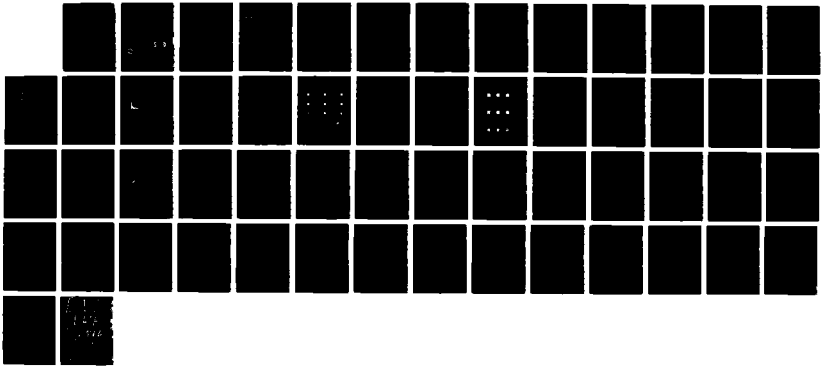
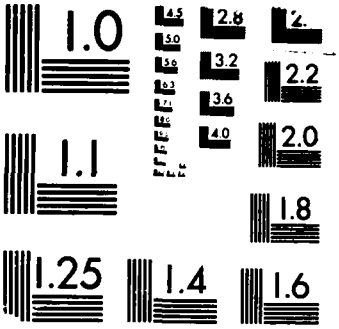
AD-A193 521 SYSTOLIC ARRAY FAULT TOLERANCE PERFORMANCE ANALYSIS(U) 1/1
NAVAL UNDERWATER SYSTEMS CENTER NEW LONDON CT NEW
LONDON LAB T C CHOINSKI ET AL. 05 APR 88 NUSC-TR-8221
UNCLASSIFIED F/G 12/5 NL





MICROCOPY RESOLUTION TEST CHART

(4)

NUSC Technical Report 8221
5 April 1988

Systolic Array Fault Tolerance Performance Analysis

T. C. Choinski
M. H. Leonhardt
Submarine Sonar Department

Naval Underwater Systems Center
Newport, Rhode Island / New London, Connecticut

PREFACE

This report was prepared under NUSC Project No. A75044 under the sponsorship of the Naval Sea Systems Command (NAVSEA, PMS-412).

The authors would like to thank R. Davis and F. Criner (Code 411) for providing helpful suggestions during the preparation of this report.

The technical reviewer for this report was R. Davis (Code 411).

REVIEWED AND APPROVED: 5 April 1988

F. V. Kingsbury

F. J. KINGSBURY
HEAD: SUBMARINE SONAR DEPARTMENT

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

A193 521

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NUSC Technical Report 8221	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Underwater Systems Center 6b. OFFICE SYMBOL (If applicable) 2151	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) New London Laboratory New London, CT 06320	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Sea Systems Command 8b. OFFICE SYMBOL (If applicable) PMS-412	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Washington, DC 20362	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. A75044 / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

SYSTOLIC ARRAY FAULT TOLERANCE PERFORMANCE ANALYSIS

12. PERSONAL AUTHOR(S)
T. C. Choinski and M. H. Leonhardt

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
summary	FROM ____ TO ____	1988 April 5	62

16. SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Adaptive Beamforming; Reliability; Fault Tolerance; Systolic Arrays; Reconfigurable Architectures; MTBF

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

The realiability performance of six different systolic array fault tolerance techniques are determined and compared in terms of mean time between failure (MTBF). The six techniques include redundant arrays, companion processors, sequential row elimination (SRE), alternate row and column elimination (ARCE), virtual arrays, and tree based architectures. The results demonstrate the importance of the switching function failure rate in achieving theoretical capability. Virtual arrays were found to have the best theoretical potential for improving the reliability of systolic arrays when high throughput is required. All techniques provided comparable performance for low throughput levels. The potential application of graceful degradation techniques to the minimum variance distortionless response (MVDR) adaptive beamforming algorithm is discussed. keywords

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ OTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL T. C. Choinski	22b. TELEPHONE (Include Area Code) (203) 440-5391 22c. OFFICE SYMBOL 2151

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SYSTOLIC ARRAY FAULT TOLERANCE PERFORMANCE ANALYSIS

INTRODUCTION

Systolic array technology offers increased throughput using algorithm-efficient architectures implemented with state-of-the-art very large scale integration (VLSI) technology (references 1-4). Several architectural issues must be addressed before this technology can transition from the laboratory environment to systems in the field. One of these issues is reliability. Since the systolic array approach is hardware intensive, increased throughput is accompanied by deteriorated reliability performance. Therefore, fault tolerance design concepts must be incorporated into systolic array architectures to restore the lost reliability performance.

The application of fault tolerance techniques involves four basic areas: error detection, error correction, fault isolation, and system reconfiguration. A detailed discussion of all four aspects of fault tolerance is beyond the scope of this report; therefore, the primary focus will be on system reconfiguration performance. The objective of the study reported is to identify the reconfiguration techniques which will provide the most dependable performance for systolic array architectures. Once identified, these fault tolerant architectures can establish the framework for error detection, error correction, and fault isolation. If dependable performance cannot be achieved through reconfiguration, any effort placed in detection, isolation, and correction would be misdirected.

Systolic array reliability performance will be compared for several fault tolerance techniques by providing estimates of the mean time between failure (MTBF), a universal reliability performance attribute. Spatially and temporally fault tolerant techniques will be examined. Spatial fault tolerant techniques use hardware reserves to compensate for component failures. Temporal fault tolerance techniques exploit graceful degradation in algorithm execution time and/or system performance to offset component failures and, thus, mitigate the initial hardware reserve level.

The MTBF is used as the performance measure since it is a distinctive requirement for military systems and is used to estimate availability, logistics, and life cycle costs. Improvement in reliability is measured through the comparison of the MTBF of the fault tolerant systolic array

and the MTBF of the systolic array without a fault tolerant architecture. The acquired improvement will be compared to the additional reserve hardware required for a given fault tolerant approach. For example, the enhancement for 25%, 50%, 75%, and 100% redundancy levels will be assessed where appropriate. This comparison will help assess the cost effectiveness of a fault tolerant architecture.

SYSTOLIC ARRAY RELIABILITY

The reliability of a circuit component is the probability of failure-free performance for a required function, under stated conditions, for a given period of time. Since it is impossible to categorically state that a component will perform properly, the most that can be done is to calculate or measure a probability for successful operation. Reliability screening procedures can reduce the instantaneous failure rate, or hazard rate, of a component to a constant value over its useful lifetime so its overall reliability can be expressed by the following exponential function (references 5 and 6):

$$R(t) = e^{-\lambda_c t},$$

where

$R(t)$ = the probability that a component will function properly (component reliability),
t = time, and
λ_c = component instantaneous failure rate (hazard rate).

The reliability performance of a component is commonly expressed in terms of its mean time between failure. The MTBF of a component can be established by integrating the reliability function, R(t), over all time so that

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

Systolic arrays reach high computational rates because algorithms can be projected onto interconnected arrays of processing elements. The maximum theoretical throughput is linearly related to the quantity of processing elements in the systolic array. Theoretically, N processing elements, combined in an array, will provide N times the throughput of a single processing element. By definition, the systolic array method is hardware intensive.

The reliability of a systolic array designed without fault tolerant considerations depends on the probability of each and every processing element remaining functional. This condition generates a serial reliability model. The reliability of a systolic array having identical processing elements is

$$R_{Systolic}(t) = e^{-\lambda_{PE_1}t}\, e^{-\lambda_{PE_2}t} \cdots e^{-\lambda_{PE_N}t}$$

$$= e^{-N\lambda_{PE}t} \quad ; \text{ for } \lambda_{PE_1} = \lambda_{PE_2} \cdots = \lambda_{PE_N}$$

$$= R^N.$$

Likewise, the MTBF of a systolic array with identical processing elements is

$$MTBF_{Systolic} = \int_0^\infty e^{-N\lambda_{PE}t}\, dt = (N\lambda_{PE})^{-1} = M.$$

These results show that the MTBF of a systolic array without fault tolerant architectural considerations is inversely proportional to its maximum theoretical throughput or N. An increase in computational speed will be offset by a corresponding decrease in reliability, when expressed in terms of MTBF. Therefore, reliability and, consequently, fault tolerance should be a major consideration in the design of embedded systolic arrays. Ideally, the enhancement provided by fault tolerant techniques should be proportional to N to offset the degradation in MTBF. Furthermore, the improvement needed in MTBF would actually be greater than N when less than 100% computational efficiencies are considered (references 7 and 8).

FAULT TOLERANT SYSTOLIC ARRAYS

The object of the systolic array fault tolerance approaches described in this report is to fail-operational (reference 9). A fail-operational mode will ensure that the systolic array operates correctly in the presence of faults (component failures). Thus, component failures will not precipitate an unacceptable reduction in system performance. For example, an application may require the systolic array to be capable of performing matrix operations even though a processing element(s) has failed. There are two ways a systolic array can fail operational: (1) to encounter the failure and provide the maximum required performance and (2) to encounter the failure with an acceptable level of graceful degradation in performance. Both techniques

necessitate a hardware reserve and/or architectural reconfiguration capability.

Spatial redundancy and temporal redundancy are two generic approaches for fault tolerance. Spatial redundancy capitalizes on additional hardware to provide the same throughput (maximum array size) when a failure occurs. Temporal redundancy eases the required throughput when a failure occurs and provides graceful degradation in system performance. Either the same amount of information is processed in a longer time period, or less information is processed in the same time period.

Spatially redundant methods yield optimal system performance, but at the expense of expanded physical size and monetary cost. The spatial redundancy methods maintain system performance using oversized arrays. Six spatial redundancy methods will be discussed in this report. Four methods are generic and can be used for any systolic array application and the other two methods are based on specific architectural configurations. The first and most obvious method, redundant arrays, is the inclusion of one or more alternate systolic arrays in the system. If the primary array fails an alternate array is switched into the system. The second method, companion processors, also uses a fairly straightforward concept by providing redundancy at the processing element level instead of the array level. Each processing element has one companion which can be interposed when one element has failed. The third and fourth methods provide redundancy by switching a group of failed processing elements with a group of functioning processing elements. The two methods identified in literature have been referred to as sequential row elimination (SRE) and alternate row and column elimination (ARCE), which is contingent upon a rectangular systolic array configuration. The fifth method uses reserve hardware with dynamic virtual channel reconfiguration. The virtual channel method is the most sophisticated and complex approach to systolic array fault tolerance. The final method provides spatial redundancy specifically for tree architectures.

Redundant hardware fault tolerance replicates common hardware components. The spatial fault tolerance approach uses hardware, i.e., circuit cards, that are identical in design and function as well as construction. This requirement is especially important in systems where spare parts, life cycle costs and logistics is of the utmost importance.

One systolic array application, adaptive beamforming, is novel in that an inherent temporal fault tolerance concept is built into the algorithm. The minimum variance distortionless response (MVDR) algorithm exemplifies this concept. The MVDR algorithm allows for beamforming

subarrays instead of hydrophones. Therefore, when a component failure occurs and reduces the available systolic array size, all the hydrophone inputs can still be processed by making the subarrays larger. Larger subarrays will reduce the covariance matrices that have to be processed to form the adaptive weights. The hydrophone array will provide the same array gain but the number of interference sources that can be cancelled will be smaller. Processing time and array gain can be maintained at the expense of the reduction in degrees of freedom. For large sonar arrays this approach may be beneficial. Like spatial fault tolerance, the subarray fault tolerance approach does require the ability to reconfigure the array. In fact, fault tolerance through graceful degradation, necessitates the use of one of the previously discussed spatially redundant techniques. SRE, ARCE, virtual arrays, or tree architectures can be used in a graceful degradation mode. However, the application of graceful degradation would reduce hardware reserve requirements significantly. The subarray method is the only temporally redundant approach that will be discussed in this report because of its specific application to adaptive beamforming. Proofs and analysis programs for each approach are shown in the appendices (A and B).

