June 1993

UILU-ENG-93-2222
CRHC-93-12

Center for Reliable and High-Performance Computing

CONFIDENCE IN PROCESSOR ARRAY OUTPUTS UNDER PERIODIC APPLICATION OF CONCURRANT ERROR DETECTION

P. P. Chen, A. N. Mourad, and W. Kent Fuchs

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-93-2222 CRHC-93-12	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois	6b. OFFICE SYMBOL (If applicable) N/A
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research National Aeronautics and Space Admin. 7b. ADDRESS (City, State, and ZIP Code) Arlington, VA Moffitt Field, CA
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 7a a	8b. OFFICE SYMBOL (If applicable)
8c. ADDRESS (City, State, and ZIP Code) 7b	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

10. SOURCE OF FUNDING NUMBERS			
PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Confidence in Processor Array Outputs Under Periodic Application of Concurrent Error Detection

12. PERSONAL AUTHOR(S) CHEN, P.P., A. N. Mourad, and W. Kent Fuchs

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	1993 June 10	32

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	fault tolerance, concurrent error detection, processor arrays, confidence in outputs, error coverage

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

Processor arrays, featuring modularity, regular interconnection and high parallelism, are well suited for VLSI/WSI implementations and specific applications with high computational requirements. Preserving data integrity can be important for certain applications; error detection is one aspect of fault tolerance. Concurrent error detection (CED) techniques, which check normal system operations, may detect transient and intermittent faults with greater probability than off-line testing methods.

This paper describes the Periodic Application of Concurrent Error Detection (PACED) technique which allows the performance costs incurred through the use of time-redundant CED in processor array architectures to be reduced. The application of CED is varied in both time and space to provide probabilistic detection of errors in processor arrays. The confidence to place on the outputs of processor arrays using PACED is studied. Formulae are derived that predict, upon error detection, the amount of output to suspect as possibly erroneous, for single processors, linear arrays, and two-dimensional mesh processor arrays. The results indicate that the error coverage can be surprisingly high when PACED is applied in processor arrays, e.g., 95checking performed 50

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL

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

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

CONFIDENCE IN PROCESSOR ARRAY OUTPUTS UNDER PERIODIC APPLICATION OF CONCURRENT ERROR DETECTION

Paul P. Chen, Antoine N. Mourad, and W. Kent Fuchs

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 West Main St.
Urbana, IL 61801

Contact: W. Kent Fuchs
Phone: 217/333-8294
FAX: 217/244-5686
Email: fuchs@crhc.uiuc.edu

Key words — Fault tolerance, Concurrent error detection, Processor arrays, Error analysis

Reader Aids —

Purpose: Present an analysis

Special math needed for explanations: Probability theory

Special math needed to use results: None

Results useful to: Fault-tolerant system designers, Processor array designers & users

Summary & Conclusions — Processor arrays, featuring modularity, regular interconnection and high parallelism, are well suited for VLSI/WSI implementations and specific applications with high computational requirements. Error detection and recovery can be important for certain applications of processor arrays. Concurrent error detection (CED) techniques, which check normal system operations, have been designed to detect errors caused by transient and intermittent faults. However, CED techniques typically suffer from costly hardware penalties or performance costs.

This paper describes the Periodic Application of Concurrent Error Detection (PACED) technique which allows the performance costs incurred through the use of time-redundant CED in processor array architectures to be reduced. The application of CED is varied in both time and space to provide probabilistic detection of errors in processor arrays. Formulae are derived that predict, upon error detection, the amount of output to suspect as possibly erroneous, for single processors, linear arrays, and two-dimensional mesh processor arrays. The results indicate that the error coverage can be surprisingly high when PACED is applied in processor arrays, e.g., 95% for checking performed 50% of the time.

This research was supported in part by the SDIO/IST and managed by the Office of Naval Research under contract N00014-89-K-0070, in part by the National Aeronautics and Space Administration (NASA) under Contract NAG 1-613, and in part by the Department of the Navy and managed by the Office of the Chief of Naval Research under Grant N00014-91-J-1283.

1. INTRODUCTION

Acronyms

CED	concurrent error detection
PACED	periodic application of concurrent error detection
PE	processing element (of an array)

Notation

M	period of CED application
N	duration of CED application
O	checking offset
$CS_{M,N}$	checking sequence array

Preserving data integrity in processor arrays that feature modularity, regular interconnection, and high parallelism, can be important for certain applications; error detection is one aspect of fault tolerance. Concurrent error detection (CED) techniques, which check normal system operations, may detect transient and intermittent faults with greater probability than off-line testing methods. Techniques such as rollback, instruction retry, and roll forward can be combined with CED for error recovery.

Use of time-redundant CED techniques can reduce the hardware overhead of error detection, but may degrade system performance. Periodic application of CED (PACED), as described in this paper, can reduce the performance degradation incurred through the use of time-redundant CED in processor array architectures [2].

Without continuous checking, undetected errors may occur. In this paper, the confidence to place on the outputs of a single processor using PACED is studied; formulae are derived that predict, upon error detection, the amount of output to suspect as possibly erroneous. In linear and two-dimensional mesh processor arrays, if detectable errors propagate, then the amount of output to suspect can be limited. The estimated PACED error coverages for the single processor and array architectures are also studied.

When a single processor uses PACED, it can be parameterized by M, the period of CED application, and N, the duration of CED application ($0 \le N \le M$). M and N govern the time distribution of CED: in any period of M computation cycles, N cycles are checked and $M - N$ cycles are unchecked. With small N/M, less performance degradation can usually be expected, but small N/M also reduces the probability of error detection.

When PACED is applied to PEs of a processor array, M and N may vary at each PE in the array. The PE checking offset O determines the initialization of each PE's first M-cycle period; varying O at each PE in an array can create patterns of checking in the array. The *checking sequence*, $CS_{M,N}$, is defined as an array of M values:

$$CS_{M,N}[r] = 1, \text{ for } 0 \le r \le N - 1,$$

$$CS_{M,N}[r] = 0, \text{ for } N \le r \le M - 1.$$

EXAMPLE 1.1: The checking sequence for $M = 5$ and $N = 2$ is $CS_{5,2} = (1, 1, 0, 0, 0)$. □

2. PACED IN A SINGLE PROCESSOR

2.1. *Error Arrival Model*

Notation

C	confidence to place on processor outputs
K	length of "cluster interval"
L	length of "undetected-errors interval"
q	Pr{particular CED technique detects error | error exists}
λ	mean error arrival rate

This study concentrates on the correctness of outputs, and thus on errors. It is assumed that faults cause errors, that errors arrive in clusters or bursts [3], that error clusters follow a Poisson arrival process with a constant mean arrival rate, and that errors within clusters are also Poisson distributed. No assumptions are made concerning the types or distributions of the faults

themselves. The following example supports this assumption.

EXAMPLE 2.1: A two-phase hyperexponential function was fit to error arrival data from one machine in a seven-unit VAXcluster system [4]. The density of the fitted distribution $\hat{f}(t)$ is $0.88(0.829e^{-0.829t}) + 0.12(0.012\ e^{-0.012t})$ (t in minutes). The fit was tested using the chi-square test and could not be rejected at the 0.28 significance level, with $r^2 = 0.99997$. □

The two-phase hyperexponential distribution function can be interpreted as exponentially-distributed errors (with parameter 0.829/min) arriving in infrequent, exponentially-distributed clusters (with parameter 0.012/min) [4].

2.2. *Confidence Analysis*

When an error is detected at a processor using PACED, the current outputs of the processor should be suspected as possibly erroneous: the detected error may be part of an error cluster and other errors in the cluster may have gone undetected. The next two subsections determine, when an error is detected, two intervals of time during which outputs should be suspected as possibly erroneous: one interval prior, and one subsequent, to the detected error. (If, upon error detection, specific action is taken either to eliminate the source of errors or to increase the amount of checking performed, then the outputs produced subsequent to the error detection need not be suspected.) These intervals are determined using two criteria, assuming that the detection of an error is independent of whether any other error is detected. 1/ Cluster Intervals: Outputs produced in intervals which bound an error cluster are suspected. 2/ Undetected-Errors Intervals: Outputs produced in intervals from the time of the current detected error backward to the first undetected error and forward to the last undetected error are suspected.

The proofs for the theorems in this section use the following lemmas.

LEMMA 2.1: In a processor using PACED with $1 \leq N \leq M$ and where the CED technique has detection probability $q \leq 1$, detected and undetected error arrivals are each exponentially distributed.

