_j)$,
b) $\varphi(m_i, E_i) \oplus x_i = \varphi(m_j, E_j) \oplus x_j$,
c) $\mathrm{wt}(x_i) = \mathrm{wt}(x_j) = r$,
d) $\mathrm{wt}(x_i \oplus x_j) \leq 2t$.

Define now two error-vectors e_i and e_j as follows: For any k, $1 \leq k \leq n$, the kth component $(e_i)_k$ of e_i equals 1 if and only if $(x_i)_k = 1$ and $(x_j)_k = 0$, while the kth component $(e_j)_k$ of e_j equals 1 if and only if $(x_j)_k = 1$ and $(x_i)_k = 0$. Then one can easily see that e_i and e_j have the following properties:

a') $e_i \in \mathcal{V}(E_i)$, $e_j \in \mathcal{V}(E_j)$,
b') $\varphi(m_i, E_i) \oplus e_i = \varphi(m_j, E_j) \oplus e_j$,
c') $\mathrm{wt}(e_i) = \mathrm{wt}(e_j) \leq t$ (since $x_i \oplus x_j = e_i \oplus e_j$, $\mathrm{wt}(e_i) = \mathrm{wt}(e_j)$ and $\mathrm{wt}(x_i \oplus x_j) \leq 2t$).

Hence, receiving the vector $y = \varphi(m_i, E_i) \oplus e_i = \varphi(m_j, E_j) \oplus e_j$, the decoder can not distinguish which of the two events has occurred: The codeword $\varphi(m_i, E_i)$ was sent and the error vector was e_i, or the codeword $\varphi(m_j, E_j)$ was sent and the error vector was e_j. Therefore, the decoder does not know which of the messages m_i and m_j was sent. This means that our code cannot correct all $\leq t$-on-l localized errors, contrary to the assumption at the beginning of the proof. Theorem 2 is proved.

Remark: A problem similar to the one considered in this correspondence can be formulated also for nonbinary channels with partially localized errors. In this case one can easily generalize the upper and lower bounds from Theorems 1–3. Unfortunately, even for channels with localized errors considered in [2] (the case $t = l$ in the notations of the present correspondence), these upper and lower bounds do not coincide. Namely, while the upper bound for $t = l$ is the q-ary Hamming bound, the lower bound is worse that the Hamming bound. At present we do not have any reasonable conjecture about the exact asymptotic formula for the rate in the q-ary case.

References

[1] A. V. Kuznetsov and B. S. Tsybakov, "Coding for memories with defect," *Probl. Peredach. Inform.*, vol. 10, no. 2, pp. 52–60, 1974.

[2] L. A. Bassalygo, S. I. Gelfand, and M. S. Pinsker, "Coding for channels with localized errors," in *Proc. Fourth Soviet–Swedish Workshop in Inform. Theory*, Gotland, Sweden, 1989.

[3] J. H. van Lint, "Coding for channels with localized errors," in *Beauty is Our Business*. Berlin: Springer-Verlag, 1990, pp. 274–279.

[4] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1977.

The Reliability of Semiconductor RAM Memories with On-Chip Error-Correction Coding

Rodney M. Goodman and Masahiro Sayano

***Abstract*—The mean lifetimes are studied of semiconductor memories that have been encoded with an on-chip single error-correcting code along each row of memory cells. Specifically, the effects of single-cell soft errors and various hardware failures (single-cell, row, column, row–column, and entire chip) in the presence of soft-error scrubbing are examined. An expression is presented for computing the mean time to failure of such memories in the presence of these types of errors using the Poisson approximation; the expression has been confirmed experimentally to accurately model the mean time to failure of memories protected by single error-correcting codes. These analyses will enable the system designer to accurately assess the improvement in mean time to failure (MTTF) bought by the use of error-control coding.**

***Index Terms*—Error-correction coding, random access memory, soft-error scrubbing, mean time to failure.**

I. Introduction

A typical $N \times 1$ semiconductor RAM memory is composed of a two-dimensional array of N memory cells with word lines along the rows and bit lines along the columns. When a bit is accessed, the word line is activated, allowing the entire row of memory cells to be read by the bit lines. One of these bits is then chosen by the column selection circuitry, and that bit is then outputted. (See Fig. 1.)

These $N \times 1$ chips are organized on boards to create byte-wide memory systems. Typically, for SIMM-type memory modules, eight such chips are aligned to provide an 8-bit byte output as shown on Fig. 2(a). The bits from the same addresses on each chip compose a byte; thus, each chip provides one bit of the byte. On a larger system, the board may be composed of many rows of chips. Fig. 2(b) shows a case where rows of $N \times 1$ chips are used to form a multipage memory board. Often, in a large multipage memory, the rows of the memory are encoded with an (n, k) error-correcting code (information chips are shown white; parity chips are shown shaded). This is an example of board-level error-correcting coding [1], [2]. Systems of this form are the most common; they are based on $N \times 1$ memory chips and therefore most often use Hamming codes. Other schemes are used when the chips are byte-wide chips [3].

Memory chips are subject to several types of failure modes. The two main classes of errors are hard and soft errors. Soft errors, induced most commonly by alpha particle radiation and noise, can affect the content of a cell temporarily by upsetting the charge stored in the cell; this occurs for both static and dynamic memory cell types. The effect is not permanent since no physical damage is done to the chip [4]. As memory cells decrease in size, they individually become more susceptible to this type of damage; furthermore, one alpha particle can affect more than one cell, resulting in clusters of errors [5], as shown in Fig. 3.

Manuscript received November 27, 1990. This work was supported in part by NSF Grant MIP8711568. This work was presented in part at the International Symposium on Information Theory and Its Applications, Waikiki, HI, November 27–30, 1990.

The authors are with the California Institute of Technology, Mail Code 116-81, Pasadena, CA 91125.

IEEE Log Number 9143291.

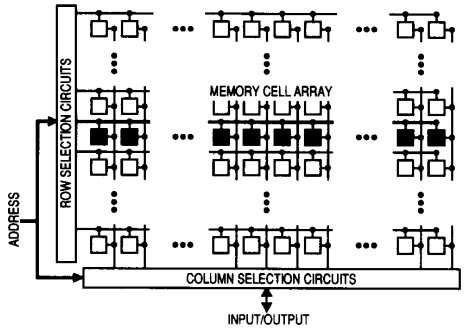

Fig. 1. Random access memory structure showing one word line chosen, allowing access to entire row.

There are five common types of hard error or hardware failure. The most common type by far is a single cell hard failure, where a defect occurs in a single cell, making it no longer able to reliably hold data [6]. There are also other less common failure modes. Failures of row selection circuitry (row failure) and column selection circuitry (column failure) cause an entire row or column to be unreliable. Shorting of row select and column sense lines (row–column failure) can cause a row and a column to fail simultaneously. Finally, the entire chip can fail if a chip selection or power circuit fails [6]. These types are shown in Fig. 4. Other techniques such as row and column replacement are used to increase the reliability of chips at the time of manufacture [7] and may be better suited to deal with row, column, and row–column failures; we do not address these techniques here. Also, we do not deal explicitly with failures of the addressing mechanisms here, since these are prone to cause multiple-row or multiple-column failures [8].

Additional hard-error types may occur from layout strategies used on large chips. For example, long selection lines cause the chip to be slow and error-prone due to large capacitance these lines have in relation to the cell capacitance or driving ability. Thus, large memory chips are broken into blocks, with each block having its own selection circuitry. There may be as few as two or as many as sixteen blocks of this type; large chips may have even more. This makes possible single and multiple block failures and full and partial row and column failures in addition to single cell, row–column, and entire chip failures as shown in Fig. 5. The number of types of errors increases as more blocks are employed since there are more failure modes possible.

As memory capacity per chip increases, the effect of these errors becomes too great for board level coding to handle. In particular soft errors can drastically reduce the effective mean time to failure of an individual chip. It is thus natural to move to on-chip ECC (error correction coding) in order to get higher chip reliability. In addition, it may be necessary to combine chip and board coding to maintain the required reliability [9]. The most natural architecture for on-chip ECC is to place a single error correcting code along each row of the memory. These are most typically SEC (single error-correcting) Hamming codes that are shortened to have an information length of some integer power of two. Thus, when a row is accessed, the entire codeword can be simultaneously read, then corrected if necessary, prior to outputting the single bit required by the column selection. If no more than a single error occurs along each row, the memory will remain functional and will reliably hold data. Furthermore, since soft errors can be removed by reading, correcting, and rewriting the data at regular intervals (soft-error scrubbing), repeated soft errors on a codeword will not cause memory failure unless more than one error occurs before the memory can be scrubbed.

Soft errors, however, can be clustered, as shown in Fig. 3, and therefore there must be some means to handle clustered errors. One way is to use burst error-correcting codes, but these tend to have higher redundancy and be more complex, resulting in more difficult and time-consuming encoding and decoding, than Hamming codes. These are not desirable characteristics. Instead, this effect is most commonly minimized by employing good layout format and spatially interleaved cells so that cells which belong to the same codeword are spaced apart, as shown in Fig. 6. This allows the use of single error correcting codes but requires that each row employ more than one codeword [5], [10]. Note also that by placing more than one codeword for each row and modeling alpha particles as induced clusters of errors, there arises the possibility that two soft-error strikes, even if not in the same codeword, can cause failure. A simple model assumes that each alpha particle effectively affects only a single cell and that two alpha particles must be in the same row to affect the same codeword; this assumption will be used here.

Hard errors, in contrast, cannot be removed, only corrected, by this method. Thus, one single-cell hard error in a codeword will put the memory in a state where the next error of any type in that codeword will cause memory failure. Likewise, one column failure in the chip will also cause the memory to fail with the next error of any type in *any* codeword in the same block, since this error causes a single error in every codeword in the memory block. Other types of hard errors will overwhelm the error-correcting code and cause memory failure with their first occurrence. Note also that placing more than one codeword for each row creates the possibility that multiple single cell or column hard errors may not cause failure.

