A resolution test chart featuring several groups of patterns. Each group consists of five vertical lines of varying thicknesses and five horizontal lines of varying thicknesses. The patterns are labeled with numbers: 1.0, 1.1, 1.25, 1.4, 1.6, 1.8, 2.0, 2.2, 2.5, 2.8, 3.2, 3.6, 4.0, 4.5, 5.0, 5.6, 6.3, 7.1, 8.0, and 9.0. The patterns are arranged in a grid-like fashion, with the numbers increasing from top-left to bottom-right.

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

On Reconfigurability of Some Regular Architectures

by

Amiyaranjan Nayak

A thesis submitted to the
Faculty of Graduate Studies and Research
in the partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems & Computer Engineering
Carleton University
Ottawa, Ontario, Canada

April 1991

©Amiyaranjan Nayak 1991

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadienne

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-68860-2

Canada

The undersigned hereby recommend to the Faculty of Graduate Studies and Research
acceptance of the thesis

On Reconfigurability of Some Regular Architectures

submitted by Amiyaranjan Nayak in partial fulfillment of the requirements for the
degree of Doctor of Philosophy.

Niche Lutho

Mortose N. N. N.

Thesis Supervisor(s)

Chamawant
External Examiner

V. S. S. S.

Chairman,
Department of Systems and
Computer Engineering

Department of Systems & Computer Engineering
Faculty of Engineering
Carleton University
April 1991

Abstract

Fault tolerance is the survival attribute of computer systems; when a system is able to recover automatically from internal faults without suffering an externally perceivable failure, the system is said to be fault-tolerant. A common approach for achieving fault tolerance is through the incorporation of redundancy.

The effectiveness of a redundancy-based fault tolerance scheme cannot simply be measured by the amount of redundancy built into the system, but also by the ability of the system to reconfigure in the presence of faults and to carry on its operation. Therefore, redundancy and reconfigurability are the two key aspects of fault tolerance that are directly related to reliability and low yield problems. Although many researchers have come forward to present their views in solving problems of low yield, the importance of the spare resources and fault distribution and their impact on reconfigurability have not been thoroughly investigated. This thesis studies the characteristics of catastrophic fault patterns; that is, patterns of faults whose occurrence can have catastrophic effects on the system and cannot be overcome regardless of how "intelligent" is the employed reconfiguration algorithm. Limits are established on the reconfigurability of regular networks such as one-dimensional systolic arrays, two-dimensional systolic arrays, and ring topologies.

Acknowledgement

I would like to express my sincere gratitude to my thesis supervisors, Prof. Nicola Santoro and Prof. Morteza Niktash for their constant help and guidance throughout the process of this research work. I am very grateful to Prof. Nicola Santoro for the time he spent working with me to ensure that I would be proud of the final results. His feedback over the years has been a major force behind my development as a researcher, as a writer and as an individual. I sincerely thank Prof. Morteza Niktash for his useful feedbacks, suggestions, and encouragement throughout the development of this thesis. His suggestions for improvements, right until the end, substantially strengthened the thesis.

I am grateful to my family members who showed their support in my efforts with continued interest and encouragement.

I thank all my thesis committee members (Prof. S. R. Das, Prof. F. Dehne, Prof. R. Goubran, Prof. J. Knight, and Prof. C. V. Ramamoorthy) for their wise remarks. In particular, I wish to express my gratitude to Prof. S. R. Das for his terrific help. He has been a valued advisor on all technical and professional matters. I also thank Prof. C. V. Ramamoorthy for his role as the external examiner. He provided useful comments on the dissertation. I have benefited a lot from the discussions I had with him during his stay in Ottawa.

I appreciate the help of Mr. Vincenzo Acciaro and Dr. Richard Tan in many technical matters. I am very grateful to them.

I would also like to thank all my friends and well-wishers, particularly, Patrick Hui, Jim Brown, Kerry Zielke and Jay Doshi for their moral support over the years.

Table of Contents

Chapter 1 Introduction	1
1.1 The Framework	1
1.2 The Results	6
1.3 Importance & Applicability of the Results	7
1.4 Outline of the Thesis	10
Chapter 2 Background	12
2.1 Introduction	12
2.2 Background on Fault Tolerance Techniques	13
2.2.1 Fault Tolerance Terminologies	13
2.2.2 Techniques of Fault Tolerance	14
2.3 Background on Regular Architectures	17
2.3.1 Systolic Arrays	17
2.3.2 Ring Networks	19
2.3.3 Tree Topologies	20
2.4 Fault Tolerance in Regular Architectures	21
2.4.1 Fault-Tolerant Systolic Arrays	22
2.4.2 Fault-Tolerant Ring Networks	27
2.4.3 Fault-Tolerant Tree Topologies	31
2.5 Chapter Summary	33
Chapter 3 Characterization of Catastrophic Fault Patterns	34
3.1 Introduction	34
3.2 Preliminaries	35
3.3 Deadly Integer Sets and Catastrophic Fault Patterns	38
3.3.1 Cardinality of Catastrophic Fault Patterns	42
3.3.2 Width of Catastrophic Fault Patterns	43