PROOF: See Appendix. □

LEMMA 2.2: The detected and undetected intracluster error Poisson arrival processes are independent.

PROOF: See Appendix. □

2.2.1. *Fault-active intervals*

When an error is detected, if no other errors were detected within K time units either prior to or subsequent to the detection, then with probability C the error cluster is contained within these cluster intervals. Assuming clusters occur infrequently, then outputs produced earlier than K time units before, or later than K time units after, the detected error can be trusted with confidence C (i.e., are correct with probability C). All outputs produced within K time units before or after the detected error should be suspected as possibly erroneous: the outputs may be used but the user should be aware that some of this output may be incorrect. Figure 2.1 illustrates the K-length cluster intervals.

THEOREM 2.1: Let a processor use PACED with $1 \leq N \leq M$ and $q \leq 1$. Upon error detection, outputs produced either before K time units prior to, or after K time units subsequent to, the detected error can be trusted with confidence C, where K satisfies:

$$K \geq -\frac{M}{N}\frac{1}{\lambda q}\ln(1-C) \ .$$

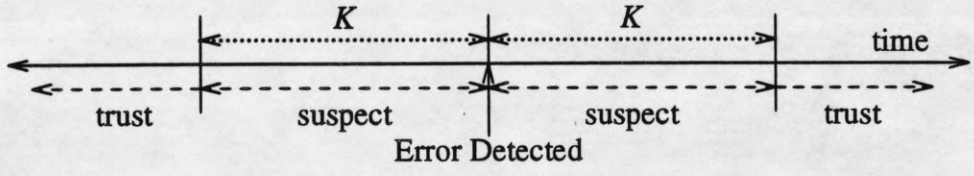

Figure 2.1. Cluster intervals of length K.

Outputs produced within K time units before or after the detected error should be suspected as possibly erroneous, since the error cluster is contained in those intervals with probability C.

PROOF: See Appendix. □

For multiple detections, cluster intervals are taken around each detection with no significance attached to overlaps. If a second error detection occurs within K of a first, then the following outputs should be suspected: those produced within K before the first detection, those produced between the two detections, and those produced within K after the second detection.

EXAMPLE 2.2: Let $q = 1$, $N/M = 0.5$, and $C = 0.99$. Using $\lambda = 0.829$ err/min (Example 2.1), if an error is detected, then outputs generated earlier than 11.1 min prior to the detected error, or later than 11.1 min after the detected error, can be trusted with a confidence of 0.99, provided no other errors are detected within 11.1 min of the detected error. All outputs produced less than 11.1 min before or after the detected error should be suspected as possibly erroneous. □

Figure 2.2(a) shows C versus λ and K, given $q = 1$ and $N/M = 0.5$. With small λ, K needs to be larger to achieve a given C. Small values of K can reach high confidence levels when λ is larger: for $\lambda \geq 0.5$ err/min, $C \geq 0.95$ can be achieved with $K > 12$ min with $N/M = 0.5$.

Figure 2.2(b) shows how C is affected either by N/M (with $q = 1$) or by q (with $N/M = 1$), given $\lambda = 0.829$ err/min. (In the expression for K given in Theorem 2.1, setting $q = 1$ and varying N/M from 0 to 1 is equivalent to setting $N/M = 1$ and varying q from 0 to 1.) Given $K = 12$ min, $N/M \geq 0.3$ (if $q = 1$) or $q \geq 0.3$ (if $N/M = 1$) suffices to give $C \geq 0.95$. This is an encouraging result: if $q = 1$, designers can use non-continuous checking (the goal of PACED) and still achieve high confidence in outputs produced near a detected error. If $N/M = 1$, then the precise value of q is not critical, for large enough K: this is encouraging as well, as it may be difficult to estimate q accurately.

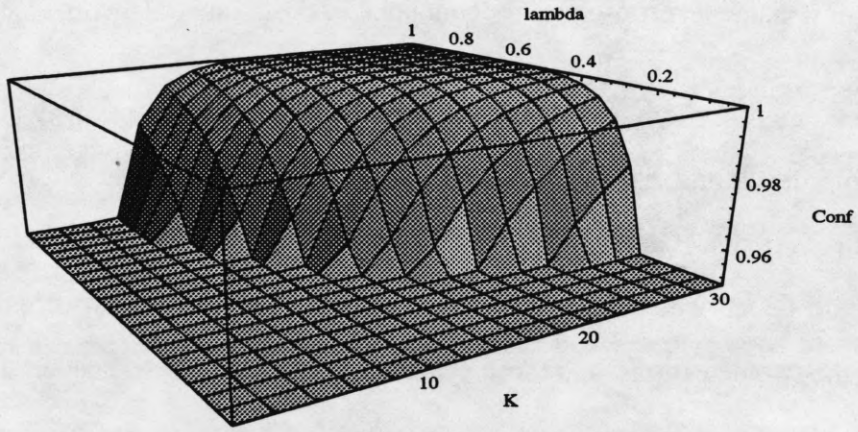

Figure 2.2(a). Cluster intervals, C vs. λ,

N=5, M=10, q=1.

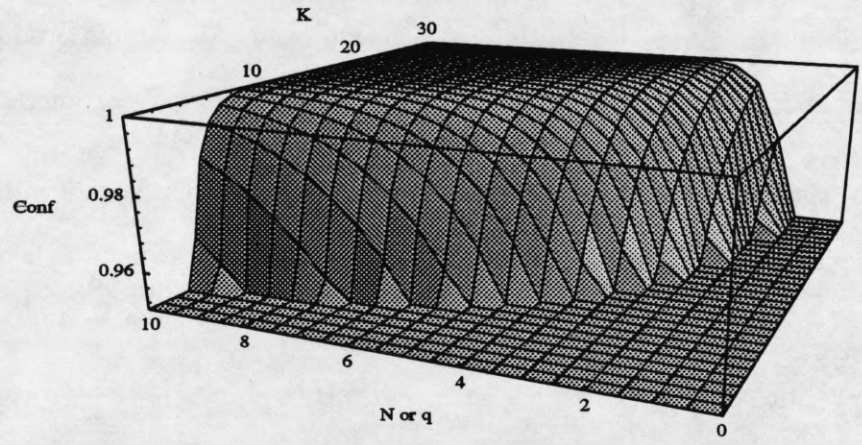

Figure 2.2(b). Cluster intervals,

C vs. N (q=1) or q (N/M=1).

2.2.2. *Undetected-errors intervals*

Theorem 2.1 determined the length of cluster intervals during which the error cluster probably existed. If, upon error detection, only those outputs since the first undetected error and until the last undetected error are suspected, then the time intervals in which to suspect outputs will be shorter. These intervals have length L. Figure 2.3 shows the relationship between K and L.

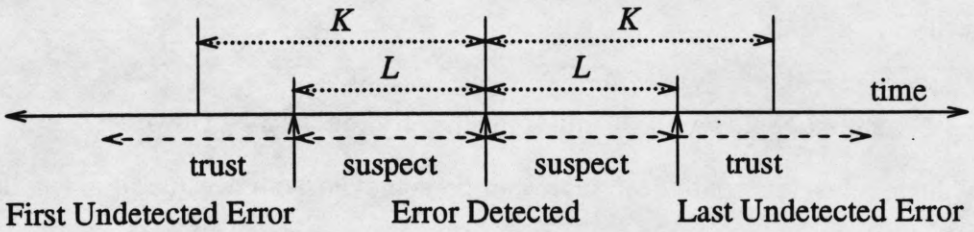

Figure 2.3. Undetected-errors intervals of length L.

THEOREM 2.2: Let a processor use PACED with $1 \le N \le M$ and $q \le 1$. Upon error detection:

Case (a): Outputs produced prior to L time units before the detected error can be trusted with confidence C; the interval extends from the detected error back to the probable time of the first undetected error.

Case (b): Outputs produced subsequent to L time units after the detected error can be trusted with confidence C; the interval extends from the detected error forward to the probable time of the last undetected error.

In both cases, the length L satisfies:

$$L \ge -\frac{M}{N}\frac{1}{\lambda q}\ln\left(\frac{1-C}{1-q\dfrac{N}{M}}\right).$$

Outputs produced within L time units before or after the detected error should be suspected as possibly erroneous.

