M. R. Libson
H. E. Harvey

A General-Purpose Memory Reliability Simulator

With rapid advances in computer memory capacity and performance, coupled with corresponding increases in the expense of field service calls, memory reliability and optimal maintenance strategies have become more and more important in terms of customer satisfaction and field service cost. At the same time, significant improvements in error correction and recovery over recent years have made the prediction of uncorrectable error and field service frequency much more difficult. This paper describes a Monte Carlo simulator which can predict uncorrectable error rates and field-replaceable-unit replacement rates for a wide range of memory architectures and under a variety of maintenance strategies. The model provides valuable information for performing sensitivity studies of intrinsic failure rates for memory components, for performing tradeoff studies of alternative storage module and card organizations, for evaluating system functions, and for establishing optimum maintenance strategies.

Introduction

The use of analytical techniques for determining the reliability of semiconductor memories with error correction capability is generally impractical if one wishes to consider a wide range of memory architectures and failure modes and to take into account the effects of a variety of maintenance strategies. The main reason for this is that simplifying assumptions must usually be made and these can often result in misleading conclusions. An alternative approach to the problem is to employ simulation techniques. This introduces the additional challenges of designing a simulator capable of modeling today's large computer memory capacities (16+ megabytes) in a reasonable memory space and of simulating a sufficient number of systems (system life is usually in the range of 60 000 to 100 000 hours) to achieve statistical credibility in a moderate amount of CPU time.

Simulation is readily adaptable to mimicking computer memory throughout its life in actual field use. Component failures and their locations can be generated by Monte Carlo techniques. With appropriate bookkeeping, the repair of *field-replaceable units* (FRUs) can then be initiated by such triggers as time (periodic maintenance), the number of faulty bits in the memory (preventive maintenance), or the detection of an *uncorrectable error* (UE). Reliability measures such as UE rates, FRU replacement rates, and *mean time between fails* (MTBF) can then be statistically determined.

This paper describes the Burlington Memory Reliability Simulator (BMRS), a general-purpose Monte Carlo memory reliability simulator which has been used extensively at IBM to model many of the memories used in various IBM computers as well as many proposed memory architectures. Memory reliability models that existed prior to BMRS contained numerous restrictions (e.g., rigid architecture, fixed failure modes, constant failure rates, and limited, if any, maintenance strategies) [1–9]. The BMRS program has resolved these deficiencies by adopting a Monte Carlo approach in conjunction with some unique methods for generating component times-to-fail, UE detection, and bookkeeping during the simulation in order to contain CPU time and memory space requirements. Although the required space and run times are a function of many variables, a BMRS run modeling 5000 16-megabyte systems with a 100 000-hour system life typically runs in less than one megabyte and five CPU minutes (IBM 3081). The accuracy of the results of BMRS obviously follows the laws of large numbers, so that accurate UE rates depend on the occurrence of "enough" UEs, and accurate FRU replacement rates depend on the occurrence of "enough" FRU replacements, and so on. Determination of how many samples are needed to produce "enough" of the desired parameter cannot be made until after the simulation has been run, unfortunately. However, past experience with BMRS has shown that, with today's memory architectures and failure

rates, 5000 samples is almost always more than sufficient to produce stable and consistent results without consuming a great deal of CPU time.

The next section of the paper describes the capabilities of the simulator in terms of memory organizations, failure modes, failure rates, and maintenance strategies. The modeling of *n*-bit *error-correcting code* (ECC), initial defects, and alpha-particle failures is also discussed in this section. The section following that discusses the methods and techniques used in the simulation. This involves some of the bookkeeping and data-management techniques, the Monte Carlo methods, and the action sequence of the simulation. The final section provides a summary of the major features of the simulator and its applications. Definitions and details of the various maintenance triggers and actions provided for by the program are included in an appendix.

Model capabilities

• *Memory architecture*

In order to achieve general-purpose simulation capability, the computer memory architecture must be defined to the program in such a way that a wide range of memory organizations can be described. A modified version of the memory model described by Mikhail, Bartoldus, and Rutledge [1] is the vehicle used to achieve generality. The memory is assumed to consist of four functional levels: the memory itself at the highest level, then the card level which consists of the FRUs (by definition), followed by the chip level [10], and finally by the cell level. Each level is defined as a rectangular set of the next lower level. For example,

$$\text{MEMORY} = X_1 \times Y_1 \text{ CARD}$$

defines a memory consisting of X_1 rows of cards (each row of cards is one *basic storage module*, or BSM, by definition) by Y_1 columns of cards (or FRUs). Each card column is assumed to map onto one or more bit fields. The card level is defined by

$$\text{CARD} = F \times X_2 \times Y_2 \text{ CHIP},$$

where F is the number of bit fields each card maps onto and X_2 and Y_2 are the number of rows and columns, respectively, of chips per bit field per card. Hence, the total number of bits per ECC word in the memory is

$$\textit{No. of bits/ECC word} = Y_1 \times F,$$

and the total number of chips in the memory is

$$\textit{No. of chips} = X_1 \times Y_1 \times F \times X_2 \times Y_2.$$

Finally, the chip level is defined by

$$\text{CHIP} = X_3 \times Y_3 \text{ CELL},$$

where X_3 and Y_3 are the number of rows and columns, respectively, of cells per chip. **Figure 1** illustrates a typical sample of a 72-bit-per-ECC-word memory and the relationships among the four levels.

