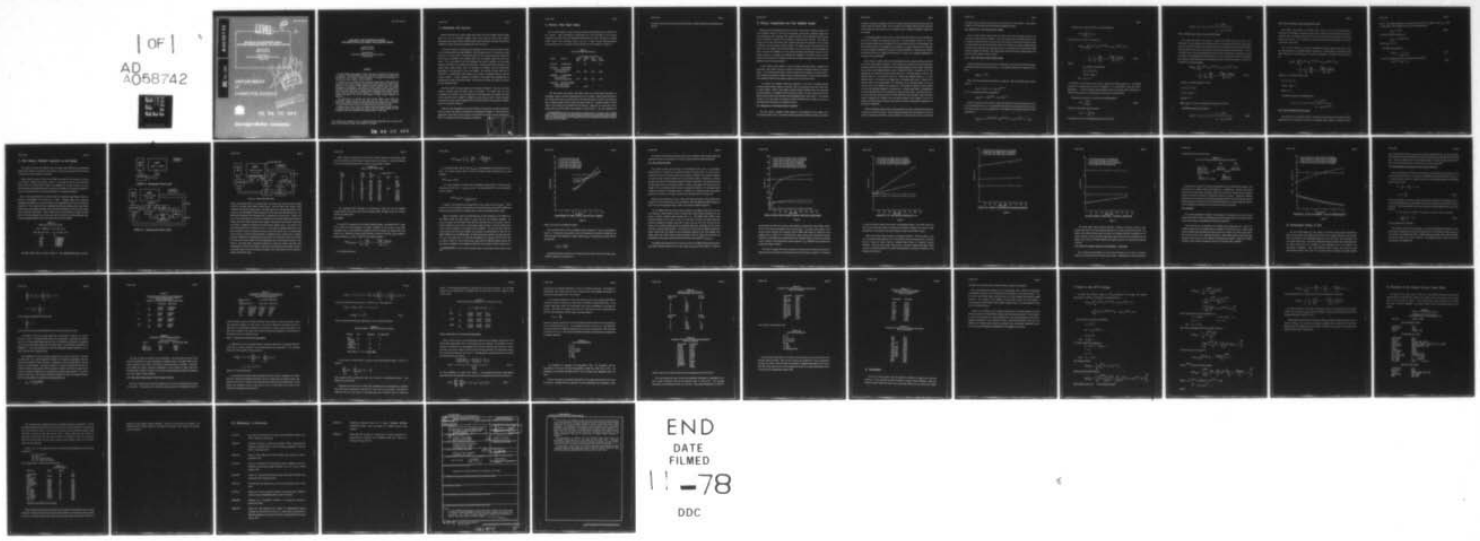
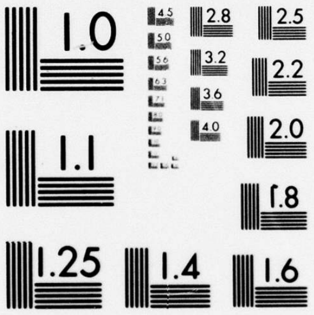
AD-A058 742 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/2
RELIABILITY AND PERFORMANCE MODELS FOR ERROR CORRECTING MEMORY --ETC(U)
MAY 78 S A ELKIND, D P SIEWIOREK N00014-77-C-0103
UNCLASSIFIED CMU-CS-78-118 NL

| OF |

AD
A058742



END
DATE
FILMED
11-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

CMU-CS-78-118

12

RELIABILITY AND PERFORMANCE MODELS FOR ERROR CORRECTING MEMORY AND REGISTER ARRAYS

Steven A. Elkind
Daniel P. Siewiorek

Departments of
Electrical Engineering and Computer Science

22 May 1978

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

CMU-CS-78-118

RELIABILITY AND PERFORMANCE MODELS FOR ERROR CORRECTING MEMORY AND REGISTER ARRAYS

Steven A. Elkind
Daniel P. Siewiorek

Departments of
Electrical Engineering and Computer Science
22 May 1978

Abstract

Current digital system design practices make heavy utilization of various types of memories. RAMS and ROMS are used in main memory, register files, caches, and microstores. As a result, it becomes important to recognize the implications of memory chip failure modes for system reliability.

A brief survey of available memory chip failure mode data is made and shows that partial chip failures are more prevalent than whole chip failures. Based on the findings of this survey, reliability models for memory systems with error coding techniques are developed. The effect of memory support circuitry on memory reliability, usually ignored in the development of analytical models, is included. It is shown that for wide ranges of memory system parameters and memory element failure rates the memory system reliability is dominated by the effect of the support electronics. The use of these models in design tradeoff decisions is explored.

The performance of systems with fault tolerant memory when there are correctable failures present, an area which has seen little work, is analyzed. Performance models for systems with fault tolerant main memory, as well as those with fault tolerant microstore, are developed and their properties explored.

Hamming code is one of the error corecting techniques considered. Block codes, commonly used for tape media but rarely if ever for RAM or ROM, are also considered and found competitive with Hamming codes in many cases.

This research was supported in part by Digital Equipment Corporation and in part by the office of Naval Research under contract N00014-77-C-0103.

1. Introduction and Overview

Memory elements are currently being used heavily in all areas of digital systems design. They see use as microstores, register files, caches, and of course, main memory. Improvement of memory element reliability, then, will have the effect of greatly improving the reliability of those systems being designed now and in the future.

A primary method of increasing memory reliability is the use of error correction codes, such as Hamming and block codes [Pete72]. These techniques result in a tolerance to single bit faults in a memory word. The tradeoff involved is one involving system cost, complexity, performance, servicability, and reliability (these last two together determine field repair costs). For an increase in cost and complexity, the memory reliability can be greatly enhanced with little or no decrease in performance. To help in the tradeoff decisions to be made during system design, tools for predicting the reliability and the performance degradation in the presence of errors are needed. The accuracy of these tools can be increased by examining the failure modes of the memory components used to build the memory systems. A study, summarized in Chapter 2, was made of the available data on semiconductor memory chip failure modes. The results of the study were used as the basis for the reliability models

The tools needed for design decisions are developed in Chapter 3. Two error correcting schemes, Hamming codes and block codes, are examined. The design tools are then formulated to be used for any size of single error correcting memory, and include the effects of the support circuitry needed to complete an entire memory system. Easily calculable formulae for memory system reliability, MTTF, and hazard function are presented in Chapter 3 and are analyzed in Chapter 4.

Models for the degradation of system performance in the presence of memory component failures in both main memory and microstore are developed in Chapter 5. The performance degradation when failures have occured in the memories of example systems, including the PDP-11, is analyzed in the final subsections of Chapter 5.

2. Memory Chip Failure Modes

Data on semiconductor memory chip failure modes during operating life is available from few sources. Most semiconductor manufacturers are more interested in the physical mechanisms of failure than in the functional characteristics of a failure. What data is available comes mostly from screening, burn-in, and to a lesser extent, high-temperature accelerated-life tests. A summary of some of the data we have collected is in Table 2-1.

TABLE 2-1
Chip Failure Mode Data Summary

		- - Percentage of All Failures - -			
Source	Devices	Whole Chip	Single Bit	Row/ Column	Not Known
[Texa7?]	4K MOS RAM	-	92	-	8
[Pasc75]	4K MOS RAM (burn-in & cell stress screening tests)	11.8	35.3	29.4	23.5
[Rick76]	varied PROMs (accel. life tests, using some guessing)	17.9	53.9	15.3	12.9
[Gear76]	8K MOS UV PROM (700k device hours in accel. life testing)	-	100.0	-	-

The data shows that memory chip failure modes are, unsurprisingly, dependent on technology, process, and device design and thus may vary widely. Failure mode distributions also change with time for a given device as the fabrication process matures[1]. Nevertheless, there is good evidence that the whole chip failure modes (i.e. complete inability to store and/or retrieve data) do not dominate for most devices. Single bit, row, and column failure modes seem to be the effect of the majority of device failures. This fact motivated the

[1] The TI data indicates that 92% of the failures observed were single bit failures. A conversation with a TI reliability engineer revealed that recent tests show only about half of the failures observed were single bit failures, reflecting pocess improvements since [Texa7?] was released. Although these tests were conducted for operating periods short relative to actual field operating life, the TI engineer felt that long term field data would still show a dominant percentage of partial array failures.

formulation of the error correcting code (ECC) memory models presented in the following two sections.

3. Memory Organizations and Their Reliability Models

Wang and Lovelace [Wang77] present a model for main memory reliability, based on the use of 4096 bit chips in a 16 bit per word memory system using a Hamming single error correcting/double error detecting code. Another model by Levine and Meyers [Levi76] uses charts and tables to determine the reliability of Hamming coded memories. The model is based on the whole chip failure mode. Neither model allows for the effect of the control reliability on the total memory system reliability. The models presented and examined in this paper cover any single error correction scheme for any size memory, and are developed in such a way that the reliability of all the control, correction, and interface circuitry for the memory element is included, thus modelling the reliabilty of the entire memory system. Further, a new formula is drived that can be used to efficiently calculate mean time to failure (MTTF) of any of the various models.

In this section three models for error-correcting code (ECC) memory reliability are presented. Each model is based on a different assumption of dominant memory chip failure mode. Two of them provide upper and lower bounds for the reliability of an ECC memory. For comparison we present a model for the non-redundant memory. All of the models assume that component failures in the memory support circuitry cannot be survived.

To develop the reliability models the properties of two error correcting schemes, Hamming codes and block codes, are examined. One of the measures to be used is mean time to failure (MTTF). MTTF is used widely in design trade-off decisions and in such business planning activities as availability and life cycle cost activities. The other measure will be the hazard function $z(t)$, which expresses the instantaneous failure rates at time t. The hazard function is not only easier to measure in practical situations but its shape can also say something about the shape of the reliability function.

