SUID-AFRIKAANSE REKENARSIMPOSIUM
SOUTH AFRICAN COMPUTER SYMPOSIUM
HOLIDAY IN PRETORIA
JULIE 1 - 3 JULY 1987

4th
Proceedings
of the
4th South African Computer Symposium

Holiday Inn, Pretoria
1 – 3 July 1987

edited by

Pieter Kritsinger
Computer Science Department
University of Cape Town
PREFACE

Computer science is an emerging discipline which is having difficulty in being recognised as a worthy member of the sciences. I will paraphrase John Hopcroft, co-winner of the 1986 Turing Award, when, during a recent interview, he said that the primary reason for the lack of recognition, is the age of our researchers. Probably not one of the researchers who presented their work at this symposium is older than 45. I know of no computer scientist in South Africa who is in a position where (s)he can affect funding priorities. As far as I know we have no representation on any of the committees of the Foundation for Research Development and for our Afrikaans speaking fraternity, none who is a member of the Akademie vir Wetenskap en Kuns. It will take time and conscious effort to establish our presence. The same is true of course for our universities. Again, with one exception, I know of no dean of a science faculty, vice-principal or principal who is a computer scientist. We consequently spend an enormous amount of time trying to explain the needs of computer science and its difficulties. I believe this symposium is a further step towards accreditation by our peers and superiors from the other sciences.

The total number of papers submitted to the Programme Committee for consideration was 34. Each paper was reviewed by three persons knowledgeable in the field it represents. Of those submitted, 23 were finally selected for inclusion in the symposium. As a result the overall quality of the papers is high and as a computer science community in Africa we can be justly proud of the final programme.

This is the fourth in the series of South African computer symposia. This year the symposium is sponsored by the Computer Society of South Africa (CSSA), the South African Institute for Computer Scientists and the local IFIP Committee. The executive director of the CSSA and his staff deserve warm thanks for handling the organisation as well as they have, while the Organising Committee provided Derrick and I with very valuable advice.

Finally I would like to express my sincere appreciation to the authors, to the members of the Programme Committee and particularly the reviewers. Without the kind cooperation of everyone, this symposium would not have taken place.

Pieter Kritzinger
SYMPOSIUM CHAIRMAN: PS Kritzinger, University of Cape Town.
SYMPOSIUM CO-CHAIRMAN: D Kourie, University of Pretoria.
MEMBERS OF THE PROGRAMME COMMITTEE

Judy Bishop, Witwatersrand University
Chris Bornman, UNISA
Hannes de Beer, Potchefstroom University
Gideon de Kock, Port Elizabeth University
Jaap Kies, Western Cape University
Derrick Kourie, Pretoria University
Pieter Kritzinger, Cape Town University
Tony Krzesinski, Stellenbosch University
Michael Laidlaw, Durban Westville University
Peter Lay, Cape Town University
Ken MacGregor, Cape Town University
Theo McDonald, Orange Free State University
Jan Oosthuizen, University of the North
Dennis Riordan, Rhodes University
Alan Sartori-Angus, Natal University
John Shochot, Witwatersrand University
Theuns Smith, Rand Afrikaans University
Trevor Turton, ISM (Pty) Ltd
Gerrit Wiechers, Infoplan.
LIST OF REVIEWERS

BERMAN Sonia
BISHOP Judy
BORNMAN Chris
CAREY Chris
CHERENACK Paul
DE BEER Hannes
DE VILLIERS Pieter
GORRINGE Pen
KIES Jaap
KOURIE Derrick
KRITZINGER Pieter
KRZESINSKI Tony
LAIDLAW Michael
LAY Peter

MacGREGOR Ken
MATTISON Keith
McDONALD Theo
RENNHACKKAMP Martin
RIORDAN Denis
SATORI-ANGUS Alan
SCHOCCHOT John
SMITH Theuns
TURTON Trevor
VAN DEN HEEVER Roelf
VAN ROOYEN Hester
VON SOLMS Basie
VOS Koos
TABLE OF CONTENTS

Keynote Address

"An Extensible System and Programming Tool for Workstation Computers." ........................................... 1
Niklaus Wirth, ETH, Zurich

Invited Lectures

"The Relationship of Natural and Artificial Intelligence." .................. not included in Proceedings.
G Lasker, University of Windsor, Ontario.