PROOF: See Appendix. □

EXAMPLE 2.3: For the processor in Example 2.3 ($q = 1$, $\lambda = 0.829$ err/min, $N/M = 0.5$, $C = 0.99$), when an error is detected, the outputs generated prior to 9.4 min before, or subsequent to 9.4 min after, the detected error can be trusted with a confidence of 0.99, provided no other errors are detected in those time intervals. All outputs produced less than 9.4 min before or after should

be suspected as possibly erroneous. In this example, use of L-length intervals reduces the amount of output to suspect by about 15%. □

Plotting C versus L yields graphs similar in shape to Figure 2.2. However, for a given set of parameter values, a desired confidence level can be achieved with a value of L smaller than the necessary value of K.

2.3. *Error Coverage*

Notation

t_0	an error arrival time
τ	length of a time interval
$G(z, \tau)$	generating function for the number of error arrivals
a_1, a_2, b, c	parameters of $G(z, \tau)$
$\alpha, \lambda_1, \lambda_2$	parameters of two-phase hyperexponential distribution

The *error coverage* of the PACED technique is the probability that an error will be detected. When an error is detected, the backward undetected-errors interval will, in effect, "cover" all the undetected errors in that interval by casting them under suspicion. Similarly, the forward undetected-errors interval will "cover" any future undetected errors. Hence, an error will not be covered only if no other errors are detected in the time intervals of length L before and after it.

The probability that an error will not be covered is the probability that the error itself goes undetected and that no other errors are detected in the L-length intervals before and after the error. This probability will be determined using the following generating function.

LEMMA 2.3: The generating function $G(z, \tau)$, for the number of error arrivals from a two-phase hyperexponential distribution with pdf of the form $\alpha\lambda_1 e^{-\lambda_1 t} + (1-\alpha)\lambda_2 e^{-\lambda_2 t}$, in an interval $[t_0, t_0 + \tau]$ or $[t_0 - \tau, t_0]$, given that there was an arrival at t_0, is given by:

$$G(z, \tau) = \frac{a_1 - b}{a_1 - a_2} e^{-a_1 \tau} + \frac{b - a_2}{a_1 - a_2} e^{-a_2 \tau},$$

where $0 \le z \le 1$, a_1 and a_2 are the roots of the following quadratic equation in s:

$$s^2 + (\lambda_1(1 - \alpha z) + \lambda_2(1 - (1 - \alpha)z))s + (1 - z)\lambda_1\lambda_2 = 0,$$

$b = (1 - \alpha)\lambda_1 + \alpha\lambda_2$, and α, λ_1, and λ_2 are the parameters of the distribution (Example 2.1).

PROOF: See Appendix. □

Then,

$$\Pr\{\text{error not covered}\} = G\left(1 - q\frac{N}{M}, L\right)^2 (1 - q\frac{N}{M}).$$

Since Pr{error covered} = 1 − Pr{error not civered}, then:

$$\text{estimated error coverage} = 1 - G\left(1 - q\frac{N}{M}, L\right)^2 (1 - q\frac{N}{M}).$$

EXAMPLE 2.4: For a single processor using PACED, let $q = 1$, $N/M = 0.3$, $C = 0.99$, and the distribution of Example 2.1 model error arrivals, *viz.*, $\hat{f}(t) = 0.88(0.829\ e^{-0.829t}) + 0.12(0.012\ e^{-0.012t})$. This gives $L = 17.1$ min and the estimated error coverage = 94.5%. Hence, with only 30% checking, almost 95% coverage can be achieved.

Figure 2.5 plots the estimated error coverage versus N/M when $q = 1$, $M = 10$, and $C = 0.99$. The plot shows the coverage is > 95% for all values of $N/M \ge 0.4$. High coverage can be obtained with small N/M because, as the length L of the undetected-errors interval is long, many error arrivals would be expected to occur. Then, at least one of them would likely be detected, leading to the coverage of the error by casting suspicion on outputs produced at the time of the error. □

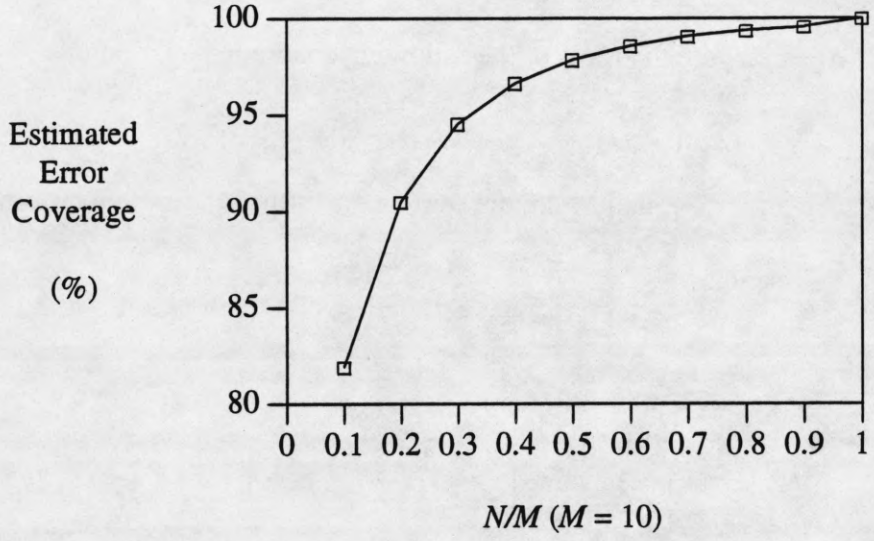

Figure 2.5. Single processor estimated error coverage,

$q = 1, \mu = 11.1$ min/err.

3. PACED IN A LINEAR ARRAY

Notation

V	number of PEs in linear array
L	error detection latency
D	error propagation distance

For a V-PE unidirectional linear processor array, inputs enter at the top and left of the array; outputs are produced at the bottom and right. Data flow only from left to right and from top to bottom. The computational activity at each PE consists of receiving input, performing a task with or without applying CED, and sending output. A *task* is a fine-grained set of data manipulations, such as a multiply-accumulate operation. Such arrays have implemented algorithms such as FFT processing [5] and image edge detection [6]. For two PEs in the array PE_i and PE_j, if $i < j$, then PE_i i's *upstream* of PE_j and PE_j is *downstream* from PE_i.

Assumptions

1. All communication channels in the array are fault-free.
2. If an erroneous array output is produced by a PE, an erroneous propagating output will also be produced and sent downstream (e.g., by using the AN-code [7]).
3. PEs are code-disjoint: erroneous inputs or state values will cause erroneous propagating outputs.

Assumption 2 ensures an error can be detected downstream if an erroneous array output is produced; Assumption 3 ensures PEs that are not checking will propagate errors.

3.1. *Error Detection Latency*

The *error detection latency* L is the number of computation cycles an error propagates until it is detected; L_{max} is the maximal value. Use of $O_i = (N_i i) \bmod M_i$ has been shown to minimize L_{max} in linear arrays [8].

LEMMA 3.1: Given a V-PE unidirectional linear processor array using PACED with $q = 1$, $M_i = M$, $N_i = N$, $1 \le N \le M$, and $O_i = (Ni) \bmod M$, it can be shown that the detection latency of an error created in the unchecked cycle r at PE_i, L_r, is $\lceil (M - r)/N \rceil$, where $N \le r \le M - 1$ and $i < V - L_{max}$. The maximum error detection latency in the array, L_{max}, is $\lceil (M - N)/N \rceil$, for all PE_i such that $i < V - L_{max}$.

PROOF: See Appendix. □

EXAMPLE 3.1: Figure 3.1 shows the checking pattern in a 7-PE unidirectional array with $M_i = 5$, $N_i = 2$, and $O_i = (2i) \bmod 5$. Computation cycles proceed vertically; each row shows the checking activity in the array during a cycle, where — and × represent a PE doing a task and a checked task, respectively. The checking pattern sets up waves of checked cycles that advance upstream over time to catch propagating errors.

An error created at PE_2 in cycle 10 (marked by *) would be detected by PE_4 in cycle 12 (labeled L_2), hence $L_2 = 2$. An error created at PE_2 in cycle 11 (marked by o) or cycle 12 (‡) would be detected by PE_3 in cycle 12 (L_3) or cycle 13 (L_4): $L_3 = 1$ and $L_4 = 1$, respectively. Finally, $L_{max} = L_2 = 2$. □

3.2. *Error Propagation Distance*

The maximum number of unchecked cycles through which a detected error could have propagated will enable determination of the amount of previously produced output to suspect as possibly erroneous from each PE in the linear array, upon error detection.