3.1. Single Error Correcting Memory Properties

The ECC memory reliability models depend on the properties of the single error correcting schemes used. In examining the Hamming and block code ECC schemes, two types of memorv words are considered. The first is called a logical word and is the word that the system using the memory needs. Error correction in the memory itself is done for physical words, which are made up of one or more logical words in addition to whatever coding bits are needed.

For Hamming codes a b bit word has c coding bits (which may or may not include the extra bit for double error detection) added to it. The total number of bits is $k = (b+c)$. Several logical words may be combined into a larger physical word for error encoding, thus decreasing the total of the coding bits in the memory. If j logical words go into a physical word that includes e coding bits, the physical word size becomes $k = (bj+e)$, and the number of physical words in an x logical word memory is $w = (x/j)$. For a complete explanation of Hamming codes, see [Pete72].

Block codes are widely used for tape media-based memory systems, but have seen little or no use in other types of memories. In this scheme, each word has a parity bit appended (horizontal parity bit) and j words of b bits are grouped together to form a block. Each block has an extra word associated with it, each of whose (b+1) bits is the parity bit for the appropriate bit slice of the block (vertical parity bits). The total number of bits in the physical word is $k = (b+1)*(j+1)$, and for an x logical word memory there are $w = (x/j)$ physical words. In the case of a single error, a horizontal parity error is found, and a vertical parity word reconstructed. The intersection of the horizontal parity error and vertical parity error pinpoint the erroneous bit. This method of coding also allows double errors to be detected, although not recovered from. The block code suffers no degradation over the Hamming code in error detection due to the horizontal parity. Correction, however, is slower since the vertical parity of the whole block has to be calculated. Since correction occurs infrequently for transient errors this slow correction is not a penalty. Even in the presence of hard failures, the block code suffers very little in performance degradation as illustrated in Chapter 5.

Both the Hamming coded and block coded memories, then, have k bit physical words and w physical words in the memory. The only difference between the two schemes as far as the model is concerned is that these values are different. In each case, the memory can tolerate no more than one failure in the k bits of a given word in a w word memory. This common property is the one upon which the following development is based.

3.2. Single Error Correcting Memory Models

The first ECC memory model assumes that single memory bit cell failures dominate, and provides an upper bound on reliability. The second assumes that the dominant failure mode is complete functional failure of memory chips, and provides a lower bound. Between these two extremes lie row and column failures in the arrays internal to the chips, and distributions of whole chip, single cell, and row/column failures. For completeness a third model for ECC memory reliability is presented. This model assumes that the row (column) failure mode is the dominant failure mode for memory devices.

3.2.1. Single Bit Failure Mode (SBFM) Model

Single bit cell failures are assumed to be independent events, with each cell following the exponential failure law with failure rate λ_b. The reliability function for a single bit cell is then

$$R_b(t) = e^{-\lambda_b t}$$

Each k bit word can tolerate the failure of a single bit. Thus the reliability R_g of a given word is :

$$R_g(t) = R_b^{\,k} + k\ (1 - R_b)\ R_b^{\,(k-1)}$$

For a w word memory the array reliability is

$$R_{asb}(t) = (\ kR_b^{\,(k-1)} - (k-1)\ R_b^{\,k}\ )^{w}$$

Fault-free operation of the memory requires the selection, control, and decoding circuitry to be functioning correctly. It is assumed that these also follow exponential failure processes with failure rates λ_s, λ_k, and λ_d respectively. The reliability of the complete memory is then expressed as :

$$R_{msb}(t) = e^{-(\lambda_s+\lambda_k+\lambda_d)\,t}\ (ke^{-(k-1)\lambda_b t} - (k-1)\ e^{-k\lambda_b t}\ )^{w} \quad (3\text{-}1)$$

The mean time to failure (MTTF) is defined [Shoo68] by

$$MTTF = \int_0^\infty R(t)\, dt$$

The mean time to failure of the memory is :

$$MTTF_{sb} = \int_0^\infty e^{-\lambda_e t} \left(ke^{-(k-1)\lambda_b t} - (k-1)\, e^{-k\lambda_b t}\right)^w dt$$

$$= \frac{1}{\lambda_b}\left(\frac{1}{f_0} + \frac{kg_0}{f_0 f_1} + \ldots + \frac{k^w g_0 \ldots g_{(w-1)}}{f_0 \ldots f_w}\right) \qquad (3\text{-}2)$$

where

λ_b = memory bit failure rate,

$\lambda_e = \lambda_s + \lambda_k + \lambda_d$,

$f_i = wk + \lambda_e/\lambda_b - i$,

and $g_i = w - i$.

The MTTF of the memory array alone is obtained by setting $\lambda_e/\lambda_b = 0$. A detailed derivation of equation (3-2) is given in Appendix I. It is significant to note that MTTF calculations for ECC memories were extremely tedious (e.g. Monte Carlo simulations or numerical integrations were used) before the derivation of equation (3-2).

The hazard function z(t) of a system is defined [Shoo68] by

$$z(t) = \frac{f(t)}{R(t)} \qquad (3\text{-}3)$$

where f(t) is the failure density function

$$f(t) = -\frac{\partial}{\partial t} R(t) .$$

The hazard function for the SBFM model can be shown to be

$$z_{sb}(t) = \lambda_e + \lambda_b\, w\, k\, (k-1) \frac{(1 - e^{-\lambda_b t})}{(k - (k-1)\, e^{-\lambda_b t})} .$$

3.2.2. Whole Chip Failure Mode (WCFM) Model

The whole chip failure mode model is similar to the SBFM model with only a few differences. Since a single error correcting memory architecture with more than one bit per word per chip is not tolerant of whole chip failures, it is assumed that words are distributed over the chips in such a way that no word has more than one bit on the same chip. Thus for a memory with k bit words, implemented with d bit chips, the parameter w (number of words in memory) in the SBFM model is transformed to h = w/d. In effect the memory is organized into rows of k chips each, every row containing d words; h is then the number of such rows. The MTTF is then expressed by

$$MTTF_{wc} = \int_0^\infty e^{-\lambda_e t} \left(k e^{-(k-1)\lambda_c t} - (k-1)\, e^{-k\lambda_c t}\right)^h dt$$

$$= \frac{1}{\lambda_c} \left(\frac{1}{f_0} + \frac{k g_0}{f_0 f_1} + \ldots + \frac{k^h g_0 \cdots g_{(h-1)}}{f_0 \cdots f_h}\right) \qquad (3\text{-}4)$$

where λ_c = memory chip failure rate,

$\lambda_e = \lambda_s + \lambda_k + \lambda_d$,

$f_i = hk + \lambda_e/\lambda_c - i$,

and $g_i = h - i$.

When $\lambda_e/\lambda_c = 0$ this is an expression for memory array MTTF.

The WCFM model hazard function is

$$z_{wc}(t) = \lambda_e + \lambda_c\, h\, k\, (k-1) \frac{(1 - e^{-\lambda_c t})}{(k - (k-1)\, e^{-\lambda_c t})} . \qquad (3\text{-}5)$$

3.2.3. Row (Column) Failure Mode (RFM) Model

This model is also similar to the SBFM model. There is, as in the previous model, a restriction on the memory architecture, albeit not as stringent. Having more than one bit per word per chip is possible as long as there is no more than one bit per word per row (column) internal to the chip. If this condition is not met, this model should be replaced by the whole chip failure mode model.

For a w word memory of k bit words implemented with d bit memory chips havin q bits per row (column), w of the SBFM model is replaced by p = w*q/d, which is the number of one word wide sets of rows (columns) in the memory architecture. The MTTF is then expressed by

$$MTTF_r = \int_0^\infty e^{-\lambda_e t} (ke^{-(k-1)\lambda_r t} - (k-1)\, e^{-k\lambda_r t})^p \, dt$$

$$= \frac{1}{\lambda_r} \left(\frac{1}{f_0} + \frac{kg_0}{f_0 f_1} + \ldots + \frac{k^p g_0 \cdots g_{(p-1)}}{f_0 \cdots f_p} \right)$$

where λ_r = row (column) failure rate,

$\lambda_e = \lambda_s + \lambda_k + \lambda_d$,

$f_i = pk + \lambda_e/\lambda_r - i$,

and $g_i = p - i$.

The hazard function for the RFM model is

$$z_r\,(t) = \lambda_e + \lambda_r\, p\, k\, (k-1) \frac{(1 - e^{-\lambda_r t})}{(k - (k-1)\, e^{-\lambda_r t})} .$$

3.3. Non-Redundant Memory Model

The model for non-redundant memory is based on the assumptions that components have exponential failure processes and that any component failure results in complete memory failure. The control, selection, and storage array circuitry have failure rates λ_k, λ_s, and λ_a respectively. The reliability of the array is then expressed by

$$R_{anr} = e^{-\lambda_a t} \tag{3-6}$$

and that of the entire memory by

$$R_{mnr} = e^{-(\lambda_{enr}+\lambda_a)t} ,$$

where $\lambda_{enr} = \lambda_k + \lambda_s$.

The MTTF of the memory is

$$MTTF_{nr} = \frac{1}{\lambda_{enr} + \lambda_a} . \tag{3-7}$$

The non-redundant memory has the constant hazard function

$$z_{nr}(t) = \lambda_{enr} + \lambda_a . \tag{3-8}$$

4. ECC Memory Reliability Exploration via the Models

The single bit failure mode (SBFM), whole chip failure mode (WCFM), and nonredundant (NR) memory models are compared in this section. The measures used are MTTF, the hazard function z(t), and the reliability function R(t).

Where specific values for memory chip reliability are used, they are based on the failure rate range for 4096 bit chips found in Table 4-1. The ranges in Table 4-1 cover observed failure rates for state-of-the-art chips. The reliabilities of control circuitry for error correcting and nonredundant memories are derived from models for the memories depicted in Figure 1, assuming the use of standard SSI/MSI logic. The memories modeled in

Figure 1 are assumed to be "bare bones" memories of relatively simple design. Figure 1a depicts a nonredundant b bit memory of w words. Hamming single error correcting capabilities are added to it as shown in Figure 1b by increasing the array size to include the coding bits. Extra control and data manipulation facilities (e.g. MUXes, parity trees, XORs, registers, etc.) are added to perform error correction and detection, as well as error coding when writing into the memory. When j logical words are combined into a larger physical word to limit the increase in array size, extra logic in the form of wider data paths, more complex coding/decoding circuitry, and a final one-of-j switch is needed.

