

 **TANDEM** COMPUTERS

Reliability Improvement Through Static RAM Sparing

Robert W. Horst

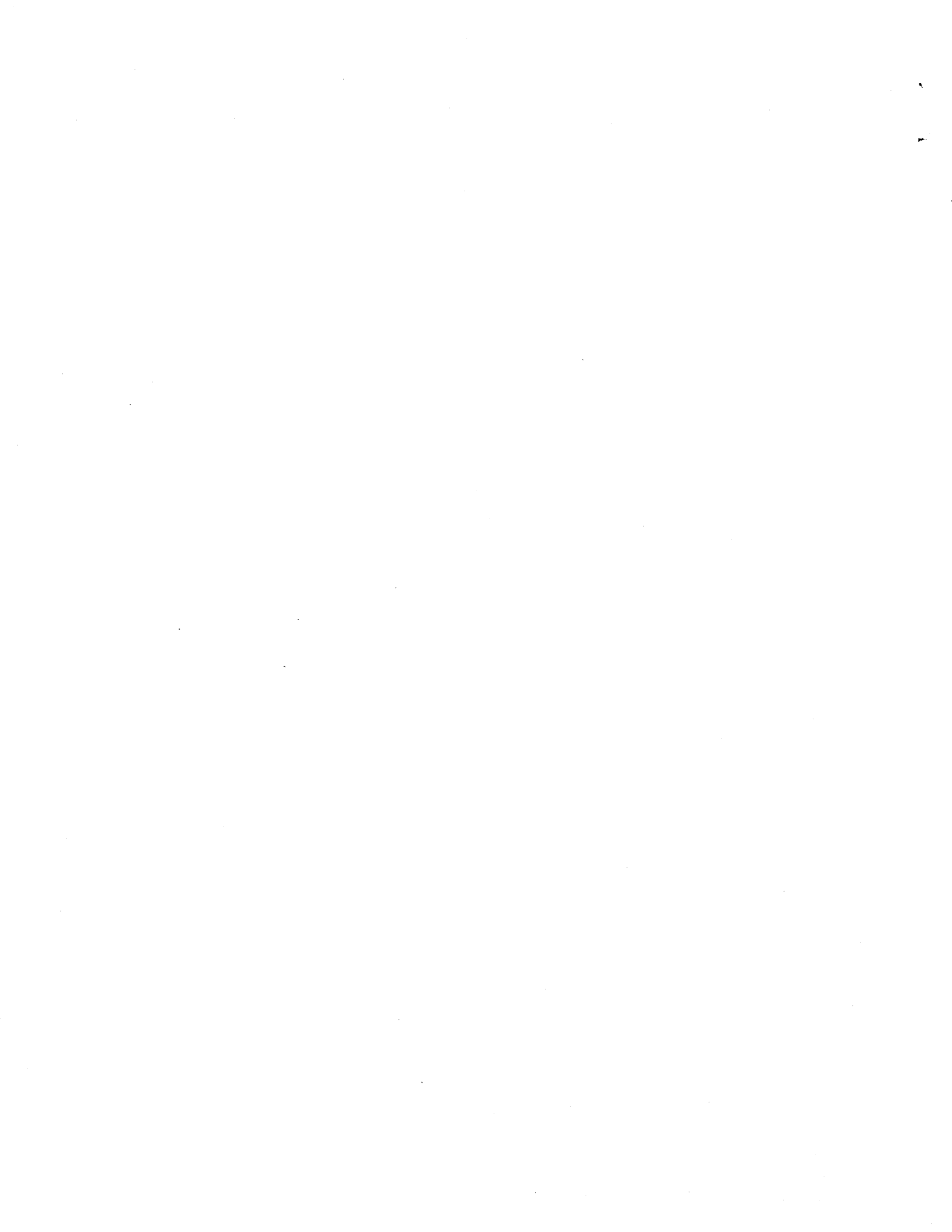
Technical Report 87.1
January 1987
Part Number 87618

RELIABILITY IMPROVEMENT THROUGH STATIC RAM SPARING

Robert W. Horst

January 1987

Tandem Technical Report 87.1



RELIABILITY IMPROVEMENT THROUGH STATIC RAM SPARING

Robert W. Horst

January 1987

ABSTRACT

This paper presents a practical approach to improving the failure rate of static RAM arrays through the use of spare RAMs. Assuming no preventative maintenance is performed, the repair rate of a spared array is a function of the failure rates of the other logic and other spared arrays on the same field-repairable unit. General equations are developed to express failure rate improvements for multiple spare RAM groups and varying amounts of other non-spared logic. The use of RAM sparing in the Tandem NonStop VLX is discussed.

TABLE OF CONTENTS

Introduction.....1
Previous Approaches.....2
The RAM Sparing Approach.....3
Repair Policies.....4
Analysis of Spared Array Failure Rates.....5
Multiple Different Spared Arrays.....8
Sparing in the NonStop VLX processor.....9
Conclusions.....10
References.....10
Figures.....12
BASIC Program to compute failure rates.....Appendix A

1. Introduction

The use of higher levels of integration has helped improve the reliability of new computer systems. Increasingly, the central processors are built from high performance microprocessors, or are built out of high density gate arrays or custom VLSI. All of these technologies require fewer parts and less power than earlier discrete logic implementations, resulting in dramatically improved reliability.

Another trend, however, is to increase the processor performance through the use of large cache memories. These caches are built from high speed static RAM memory components. Even though RAM densities have improved greatly, the part count of caches has not decreased in proportion; instead, larger caches are used to boost performance even more. In addition to their use in caches, static RAMs are used in large numbers for translation lookaside buffers, large scratchpad memories, and for control stores. As a result, failure rates of new systems are beginning to be dominated by the failure rates of the static RAMs.

The reliability of the static RAMs is becoming more questionable because static RAMs are no longer internally static. In order to attain much higher speeds and densities, RAM designers have borrowed some of the tricks used in dynamic RAMs and now use address transition detection to generate a series of clock pulses for each new memory access [FLAN86, HARD81]. The design of a totally foolproof circuit for address transition detection and clocking has proven to be an extremely difficult problem. It is difficult to rigorously prove that a circuit will not be fooled by narrow address line glitches or inputs hovering near threshold under all conditions of temperature, voltage, and process variation. Devising tests for all possible combinations of these parameters is not possible either. A careful system designer is wise to design his system, if possible, to continue operation despite a faulty RAM which has slipped through the component screening process.