SPATIAL FAULT TOLERANCE

REDUNDANT ARRAYS

The redundant array method is illustrated in figure 1. This spatially redundant method includes the addition of an alternate systolic array in case the primary array fails. Accordingly, a minimum of 100% hardware redundancy is required. If one alternate array is used the reliability is

$$R_{Redundant}(t) = R^{2N} + 2R^{N} (1 - R^{N}),$$

with an associated MTBF of

$$MTBF_{Redundant} = M/2 + 2M - 2M/2 = 1.5\ M.$$

The redundant array method offers a fixed improvement in MTBF of 50% over the systolic array MTBF, while doubling the total hardware required. Ideally, the improvement should increase linearly with the array size, N, thereby negating the diminished systolic array MTBF. A linearly related improvement would yield a reliability performance that is independent of array size and would be a design objective for optimal performance. From a technical standpoint, the redundant array method is neither cost

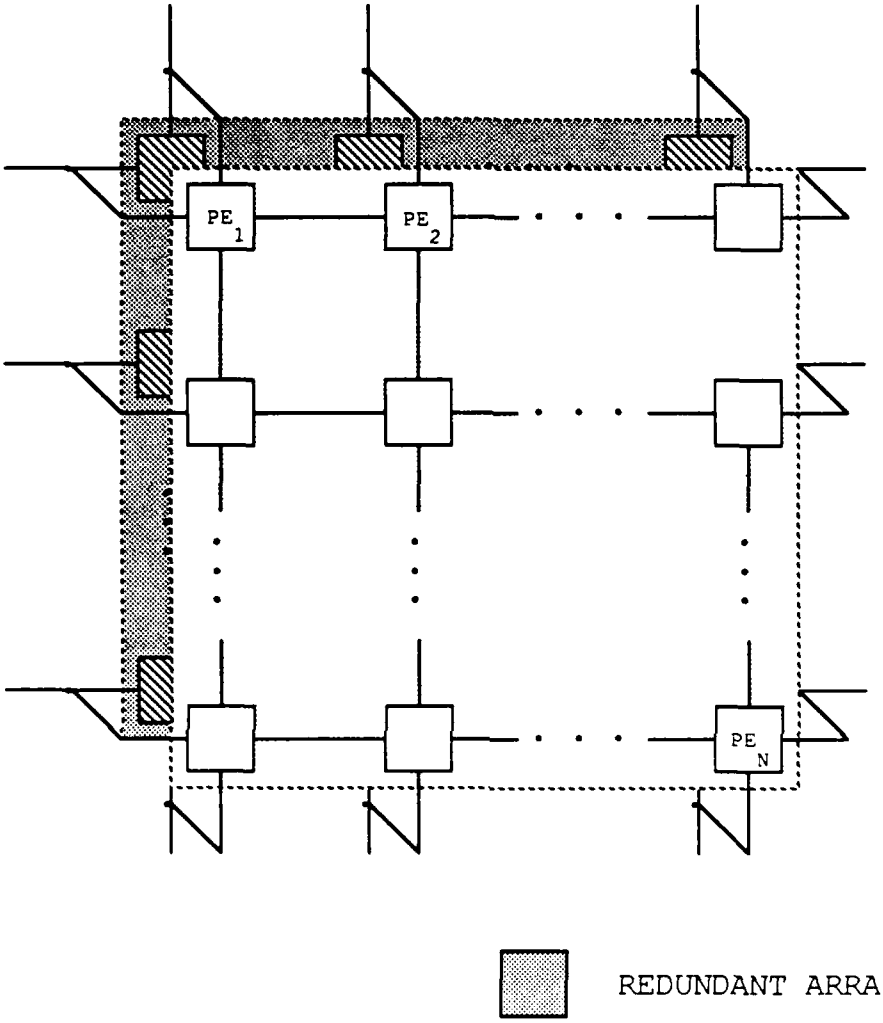


Figure 1. Redundant Array Fault Tolerance

effective nor desirable. The linear reduction in $MTBF_{Systolic}$ due to array size has not been alleviated.

COMPANION PROCESSORS

Like the redundant array method, the companion processor fault tolerance method entails a hardware duplication. However, the companion processor method inserts redundancy at the processing element level rather than at the array level, as depicted in figure 2. If any single processing element fails it is replaced with its companion processor. This fault tolerance method offers the advantage of an elementary switch implementation. The switching function can be fulfilled by tri-state buffers at the output ports of each processing element. This type of switch mechanism would have a low failure rate, since minimal hardware is required to implement it. The disadvantage of this method is that a given reserve processing element can only be used to replace its single companion in the array. Consequently, a systolic array can fail even though reserve processing elements are operational.

The reliability for the companion processor method is

$$R_{Companion}(t) = e^{-N\lambda_{CS}t} \left[R^2 + 2R(1 - R)\right]^N.$$

$$= R^{NF} (2 - R)^N,$$

where

λ_{CS} = failure rate for the companion switch,
F = $(1 + 1/sr)$, a measure of the failure rate of the switch compared to the failure rate of the processing element, and
sr = switch ratio $(\lambda_{PE}/\lambda_{CS})$,

using the binomial theorem (reference 10) to expand the exponential term, the reliability can also be expressed as

$$R_{Companion}(t) = \sum_{i=0}^{N} \binom{N}{i} 2^{(N-i)} (-1)^i R^{(i+NF)},$$

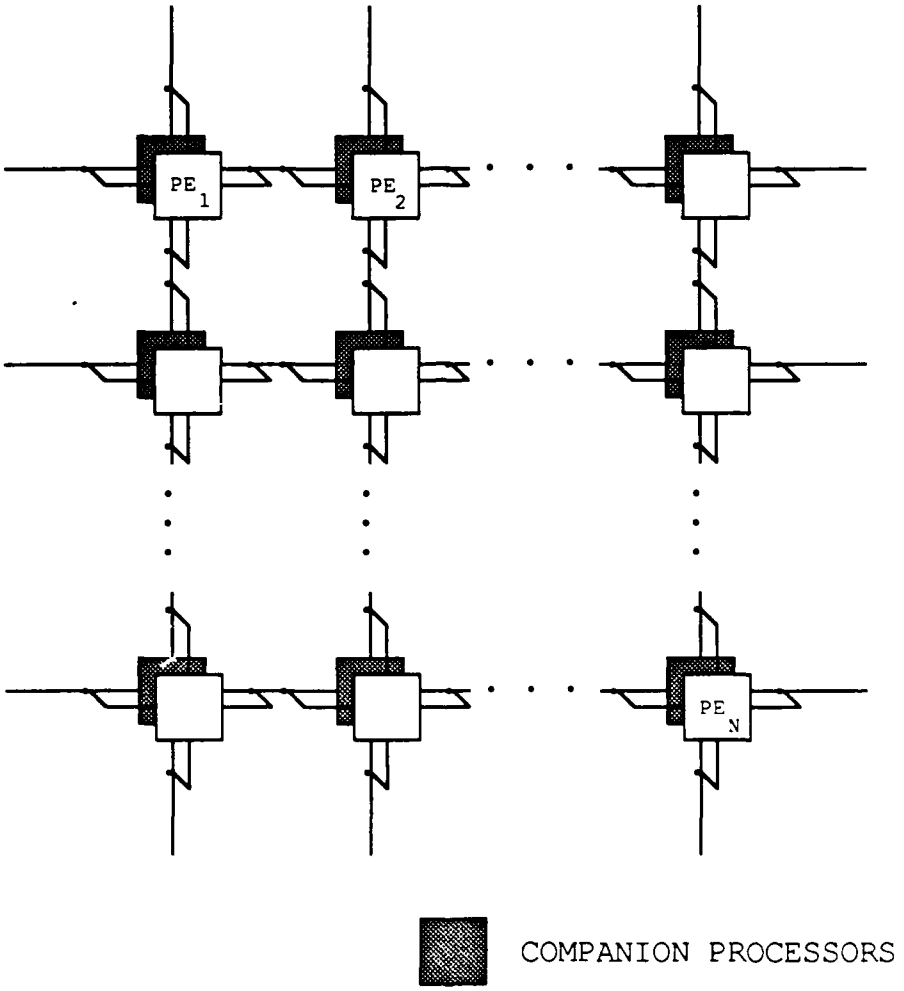


Figure 2. Companion Processor Architecture

where

$$\binom{N}{i} = \frac{N!}{i!\ (N-i)!} .$$

After integrating $R_{Companion}(t)$, the MTBF is found to be

$$MTBF_{Companion} = M \sum_{i=0}^{N} \binom{N}{i} 2^{(N-i)}\ (-1)^{i}\ [N/(i+NF)] .$$

The MTBF for the single systolic array, redundant array, and companion processor fault tolerance methods are sketched in figure 3 in terms of single processing element MTBF versus array size. The companion processor data were generated under the assumption that the switch mechanism is ideal, i.e., the switch failure rate is zero, or F=1.

The companion processor method produces the following improvement for a systolic array:

$$I_{Companion} = \sum_{i=0}^{N} \binom{N}{i} 2^{(N-i)}\ (-1)^{i}\ [N/(i+NF)] .$$

The improvement versus array size is shown for various failure rate ratios in figure 4. $MTBF_{Companion}$ depends on the ratio of the processing element failure rate to the switch failure rate. The improvement was calculated for several failure rate ratios. The outcome shows that when the switch failure rate is comparable to the processing element failure rate, $F \cong 2$, little or no improvement in $MTBF_{Companion}$ is encountered compared to the dual array approach. A switch ratio of 40 or greater, $F \cong 1$, will bring the improvement near the theoretical maximum, which ideally occurs when $\lambda_{CS} = 0$. It is beneficial to note that the maximum theoretical improvement approximates the square root of the number of processing elements. The improvement is better than the redundant array method but still does not provide a linear compensation for the reduction in $MTBF_{Systolic}$ with array size. The hardware is doubled for nearly a $(N)^{1/2}$ improvement in MTBF.