"Software Engineering: What Can We Expect in the Future?" ........ not included in Proceedings.
D Teichrow, University of Michigan, U.S.A.

Computer Languages I

"SPS-Algo·l: Semantic Constructs for a Persistent Programming Language." .................. 13
S Berman, University of Cape Town.

"Petri Net Topologies for a Specification Language." .... 25
R Watson, University of the Witwatersrand.

"Towards a Programming Environment Standard in LISP." ... 45
R Mori, University of Cape Town.

"ADA for Multiprocessors: Some Problems and Solutions." .. 63
J Bishop, University of the Witwatersrand.

Computer Graphics

"Polygon Shading on Vector Type Devices." .............. 75
C F Scheepers, CSIR.

"Hidden Surface Elimination in Raster Graphics Using Visigrams." ................................. 97
P Gorringe, CSIR.

Database Systems I

"On Syntax and Semantics Related to Incomplete Information Databases." .................. 109
M E Orlowska, UNISA.

"Modelling Distributed Database Concurrency Control Overheads." ................................. 131
M H Rennhackkamp, University of Stellenbosch.

Operating Systems

"The Development of a Fault Tolerant System for a Real-time Environment." .................. 149
M Morris, CSIR.

"A New General-purpose Operating System." .............. 161
B H Venter, CSIR.
Computer Languages II

"The Representation of Chemical Structures by Random Context Structure Grammars." ......................... 175
E M Ehlers and B von Solms, RAU.

"A Generalised Expression Structure." ......................... 189
W van Biljon, CSIR.

Computer Networks and Protocols I

"An Approximate Solution Method for Multiclass Queueing Networks with State Dependent Routing and Window Row Control." ........................................... 203
A E Krzesinski, University of Stellenbosch.

"A Protocol Validation System." ......................... 227
J Punt, University of Cape Town.

Computer Networks and Protocols II

"Protocol Performance Using Image Protocols." ............ 251
P S Kritzinger, University of Cape Town.

Artificial Intelligence

"A Data Structure for Exchanging Geographic Information." ............................................. 267
A Cooper, CSIR.

"The Design and Use of a Prolog Trace Generator for CSP." ......................... 279
D G Kourie, University of Pretoria.

Database Systems II

"An Approach to Direct End-user Usage of Multiple Databases." ........................................... 297
M J Phillips, CSIR.

"A Semantic Data Model Approach to Logical Data Independence." ......................... 329
S Berman, University of Cape Town.

Information Systems

"The ELSIM Language: an FSM-based Language for the ELSIM SEE." ............................................. 343
L du Plessis and C Bornman, UNISA.

"Three Packaging Rules for Information System Design." . 363
J Mende, University of the Witwatersrand.
Computer Languages III

"Experience with a Pattern-matching Code Generator." ... 371
M A Mulders, D A Sewry and W R van Biljon, CSIR.

"Set-oriented Functional Style of Programming." ....... 385
C Mueller, University of the Witwatersrand.

Tutorial

The use of Modula-2 in Software Engineering." ........... 399
N Wirth, ETH, Zurich.
## DAY 1