Devising a way to tolerate RAM failures has a double benefit; system availability improves, and service costs decrease. If the system itself is fault tolerant, there is little room left for improvement in the hardware availability because system failures are dominated by operational and software failures [GRAY85]. However, all systems benefit from reliability improvements resulting in reduced service costs and reduced requirements for spare replacement boards. Hence, unlike previous papers which consider sparing to improve system availability, this paper considers RAM sparing as a way to reduce service costs.

2. Previous Approaches

Error correcting codes are widely used to survive dynamic RAM failures in main memories. Typically, modified Hamming codes are used for single-bit error correction and double-bit error detection. For 32 bit data words, this requires an additional 8 check bits, a fairly large overhead of 25%.

Another problem with ECC codes for static RAMs is that significant performance may be lost. In order to check an ECC code, first a syndrome is generated through several levels of exclusive-OR gates. The syndrome is passed through an encoder and to another level of exclusive-OR to flip the bit in error. In many systems, this takes an entire clock cycle. To slow down a processor's cache access by an extra clock cycle generally results in an unacceptable performance degradation. Even if the system is designed to inject an extra cycle only when a single bit error occurs, the resultant performance loss is great enough to require a service call.

The general approach of standby redundancy has been widely studied [BOUR69, NG80]. These papers generally consider the effect of standby redundancy on system availability, but do not suggest or analyze the use of standby redundancy for improving subsystem repair rates. [GROS81] has considered the use of small static RAMs to provide spare bits for dynamic RAM arrays, but this requires a much different analysis than the problem of whole RAM sparing considered here.

This paper directly addresses the theoretical and practical problems encountered in designing a RAM sparing scheme. Unlike previous papers, it takes into account the the possibility of multiple groups of spared modules on the same field replaceable unit (FRU), and it uses a more realistic repair policy.

3. The RAM sparing approach

A block diagram for RAM sparing is shown in Figure 1. The basic organization is that of K-out-of-N hot standby redundancy. Since the reliability gains generally do not warrant more than a single spare per group, this reduces to K-out-of-K+1.

A group of K RAMs has an additional spare RAM which may be substituted for any one failed RAM. Generally one of the K RAMs stores parity to allow the detection of any RAM failure. When a hard RAM failure is detected, the SPARE SELECT register is loaded with a value to select

the RAM for replacement. Write data to the spare is driven by a multiplexer which selects the appropriate write data line. Individual 2:1 output multiplexers driven by a decode of the SPARE SELECT allow one read data line to be driven from the spare instead of its associated RAM.

Multiple groups of spared RAMs may be configured by duplicating the structure of Figure 1. In some cases, multiple groups are required due to multiple arrays receiving different address or control lines. In other cases, dividing an array into multiple groups may be done purposely to further improve the failure rate. Tradeoffs in selection of the group size are considered in section 5.

Performance degradation is minimal since only a 2:1 multiplexer is inserted in the read data path. In many technologies this can be performed by a single AND gate delay plus a wired-OR. There may be some additional delay due to the added loading of the spare line, but this effect is small if the multiplexing is done internally to a gate array or custom VLSI component. Write data timing is usually not as critical, and the extra multiplexer on the spare RAM usually poses little problem.

The system may be designed to dynamically or statically invoke the sparing. If sparing is invoked statically, a RAM failure would cause a failure of the system (if non fault-tolerant) or module (if fault tolerant). It then would be brought back on-line manually or automatically through the maintenance subsystem with the spare selected.

It is more desirable to reconfigure the spare dynamically. In order to do this, there must be a backup copy of the failed RAM's data stored somewhere in the system. In a store-through cache, the backup data is readily available through the copy in main memory. Once the spare is invoked, every location in the spare which is not correct causes a parity error. Each parity error causes a cache miss which brings in a fresh copy of the data from main memory. Other arrays which do not have backup data readily available may require extra memory to allow the correct data to be refetched or reconstructed.

Allowing the arrays to be spared dynamically has an added benefit; the same backup mechanism can be used to recover from intermittent errors both in normal and spared operation. In fact, it is best to invoke the spare only after a predetermined threshold of soft errors has occurred.

4. Repair Policies

There are three reasonable policies for repairing the board after the first RAM failure: field replacement of the bad RAM on the next regular service call, replacement of the board on the next service call, and replacement of the board only after that board fails (due to a double RAM failure, or other logic failure). These alternatives are examined individually as follows:

Field RAM replacement

In order to field-replace the RAMs, they must be socketed. Experience on early Tandem processors has made us wary of this option for several reasons. If inexpensive sockets are used, there tend to be long term reliability problems associated with the sockets themselves; if expensive sockets are used, the extra cost may negate the service cost savings. Also, there are practical problems of stocking the field with correct spare RAMs and guaranteeing they will be handled properly to avoid physical and electrostatic damage. For these reasons, we do not believe socketing of large arrays is generally a good option. Instead, we solder the RAMs directly into the boards.

Board replacement for single RAM failures

Another replacement policy is to swap the board containing a single RAM failure the next time a CE is on-site for preventative maintenance or for any other failure requiring service. With this policy, only a portion of the service cost reduction is realized. The number of service calls is reduced, but the board repair cost is not reduced, and the field spare inventory is not reduced. In addition, customers generally prefer not to repair something which is not broken (as far as they can determine). They may prefer to retain the board which has a single failure, but which as performed faithfully for the last several months, as opposed to swapping-in the unknown spare the CE has brought. Even if the spare is good, anytime a CE has to touch a machine, there is some potential for human error which could cause more problems.

Board replacement for board failures

Tandem has chosen to adopt the policy of swapping the board only after it has failed completely either due to a double RAM failure, or to failure of some other logic on the same FRU. This policy gives the maximum savings from the RAM sparing, because some multiple component failures are repaired in a single repair cycle instead of individual cycles for each failure. In a fault-tolerant system, since the board failure does not cause a system failure, there is little impact on the customer if the board finally fails due to a second RAM failure. In not fault-tolerant systems, this policy may or may not make sense depending on the desired tradeoff between high availability and low service costs.

The following analysis assumes the repair-on-board-failure policy. Note that this policy is different from the those assumed in previous papers on standby redundancy with repair [NG80].

5. Analysis of Spared Array Failure Rates

The assumption of the repair-on-board-failure policy creates the interesting result that the failure rate of each spared array is a function of the rest of the logic on the same board. If there are many other components on the board, it is unlikely that a second RAM will fail before another component on the board, and the first failure will be fixed for free. The apparent failure rate (double failures) of a spared array approaches zero as the amount of other logic increases. On the other hand, if there is little other logic on the board, sparing can have a greater impact on improving the overall failure rate of that board.