LEMMA 3.2: Given a V-PE unidirectional linear processor array using PACED with perfect detection ($q = 1$), let $M_i = M$, $N_i = N$, and $1 \le N \le M$. Using $O_i = (Ni) \bmod M$, it can be shown that an error detected by PE_i's r^{th} checked cycle, $0 \le r \le N - 1$, propagated through at most D_r unchecked cycles, where $D_r = \min(i, \lceil (M + r + 1)/N \rceil - 2)$.

PROOF: See Appendix. □

EXAMPLE 3.2: Using the array of Example 3.1 (Figure 3.1), D_0 for an error detected at PE_3 at computation cycle 12 is $\lceil (5 + 0 + 1)/2 \rceil - 2 = 1$, because PE_1 checked computation cycle 10. For an error detected at PE_3 at computation cycle 13, $D_1 = \lceil (5 + 1 + 1)/2 \rceil - 2 = 2$, because PE_0 checked computation cycle 10. □

computation cycle	PE 0	1	2	3	4	5	6
10	×	×	*	—	—	×	×
11	×	—	o	—	×	×	—
12	—	—	‡	L_3	L_2	—	—
13	—	—	×	L_4	—	—	—
14	—	×	×	—	—	—	×

Figure 3.1. Checking pattern in a 7-PE array.

3.3. *Suspected Outputs*

Upon error detection, outputs produced both in the recent past (Theorem 3.1) and the near future (Theorem 3.2) should be suspected as possibly erroneous.

THEOREM 3.1: Given a V-PE unidirectional linear array using PACED with perfect detection ($q = 1$), let $M_i = M$, $N_i = N$, $1 \le N \le M$, and $O_i = (Ni) \bmod M$. If PE_i detects an error at its r^{th} checked cycle in computation cycle c, $0 \le r \le N - 1$, then the output from PE_i in c should be suspected as possibly erroneous. Also, the outputs produced by PE_{i-k} in cycle $c - k$, for $1 \le k \le D_r$, should be suspected. All other unsuspected, previously-produced outputs can be trusted with a confidence of 1, unless a later error detection makes it necessary to suspect them.

PROOF: See Appendix. □

EXAMPLE 3.3: Figure 3.2 shows a 10-PE unidirectional linear array with $M_i = 13$, $N_i = 3$, and $O_i = (3i) \bmod 13$. Let PE_8 detect an error in cycle c (**X** in the figure). The output from PE_8 in cycle c should be suspected. Also, the outputs of PE_7, PE_6, PE_5, and PE_4 in cycles $c - 1$, $c - 2$, $c - 3$, and $c - 4$, respectively (marked by *), should be suspected as possibly erroneous. All other outputs generated up through cycle c can be trusted with a confidence of 1, unless a later error detection makes it necessary to suspect them. □

computation cycle	PE 0	1	2	3	4	5	6	7	8	9
$c-5$	—	—	—	×	×	—	—	—	—	—
$c-4$	—	—	—	×	*	—	—	—	—	×
$c-3$	—	—	×	×	—	*	—	—	—	×
$c-2$	—	—	×	—	—	—	*	—	×	×
$c-1$	—	×	×	—	—	—	—	*	×	—
c	—	×	—	—	—	—	—	×	**X**	—

Figure 3.2. Suspected previously produced outputs, 10-PE array.

Future outputs need to be suspected when an error is detected at one of the end elements PE_i, where $i \geq V - L_{max}$.

THEOREM 3.2: Given a V-PE unidirectional linear array using PACED with perfect detection ($q = 1$), let $M_i = M$, $N_i = N$, $1 \leq N \leq M$, and $O_i = (Ni) \bmod M$. If $PE_{V-L_{max}+i}$ detects an error at its r^{th} checked cycle in computation cycle c, where $0 \leq r \leq N-1$ and $0 \leq i \leq L_{max} - 1$, then the following outputs should be suspected as possibly erroneous.

Case (a) If $(r + (L_{max} - 1 - i)N + k) \bmod M \geq N$, then the outputs from $PE_{V-L_{max}+i+j}$ in cycle $c + j + k$ should be suspected, where $0 \leq j \leq L_{max} - 1 - i$; if $r < N - 1$, then $k = 0$, otherwise $0 \leq k \leq M - N$ $(r = N - 1)$.

Case (b) All output from PE_{V-1} in cycles $c + L_{max} - 1 - i$ until its next checked cycle should be suspected.

All other unsuspected, future outputs can be trusted with a confidence of 1, unless a future error detection makes it necessary to suspect them.

PROOF: See Appendix. □

EXAMPLE 3.4: Figure 3.3 shows the 10-PE linear array of Example 3.3 where $L_{max} = 4$. PE_6 has detected an error at check $r = 2$ in cycle c (marked **X**). The future outputs to suspect are those from: PE_6 in cycle $c + 1$, PE_7 in cycles $c + 1$ and $c + 2$, PE_8 in cycles $c + 2$ and $c + 3$, and PE_9 in cycles $c + 3$ and $c + 4$ (marked *), plus the detection (**X**). All other unsuspected, future outputs can be trusted with a confidence of 1, unless a later error detection makes it necessary to suspect them. □

The patterns of outputs to suspect upon error detection are static; they can be determined pre-run time and retrieved, when needed, with little run time overhead.

The amount of output to suspect upon error detection in the linear array is much less than that for the single processor: using the undetected-errors intervals in the single processor

computation cycle	PE 0	1	2	3	4	5	6	7	8	9
c	—	—	—	—	—	×	**X**	—	—	—
$c+1$	—	—	—	—	—	×	*	*	—	—
$c+2$	—	—	—	—	×	×	—	*	*	—
$c+3$	—	—	—	—	×	—	—	—	*	*
$c+4$	—	—	—	×	×	—	—	—	—	*
$c+5$	—	—	—	×	—	—	—	—	—	×

Figure 3.3. Suspected future outputs, 10-PE array.

(Example 2.3: $q = 1$, $\lambda = 0.829$ err/min, $N/M = 0.5$, $C = 0.99$), outputs from 18.8 min (9.4 min before and after an error detection) should be suspected. In the linear array, only a few tens of computation cycles' outputs need ever be suspected; with 15–20 μs cycle times in VLSI implementations [6], less than one second's output would need to be suspected. Since PEs can check other PE outputs, PACED can give high confidence in most array outputs upon error detection with non-continuous checking.

3.4. *Error Coverage*

Assuming errors occur uniformly distributed throughout the linear array, an estimate of the error coverage can be made. Since all M-cycle periods are identical, it suffices to examine the coverage of one M-cycle period. There are MV potential sites in one M-cycle period at which errors may occur: one for each PE in each cycle. Since it is assumed errors propagate unmasked through the array, only some of these sites could lead to propagation of undetected errors out of the array, if an error were to occur. The number of these sites divided by the number of potential sites gives an estimate of the error coverage.

Figure 3.4 shows the estimated error coverage for a 16-PE linear array as a function of N/M, when $M = 10$ and $q = 1$. The graph shows that even for small values of N/M, the error coverage is quite high (greater than 70% for $N/M = 0.1$). The coverage climbs quickly as N/M increases; any checking ratio ≥ 0.4 gives an estimated error coverage $\geq 95\%$. The cooperation amongst the

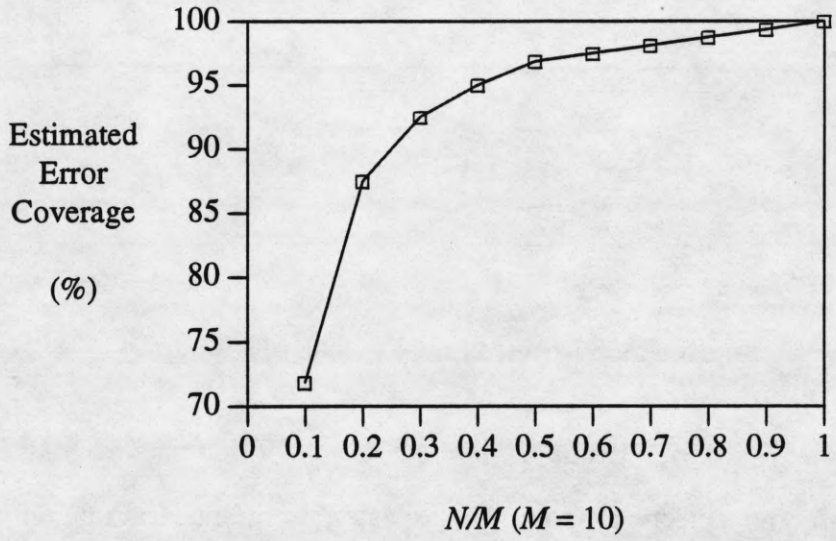

Figure 3.4. Estimated error coverage for a 16-PE linear array.