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>07h30</td>
<td>Registration and Coffee.</td>
</tr>
<tr>
<td>08h45</td>
<td>Welcoming address, President of the South African Institute of Computer Scientists,</td>
</tr>
<tr>
<td></td>
<td>Dr. G. Wiechers.</td>
</tr>
<tr>
<td>09h00</td>
<td>Invited Lecture.</td>
</tr>
<tr>
<td></td>
<td>Professor D. Teichrow, University of Michigan.</td>
</tr>
<tr>
<td></td>
<td><em>Software Engineering, ... What Can We Expect in the Future.</em></td>
</tr>
<tr>
<td>10h00</td>
<td><strong>COFFEE</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Computer Languages I.</strong> Chairman: G. Wiechers.</td>
</tr>
<tr>
<td>10h15</td>
<td>S. Berman, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td><em>SPS-Algo: Semantic Constructs for a Persistent Programming Language.</em></td>
</tr>
<tr>
<td>10h50</td>
<td>A. Watson, University of the Witwatersrand.</td>
</tr>
<tr>
<td></td>
<td><em>Petri Net Topologies for a Specification Language.</em></td>
</tr>
<tr>
<td>11h25</td>
<td>A. Mori, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td><em>Towards a Programming Environment Standard in USP.</em></td>
</tr>
<tr>
<td>11h50</td>
<td>J. Bishop, University of the Witwatersrand.</td>
</tr>
<tr>
<td></td>
<td><em>ADA for Multiprocessors: Some Problems and Solutions.</em></td>
</tr>
<tr>
<td>12h30</td>
<td><strong>LUNCH</strong></td>
</tr>
</tbody>
</table>

### 14h00

- **Computer Graphics.**
  - Chairman: D. Kourie
  - C. F. Scheepers, CSIR.
  - *Polygon Shading on Vector Type Devices.*

### 14h35

- P. Gorringe, CSIR.

### 15h15

- **COFFEE**

### 15h30

- **Database Systems I.**
  - Chairman: B. von Solms.
  - M.E. Orlowska, UNISA.
  - *On Syntax and Semantics Related to Incomplete Information Databases.*

### 16h05

- M.H. Rennhakkamp,
  Stellenbosch University.
- *Modelling Distributed Database Concurrency Control Overheads*

### 18h00

- **Cocktail Party in Cullinan Room A.**

### Operating Systems.

- Chairman: K. MacGregor.
- M. Morris, UNISA.
- *The Development of a Fault Tolerant System for a Real-time Environment.*

### 16h05

- W. van Biljon, CSIR.
- *A Generalised Expression Structure.*

### Computer Languages II.

- Chairman: J. Bishop.