If there are multiple groups of spared RAMs, each group affects the reliability of the others. With many groups, multiple single failures are likely to accumulate before the first double failure, again raising the probability of fixing multiple failures at a time.

Following is the derivation of a model which can be used to explain these effects. It is intended to be useful for design engineers in making tradeoffs in sparing, as well as by reliability engineers in computing expected failure rates of boards with spared RAMs.

Assumptions

The model used will be accurate only if the following assumptions are met:

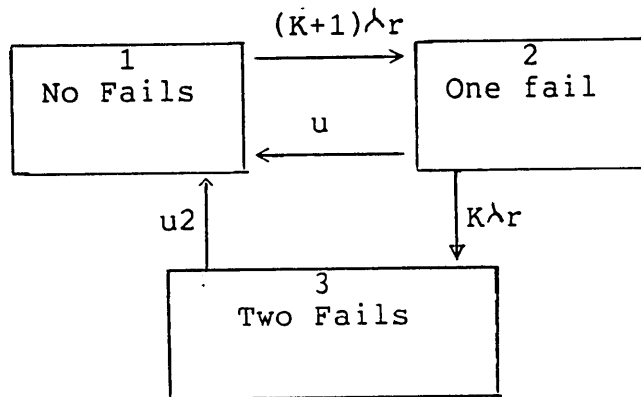
1. Component failure rates are all independent and exponentially distributed.
2. In spared arrays, there are no undetected single failures which can cause board failures.
3. After repair, the board is as good as a new board.
4. Repair rates are exponentially distributed and equal to the failure rate of the other logic on the board.

Definition of Terms:

λ_r	- Failure rate of one RAM
λ_b	- Failure rate of the Board (FRU)
λ_{sg}	- Failure rate of a Spared Group of RAMs
MTTF _{sg}	- Mean Time To (double) Failures of a Spared Group (= $1/\lambda_{sg}$)
K	- The number of RAMs in each group not counting the spare
G	- The number of identical spared groups on the board
L	- A constant used to express the non-spared logic failure rate in RAM equivalents. ($\lambda_{logic} = L\lambda_r$)

- u - Repair rate of the board to fix problems other than double failures in this group.
- u2 - Repair rate to fix double failures in this group.
- F1r - Failure rate Improvement per RAM. Equals the ratio of the failure rate of a group of K unspared RAMs to a group of K+1 spared RAMs.
- F1b - Failure rate improvement of the Board. Ratio of failure rates with and without sparing.

The Markov graph and transition matrix for one spared group are shown below.



$$A = \begin{bmatrix} -(K+1)\lambda_r & (K+1)\lambda_r & 0 \\ u & -u-k\lambda_r & k\lambda_r \\ u2 & 0 & -u2 \end{bmatrix}$$

Starting in state 1, the first failure occurs on the first transition to state 3. Solving for the MTTF using the linear algebra method described in [DHIL81]:

$$(1) \quad \text{MTTF}_{sg} = \frac{u + K\lambda_r + (K+1)\lambda_r}{\lambda_r \cdot (K+1)K}$$

The repair rate, u, equals the failure rate of the rest of the logic on the board. The trick in this analysis is to realize that because all groups are identical, the failure rate improvement of the other groups on the board is the same as the group whose failure rate we are trying to determine. Hence, the repair rate of this group equals the failure rate of the other groups plus the nonspared logic:

$$(2) \quad u = (G-1)(K+1)\lambda_r / F1r + L\lambda_r$$

Substituting for u in (1),

$$(3) \quad \text{MTTFsg} = \frac{(G-1)(K+1)/\text{FIr} + L + 2K + 1}{\lambda_r(K+1)K}$$

The failure rate improvement of the RAMs in this group is given by:

$$(4) \quad \text{FIr} = (K+1)\lambda_r/\lambda_{sg} \quad \text{where } \lambda_{sg} = 1/\text{MTTFsg} \quad \text{then}$$

$$(5) \quad \text{MTTFsg} = \text{FIr}/(K+1)\lambda_r$$

Setting equations (3) and (5) equal to each other yields a quadratic equation in FIr. Selecting the positive root gives:

$$(6) \quad \text{FIr} = \frac{L + 2K + 1 + \sqrt{(L + 2K + 1)^2 + 4K(G-1)(K+1)}}{2K}$$

And the total failure rate of the spared board is

$$(7) \quad \lambda_{sb} = (L + G(K+1)/\text{FIr})\lambda_r$$

The failure rate before sparing was

$$(8) \quad \lambda_b = (L + GK)\lambda_r$$

Then the board failure rate improvement ratio is

$$(9) \quad \text{FIB} = \lambda_b/\lambda_{sb} = \frac{L + GK}{L + G(K+1)/\text{FIr}}$$

The graphs of Figures 2a - 2c show FIr versus L for varying numbers of RAMs and groups. These graphs show how failure rate improvement of the RAMs improves with either more nonspared logic on the board (large L) or with the array partitioned into more groups (large G). These graphs can be used by a designer to get a rough idea of how failure rates will change with sparing. For instance, if the failure rate of the rest of the logic is estimated at about the same as 100 RAMs, then the failure rate of each RAM in a spared group of 64 will appear to improve by a factor of about 3.6 (i.e. in the failure rate calculations, it will look like only $66/3.6 = 18.3$ RAMs). Partitioning the array into two spared groups will make the failure rate improvement about 5.2 or equal to only $67/5.2 = 12.9$ RAMs.

Figures 3a-3b graph FIB versus L for varying numbers of RAMs and groups. These graphs show that sparing has greater overall impact on the board's failure rate with fewer nonspared components. They show that for small L, partitioning the logic into multiple spared groups can have a large impact on board failure rate, but with large L there may be little point in multiple groups; for large L, the nonspared logic is much more likely to fail before a second RAM failure, hence there is not much point in tolerating multiple RAM failures.

6. Multiple Different Spared Arrays

The closed form solution to spared array failure rates in equations (6) and (7) works only if all RAM arrays on the board have identical organizations and their RAMs have the same failure rates. On large boards, there may be multiple different types of spared arrays. One way to determine this failure rate would be to develop a unique Markov graph for the entire board and solve it. This approach is taken in [GOYA86], but the method proposed is extremely complex, and it still does not take into account the preferred repair policy. Below is a much simpler approach which builds on the above solution for identical arrays.