PEs that allows propagated errors to be detected affords this rise in coverage for small *N/M*. This result is promising, as it allows low checking ratios and thus, low performance cost, while still maintaining good error coverage.

4. PACED IN A TWO-DIMENSIONAL ARRAY

Notation

U	number of rows of PEs in 2-D array
V	number of columns of PEs in 2-D array
RISE, RUN	determine $O_{i,j}$, giving slope of checking pattern
L	error detection latency

For a $U \times V$ two-dimensional (2-D) mesh-connected processor array, inputs enter at the top and left of the array; outputs are produced at the bottom and right. Data may only flow from left to right and from top to bottom. Similar arrays have implemented algorithms such as matrix operations [9] and image processing [10].

For two PEs in the array $PE_{i,j}$ and $PE_{k,l}$, if $i < k$ or $j < l$, then $PE_{i,j}$ is *upstream* of $PE_{k,l}$ and $PE_{k,l}$ is *downstream* from $PE_{i,j}$. The offset $O_{i,j}$ is determined by two parameters called *RISE* and *RUN*: *RISE/RUN* gives the slope of the waves of checking in the checking pattern. The confidence analysis of the 2-D array is based on the same assumptions as in Section 3.

4.1. *Error Detection Latency*

An algorithm is used to determine L_{max} and L_r, the latency of an error created in an unchecked computation cycle r of $PE_{i,j}$, for $N_{i,j} \le r \le M_{i,j-1}$. When PACED is applied to a 2-D array, $O_{i,j} = (M_{i,j} + i + j - (U-1-i)RUN - (V-1-j)RISE) \bmod M_{i,j}$. Depending on *RISE* and *RUN*, any PE in the 2-D array may create an error that propagates undetected, so L_{max} is defined as the largest finite error detection latency.

EXAMPLE 4.1: Figure 4.1 shows a 10×10 array amidst a computation, with $M_{i,j} = 10$, $N_{i,j} = 3$, *RISE/RUN* = 2/1, and $O_{i,j} = (2i + 3j - 17) \bmod 10$. The detection latency for an error created at $PE_{2,5}$ in cycle c (**e** in the figure), when $PE_{2,5}$ performs its 6th check, is $L_5 = 2$, since both $PE_{3,6}$ and $PE_{2,7}$ detect the error in cycle $c+2$. The figure shows how the error propagates through the array (* in the figure) until detection (**X**). For this array, $L_{max} = L_N = L_3 = 3$. □

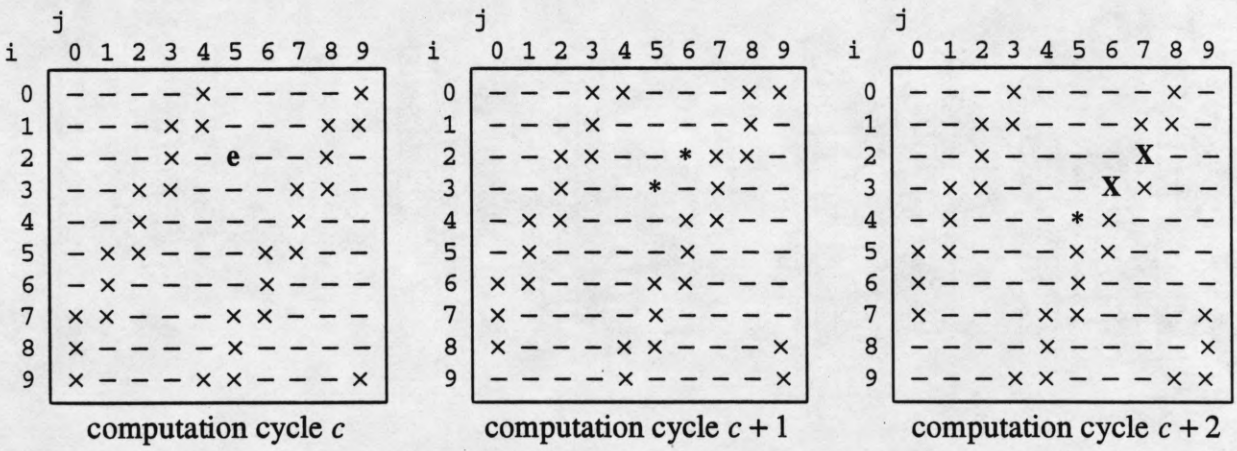

Figure 4.1. Error detection latency L_5.

4.2. *Suspected Outputs*

In the 2-D array, the outputs to suspect upon error detection are determined using algorithms that run in $O(UV \cdot N_{i,j})$ time, assuming $N_{i,j}$ is constant for all $PE_{i,j}$.

EXAMPLE 4.2: Figure 4.2 shows a 10×10 processor array using PACED with $M_{i,j} = 13$, $N_{i,j} = 5$, *RISE/RUN* = 3/1, and $O_{i,j} = (2i + 4j - 23) \bmod 13$. Each grid shows the array in one computation cycle. The outputs to suspect are marked @ (the error detection) and * (whence the error might have propagated).

If an error is detected at $PE_{9,9}$ in cycle c, its output should be suspected as possibly erroneous. Also, the outputs from the following $PE_{i,j}$ should also be suspected: $PE_{9,8}$ and $PE_{8,9}$ in cycle $c-1$; $PE_{9,7}$, $PE_{8,8}$, and $PE_{7,9}$ in cycle $c-2$; $PE_{7,8}$ and $PE_{6,9}$ in cycle $c-3$; and $PE_{4,9}$ in cycle $c-4$. All other unsuspected, previously-produced outputs can be trusted with a confidence of 1, unless a later error detection makes it necessary to suspect them. □

EXAMPLE 4.3: Figure 4.3 shows a 10×10 processor array using PACED with $M_{i,j} = 10$, $N_{i,j} = 3$, *RISE/RUN* = 2/1, and $O_{i,j} = (2i + 3j - 17) \bmod 10$. The figure is notated as in Figure 4.2.

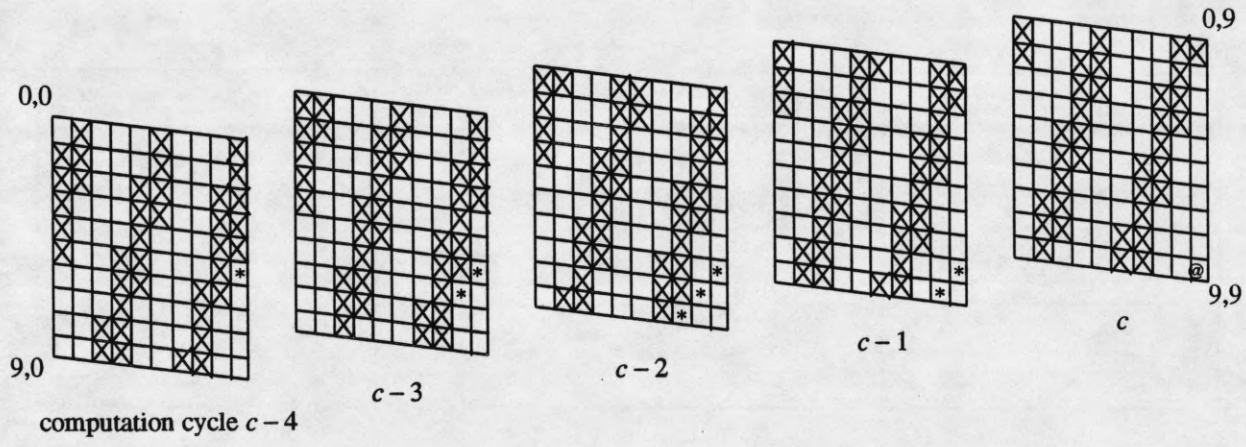

Figure 4.2. Suspected previously produced outputs, 10×10 array.