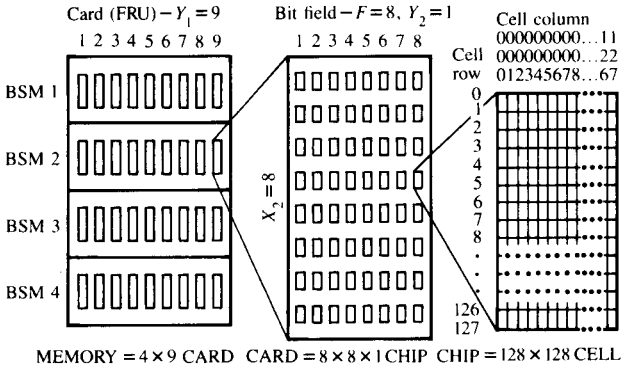

Figure 1 Example of a 4-megabyte, 72-bit-per-ECC-word memory comprised of four 1-megabyte BSMs. Each BSM is comprised of nine cards, each of which contains 64 16K-bit chips.

In order to accommodate memory architectures which are not quite as uniform as the one just described, multiple types of cards and/or chips can be defined by specifying multiple component names with their respective dimensions followed by a generic name for the level with the total dimensions of the level. If the memory architecture described in Fig. 1, for example, consists of two different types of cards and one of the card types consists of two different types of chips, with the total number of cards and chips equal to that in the example, the definition of such a memory might be specified as follows:

MEMORY = 4 × 5 CARDA, 4 × 4 CARDB, 4 × 9 CARD;

CARDA = 8 × 6 × 1 CHIPA, 8 × 2 × 1 CHIPB,
8 × 8 × 1 CHIP;

CARDB = 8 × 8 × 1 CHIPC;

CHIPA = 128 × 128 CELLA;

CHIPB = 128 × 128 CELLB;

CHIPC = 128 × 128 CELLC;.

Note that CARD and CHIP do not require definition, because they have been defined as the generic names for their respective levels, and that their dimensions are the total number of rows and columns on those levels.

Defining memories in such a manner permits one to simulate not only memories with more than one card organization, but also memories which contain chips from more than one manufacturer with correspondingly different intrinsic failure rates.

While the method of defining a memory organization as described may not suffice for every computer memory ever

designed, certainly the vast majority of memory architectures can be described within such a framework, and thus the goal of being widely applicable in terms of memory organizations is satisfied.

• *Failure modes and rates*

The failure modes associated with a computer memory play a critical role in determining the reliability of that memory. Furthermore, the failure modes are a function of the technology, the manufacturing process, and the device design for the memory chip, and thus may vary widely [2]. It is important, therefore, to model as many failure modes as possible for reliability studies. However, most memory reliability models restrict the permissible failure modes in order to simplify the mathematics involved. The restrictions range from equating every fail with an entire chip failing (under the assumption that chip failures are the dominant mode [4], or to get a worst-case estimate of reliability [3, 6]), to allowing rows and columns of cells and chips to fail as well as single cells and chips [1, 2, 7]. In actuality, there is evidence that partial-chip failures rather than whole-chip failures are the dominant failure mode for most chips and that failure modes can include double cells, double word and bit lines, and chip sections [2, 11]. The ability to model chip-section failure modes increases in importance for memories composed of chips with modular architectures (independent islands). Furthermore, as chip densities increase, it is believed that alpha particles may affect several adjacent cells as opposed to single cells.

Any rectangular subset of any of the memory architecture levels can be defined as a failure mode to BMRS. The only restriction imposed is that the dimensions of the failure mode be evenly divisible into the memory architecture in both the X and Y directions. Using the architecture described in the previous section, we can define failure modes as follows:

Memory architecture	*Possible failure modes*
MEMORY = 4 × 9 CARD	CELLCKTY = 1 × 1 CELL
CARD = 8 × 8 × 1 CHIP	ISLAND = 64 × 64 CELL
CHIP = 128 × 128 CELL	SUPPORT = 8 × 8 × 1 CHIP

In recent years, a number of articles have been published concerning alpha-particle-induced errors and their impact on system reliability [12–15]. The modeling of these errors is important not only in studying their impact, but in studying potential solutions as well. BMRS allows for modeling of transient failures in general by assuming that any failure mode whose first character is a "?" is transient. Thus, the same freedom for hard failure mode definition is available for transient failure modes. A transient failure is assumed to disappear immediately after occurring unless a retention time is specified, in which case it remains active until the specified time has elapsed. During the time that intermittent failures are active, they may align with other failures to cause "soft" UEs.

The effect of nonconstant component failure rates, and, in particular, the phenomenon known as "infant mortality," on system reliability can be dramatic [1, 16, 17]. Therefore, it is important that the model being used to evaluate system reliability allows for nonconstant component failure rates; most existing models do not do this [3–9]. Piecewise-linear failure rates or shape and scale parameters for the Weibull distribution hazard function [18] may be provided as input for any failure mode defined to BMRS.

The process known as "vintage learning" (maturing and improvement of the fabrication process) may have an important positive impact on system reliability and should be modeled in some cases [19]. The vintage learning process is simulated by allowing multiple sets of failure rates for the same component. Each set is associated with a *power-on-hours* (POH) value and becomes effective only for those components which reside on FRUs replaced after the specified POH. Thus, for example, if a chip has one set of failure rates associated with 0 POH and another set of failure rates associated with 60 000 POH, the second set takes effect only for chips which are located on FRUs that have been repaired after 60 000 POH.