If each array is treated individually, then the failure rate of each array will be overestimated; board failures due to a double failure on one array are not figured into the repair frequency of the other arrays. This gives a way to obtain an upper bound on the board failure rate. For N arrays on the board:

$$(10) \quad \lambda_{\text{bupper}} = L\lambda_r + \sum_{i=1}^N (K_i + 1)G_i\lambda_r / \text{FIR}(L, L_i, G_i)$$

$$\text{Where } \text{FIR}(l, k, g) = \frac{1 + 2k + 1 + \sqrt{(1 + 2k + 1) + 4k(g-1)(k+1)}}{2k}$$

A lower bound on the board failure rate can be obtained by assuming the repair frequency for each array equals the upper bound of the failure rate of the board.

$$(11) \quad \lambda_{\text{blower}} = L\lambda_r + \sum_{i=1}^N (K_i + 1)G_i\lambda_r / \text{FIR}(\lambda_{\text{bupper}}/\lambda_r, L_i, G_i)$$

To illustrate this process, consider a board with L=100 and two arrays of 33 + spare RAMs. For this example we will use identical failure rates of the RAMs of .2 FPM (fails per million hours). Identical arrays and failure rates were picked for this example in order to compare the exact solution using equations (6) and (7) with the upper and lower bounds.

$$\text{FIR}(l, k, g) = \text{FIR}(100, 33, 2) = 5.26$$

$$\text{Fib} = .2(100 + 66/5.26) = 22.5 \text{ FPM} \quad (\text{actual failure rate})$$

treating the arrays individually,

$$\text{FIR}(l, k, g) = \text{FIR}(100, 33, 1) = 5.06$$

$$\text{and } \text{Fibupper} = .2*100 + .2 * 33/5.06 + .2 * 33/5.06 = 22.6 \text{ FPM}$$

and for the lower bound,

$$F_{ir}(1,k,g) = F_{ir}(22.6/.2,33,1) = 5.45$$

$$\text{and } F_{lower} = .2*100 + .2 * 33/5.45 + .2 * 33/5.45 = 22.4 \text{ FPM}$$

The spread between upper and lower bound is less than 1% of the total failure rate. This is far more accurate than the calculated failure rates of the individual components. In practice, most boards which have multiple different arrays will also have enough other logic (large L) that the spread between upper and lower bounds is narrow, and the upper bound can be used as the failure rate.

7. Sparing in the Tandem NonStop VLX processor

The NonStop VLX system was introduced in 1986 as the top end of Tandem's line of fault-tolerant business computers [ELEC86]. The system consists of from four to sixteen processors which communicate over a high speed inter-processor bus. Each processor consists of two processor boards plus one or two memory boards. One of the processor boards, the Data Path (DP) board has the ALU and 64 Kbyte cache memory. The second processor board, the Interface (IF) board, contains interfaces to the other processors and the IO bus as well as the 120 bit by 8K word control store. The logic on both boards is implemented with 2000 gate bipolar gate arrays. The caches and control store are built from 64Kbit and 16Kbit high speed CMOS static RAMs. Sparing is used on large arrays on both the DP and IF boards.

The DP board uses sparing for the cache memory. The cache is store through, which allows intermittent RAM failures to be handled by refetching the data from main memory. After an error threshold is exceeded, the microcode decides to invoke the spare. The spare is switched in, and the correct data is faulted in using the soft error mechanism. The array has 32 data RAMs, 8 parity RAMs (nibble parity) and one spare. The remaining logic on the board has a failure rate equivalent to 107 RAMs. Thus, $L=107$, $K=40$ and $G=1$ which gives $F_{ir} = 4.7$.

There is a second small array on the DP for scratchpad register storage. The array has 5 RAMs which are duplicated in order to provide soft error recovery. This is equivalent to a single RAM with one spare, where the failure rate of each equivalent RAM is equal to five times the actual RAM failure rate. For this array, $L=21$, $K=1$ and $G=1$ which gives a failure $F_{ir} = 24$.

Bounds on the total DP board failure rate are determined using equations (10) and (11) for the two spared arrays. The difference between the bounds amounts to about a 1% error. The analysis predicts 31% fewer failures on the DP board due to sparing.

The IF board uses sparing in the control store. The control store design is itself unusual in that there are two identical copies which are accessed beginning on alternate cycles. By spreading each RAM access over two machine cycles, a faster cycle time can be used, and performance is improved. This design also tolerates soft errors (with

some performance penalty) by switching to the alternate bank. The microcode switches in the spare once a soft error threshold has been exceeded.

The control store is actually four different arrays; an additional pair of Entry Control Stores (ECSs) addressed by the macroinstruction holds the first microcode line of each macroinstruction. Each of the four arrays has 15 RAMs plus a spare, but due to control logic limitations, the ECS spares cannot be controlled independently of the main control store. As a result, the configuration is equivalent to two groups of 31 + spare RAM equivalents, where the failure rate of each RAM equivalent equals the sum of the ECS RAM failure rate and the main control store RAM failure rate. Then for the IF board, $L=44.4$, $K=15$ and $G=2$. From equation (6), $FIR = 5.8$. The IF board failure rate improved by 47% due to sparing.

8. Conclusions

In the past, standby redundancy has been considered as a way to improve availability or expected mission times of fault tolerant computers. This paper has considered the same type of redundancy as a way to improve failure rates of static RAM arrays. New analysis methods were developed to predict the failure rates of such arrays. Finally, the use of sparing in the NonStop VLX has shown it to be a viable approach to significantly improving repair rates in a commercial product. It is expected that many other systems could similarly benefit from incorporation of static RAM sparing.

9. References

- [BOUR69] Bouricius, W.G., W.C. Carter and P.R. Schneider, "Reliability Modeling Techniques for Self-Repairing Computer Systems", from Proc. ACM Annual Conference, pp 295-309, 1969.
- [DHIL81] Dhillon, B.S. and C. Singh, "Engineering Reliability", New York, John Wiley & Sons, pp 322-325, 1981.
- [ELEC86] "Tandem Makes a Good Thing Better", Electronics, pp 34-38, April 14, 1986.
- [FLAN86] Flannagan, S. et al, "Two 64K CMOS SRAMs with 13ns Access Time", ISSCC Digest of Technical Papers, pp 208-209, Feb, 1986.
- [GOYA86] Goyal, A., et al, "The System Availability Estimator", Proc. 16th Fault Tolerant Computing Symposium, pp 84-89, July, 1986.
- [GRAY85] "Why Do Computers Stop and What Can We Do About It?", Tandem Technical Report TR85.7, Cupertino, CA, 1985.
- [GROS81] Grosspietsch, K.E., J. Kaiser and E. Nett, "A Dynamic Stand-by System for Random Access Memories", Proc. 11th Fault Tolerant Computing Symposium, pp 268-270, June, 1981.