Several memory chips which employ on-chip error correcting codes have been fabricated. A production chip fabricated by Micron Technologies employs a (12,8) Hamming code, interleave depth 4, on a 256 Kbit$\times$1 RAM [10]. Asakura *et al.*, have employed a (40,32) SEC–DED (single error-correcting–double error-detecting) modified Hamming code on a 1 Mbit cache DRAM [11]. Horiguchi, *et al.* have also constructed a RAM with coding, although this is a multilevel, 4 bits/cell 4 Mbit DRAM [5]. The code used is still a single error-correcting code, though it uses a (131,128) cyclic 16-ary code with interleave depth 8 to correct single-cell errors, the equivalent to four bit errors. Chiueh, Goodman, and Sayano [12] have constructed a 2 Kbit$\times$1 static RAM chip with a double error-correcting linear sum code proposed by Fuja, Heegard, and Goodman [13]. Most codes used in practice have been simple binary Hamming type codes, in order to reduce complexity of the on-chip decoder.

Most previous MTTF calculations were based entirely or in part on the binomial distribution [1], [14]–[17]. The results tended to be complex, and some were simplified by taking the Poisson approximation to the binomial distribution at one point or another. Using the Poisson distribution from the start, first done in [18], provides a far less complex yet accurate result. We will again use the Poisson distribution in this analysis. Furthermore, row-column type hard errors were first accurately modeled in [19]; previous papers did not address this failure mode and therefore inaccurately modeled the MTTF. We now extend our work to include the effect of soft error scrubbing in semiconductor random access memories coded with on-chip single error correcting codes and subject to both hard and soft errors.

lossless of finite order can be viewed as "deterministic with bounded delay."

Fig. 2. Example for bound of Theorem 5.

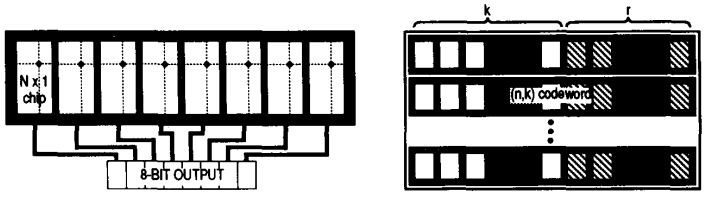

Fig. 2. Board-level organization of $N \times 1$ memory chips. SIMM's module and multipage board with ECC are shown. Each chip contributes one bit to output byte or error-correction codeword.

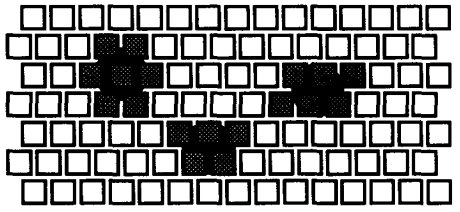

Fig. 3. Cluster error patterns created by alpha particle strikes (soft errors). Affected cells are darkened.

Analytical models such as the one presented here are becoming of increasing interest to system designers, as they enable an accurate assessment of the improvement in mean time to failure due to error control coding, without the need for lengthy and complex simulations, which are often prone to error. These analyses are particularly important for memory systems that must operate autonomously in harsh environments, such as those in oceanic probes, satellites, and spacecraft. Many such memory systems utilize similar coding schemes to those specifically analyzed here. In some cases, however, more complex coding schemes such as spares swapping, or concatenated on-chip and board-level coding, have been proposed. The techniques and models presented here should enable the coding theorist to extend our work to assess the performance of most of these current and proposed coded memory systems in an efficient manner.

In this correspondence, Section II presents the Poisson failure model which will be used in subsequent sections as the mathematical representation of the chip. Section III then develops the model for both hard and soft single-cell failures with soft-error scrubbing. Characteristics of the model will be explored for various limits of parameters. Section IV then expands the model to include multiple cell-hard failures (such as row, column, row-column, and entire chip failures). In addition, the case where chips are composed of multiple blocks of memory cell arrays will be addressed here. Section V presents experimental results from computer simulations, which help confirm the accuracy of the model for varying parameter values.

II. Poisson Failure Model

We would like to create an accurate model of the mean time to failure of a semiconductor memory with a single error-correcting code placed along each row. However, the model must also be computationally simple enough to ease its determination. The mathematical model used here will be the Poisson distribution for determination of probability of error in a given codeword. The accuracy of this model has been justified in our previous work [18]–[21]. A short review of the model is provided next.

The Poisson distribution gives the probability that there are no events (failures) in time t as

$$P_0(t) = e^{-\lambda t},$$

where λ is the mean event arrival rate, and the probability that there is exactly one event (failure) in time t as

$$P_1(t) = \lambda t e^{-\lambda t}.$$

The row reliability function is the probability that the row has not failed in time t; for a codeword protected by a single error-correcting code and error arrival rate per row λ, this becomes [21]

$$\begin{aligned} R(t) &= P(0 \text{ or } 1 \text{ error}) \\ &= (1+\lambda t)e^{-\lambda t}. \end{aligned}$$

By assuming independence among the rows, the chip reliability function, i.e., the probability that the entire chip has not yet failed by time t, is given by

$$R_{\text{chip}}(t) = R^M(t),$$

where there are M codewords in a chip.

The mean time to failure is

$$\text{MTTF} = \int_0^\infty R^M(t)\,dt$$

as shown in [21]. Modeling such a function may be time-intensive, so for simulation purposes we computed mean events to failure (METF). This is the average number of events which must occur before memory failure occurs; if this number is large, then by Wald's identity (and Little's Law, which states that, for Poisson processes, the mean waiting time—in this case, for failure—is the product of the mean number in the system and the inverse of the mean arrival rate) the approximation

$$\text{MTTF} = \frac{1}{\lambda}\text{METF} \tag{1}$$

can be used, because $1/\lambda$ represents the mean arrival rate of events (mean rate of errors) in the Poisson distribution [19].

III. Analysis of Single-Cell Failures

A. The MTTF Calculation

Initially the case of only single-cell hard and soft errors attacking the memory will be discussed. Furthermore, soft errors are confined to affecting only one cell. The symbols used are the following.

λ_h Single-cell hard-error arrival rate per cell.
λ_s Single-cell soft-error arrival rate per cell.

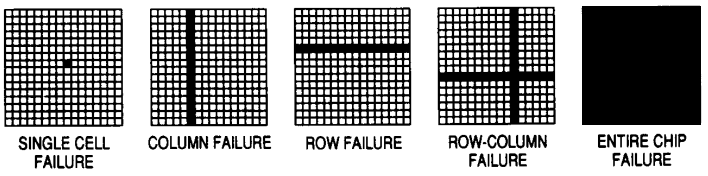

Fig. 4. Types of hard errors.

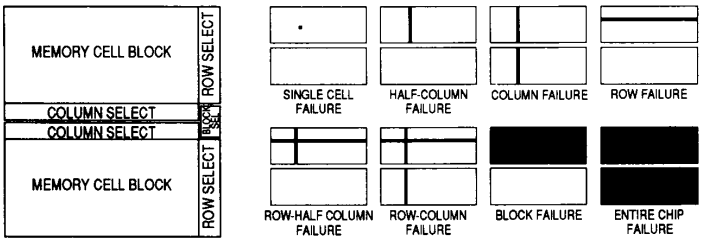

Fig. 5. Two-block ram chip and some associated types of hard errors.

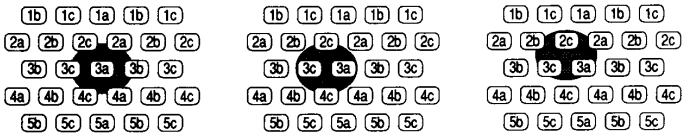

Fig. 6. Spatial interleaving allowing use of SEC codes. In this case interleave depth is 3 (three codewords per row, shown as a, b, and c).

λ_{sc} $\lambda_h + \lambda_s$ = total single-cell error arrival rate per cell.
t_s Soft-error-scrubbing period.
n Number of bits in a codeword.
M Number of codewords on a chip.

Soft-error scrubbing will be conducted. This means that at periodic intervals t_s, the entire chip will be subject to internal error correction where all codewords are read, corrected, and rewritten. This cannot remove hard errors, but it can remove soft errors, so long as no more than one error occurs in a single codeword. Note that no restriction is placed here on the number of codewords in each word line (in each row). Therefore, the word line may contain more than one codeword, such as in spatial interleaving. Thus, the number of codewords does not necessarily equal the number of rows on the chip. Also, there may be multiple blocks on the chip in a manner similar to that shown in Fig. 5.

The following two assumptions will be made.

Assumption 1: The scrub cycle must be small compared to the mean time to failure:

$$t_s \ll \text{MTTF}.$$

Since the scrub cycle (which, in dynamic RAM's, occurs in conjunction with the refresh cycle) is rarely longer than 100 seconds, a system with a mean time to failure which is not far greater than this is not reliable enough for use.

Assumption 2: The hard-error rate must be small compared to the soft-error rate:

$$\lambda_h \ll \lambda_s.$$

Hard errors cannot be removed by scrubbing and therefore accumulate; soft errors, which occur more frequently [6], can be removed. If hard errors occur with greater frequency than soft errors, then soft error scrubbing is useless ($\lambda_h \gg \lambda_s$ and λ_s is small enough so that scrubbing need not be done) or the chip is unreliable for use ($\lambda_h \gg \lambda_s$ and λ_h is too large for the chip to be used reliably). These assumptions are true for all practical memory chips; a rigorous argument for justifying these assumptions in the model, along with a more accurate but complex analysis, is presented in Appendix B. No mention is made of the reading and writing mechanism here because if scrubbing is done, each word is accessed within the period of one-scrub cycle. If failure is declared only after a word is accessed and found to be in error, since Assumption 1 holds true, the increased time before failure is declared is negligible compared to the case where failure is declared immediately following two failures in any codeword.