• *Maintenance strategies*

It has been shown that storage system reliability may vary significantly with the maintenance strategy employed and that maintenance strategies play a critical role in improving memory system reliability [19, 20]. However, most reliability models stop short of considering the effect of maintenance on reliability. Those models that do take this into account consider only maintenance strategies which completely clean the memory at maintenance time, i.e., one-card or renewal systems which replace the entire memory whenever a repair is made or those in which all components have constant failure rates and the strategy is to replace every failed component at repair time. The reason is that an analytical approach for reliability analysis of memory systems with ECC under a variety of maintenance strategies which do not completely clean or renew the memory is extremely difficult, if not impossible [15, 17]. Both strategies that completely clean the memory make the repaired system identical to the original system or good as new; but neither strategy, with the exception of the one-card system for small machines, is very realistic or practical for memories with ECC. The effects of maintenance strategy on system reliability can be rather surprising at times (e.g., periodic maintenance can, in certain cases, make system reliability worse instead of better [11]). Therefore, it is crucial to be able to model a wide variety of maintenance strategies in order to determine the optimal strategy for any given memory and to estimate field service costs and field stocking plans. BMRS provides this capability with four types of maintenance triggers, viz., *scheduled maintenance* (SM), *threshold* or *deferred maintenance* (DM), *hard uncorrectable error* (HUE), and *soft uncorrectable error* (SUE). (Definitions and

details of the various maintenance triggers and actions are provided in the Appendix.) Furthermore, there is provision for *conditional maintenance* to be taken only if the primary maintenance did not alleviate the condition that triggered the maintenance originally. Each trigger can initiate any of a wide range of maintenance actions. **Table 1** presents an overview of the available combinations of maintenance actions and triggers.

• *Additional capabilities*

In the past, it has been argued that *double error correction/triple error detection* (DEC/TED) requires too many check bits and complicated decoding circuitry and that it degrades memory performance too much to be economically or practically feasible [15, 17]. However, with the growing complexity of memory chips and their increasing sensitivity to defects and radiation, coupled with increases in chip density and main memory size, DEC/TED may be an option worthy of a second look for some systems as a potential solution to the corresponding reliability problems. The error-correction capability of the system being modeled is an input parameter for BMRS, and thus the program can model systems with no error correction, one-bit error correction, two-bit error correction, and, in general, n-bit error correction.

With the advent of error correction coupled with the increases in density, sensitivity to error, and complexity of chips mentioned previously, the economic practicality of shipping memory with existing faults (all correctable by ECC, of course) is becoming worthy of study in terms of cost savings, which can then be passed along to customers, due to increased manufacturing yield, alleviated parts-supply problems and reduced rework load. These savings must be weighed against the potential problems of higher UE rates and FRU replacement rates. BMRS provides for studies involving initial defects by allowing the user to specify any number of defects for any defined failure mode at time 0, provided the defects do not cause an uncorrectable error.

Because alpha-particle failure rates can be one or two orders of magnitude higher than basic intrinsic failure rates [13–15], for some memories it may be impractical to simulate alpha-particle fails along with the hard fails because of memory space and CPU time limitations. For these cases, an analytical calculation for the alpha-particle-induced UE rate for one-bit ECC machines is provided in the model by the following equation:

$$SUE(t) = \alpha \cdot (W - 1) \cdot A(t), \qquad \%/\text{kPOH},$$

where α is the alpha failure rate (%/kPOH/bit), W is the number of bits per ECC word, and $A(t)$ is the number of hard failed bits in the system at time t [21]. The equation simply states that the alpha-particle-induced UE rate is equal to the probability that an alpha fail occurs at time t in one of the $W - 1$ good bits of a computer word of length W which already has one failed bit in it. $A(t)$ is known from the simulation of the hard fails. The equation assumes that no maintenance action results from soft UEs, that each alpha fail affects only one bit, and that only hard-soft UEs need be considered, i.e., the retention time for alpha fails is zero.

Table 1 Table depicting valid maintenance actions which may be specified with maintenance triggers. A maximum of four actions may be specified with each trigger.

Maintenance action	*Maintenance trigger*				
	SM	*DM*	*HUE*	*SUE*	*Conditional*
Clean	Y	Y	Y	Y	Y
Worst	Y	Y	Y	Y	Y
Threshold	Y	Y	Y	Y	Y
Spare	N	N	Y	Y	Y
Spares	Y	Y	Y	Y	Y
Swap	N	N	Y	Y	N
Deallocation	N	N	Y	N	N
Conditional	N	N	Y	Y	–

Model methodology

The BMRS program has been designed to minimize both CPU time and region size. The memory architecture is defined once for any new memory and is stored in a permanent data base. BMRS also dynamically builds the simulator source code by including only the source code required to model the maintenance strategies and ECC level specified by the user, thus eliminating a great deal of decision making during the simulation.

The simulator consists of five basic steps (**Figure 2**), viz., generation of component fail times and locations; performance of pending SM and/or DM maintenance; determination of the impact of component failures on the memory; performance of any required UE maintenance strategy; and the recording of each event observed throughout the life of the memory. The three main ingredients of the simulator which minimize CPU time and memory space are the techniques of generating times-to-fail, the detection of UEs, and the bookkeeping methods employed to track the status of the memory at any given time.

• *Component failure generation*

Rather than generate a time-to-fail for each component present in the system, which could easily number in the millions just for cells, and then sort and select only those failures which occur prior to the end of the system life, a less well-known but far more efficient and convenient method for component failure generation is used. A technique for generating the first r-order statistics ($X_1 < \cdots < X_r$) of a sample of size n from a population with the distribution function $F(x) = P[X \leq x]$ is