### Database Systems I.

- Chairman: B. von Solms.
- M.E. Orlowska, UNISA.
- *On Syntax and Semantics Related to Incomplete Information Databases.*
<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>08h30</td>
<td>Keynote Address by Professor Niklaus Wirth, Swiss Federal Institute</td>
</tr>
<tr>
<td></td>
<td>for Technology, Zurich.</td>
</tr>
<tr>
<td></td>
<td>An Extensible System and a Programming Tool for Workstation Computers.</td>
</tr>
<tr>
<td>09h30</td>
<td>Computer Networks and Protocols I. Chairman: P.S. Kritzinger.</td>
</tr>
<tr>
<td></td>
<td>A.E. Krzesinski, University of Stellenbosch.</td>
</tr>
<tr>
<td></td>
<td>An Approximate Solution Method for Multiclass Queueing Networks</td>
</tr>
<tr>
<td></td>
<td>with State Dependent Routing and Window Flow Control.</td>
</tr>
<tr>
<td>10h05</td>
<td>J. Punt, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td>A Protocol Validation System.</td>
</tr>
<tr>
<td>10h30</td>
<td>COFFEE</td>
</tr>
<tr>
<td>11h00</td>
<td>Computer Networks and Protocols II. Chairman: A. van der Heever.</td>
</tr>
<tr>
<td></td>
<td>P.S. Kritzinger, University of Cape Town.</td>
</tr>
<tr>
<td>11h35</td>
<td>Invited Lecture by Professor G. Lasker, University of Windsor, Ontario.</td>
</tr>
<tr>
<td></td>
<td>The Relationship of Natural and Artificial Intelligence.</td>
</tr>
<tr>
<td>12h30</td>
<td>LUNCH</td>
</tr>
<tr>
<td>14h00</td>
<td>Artificial Intelligence.</td>
</tr>
<tr>
<td></td>
<td>Chairman: G. Lasker.</td>
</tr>
<tr>
<td></td>
<td>A. Cooper, CSIR</td>
</tr>
<tr>
<td></td>
<td>A Data Structure for Exchanging Geographic Information.</td>
</tr>
<tr>
<td>14h35</td>
<td>R. I. Newcombe, University of Cape Town and A. Rado, National Library</td>
</tr>
<tr>
<td></td>
<td>of Medicine, Maryland.</td>
</tr>
<tr>
<td></td>
<td>Strategies for Automatic Indexing and Thesaurus Building.</td>
</tr>
<tr>
<td>15h15</td>
<td>COFFEE</td>
</tr>
<tr>
<td>15h30</td>
<td>Database Systems II.</td>
</tr>
<tr>
<td></td>
<td>Chairman: C. Bornman.</td>
</tr>
<tr>
<td></td>
<td>M.J. Philips, CSIR</td>
</tr>
<tr>
<td></td>
<td>An Approach to Direct End-user Usage of Multiple Databases.</td>
</tr>
<tr>
<td>16h05</td>
<td>S. Berman, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td>A Semantic Data Model Approach to Logical Data Independence.</td>
</tr>
<tr>
<td>16h45</td>
<td>Open Forum with professors G. Lasker, D. Teichrow and N. Wirth.</td>
</tr>
<tr>
<td></td>
<td>Moderator: Dr. D. Jacobson.</td>
</tr>
<tr>
<td>19h30</td>
<td>Symposium Banquet in Cullinan Room.</td>
</tr>
<tr>
<td></td>
<td>Guest speaker, Dr. D. Jacobson., Group Executive: Technology,</td>
</tr>
<tr>
<td></td>
<td>Allied Technologies Limited.</td>
</tr>
<tr>
<td>Time</td>
<td>Event</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>08h00</td>
<td>Registration (Tutorial only).</td>
</tr>
<tr>
<td>08h30</td>
<td>Tutorial. The Tutorial will be given by professor Niklaus Wirth, Division of Computer Science, Swiss Federal Institute of Technology, Zurich. The use of Modula-2 in Software Engineering. Topics to be covered include:</td>
</tr>
<tr>
<td></td>
<td>What is Software Engineering?</td>
</tr>
<tr>
<td></td>
<td>Data types and structures.</td>
</tr>
<tr>
<td></td>
<td>Modularization and information hiding.</td>
</tr>
<tr>
<td></td>
<td>Definition and implementation parts.</td>
</tr>
<tr>
<td></td>
<td>Separate compilation with type checking.</td>
</tr>
<tr>
<td></td>
<td>Facilities to express concurrency.</td>
</tr>
<tr>
<td></td>
<td>Pompous programming style.</td>
</tr>
<tr>
<td></td>
<td>What could be excluded?</td>
</tr>
<tr>
<td>12h15</td>
<td>Close of Symposium.</td>
</tr>
<tr>
<td>12h30</td>
<td>LUNCH</td>
</tr>
</tbody>
</table>
The Development of a Software Fault Tolerant System for a Real Time Control Environment

M A Morris
Department of Computer Science, UNISA

Abstract

Difference equation methods are used to develop redundant (n-version) programs required for high reliability computing systems. The n-version programs are developed by exploiting the iterative nature of real time software.

The difference equation based programs are then combined with standard programs in a triple modular redundancy (TMR) hardware system by staggering the inputs to two of the modules in a TMR type system. The staggering of the inputs simulate the effect of using six different, but functionally identical program versions, within the system.

Test results indicate that a high degree of software redundancy is achieved in the described system.
Techniques for Software Fault Tolerance in a Real Time Environment

1. Introduction

In recent years the use of computers for guidance systems in aircraft, missiles and satellites have dramatically increased the demands for ultra reliable computer systems. The costs associated with failure of these guidance systems have resulted in high, sometimes prohibitive costs.

Consequently stringent performance requirements have been placed on such computer systems: for example a digital flight controller requiring a MTBF of 430 hours (a not unreasonable requirement for a modern aircraft) would require a software failure probability of

\[ 5 \times 10^{-8} \text{ errors/hr.} \]

To demonstrate the above MTBF it would be necessary to test a system for approximately 60 million hours or 6400 years [1]. Alternative methods of assuring high reliability are clearly required.

This paper describes the development of fault tolerant software specifically orientated towards guidance systems with the following characteristics:

- they are real time embedded multiprocessor systems, and
- they are used to perform continuous calculations of control and navigation algorithms (in this paper we use the term "numeric processing" for this type of software).