The situation is therefore this: While there are no hard errors in a codeword, the memory does not fail so long as no more than one soft error occurs in each time interval t_s in each codeword. If a hard error *does* occur, then no further errors can be tolerated in that same codeword. The two cases, when a hard error has not occurred and exactly one hard error has occurred by time t, will be treated separately.

The probability of codeword success at time t with no hard errors occurring is the probability that no hard errors have occurred and that only zero or one soft error has occurred in any time segment t_s from time 0 to t. Thus,

$$\begin{aligned} R_0(t) &= P(\text{no hard errors})P(0 \text{ or } 1 \text{ soft error}) \\ &= \left[e^{-\lambda_h nt}\right]\left[e^{-\lambda_s nt_s} + \lambda_s nt_s e^{-\lambda_s nt_s}\right]^{t/t_s} \\ &= e^{-\lambda_{sc} nt}(1 + \lambda_s nt_s)^{t/t_s}. \end{aligned} \tag{2}$$

The probability of codeword success at time t with the occurrence of exactly one hard error *before* time t is more complex. First, assume that the hard error had occurred at time τ.

Therefore, the probability of success is the joint probability that there were no codeword failures given no hard failures have occurred up to time τ, that exactly one hard error occurred between time τ and $\tau + d\tau$, and that neither soft nor hard errors occurred from time τ to t. This is then integrated over τ from 0 to t to consider all possible times that the hard error could have occurred.

$$\begin{aligned} R_1(t) &= \int_0^t R_0(\tau) P(1 \text{ hard error in } d\tau) P(\text{no errors in } \tau \text{ to } t) \\ &= \int_0^t R_0(\tau)[\lambda_h n\, d\tau][e^{-\lambda_{sc} n(t-\tau)}] \\ &= e^{-\lambda_{sc} nt} \lambda_h n \int_0^t (1+\lambda_s n t_s)^{\tau/t_s}\, dt \\ &= \frac{e^{-\lambda_{sc} nt} \lambda_h n t_s}{\ln(1+\lambda_s n t_s)} \left[(1+\lambda_s n t_s)^{t/t_s} - 1\right]. \end{aligned} \tag{3}$$

Therefore, the reliability function for each codeword is now given by combining (2) and (3).

$$\begin{aligned} R(t) &= R_0(t) + R_1(t) \\ &= e^{-\lambda_{sc} nt}(1+\lambda_s n t_s)^{t/t_s} \\ &\quad + \frac{e^{-\lambda_{sc} nt} \lambda_h n t_s}{\ln(1+\lambda_s n t_s)} \left[(1+\lambda_s n t_s)^{t/t_s} - 1\right]. \end{aligned} \tag{4}$$

The mean time to failure (MTTF) for a chip with M codewords is then

$$\begin{aligned} \text{MTTF} &= \int_0^\infty R^M(t)\, dt \\ &= \int_0^\infty e^{-\lambda_{sc} nMt} \left[\exp\left[\frac{t}{t_s}\ln(1+\lambda_s n t_s)\right] \right. \\ &\quad \left. \cdot \left(1 + \frac{t_s \lambda_h n}{\ln(1+\lambda_s n t_s)}\right) - \frac{t_s \lambda_h n}{\ln(1+\lambda_s n t_s)}\right]^M dt. \end{aligned} \tag{5}$$

As shown in Appendix A, this becomes

$$\begin{aligned} \text{MTTF} &= \sum_{i=0}^{M} \binom{M}{i} \\ &\quad \cdot \frac{\left(1 + \frac{t_s \lambda_h n}{\ln(1+\lambda_s n t_s)}\right)^i \left(-\frac{t_s \lambda_h n}{\ln(1+\lambda_s n t_s)}\right)^{M-i}}{\lambda_{sc} nM - \frac{i}{t_s}\ln(1+\lambda_s n t_s)}. \end{aligned} \tag{6}$$

Alternatively, by making the substitutions

$$y = \lambda_s n t_s$$

$$z = \frac{\lambda_h}{\lambda_s},$$

the mean time to failure can be represented in a clearer form:

$$\text{MTTF} = \frac{1}{\lambda_{sc} nM} \sum_{i=0}^{M} \binom{M}{i} \frac{\left(1 + \frac{yz}{\ln(1+y)}\right)^i \left(-\frac{yz}{\ln(1+y)}\right)^{M-i}}{1 - \frac{i}{M}\frac{\ln(1+y)}{y(z+1)}}. \tag{7}$$

The coding gain, which is the advantage of the mean time to failure over the uncoded case, is therefore

$$\begin{aligned} CG &= \frac{k}{n} \frac{\text{MTTF}_{\text{coded}}}{\text{MTTF}_{\text{uncoded}}} \\ &= \frac{k}{n} \sum_{i=0}^{M} \binom{M}{i} \frac{\left(1 + \frac{yz}{\ln(1+y)}\right)^i \left(-\frac{yz}{\ln(1+y)}\right)^{M-i}}{1 - \frac{i}{M}\frac{\ln(1+y)}{y(z+1)}}, \end{aligned} \tag{8}$$

where (n, k) codes are used.

The mean time to failure, therefore, is effectively a function of three parameters, y, z, and M. The parameter y represents the relation between the scrub interval and the soft-error rate; it is the mean number of soft errors that strike each codeword in a scrub period. The parameter z is the ratio of soft to hard error rates. Note that if λ_s, λ_h, and t_s are adjusted such that y and z are kept constant, then the coding gain remains unchanged: The parameter y can be kept constant by maintaining a constant $\lambda_s t_s$; the parameter z can be kept constant by altering λ_s and λ_h proportionally. Such a dependence implies that time is being scaled.

Note that (7) appears similar to a binomial expansion of order M, a consequence of expanding and integrating $R^M(t)$. Also, if y and z are small, then the denominator term will be small, and therefore each term of the sum will be large. This means that if the hard-error rate is small compared to the soft-error rate, and if soft errors are scrubbed out before they are allowed to accumulate, MTTF will be large, a conclusion which confirms intuition.

Subsequent subsections will explore various special cases of (7). Specifically, of interest are the limit of very fast scrubbing ($t_s \to 0$) and the limit of no hard errors ($\lambda_h \to 0$). In addition, an analysis will be made for cases when Assumptions 1 and 2 are violated. These cases will be examined for three reasons. First, these cases, as will be shown, will provide a connection between the results of previous work (most notably, [19]–[21]). Second, we wish to check if the model breaks down predictably, based on intuition, simulation, and previous work, if the assumptions are violated. Third, the special cases are examined for completeness.

B. MTTF with Fast Scrubbing

If the interval between each soft error scrub is allowed to decrease to zero, that is, scrubbing is done continuously, there is a maximum near time to failure which cannot be exceeded. In this limit $t_s \to 0$, or equivalently $y \to 0$, (7) becomes

$$\begin{aligned} \text{MTTF} &= \frac{1}{\lambda_{sc} nM} \sum_{i=0}^{M} \binom{M}{i} \frac{(1+z)^i (-z)^{M-i}}{1 - \frac{i}{M}\frac{1}{z+1}} \\ &= \frac{1}{\lambda_{sc} nM} \sum_{i=0}^{M} \binom{M}{i} \frac{\left(1 + \frac{\lambda_h}{\lambda_s}\right)^i \left(-\frac{\lambda_h}{\lambda_s}\right)^{M-i}}{1 - \frac{i}{M}\frac{1}{\lambda_h/\lambda_s + 1}}. \end{aligned} \tag{9}$$

The coding gain is therefore

$$CG = \frac{k}{n}\sum_{i=0}^{M}\binom{M}{i}\frac{\left(1+\frac{\lambda_h}{\lambda_s}\right)^{i}\left(-\frac{\lambda_h}{\lambda_s}\right)^{M-i}}{1-\frac{i}{M}\frac{1}{\lambda_h/\lambda_s+1}}. \tag{10}$$

Equations (9) and (10) are simpler expressions to evaluate than (7) and (8), and they show the dependence of MTTF on λ_h/λ_s. Thus, if $t_s \to 0$, the ratio between the hard- and soft-error rates becomes important.

Note that if λ_h/λ_s is extremely small, the last term of the summation is (9) the dominant term:

$$\text{MTTF} \approx \frac{1}{\lambda_{sc}nM}\frac{\left(1+\frac{\lambda_h}{\lambda_s}\right)^{M}}{\frac{\lambda_h}{\lambda_h+\lambda_s}}$$

$$\approx \frac{1}{\lambda_h nM} \tag{11}$$

$$CG \approx \frac{k}{n}\left(1+\frac{\lambda_s}{\lambda_h}\right). \tag{12}$$

This means that if the soft errors occur very quickly compared to the hard error rate but slow enough to be scrubbed out before more than one can accumulate in a single codeword, then the MTTF approaches that of the unprotected chip with only hard errors. This is reasonable, since when a hard error occurs, a soft error has a high probability of occurring relatively immediately after it in the same codeword, thus causing a failure. However, since the scrub interval is short enough, soft errors without a hard error in the same codeword never accumulate fast enough to cause failure. Thus, the model behaves as expected for $t_s \to 0$. Note that this provides an upper bound on performance: Given an error arrival rate of λ_s and λ_h, the MTTF cannot exceed (9) even with extremely fast soft-error scrubbing.