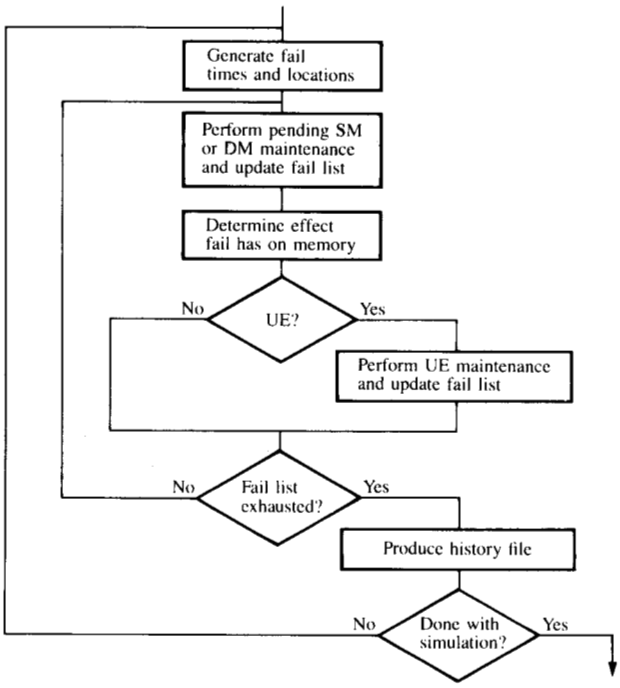

Figure 2 BMRS simulation flow.

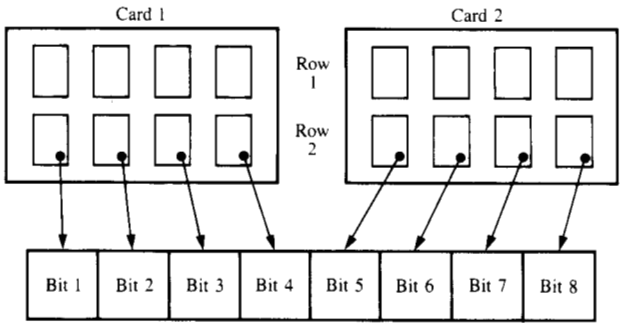

Figure 3 Example of the mapping of an eight-bit word from eight cells in the same relative location on eight different chips in the same relative location on two cards to form an eight-bit ECC word.

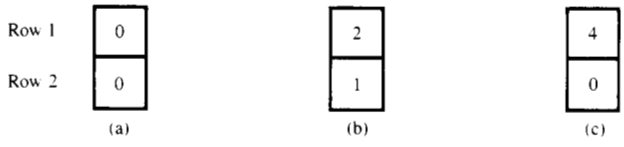

Figure 4 Sample System Status tables: (a) at start of simulation; (b) after processing of first three component failures from Table 3; and (c) after performing maintenance resulting from UE occurring while processing fourth failure from Table 4.

described and proven in [22]. BMRS uses this method to calculate ordered times-to-fail for each component type for all failures occurring within the system life. A sufficient number of ordered times-to-fail is computed by the equation

$$T_i = F^{-1}\left[1 - \prod_{j=1}^{i} U_j^{\frac{1}{n-j+1}}\right]_{i=1,\cdots,r},$$

where

T_i is an ordered time-to-fail,

$F(t) = P(T \le t)$ is the cumulative distribution function of time to fail over the interval $0 \le t < \infty$ for a given component type,

r is the desired number of ordered times-to-fail to be generated (should be large enough to ensure that T_r is greater than system life),

U_j is a member of a sequence $(U_1, \cdots, U_r)$ of independent uniformly distributed random variables on the interval (0, 1),

$F^{-1}(u)$ is the inverse function of F defined by $F^{-1}(u) = [t : F(t) = u]$, $0 < u < 1$,

and

n is the total number of components available.

The physical location of each component failure is determined randomly, such that no two failures of the same type will be at the same location. The lists of fail times and locations of each component type are merged to form a single fail list, as illustrated in **Table 2.**

• *UE detection*

Bits in the ECC word are mapped from cells located on chips from each bit field. The cells are located in the same relative position on each chip and the chips are located in the same relative position in each bit field. For example, consider a BSM consisting of two cards, each card consisting of four bit fields and each bit field consisting of two rows of chips (**Figure 3**). Eight cells (one cell per bit) are selected for mapping into the ECC word. The eight cells are located in the same relative position on eight different chips, each chip in the same relative bit field position on the cards. If two cells fail on different chips in the same row, an ECC word will have a two-bit error only if the cells are located in the same relative position on the chips. It is easily seen that if any row has two chip failures, there are many words (equal to the number of cells per chip) having two-bit errors.

BMRS builds a *system status table* (SST) [**Figure 4(a)**] which represents the bit field organization of the memory architecture for UE detection. A nonzero entry in any SST element indicates that there is at least one component failure in the row of chips the element represents. More precisely, it indicates the index of the component failure in the fail list. If the SST element representing the current failure already contains a nonzero entry, the possibility of an uncorrectable error exists for a single-bit ECC memory. Thus, the SST table is used to quickly determine the presence of a UE by avoiding a search through the entire fail list. A UE will occur only if the two components affect the same ECC memory word in different bits (i.e., identical cell locations on different chips).

Table 2 Sample fail list as generated prior to start of simulation and at 2500 POH (just prior to processing fail Index 3).

Index	*Time*	*Component*	*Physical location*							*Chain*	*Status*
			BSM	*Card*	*Bit field*	*Chip*		*Cell*			
						Row	*Col*	*Row*	*Col*		
1	400	BIT LINE	1	1	4	2	1	—	28	0	—
2	1000	CHIP	1	1	2	1	1	—	—	0	—
→3	2500	BIT LINE	1	1	4	2	1	—	32	0	—
4	6600	CELL	1	2	3	1	1	32	64	0	—
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.

Table 3 Sample fail list after processing of Index 3 and just prior to processing of Index 4. Note that Index 3 has been chained to Index 1 by the chain column.