If an error is detected at $PE_{8,8}$ in cycle c (marked @) its output should be suspected as possibly erroneous. Also, the outputs from the following $PE_{i,j}$ should also be suspected: $PE_{8,9}$ and $PE_{9,8}$ in cycle $c+1$, and $PE_{9,9}$ in cycle $c+2$ (all marked by *). All other unsuspected, future outputs can be trusted with a confidence of 1 (until, of course, the next error detection). □

As in the linear array, the patterns of outputs to suspect upon error detection are static, and can be determined before run time and retrieved as needed upon error detection at run time.

An C analysis program was written to determine the minimum number of outputs to suspect for varying PACED parameters. A 20×20 array was tested, using $M_{i,j} = 15$, $N_{i,j} = 1, 2, \cdots 15$, $O_{i,j} = (15 + i + j - (19 - i)RUN - (19 - j)RISE) \bmod 15$, and $q = 1$, and varying *RISE* and *RUN*. The experiment found that only approximately one second's output needed to be suspected upon error detection. Again, this is a great improvement over the amount of suspected output in the single processor case, due to the cooperation of PEs checking other PE outputs to afford high confidence in outputs with only periodic checking.

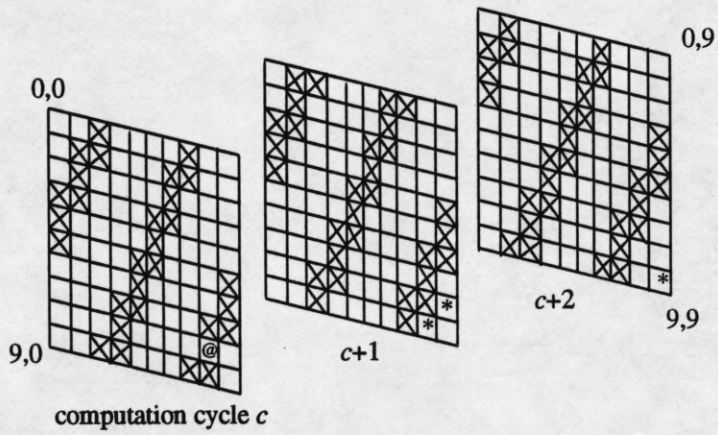

Figure 4.3. Suspected future outputs, 10×10 array.

4.3. *Error Coverage*

The error coverage in the 2-D array can be estimated if it is assumed that errors occur uniformly distributed throughout the array. Only one M-cycle period need be examined, as all other M-cycle periods are identical and have the same coverage. In one M-cycle period a $U \times V$ 2-D mesh array has MUV potential sites at which error may occur: one for each PE of the array, in each cycle. Since it is assumed that errors propagate unmasked through the array, only a fraction of the potential sites an lead to errors' propagating undetected out of the array, if an error occurs. The estimated error coverage is the number of these sites divided by the total number of potential sites.

Figure 4.4 shows the estimated error coverage for a 4×4 PE mesh array as a function of N/M, when $M = 10$ and $q = 1$. When N/M is small, the error coverage is low; but the coverage increases quickly as N/M increases: greater than 95% coverage can be achieved with N/M just 0.5 or greater. As for linear array, infrequent checking can yield high error coverage — and infrequent checking can reduce the performance cost of applying CED.

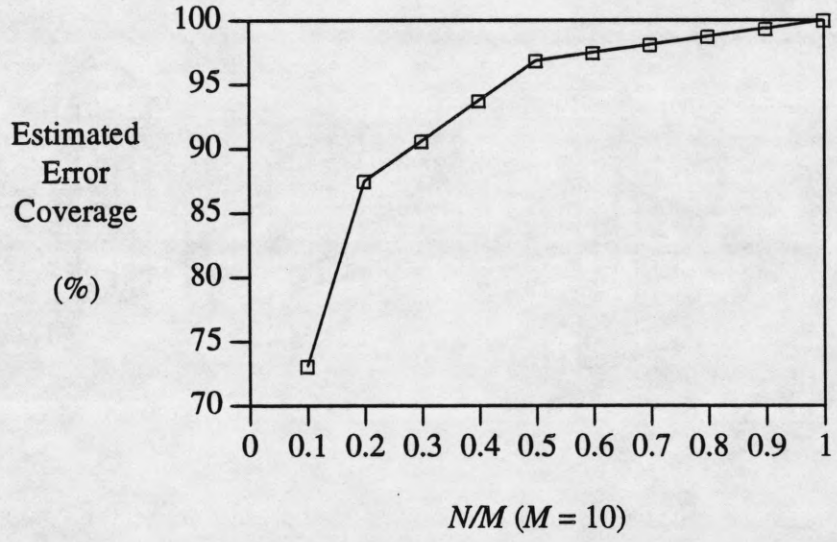

Figure 4.4. Estimated error coverage for a 4×4 mesh array.

APPENDIX

PROOF OF LEMMA 2.1: Let E_t represent the number of error arrivals in a time interval of length t, with exponentially distributed interarrival times. Let D_t and U_t represent the number of detected and undetected errors that arrive in the same time interval, respectively. If the interarrival times are exponentially distributed, this implies that the arrivals follow a Poisson process.

$$\Pr\{D_t = k\} = \sum_{n=k}^{\infty} \Pr\{E_t = n\} \binom{n}{k} \left(q\frac{N}{M}\right)^k \left(1 - q\frac{N}{M}\right)^{n-k}$$

$$= \left(\frac{q\frac{N}{M}}{1 - q\frac{N}{M}}\right)^k \sum_{n=k}^{\infty} \frac{(\lambda t)^n}{n!} e^{-\lambda t} \frac{n!}{k!(n-k)!} \left(1 - q\frac{N}{M}\right)^n$$

$$= \frac{\left(\lambda q \frac{N}{M} t\right)^k}{k!} e^{-\lambda q \frac{N}{M} t}$$

This is a Poisson distribution, with modified error arrival rate $\lambda' = \lambda qN/M$. The proof that undetected arrivals are exponentially distributed is substantially similar and results in a Poisson process with a modified error arrival rate $\lambda'' = \lambda(1 - qN/M)$. □

PROOF OF LEMMA 2.2:

$$\frac{\Pr\{U_t = k \;\&\; D_t = l\}}{\Pr\{D_t = l\}} = \frac{\Pr\{E_t = k + l\} \binom{k+l}{k} \left(1 - q\frac{N}{M}\right)^k \left(q\frac{N}{M}\right)^l}{\frac{\left(\lambda q \frac{N}{M} t\right)^l}{l!} e^{-\lambda q \frac{N}{M} t}}$$

$$= \frac{\frac{(\lambda t)^{k+l}}{(k+l)!} \binom{k+l}{k} \left(1 - q\frac{N}{M}\right)^k \left(q\frac{N}{M}\right)^l}{\frac{\left(\lambda q \frac{N}{M} t\right)^l}{l!} e^{-\lambda q \frac{N}{M} t}}$$

$$= \frac{\left(\lambda\left(1 - q\frac{N}{M}\right)t\right)^k}{k!} e^{-\lambda(1-q\frac{N}{M})t}$$

$$= \Pr\{U_t = k\}$$

Thus,

$$\Pr\{U_t = k \;\&\; D_t = l\} = \Pr\{U_t = k\} \cdot \Pr\{D_t = l\} \; . \qquad \square$$

PROOF OF THEOREM 2.1: Let D represent the detected error interarrival time. Since detected errors follow a Poisson process with parameter $\lambda qN/M$, $\Pr\{D > t\} = e^{-\lambda qNt/M}$ and $\Pr\{D \le t\} = 1 - e^{-\lambda qNt/M}$.

Let K be the length of a time interval such that $\Pr\{D \le K\} \ge C$. Then,

$$1 - e^{-\lambda q\frac{N}{M}K} \ge C$$

$$K \ge -\frac{M}{N}\frac{1}{\lambda q}\ln(1 - C) \; . \qquad \square$$

PROOF OF THEOREM 2.2: *Case (a)*: Let D and U represent the detected and undetected error interarrival times, respectively. From Lemma 2.1, both random variables are exponentially distributed with parameters $\lambda' = \lambda qN/M$ and $\lambda'' = \lambda(1 - qN/M)$, respectively.