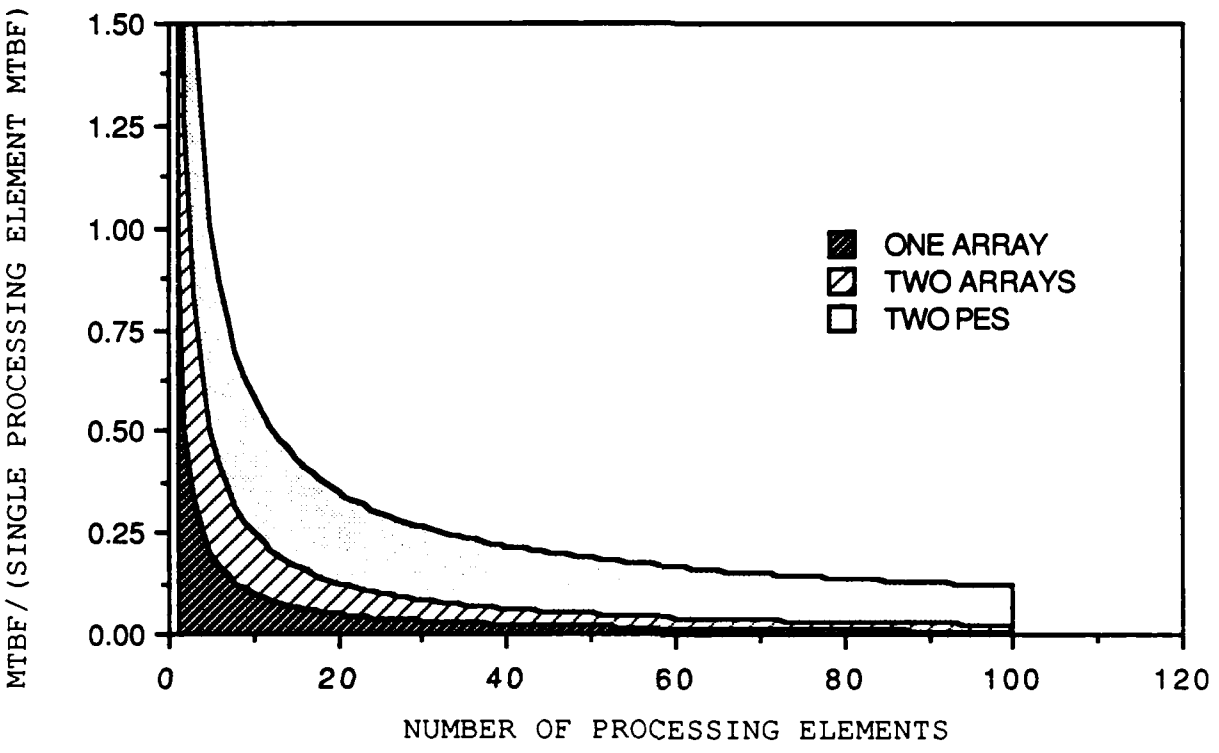


Figure 3. Systolic Array Reliability

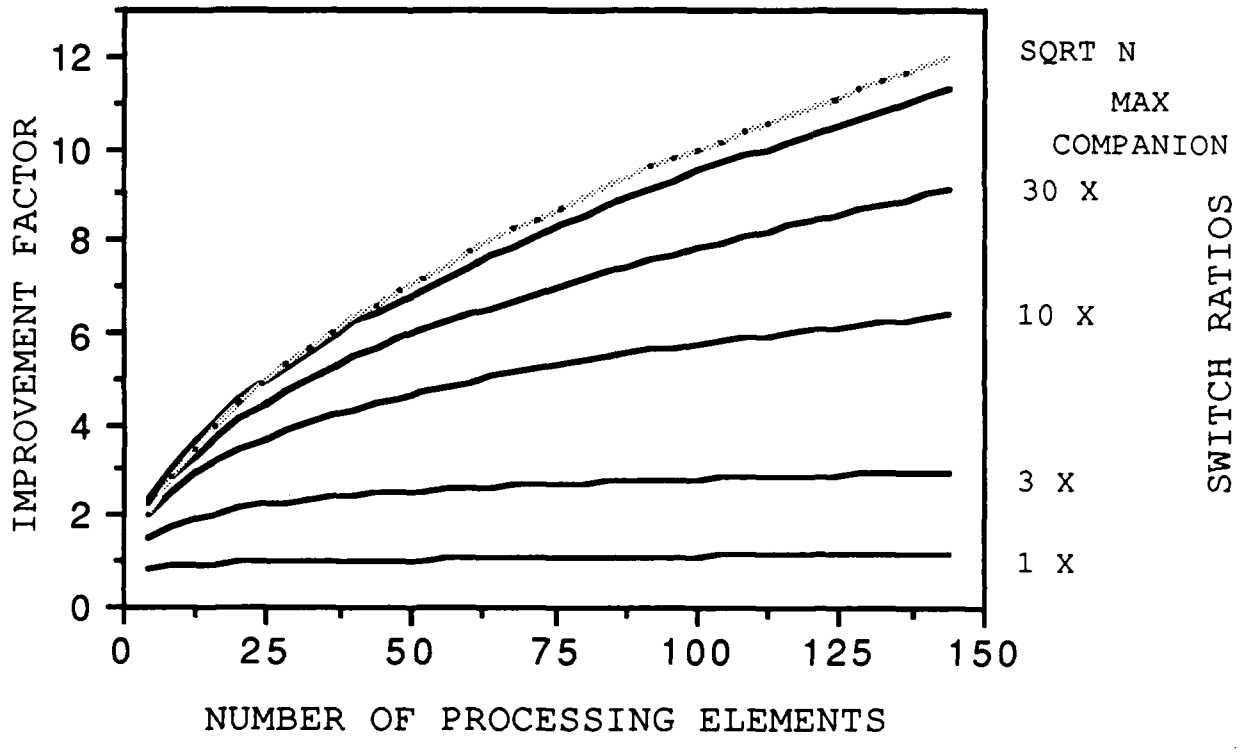


Figure 4. Companion Processor Performance

SEQUENTIAL ROW ELIMINATION (SRE)

Reconfiguration using reserve hardware can occur at the processing element level or at a level that consists of groups of processing elements. Group fault tolerance techniques include the SRE and ARCE methods presented by C. S. Raghavendra (reference 11).

The SRE and ARCE methods achieve fault tolerance by successively removing rows, or rows and columns, in an array that has faulty processors. Programmable switches and interconnections facilitate the removal of a faulty row or column. The SRE method will be addressed in this section as illustrated in figure 5.

The SRE fault tolerance substitutes faulted rows of processing elements with good rows when one processing element in a row has failed. The switching function controls the exclusion of an entire row. For the SRE method the reliability is

$$R(t) = e^{(-n^2/sr)t} \sum_{k=0}^{D} \sum_{i=0}^{k} \frac{\prod_{j=0}^{k-1} A_j}{\prod_{\substack{j=0 \\ j\neq i}}^{k} (A_j - A_i)} e^{-A_i t},$$

where

n = number of elements,

sr = switch ratio ($\lambda_{PE}/\lambda_{switch}$),

$A_j = n(n - j)\lambda$,

$A_i = n(n - i)\lambda$,

D = Maximum # of rows - Minimum # of rows, and

k = counting variable (0 ... D).

The MTBF improvement was determined by integrating the reliability function over time. Figure 6 shows the effect of array size and switch failure rate on MTBF improvement. Square array architectures were postulated. Notice the switches used in this case are affiliated with an entire row rather than one single processing element.

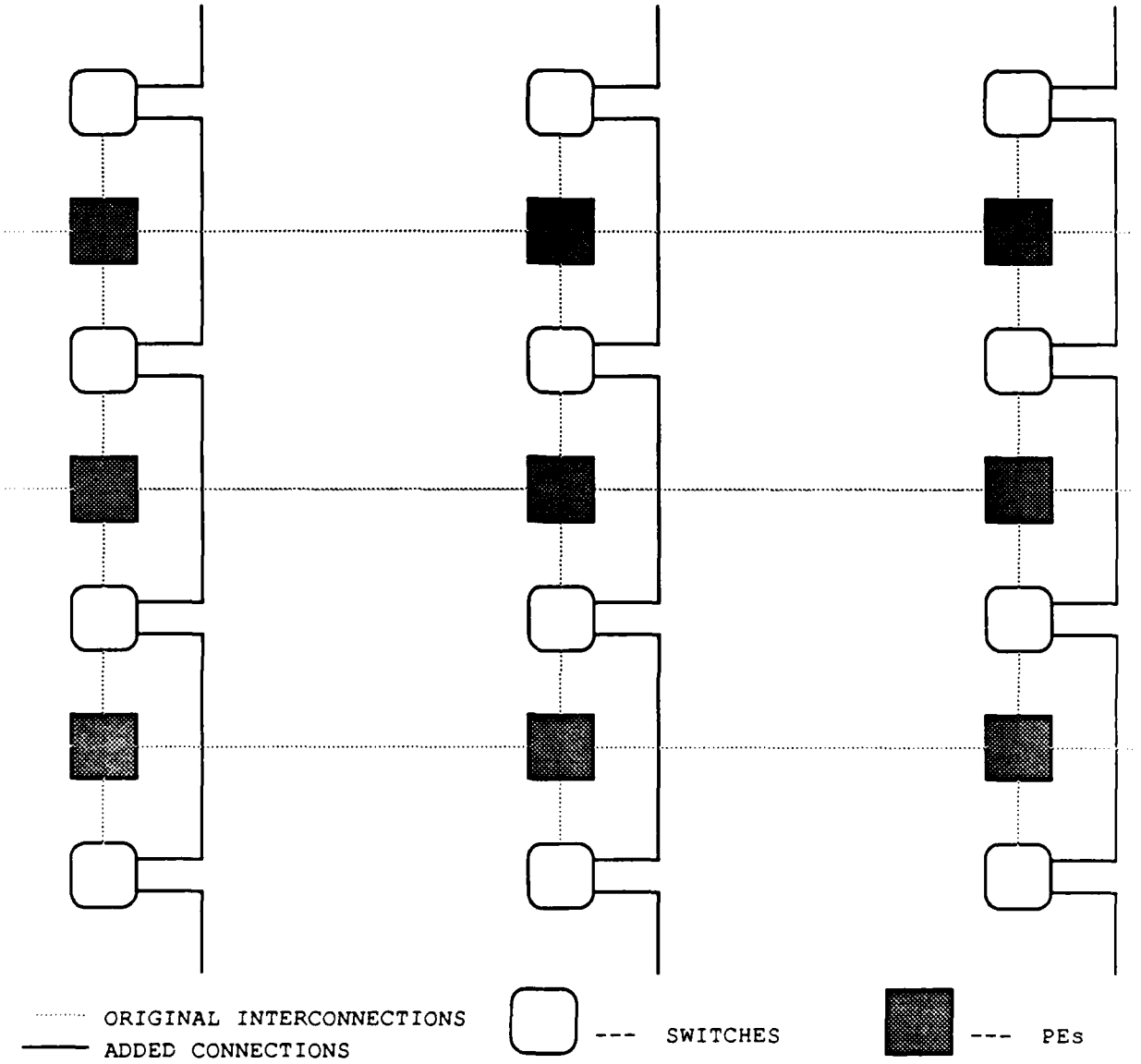


Figure 5. Sequential Row Elimination Architecture

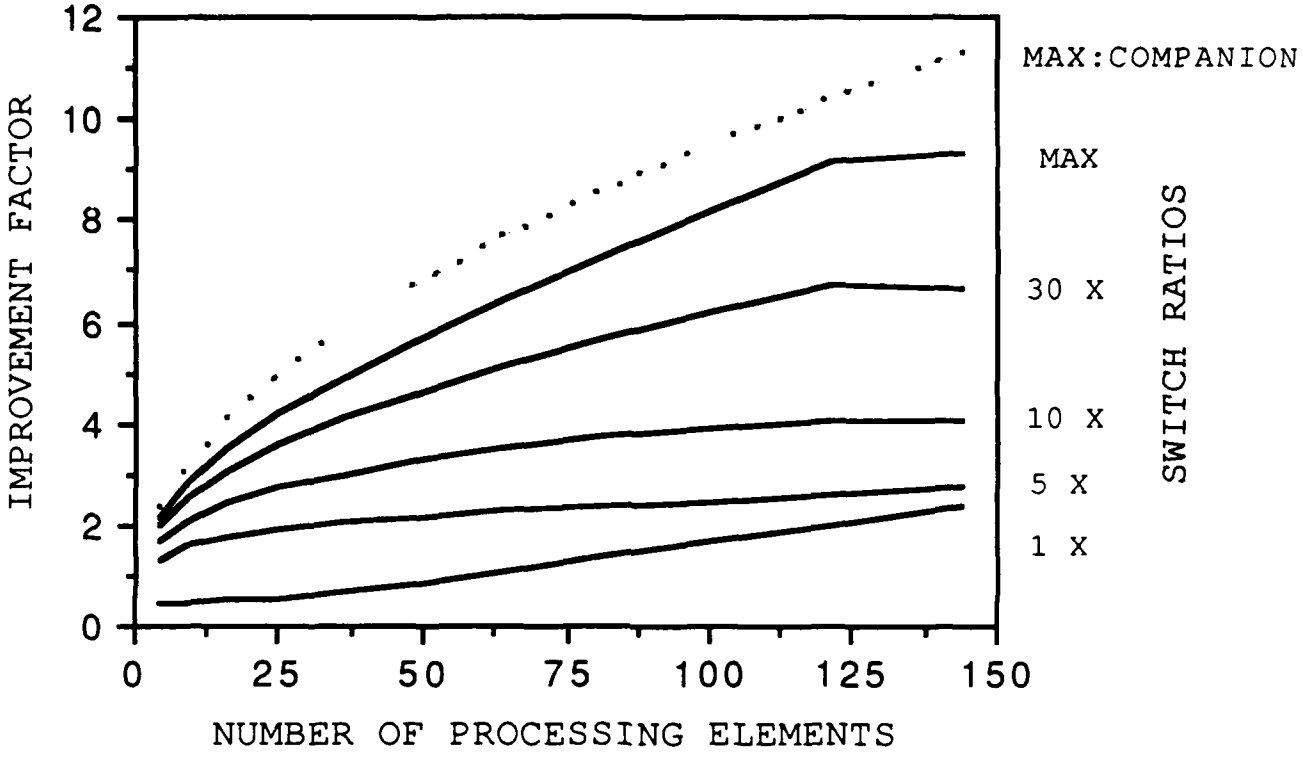


Figure 6. SRE Performance

ALTERNATE ROW AND COLUMN ELIMINATION (ARCE)

The ARCE method switches out a row or a column when a processing element fails. This elimination is based on an alternating removal concept. The hardware is shown in figure 7. The reliability for ARCE is

$$R(t) = e^{(-n^2/sr)t} \sum_{k=0}^{D} \sum_{i=0}^{k} \frac{\prod_{j=0}^{k-1} A_j}{\prod_{\substack{j=0 \\ j\neq i}}^{k} (A_j - A_i)} e^{-A_i t},$$

where

$A_j = (n - \lceil j/2 \rceil)(n - \lfloor j/2 \rfloor)\lambda$

$A_i = (n - \lceil i/2 \rceil)(n - \lfloor i/2 \rfloor)\lambda$, and

$D =$ Maximum # of rows & columns - Minimum # of rows & columns.

Once again, the reliability was integrated over all time to get the improvement in MTBF. Since the arrays were assumed square, the hardware reserve percentages varied. Table 1 shows the results for up to 125% redundancy levels. Switch ratios (sr) from 1 to 100,000 (an ideal switch) are encompassed.

The ARCE method limits architectural configurations to square or rectangular arrays. The companion processor method demonstrates a higher MTBF and is suitable for any architectural configuration.

VIRTUAL CHANNEL FAULT TOLERANCE

The virtual channel technique organizes operational processing elements in a systolic array. Connections between switch ports form virtual channels. Figure 8 portrays the virtual channel method as outlined by H. T. Kung and O. Menzilcioglu (reference 12). Unlike other approaches described in this report, the virtual channel method utilizes all available operational processing elements. The redundant array, SRE, and ARCE fault tolerance methods may actually drop out operational processing elements along with the faulty processing elements, thus, rendering them inactive. The virtual channel technique averts this situation.