Index	*Time*	*Component*	*Physical location*							*Chain*	*Status*
			BSM	*Card*	*Bit field*	*Chip*		*Cell*			
						Row	*Col*	*Row*	*Col*		
1	400	BIT LINE	1	1	4	2	1	—	28	3	—
2	1000	CHIP	1	1	2	1	1	—	—	0	—
3	2500	BIT LINE	1	1	4	2	1	—	32	0	—
→4	6600	CELL	1	2	3	1	1	32	64	0	—
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.

Table 4 Sample fail list after processing of UE at 6600 POH (Index 4). Note change in status column for component failures which have been removed from the system because of FRU replacement.

Index	*Time*	*Component*	*Physical location*							*Chain*	*Status*
			BSM	*Card*	*Bit field*	*Chip*		*Cell*			
						Row	*Col*	*Row*	*Col*		
1	400	BIT LINE	1	1	4	2	1	—	28	3	1
2	1000	CHIP	1	1	2	1	1	—	—	0	1
3	2500	BIT LINE	1	1	4	2	1	—	32	0	1
4	6600	CELL	1	2	3	1	1	32	64	0	—
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.

Referring to **Figure 4(b)**, consider a single-bit ECC memory with the fail list shown in Table 2. The fail currently being processed is the BIT LINE at Index 3. Because the BIT LINE failure occurred on a chip in the second row and the SST entry is nonzero, the possibility of a UE exists. However, since no two cells align (they affect different cell columns), a UE does not occur. Failures represented by the same SST entry which do not result in uncorrectable errors are chained together in the fail list, as shown by Index 1 in **Table 3.** Only the failure whose index is in the SST table, and those failures which are chained to it, need be checked for UE detection, since these are the only failures which could possibly align

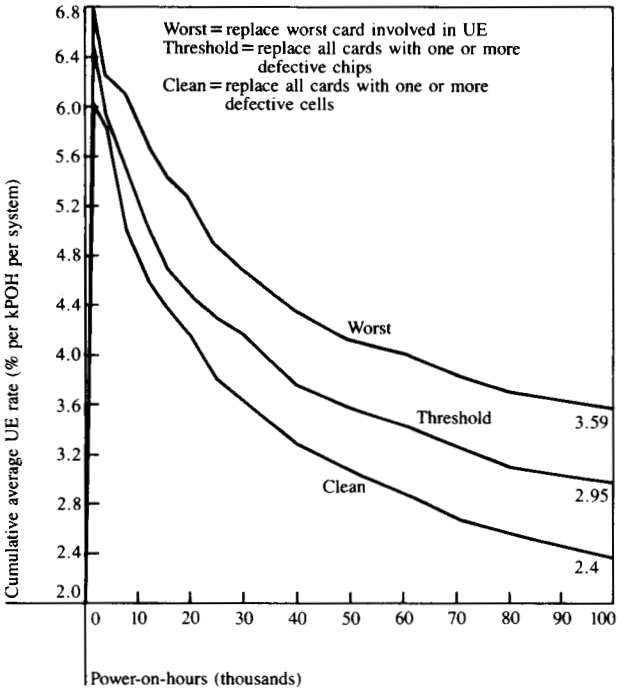

Figure 5 Uncorrectable Error rate results from three simulations with different hard uncorrectable error maintenance strategies for each. The memory simulated was 16 megabytes consisting of four 4-megabyte BSMs. Each BSM consisted of nine cards and each card contained eight rows and eight columns of 64K-bit chips (total of 64 chips per card).

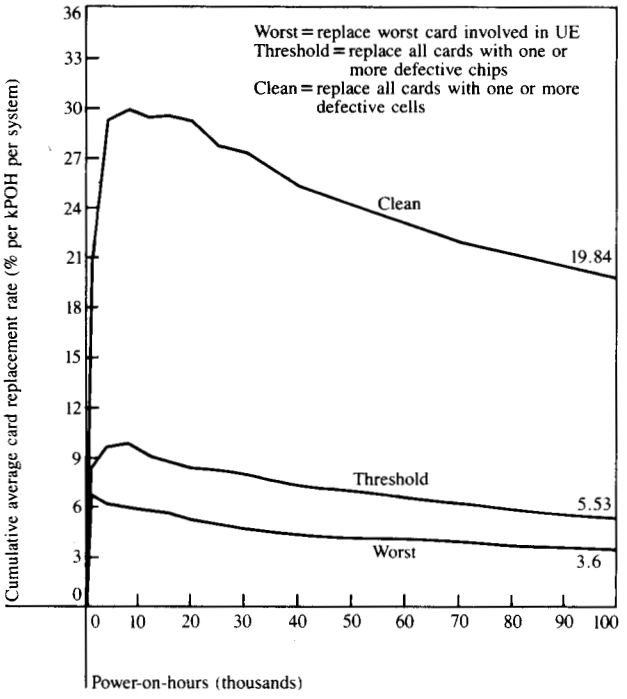

Figure 6 Card replacement rates for the simulations described in Fig. 5.

with the current fail. Many searches of the entire fail list are thus avoided, resulting in a very significant improvement in CPU usage.

Now consider Index 4, Table 3. The CELL failure aligns with the CHIP failure (Index 2), resulting in a UE. Assuming a UE maintenance of WORST (remove the FRU with the greatest number of defective cells), the card containing the chip fail (Index 2) would be removed and the "Status" flag would be set so that it no longer has any effect on the memory. Notice that the status flags for Indices 1 and 3 have also been set since these component failures reside on the same card as the chip failure (**Table 4**). The SST is updated to reflect the removed component failures [**Figure 4(c)**].

• *Bookkeeping*

Simulation commences with the first fail in the fail list. SM or DM maintenance is performed, if required. Components which reside on those FRUs replaced at maintenance time are flagged in the fail list as "removed from memory." "Removed" components no longer have any effect on the memory. Components which reside on replaced FRUs but which have not yet been processed ("detected") are deleted from the fail list. New failures are generated for components on the new FRUs.