In addition the cost of system failure overrides all other cost considerations during the system development [2], [3].

Since all fault tolerant software systems require some form of redundancy to ensure correct operation during the presence of software faults, Section 2 of the paper outlines a method used to build functionally identical software modules (versions) from an original program.

Section 3 then describes how these functionally identical modules are incorporated into multiprocessor systems to provide increased system reliability through software redundancy.

Section 4 then introduces a technique, called the iteration shifted method, that has been developed to increase the fault tolerance of a triple redundant system.

Section 5 reports on the results of tests carried out using the iteration shifted technique.
2. Developing n=version programs.

Identical software systems fail under identical conditions within a redundant computer system. Consequently, software cannot simply be reproduced throughout a redundant computer system to ensure high reliability. True redundant software requires "different", but functionally identical software modules. Avizienis [4],[5] uses the term "n-version" programming to describe these programs.

The development of n-version software poses problems. It often requires complete duplication of the entire programming effort using different programmers, languages and wherever possible, algorithms. The problem is compounded by the fact that no metric is available to measure the "differentness" or "distinctness" [6] of functionally identical programs.

In this paper we suggest a partial solution to the problem of developing n-version programs for use in numeric processing. The constraint is that the program to be duplicated must be used in an iterative real time environment. In practice this constraint has not proved to be a problem since, as Shepherd [1] has pointed out, the majority of software faults in guidance systems can be traced back to the inaccuracies that develop during the iterative real time processing of the control algorithms.

Consider the following simple model of a real time software system:
where

\[ X \text{ is the } i \text{ inputs to the system during the } n\text{-th iteration,} \]
\[ Y \text{ is the output from the module on the } n\text{-th iteration} \]
\[ \text{nout} \]
\[ \text{and} \]
\[ P \text{ is the program implementation of the required control} \]
\[ \text{algorithm.} \]

Since we are dealing with control algorithms it can be assumed that all input and output must be discrete representations of continuous input and output data. Consequently we can give the relationship between \( X_i \) and \( Y \) for the \( n \)th iteration as

\[ Y = P(X_{n\text{out}}, ..., X_{mn}) \]

and for the \( n+1 \)th iteration as

\[ Y = P(X_{n+1\text{out}}, ..., X_{mn+1}) \]

Then as

\[ \Delta X = X_{in+1} - X_{in} \]

we can say that

\[ X_{in+1} = X_{in} + \Delta X_{in+1} \]

so that

\[ Y = P(X_{n+1\text{ln}}, ..., X_{mn+1}) \]

When arithmetic and trigonometric operations are used (as is the case in control and guidance algorithms) a relationship can be found such that

\[ Y = P(X_{n\text{ln}}, ..., X_{mn}) + Q(X_{n\text{ln}}, \Delta X_{mn}, ..., X_{mn+1}, \Delta X_{mn+1}) \]

and since

\[ Y = P(X_{n\text{ln}}, ..., X_{mn}) \]

we may write

\[ Y = Y_{n+1} + Q(X_{n\text{ln}}, \Delta X_{mn}, ..., X_{mn+1}, \Delta X_{mn+1}) \]

enabling an alternative program version to be developed using the output of the previous iteration of the program plus the
difference between the present and previous input values. Further
details of this method and the various strategies, including the
system costs associated with each strategy, are presented
elsewhere [10].

3. An algorithm for fault tolerance in real time systems

A simple Triple Modular Redundant (TMR) system [7] is given in
the following diagram:

where each M represents a single autonomous system and V
represents a voting mechanism for selecting a majority output
from the system. Assuming no single point failure, the
probability of a TMR system failure is given by

\[ P_{sysf} = P_{hf} + P_{sf} \]

or for a three version program system

\[ P = \left( P_{sysf} + P_{hf} \right) \]

where

- \( P_{sysf} \) is the probability of system failure
- \( P_{hf} \) is the probability of a hardware failure
- \( P_{sf} \) is the probability of a software failure.
In the three version system we assume for illustration that

\[ p = p = p = p \]
\[ sf \quad sf_1 \quad sf_2 \quad sf_3 \]

Estimates based on the L1011 < 500 ACS system have placed the rate of software failure in the range

\[ 1 \times 10^{-2} \quad 1 \times 10^{-3} \]

whereas the more refined techniques available for error detection and correction available for hardware designer provide a much lower failure probability [1]. Consequently, it is possible to say that

\[ \frac{p}{p_{sysf}} \approx \frac{3}{sf} \]

causing software reliability to dominate the question of system reliability.