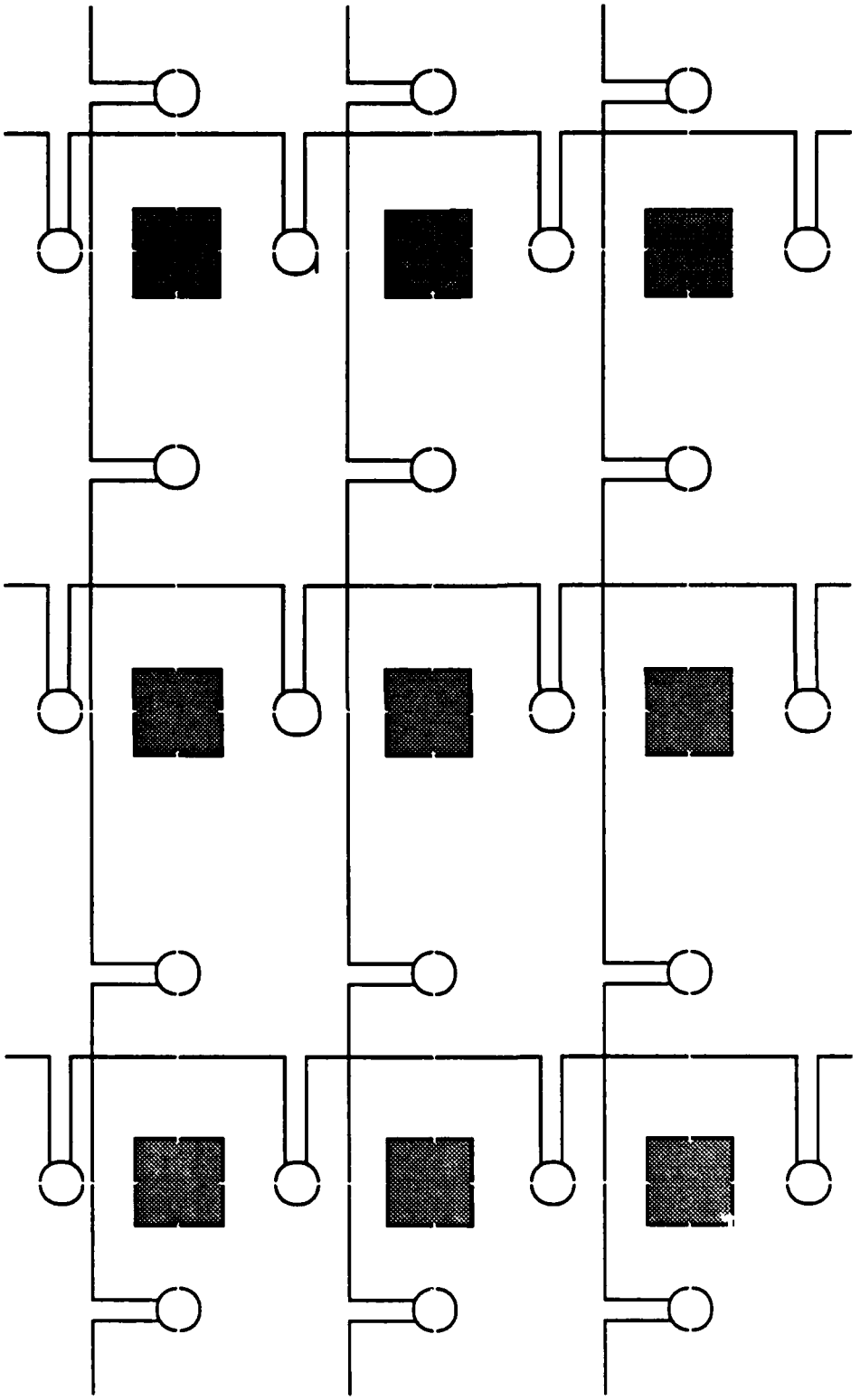


Figure 7. ARCE Architecture

Table 1. ARCE Data

Number of Elements	% Redundancy	Number of States	Improvement Factors SR = 1X	SR = 10X	SR = MAX
4	125	2	.4172	1.5764	2.1110
9	77	2	.5202	1.7944	2.3128
16	56	2	.5898	1.9285	2.4418
16	125	4	.4584	2.2635	3.4180
25	44	2	.6511	2.0146	2.5286
25	96	4	.5607	2.4804	3.6393
36	36	2	.7370	2.0501	2.5683
36	77	4	.6894	2.6535	3.8265
36	125	6	.6412	2.8014	4.7485
49	30	2	.8798	2.0243	2.5361
49	65	4	.8618	2.7713	3.9786
49	104	6	.8373	3.0293	5.0233
64	26	2	1.0894	1.9631	2.4334
64	56	4	1.0848	2.7918	4.0239
64	89	6	1.0752	3.2013	5.2903
64	125	8	1.0697	3.2523	6.1947
81	23	2	1.3564	1.9322	2.3174
81	49	4	1.3558	2.7142	3.8940
81	78	6	1.3528	3.2617	5.4271
81	109	8	1.3510	3.4461	6.5734
100	21	2	1.6681	1.9949	2.2697
100	44	4	1.6681	2.6114	3.6225
100	69	6	1.6674	3.2035	5.3023
100	96	8	1.6669	3.5261	6.7749
100	125	10	1.6667	3.5753	7.8651
121	23	2	2.0169	2.1770	2.3465
121	48	4	2.0169	2.5831	3.3391
121	75	6	2.0168	3.0998	4.9202
121	104	8	2.0167	3.4952	6.6255
144	25	2	2.4000	2.4680	2.5583
144	52	4	2.4000	2.6929	3.1811
144	81	6	2.4000	3.0607	4.4392
144	112	8	2.4000	3.4305	6.1209

SR - Switch Ratio

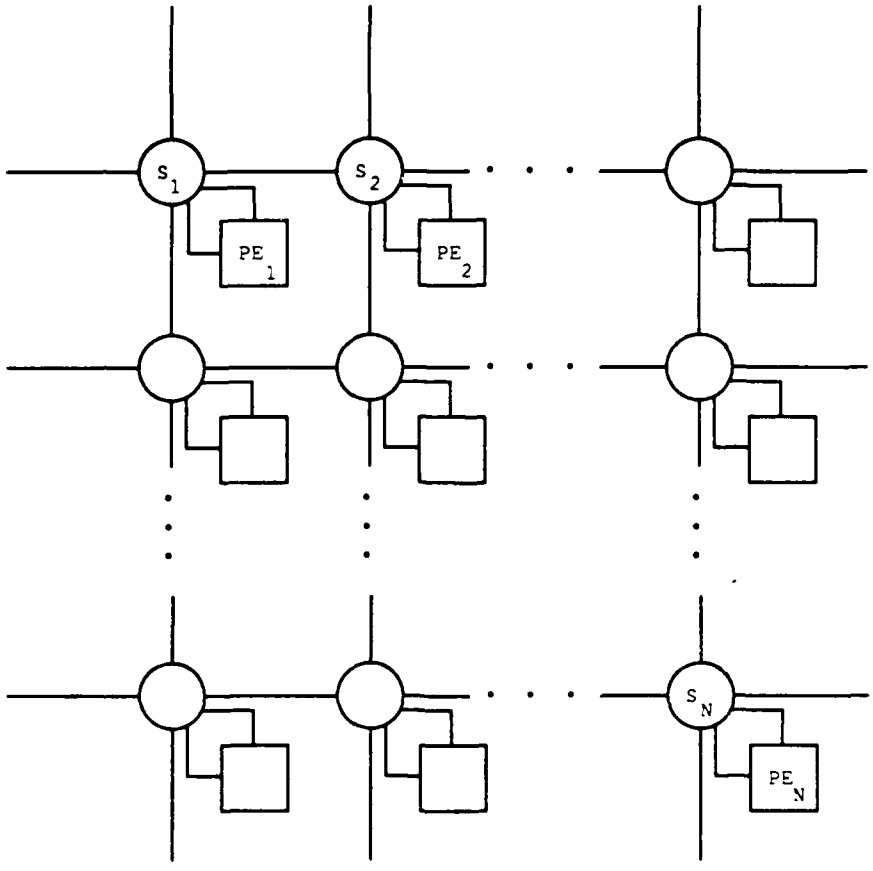


Figure. 8. Virtual Channel Architecture

Ideally the virtual channel fault tolerance method provides for any combination of operational processing elements given any combination of faulty processing elements. Applying probability theory for Bernoulli trials (references 13 and 14) the reliability for the virtual channel method with maximum coverage is

$$R_{Virtual}(t) = e^{-(N+K)\lambda_{VS}t} \sum_{i=0}^{K} \binom{N+K}{i} R^{(N+K-i)} (1-R)^{i},$$

where

N = the number of functionally required processing elements,
i = the possible number of failed processing elements, and
λ_{VS} = virtual channel switch failure rate.

The binomial theorem expands the equation to

$$R_{Virtual}(t) = \sum_{i=0}^{K} \sum_{p=0}^{i} \binom{N+K}{p} \binom{i}{p} (-1)^{p} R^{(FN+K-i+p)},$$

which leads to an MTBF improvement of

$$I_{Virtual} = \sum_{i=0}^{K} \sum_{p=0}^{i} \binom{N+K}{i} \binom{i}{p} (-1)^{p} \left[N/(FN+K-i+p)\right].$$

Figures 9 through 12 show the improvement expected with several switch ratios for 25%, 50%, 75%, and 100% hardware reserve levels. The virtual switch failure rate is crucial in evaluating the overall MTBF for the systolic array. The virtual array switch is also more complex than the switch used in the companion processor, the SRE, or the ARCE fault tolerance methods, and is expected to be less reliable.

The performance for the virtual channel method is limited in practice by the array size, number of reserve processing elements, and the virtual channel reconfiguration combinations. For example, a systolic array with 40 required processing elements and 20 spares would have to account for a maximum of 7.54×10^{10} possible reconfiguration combinations if 10 processing elements had failed (for complete fault coverage). For that reason, the improvement

for larger arrays would be somewhat less than the theoretical predictions. But, for smaller systolic arrays, the performance is accurate because the reconfiguration complexity is minimal. Fault coverage must be integrated into the analysis to determine practically realizable improvement levels. Nevertheless, the virtual channel fault tolerance method does have the theoretical capacity to provide the best improvement for the least amount of hardware reserve.

A closer observation of the virtual array performance data reveals a linear theoretical improvement. Table 2 presents the linear approximations for the maximum theoretical improvement curves given in figures 9 through 12. The best improvement requires 100% hardware redundancy but only diminishes the total systolic array MTBF to 30% of the MTBF of a single processing element. The improvement is 0.7(N). The virtual array fault tolerance technique has the theoretical potential to provide improvement in MTBF that is independent of array size. This feature was not achieved by the other techniques discussed in this report.

Table 2. Virtual Array Linear Approximations

Redundancy	Slope	Y-intercept	Equation (I =)
25 %	0.2254	1.0350	0.236N + 1.10
50 %	0.4176	0.2780	0.418N + 1.05
75 %	0.5487	1.2518	0.572N + 1.03
100 %	0.7048	0.7829	0 .703N + 1.03