[HARD81] Hardee, K. and R. Sud, "A Fault-Tolerant 30 ns/375mw 16Kx1 NMOS Static RAM", IEEE Journal of Solid-State Circuits, pp 435-443, Oct. 1981.

[NG80] NG, Y., and A. Avizienis, "A Unified Reliability Model for Fault-Tolerant Computers", from IEEE Trans Computers, pp 1002-1011, Nov., 1980.

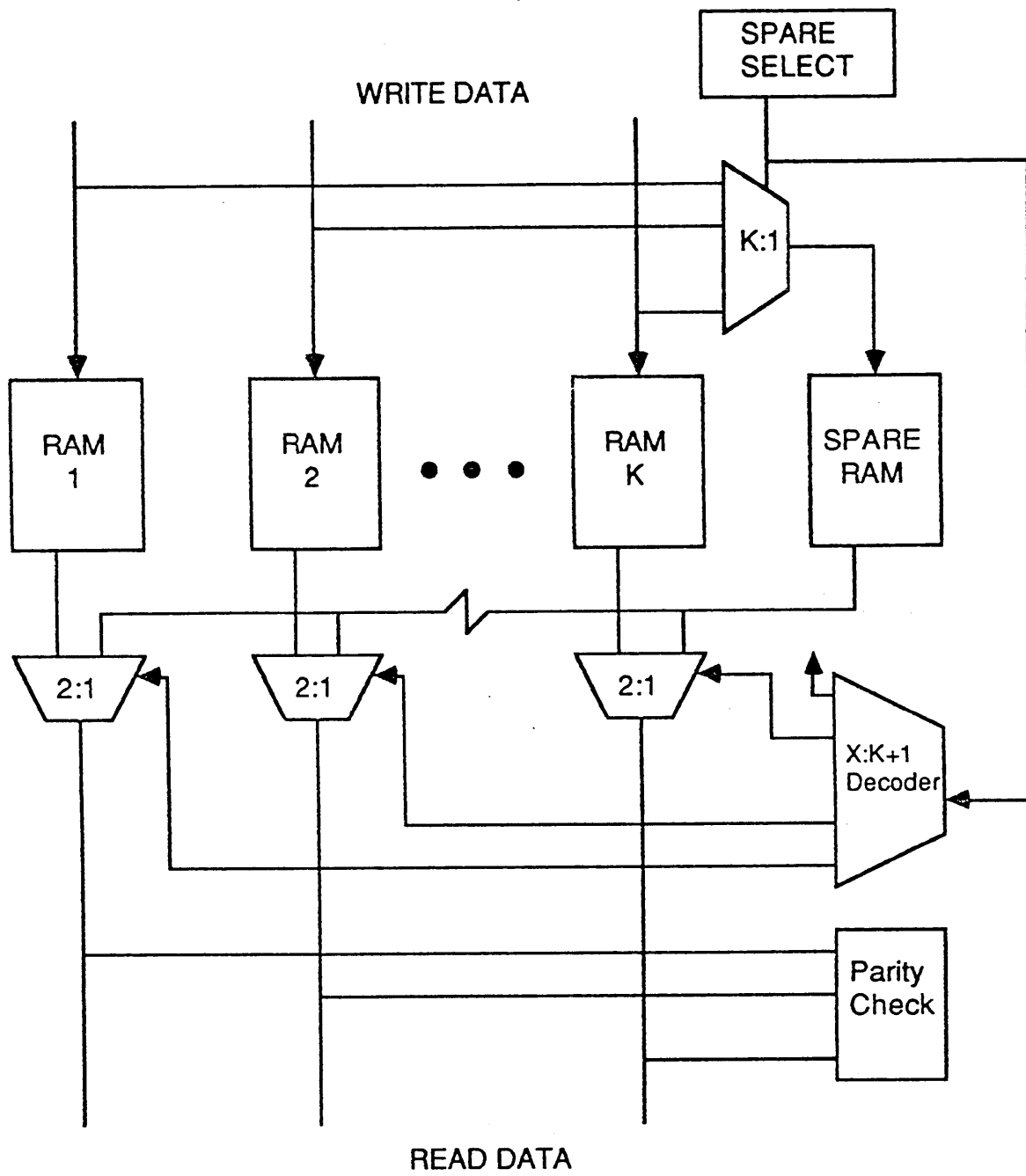


Figure 1. Block diagram of RAM sparing.

EFFECTIVE RAM FAILURE RATE IMPROVEMENT
 FOR SPARED DATA RAMS
 WITH 64 DATA RAMS (+ PARITY + SPARES)