TABLE 4-1
Memory Chip Failure Rates

chip : 4096 bit, 1 bit per word

chip failure rate λ_c	bit failure rate λ_b
0.05	0.0000122
0.2	0.0000488
0.5	0.000122
3.0	0.000732
5.0	0.00122

The block coded memory is shown in Figure 1c. The coding/decoding logic for block

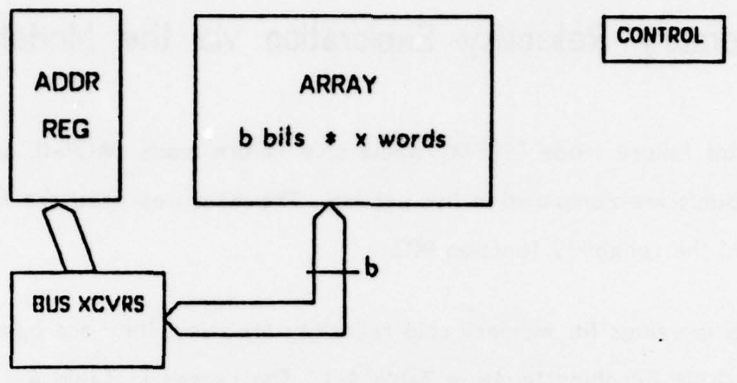


FIGURE 1a. Nonredundant memory model

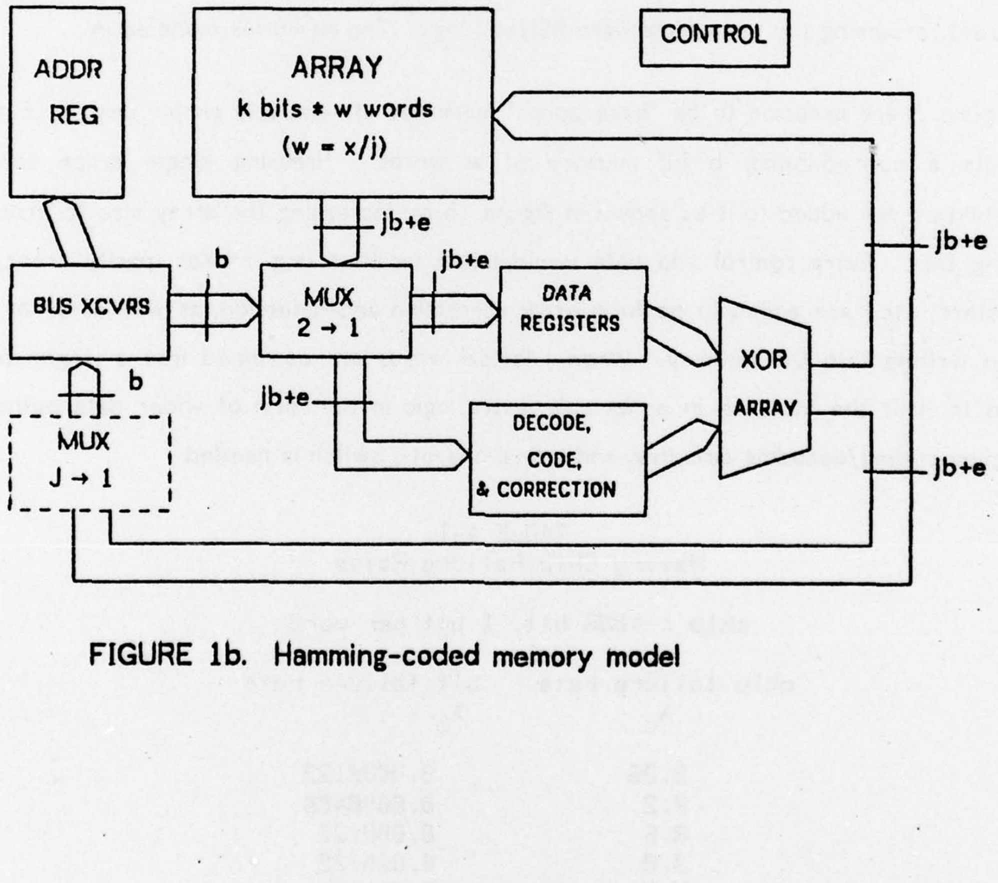


FIGURE 1b. Hamming-coded memory model

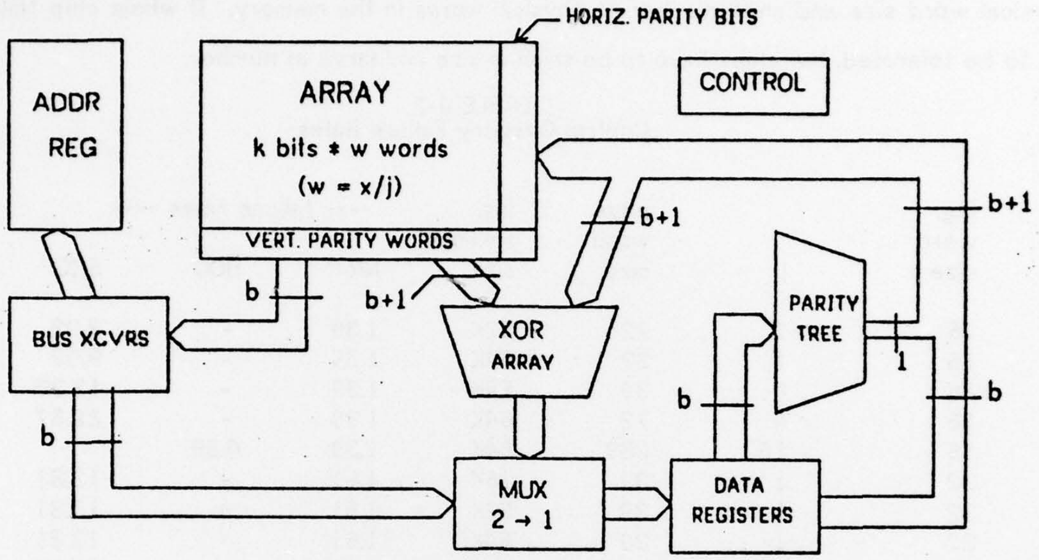


Figure 1c. Block-coded RAM model

coding is less complex than for a Hamming code. For example, only one parity tree is needed where the Hamming coded memory needs several. Also the block code requires fewer redundant bits than the Hamming code. The block code works in the following manner. When a word is read and XORed with zeros being fed into the other leg of the XOR array (0 is the XOR identity operator), the horizontal parity is calculated by the parity tree. If there is an error, the vertical parity for the block is calculated by successively XORing words from the memory block with what is already in the register. Note that the vertical parity word could be stored in a register file outside of the linear memory address space. The results of the new vertical parity point to the bit in error. If more than one horizontal parity bit in the block indicates an error, a multiple bit failure has occurred and the error is unrecoverable. In the case of a write, the horizontal parity is calculated and the vertical parity is updated simply by XORing the new and old data words with the old vertical parity word. Since writes to memory occur only 10-30% of the time, degradation due to vertical parity update is small. However, the block code is particularly effective for read only memory since the extra complication on writes is not necessary. Note that the vertical parity word could be stored in a separate memory array, thus allowing the update of the vertical parity word to proceed in parallel with the data write.

Block coding of small memories presents some problems because of the relatively large physical word size and small number of physical words in the memory. If whole chip failures are to be tolerated, the chips have to be small in size and large in number.

TABLE 4-2
Control Circuitry Failure Rates

log. word size	j	phys. word size	log. memory size	---- failure rates ---- NR	BCC	ECC
16	1	22	32K	1.39	-	9.02
16	1	22	64K	1.39	-	9.02
16	2	39	64K	1.39	-	13.35
16	4	72	64K	1.39	-	22.67
16	16	289	64K	1.39	4.39	-
32	1	39	16K	1.61	-	12.81
32	1	39	32K	1.61	-	12.81
32	1	39	64K	1.61	-	12.81
64	1	72	8K	2.06	-	20.39
64	1	72	32K	2.06	-	20.39
64	1	72	64k	2.06	-	20.39

The resulting control reliabilities are summarized in Table 4-2, and the detailed design/reliability derivations can be found in Appendix APPB. All failure rates are in units of failures per million hours.