Each event (e.g., UE, memory maintenance, etc.) is recorded in the fail list as it occurs during simulation. At the completion of a memory simulation, the fail list contains all events observed during the life of the memory. The fail list is saved in a temporary data base (the history file) for later use in report generation.

The entire process, from component failure generation to storing the fail list in the history file, is repeated for the requested sample size. BMRS retrieves the history file upon completion of all simulations and calculates the requested statistics. These can include reports on the number of failures of each component type, the distribution of component alignments causing UE, the number and rate of UEs per system, the number and rate of replaced FRUs per system, the average number of bad cells in the system at any time, and mean-time-between-fails, as well as many other reports which can be obtained at the user's request. The burden of statistical calculations is thus removed from the simulation process and placed in the report-generation process.

Summary and applications

This paper has described a Monte Carlo simulation program which predicts memory reliability in terms of uncorrectable error rates, field-replaceable-unit repair rates, and mean-time-between-fails. Variables affecting memory reliability (many of which must be discounted or simplified by analytical models), such as memory architecture, failure modes, nonconstant failure rates, vintage learning, maintenance actions, alpha

particles, etc., are all accounted for in the simulation. The model is flexible enough to simulate most memory architectures and failure modes as well as a wide range of maintenance strategies. The simulator employs efficient bookkeeping and data-management techniques so that it can model a large number of high-capacity memories in a minimum amount of memory space and CPU time. Work is currently in progress to add additional error recovery and dispersal capability to the program in the form of fault alignment exclusion [23] and a general sparing mechanism.

The simulation results have a wide range of applications. The program is used to provide RAS parameters (UE rates and FRU replacement rates) for estimating field service costs, field stocking requirements, and customer satisfaction. It is used to generate reliability specifications and reliability objectives for memory components and products. It has applications during the design cycle for evaluating and performing tradeoff studies of proposed basic storage module (BSM) and card architectures.

Optimization and tradeoff studies of various maintenance strategies (**Figures 5** and **6**) to minimize field service costs can be performed with BMRS, as well as reliability sensitivity studies of component failure modes and rates (**Figures 7** and **8**).

Simulation results can be used as input for evaluating the impact of soft errors and the effectiveness of system functions such as ECC (**Figures 9** and **10**), sparing, fault alignment exclusion, and page deallocation. Economic feasibility studies for shipping memories with initial faults can also be performed. In short, BMRS is a valuable and powerful modeling tool for the wide range of organizations which must consider memory reliability to perform their functions.

Appendix: Maintenance definitions

- *Maintenance triggers*

Scheduled Maintenance (SM)—performed at a user-specified *power-on-hours* (POH).

Threshold or Deferred Maintenance (DM)—performed when the system has reached or exceeded a user-specified number of bad bits (threshold) on any FRU or BSM. The maintenance can be deferred to a specified POH delay after the system has reached the threshold.

Hard Uncorrectable Error maintenance (HUE)—performed when the number of errors in any ECC word exceeds the ECC capability and all the errors are hard.

Soft Uncorrectable Error maintenance (SUE)—performed when the number of errors in any ECC word exceeds the ECC capability and at least one of the errors is intermittent or transient.

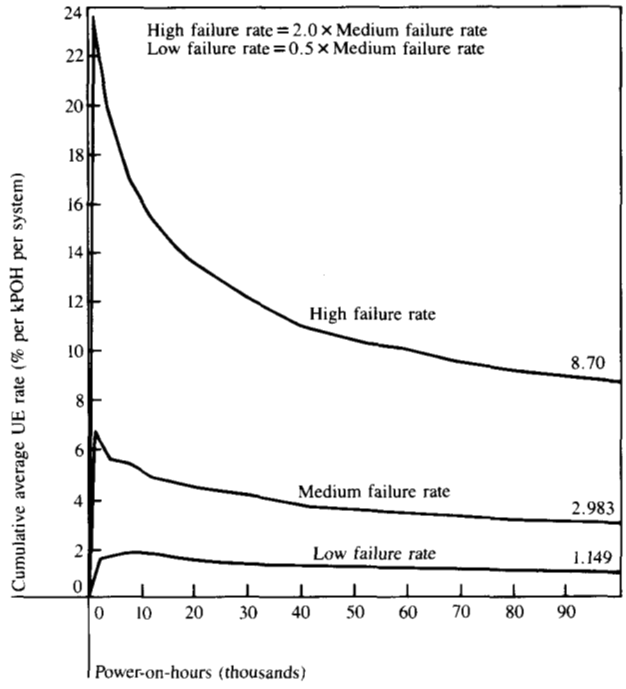

Figure 7 Uncorrectable Error rate results from three simulations with different array component failure rates for each. The memory simulated is the memory described in Fig. 5 with a threshold hard uncorrectable error maintenance strategy.