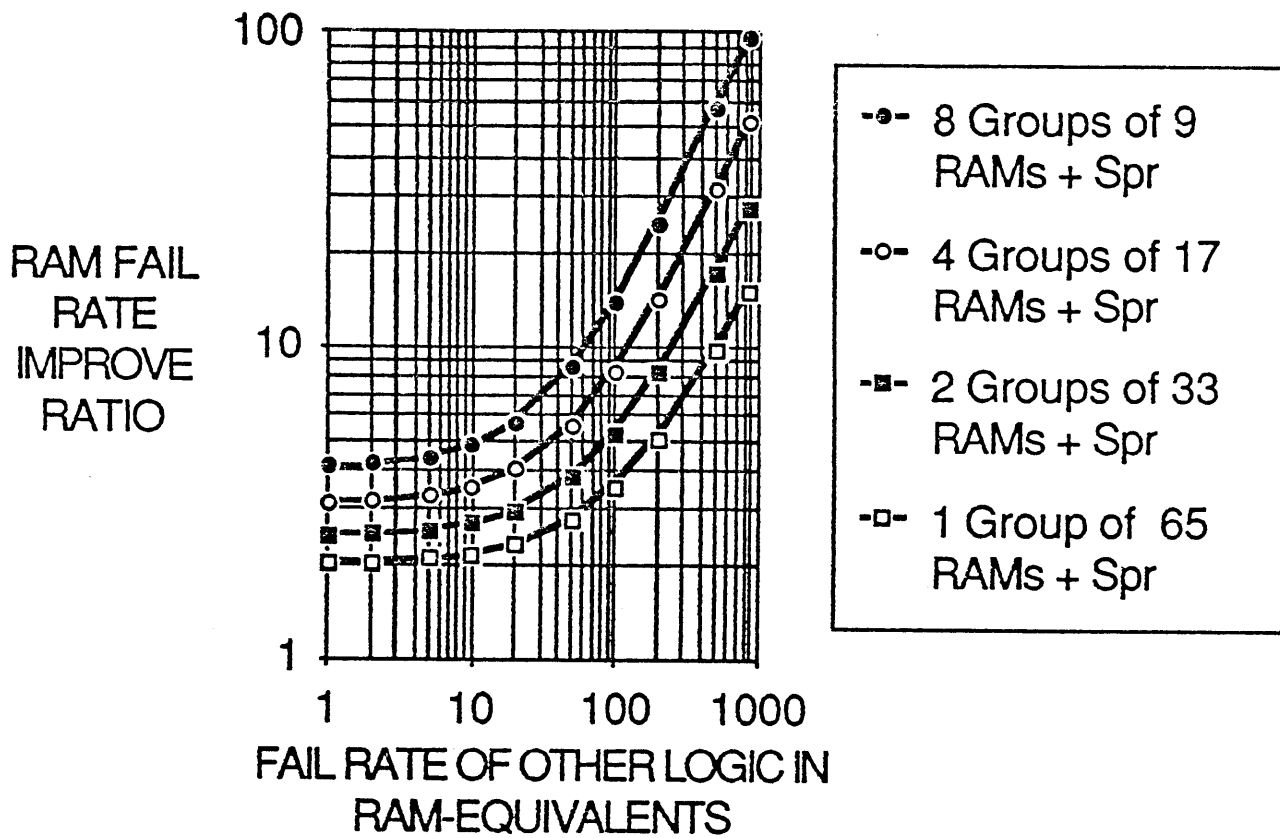


Figure 2a. FIR versus L for 64 data RAMs plus parity in 1,2,4 or 8 spared groups.

EFFECTIVE RAM FAILURE RATE IMPROVEMENT
 FOR SPARED DATA RAMS
 WITH 32 DATA RAMS (+ PARITY + SPARES)

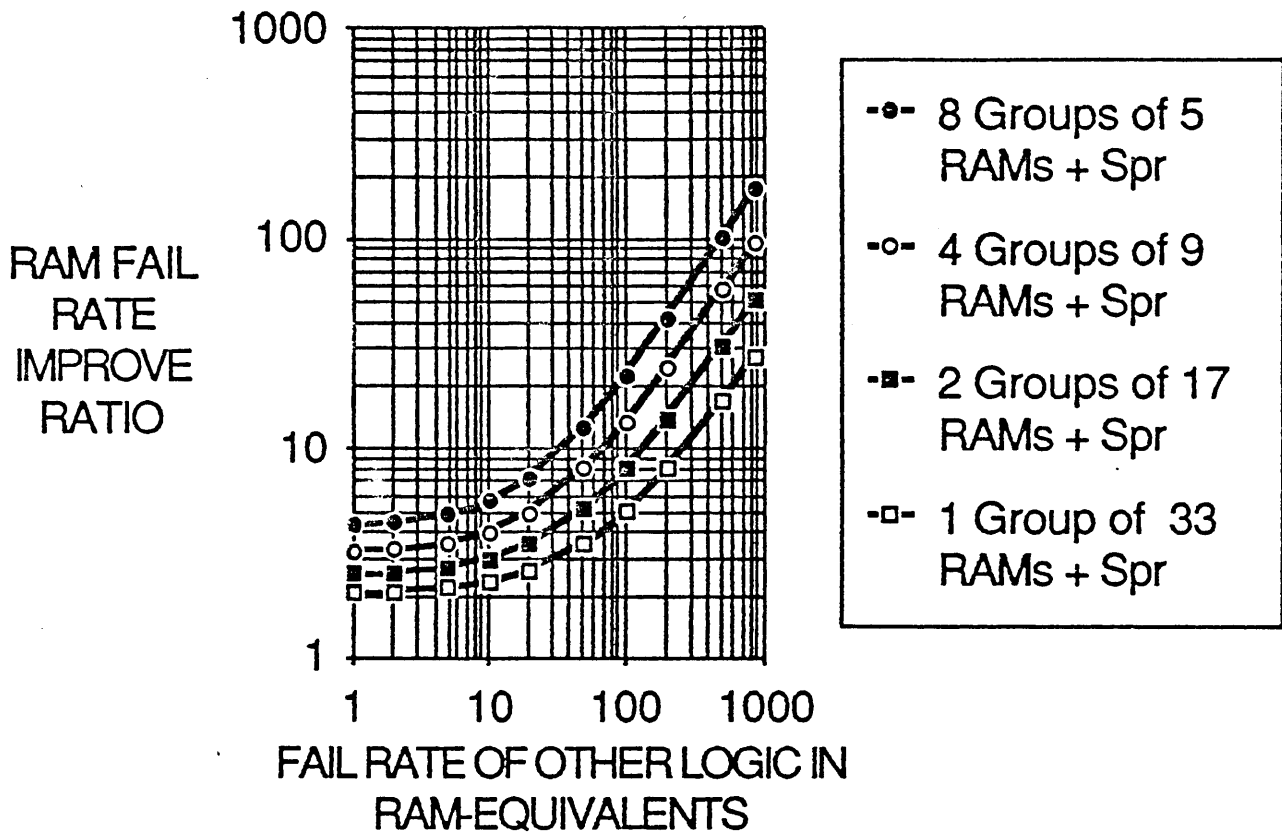


Figure 2b. FIR versus L for 32 data RAMs plus parity in 1,2,4 or 8 spared groups.