C. Soft Errors as the Dominant Error Type

For the case of only soft errors occurring, i.e., $\lambda_h \to 0$ (i.e., $z \to 0$) the chip can fail only if two or more soft errors occur in the same codeword and in the same scrub interval. We expect, therefore, that the mean time to failure will be

$$\text{MTTF} = \int_0^\infty R_0^M(t)\,dt$$

$$= \int_0^\infty e^{-\lambda_s nMt}(1+\lambda_s nt_s)^{Mt/t_s}\,dt$$

$$= \int_0^\infty \exp\left[-\lambda_s nMt + \frac{Mt}{t_s}\ln(1+\lambda_s nt_s)\right]dt$$

$$= \left[\lambda_s nM - \frac{M}{t_s}\ln(1+\lambda_s nt_s)\right]^{-1}. \tag{13}$$

If the model is accurate, then the same result should be achieved when (7) is used. By taking $z = 0$ in (7), we obtain

$$\text{MTTF} = \frac{1}{\lambda_{sc}nM}\sum_{i=0}^{M}\binom{M}{i}\frac{1^i 0^{M-i}}{1-\frac{i}{M}\frac{\ln(1+y)}{y}}$$

$$= \frac{1}{\lambda_s nM}\left[1-\frac{\ln(1+\lambda_s nt_s)}{\lambda_s nt_s}\right]^{-1}. \tag{14}$$

The coding gain is

$$CG = \frac{k}{n}\left[1-\frac{\ln(1+\lambda_s nt_s)}{\lambda_s nt_s}\right]^{-1}. \tag{15}$$

Equation (14) is the same as (13). Note that if $\lambda_s nt_s$ is held constant, then the coding gain remains constant, as expected: Time scaling should not affect the coding gain. Also note that forcing $t_s \to 0$ now yields MTTF $\to \infty$, as there are no hard errors to build up on the chip to cause two or more errors in a codeword during any single scrub cycle. The asymptotic behavior of this case can be determined by taking the Taylor series of $\ln(1+y)$:

$$\text{MTTF} = \frac{1}{\lambda_s nM}\left[1-\frac{\ln(1+y)}{y}\right]^{-1}$$

$$= \frac{1}{\lambda_s nM}\left[1-\frac{y-\frac{1}{2}y^2+\frac{1}{3}y^2\cdots}{y}\right]^{-1}$$

$$\approx \frac{1}{\lambda_s nM}\left[\frac{2}{y}\right] \tag{16}$$

$$CG \approx \frac{k}{n}\left[\frac{2}{y}\right], \tag{17}$$

for small y. Thus, the MTTF and the scrub interval are inversely related.

This analysis has given the maximum possible MTTF given t_s and λ_s for a highly reliable chip; the upper bound on performance for a given t_s and λ_s is therefore easily found using (13).

D. Very Slow Soft Scrubbing

For the case of very slow soft-error scrubbing, Assumption 1 is violated, and (7) does not apply. The analysis must be reconducted to develop a new representation. Violation of Assumption 1 without violation of Assumption 2, for typical values for the other parameters, corresponds to having a very long scrub period. This case is not realistic: Having a long scrub cycle is useless, since if many errors are allowed to occur in a single codeword before soft-error scrubbing is conducted the error correcting power of the code is more likely to be overwhelmed. Though this case is not expected to occur in reality, it will be studied to determine how chip performance can be expected to degrade. In this case, instead of the continuous scrub period analysis conducted thus far, a discrete scrub period analysis must be made. In previous sections, since Assumption 1 assured the passing of many scrub periods before a chip failure was expected to occur, a continuous scrub period analysis, as done in (2), was possible. The equivalent analysis using discrete scrub performance intervals is much more complex. (This analysis is covered briefly in Appendix B, where its complexity becomes evident.)

Instead, an analysis of the special case $t_s \to \infty$ will be conducted. This will provide a lower bound and an intuitive feel for

the behavior of the model. The model must be reconstructed from the reliability function, where

$$R_0(t) = e^{-\lambda_{sc}nt}[1+\lambda_s nt] \tag{18}$$

and

$$R_1(t) = \lambda_h nte^{-\lambda_{sc}nt} \tag{19}$$

are now used to determine the reliability function instead of (2) and (3). $R_0(t)$ represents the probability that the codeword has yet to fail given that no hard errors have occurred. In this case, $R_0(t)$ is the probability that zero or one soft error and no hard errors have occurred. $R_1(t)$ is the probability that the codeword has yet to fail given that exactly one hard error has occurred. Thus, the new reliability function is the sum of (18) and (19)

$$R(t) = e^{-\lambda_{sc}nt}[1+\lambda_{sc}nt]. \tag{20}$$

Therefore, the mean time to failure becomes

$$\begin{aligned}\text{MTTF} &= \int_0^\infty e^{-\lambda_{sc}nMt}[1+\lambda_{sc}nt]^M\,dt \\ &= \int_0^\infty e^{-\lambda_{sc}nMt}\sum_{i=0}^{M}\binom{M}{i}(\lambda_{sc}nt)^i\,dt \\ &= \frac{1}{\lambda_{sc}nM}\sum_{i=0}^{M}\binom{M}{i}\frac{i!}{M^i}.\end{aligned} \tag{21}$$

Thus, we have a factor of

$$B(M) = \sum_{i=0}^{M}\binom{M}{i}\frac{i!}{M^i} \tag{22}$$

in the equation that is only a function of the number of codewords in the chip. Note that the MTTF is greater than the uncoded mean time to failure by the factor $B(M)$. This factor $B(M)$ was analyzed as an extension of the Birthday Surprise problem in [19] and [20] and has been shown to be, for large M,

$$B(M) = \sqrt{\frac{\pi M}{2}} + \frac{2}{3} + O(M^{-1/2}). \tag{23}$$

Note that (23) is equivalent to the solution found via a different analysis in [22].

The coding gain, therefore, is

$$CG = \frac{k}{n}B(M) \tag{24}$$

and is no longer a function of the error arrival rates. This is reasonable, as the lack of soft error scrubbing removes the time dependence. Only the error-correcting code's correctional power is important, as reflected by $B(M)$. Some values of the $B(M)$ are listed in Table I. Note that for typical memory chips of the order of 1 Mbit and larger the error between the actual value of $B(M)$ and the value found by (23) is much less than 1%.

TABLE I
SOME VALUES OF FACTOR $B(M)$*

M	$B(M)$	% Error	M	$B(M)$	% Error
1	2.00	-4.00	2^{13}	144.1	-1.01×10^{-3}
2	2.50	-2.44	2^{14}	161.1	-5.05×10^{-4}
2^2	3.22	-1.41	2^{15}	227.5	-2.53×10^{-4}
2^3	4.25	-7.87×10^{-1}	2^{16}	321.52	-1.27×10^{-4}
2^4	5.70	-4.27×10^{-1}	2^{17}	454.42	-6.34×10^{-5}
2^5	7.77	-2.26×10^{-1}	2^{18}	642.36	-3.17×10^{-5}
2^6	10.71	-1.18×10^{-1}	2^{19}	908.16	-1.59×10^{-5}
2^7	14.86	-6.06×10^{-2}	2^{20}	1284.06	-7.93×10^{-6}
2^8	20.73	-3.09×10^{-2}	2^{21}	1815.66	-3.96×10^{-6}
2^9	29.03	-1.57×10^{-2}	2^{22}	2567.45	-1.98×10^{-6}
2^{10}	40.78	-7.93×10^{-3}	2^{23}	3630.65	-9.87×10^{-7}
2^{11}	57.39	-4.00×10^{-3}	2^{24}	5134.24	-4.90×10^{-7}
2^{12}	80.88	-2.01×10^{-3}			

*Also tabled are errors between approximation (23) and the exact values.

E. Hard-Error Dominance

For the case where hard errors dominate, Assumption 2 does not hold. A new model will be constructed in this section for this case. Violation of Assumption 2 corresponds to having hard errors as the dominant-error type. This is not a realistic situation: Hard errors are much less common than soft errors [6], and if they were not, either the chip itself is very unreliable and therefore not suitable for use (λ_h is too large) or soft-error scrubbing is not needed (λ_s is very small). In both cases, soft error scrubbing is useless because in the former, most of the errors cannot be removed by scrubbing, and in the latter, there are no soft errors to be scrubbed. However, for completeness we do consider this case.

For simplicity and for an intuitive feel of the behavior of the chip for very large λ_h/λ_s, the limiting case $\lambda_h/\lambda_s \to \infty$ will be examined. The reliability function can be reconstructed from the basic Poisson model, or it can be adapted from (4); both give equivalent models. This occurs because if the equations given in Appendix B, the correct complete analysis, were carried out, the range of t where $R(t)$ is significant will be such that $|t/t_s| < 1$. This is true because if hard errors are more common than soft errors, then they are likely to cause failure by themselves far more quickly than soft errors. Since soft errors are assumed to occur at a rate slow enough to be scrubbed out, any arrival of errors much faster than this implies Assumption 2, that the scrub interval is much less than the MTTF, is violated. Completing this analysis yields (4) as the reliability function.