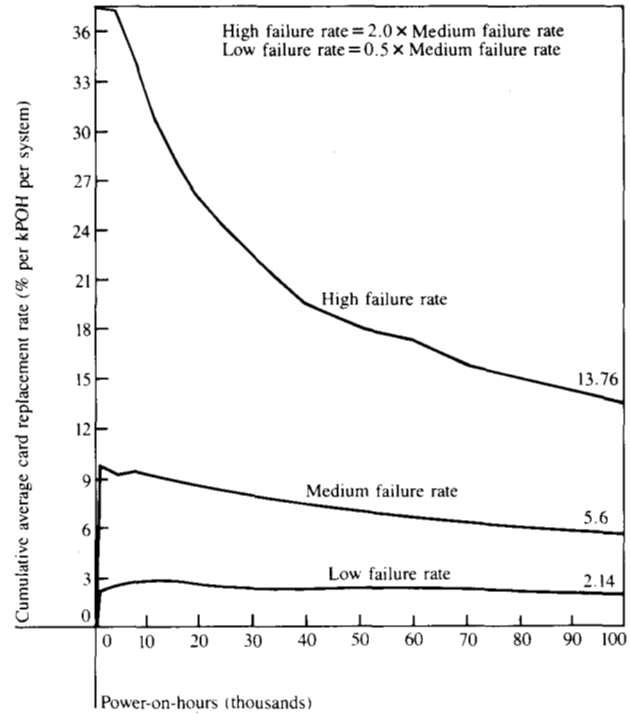

Figure 8 Card replacement rates for the simulations described in Fig. 7.

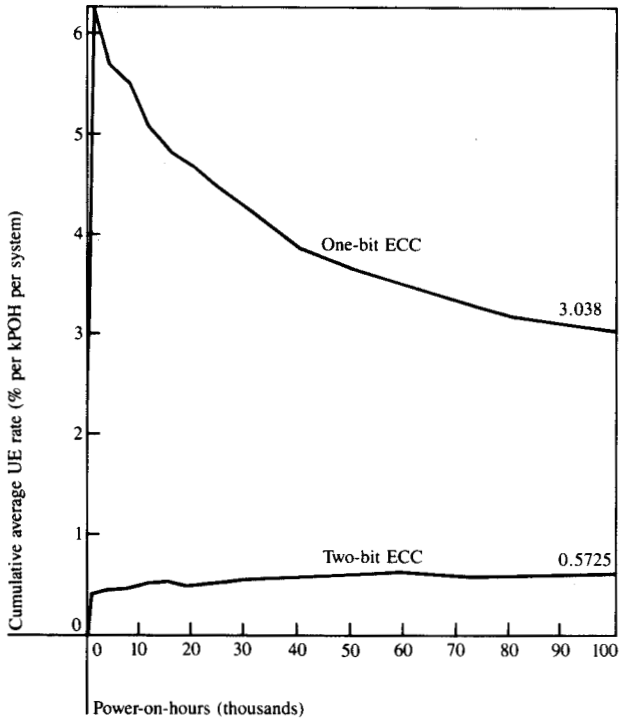

Figure 9 Uncorrectable Error rate results from simulations with one-bit ECC and two-bit ECC. The memory simulated is the memory described in Fig. 5 with a threshold hard uncorrectable error maintenance strategy.

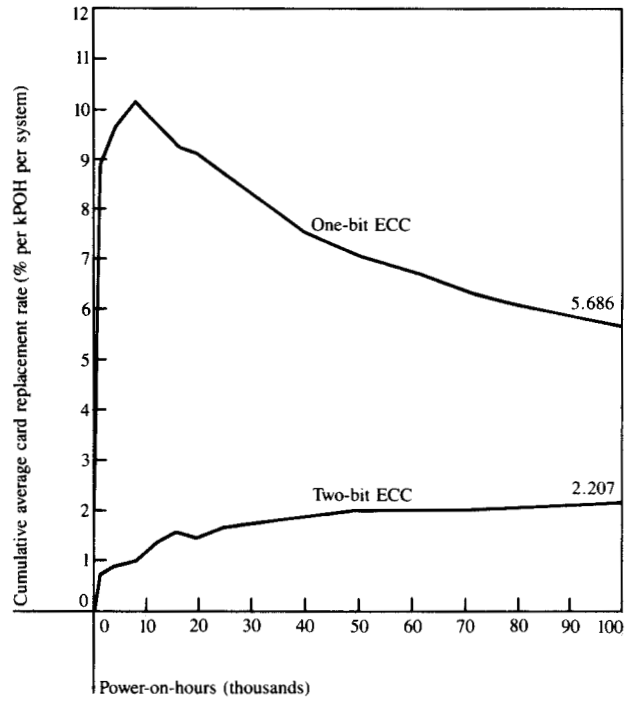

Figure 10 Card replacement rates for the simulations described in Fig. 9.

- *Maintenance actions*

Worst—Causes the FRU with the greatest number of bad cells to be replaced.

Clean—All cards containing one or more bad cells are replaced.

Threshold—All cards containing a user-specified number or more of bad cells are replaced.

Spare—One redundant chip for each row of chips specified in the memory architecture is available to replace the chip containing the larger fault that caused an uncorrectable error. This action is valid only for the UE maintenance triggers and for memories with one-bit ECC.

Spares—Similar to Spare except that all available spares are used at system maintenance. Unlike Spare, Spares is a valid SM and DM maintenance strategy as well as UE.

Swap—UE maintenance action causing one of the FRUs involved in the UE to be exchanged with a FRU in the corresponding position of another BSM to disperse the faults.

Deallocation—Hard UE maintenance action for one-bit ECC systems causing one of the two UE-causing faults (user-specified) and a user-specified number of units to be deallocated. With an appropriate specification for the fault to be deallocated and for the number of units to be deallocated, the deallocation action can be used to approximate page deallocation, fault alignment exclusion, and redundancy.

Conditional—Conditional maintenance actions are taken only when the previous action fails to correct the UE. Any of the previously described actions except Swap and Deallocation can be made conditional.

Up to four maintenance actions to be performed in sequence or conditionally can be specified with each maintenance trigger.

Discrete sets of POH can be specified with each maintenance action so that different maintenance strategies can be employed during various stages of the product life.

Acknowledgments

The authors wish to recognize D. Withers for his early participation in the development of the model, R. A. Rutledge for his invaluable technical contributions, R. J. Scaccia for his support and suggestions, and S. K. Kwon, L. J. Lareau, L. D. Burns, and J. LaBonte for their contributions to the development of the program.