In many cases a software TMR system does not approach the probability required in a digital flight controller. For example, based on the above values and calculations, a TMR would in the worst case have

\[ p \approx 1 \times 10^{-6} \]
\[ sysf \]

which does not meet the required software failure probability requirements of a typical digital controller.

One solution to this problem is to add yet more programs versions to the fault tolerant system. The following diagram demonstrates how six program versions may be incorporated, with appropriate voting mechanisms, into a fault tolerant system,
giving a system failure probability of

\[ p = (p + p \cdot p)^3 \]

assuming once again that

\[ p = p = p = p = p = p \]

The above system would meet the MTBF requirements. However, in practice the need to develop six distinct program versions is likely to prove problematic.

We now describe the iteration shifted method that has been developed to provide the required failure probability without the use of six program versions.

4. The iteration shifted method for fault tolerance.

Software errors that occur after a program has undergone extensive testing are called residual errors and result from the inadequacies of the software testing procedure. As Randell [8] points out: "the software is often at least an order of magnitude more complex than the hardware, presenting the tester with a great many possible states not all of which can be tested".

In this paper we argue that in general residual errors in a real time iterative system are the result of an interaction between

- a specific set of input variables
and
- an anticipated or unanticipated program state (code and variables).

We further argue that if identical well tested post production programs do not receive identical inputs on a particular iteration of the system, the probability of these identical programs failing simultaneously is small. (In practice, it has been difficult to quantify the above claim since transient software errors are difficult to assess, but an informal survey [9] does seem to support the above assumptions.)

In the fault tolerant system we describe, we have assumed that identical programs may be viewed as different program versions within a single iteration provided that the programs receive different input data during a single iteration. A prime objective of the shifted iteration system is to ensure that identical programs distributed across the TMR hardware do not receive identical inputs on any given iteration.
The structure of the proposed fault tolerant system is as follows:

- the hardware is based on the standard TMR configuration described earlier;
- each hardware module Mi contains a copy of the program (A) to be run on the system and a second version of the program (B) developed by the difference equation technique;
- the input data to module 2 (M2) is delayed by a single iteration and the input data to module 3 (M3) is delayed by two iterations;
- each module Mi contains a data output table containing and a program called a data table manager. (These items will be discussed shortly.)

The following diagram summarizes the iterative shifted system:

The input/output relationship for each program version is shown in the following table. Since the inputs to module 2 and module 3 are delayed by one and two iterations respectively, all six programs provide the same output.
For any given iteration the inputs to the three modules will be different for each module. This point is fundamental to the operation of the method since we are assuming that for each iteration two different input/program combinations have a very low probability of failing simultaneously even though the programs are identical.

The staggering of the inputs results in the following table.

<table>
<thead>
<tr>
<th>Module 1</th>
<th>Module 2</th>
<th>Module 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>A.</td>
<td>A.</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>n=2</td>
<td>n-1</td>
<td>n</td>
</tr>
<tr>
<td>(no delay)</td>
<td>(delayed 1 iteration giving Y)</td>
<td>(delayed 2 iterations giving Y)</td>
</tr>
<tr>
<td>B.</td>
<td>B.</td>
<td>B.</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>n-2</td>
<td>n-1</td>
<td>n</td>
</tr>
<tr>
<td>(no delay)</td>
<td>(delayed 1 iteration giving Y)</td>
<td>(delayed 2 iterations giving Y)</td>
</tr>
</tbody>
</table>

The system proceeds by a series of lock steps via synchronizing mechanisms built into the table managers. On completion of each iteration control is passed to the table managers in each module.
and the private tables for the output $Y$ are updated accordingly.