Using the substitution $z = \lambda_h/\lambda_s$, the reliability function (4) becomes

$$\begin{aligned}R(t) = {} & e^{-\lambda_s(1+z)nt}\left(1+\frac{\lambda_h nt_s}{z}\right)^{t/t_s} \\ & + \frac{e^{-\lambda_s(1+z)nt}\lambda_h nt_s}{\ln\left(1+\frac{\lambda_h nt_s}{z}\right)}\left[\left(1+\frac{\lambda_h nt_s}{z}\right)^{t/t_s}-1\right].\end{aligned} \tag{25}$$

By letting $z \to \infty$ and using

$$\ln(1+x) \approx x \qquad \text{as } x \to 0$$

and

$$(1+x)^k \approx 1+kx, \qquad \text{as } x \to 0,$$

(25) becomes

$$\begin{aligned}R(t) &= e^{-\lambda_s znt} + e^{-\lambda_s znt}\frac{\lambda_h nt_s}{z} + e^{-\lambda_s znt}\lambda_h nt\frac{z}{\lambda_h nt_s}\frac{\lambda_h nt}{z}, \\ &= e^{-\lambda_h nt} + e^{-\lambda_h nt}\lambda_h nt, \\ &= e^{-\lambda_h nt}[1+\lambda_h nt].\end{aligned} \tag{26}$$

Note that (26) is the same form as (20); therefore

$$\text{MTTF} = \frac{1}{\lambda_h nM}\sum_{i=0}^{M}\binom{M}{i}\frac{i!}{M^i}. \tag{27}$$

Again, the factor given in (22)

$$B(M) = \sum_{i=0}^{M} \binom{M}{i} \frac{i!}{M^i}$$

appears in the equation. The coding gain is again given by (24)

$$CG = \frac{k}{n} B(M).$$

Equations (21) and (27), the results of Section III-D and this section, respectively, are identical except for the value of λ they employ; (21) uses λ_{sc} while (27) uses λ_h. The coding gain, however, is identical whenever either Assumptions 1 or 2 are violated.

IV. The General Case: Multiple Types of Hard Errors

A. Multiple Hard-Error Types with One Block per Chip

The preceding section dealt with single-cell hard and soft errors in an M-codeword memory encoded along the rows with a single error-correcting code. Here the situation will be the same, except that other types of hard errors, namely row failures, column failures, row–column failures, and entire chip failures will also be considered. Analysis will again be done via Poisson distribution. In this analysis, coding will be restricted to one codeword per row for the sake of simplicity, and the chip is assumed to be composed of one block.

Since the memory is encoded with codewords along rows, the entire chip can survive exactly one column failure. However, the subsequent appearance of a single error of *any* type anywhere in the chip will then cause a failure. Conversely, *any* other type of multiple-cell hard error (row, row–column, and entire chip failures) at *any* time will cause the chip to fail *immediately*. These latter types of errors will be designated catastrophic failures.

The symbols used are the following.

λ_h	Single-cell hard-error arrival rate per cell.
λ_s	Single-cell soft-error arrival rate per cell.
λ_{sc}	$\lambda_h + \lambda_s$ = total single-cell error-available rate per cell.
λ_c	Column failure rate per chip.
λ_r	Row-failure rate per chip.
λ_{rc}	Row–column-failure rate per chip.
λ_{wc}	Whole-chip failure rate per chip.
λ_f	$\lambda_r + \lambda_{rc} + \lambda_{wc}$ = catastrophic-failure rate per chip.
λ_{mc}	$\lambda_r + \lambda_{rc} + \lambda_{wc} + \lambda_c$ = multiple-cell error-arrival rate per chip.
λ	$\lambda_{mc} + M_n \lambda_{sc}$ = total error-arrival rate per chip.
t_s	Soft-error scrubbing period.
n	Number of bits in a codeword.
M	Number of codewords on a chip.

The analysis here is for when Assumptions 1 and 2 hold. Therefore, the codeword level reliability function remains unchanged from (4) in the range where the two assumptions are valid; however, the chip level reliability function is no longer the row reliability function raised to the power of M. The probability of the chip success at time t is controlled by the following: 1) There must be no chip level failures (multicell hard errors) or codeword level failures that cause the memory to fail; 2) if there is a column failure (a chip level failure) at time τ, there must be no further errors of any type from time τ to t.

The following:

$$\begin{aligned} R_c(t) &= P(\text{no multicell failures})P(\text{no single cell failures}) \\ &\quad + \int_0^t P(\text{no chip failures in 0 to } \tau) \\ &\quad \cdot P(1 \text{ column failure in } d\tau) P(\text{no errors in } \tau \text{ to } t), \\ &= \left[e^{-\lambda_{mc} t}\right]\left[\left[R_0(t) + R_1(t)\right]^M\right] \\ &\quad + \int_0^t \left[e^{-\lambda_{mc}\tau} R_0^M(\tau)\right]\left[\lambda_c \, d\tau\right]\left[e^{-\lambda(t-\tau)}\right], \\ &= e^{-\lambda t}\left[e^{c_1 t} c_2 + c_3\right]^M + \lambda_c e^{-\lambda t} \int_0^t e^{c_1 M \tau} d\tau, \end{aligned} \tag{28}$$

where, as shown in Appendix A,

$$c_1 = \frac{1}{t_s} \ln(1 + \lambda_s n t_s),$$

$$c_2 = 1 + \frac{t_s \lambda_h n}{\ln(1 + \lambda_s n t_s)},$$

$$c_3 = -\frac{t_s \lambda_h n}{\ln(1 + \lambda_s n t_s)}.$$

Integration of (28) yields a chip reliability function of

$$R_c(t) = e^{-\lambda t}\left[e^{c_1 t} c_2 + c_3\right]^M + \frac{\lambda_c e^{-\lambda t}}{M c_1}\left[e^{c_1 M t} - 1\right].$$

The mean time to failure (MTTF) is now

$$\begin{aligned} \text{MTTF} &= \int_0^\infty R_c(t)\, dt, \\ &= \left[\int_0^\infty e^{-\lambda t}\left[e^{c_1 t} c_2 + c_3\right]^M dt\right] \\ &\quad + \left[\frac{\lambda_c}{M c_1} \int_0^\infty \left(\exp\left[-\lambda t + c_1 M t\right] - \exp\left[-\lambda t\right]\right) dt\right], \\ &= \left[\sum_{i=0}^{M} \binom{M}{i} \frac{c_2^i c_3^{M-i}}{\lambda - c_1 i}\right] + \left[\frac{\lambda_c}{M c_1}\left(\frac{1}{\lambda - M c_1} - \frac{1}{\lambda}\right)\right], \\ &= \left[\sum_{i=0}^{M} \binom{M}{i} \frac{c_2^i c_3^{M-i}}{\lambda - c_1 i}\right] + \frac{\lambda_c}{\lambda}\left[\lambda - M c_1\right]^{-1}, \\ &= \sum_{i=0}^{M} \binom{M}{i} \frac{\left(1 + \dfrac{t_s \lambda_h n}{\ln(1 + \lambda_s n t_s)}\right)^i \left(-\dfrac{t_s \lambda_h n}{\ln(1 + \lambda_s n t_s)}\right)^{M-i}}{\lambda - \dfrac{i}{t_s} \ln(1 + \lambda_s n t_s)} \\ &\quad + \frac{\lambda_c}{\lambda}\left[\lambda - \frac{M}{t_s} \ln(1 + \lambda_s n t_s)\right]^{-1}. \end{aligned} \tag{29}$$

The coding gain is

$$CG = \frac{k}{n\lambda} \sum_{i=0}^{M} \binom{M}{i} \frac{\left(1 + \dfrac{t_s \lambda_h n}{\ln(1 + \lambda_s n t_s)}\right)^i \left(-\dfrac{t_s \lambda_h n}{\ln(1 + \lambda_s n t_s)}\right)^{M-i}}{\lambda - \dfrac{i}{t_s} \ln(1 + \lambda_s n t_s)} + \frac{k \lambda_c}{n \lambda^2}\left[\lambda - \frac{M}{t_s} \ln(1 + \lambda_s n t_s)\right]^{-1}, \tag{30}$$

where (n, k) codes are used.

Thus, the mean time to failure in (29) is modified from (6) by two factors. First, instead of the total single-cell-error

rate $Mn\lambda_{sc}$ in the denominator of the summation, the total error rate λ is used. Since λ is slightly larger than $Mn\lambda_{sc}$, the denominator has increased slightly; thus, each term of the summation in (29) is slightly smaller than is (6), leading to a smaller overall sum in (29) than in (6). Second, an additive term which increases the MTTF is present. This second term in (29) can be approximated as follows:

$$\frac{\lambda_c}{\lambda}\left[\lambda - \frac{M}{t_s}\ln(1+\lambda_s n t_s)\right]^{-1} \le \frac{\lambda_c}{\lambda}\left[\lambda - \frac{M}{t_s}\lambda_s n t_s\right]^{-1},$$

$$= \frac{\lambda_c}{\lambda}[\lambda_{mc} + Mn\lambda_h]^{-1},$$

$$= \frac{1}{\lambda}\left[\frac{\lambda_c}{\lambda_{mc} + Mn\lambda_h}\right]. \quad (31)$$

The additive term, therefore, is shown by (31) as being small compared to the summation term in (30). In typical memory chips, λ_c, the rate of column failure, is less than 10% of $\lambda_{mc} + Mn\lambda_h$, the total hardware failure rate [21]. Therefore, the additive term in (30) is typically less than one-tenth the mean time to failure for an unprotected chip. A protected chip can be expected to have a much longer MTTF than an unprotected one (i.e., $CG \gg 1$), so the additive term is insignificant in most cases to the total MTTF.

B. Multiple Hard-Error Types with Multiple Blocks per Chip

If the chip is subdivided into several identical blocks as shown in Fig. 5, some assumptions need to be made regarding the structure of the chip and its failure modes. The simplest case to analyze (and the design with the simplest and most dense construction) is when, as shown in [6], row, row–column, and column errors occur across the entire chip, so that the only additional hard error type from the five discussed in Section IV-A is block failures. (Chips of this type are those with only one set of selection circuitry instead of separate selection circuits for each block and read–write circuits impervious to failure. There can be multiple-block failures, but since any type of block failure, as well as row and row–column failures, are catastrophic, the error-arrival rates of all of these errors can be lumped together:

$$\lambda_f = \lambda_r + \lambda_{rc} + \lambda_{wc} + \lambda_{\text{block}}$$

$$\lambda_{\text{block}} = \sum \text{ arrival rates of all types of block failures.}$$

Only the error arrival rate λ_{sc} is altered, and the equation for mean time to failure remains unchanged from (29).