4.1. MTTF

To make MTTF comparisons of the SBFM and WCFM models, a normalized MTTF is used. This is done to avoid dependence on specific reliabilities of the current or any other technology, and was accomplished by multiplying the MTTF formulae from Sections 3.2.1 and 3.2.2 by λ_b. When this is done the MTTF becomes a function of the ratio λ_e/λ_b, instead of being a function of λ_e and λ_b. The MTTF of the memory becomes

$$MTTF_{sb.norm} = \left(\frac{1}{f_0} + \frac{kg_0}{f_0 f_1} + \dots + \frac{k^w g_0 \dots g_{(w-1)}}{f_0 \dots f_w} \right)$$

for the SBFM model, and

$$MTTF_{wc.norm} = \frac{1}{d}\left(\frac{1}{f_0} + \frac{kg_0}{f_0 f_1} + \cdots + \frac{k^h g_0 \cdots g_{(h-1)}}{f_0 \cdots f_h}\right)$$

for the WCFM model. Note that $MTTF_{wc.norm}$ is still dependent on the number of bits per chip. The control circuitry MTTF can also be normalized using multiplication by λ_b, and becomes

$$MTTF_{e.norm} = \lambda_b/\lambda_e.$$

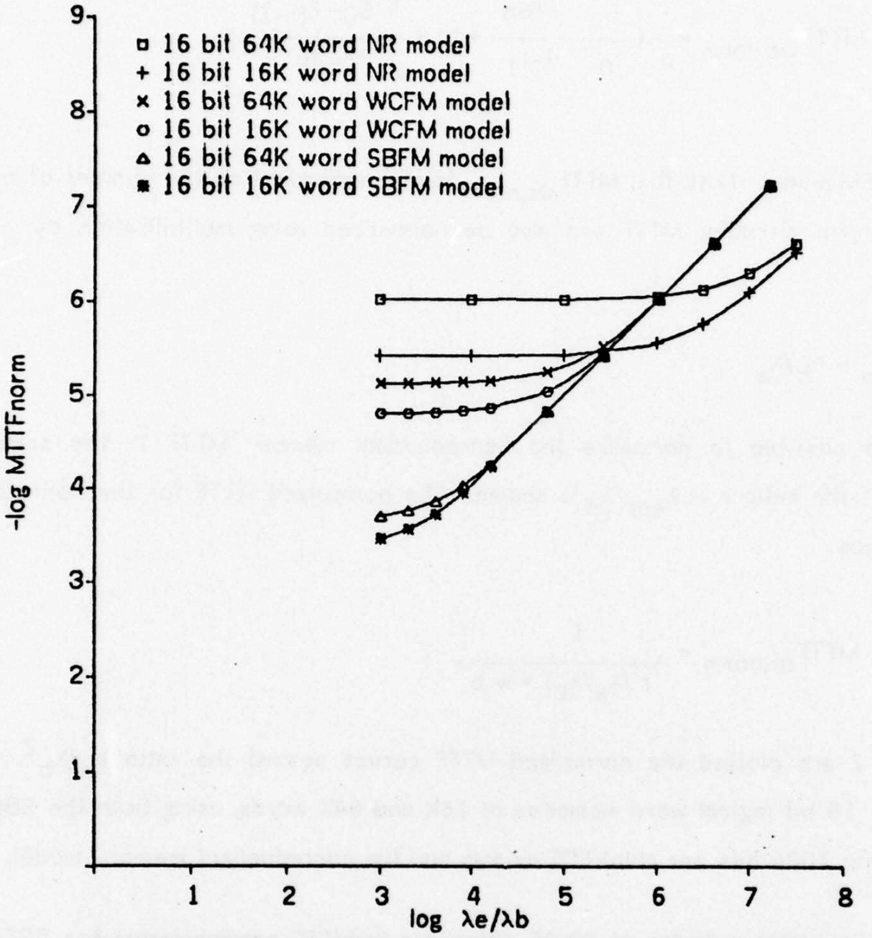
It is also possible to normalize the nonredundant memory MTTF in the same way, presuming that the ratio $r = \lambda_{enr}/\lambda_e$ is known. The normalized MTTF for the nonredundant memory becomes

$$MTTF_{nr.norm} = \frac{1}{r\,(\lambda_e/\lambda_b) + w\,b}\ .$$

In Figure 2 are plotted the normalized MTTF curves against the ratio λ_e/λ_b[2]. These curves are for 16 bit logical word memories of 16K and 64K words, using both the SBFM and WCFM (assuming 1024 bits per chip) ECC models and the nonredundant memory model.

Figure 2 illustrates a factor of 20-25 difference in MTTF prediction for the SBFM over the WCFM model for small values of λ_e/λ_b, with the size memories modeled. As λ_e/λ_b increases, the ECC memory MTTF becomes essentially that of the support circuitry (which would plot as a line with unit slope going through the origin). Thus the limiting factor on the memory reliability is the support circuitry reliability. The plot also shows that the ratio λ_e/λ_b at which the array reliability can be ignored in computing MTTF is lower for the SBFM model than for the WCFM model. This difference becomes greater for larger chip size. For λ_e in the range from 1 to 100 this corresponds to a λ_e/λ_b of 10^4 to 10^6 for the λ_b values in Table 4-1. This is well into the range where the SBFM assumption shows that the memory reliability can be modeled as simply that of the support circuitry, and just at or

[2] To interpret Figure 2 in terms of a specific technology, calculate $-\log \lambda_b$ and subtract from the Figure 2 vertical scale



COMPARISON OF SBFM, WCFM, AND NR MTTF MODELS

Figure 2

below that range for the WCFM assumption.

. The normalized MTTF for the nonredundant memory (assuming $r = 0.1$) is also plotted in Figure 2. It shows the same behavior as the ECC memories, i.e. the MTTF is limited by the control circuitry MTTF, although at a higher value of λ_e/λ_b. It also points up the fact that by the time that

$$\lambda_e/\lambda_b \geq \frac{w\ b}{(1-r)} ,$$

nonredundant memory becomes more reliable than ECC memory, and that for large λ_e/λ_b, its MTTF is greater by the factor $1/r$.

In summary, the formulas and derived curves, such as Figure 2, can be used to select the appropriate memory organization as a function of λ_e/λ_b and failure modes assumptions.

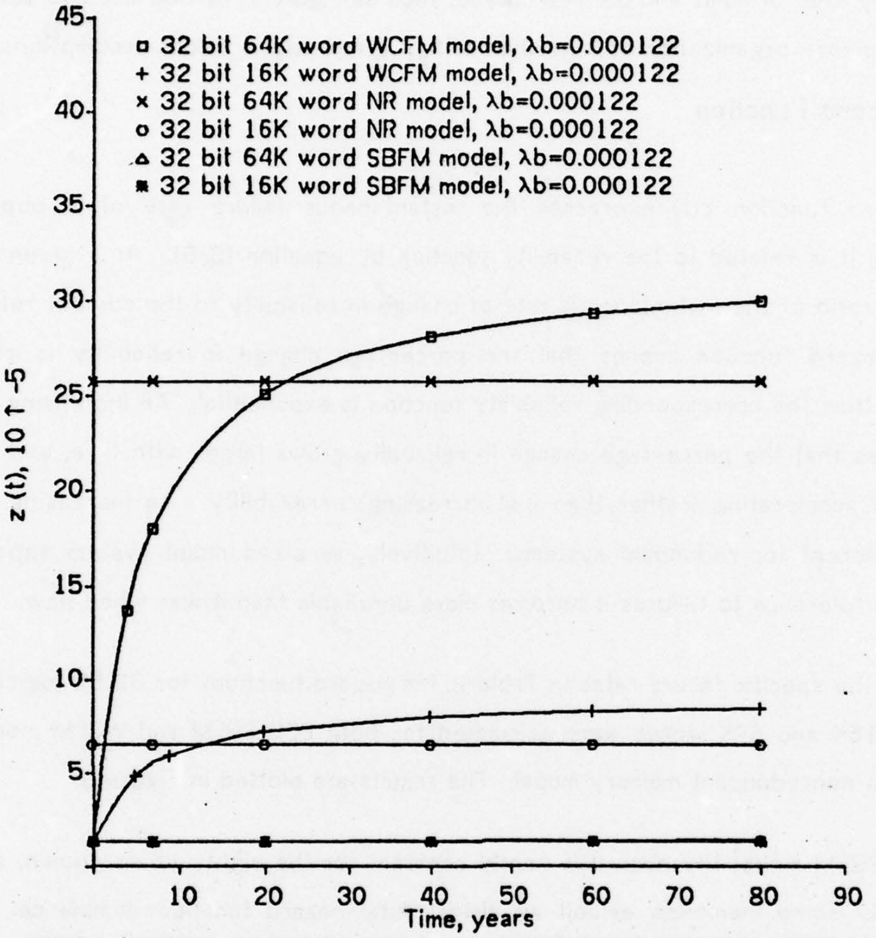
4.2. The Hazard Function

The hazard function z(t) expresses the instantaneous failure rate of a population. Mathematically it is related to the reliability function by equation (3-3). At a given time it measures the ratio of the instantaneous rate of change in reliability to the current reliability. A constant hazard function implies that the percentage change in reliability is constant through time, thus the corresponding reliability function is exponential. An increasing hazard function implies that the percentage change in reliability grows larger with time, and can be thought of as accelerating (rather than just increasing) unreliability. An increasing hazard function is inherent for redundant systems. Intuitively, as a redundant system approaches the limit of its tolerance to failures it becomes more unreliable than it was when new.

Based on the specific failure rates in Table 2, the hazard functions for 32 bit logical word memories of 16K and 64K words were calculated for both ECC SBFM and WCFM models, as well as for the nonredundant memory model. The results are plotted in Figure 3.

For the SBFM model the hazard is nearly constant for the eighty years shown, and the two differently sized memories exhibit an almost total hazard function dominance by the control circuitry's constant hazard function $z(t) = \lambda_e$. The WCFM model exhibits much different behavior for this ratio of λ_e/λ_b. For both sizes of memory the hazard functions increase throughout the eighty years, with a rapid rise in the first 10 to 20 years as the memory array hazard function grows and eventually dwarfs the contribution of the control circuitry's constant hazard function. At the end of 15 to 25 years the WCFM models have larger hazards than the models for the nonredundant memories of the same (logical) size. These latter also exhibit constant hazard functions, which for larger size memories are dominated by the greater constant hazard of the memory array alone (i.e. $\lambda_a >> \lambda_{enr}$).