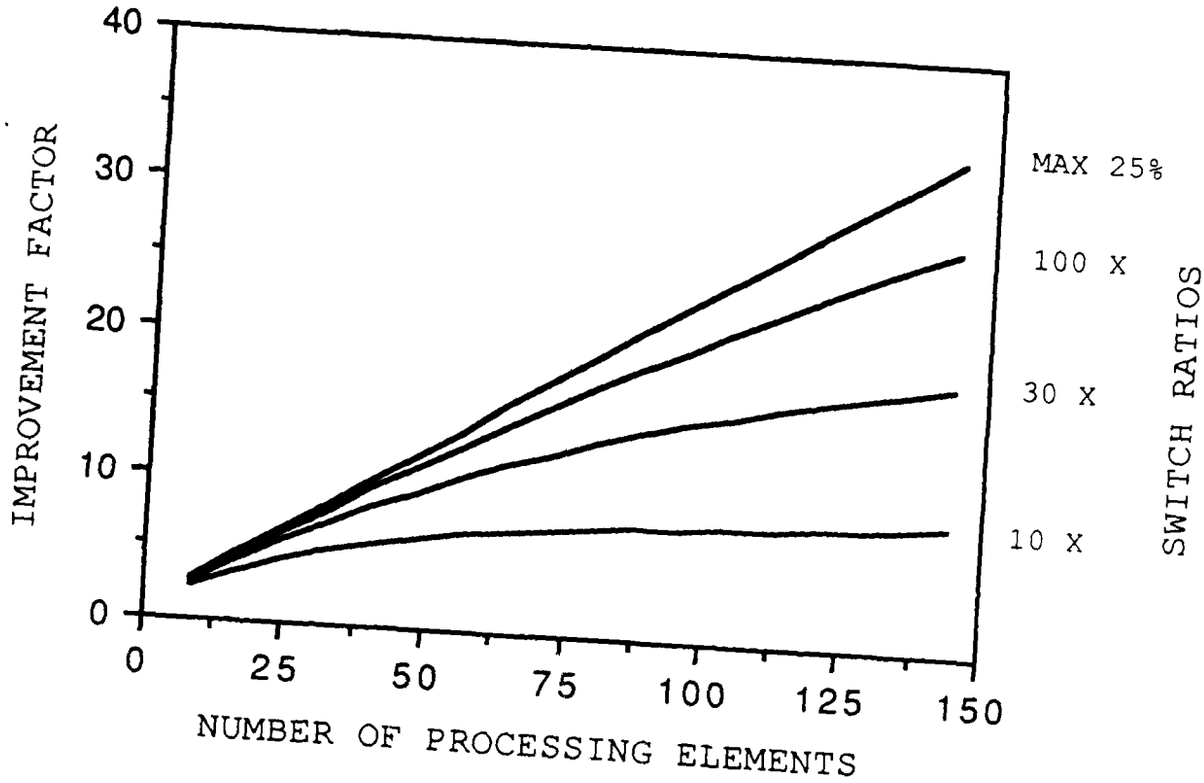


Figure 9. Virtual Array: 25% Redundancy

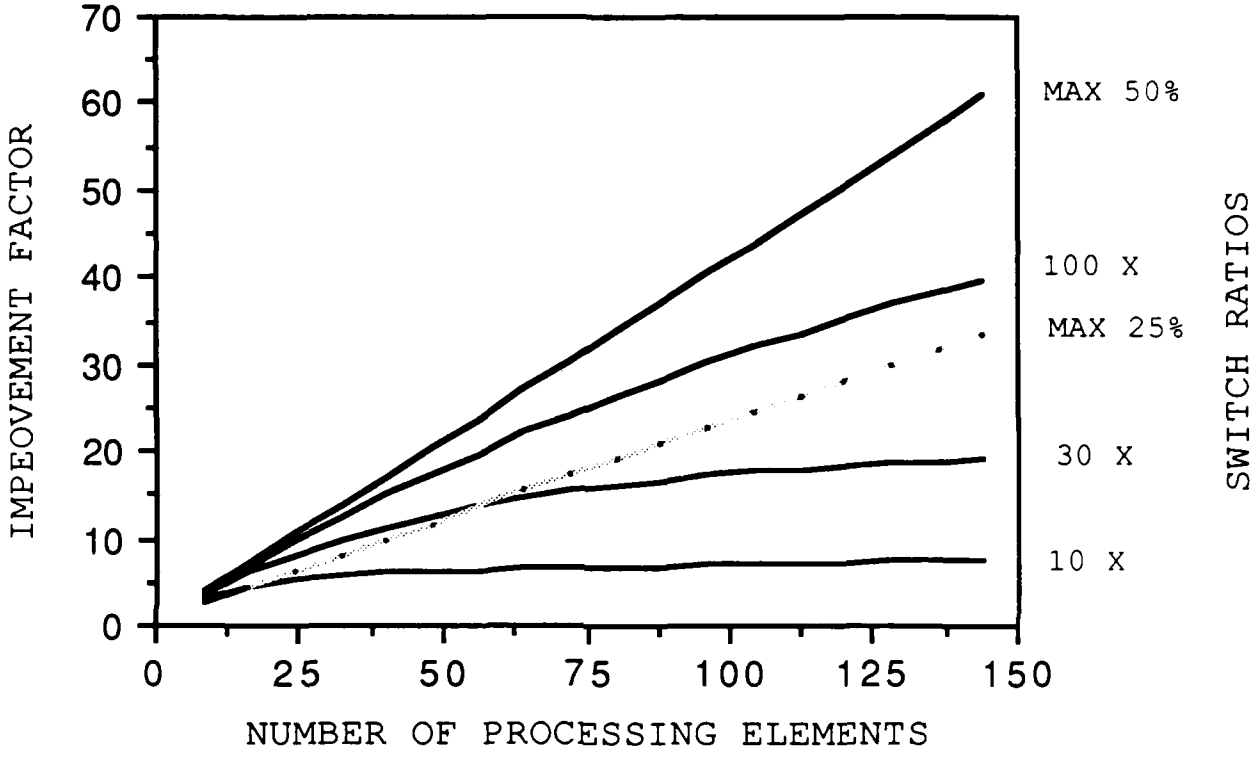


Figure 10. Virtual Array: 50% Redundancy

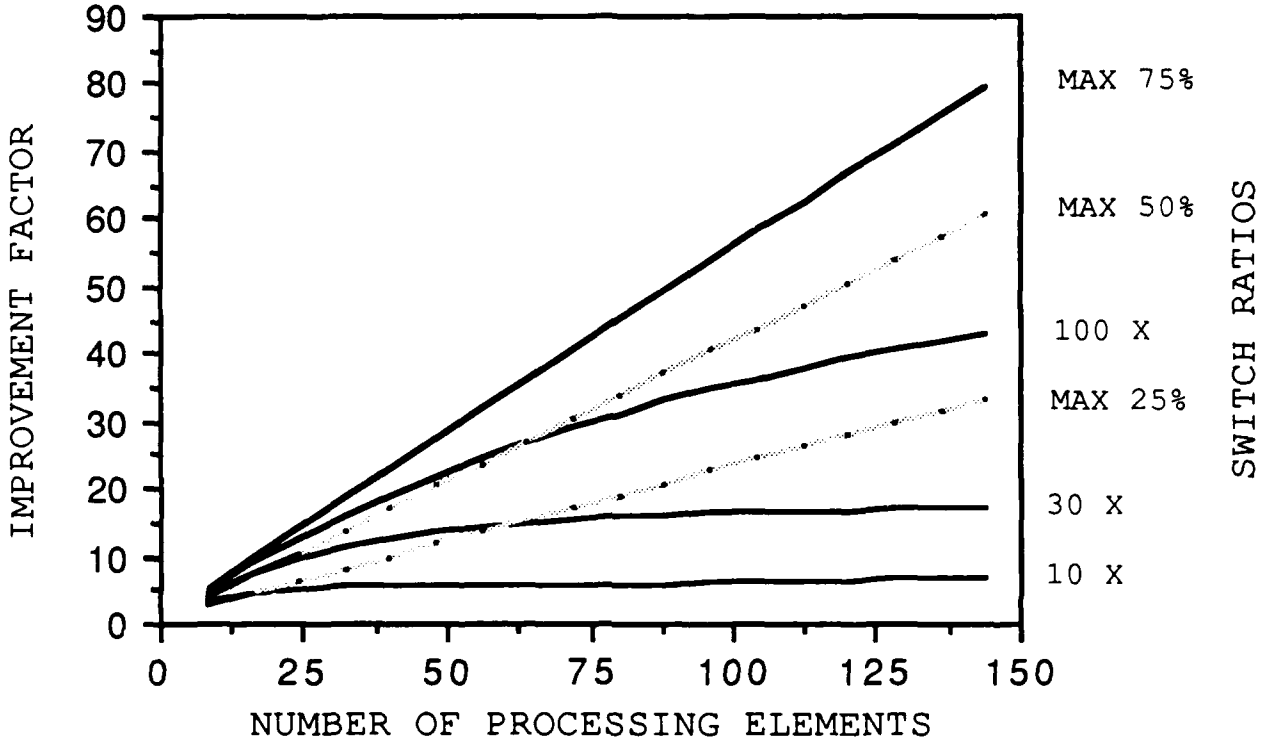


Figure 11. Virtual Array: 75% Redundancy

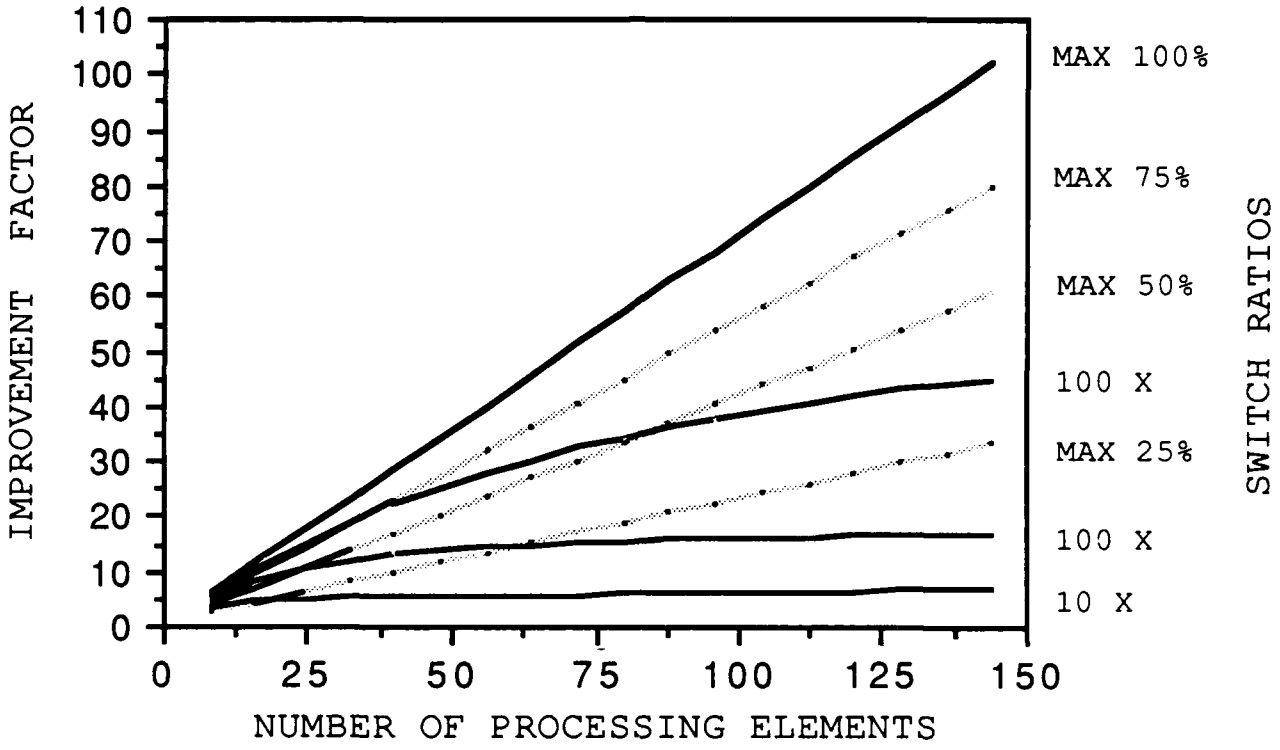


Figure 12. Virtual Array: 100% Redundancy

TREE ARCHITECTURES

Fault tolerant tree architectures can be designed by interconnecting a redundant tree as depicted in figure 13. Processing elements are paired so an operational processing element can assume the computational burden of a failed neighbor (reference 15). This idea is similar to the companion processors. In fact, the MTBF for the minimum throughput will be the same

$$R_{Tree}(t) = e^{-N\lambda_{TS}t} [R^2 + 2 R (1 - R)]^N$$

$$= R^{NF} (2 - R)^N,$$

where

λ_{TS} = the failure rate of the switch used to switch processing elements in the tree architecture.

However, the tree architecture doubles the available throughput of the companion processor method when failures have not occurred. The companion processor architecture does not allow for the reserve processing elements to be used until a failure occurs. The tree architecture can provide for graceful degradation. Although the performance for the companion processor *fault tolerance method* is applicable for tree architectures, care must be exercised to interpret the results correctly. For the companion processor approach N is related to both the minimum and maximum throughput available from the array. Moreover, N is the minimum throughput of the array, but the maximum available throughput is 2N for tree architectures.

TEMPORAL FAULT TOLERANCE THROUGH GRACEFUL DEGRADATION

Graceful degradation fault tolerance techniques allow reduced performance when a processing element has failed. The systolic array performs the identical algorithm, but with less throughput. Graceful degradation can be demonstrated with the MVDR beamforming algorithm.