EFFECTIVE RAM FAILURE RATE IMPROVEMENT
 FOR SPARED DATA RAMS
 WITH 16 DATA RAMS (+ PARITY + SPARES)

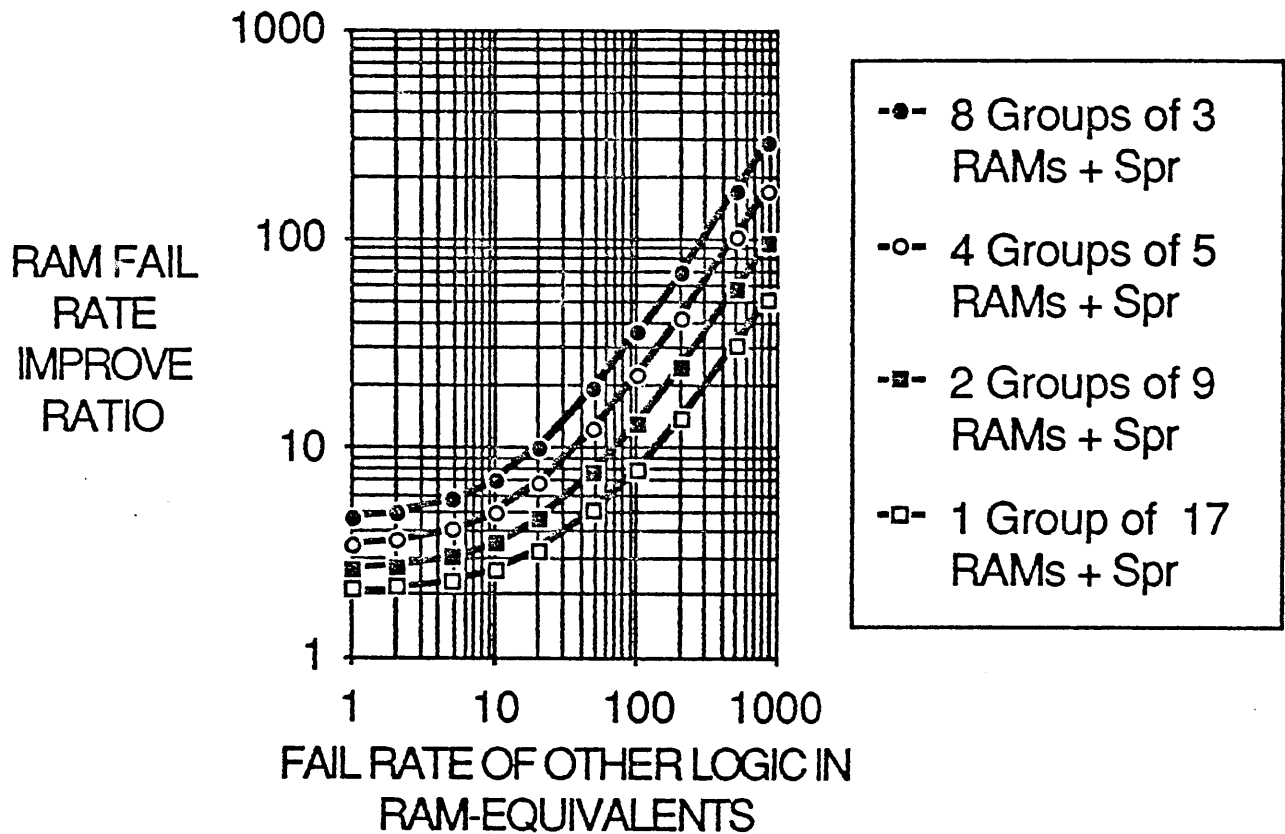


Figure 2c. FIR versus L for 16 data RAMs plus parity in 1,2,4 or 8 spared groups.

EFFECTIVE BOARD FAILURE RATE IMPROVEMENT
 FOR SPARED DATA RAMS
 WITH 64 DATA RAMS (+ PARITY + SPARES)

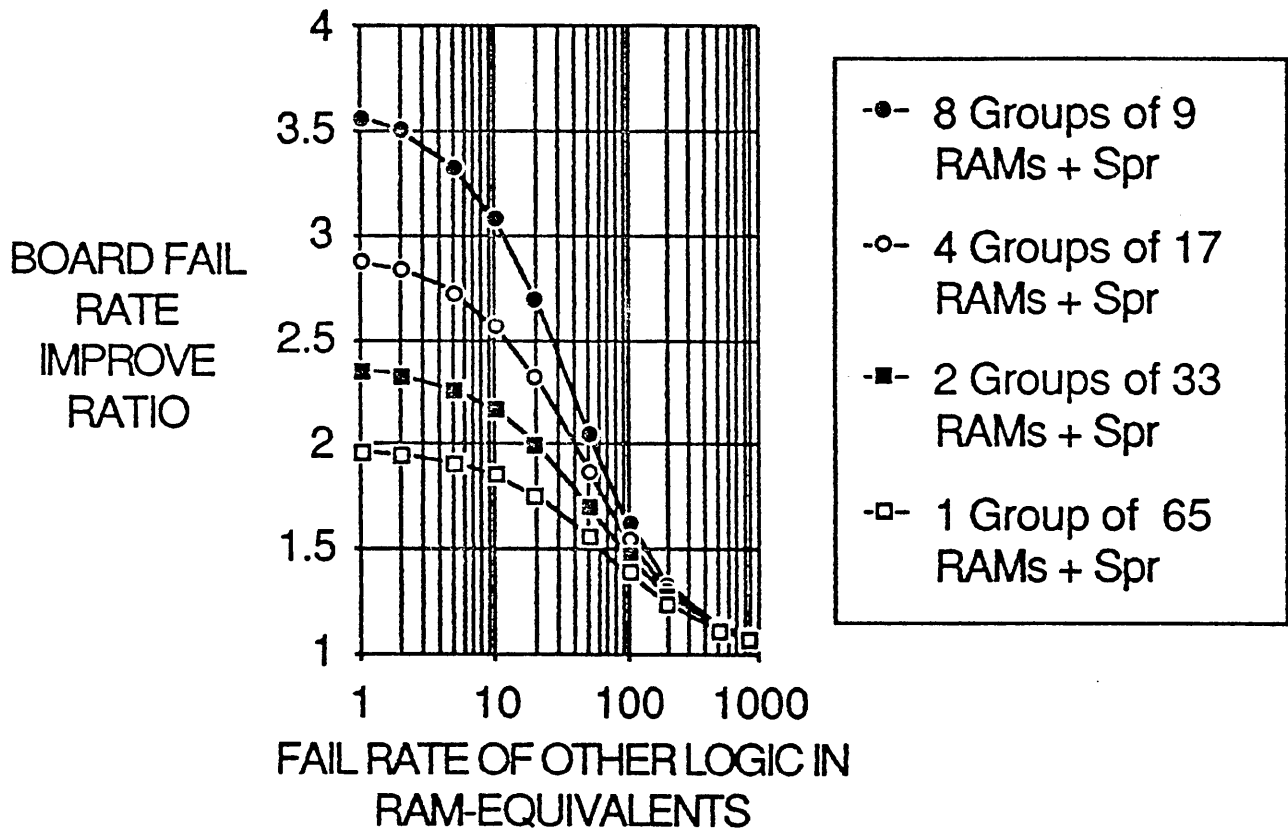


Figure 3a. F1b versus L for 64 data RAMs plus parity in 1,2,4 or 8 spared groups.