The SBFM model hazard function is the same in form as the WCFM model hazard function, but exhibits different behavior for the same values of λ_e/λ_b, as seen in Figure 3. Figure 4

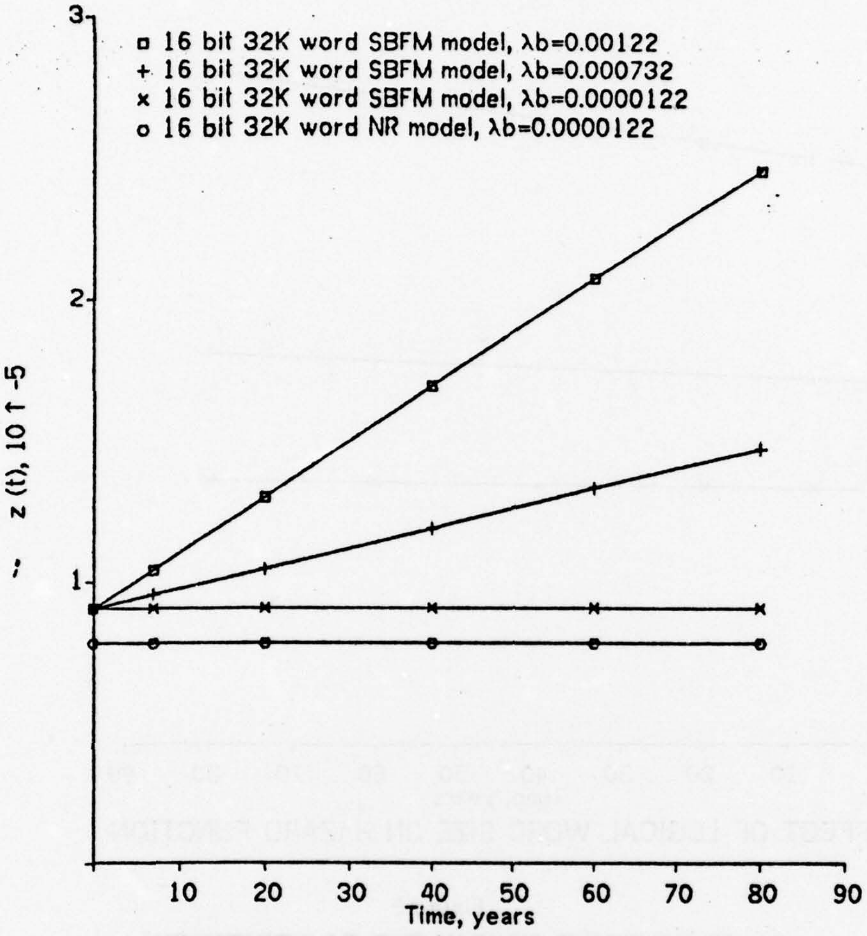


SBFM, WCFM, AND NR MODEL HAZARD FUNCTION COMPARISON

Figure 3

demonstrates the effect of varying λ_b while holding λ_e constant (e.g. more reliable memory for the same control technology, thus increasing λ_e/λ_b). For larger λ_b the memory array hazard function becomes more important and the SBFM model begins to exhibit the same qualities seen in Figure 3 for the WCFM model. Below some λ_b the nonredundant memory model has a consistently lower hazard function than the SBFM model, as shown by the lowest curve in Figure 4 (the nonredundant memory models for $\lambda_b \geq 0.000732$ are well above the range of the Figure 4 plots).

The effect of logical word size on memories of the same size (in words) and the effect of logical memory size (in terms of the total number of bits) are shown in Figure 5. An increase

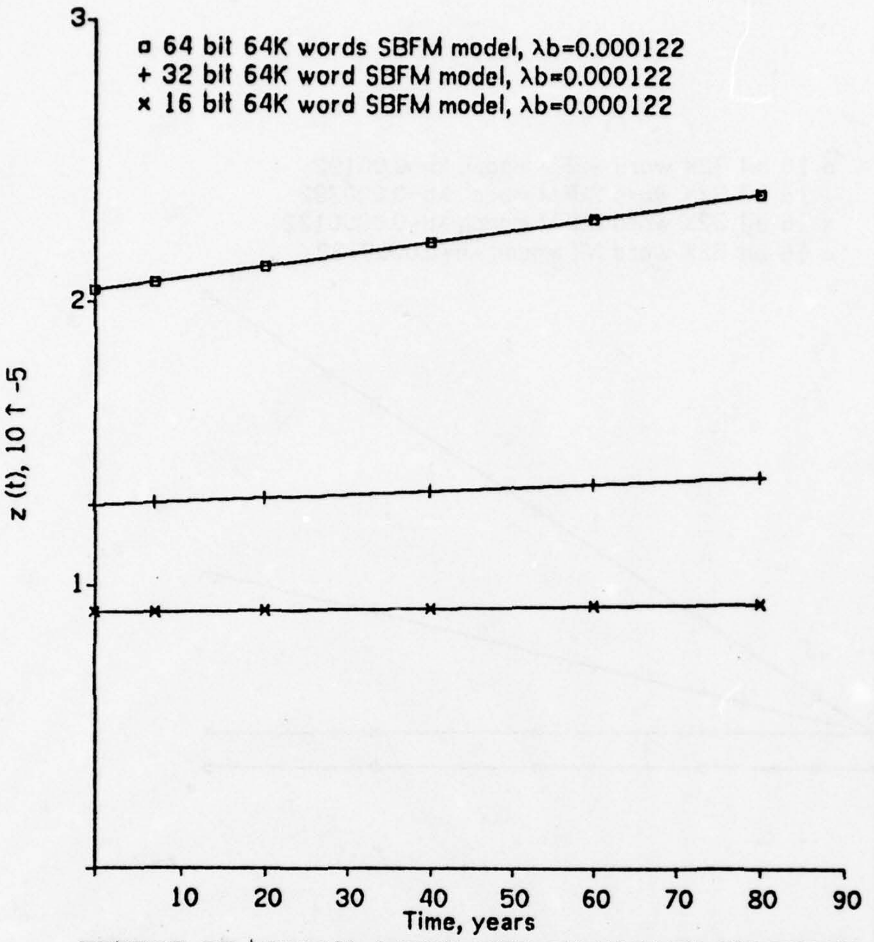


SENSITVITY OF THE HAZARD FUNCTION TO λb

Figure 4

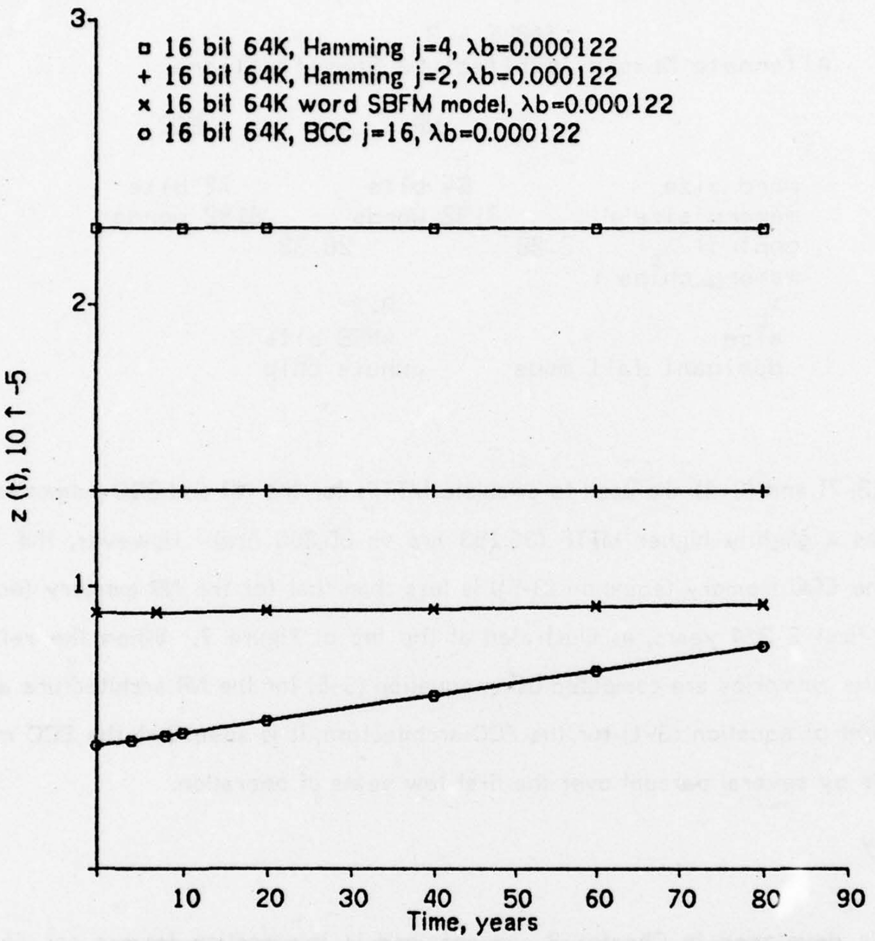
in word size and/or memory size causes a corresponding increase in the initial hazard (i.e. that of the control alone), while the increase in the number of memory cells causes a larger dependence on the memory array hazard function in spite of the increase in λ_e/λ_b.

Block and Hamming coded memories are compared in Figure 6. The three upper curves are for Hamming coded memories with 1, 2 and 4 logical words per physical word. The major effect of saving on memory chips by combining logical words is to decrease memory reliability. Since the control logic failure process is dominant, adding control logic simply increases the hazard function by a constant (corresponding to a decrease in reliability of the Hamming code memory).



EFFECT OF LOGICAL WORD SIZE ON HAZARD FUNCTION

Figure 5



z(t) FOR LOGICAL WORD SIZE < PHYSICAL WORD SIZE

Figure 6

The block coded memory behaves differently. Because its control circuitry is less complex than that of any of the Hamming memories, it has a lower initial hazard function. The slope of the hazard function shows the effect of the greater inherent unreliability of the larger word size. Even so, the block code memory does not become more unreliable over the eighty years because its hazard function never gets as large as the Hamming code hazard functions.

4.3. MTTF, the Hazard Function, and Reliability: an Example

This subsection brings together all of the tools developed so far to help in a decision between a nonredundant and a Hamming coded memory. Specifications for the two alternate architectures are given in Table 4-3.