The MVDR algorithm can be implemented in a subarray (SA) mode (reference 16). In the SA mode, groups of hydrophones are prebeamformed conventionally. These SA outputs are input to the adaptive beamforming algorithm for the entire array. The computational burden is trimmed down from N^3 by a factor of P^2, where P is the number of hydrophone elements in each SA. The actual computational load is $[(N/P)^2 P + (N/P)^2 N]$. For example, an array with

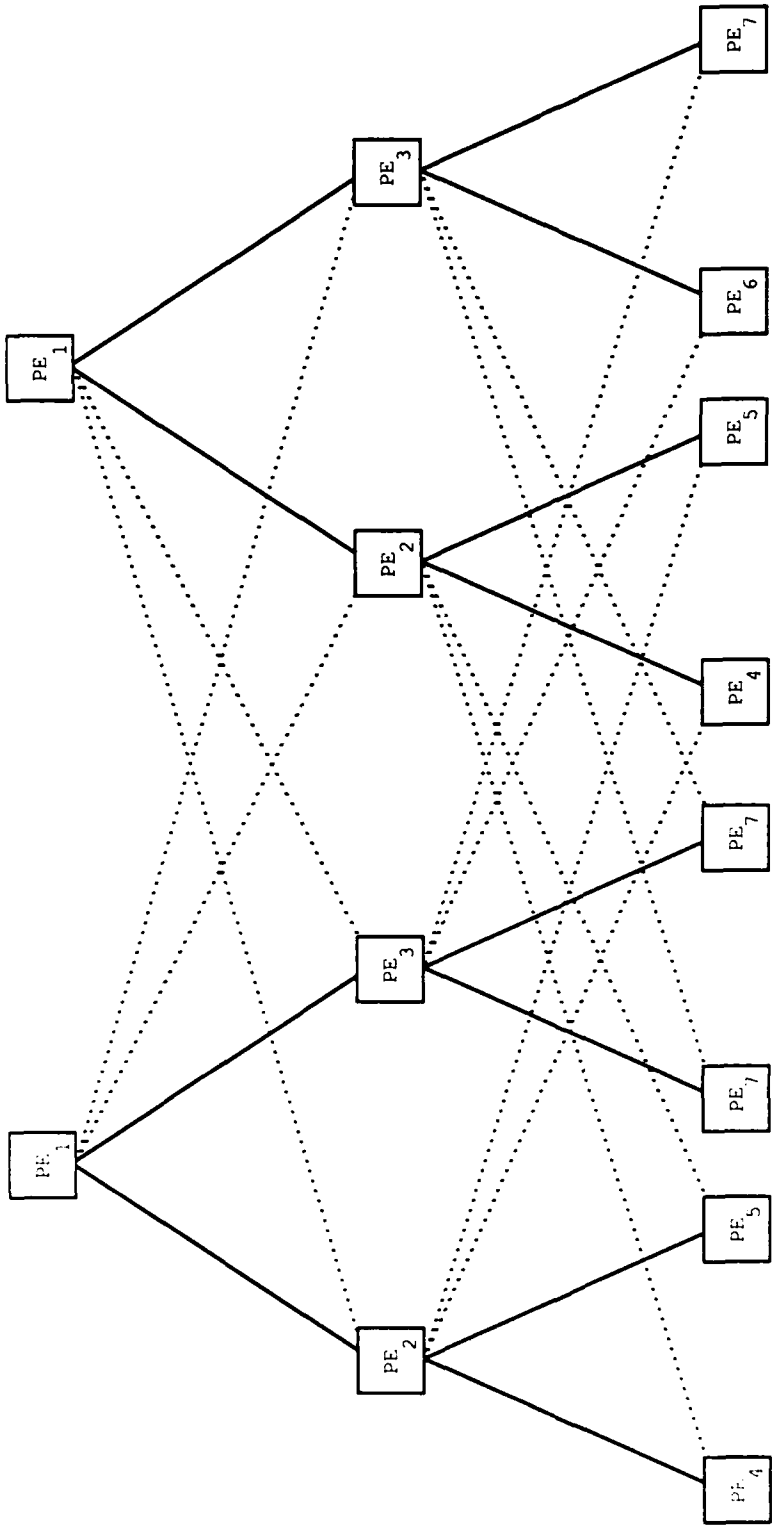


Figure 13. Tree Architecture

100 elements divided into SA's consisting of 2 elements (50 groups of 2) would have a computational load that is 25% of the load for the full array. Memory storage requirements for covariance matrices would also be reduced accordingly.

The SA approach sacrifices degrees of freedom for execution time. The algorithm is executed on a systolic array with less than optimal throughput capability within the same execution time. But, the number of interfering sources which can be attenuated by the adaptive beamforming algorithm is smaller. This concession may be acceptable for large arrays with many hydrophone sensors.

A fault tolerance method which exploited the SA approach would be capable of reconfiguring not only the hardware, but also the software to allow for the addition of conventional beamforming of the sensors in the SA's. Algorithm fault tolerance for linear system solvers has been addressed by Luk and Park (reference 17). The reliability performance of this graceful degradation fault tolerance technique would depend on system requirements including initial size of the transducer array, minimum reconfigured throughput of the systolic array, maximum throughput of the systolic array and maximum number of interference sources that are required to be attenuated by the adaptive beamformer. These system requirements would be integrated into one of the spatially redundant design approaches described earlier. The cost of implementing a given spatially redundant fault tolerance approach is mitigated because of the reduction in hardware reserves. For example, a 100 element array with 100% redundancy using virtual channels and ideal switches would have an MTBF of $0.7/\lambda$. If a minimum throughput from 50 processors was acceptable from a 100 element array, the very same MTBF would be reached with 50% of the hardware. The intent of this report is merely to propose this fault tolerant approach as a possibility for systolic array fault tolerance. A detailed assessment of the application of this graceful degradation technique is a topic for future work with a concentration in the area of specific applications.

CONCLUSIONS

Systolic array reliability performance can be enhanced through the use of fault tolerant reconfiguration techniques. Figure 14 points out the difference between the MTBF of a systolic array and a fault tolerant systolic array (Virtual method). The ideal theoretical throughput is also shown along with Amdahl's throughput. The line on the graph that corresponds to Amdahl's throughput shows the actual throughput that can be expected from the system. Spatial and temporal redundancy techniques were studied and their performance quantified in terms of mean time between failure (MTBF).

The results of the investigation clearly show that virtual array fault tolerance offers the best theoretical improvement for high throughput requirements. In fact, a constant MTBF, independent of array size, can be demonstrated. The theoretical improvement is nearly an order of magnitude better than redundant arrays, companion processor, SRE, or tree based approaches. The virtual channel technique yields a theoretical improvement of 0.7(N). However, the practical performance is a function of the ratio of the switching mechanism failure rate to the processing element failure rate. The improvement is dependent on hardware design implementations.

The fault tolerance techniques discussed provide comparable performance for low throughput levels. In fact, each technique converges to an improvement of 1.5 as the number of processing elements approaches unity (assuming ideal switching mechanisms). Therefore, for low throughput levels, the complexity in implementing each fault tolerance technique should be the major concern when cost is a priority.

In addition, the performance of each fault tolerance technique analyzed is highly dependent on the reliability performance of the switching mechanism used in the design. This dependence cannot be overlooked in any assessment. The switch design is an important consideration. Generally, in terms of failure rate, the switch must outperform the processing element by at least one order of magnitude and preferably more than two orders of magnitude.

The results reported in this report assume that operational processing elements and hardware reserves operate over the same time period. Additional improvement can be sought with stand-by connections for the hardware reserves. Stand-by connections would eliminate any overlap

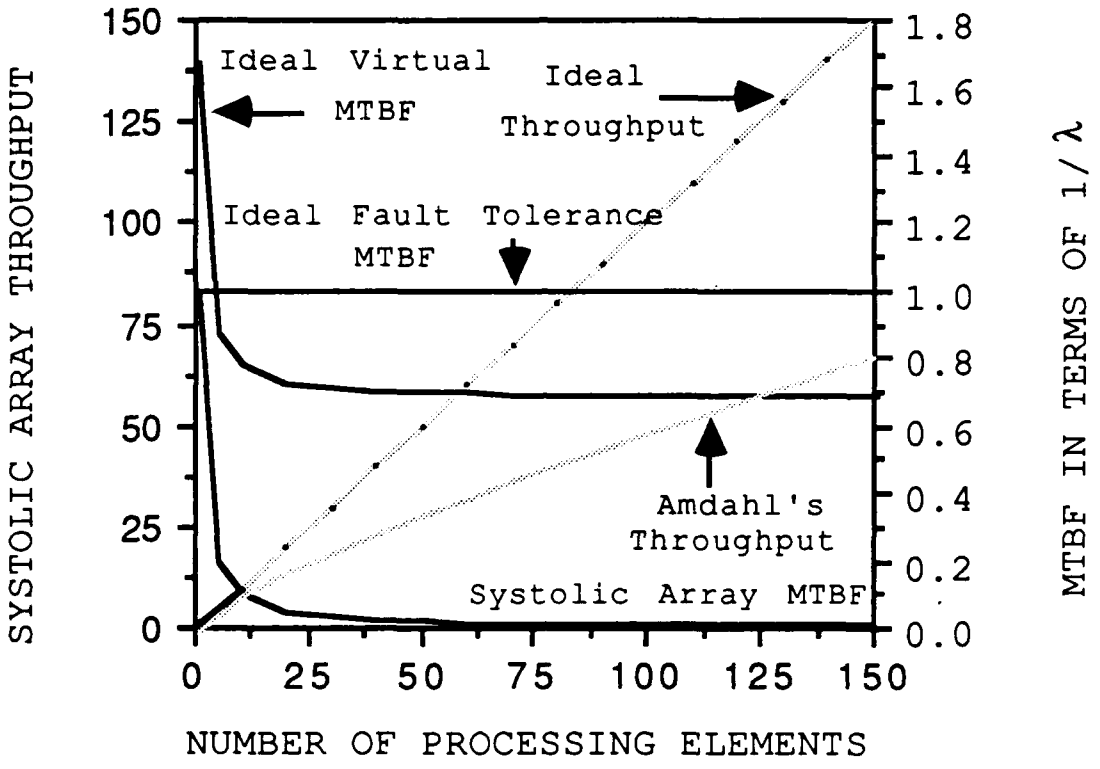


Figure 14. Comparison of Systolic and Virtual Arrays

in the useful life of operational processing elements and reserves to maximize the total MTBF (reference 18).

In addition, the improvement levels outlined are not sufficient to recover the loss in reliability due to array size. The design of highly reliable components, i.e., processing elements, should accompany the application of fault tolerance architectures.

Graceful degradation can provide a cost effective approach to fault tolerance. A cost versus system performance comparison was addressed for the minimum variance distortionless response beamformer.

The results described in this report provide only a stepping stone in the path of maturation for reliable systolic array architectures. Without question, there are several unresolved issues which require further examination, i.e.,

1. suitability of available architectures to specific system applications,

2. theoretical versus practicable performance,

3. reliability attainable through standby operation, and

4. architecture performance validation via modeling and simulation.

APPENDIX A

PROOFS

REDUNDANT ARRAYS

Pr(array "A" or array "B" works) = Pr("A" works and "B" works)
+ Pr("A" works and "B" doesn't work)
+ Pr("A" doesn't work and "B" works).

From report: $Pr(\text{"A" works}) = Pr(\text{"B" works}) = e^{-n\lambda t}$

$$Pr(\text{array "A" or array "B" works}) = e^{-n\lambda t} e^{-n\lambda t} + e^{-n\lambda t}(1-e^{-n\lambda t}) + (1 - e^{-\lambda t}) e^{-\lambda t}$$

$$= e^{-2n\lambda t} + 2e^{-n\lambda t}(1 - e^{n\lambda t})$$

$$= e^{-n\lambda t}(2 - e^{-n\lambda t})$$

$$MTBF = \int_0^\infty R(t)\,dt = \int_0^\infty [2e^{-n\lambda t} - e^{-2n\lambda t}]dt$$

$$MTBF = 2/(\lambda n) - 1/(2\lambda n) = 1.5\,(1/(\lambda n)) = 1.5\ M$$

Improvement = 1.5.

COMPANION PROCESSORS

Pr(companion array works) = Pr(all switches work) [Pr(one processing element or its companion work)]^n

Pr(one processing element or its companion works) = Pr(both work)
+ 2Pr(one works)Pr(the other doesn't work)

Pr(one processing element or its companion works) =
$e^{-\lambda t} e^{-\lambda t} + 2e^{-\lambda t}(1 - e^{-\lambda t}) = R^2 + 2R(1 - R)$

$$\Pr(\text{companion array works}) = e^{-n\lambda_s t}\left[e^{-2\lambda t} + 2e^{-\lambda t}(1-e^{-\lambda t})\right]^n$$

$$\Pr(\text{companion array works}) = e^{-[n\lambda_s t + n\lambda t]}\left[2 - e^{-\lambda t}\right]^n$$

$$\Pr(\text{companion array works}) = e^{-n\lambda t[1 + \lambda_s/\lambda]}\left[2 - e^{-\lambda t}\right]^n$$

$$\Pr(\text{companion array works}) = e^{-n\lambda f t}\left[2 - e^{-\lambda t}\right]^n$$

$$MTBF = \int_0^\infty e^{-n\lambda f t}\left[2 - e^{-\lambda t}\right]^n$$

Binomial Theorem: $(A + B)^n = \Sigma \binom{n}{i} A^{(n-i)} B^i$

$$MTBF = \int_0^\infty e^{-n\lambda f t} \sum_{i=0}^{n} \binom{n}{i} 2^{(n-i)} (-1)^i (e^{-i\lambda t})$$

$$MTBF = \int_0^\infty \sum_{i=0}^{n} \binom{n}{i} 2^{(n-i)} (-1)^i e^{-\lambda t(i+nf)}$$

$$MTBF = \sum_{i=0}^{n} \left(\binom{n}{i} 2^{(n-i)} (-1)^i\right) / \left(\lambda(i+nf)\right)$$

$$MTBF = 1/(n\lambda) \sum_{i=0}^{n} \binom{n}{i} 2^{(n-i)} (-1)^i n / (i+nf)$$

$$\text{Improvement} = \sum_{i=0}^{n} \binom{n}{i} 2^{(n-i)} (-1)^i \; n \; / \; (i+nf).$$

SEQUENTIAL ROW ELIMINATION (SRE) AND ALTERNATE ROW-COLUMN (ARCE) ELIMINATION TECHNIQUES

Given a discrete-state continuous-transtition system which has exponentially distributed faults, it can be modeled by a Markov model, i.e.,

SRE : in state #0 n X n elements function,
in state #1 (n-1) X n elements function,
in state #2 (n-2) X n elements function, etc.