For other cases, such as complexity independent selection and read–write circuitry for each block, the analysis becomes complex. Since single-cell failures are the most common type of hardware failure, and since complex models do not provide the insight of more simple ones, analysis of these complex cases will not be addressed here.

V. Simulation Results

Theoretical expectations were calculated and compared with simulation results. Testing the single-cell simple result (6) was done because it is simpler than testing the multiple hard-cell failure-mode model (29) and because (29) is based on (6). Since mean time to failure simulations are very time-consuming, the mean events to failure (METF) were computed. MTTF can be determined from METF and the mean-error arrival rate λ via (1):

$$\text{MTTF} = \frac{1}{\lambda}\text{METF}.$$

The determination of the METF was obtained by first assuming the following: Since typical chips have much higher probabilities of single-cell errors, both hard and soft, than other types of errors, the error rate used was λ_{sc}, the single-cell error arrival rate, not λ, the total arrival rate. Also, in most realistic cases, Assumptions 1 and 2 hold, so the hard-error arrival rate is much lower than the soft-error arrival rate, and the refresh cycle length is shorter than the mean time between soft-error arrivals.

Using these two assumptions, the METF was computed as follows. An error of either type (hard or soft) was assumed to occur, with any number of additional errors determined randomly with a Poisson distribution, in a single scrub cycle time period. The type of error was determined randomly so that the average ratio of errors was λ_h/λ_s. These errors were allowed to occur in any one of M codewords. If more than one error occurred in any codeword, a failure was declared. Scrubbing was conducted after these error insertions; all soft errors were cleared while hard errors remained.

The number of errors required to result in a chip failure were recorded and averaged; this provided the mean events to failure. This procedure is justified by the following. Since errors occur only occasionally in each scrub cycle, those scrub cycles where no error occurs can be ignored; this means that all scrub cycles where *at least* one error of either type occurs are examined but all other scrub cycles are ignored. Since the mean events to failure is typically large—the effect of soft errors is diminished by scrubbing and hard errors are rare—the mean time to failure can be approximated by the product of the mean-error interarrival time $1/\lambda_{sc}$ and the mean events to failure.

Results were obtained by averaging over 2000 tries; some are plotted next. All parameters are per-chip: Codewords per chip, hard-error arrival rates per chip, and soft-error arrival rates per chip. Thus, there is no dependence on the length of the codeword n. The parameters used here are defined as follows:

$\lambda'_h = \lambda_h nM$ = single-cell hard-error arrival rate per chip;

$\lambda'_s = \lambda_s nM$ = single-cell soft-error arrival rate per chip.

A. Varying the Scrub Interval

Fig. 7 shows the effect of keeping the error rates constant while the scrub cycle time was varied. This log plot shows that there is a threshold characteristic for soft error scrubbing; if the chip is scrubbed faster than some rate, there is little increase in MTTF. Also, if the chip is scrubbed much slower, the MTTF does not decrease much beyond some other level. This transition is directly linked to the failure mode of the chip: When the scrub cycle time is slow, then most of the chip failures are of the soft-soft type, where two soft errors cause chip failure, because $\lambda'_s \gg \lambda'_h$. These failure modes were confirmed through simulations. When the chip is scrubbed at a faster rate, the failures are

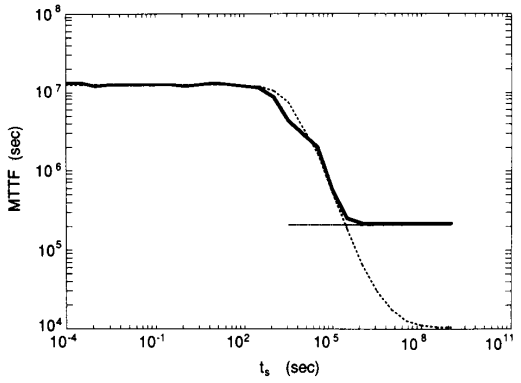

Fig. 7. MTTF with $\lambda'_h = 10^{-7}$ sec^{-1}, $\lambda'_s = 10^{-4}$ sec^{-1}, $M = 256$ codewords. Model (7) in dashed thin line; model (21) in dot-dashed thin line; simulation results in bold solid line.

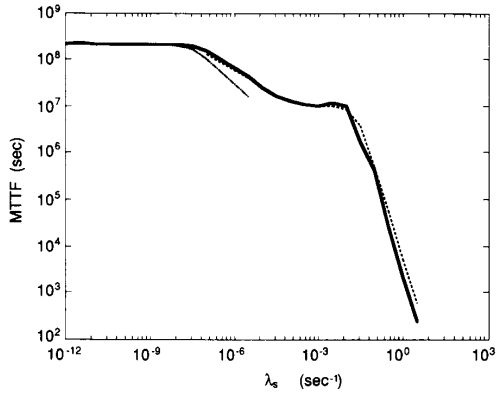

Fig. 8. MTTF with $\lambda'_h = 10^{-7}$ sec^{-1}, $t_s = 10^{-1}$ sec, $M = 256$ codewords. Model (7) in dashed thin line; model (27) in dot-dashed thin line; simulation results in bold solid line.

of the soft-hard type, where one soft and one hard error in a codeword causes chip failure.

Note that the model (7) matches the simulation results closely except for large values of scrub cycle time, where model (21) holds. The data smoothly switches from one model to the other. For small t_s, the MTTF should be as given in (9) or (11); this is shown in Fig. 7. In the case plotted, since $\lambda_h \ll \lambda_s$, (11) should hold:

$$\text{MTTF} = \frac{1}{\lambda_h n M}.$$

This is indeed the case, as $\lambda'_h = 10^{-7}$ sec^{-1}, and the MTTF levels off at about 10^7 sec. For large t_s, we expect the mean time to failure to be as given in (21):

$$\text{MTTF} = \frac{1}{\lambda_{sc} n M} \sum_{i=0}^{M} \binom{M}{i} \frac{i!}{M^i}$$

and for $M = 256$ we have

$$B(M) = \sum_{i=0}^{M} \binom{M}{i} \frac{i!}{M^i} = 20.73.$$

Again the model matches simulation, as $\lambda'_{sc} = 10^{-4}$ sec^{-1} and the MTTF levels off at 2.1×10^5 sec.

Thus, there are two flat regions for the MTTF. The MTTF is expected to be near the hard error rate if scrubbing is fast, because soft errors do not have time to build up and hard errors remain. Failure primarily occurs from having a soft error occurring in the same codeword as a hard error (which is not removed by scrubbing). Since the hard-error rate is much slower than the soft-error rate, the relative time period between a hard-error strike and a subsequent soft-error strike in the same codeword is small compared to the time period from start to first hard-error strike. Therefore, the MTTF approaches the MTTF with no protection and only hard errors, as expected by (11):

$$\text{MTTF} \approx \frac{1}{\lambda_h n M}, \qquad \text{as } t_s \to 0,\ \lambda_h \ll \lambda_s.$$

When scrubbing is slow, the model described by (21) in Section III-D applies, and this predicts a plateau at

$$\text{MTTF} = \frac{1}{\lambda_{sc} n M} B(M),$$

when $t_s \to \infty$. Note that in this case, soft-error scrubbing is not effective; therefore, the MTTF is determined mainly by the number of codewords and not the arrival times or scrub periods.

A point of interest to system designers would be the knee of the curve, where the MTTF drops from its higher plateau. In Fig. 7 this occurs at about 10^4 seconds, when the MTTF drops lower than 10^7 seconds. This is the maximum scrub interval permissible before system performance degrades significantly. Taking the point when the MTTF is $1/\sqrt{2}$ that of its plateau value as the knee, experimentally we have found that

$$\lambda_s n t_s = \alpha \frac{\lambda_h}{\lambda_s}, \tag{32}$$

where $0.8 < \alpha < 1$ gives the location of this corner point. The parameter α for relatively small t_s remains at about 0.83; for larger t_s, $\alpha \to 1$. Since the knee was defined at an arbitrary point, this relation provides only a rule-of-thumb maximum for t_s:

$$t_s < \alpha \frac{\lambda_h}{\lambda_s^2 n}. \tag{33}$$

B. Varying the Soft-Error Rate

If the soft-error rate is altered, then as shown in Fig. 8, there are two plateaus in the MTTF. Again, model and simulation are close. If λ'_s is very large, then most of the failures are of the soft–soft type, and soft-error scrubbing is not removing soft errors fast enough. Thus, as the soft-error arrival rate increases, MTTF decreases. The decrease is linear on the log–log plot, and this confirms (14), the case where soft errors dominate, or

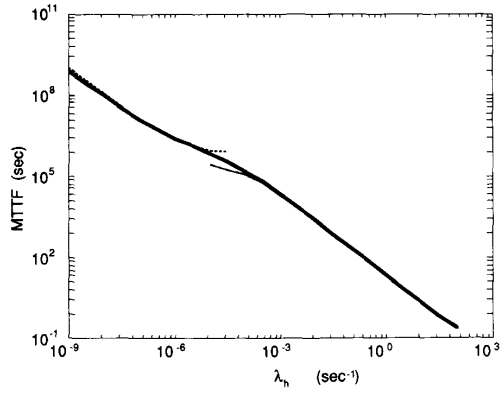

Fig. 9. MTTF with $\lambda'_s = 10^{-4}$ sec^{-1}, $t_s = 10^{-1}$ sec, $M = 256$ codewords. Model (7) in dashed thin line; model (27) in dot-dashed thin line; simulation results in bold solid line.