TABLE 4-3
Alternate Memory Architecture Specifications

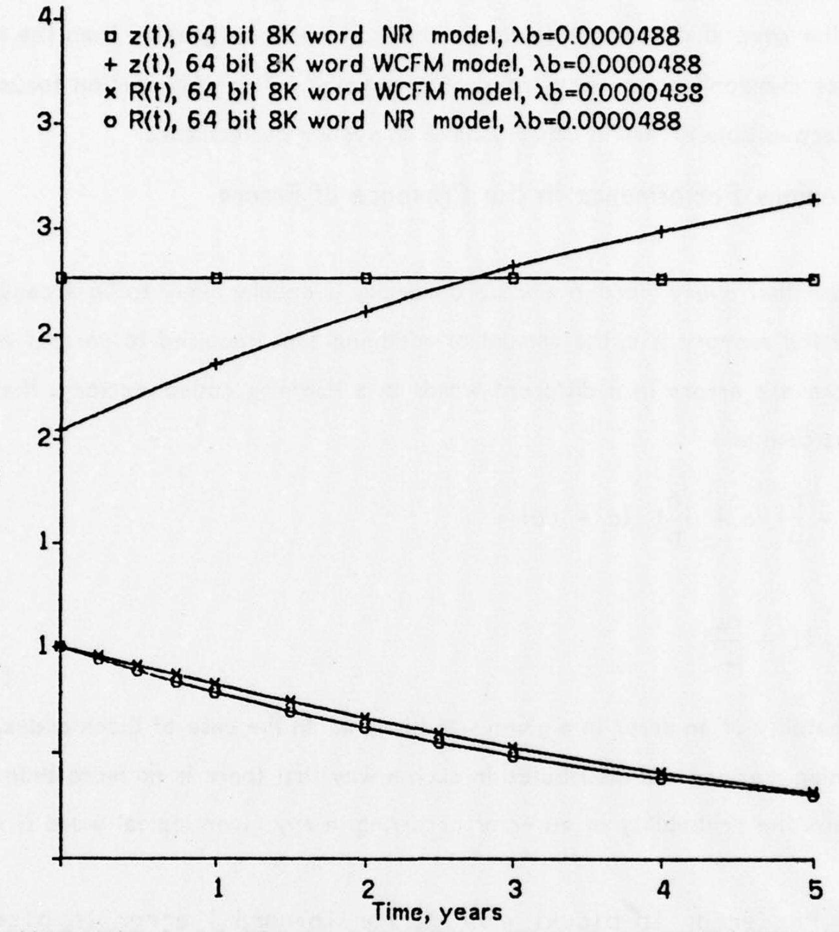
	NR	ECC
word size	64 bits	72 bits
memory size	8192 words	8192 words
control λ_e	2.06	20.39
memory chips :		
λ_c	0.2	
size	4096 bits	
dominant fail mode	whole chip	

Equations (3-7) and (3-4) are used to calculate MTTFs for the NR and ECC memories. The NR memory has a slightly higher MTTF (36,153 hrs vs 35,800 hrs). However, the hazard function for the ECC memory (equation (3-5)) is less than that for the NR memory (equation (3-8)) for the first 2 3/4 years, as illustrated at the top of Figure 7. When the reliability functions for the memories are computed using equation (3-6) for the NR architecture and the WCFM *equivalent of equation* (3-1) for the ECC architecture, it is seen that the ECC memory is more reliable by several percent over the first few years of operation.

4.4. Summary

The models developed in Chapter 3 and analyzed in this section form a set of easily applied tools which can help in evaluating memory system design spaces. One indicator alone is often not enough, as demonstrated in the previous subsection.

ECC memories are not inherently more reliable than nonredundant ones. With very reliable memory chips the limiting factor on reliability is the control circuitry. When using standard SSI/MSI logic Hamming code control circuitry has a failure rate several times that of the control circuitry for an equivalent NR memory. Block coded memory, which needs less complex control circuitry, is more reliable than Hamming code memory. Using more reliable LSI logic for ECC control would greatly improve the total ECC memory reliability.



CHOOSING A 64 BIT 8K MEMORY - ECC OR NONREDUNDANT ?

Figure 7

5. Performance Effects of ECC

The use of ECC memory for main memory or microstore affects system performance. Since in most cases error checking can be carried out in parallel with the use of the data there will usually be no performance change in an error-free state. This is possible if no irreversible actions (e.g. overwriting information needed to restart the current operation) are taken before the error checking has been completed, and if the hardware has stall/restart capabilities. Most processor/main memory systems and vertically coded microemulators belong in this class. As a counterexample a horizontally microcoded machine with a short microcycle and a very long word width would not allow this, as the propagation time through the several decoding tree levels required would be greater than the microcycle time. This case shouldn't occur very frequently, however. Thus this section focuses on the effect which recoverable errors in memory have on system performance.

5.1. Main Memory Performance in the Presence of Errors

First, assume that every word in a w word memory is equally likely to be accessed. If the access time of the memory is c, the amount of additional time required to correct an error is ϵc. When there are errors in n different words in a Hamming coded memory, the expected memory access time is

$$(1 - \frac{n}{w})\ c + (\frac{n}{w})\ (c + \epsilon c)$$

$$= c\ (1 + \frac{n\epsilon}{w}) \qquad (5\text{-}1)$$

since the probability of an error in a given word is n/w. In the case of block codes, errors in a still-functioning memory are distributed in such a way that there is no more than one error per block. Thus the probability of an error occurring in any given logical word (j words per block) is

$$P = \Pr\ [\text{error in block}] * \Pr\ [\text{error in word} \mid \text{error in block}]$$

$$= \frac{n}{(w/j)} * \frac{1}{j} = \frac{n}{w} \qquad (5\text{-}2)$$

so that equation (5-1) still holds.

Next assume that the access frequency is not uniform throughout the memory, so that some memory segments, such as those containing parts of the operating system kernel, are more likely to be accessed than others. Suppose that each location i has access probability P_i, and that there are n errors in memory. The expected memory access time can be expressed as the weighted sum

$$\sum_{i=1}^{w} P_i \left(1 - \frac{n}{w}\right) c + \sum_{i=1}^{w} P_i \left(\frac{n}{w}\right) (c + \epsilon c)$$

$$= \left(c + \frac{n}{w} \epsilon c\right) \sum_{i=1}^{w} P_i ,$$

which reduces to equation (5-1) as well since

$$\sum_{i=1}^{w} P_i = 1.$$

Thus in both cases the expected degradation of the memory access time is $n\epsilon/w$.

The effects of errors on memory access time are illustrated in Table 5-1 for several values of n and w. Two types of ECC memory are represented: a Hamming code memory with an ϵ of 1, and a block coded memory with an ϵ of 128 due to the necessity of reading all of the words in the block to determine the vertical parity. The performance degradation is negligible (< 1%) for the Hamming code, while the degradation becomes significant for the block code only when n becomes large.

The degradation of system performance depends on how often the memory is accessed. A system with a low memory bandwidth utilization will exhibit less degradation than one where the bandwidth is almost saturated. A comparison of three different PDP-11 systems serves as a good example. The data in Table 5-2 are drawn from [Snow77] and are the result of dynamic measurements of PDP-11 programs. Another result from the same source is that an average of 2.3166 memory references are caused for each instruction. If T_m is the memory access time, T_I the average instruction execution time, and D the expected memory access time degradation, the expected system degradation D_s is

$$D_s = \frac{D\ T_m\ (2.3166)}{T_I} .$$

TABLE 5-1

Normalized Expected Memory Access Degradation
As a Function of Memory Size (w words),
Number of Failures (n),
and ECC ($\epsilon=1$ for Hamming, $\epsilon=128$ for Block Code)

n	ϵ	16K words	128K words
1	1	0.00006	0.0000076
	128	0.0078	0.00098
4	1	0.0002	0.000031
	128	0.031	0.004
10	1	0.0006	0.000076
	128	0.078	0.0098
100	1	0.006	0.00076
	128	0.78	0.098

TABLE 5-2
Timing Data for PDP-11 Computer Systems

	time in microseconds for :	
system	memory access	avg. instruction exec.
LSI-11	.400	5.883
PDP-11/10	.600	4.096
PDP-11/34	.940	3.129

The data in Table 5-3 results from this expression. Even when there is severe (10%) memory degradation, the system degradation is negligible except for the PDP-11/34 system, whose processor comes close to saturating its processor-memory bandwidth. Therefore, even though the memory performance degradation is more serious for block codes than Hamming codes as shown in Table 5-1, the overall system performance would be comparable over wide ranges of failure situations.

5.2. Microstore Performance in the Presence of Errors

Microstore reliability is becoming more important as the use of microcoded system design is increasing. The growing size of microstores being used and the subsequent effect on

TABLE 5-3
Instruction Degradation for Various Models of The PDP-11 Assuming Various Amounts of Memory Access Degradation

Memory Cycle Time Degradation	Instruction degradation LSI-11	PDP-11/10	PDP-11/34
0.0001	0.0000158	0.000034	0.00007
0.001	0.000158	0.00034	0.0007
0.01	0.00158	0.0034	0.007
0.1	0.0158	0.034	0.07
1.0	0.158	0.34	0.7

system reliability makes error coding techniques more attractive. Unlike main memory where very degraded segments of main memory can be left unallocated, degraded sections of microcode are permanently allocated and will continue to affect system performance until they can be repaired. For these reasons system performance degradation in the presence of microstore errors is an important issue.

5.2.1. A Model for Performance Degradation

A simplified view of a microcoded machine is outlined in Table 5-4. It is assumed that all F fetch and S service microwords are executed during each macrocycle. The expected macrocycle time M_0 with no errors present is

$$E[M_0] = (F + S + \sum_{j=1}^{a} A_j P_j + \sum_{k=1}^{i} I_k P_k)\ m$$

$$= (F + S + \bar{A} + \bar{I})\ m$$

where m is the microcycle time.

In formulating the performance degradation model two further assumptions are made. The first is that the probability distribution of errors is uniform over all memory words. The second is that an error code with one logical word per physical word is being used. If the excess time needed to correct a word with an error is ϵm and there are n errors in the memory, the expected macrocycle time is

$$E[M_n] = F + S + \bar{A} + \bar{I} + (F\frac{n}{w} + S\frac{n}{w} + \sum_{j=1}^{a} P_j\, A_j\, \frac{n}{w} + \sum_{k=1}^{i} P_k\, I_k\, \frac{n}{w})\ \epsilon m$$

since the expected number of errors in I words is In/w. This reduces to

$$E[Mn] = E[M_0] + \frac{n\epsilon}{w}\ (F + S + \bar{A} + \bar{I})$$

$$= E[M_0]\ (1 + \frac{n\epsilon}{w}) \qquad (5\text{-}3)$$

Thus the expected performance degradation is $n\epsilon/w$, as with main memory.