For each state another row has been removed, so the failure transition rate can be written as $A_i = (n-i)n\lambda$.

ARCE : in state #0 n X n elements function
in state #1 (n-1) X n elements function
in state #2 (n-1) X (n-1) elements function, etc.

For each state a row or a column has been removed, so the failure transition rate can be written as

$$A_i = (n - \lceil (i/2) \rceil)(n - \lfloor (i/2) \rfloor)$$

The bars signify the round-up and round-down functions.

Pr(system is in present state for Δt)
= Pr(system was in last state)
+ Pr(no change in state will occur),

but

Pr(system was in last state) = Pr(last state)Pr(transition from last state),

Pr(no change in state will occur) = Pr(current state)Pr(no transition).

By using the above results, a formula for both SRE and ARCE can be derived (reference 14), i.e.,

$$Pr_0(t+\Delta t) = Pr_0(1 - A_0\Delta t)$$

$$Pr_1(t+\Delta t) = Pr_0A_0\Delta t + Pr_1(1 - A_1\Delta t)$$

$$Pr_2(t+\Delta t) = Pr_1A_1\Delta t + Pr_2(1 - A_2\Delta t).$$

Taking the derivative with respect to Dt and taking the limit as $\Delta t \rightarrow 0$ we have

$$Pr_0'(t) = -A_0Pr_0(t)$$

$$Pr_1'(t) = A_0Pr_0(t) - A_1Pr_1(t)$$

$$Pr_2'(t) = A_1Pr_1(t) - A_2Pr_2(t)$$

The generalized formula is $Pr_k'(t) = A_{k-1}Pr_{k-1}(t) - A_kPr_k(t)$.

Taking the Laplace transform, i.e.,

$$sPr_k(s) = -A_kPr_k(s) + A_{k-1}Pr_{k-1}(s)$$

but,

$$sPr_{k-1}(s) = -A_{k-1}Pr_{k-1}(s) + A_{k-2}Pr_{k-2}(s)$$

$$Pr_{k-1}(s) = (A_{k-2}Pr_{k-2}(s))/(s + A_{k-1}).$$

In general

$$Pr_i(s) = (A_{i-1}Pr_{i-1}(s)) \; / \; (s + A_i);$$

therefore (reference 10),

$$Pr_k(s) = 1 \; / \; (s + A_k)[A_{k-1}(A_{k-2}/(s+A_{k-1})(A_{k-3})/(s+A_{k-2}) \; . \; . \; . \; (A_0/(s+A_0))]$$

$$Prk(s) = \frac{\prod_{j=0}^{k-1} A_j}{\prod_{j=0}^{k} (s + A_j)} .$$

To find Prk(t),

$$L^{-1}[F(s)] = L^{-1}\left[\frac{1}{\prod_{j=0}^{k} (s + A_j)}\right] .$$

By partial fraction expansion (reference 19), we get

$$\frac{1}{\prod_{j=0}^{k} (s + A_j)} = \sum_{i=0}^{k} \frac{Z_i}{(s + A_j)}$$

where

$$Z_i = \frac{1}{\prod_{\substack{j=0 \\ j\neq i}}^{k} (s + A_j)} \Bigg|_{@s = -A_i} = \frac{1}{\prod_{\substack{j=0 \\ j\neq i}}^{k} (A_j - A_i)} .$$

Therefore the inverse Laplace of

$$\sum_{i=0}^{k} \frac{1}{(s + A_i) \prod_{\substack{j=0 \\ j\neq i}}^{k} (A_j - A_i)}$$

is

$$\sum_{i=0}^{k} \frac{e^{-A_i t}}{\prod_{\substack{j=0 \\ j\neq i}}^{k} (A_j - A_i)},$$

which leads to the final result of

$$Pr_k(t) = \sum_{i=0}^{k} \frac{\prod_{j=0}^{k-1} A_j}{\prod_{\substack{j=0 \\ j\neq i}}^{k} (A_j - A_i)} e^{-A_i t}.$$

The MTBF is

$$\int_0^{\infty} \sum_{i=0}^{k} \frac{\prod_{j=0}^{k-1} A_j}{\prod_{\substack{j=0 \\ j\neq i}}^{k} (A_j - A_i)} e^{-A_i t} \, dt$$

VIRTUAL ARRAYS

Pr(minimum size array works) = Pr(all switches works

* [Pr(all elements work) + Pr(all elements except one work)

+ ...+ Pr(all elements except k spares work)]

$$\Pr(\text{all elements work}) = e^{-(n+k)\lambda t}$$

$$\Pr(\text{all elements except one work}) = \binom{n+k}{1} e^{-(n+k-1)\lambda t}$$

.
.
.

$$\Pr(\text{all elements except k spares work}) = \binom{n+k}{k} e^{-n\lambda t}(1 - e^{-\lambda t})^k$$

$$\Pr(\text{all switches work}) = e^{-(n+k)\lambda_s t}$$

$$\Pr(\text{minimum size array works}) = e^{-(n+k)\lambda_s t} \sum_{i=0}^{K} \binom{n+k}{i} e^{-(n+k-i)\lambda t}(1-e^{-\lambda t})^i$$

$$\Pr(\text{minimum size array works}) = \sum_{i=0}^{K} \binom{n+k}{i} e^{-[(n+k)[1+(\lambda_s/\lambda]-i]\lambda t}(1-e^{-\lambda t})^i$$

$$\Pr(\text{minimum size array works}) = \sum_{i=0}^{K} \binom{n+k}{i} e^{-[(n+k)f-i]\lambda t}(1-e^{-\lambda t})^i$$

$$\text{Pr(minimum size array works)} = \sum_{i=0}^{K} \binom{n+k}{i} e^{-[(n+k)f-i]\lambda t} \sum_{p=0}^{i} \binom{i}{p} (-1)^p e^{-p\lambda t}$$

$$\text{Pr(minimum size array works)} = \sum_{i=0}^{k} \sum_{p=0}^{i} \binom{n+k}{i}\binom{i}{p} e^{-[(n+k)f-i+p]\lambda t} (-1)^p$$

$$\text{MTBF} = \int_0^{\infty} \sum_{i=0}^{k} \sum_{p=0}^{i} \binom{n+k}{i}\binom{i}{p} e^{-[(n+k)f-i+p]\lambda t} (-1)^p$$

$$\text{MTBF} = \int_0^{\infty} \sum_{i=0}^{k} \sum_{p=0}^{i} \binom{n+k}{i}\binom{i}{p} (-1)^p [n \,/\, ((n+k)f-i+p)$$

$$\text{MTBF} = M \sum_{i=0}^{k} \sum_{p=0}^{i} \binom{n+k}{i}\binom{i}{p} (-1)^p \,[n \,/\, ((n+k)f-i+p)]$$