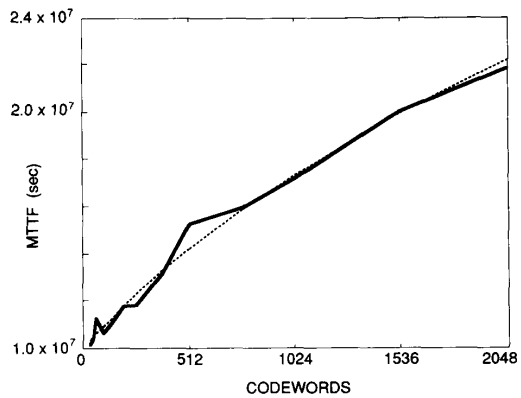

Fig. 10. MTTF with $\lambda'_h = 10^{-7}$ sec^{-1}, $\lambda'_s = 10^{-4}$ sec^{-1}, $t_s = 10^{-1}$ sec. Model (7) in dashed thin line; simulation results in bold solid line.

$\lambda_h \to 0$ in relation to λ_s:

$$\text{MTTF} = \frac{1}{\lambda_s nM}\left[1 - \frac{\ln(1+\lambda_s n t_s)}{\lambda_s n t_s}\right]^{-1},$$

$$\sim \frac{1}{\lambda_s nM}.$$

In the flat center region, most of the failures are the soft–hard type; the MTTF remains relatively unchanging: Here, scrubbing removes soft errors quickly enough, not allowing more than one soft error per codeword before wiping them clean. Hard errors, however, remain and effectively neutralize the advantage of coding; only then can a single soft-error cause a failure: The MTTF in this region is approximately the hard error rate λ'_h. Since the hard error rate is constant, the MTTF also is slow to change. But as λ'_s decreases further, it takes longer for the next soft strike to hit a codeword already containing a hard strike, so the MTTF increases, until the hard–hard failure mode dominates for $\lambda'_s \ll \lambda'_h$. This is the case where soft errors are practically nonexistent and therefore hard errors are dominant: Assumption 2 is no longer valid, thus requiring the analysis given in Section III-E and the use of (27). The MTTF levels off at approximately $B(M)$ times the flat center region (soft–hard failure mode) as expected, and since λ'_h is kept constant, the curve is flat.

Note that the lower knee of this curve, as the failure mode shifts from soft–hard to soft–soft, occurs at the same point found by (32) in Section V-A,

$$\lambda_s n t_s = \alpha \frac{\lambda_h}{\lambda_s},$$

where $0.8 < \alpha < 1$. In Fig. 8 this occurs at $\lambda_s \approx 0.1$ sec^{-1}. Solving (32) for λ_s yields

$$\lambda_s < \sqrt{\frac{\alpha \lambda_h}{t_s n}}. \tag{34}$$

C. Varying the Hard-Error Rate

The effect of altering the hard error rate λ'_h is shown in Fig. 9. For the left half of the curve, the failure modes are primarily soft–hard. This means that soft errors are scrubbed out fast enough to prevent accumulation in a codeword during a single scrub cycle. Also, hard errors are much less frequent than soft errors, so the codeword fails primarily when a soft error strikes a codeword whose error-correcting power has been negated previously by a hard error. This is the case described by (11),

$$\text{MTTF} = \frac{1}{\lambda_h nM}.$$

Fig. 9 is roughly linear with a negative slope in this region, therefore confirming the inverse relation given by (11). Note again that the model tracks the simulation closely.

Had the failure mode been soft–soft, that is, the no hard-error case discussed in Section III-C, the MTTF would have taken the form given by (14)

$$\text{MTTF} = \left[\lambda_{sc} nM - \frac{M}{t_s}(1 + \lambda_s n t_s)\right]^{-1},$$

and would therefore be inversely related to λ_s, not λ_h. This failure mode is not visible in Fig. 9. Had the failure mode been hard–hard, as it is for the right half of the curve, then Assumption 2 is not valid, so the model developed in Section III-E (27) must be used. This is also plotted on Fig. 9, and is shown to track the experimental data well. Equation (27) is inversely proportional to λ'_h, but since (27) is similar to (11) and is only scaled by the factor $B(M)$, an increase in MTTF is expected; this is shown in the slight upward perturbation in the plot between the regions where the two models (11) and (27) meet.

D. Varying the Number of Codewords

Fig. 10 shows the effect on the MTTF of altering the number of codewords M while the per-chip error-arrival rates remain constant. This is the case where the chip is coded with varying size codewords while the number of cells remains constant. As shown in Fig. 10, the increase in MTTF from increasing the number of codewords is small; in previous plots, the increases have been over several orders of magnitude. Here, the range is linear, and the MTTF is barely doubled by squaring the number of codewords in the chip. This confirms the relation found in [21]:

$$\text{METF} \sim \sqrt{M}. \tag{35}$$

Again, the model is close to the simulation results.

VI. Conclusion

We have presented an accurate model for the mean time to failure of a semiconductor RAM protected with a single error-correcting code along its word lines. The model takes into account all types of hard errors, single-cell soft errors, and soft-error scrubbing to derive the mean time to failure, thus providing a complete picture of the gain obtained by coding a memory chip with a single error-correcting code along the rows. The equation for MTTF presented can be directly used by the system designer wishing to assess the benefits of coding for memory protecting.

VII. Acknowledgment

The authors would like to thank Robert J. McEliece for comments and suggestions.

Appendix A

Here we solve the integral for MTTF for the simple case of only single-cell hard and soft failures. From (5), the mean time to failure for a chip with M codewords is given by

$$\mathrm{MTTF} = \int_0^\infty e^{-\lambda_{sc} n M t}\left[\exp\left[\frac{t}{t_s}\ln(1+\lambda_s n t_s)\right]\left(1+\frac{t_s\lambda_h n}{\ln(1+\lambda_s n t_s)}\right) - \frac{t_s\lambda_h n}{\ln(1+\lambda_s n t_s)}\right]^M dt.$$

This is of the form

$$\mathrm{MTTF} = \int_0^\infty e^{c_0 t}\left[e^{c_1 t}c_2 + c_3\right]^M dt, \tag{36}$$

where

$$c_0 = -\lambda_{sc} n M,$$

$$c_1 = \frac{1}{t_s}\ln(1+\lambda_s n t_s),$$

$$c_2 = 1+\frac{t_s\lambda_h n}{\ln(1+\lambda_s n t_s)},$$

$$c_3 = -\frac{t_s\lambda_h n}{\ln(1+\lambda_s n t_s)}.$$

Equation (36) can be evaluated by making the substitution $y = e^{c_1 t}$:

$$\mathrm{MTTF} = \frac{1}{c_1}\int_1^\infty y^{(c_0/c_1)-1}\left[y c_2 + c_3\right]^M dy,$$

$$= \sum_{i=0}^{M}\binom{M}{i}\frac{c_2^i c_3^{M-i} y^{(c_0/c_1)+i}}{c_0+ic_1}\Bigg|_1^\infty,$$

which does not converge unless

$$y^{(c_0/c_1)+i} \to 0 \text{ as } y \to \infty,$$

which means that $c_0 + c_1 i < 0$. Since $c_0 < 0$ and $c_1 > 0$, the condition may be violated when i is as large as possible, i.e., when $i = M$. But $\ln(1+x) < x$, so

$$\begin{aligned} c_0 + c_1 M &= -\lambda_{sc} n M + \frac{M}{t_s}\ln(1+\lambda_s n t_s), \\ &< -\lambda_{sc} n M + \lambda_s n M, \\ &= -\lambda_h n M, \\ &< 0. \end{aligned}$$

Also, the denominator cannot be zero for any i, $0 \le i \le M$:

$$\lambda_{sc} n M \ne \frac{i}{t_s}\ln(1+t_s n\lambda_s), \qquad \text{for } 0 \le i \le M. \tag{37}$$

Using $\ln(1+x) < x$ again,

$$\lambda_s n i > i\frac{\ln(1+\lambda_s n t_s)}{t_s}.$$

Since $\lambda_{sc} > \lambda_s$, and since $i \le M$, the condition set forth in (37) is satisfied.

The equation for mean time to failure becomes

$$\mathrm{MTTF} = -\sum_{i=0}^{M}\binom{M}{i}\frac{c_2^i c_3^{M-i}}{c_0+c_1 i} = \sum_{i=0}^{M}\binom{M}{i}\frac{\left(1+\frac{t_s\lambda_h n}{\ln(1+\lambda_s n t_s)}\right)^i\left(-\frac{t_s\lambda_h n}{\ln(1+\lambda_s n t_s)}\right)^{M-i}}{\lambda_{sc} n M - \frac{i}{t_s}\ln(1+\lambda_s n t_s)}. \tag{6}$$

Appendix B

In Section III-A, Assumptions 1 and 2 were presented and justified as reasonable characteristics of real chips.