TABLE 5-4
Microstore Model - Allocation and Access Frequency

Purpose	Size	P[access]	# in Microstore
fetch	F	1	1
interrupt service	S	1	1
addressing mode	A_j	P_j	a
instruction	I_k	P_k	i
total memory	$w = F + S + \sum A_j + \sum I_k$		

In the case of a block code of j words per block, the expected number of errors in I words is

$$\sum_{i=1}^{I} P(i) = \sum_{i=1}^{I} \frac{n}{w/j} * \frac{1}{j} = \frac{In}{w}$$

(see equation (5-2)), assuming that there are n errors in a functioning microstore. Thus equation (5-3) still holds.

Computers with microstores of 256, 1024, and 4096 words are considered as examples. The performance degradation expected with n=1,2, and 3 errors present is presented in Table 5-5 for the cases where ϵ=1 (Hamming code) and ϵ=16 (block code, 16 words per block). The expected degradation is negligible for the Hamming code case. For the block coded case the degradation is negligible when the block size is small in relation to the memory size.

TABLE 5-5
Expected Macrocycle Degradation in the Presence of Errors

w	ϵ	number of errors 1	2	3
256	1	0.0039	0.0078	0.017
	16	0.0625	0.125	0.1875
1024	1	0.0010	0.0020	0.0029
	16	0.0156	0.0313	0.0469
4096	1	0.0002	0.0005	0.0007
	16	0.0039	0.0078	0.0117

5.2.2. Distribution of Performance Degradation

Given a machine such as the one outlined in Table 5-4, the probability distribution of the performance degradation with n errors present can be computed. The locations of the n errors in microstore can be represented by the vector $\underline{f}$, which has one element for each of the fetch and service areas as well as for each of the instructions and addressing modes. Thus $\underline{f}$ has (2+a+i) elements which sum to n. The degradation probability distribution can then be computed using the formula

$$P(\underline{f}) = \frac{\binom{F}{f_F}\binom{S}{f_S}\binom{A_1}{f_{A_1}}\cdots\binom{A_a}{f_{A_a}}\binom{I_1}{f_{I_1}}\cdots\binom{I_i}{f_{I_i}}}{\binom{W}{n}} \qquad (5\text{-}4)$$

for the probability of a given error vector . The expected performance degradation associated with the combination of this vector over all addressing modes and instructions is

$$D(\underline{f}) = \sum_{j=1}^{a}\sum_{k=1}^{i}(f_F + f_S + f_{A_j} + f_{I_k})\, P_j\, P_k\, \epsilon \qquad (5\text{-}5)$$

which gives the expected degradation in terms of microstore cycle time. This quantity is evaluated for each valid $\underline{f}$ and the results compiled to give the probability distribution for performance degradation with n errors present.

As an example, consider the case of one microstore error in the machine described in Tables 5-6 and 5-7, which detail a simple model of the PDP-11 based on [Snow77]. In this example, addressing modes with approximately equal access probabilities are grouped into classes. Instructions are treated in the same way. With only one error present,equation (5-4) for the probability of the the vector f occuring reduces to

$$P(\underline{f}) = \frac{R_x}{w}$$

where the nonzero element of $\underline{f}$ is in the xth element, and the section of code containing the error is represented by R_x (i.e. R_x corresponds to one of F, S, A_b, or I_c). The application of formula (5-5) also simplifies considerably since only one functional area of the microcode can have an error in it. The resulting probability distribution of the performance degradation is listed in Table 5-8.

TABLE 5-6
Microstore Specifications

$F = 3$
$S = 10$
$A_j = 3$ for all j
$I_k = 3$ for all k
$a = 16$
$i = 65$
$w = 256$

The probability of negligible (<1%) degradation is 93%. The probability that the degradation is less than the expected degradation (.0039 from Table 5-5) is 86%. The probability of noticeable degradation (>5%) is only 5%, while severe degradation does not occur.

Table 5-9 contains the probability distribution for the above machine when two errors are present. Although there is a possibility of severe degradation, the probability is small

TABLE 5-7
Assumed Dynamic Distributions of
Addressing Modes and Instructions

Addressing Class	P_j	Number In Class
1	0.3	1
2	0.15	2
3	0.08	2
4	0.05	3
5	0.03	4
6	0.0	4

Instruction Class	P_k	Number In Class
1	0.2	1
2	0.08	2
3	0.03	10
4	0.015	16
5	0.003	36

TABLE 5-8
Probability Distribution of Performance Degradation
(one error present)

Degradation	Probability
0.0526	0.0508
0.0159	0.0117
0.0108	0.0117
0.00796	0.0234
0.00434	0.0234
0.00424	0.0234
0.00265	0.0352
0.00163	0.117
0.00159	0.0469
0.000813	0.188
0.000163	0.422
0	0.0469

(.24%), while there is an 86% probability that the degradation will be less than 1%.

This same method was used to derive the probability distributions of degradation for a block coded microstore with the characteristics given in Table 5-10. The resulting distributions are presented in Table 5-11 and 5-12. The dynamic distributions of Table 5-7

TABLE 5-9
Probability Distribution of Performance Degradation
(two errors present)

Degradation	Probability
.105	.0024
.060-.068	.0048
.052-.057	.0917
.032	.0001
.020-.024	.0013
.016-.019	.0214
.011-.015	.0204
.005-.010	.0630
.002-.004	.1634
.001-.002	.3741
.0003	.1770
.0	.0780

were used for this example as well.

TABLE 5-10
Microstore Specifications

$F = 4$
$S = 12$
$A_j = 4$ for all j
$I_k = 4$ for all k
$a = 16$
$i = 64$
$w = 336$
16 words per block

The performance degradation for the block code microstore is more severe than for the Hamming coded microstore. With one error present, the probability of severe degradation (greater than 10%) is about 8%, while the probability of negligible degradation (1% or less) is only 65%. When there are two errors present, the chance of a severe performance loss is 17% and that of a benign failure drops to 40%.

TABLE 5-11
Probability Distribution for Degradation
With a Block Coded Microstore
(one error present)

Degradation	Probability
0.667	0.0476
0.2	0.0119
0.133	0.0119
0.1	0.0238
0.053	0.0595
0.033	0.0357
0.02	0.1666
0.01	0.1905
0.002	.4167
0.0	0.0476

TABLE 5-12
Probability Distribution of Performance Degradation
(two errors present)

.867	.0011
.800	.0011
.767	.0023
.667-.720	.0864
.300	.0006
.153-.253	.0245
.100-.143	.0563
.073-.087	.0090
.053-.064	.0710
.030-.040	.1080
.020-.025	.2314
.012	.1592
.0004	.1547
0.0	.0773

6. Conclusions

The way in which memory chips fail affects the reliability of single error correcting memories. It also dictates the choice of models for memory system reliabilities. When the dominant failure mode, chip failure rate, and control failure rate are known, the models presented can be used in making tradeoff analyses in memory system design.

Error correcting code memories are not automatically more reliable than equivalent nonredundant memories, as the limiting factor is the reliability of the control circuitry. For the same reason block coded memories tend to be more reliable than Hamming coded memories. An increase in the reliability of the control circuitry will bring about a corresponding increase in ECC memory system reliability, which is a strong argument for the use of LSI control circuitry.

When error checking and use of data is paralled, error correcting memories can have performance similar to nonredundant memories when no failures are present. In the majority of cases the performance of systems containing error correcting code memories experiences negligible degradation in the presence of failures. Block coded memories, which are more reliable than Hamming coded memories, experience more performance degradation when errors are present although the degradation is still negligible in most cases.

I. Proof of the MTTF Formulae

To derive the iterative formula for $MTTF_{sb}$ presented in this paper, the integral expression for $MTTF'_{sb}$ is evaluated in the following manner.

$$MTTF_{sb} = \int_0^\infty e^{-\lambda_e t} \left(k\, e^{-(k-1)\lambda_b t} - (k-1)\, e^{-k\lambda_b t}\right)^{w} dt$$

$$= \int_0^\infty e^{-\lambda_e t}\, e^{-\lambda_b (k-1) w t} \left(k - (k-1)\, e^{-\lambda_b t}\right)^{w} dt$$

The next step is to make the substitution

$$x = e^{-\lambda_b t} ,$$

$$dx = -\lambda_b\, e^{-\lambda_b t}\, dt ,$$

$$x|_{t\to\infty} = 0 ,$$

and $x|_{t=0} = 1.$

To further simplify the integral, let

$$m = (k-1)\, w + \lambda_e/\lambda_c - 1 ,$$

$$n = w ,$$

$$a = k ,$$

and $b = -(k-1).$

The integral becomes

$$MTTF_{sb} = -\frac{1}{\lambda_b} \int_1^0 x^m (a + bx)^n\, dx,$$

which has the recursive solution

$$MTTF_{sb} = -\frac{1}{\lambda_b} \left(\frac{x^{(m+1)} (a + bx)^n}{m+n+1} + \frac{an}{m+n+1} \int x^m (a + bx)^{(n-1)}\, dx \right)\Bigg|_1^0$$

After doing one more recu ion, the equation becomes

$$\mathrm{MTTF}_{sb} = -\frac{1}{\lambda_b}\left(\frac{x^{(m+1)}\ (a+bx)^n}{m+n+1}\right.$$

$$+\frac{an}{m+n+1}\left(\frac{x^{(m+1)}\ (a+bx)^{(n-1)}}{m+n}\right.$$

$$\left.\left.+\frac{a\ (n-1)}{m+n}\int x^m\ (a+bx)^{(n-2)}\ dx\ \right)\right)\Bigg|_1^0$$