The quantity $D - U$ represents the time between the first undetected error and the first detected error. The probability $\Pr\{D - U > t\}$ is now determined using a joint probability distribution.

$$\Pr\{D - U > t\} = \int_0^\infty \int_{x+t}^\infty \lambda'' e^{-\lambda'' x} \cdot \lambda' e^{-\lambda' y} dy dx$$

$$= \frac{\lambda''}{\lambda'' + \lambda'} e^{-\lambda' t}$$

It follows that $\Pr\{U - D \le t\} = 1 - \frac{\lambda''}{\lambda'' + \lambda'} e^{-\lambda' t}$. Let L be the length of a time interval such that $\Pr\{D - U \le L\} \ge C$. Then,

$$1 - \frac{\lambda''}{\lambda'' + \lambda'} e^{-\lambda' L} \geq C$$

$$L \geq -\frac{M}{N} \frac{1}{\lambda q} \ln\left(\frac{1 - C}{1 - q\frac{N}{M}}\right).$$

Hence, with confidence C, the first undetected error occurred within L time units before the detected error.

Case (b): Let U represent the time to the last undetected error before the next detected error, and V, the time to the next detected error.

First, the probability is determined that the last undetected error occurs in some infinitesimal time slice du at time u while the next detected error occurs in some infinitesimal time slice dv at time v, where $v \geq u$. (If it were known that $v < u$, i.e., no undetected errors occur before the next detected error, then none of the outputs produced between the two error detections would have to be suspected.)

The expression below has a term for each of the following conditions: 1) no errors are detected in interval u starting from the current error detection; 2) at least one error is undetected in interval du; 3) no errors occur in interval $v - u$; and 4) at least one error is detected in interval dv. (The variable U should be defined as the time of the last undetected error before the fault becomes inactive, but since the distribution of fault lifetimes is unknown, U is predicated instead on the next error detection. In the derivation, then, the next detection is allowed to take place at any time slice dv from u to infinity, in effect allowing the fault to become inactive.)

$$\begin{aligned} \Pr\{(u \leq \mathrm{U} \leq u + du) \,\&\, (v \leq \mathrm{V} \leq v + dv)\} &= e^{-\lambda' u}(1 - e^{-\lambda'' du}) e^{-\lambda(v-u)}(1 - e^{-\lambda' dv}) \\ &= \lambda'\lambda'' e^{-\lambda' u} du \cdot e^{-\lambda(v-u)} dv \end{aligned}$$

The terms $1 - e^{-\lambda'' du}$ and $1 - e^{-\lambda' dv}$ have been simplified using the approximation $1 - e^{-x} \approx x + o(x)$ as $x \to 0$.

Now, the probability that U is greater than some L is determined, using the joint probability just derived.

$$\Pr\{U \geq L\} = \int_{L}^{\infty}\int_{u}^{\infty} \lambda'\lambda'' e^{-\lambda' u} e^{\lambda u} e^{-\lambda v}\, dv\, du$$

$$= (1 - q\frac{N}{M}) e^{-\lambda' L}$$

L is determined such that $\Pr\{U \geq L\} \leq 1 - C$, where C, the confidence, is set arbitrarily close to 1.

$$\Pr\{U \geq L\} \leq 1 - C$$

$$(1 - q\frac{N}{M}) e^{-\lambda' L} \leq 1 - C$$

$$L \geq -\frac{M}{N}\frac{1}{\lambda q} \ln\left(\frac{1-C}{1-q\frac{N}{M}}\right)$$

Hence, with confidence C, the last undetected error occurred within L time units after the detected error. Outputs produced prior to L time units before the detected error, or subsequent to L time units after the detected error, can be trusted with confidence C; outputs produced within L units of time either before or after the detected error should be suspected as possibly erroneous. □

PROOF OF LEMMA 2.3: Let $N(\tau)$ represent the number of error arrivals in the time interval $[t_0, t_0 + \tau]$. Let S_n be the sum of n error interarrival times. Since there was an arrival at t_0,

$$\Pr\{N(\tau) = n\} = \Pr\{S_n \leq \tau < S_{n+1}\}.$$

This equation also applies to the number of arrivals in the interval $[t_0 - \tau, t_0]$. Let $F^{(n)}(\tau) = \Pr\{S_n \leq \tau\}$ be the CDF of S_n. Then,

$$\Pr\{N(\tau) = n\} = F^{(n)}(\tau) - F^{(n+1)}(\tau)$$

$$\sum_n \Pr\{N(\tau) = n\} z^n = \sum_n F^{(n)}(\tau)\, z^n - z^{-1} \left(\sum_n F^{(n+1)}(\tau)\, z^{n+1}\right)$$

$$G(z,\tau) = G_F(z,\tau) - z^{-1}(G_F(z,\tau) - 1)$$

$$= (1 - z^{-1})G_F(z,\tau) + z^{-1}$$

Taking the Laplace transform of $G_F(z, \tau)$:

$$L(G_F(z,\tau)) = \sum_n L(F^{(n)}(\tau))z^n$$

$$= \sum_n \frac{1}{s}\left(\frac{\alpha\lambda_1}{\lambda_1 + s} + \frac{(1-\alpha)\lambda_2}{\lambda_2 + s}\right)^n z^n$$

$$= \frac{1}{s\left(1 - \left(\frac{\alpha\lambda_1}{\lambda_1 + s} + \frac{(1-\alpha)\lambda_2}{\lambda_2 + s}\right)z\right)}$$

Hence,

$$G(z,\tau) = (1 - z^{-1})\, L^{-1}\left[\frac{1}{s\left(1 - \left(\frac{\alpha\lambda_1}{\lambda_1 + s} + \frac{(1-\alpha)\lambda_2}{\lambda_2 + s}\right)z\right)}\right] + z^{-1}$$

$$= \frac{a_1 - b}{a_1 - a_2} e^{-a_1\tau} + \frac{b - a_2}{a_1 - a_2} e^{-a_2\tau},$$

where a_1 and a_2 are the roots of:

$$s^2 + (\lambda_1(1 - \alpha z) + \lambda_2(1 - (1-\alpha)z))s + (1 - z)\lambda_1\lambda_2 = 0$$

and $b = (1 - \alpha)\lambda_1 + \alpha\lambda_2$. □

PROOF OF LEMMA 3.1: By design of the checking pattern, if $CS_{M,N}[r]$ is the checking activity at PE_i in some computation cycle c, then $CS_{M,N}[(r + y(N-1) + z) \bmod M]$ is the checking activity at PE_{i+y} in cycle $c + z$. With perfect detection, errors only propagate through unchecked cycles, so the proof only considers $N \le r \le M - 1$.

If an error occurs at PE_i during its N^{th} cycle, it will go undetected: this cycle is unchecked ($CS_{M,N}[N] = 0$). In the next cycle, the error will propagate to PE_{i+1} and be detected if $CS_{M,N}[(2N) \bmod M] = 1$ (i.e., if PE_{i+1} is checking). If $CS_{M,N}[(2N) \bmod M] = 0$, then the error

will propagate to PE_{i+2} in the next cycle, where it will be detected if $CS_{M,N}[(3N) \bmod M] = 1$; and so on.

The latency of detection of this error, L_N, is the number of computation cycles required for the error to reach a checked cycle. In terms of the checking sequence, L_N is the smallest integer number of N-bit hops needed to reach s such that $CS_{M,N}[s] = 1$ (i.e., $0 \le s \le N-1$) from N, where $CS_{M,N}[N] = 0$. This is a distance of $M - N$ bits.

$$L_N \cdot N \ge M - N$$

$$L_N = \lceil (M-N)/N \rceil$$

Similarly, L_{N+1}, the latency of an error created during the $N+1^{st}$ cycle (an unchecked cycle, since $CS_{M,N}[N+1] = 0$), is $\lceil (M-N-1)/N \rceil$. In general, an error created during cycle r (an unchecked cycle: $CS_{M,N}[r] = 0$) will have latency $L_r = \lceil (M-N-(r-N))/N \rceil = \lceil (M-r)/N \rceil$, $N \le r \le M-1$. Clearly, $L_N \ge L_{N+1} \ge \cdots \ge L_{M-1}$. Therefore, the maximum error detection latency, L_{max}, is L_N: $L_{max} = L_N = \lceil (M-N)/N \rceil$.