Assumption 1: The scrub cycle must be small compared to the mean time to failure:

$$t_s \ll \mathrm{MTTF}.$$

Assumption 2: The hard-error rate must be small compared to the soft-error rate:

$$\lambda_h \ll \lambda_s.$$

These assumptions must hold in the model described by (6) as well for the following reasons. In (2) the probability of correct operation given that no hard errors occurred was presented as

$$R_0(t) = e^{-\lambda_{sc} n t}(1+\lambda_s n t_s)^{t/t_s}.$$

This was derived by using the approximation that a smooth, continuous function that equals another piecewise continuous function at regular intervals can be used to approximate that piecewise function. That piecewise function arises from the discrete time analysis of the soft-error scrubbing effect, and it is, for the case of having no hard errors in the codeword,

$$\begin{aligned} R_0'(t) &= \left[e^{-\lambda_h t}\right]P(0 \text{ or } 1 \text{ soft error in time } 0 \text{ to } \lfloor t/t_s\rfloor) \\ &\quad \cdot P(0 \text{ or } 1 \text{ soft error in time } \lfloor t/t_s\rfloor \text{ to } t), \\ &= \left[e^{-\lambda_h n t}\right]\left[\left(e^{-\lambda_s n t_s} + \lambda_s n t_s e^{-\lambda_s n t_s}\right)^{\lfloor t/t_s\rfloor}\right] \\ &\quad \cdot\left[e^{-\lambda_s n(t-\lfloor t/t_s\rfloor t_s)}\left[1+\lambda_s n\left(t-\lfloor t/t_s\rfloor t_s\right)\right]\right], \\ &= e^{-\lambda_{sc} n t}(1+\lambda_s n t_s)^{\lfloor t/t_s\rfloor}\left[1+\lambda_s n t_s\left(t/t_s - \lfloor t/t_s\rfloor\right)\right], \end{aligned} \tag{38}$$

where $\lfloor x\rfloor$ is the largest integer less than x.

R_0' equal in value to R_0 when t/t_s is an integer; that is, $t/t_s = \lfloor t/t_s\rfloor$. In between these points, when t/t_s is not an integer, $\lfloor t/t_s\rfloor$ is constant at the last integer value. R_0 and R_0'

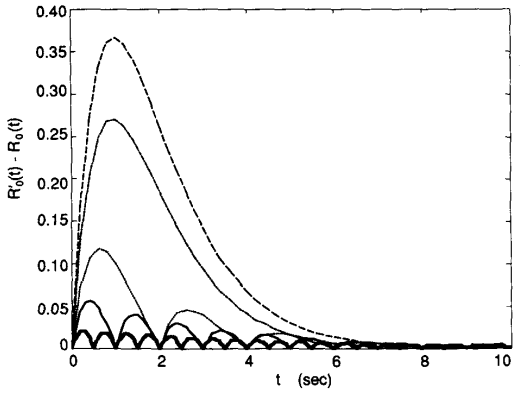

Fig. 11. Difference between $R_0'(t)$ and $R_0(t)$. Parameters were $\lambda_h = 10^{-7}$ sec^{-1}, $\lambda_s = 10^{-4}$ sec^{-1}, and $n = 1024$. Dashed line, $t_s = 1000$ sec; dot-dashed line, $t_s = 10$ sec; dotted line, $t_s = 2$ sec; solid line, $t_s = 1$ sec; bold solid line, $t_s = 0.5$ sec. Note that as t_s decreases the maximum difference between $R_0'(t)$ and $R_0(t)$ diminishes.

are of the form,

$$R_0(t) = e^{-\lambda t}(1+\lambda_s n t_s)^{t/t_s},$$
$$= e^{-\lambda t} Q(t),$$
$$R_0'(t) = e^{-\lambda t}(1+\lambda_s n t_s)^{\lfloor t/t_s \rfloor}\left[1+\lambda_s n t_s (t/t_s - \lfloor t/t_s \rfloor)\right],$$
$$= e^{-\lambda t} Q'(t).$$

As can be seen, $Q'(t)$ is a piecewise linear approximation of $Q(t)$, a monotonically increasing function. Therefore, $Q'(t) > Q(t)$, and thus, $R_0' > R_0$. Thus, (2) is a lower bound of the exact solution (38) using the Poisson approximation. However, $R_0 = R_0'$ at values of t determined by t_s and R_0 does not vary far from R_0' when t_s is sufficiently small. t_s is sufficiently small when $t_s \ll$ MTTF. This confirms the need for Assumption 1, as otherwise $R_0(t)$ will not be a good approximation for the more accurate $R_0'(t)$. (See Fig. 11.)

Note that the discrete scrub period analysis is much more complex than the continuous scrub period case, as is evident by the complexity of the reliability function. Carrying through this analysis will yield a mean time to failure model that is more complex to derive and interpret than the model presented in Section III-A, and is therefore unsuitable for use.

In (3), another assumption was made, that Assumption 2 must hold as well as Assumption 1. This assumption assures that $R_1(t)$ is a good approximation of $R_1'(t)$ in the following manner. $R_1(t)$ is the integral over τ of the product of three terms: First the probability that no failure had occurred given no hard error had occurred from 0 to τ; second, the probability that a hard error occurred in time $d\tau$; third, the probability that no error, hard or soft, occurred between τ and t. Thus, the first term is $R_0(\tau)$. This by itself justifies the use of the Assumption 1 here.

In addition, the following analysis is needed. Without soft errors, the peak of $R_1(t)$ occurs at $1/\lambda_h$. This peak will be shifted toward zero with the occurrence of soft errors. Since $R_0(t)$ is not accurate for large t_s—and equivalently small λ_s, due to the time scaling property of the model as described in (7)—there must be sufficiently many scrub cycles before $t = 1/\lambda_h$. This means that $1/\lambda_h \gg t_s$. Now $\lambda_s n t_s \sim 1$ so that soft errors do not pile up in any one codeword during one scrub cycle. Also, if $\lambda_s n t_s \gg 1$, there will be a high probability that two or more errors will occur in a single codeword within a scrub cycle, so in this case fewer errors (and therefore fewer scrub cycles) are needed to cause a failure. Thus, $\lambda_s n t_s \sim 1$ must be maintained to avoid violation of Assumption 1. This leads to

$$\frac{1}{\lambda_h} \gg t_s \sim \frac{1}{\lambda_s}$$

or

$$\frac{\lambda_h}{\lambda_s} \ll 1.$$

Thus, Assumption 2 must hold.

References

[1] D. Y. Koo and H. B. Chenoweth, "Choosing a practical model for ECC memory chip," in *Proc. 1984 IEEE Reliability and Maint. Symp.*, 1984, pp. 255–261.

[2] 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, pp. 124–134, Mar. 1984.

[3] T. Fuja and C. Heegard, "Focused codes for channels with skewed errors," *IEEE Trans. Inform. Theory*, vol. 36, no. 4, pp. 773–783, July 1990.

[4] T. C. May and M. H. Woods, "Alpha-particle-induced soft errors in dynamic memories," *IEEE Trans. Electron Devices*, vol. ED-26, pp. 2–9, Jan. 1979.

[5] M. Horiguchi *et al.*, "An experimental large-capacity semiconductor file memory using 16-levels/cell storage," *IEEE J. Solid-State Circuits*, vol. SC-23, pp. 27–33, Feb. 1988.

[6] D. Marston, "Memory system reliability with ECC," Intel Application Note AP-73, Intel Corp., 1980.

[7] T. Fuja and C. Heegard, "Row/column replacement for the control of hard defects in semiconductor RAM's," *IEEE Trans. Comput.*, vol. C-35, pp. 996–1000, Nov. 1986.

[8] T. Fuja, "Coding for the address-defect channel," in *24th Ann. Conf. Inform. Sci. Syst.*, Princeton, NJ, Mar. 21–23, 1990.

[9] ___, "The performance of random access memory systems employing on-chip and board-level error control," in *IEEE Int. Symp. Inform. Theory*, Kobe, Japan, 1988.

[10] Micron Technology Inc., "Effect of on-chip ECC on system soft errors," Micron Technology Inc. MT1256 Data Shet.

[11] M. Asakura *et al.*, "An experimental 1-Mbit cache DRAM with ECC," *IEEE J. Solid-State Circuits*, vol. SC-25, pp. 5–10, Feb. 1990.

[12] T. Chiueh, R. M. F. Goodman, and M. Sayano, "A 2K×1 static RAM chip with on-chip error correction," *IEEE J. Solid-State Circuits*, vol. 25, pp. 1290–1294, Oct. 1990.

[13] T. Fuja, C. Heegard, and R. M. F. Goodman, "Linear sum codes for random access memories," *IEEE Trans. Comput.*, vol. C-37, pp. 1030–1042, Sept. 1988.

[14] L. Levine and W. Meyers, "Semiconductor memory reliability with error-detecting and correcting code," *Comput.*, vol. 9, pp. 43–50, Oct. 1976.

[15] W. F. Mikhail, R. W. Bartoldus, and R. A. Rutledge, "The reliability of memory with single-error correction," *IEEE Trans. Comput.*, vol. C-31, pp. 560–564, June 1982.

[16] R. A. Rutledge, "Models for the reliability of memory with ECC," in *Proc. 1985 IEEE Reliability, Maint. Symp.*, pp. 57–62, 1985.

[17] H. Vinck and K. Post, "On the influence of coding on the mean time to failure for degrading memories with defects," *IEEE Trans. Inform. Theory*, vol. 35, no. 4, pp. 902–906, July 1989.

[18] R. M. F. Goodman and R. J. McEliece, "Lifetime analyses of error-control coded semiconductor RAM systems," *IEE Proc.*, vol. 129E, pp. 81–85, May 1982.

[19] ___, "Hamming codes, computer memories, and the birthday surprise," in *Proc. 20th Allerton Conf. Commun., Contr., Comput.*, 1982.

[20] M. Blaum, "Error-correcting codes for computer memories," Ph.D. thesis, California Inst. of Technol., Pasadena, CA, 1985.

[21] M. Blaum, R. M. F. Goodman, and R. J. McEliece, "The reliability of single-error protected computer memories," *IEEE Trans. Comput.*, vol. C-37, pp. 114–119, Jan. 1988.

[22] R. Krishnamoorthy and C. Heegard, "Reliability and yield: Error control in semiconductor RAM's," *IEEE Trans. Inform. Theory*, preprint, School of Elect. Eng., Cornell Univ., Ithaca, NY, May 1990.