More simplifications are now introduced. Let

$f_i = (m+n+1) - i \qquad (= \mu k + \lambda_e/\lambda_c - i),$

$g_i = n - i \qquad (= \mu - i)\ ,$

and $y = a + bx.$

With some rearranging the MTTF_{sb} equation reduces to

$$\mathrm{MTTF}_{sb} = -\frac{1}{\lambda_b}\left(\frac{x^{(m+1)}\ y^{g_0}}{f_0}\right.$$

$$\left.+\frac{a\ g_0}{f_0}\left(\frac{x^{(m+1)}\ y^{g_1}}{f_1} + \frac{a\ g_1}{f_1}\int x^m\ y^{g_2}\ dx\ \right)\right)\Bigg|_1^0$$

The final term in the recursion is

$$a\ g_{(n-1)}\int x^m\ y^{g_n}\ dx = \frac{a\ g_{(n-1)}\ x^{(m+1)}}{f_n}\ .$$

Thus, $x^{(m+1)}$ can be factored out, giving

$$\mathrm{MTTF}_{sb} = -\frac{x^{(m+1)}}{\lambda_b f_0}\left(y^{g_0} + \frac{a\ g_0}{f_1}\left(y^{g_1} + \frac{a\ g_1}{f_2}\left(\ldots\frac{a\ g_{(n-1)}}{f_n}\right)\ldots\right)\right)\Bigg|_1^0$$

When $x = 0$, $x^{(m+1)} = 0$, while at $x = 1$, $x^{(m+1)} = 1$ and

$$y^{g_i} = (k-(k-1))^{g_i} = 1,$$

giving

$$MTTF_{sb} = \frac{1}{\lambda_b f_0} \left(1 + \frac{a\, g_0}{f_1} \left(1 + \frac{a\, g_1}{f_2} \left(\ldots . \frac{a\, g_{(n-1)}}{f_n}\right)\right)\ldots .\right).$$

A final reorganization yields the formula presented in equation (3-2), namely

$$MTTF_{sb} = \frac{1}{\lambda_b} \left(\frac{1}{f_0} + \frac{a\, g_0}{f_0 f_1} + \ldots . + \frac{a^n\, g_0 \ldots . g_{(n-1)}}{f_0 \ldots . f_n}\right)$$

An important point to note is that in solving the integral m is assumed to be an integer, which in turn constrains λ_e/λ_b to also be an integer. In almost all cases this constraint is not a problem, because normally $\lambda_e >> \lambda_b$.

The derivation of the iterative formulae for $MTTF_{wc}$ and $MTTF_r$ follows the same route as that for $MTTF_{sb}$, with only the few parameter changes noted in sections 3.2.2 and 3.2.3. Depending on the implementation, the integer constraint for λ_e/λ_c may be a problem, but again, for most cases it should not be.

II. Derivation of the Support Circuitry Failure Rates

The support circuit failure rates in Table 4-2 for nonredundant, Hamming code, and block code memories are derived from models of the support circuitry required for the "no-frills" memory systems shown in Figure 1. These models provide rough estimates of the number and type of standard SSI/MSI TTL packages needed without necessitating actual circuit design.

TABLE B-1
Number of IC's in Support Circuitry

B bits/word, W words in Memory
K bits/physical word, J words/physical word

chip type	number of chips
(i) nonredundant memory	
random logic	10
bus xcvrs	$\lceil b/4 \rceil$
latches	$\lceil (\log 2\ w)/4 \rceil$
(ii) Hamming code memory	
random logic	30
bus xcvrs	$\lceil b/4 \rceil$
latches	$\lceil (\log 2\ w)/4 \rceil + \lceil k/4 \rceil$
parity trees	$(\lceil \log 2\ k \rceil \ast \lceil (\lceil k/2 \rceil \ast 10)/81 \rceil) + \lceil (k-1) \ast 10/81 \rceil$
comparators	$\lceil (2 + \lceil \log 2(j \ast b) \rceil) \ast 5/4 \rceil$
XOR	$\lceil k/4 \rceil$
inverter	$\lceil k/6 \rceil$
4→16 DEMUX	$\lceil k/16 \rceil$
2→1 MUX	$\lceil k/4 \rceil$ ($\ast 2$ iff $j=2$)
8→1 tristate MUX	$k \ast \lceil j/8 \rceil$ iff $j>16$
16→1 MUX	k iff $16 \geq j > 8$
8→1 MUX	k iff $8 \geq j > 4$
4→1 MUX	$\lceil k/2 \rceil$ iff $4 \geq j > 2$
(iii) block code memory	
random logic	30
bus xcvrs	$\lceil k/4 \rceil$
latches	$\lceil (\lceil \log 2\ w \rceil)/4 \rceil + \lceil (b+1)/4 \rceil$
parity trees	$\lceil b \ast 10/81 \rceil$
XOR	$\lceil (b+1)/4 \rceil$

The formulae used in computing numbers of packages are shown in Table B-1. For the overhead associated with control of the entire memory, an arbitrary number of "average" size chips (15 gates) was chosen. Ten such chips are used in the nonredundant memory model and thirty are used in each ECC memory model. These numbers are only "order of magnitude" guesses, but the inaccuracy involved is not enough to affect the conclusions drawn from these models since the major proportion of the support circuitry is in the data paths and data operators.

Failure rates for the integrated circuits are calculated using [DOD76] and the following assumptions:

$\pi_Q = 16.$ (class C)
$T_A = 40$ C
$\pi_E = 0.2$ (ground benign)
$\pi_L = 1.0$ (mature technology)

The resulting failure rates are listed in Table B-2.

TABLE B-2
Chip Failure Rates

chip type	model	gates*	λ
random logic	-	15	.077
latch	74175	24	.099
9 bit parity decode	74280	46	.230
comparator	7485	31	.181
4→16 DEMUX	74154	25	.103
inverter	7464	4	.048
XOR	7486	4	.039
2→1 MUX	74157	15	.077
4→1 MUX	74153	16	.082
8→1 MUX	74151	14	.074
" (tristate)	74251	17	.084
16→1 MUX	74150	25	.103
bus xcvr	-	8	.056

*obtained from [DOD76] when possible

These models do not necessarily provide accurate support circuitry failure rates of actual memories. However, they do show well the relative effects of the different coding schemes as well as of different memory parameters (e.g. word size, block size, number of words in memory) on the support circuitry reliability. Thus they can be used as in Chapter 4 to demonstrate the relative effects of the support and memory array circuitry on memory system reliability.

III. Bibliography & References

[Cox76]	Cox, G.W., and Carroll, B.D., "A Memory System Reliability Model", Proc. FTCS-7, IEEE Press 1976, p.203
[DOD76]	"Reliability Prediction of Electronic Equipment", Military Standardization Handbook MIL-HDBK-217B, U.S. Dept. of Defense, September 1974, and Notice 1, September 1976
[Gear76]	Gear, G., "Intel 2708 8K UV Erasable PROM", Intel Corporation, RR-12, September 1976
[Levi76]	Levine, L., and Meyers, W., "Semiconductor Memory Reliability with Error Detecting and Correcting Codes", Computer, vol. 9 no. 10, pp 43-50, October 1976
[Pasc75]	Pascoe, W., "2107A/2107B N-Channel Silicon Gate MOS 4K RAMS", Intel Corporation, RR-7, September 1975
[Pete72]	Peterson, W.W., and Weldon, E.J.Jr, "Error Correcting Codes", M.I.T. Press, 1972
[Rick76]	Rickers, H.C., "Microcircuit Device Reliability Memory/LSI Data", Reliability Analysis Center, RADC/RBRAC, MDR-3, Winter 1975-1976
[Shoo68]	Shooman, M.L., "Probabilistic Reliability; an Engineering Approach", McGraw-Hill, 1968
[Snow77]	Snow, E.A., and Siewiorek, D.P., "Impact of Implementation Design Tradeoffs on Performance: The PDP-11, a Case Study", Departments of Electrical Engineering and Computer Science, Carnegie-Mellon University, 29 July 1977

[Texa7?] "Preliminary Reliability Report for TI Series TMS4030, TMS4050, TMS4060 4K RAMS", Texas Instruments Inc., Bulletin CR-112, date unknown

[Wang77] Wang, S.Q., and Lovelace, K., "Improvement of Memory Reliability by Single-Bit-Error Correction", Proc. COMPCON Spring 1977, IEEE Press, February 1977, pp.175-178

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

14 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-CS-78-118	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 2 RELIABILITY AND PERFORMANCE MODELS FOR ERROR CORRECTING MEMORY AND REGISTER ARRAYS,		5. TYPE OF REPORT & PERIOD COVERED 9 Interim rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 Steven A. Elkind and Daniel P. Siewiorek		8. CONTRACT OR GRANT NUMBER(s) 15 N00014-77-C-0103
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Dept. Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE 11 22 May ~~22,~~ 1978
		13. NUMBER OF PAGES 43
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) same as above 12 46p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

→ Current digital system design practices make heavy utilization of various types of memories. RAMS and ROMS are used in main memory, register files, caches, and microstores. As a result, it becomes important to recognize the implications of memory chip failure modes for system reliability. → next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

44

403 081

YP

A brief survey of available memory chip failure mode data is made and shows that partial chip failures are more prevalent than whole chip failures. Based on the findings of this survey, reliability models for memory systems with error coding techniques are developed. The effect of memory support circuitry on memory reliability, usually ignored in the development of analytical models, is included. It is shown that for wide ranges of memory system parameters and memory element failure rates the memory system reliability is dominated by the effect of the support electronics. The use of these models in design tradeoff decisions is explored.

The performance of systems with fault tolerant memory when there are correctable failures present, an area which has seen little work, is analyzed. Performance models for systems with fault tolerant main memory, as well as those with fault tolerant microstore, are developed and their properties explored.

Hamming code is one of the error corecting techniques considered. Block codes, commonly used for tape media but rarely if ever for RAM or ROM, are also considered and found competitive with Hamming codes in many cases.