The table manager for each module also has the responsibility of detecting, warning and, where possible, repairing any software errors that occur after each iteration and finally sending the output value to the external environment. Because six possible valid output values are available to each table manager, the implementation of the required checking and warning procedures have proved to be straightforward. The table manager outputs the majority value contained in the data output table.

5. Tests, Results and Conclusions

5.1 The tests

A simulation of the iteration shifted system was developed and used to implement a second order transfer function. Various waveform types combined with a random fluctuations were digitized and used as input to the fault tolerant system.

The performance of the system was compared with the performance of a standard TMR system.

Four types of random fault conditions were introduced into the A and B program versions discussed in Section 4. These were:

- errors that caused a program versions to output an incorrect output value for a single iteration;
- errors that build up over a number of iterations in a particular program version;
- errors that terminate the operation of a single program version within a module and
- errors that terminate the operation of a complete module.
5.2 Results

The following are a representative sample of the results obtained using the above test procedures.

Sine + random input test. (120,000 iterations)

<table>
<thead>
<tr>
<th></th>
<th>std. simplex</th>
<th>TMR system</th>
<th>iteration shifted</th>
</tr>
</thead>
<tbody>
<tr>
<td>corrected errors</td>
<td></td>
<td>54381</td>
<td>61020</td>
</tr>
<tr>
<td>detected and flagged errors</td>
<td></td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>uncorrected errors</td>
<td>61127</td>
<td>6732</td>
<td>106</td>
</tr>
<tr>
<td>total introduced errors</td>
<td>61127</td>
<td>61127</td>
<td>61127</td>
</tr>
</tbody>
</table>

Investigation of uncorrected errors in the fault tolerant system indicate that a high proportion ("80\%") of the errors are due to mismatches between valid data in the output table. Further details of the tests are given in [9] and [10].

6. Conclusion

From the test results obtained, it does appear that the iterative shifted system provides an improved level of fault tolerance when compared to the standard TMR system. However, further work will be necessary to validate the procedures and techniques used in the system.

Work is continuing in two directions. Firstly, an attempt is being made to formalise the ideas used in generating n-version programs using difference equation techniques. Secondly, a survey is being undertaken to investigate the assumptions concerning residual errors made in this paper.

Acknowledgements

Thanks to Professor John Shepherd, Technical Director, GEC Avionics, Rochester for stimulating my interest in high reliability software for avionics applications and to Phil Cutcher, Project Manager - Guidance Systems, Giotto Space Probe Project, British Aerospace, Bristol for the interesting chats on satellite guidance systems and the hospitality. Thanks also to Professor Nic du Plooy, Unisa for allowing me the free time, a scarce resource in the department of computer science, to pursue the research outlined in this paper.
References

[1] J T Shepherd
   Report on the L1011 < 500 ACS System
   Report for the technical committee of the A.R.B.
   Technical Note: ARB/80/12 Cranfield Institute of Technology

   Redundancy Management Techniques for Space Shuttle Computers
   IBM Journal of Research and Development 20, January 1976

   Development of On-board Space Computer System 20
   IBM Journal of Research and Development 20, January 1976

[4] A Avizienis
   The N-Version Approach to Fault Tolerant Software
   IEEE Transactions on Software Engineering
   Vol SE-11 No 12, December 1985

[5] L Chen, A Avizienis
   N-version Programming: A fault tolerant approach to
   reliability of software operation.
   The Eighth Annual International Symposium on Fault
   Tolerant Computing, June 1979

   Distinct software: An approach to reliable computing.
   Proceedings of the 2nd USA-Japan Computing Conference.
   Tokyo 1975

[7] T Anderson, P A Lee
   Fault Tolerance - Principles and Practice

[8] B Randell
   System Structure for Software Fault Tolerance
   IEEE Transactions on software Engineering
   Vol SE 1 No 2, June 1975

   Aspects of fault tolerant software
   Technical note FT/2, Dept of CS, Unisa, October 1986

    An Approach to the Design of Fault Tolerant Software
    MSc thesis, Cranfield Institute of Technology, 1982