References and note

1. W. F. Mikhail, R. W. Bartoldus, and R. A. Rutledge, "The Reliability of Memory with Single-Error Correction," *IEEE Trans. Computers* **C-31,** No. 6, 560–564 (1982).
2. S. A. Elkind and D. P. Siewiorek, "Reliability and Performance of Error-Correcting Memory and Register Arrays," *IEEE Trans. Computers* **C-29,** No. 10, 920–927 (1980).

3. A. L. Smith, "Hard and Soft Failures in Dynamic RAM Fault Tolerant Memories," *IEEE Trans. Reliability* **R-30,** No. 1, 58–60 (1981).
4. A. V. Ferris-Prabhu, "Improving Memory Reliability Through Error Correction," *Computer Design* **18,** No. 7, 137–144 (1979).
5. D. H. Redfield, E. G. Brown, and W. A. Sims, "Monolithic Main Memory—New Reliability and Serviceability Environment," *Proceedings, 9th Annual Reliability Physics Symposium*, Las Vegas, 1971, pp. 41–45.
6. L. Levine and W. Myers, "Semiconductor Memory Reliability with Error Detecting and Correcting Codes," *Computer* **9,** 43–50 (October 1976).
7. S. Q. Wang and K. Lovelace, "Improvement of Memory Reliability by Single-Bit-Error Correction," *Proceedings, 14th IEEE Computer Society International Conference*, San Francisco, 1977, pp. 175–178.
8. G. W. Cox and B. D. Carroll, "Reliability Modeling and Analysis of Fault-Tolerant Memories," *IEEE Trans. Reliability* **R-27,** No. 1, 49–54 (1978).
9. A. Costes, C. Landrault, and J. C. Laprie, "Reliability and Availability Models for Maintained Systems Featuring Hardware Failures and Design Faults," *IEEE Trans. Computers* **C-27,** No. 6, 49–54 (1978).
10. Multi-chip modules, if present, need not be considered for simulation purposes. However, if module statistics are desired, they can be obtained by defining the module architecture (number of rows and columns of chips) to the program.
11. C. H. Stapper, A. N. McLaren, and M. Dreckmann, "Yield Model for Productivity Optimization of VLSI Memory Chips with Redundancy and Partially Good Product," *IBM J. Res. Develop.* **24,** 398–409 (1980).
12. T. C. May and M. H. Woods, "A New Physical Mechanism for Soft Errors in Dynamic Memories," *Proceedings, 16th Annual Reliability Physics Symposium*, San Diego, 1978, pp. 33–40.
13. T. C. May and M. H. Woods, "Alpha-Particle-Induced Soft Errors in Dynamic Memories," *IEEE Trans. Electron Dev.* **ED-26,** No. 4, 2–9 (1979).
14. 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 Dev.* **ED-26,** No. 4, 10–15 (1979).
15. D. C. Bossen and M. Y. Hsiao, "A System Solution to the Memory Soft Error Problem," *IBM J. Res. Develop.* **24,** 390–397 (1980).
16. D. S. Peck, "New Concerns About Integrated Circuit Reliability," *Proceedings, 16th Annual Reliability Physics Symposium*, San Diego, 1978, pp. 1–6.
17. 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, 1979, pp. 257–272.
18. J. F. Lawless, *Statistical Models and Methods for Lifetime Data*, John Wiley & Sons, Inc., New York, 1982.
19. C. G. Peattie, J. D. Adams, S. L. Carrell, T. D. George, and M. H. Valek, "Elements of Semiconductor Device Reliability," *Proc. IEEE* **62,** No. 2, 149–162 (1974).
20. S. K. Kwon, "Reliability Sensitivity of LSI Memory with Single Error Correction," *From Electronics to Microelectronics*, W. A. Kaiser and W. E. Proebster, Eds., North-Holland Publishing Co., Amsterdam, 1980.
21. R. A. Rutledge, "BMRS-II Calculation of Alpha-Particle Induced UE Rates," unpublished internal memorandum, IBM Poughkeepsie Laboratory, March 1979.
22. R. A. Rutledge, "Monte Carlo Generation of Order Statistics," *Technical Report 22.1279,* IBM Poughkeepsie Laboratory, June 1971.
23. D. C. Bossen, C. L. Chen, and M. Y. Hsiao, "Fault Alignment Exclusion for Memory Using Address Permutation," *IBM J. Res. Develop.* **28,** 170–176 (1984, this issue).

Received June 30, 1983; revised September 21, 1983

Harold E. Harvey *IBM General Technology Division, P.O. Box A, Essex Junction, Vermont 05452.* Mr. Harvey is a staff programmer in test diagnostic and analysis systems. He joined IBM in 1965 at Essex Junction. He received a B.S. in mathematics from the University of Vermont, Burlington, in 1971, and is currently lead programmer involved in logic diagnostics for products under development in Burlington.

Marvin R. Libson *IBM General Technology Division, P.O. Box A, Essex Junction, Vermont 05452.* Mr. Libson is a member of the Advanced Mathematics and Engineering Analysis Department at the Essex Junction laboratory. He is currently engaged in the programming of new developments in the area of memory reliability. He joined IBM in 1978 and prior to his present assignment worked on an interactive chip design system. Mr. Libson received a B.S. from Brooklyn College, New York, in 1967 and an M.S. from Wichita State University, Kansas, in 1970, both in mathematics. From 1968 to 1971, he was involved in applications programming for Cessna Aircraft Company, and from 1971 to 1978, he was an employee of the Systems Development Corporation programming aerospace applications for NASA.