APPENDIX B

PROOFS

```
C ***************************************************************
C *     THIS PROGRAM WILL CALCULATE THE IMPROVEMENT FACTOR OF    *
C *      COMPANION FAULT TOLERANCE METHOD GIVEN THE SWITCH RATIO *
C *      AND THE STEP SIZE.  THE STEP SIZE IS NEEDED TO          *
C *      CALCULATE THE INTEGRATION BY USING THE RIEMANN SUM.     *
C ***************************************************************

        PROGRAM  COMP
        IMPLICIT NONE

C ***************************************************************
C *       DECLARE ALL VARIABLES THAT WILL BE USED IN THE PROGRAM  *
C ***************************************************************

        REAL*4 PARTA, Z, X, Y, E, L, A, B, C, DELTA, N, SR, F, R
        REAL*16  MTTF, IMP(100)

C ***************************************************************
C *       GET THE INPUT FROM THE TERMINAL                        *
C ***************************************************************

        READ*, SR, DELTA
        F = 1. + (1./SR)

C ***************************************************************
C *     LOOP TO GET DIFFERENT VALUES OF N, WHICH IS THE NUMBER OF *
C *      PROCESSING ELEMENTS.                                    *
C *    THE FOLLOWING ROUTINE CALCULATES THE FORMULA:             *
C *                            ∞                                 *
C *                           ∫ e -nft(2 - e -t)n                *
C *                           0                                 *
C ***************************************************************

        DO N = 0.,144.,4.
                MTTF = 0.
                DO L = 0, 5., DELTA
                        A = EXP(-N*F*L)
                        R = EXP(-L)
                        B = 2. - R
                        C = N/2.
                    MTTF = MTTF + ((A * B**C) * B**C)
                END DO
                IMP(N) = MTTF * N * DELTA
```

```
C ***********************************************************************
C * PRINT THE OUTPUT IN TABLE FORM SO THAT THEY CAN BE GRAPHED LATER *
C ***********************************************************************

               PRINT 10, N, IMP(N)
10             FORMAT(5X,F4.0,5X,F10.4)
        END DO
        STOP
        END
           PROGRAM  SRE
           IMPLICIT NONE

        REAL         DT, DELTA, PC, SR, R, H, E, G, S, COEF
        REAL         O, B, A
        REAL*16      PB, A1, A2, ANSW, ANSW2, ANSWER
        INTEGER*4    K, I, J, Z, N, W, D, T, COUNT
        CHARACTER    AGAIN


C ***********************************************************************
C *  THIS PROGRAM CALCULATES THE IMPROVEMENT FACTOR OF USING      *
C *   SEQUENTIAL ROW ELIMINATION FAULT TOLERANCE METHOD, GIVEN    *
C *   THE SWITCH RATIO AND THE STEP SIZE. THE INTEGERATION  IS    *
C *   DONE BY USING SIMPSONS RULE.                                *
C ***********************************************************************

        AGAIN = 'Y'
        DO WHILE (AGAIN .EQ. 'Y')
           AGAIN ='J'
           ANSWER = 0.


C ***********************************************************************
C *  GET THE INPUT FROM THE TERMINAL AND CALCULATE THE STEP SIZE.*
C ***********************************************************************

           READ*, N, D
           H = 0
           E = 3.
           G = 60.
           DELTA = (E-H)/G
           COUNT = -1
```

```
C ********************************************************************
C *     THE FOLLOWING ROUTINE CALCULATES THE FORMULA :                 *
C *                              K-1                                   *
C *                              Π     λ_j                             *
C *           ∞    D     K       j=0                                   *
C *          ∫   Σ     Σ    ---------------  e^(-λ_i t)  dt            *
C *         0    k=0   i=0        k                                     *
C *                              Π (λ_j - λ_i)                         *
C *                             j=0                                    *
C *                             j≠0                                    *
C ********************************************************************

            DO DT = 0, 3., DELTA
                 COUNT = COUNT + 1
                 R = (E - H) / (G * 3.)
                 SR = 100000.
                    Z = N**2
                    PC = EXP((1/SR)*Z*-DT)
                    ANSW2 = 0.
                         DO K = 0, D
                                  ANSW = 0.
                                  DO I = 0, K
                                           A1 = 1.
                                           A2 = 1.
                                           DO J = 0, K-1
                                              B = D * (N - J)
                                           A1 = A1 * B
                                           END DO
                                           A = D * (N - I)
                                           PB = EXP(-A * DT)
                                           DO J = 0, K
                                              IF (J .EQ. I) GOTO 10
                                              B = D * (N - J)
                                              A2 = A2 * (B - A)
10                                         END DO
                                    ANSW = ANSW + ((A1/A2) * PB)
                                  END DO
                                  ANSW2 = ANSW2 + (ANSW*PC)
                              END DO
                              IF (IAND(COUNT,1) .EQ. 0) THEN
                                       COEF = 2.
                              ELSE
                                       COEF = 4.
                              ENDIF
                              IF (DT .EQ. 0 .OR. D .EQ. 3.) THEN
                                       COEF = 1.
                              ENDIF
                              ANSWER = ANSWER + (ANSW2*COEF)
                 END DO
```

```
C ***************************************************************
C *            PRINT  OUT  RESULTS  OF  PROGRAM                 *
C ***************************************************************

                S = D ** 2
                PRINT 20, S, ANSWER*R*S
20              FORMAT (2X,F4.0, F20.4)
        READ 30, AGAIN
30      FORMAT (A1)
        END DO
        STOP
        END

              PROGRAM  ARCE
              IMPLICIT NONE

        REAL          DT, DELTA, C, PC, SR, R, H, E, G, S, COEF, M
        REAL          O, B, A, F
        REAL*16       PB, A1, A2, ANSW, ANSW2, PA, ANSWER
        INTEGER*4     K, I, J, Z, N, W, D, T, COUNT
        CHARACTER     AGAIN


C ***************************************************************
C *  THIS PROGRAM CALCULATES THE IMPROVEMENT FACTOR OF USING     *
C *   ALTERNATE ROW-COLUMN ELIMINATION FAULT TOLERANCE METHOD,   *
C *   GIVEN THE SWITCH RATIO AND THE STEP SIZE. THE INTEGERATION *
C *   DONE BY USING SIMPSONS RULE.                               *
C ***************************************************************

        AGAIN = 'Y'
        DO WHILE (AGAIN .EQ. 'Y')
           AGAIN ='J'
           ANSWER = 0.


C ***************************************************************
C *  GET THE INPUT FROM THE TERMINAL AND CALCULATE THE STEP SIZE.*
C ***************************************************************

           READ*, N, D
           H = 0
           E = 3.
           G = 60.
           DELTA = (E-H)/G
           COUNT = -1
```

```
C ***********************************************************************
C *     THE FOLLOWING ROUTINE CALCULATES THE FORMULA :                  *
C *                                K-1                                  *
C *                               Π    λ_j                             *
C *           ∞    D    K         j=0                                   *
C *          ∫   Σ    Σ   ---------------  e^(-λ_i t)  dt               *
C *         0   k=0  i=0        k                                       *
C *                              Π (λ_j - λ_i)                          *
C *                             j=0                                     *
C *                             j≠0                                     *
C ***********************************************************************

            DO DT = 0, 3., DELTA
                 COUNT = COUNT + 1
                 R = (E - H) / (G * 3.)
                 SR = 100000.
                    Z = N**2
                    PC = EXP((1/SR)*Z*-DT)
                    ANSW2 = 0.
                         DO K = 0, D
                                 ANSW = 0.
                                 DO I = 0, K
                                         A1 = 1.
                                         A2 = 1.
                                         DO J = 0, K-1
                                           M = N - (J/2.)
                                           F = NINT(M)
                                           O = INT(M)
                                           A1 = A1 * F * O
                                         END DO
                                         C = N - (I/2.)
                                         A = NINT(C)
                                         B = INT(C)
                                         PA = A * B
                                         PB = EXP(-PA * DT)
                                         DO J = 0, K
                                            IF (J .EQ. I) GOTO 10
                                            P = N - (J/2.)
                                            Q = NINT(P)
                                            L = INT(P)
                                            A2 = A2 * ((L * Q) - PA)
10                                       END DO
                                  ANSW = ANSW + ((A1/A2) * PB)
                                 END DO
                                 ANSW2 = ANSW2 + (ANSW*PC)
                          END DO
                          IF (IAND(COUNT,1) .EQ. 0) THEN
                                 COEF = 2.
                          ELSE
                                 COEF = 4.
```

```
                        ENDIF
                        IF (DT .EQ. 0 .OR. D .EQ. 3.) THEN
                                COEF = 1.
                        ENDIF
                        ANSWER = ANSWER + (ANSW2*COEF)
                END DO

C ***************************************************************
C *            PRINT  OUT  RESULTS  OF  PROGRAM                 *
C ***************************************************************

                S = (N - (D/2.)) ** 2
                PRINT 20, S, ANSWER*R*S
20              FORMAT (2X,F4.0, F20.4)
        READ 30, AGAIN
30      FORMAT (A1)
        END DO
        STOP
        END

C ***************************************************************
C *    THIS PROGRAM CALCULATES THE IMPROVEMENT FACTORS OF USING  *
C *     VIRTUAL FAULT TOLERANCE METHOD GIVEN THE SWTICH RATIO AND *
C *     REDUNDANCY.  THE INTEGRATION OF THE  FORMULA IS DONE BY   *
C *     USING SIMPSONS RULE.                                      *
C ***************************************************************

                PROGRAM  VIR
                IMPLICIT NONE


C ***************************************************************
C *     DECLARE THE VARIABLES THAT WILL BE USED IN THE PROGRAM    *
C ***************************************************************

        REAL*16 CMB, Z, PARTB2
        REAL*16 ANSW, ANSWER, PARTA, PARTB, PARTC, PARTD, PA, PB
        REAL  R, W, U, P, F, K, SR, Y, D, DELTA
        REAL  DT, J, CONST, RK, E, A, B, CON, SIMPSONS_CON, COEFFICIENT
        INTEGER  X, N, I, RED, T, COUNT
        DIMENSION  CMB(400,402)
        LOGICAL  AGAIN
        CHARACTER  CHAR


C ***************************************************************
C *       THE FOLLOWING ROUTINE CALUCLATES PASCALS TRIANGLE.  THE *
C *        RESULTS ARE STORED IN AN ARRAY, SO THAT THEY CAN BE USED *
C *        LATER IN THE MAIN PROGRAM.  THE ROUTINE TAKES ADVANTAGE  *
C *        OF THE SYMMETRY OF PASCALS TRIANGLE.                     *
C ***************************************************************

        DO N = 1,300
                CMB(N,1) = 1.
```

```
                CMB(N,2) = N
                X = IAND(N,1)
                IF (X .EQ. 0) THEN
                        DO I = 3,(N/2)+1
                                R = N
                                W = I - 1.
                                IF (I .EQ. ((N/2)+1)) THEN
                                   CMB(N,I) = CMB(N,N+1-I)*((R+1.-W)/W)
                                ELSE
                                   CMB(N,I) = CMB(N,I-1)*((R+1.-W)/W)
                                ENDIF
                        END DO
                ELSE
                        DO I = 3,(N+1)/2
                                R = N
                                W = I - 1.
                                CMB(N,I) = CMB(N,I-1)*((R+1.-W)/W)
                        END DO
                ENDIF
                IF (X .EQ. 0) THEN
                        DO I = (N/2)+2, N+1
                                CMB(N,I) = CMB(N, N+2-I)
                        END DO
                ELSE
                        DO I = (((N+1)/2)+1), N+1
                                CMB(N,I) = CMB(N, N+2-I)
                        END DO
                ENDIF
        END DO

C ***************************************************************
C * GET RID OF THE UNSIGNIFICANT VALUES RIGHT OF THE DECIMAL POINT *
C ***************************************************************

        DO R = 1.,300.
                DO W = 1.,R + 1.
                        CMB(R,W) = QNINT(CMB(R,W))
                END DO
        END DO


C ***************************************************************
C *                START OF THE MAIN PART                       *
C *                SOLVING THE EQUATION :                       *
C *                                                             *
C *              ∞   K                                          *
C *              ∫   Σ   N+K  e^-(F(N+K) - I)t (1 - e^-t)^I     *
C *              0  I=0   I                                     *
C ***************************************************************

        AGAIN = .TRUE.
```

```
1       IF (AGAIN .EQ. .FALSE.) GO TO 100
        READ*, SR, RED
        DELTA = .5
        PRINT 60, SR,RED
60      FORMAT(/5X,'SWITCH RATIO:',F7.0,3X,'REDUNDANCY',I4)
        DO R = 96.,144.,8.
                F = 1. + (1./SR)
                IF (RED .EQ. 25) K = R/4.
                IF (RED .EQ. 50) K = R/2.
                IF (RED .EQ. 75) K = R * .75
                IF (RED .EQ. 100) K = R
                ANSWER = 0.
                RK = R + K
                CONST = RK * F
                SIMPSONS_CON = 1./6.
                COUNT = -1
                DO DT = 0, 5., DELTA
                        ANSW = 0.
                        COUNT = COUNT + 1
                        E = EXP(-DT)
                        PARTC = 1. - E
                        DO J = 0.,K
                                IF (J .EQ. 0 ) THEN
                                        PARTB = 1.
                                        PA = PARTB
                                ELSE
                                        PARTB = E**((CONST-J)/2.0)
                                        PA = (PARTC**(J/2.))*PARTB
                                ENDIF
                                ANSW = ANSW + (CMB(RK,J+1.)*PA*PA)
C ***************************************************************
C *          VALUES WHERE DIVIDED BY 2, SO THAT THE ROUND OFF      *
C *           ERRORS WOULD BE MINIMAL.                             *
C ***************************************************************
                        END DO
                        IF (IAND(COUNT,1) .EQ. 0) THEN
                                COEFFICIENT = 2.
                        ELSE
                                COEFFICIENT = 4.
                        ENDIF
                        IF (DT .EQ. 0 .OR. DT .EQ. 5.) THEN
                                COEFFICIENT = 1.
                        ENDIF
                ANSWER = ANSWER + SIMPSONS_CON*COEFFICIENT*ANSW
                END DO


C ***************************************************************
C *      PRINT OUT THE RESULT AND LOOP IF THE USER RESPONDS      *
C *       POSITIVELY.                                            *
C ***************************************************************
                PRINT 65, R,ANSWER*SIMPSONS_CON
65              FORMAT(5X,'N =',F4.0,2X,F20.4)
        END DO
```

```
        READ 70, CHAR
70      FORMAT(A1)
        IF (ICHAR(CHAR) .EQ. 78) THEN
                AGAIN = .FALSE.
        ENDIF
        GO TO 1
100     STOP
        END
```

REFERENCES

[1] N. L. Owsley, "Systolic Array Adaptive Beamforming," International Signal Processing Conference Proceedings, Florence, Italy, September 1987.

[2] T. C. Choinski and D. Sweetman, An Investigation of the Insertion of Systolic Array Technology into The Enhanced Modular Signal Processor, NUSC Technical Report 7645, 2 June 1986.

[3] N. L. Owsley, Overview of Adaptive Array Processing, NUSC Technical Report 7195, 7 June 1984.

[4] O. L. Frost, "An Algorithm for Linearly Constrained Adaptive Array Processing", Proceedings of the IEEE, Vol. 60, No. 8, August 1972, p. 926.

[5] A. Glaser and G. E. Subak-Sharp, Integrated Circuit Engineering, Addison-Wesley Publishing, Reading Massachusetts, May 1979, p. 748.

[6] C. S. Raghavendra, "Fault Tolerance in Regular Network Architectures", IEEE Micro Magazine, December 1984, p. 49.

[7] M. Annaratone, E. Arnould, T. Gross, H. T. Kung, M. Lam, O. Menzilcioglu, and J. A. Webb, "The Warp Computer: Architecture, Implementation, and Performance," IEEE Transactions on Computers, Vol. C-36, No. 12, December 1987, pp. 1523-1538.

[8] R. Bernard, "Computing at the Speed Limit", IEEE Spectrum, Vol. 19, No. 7, July 1982.

[9] D. B. Turner, R. D. Burns and H. Hecht, "Designing Micro-Based Systems for Fail-Safe Travel", IEEE Spectrum, February 1987, page 58.

[10] W. H. Berer, Standard Mathematical Tables, CRC Press Inc., Boca Raton, Florida, 1987, page 385.

[11] J. A. Fortes and C. S. Raghavendra, "Dynamically Reconfigurable Fault-Tolerant Array Processors," Proceedings of the International Conference on Fault Tolerant Computing, June 1984, page 31:4(1).

[12] H. T. Kung and O. Menzilcioglu, Virtual Channels for Fault-Tolerant Programmable Two Dimensional

Processor Arrays, Carnegie Mellon University Report, Department of Computer Science, December 1986, Defense Advanced Research Projects Agency(DOD) Contract DACA76-86-C-0023.

[13] A. Papoulis, Probability, Random Variables and Stochastic Processes, McGraw-Hill Book Company, New York, 1965, p. 58.

[14] G. H. Sandler, System Reliability Engineering, Prentice-Hall, Inc. Englewood Cliffs, N. J., 1963, p. 83.

[15] C. S. Raghavendra, A. Avizienis, and M. D. Ercegovac, "Fault Tolerance in Binary Tree Architectures," IEEE Transactions on Computers, Vol. C-33, No. 6, June 1984, p. 568.

[16] N. L. Owsley, "Systolic Array Adaptive Beamforming," International Signal Processing Conference Proceedings, Florence Italy, September 1987.

[17] F. T. Luk and H. Park, " An Analysis of Algorithm Based Fault Tolerance Techniques," SPIE International Conference Proceedings (Algorithms and Architectures for Signal Processing), Vol. 696, August 1987, pp. 222-227.

[18] G. H. Sandler, System Reliability Engineering, Prentice-Hall, Inc. Englewood Cliffs, N. J., 1963, p. 134.

[19] Erwin Kreyszig, Advanced Engineering Mathematics, 4th ed., John Wiley & Sons, New York, 1979.

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
NAVSEASYSCOM (SEA-06, -63X, CAPT Tuma, W. Kaminga, D. Howard, B. Duykers, E. Neuman, P. Mansfield, CDR B. Gallemore)	9
NOSC (Code 741, G. Byram, S. I. Chou)	2
NRL (Code 5155, R. Hilson, W. Johnson)	2
DARPA	1
CNR (OCNR-122, -13, -20)	3
SPACE & NAVAL WARFARE SYS CMD (PDW-124)	1
DTIC	12

END
DATE
FILMED
7-88
DTIC