This analysis applies to all PEs in the array except the end elements, PE_i where $i \ge V - L_{max}$. At these PE_i, an error may propagate undetected out of the array since for these PE_i there are fewer than L_{max} PEs downstream. □

PROOF OF LEMMA 3.2: Let $CS_{M,N}[0]$ at PE_i detect an error in computation cycle c. The checking activity at PE_{i-1} during cycle $c-1$ is $CS_{M,N}[(-N) \bmod M]$. The maximum number of unchecked cycles through which the detected error may have propagated, D_0, is the number of computation cycles required to reach a checked cycle, minus 1, counting backwards in time. In terms of the checking sequence, $D_0 + 1$ is the smallest integer number of N-bit hops needed to reach $CS_{M,N}[r]$, $0 \le r \le N-1$, from $CS_{M,N}[0]$. This is a distance of $M - N + 1$ bits.

$$(D_0 + 1)N \ge M - N + 1$$

$$D_0 = \lceil (M-N+1)/N \rceil - 1$$

$$= \lceil (M+1)/N \rceil - 2$$

Similarly, $D_1 = \lceil (M+2)/N \rceil - 2$. In general, $D_r = \lceil (M+r+1)/N \rceil - 2$, $0 \le r \le N-1$. For PEs near the beginning of the array, there may be fewer than D_r PEs through which the error propagated. Hence, at PE_i, $D_r = \min(i, \lceil (M+r+1)/N \rceil - 2)$, for $0 \le r \le N-1$. □

PROOF OF THEOREM 3.1: By Lemma 3.2, the detected error propagated through at most D_r unchecked cycles to reach PE_i. Thus, the error was created at some PE_{i-k} in a cycle $c-(k+y)$, where $1 \le k \le D_r$ and $y = 1, 2, 3, \cdots$.

Figure A.1 shows a 10-PE array with $M = \infty$ and $N = 2$. The **X** marks an error detection at PE_5 in cycle c and the *s mark the D_r cycles through which an error may have propagated to reach PE_5.

Suppose that the error had occurred at PE_4 in cycle $c-2$, $c-3$, or $c-4$. The error would have been detected by PE_6 in cycle c, $c-1$, or $c-1$, respectively. Suppose the error had occurred at PE_3 in cycle $c-3$, $c-4$, or $c-5$. This error would have been detected by PE_6 in cycle c or $c-1$, or by PE_7 in cycle $c-1$, respectively.

In general, any error created at PE_{i-k} before cycle $c-k$ would either have been detected by cycle c (and the appropriate outputs already suspected), or gone undetected (if the error propagated out of the array). This is a result of the checking pattern, in which each PE_i performs its

computation cycle	PE 0	1	2	3	4	5	6	7	8	9
$c-6$	—	—	—	—	—	—	—	—	—	—
$c-5$	*	—	—	—	—	—	—	—	—	—
$c-4$	—	*	—	—	—	—	—	—	—	×
$c-3$	—	—	*	—	—	—	—	—	×	×
$c-2$	—	—	—	*	—	—	—	×	×	—
$c-1$	—	—	—	—	*	—	×	×	—	—
c	—	—	—	—	—	**X**	×	—	—	—

Figure A.1. Error propagation in a 10-PE array.

last checked cycle ($CS_{M,N}[N-1]$) during the same computation cycle that PE_{i-1} performs its first checked cycle ($CS_{M,N}[0]$). Hence, only the outputs from PE_{i-k} in cycles $c-k$ need be suspected, $1 \le k \le D_r$, as well as that from PE_i in c. All other unsuspected, previously produced outputs can be trusted with a confidence of 1, unless a later error detection makes it necessary to suspect them. □

PROOF OF THEOREM 3.2: By use of $O_i = (Ni) \bmod M$ in the linear array, when PE_i in cycle c performs its r^{th} checked task ($CS_{M,N}[r] = 1$), then PE_{i+y} in cycle $c+z$ will perform $CS_{M,N}[(r + y(N-1) + z) \bmod M]$.

Now, let $PE_{V-L_{max}+i}$ detect an error in cycle c by $CS_{M,N}[r]$, for $0 \le i \le L_{max} - 1$. These $PE_{V-L_{max}+i}$ are those PEs that could create errors that propagate undetected out of the array. The detected error will propagate to PE_{V-1} in cycle $c + L_{max} - 1 - i$. In that cycle, if PE_{V-1} is not checking (i.e., $(r + (L_{max} - 1 - i)N) \bmod M \ge N$), then this error will propagate out of the array and outputs from all PEs and cycles through which the error propagated should be suspected as possibly erroneous. That is, if $(r + (L_{max} - 1 - i)N) \bmod M \ge N$, then the output from $PE_{V-L_{max}+i+j}$ in cycle $c + j$ should be suspected, $0 \le j \le L_{max} - 1 - i$. If $PE_{V-L_{max}+i}$ will check at the next cycle $c + 1$, then this gives part **a)** when $r < N - 1$ ($k = 0$).

If $r = N - 1$ ($PE_{V-L_{max}+i}$ won't check in cycle $c + 1$), then as in the above case when $r < N - 1$, if $(r + (L_{max} - 1 - i)N + k) \bmod M \ge N$, then the output from $PE_{V-L_{max}+i+j}$ in cycle $c + j + k$ should be suspected, where $0 \le j \le L_{max} - 1 - i$ and $k = 0$. In addition, for each of the next $M - N$ unchecked cycles, errors may propagate out of the array. This is likely since an error has already been detected at $PE_{V-L_{max}+i}$ and the fault may still be active while that PE is not checking. The additional outputs to suspect depend upon whether PE_{V-1} is not checking when the errors arrive there. That is, for each cycle $c + k$, $1 \le k \le M - N$, if $(r + (L_{max} - 1 - i)N + k) \bmod M \ge N$, then the output from $PE_{V-L_{max}+i+j}$ in cycle $c + j + k$ should be suspected, for $0 \le j \le L_{max} - 1 - i$. This completes part **a)** when $r = N - 1$.

Once an error propagates to PE_{V-1} while it is not checking, all of its outputs until its next checked cycle should be suspected as possibly erroneous since its outputs are not checked by any other PE. Hence, all of the outputs from PE_{V-1} in cycles $c + L_{max} - 1 - i$ (the earliest that the error, first detected at $PE_{V-L_{max}+i}$ in cycle c, could corrupt PE_{V-1}) until its next checked cycle should be suspected as possibly erroneous. This gives part **b)** in the statement of the theorem.

All other unsuspected, future outputs from the array can be trusted with a confidence of 1, unless a future error detection makes it necessary to suspect them. □

ACKNOWLEDGMENTS

The authors wish to thank Robert Dimpsey, Kumar Goswami, Inhwan Lee, and Dong Tang for their help with the curve fitting performed in Section 2.1.

REFERENCES

[1] P. P. Chen, A. N. Mourad, and W. K. Fuchs, "Confidence in processor array outputs under periodic application of concurrent error detection," Technical Report CRHC-93-12, Center for Reliable and High-Performance Computing, Univ. of Illinois, Urbana, IL, May 1993.

[2] P. P. Chen and W. K. Fuchs, "Periodic application of concurrent error detection in processor arrays," *Digest of Papers, Gov't. Microcircuits Applications Conf. (GOMAC)*, vol. XV, pp. 451-454, Nov. 1989.

[3] R. K. Iyer and P. Velardi, "Hardware-related software errors: Measurement and analysis," *IEEE Trans. Software Engineering*, vol. SE-11, no. 2, pp. 223-231, Feb. 1985.

[4] D. Tang and R. K. Iyer, "Dependability measurement and modeling of a multicomputer system," *IEEE Trans. Computers*, vol. 42, no. 1, pp. 62-75, Jan. 1993.

[5] E. Swartzlander, Jr., "Systolic FFT processors," pp. 133-140 in *Systolic Arrays*. Ed. W. Moore, A. McCabe, R. Urquhart. Bristol: Adam Hilger, 1987.

[6] J. A. Vlontzos and S. Y. Kung, "A wavefront array processor using dataflow processing elements," *Proc. 1st Int. Conf. Supercomputing (Lecture Notes in Computer Science v. 297)*, pp. 744-767, Springer-Verlag, 1987.

[7] J. F. Wakerly, *Error Detecting Codes, Self-Checking Circuits, and Applications.* New York: North-Holland, 1978.

[8] Y. M. Wang, P. Y. Chung, and W. K. Fuchs, "Design and scheduling for periodic concurrent error detection and recovery in processor arrays," Technical Report CRHC-92-08, Center for Reliable and High-Performance Computing, Univ. of Illinois, Urbana, IL, May 1992.

[9] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs: Prentice Hall, 1988.

[10] R. Bayford, "The bit-serial systolic back-projection engine (BSSBPE)," pp. 43-54 in *Application Specific Array Processors*. Ed. S. Y. Kung, E. Swartzlander, Jr., J. A. B. Fortes, K. W. Przytula. Los Alamitos: IEEE Computer Society Press, 1990.