EFFECTIVE BOARD FAILURE RATE IMPROVEMENT
 FOR SPARED DATA RAMS
 WITH 32 DATA RAMS (+ PARITY + SPARES)

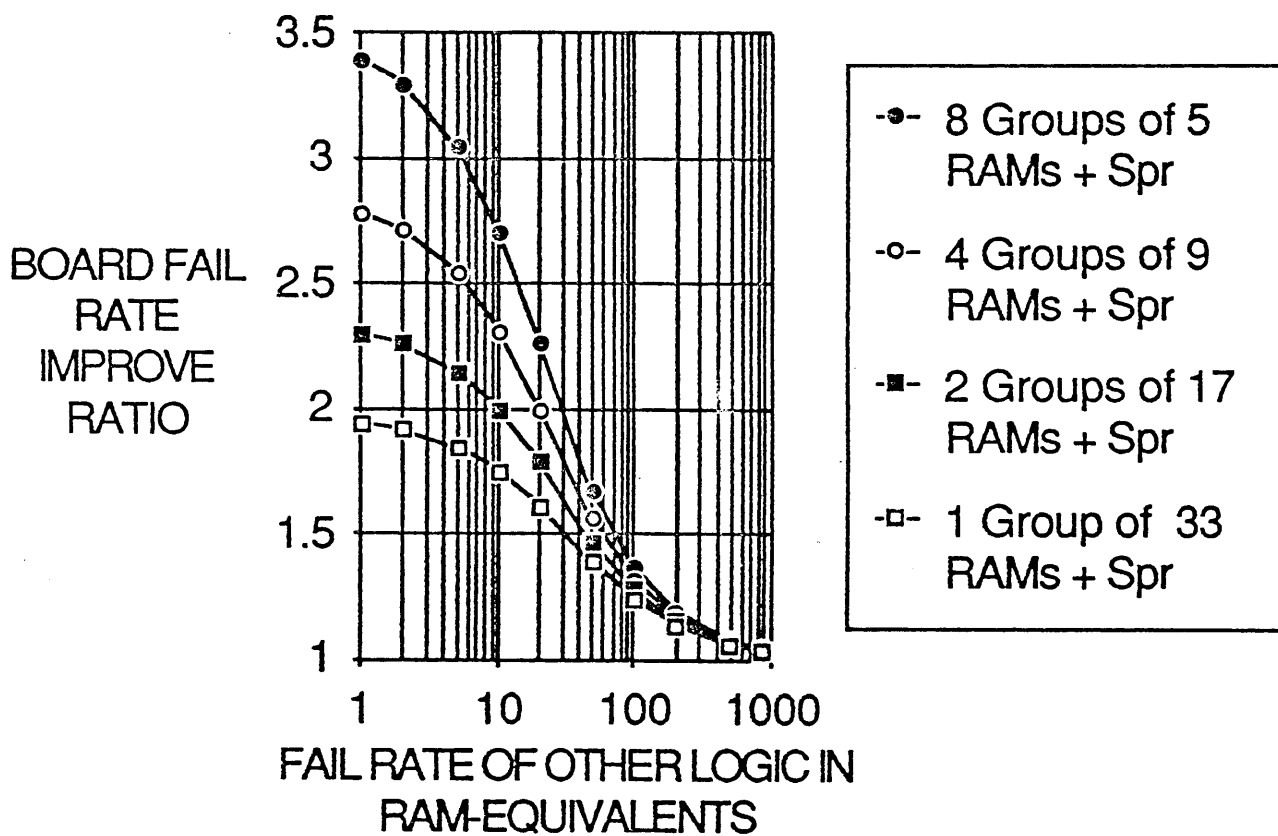


Figure 3b. F1b versus L for 32 data RAMs plus parity in 1,2,4 or 8 spared groups.

Appendix A - BASIC Program to Compute Failure Rates

Following is a simple BASIC program which can be used to compute failure rates on boards with spared RAMs.

The program computes an exact solution for the failure rate if the board contains a single spared array composed of one or more spared groups. It computes an upper bound on the failure rate if the board contains more than one array. The upper bound is usually a good enough approximation unless there is almost nothing else on the board other than the RAMs. When in doubt, the lower bound can be easily computed by rerunning the program and inputting the upper bound at the request for "Non-RAM fail rate in FPM".

A recent version of the program will normally reside on

\TSB.\$ENGR01.RAM.SPARES

secured for NETDUP access.

```

4  print ""
5  Print "RAM SPARE MTBF PROGRAM - R. Horst, 1/7/87"
6  print ""
7  print "Type control-Y to exit"
9  print ""

10 input "Non-RAM fail rate in FPM",Y1
15 input "Number of arrays on this board",NA

20 Ytot=Y1
25 Yold=Y1
30 For i= 1 to NA

35  Print ""
40  Print "For array";i;":"
45  input "RAM fail rate in FPM",Yr
50  L = Y1/Yr

55  input "# of groups of Spared RAMs",g
60  input "RAMs per group excluding spare",k

65  print "L=";L


70  a= k
75  b= -L-1-2*k
80  c= (1-g)*(K+1)
85  FI = (-b + (b*b -4 * a * c)**.5)/(2 * a)

130 print "Array";i;" RAM fail rate imrovment=",FI
140 F1b= (k*g + L)/(L + (k+1)*g/FI ) !Board fail rate improvement
150 Yarr =Yr*G*(K+1)/FI
160 print "Array";i;"Failure rate ",Yarr;"FPM"
180 Ytot=Ytot+Yarr
190 Yold=Yold+g*K*Yr
200 next i

205 Print ""
207 Print "Total RAM failure rate",Ytot-Y1;"FPM"
210 Print "***** Board Fail Rate ",Ytot;"FPM *****"
220 print "Improvement in board MTBF due to sparing = ",Yold/Ytot

500 print " "
510 goto 9

```

Distributed by
 **TANDEM** COMPUTERS
Corporate Information Center
19333 Vallco Parkway MS3-07
Cupertino, CA 95014-2599