3.3.3	Effect of G on W_F	46
3.4	Properties of Catastrophic Fault Patterns	52
3.4.1	Translation Property	52
3.4.2	Symmetric Property	52
3.4.3	Closure Property	54
3.5	Chapter Summary	57
Chapter 4 Generation and Enumeration of Catastrophic Fault Patterns		58
4.1	Introduction	58
4.2	Construction of Widest Fault Pattern	59
4.2.1	Unidirectional Links	60
4.2.2	Bidirectional Links	63
4.3	Enumeration of Catastrophic Fault Patterns	65
4.3.1	Bidirectional Links	66
4.3.2	Unidirectional Links	69
4.4	Generation of Catastrophic Fault Patterns	71
4.4.1	Bidirectional Case	71
4.4.2	Unidirectional Case	72
4.5	Determining if a Pattern is Catastrophic	73
4.6	Chapter Summary	75
Chapter 5 Limits to Reconfigurability of Systolic Architectures		76
5.1	Introduction	76
5.2	Preliminaries	77
5.3	One-Dimensional Systolic Arrays	81
5.4	Two-Dimensional Systolic Arrays	86
5.5	Chapter Summary	91
Chapter 6 Limits to Reconfigurability of Ring Topologies		92
6.1	Introduction	92
6.2	Preliminaries	93

6.3	Fault-Tolerance Analysis of Chordal Rings	98
6.4	Limits to Reconfigurability	104
6.5	Chapter Summary	106
Chapter 7 Summary and Conclusions		107
7.1	Summary of Main Contributions	107
7.2	Open Problems	108
7.3	Applications	109
7.4	Areas of Future Work	111
Glossary		112
Bibliography		115

List of Tables

3.1	Base set solutions for $G = \{1, 3, 7, 10\}$	56
-----	--	----

List of Figures

1.1	A block fault of size 4 for link $\{1,2,4\}$	4
1.2	A fault pattern ($g_k = 4$) for links $\{1,2,4\}$	5
2.1	A systolic array of processors	18
2.2	A ring network of processors	20
2.3	A tree of processors	21
2.4	Interstitial redundancy scheme	23
2.5	Single cell bypass (SCB) scheme	27
2.6	A daisy loop network	29
2.7	20-node optimal forward loop backward hop network	30
2.8	A fault-tolerant hyper tree	32
3.1	A linear systolic array of processors	35
3.2(a)	A fault pattern of size 4	36
3.2(b)	A different fault pattern of size 4	37
3.3	A disconnected array	37
3.4	Distribution of "x"s for unidirectional $G = \{1, g_k\}$	44
3.5	Distribution of "x"s for bidirectional $G = \{1, g_k\}$	46
3.6	A catastrophic fault pattern for $G = \{1, 2, 12\}$	48
3.7	A catastrophic fault pattern for $G = \{1, 7, 9\}$	49
3.8	A catastrophic fault pattern for $G = \{1, 5, 10\}$	49
3.9	A catastrophic fault pattern for $G = \{1, 7, 12\}$	50
3.10	A catastrophic fault pattern for $G = \{1, 2, 12\}$	51
3.11(a)	A symmetric fault pattern for $G = \{1, 3, 7, 10\}$	53
3.11(b)	A symmetric fault pattern not catastrophic for $G = \{1, 3, 7, 10\}$	54
4.1	A catastrophic fault pattern for unidirectional $G = \{1, 2, 4, 8\}$	61
4.2	A catastrophic fault pattern for bidirectional $G = \{1, 3, 9\}$	64
4.3	All catastrophic fault patterns for bidirectional $G = \{1, 5\}$	69
4.4	Fault pattern # 1	74

4.5	Boolean matrix for fault pattern # 1	74
4.6	Fault pattern # 2	74
4.7	Boolean matrix for fault pattern # 2	75
5.1	1-Dimensional array of PEs	78
5.2	2-Dimensional array of PEs	79
5.3	Three basic models	80
5.4	1-D array of PEs with a block fault	82
5.5	1-D array with block fault in input side	82
5.6	1-D bidirectional array with a blocked I/O	83
5.7	1-D bidirectional array with $< 2g_k$ faults	83
5.8	1-D array with alternating block faults	84
5.9	1-D array with alternating fault patterns	86
5.10	$\sqrt{K} \times \sqrt{K}$ cluster faults in 2-D array	87
5.11	Cluster faults in 2-D array	88
5.12	g_k Non-contiguous faulty columns	88
5.13	2-D array with both faulty rows and columns	89
5.14	Fault bands in a bidirectional 2-D array	90
6.1	A chordal ring	93
6.2	A blocking decomposition $\mathcal{F} = \{F_1, F_2\}$	96
6.3	A blocking fault pattern	97
6.4	Ring showing α blocking patterns	105

List of Symbols

F	A fault pattern
G	A link configuration
g_k	The length of the longest link in G
W_F	The width of a fault pattern
Z	The set of integers
Z^+	The set of positive integers
$Z[x]$	The set $\{1, 2, \dots, x\}$ for $x \in Z^+$
$\ X\ $	The cardinality of the set X
$R(x)$	The right neighbor of x
$L(x)$	The left neighbor of x
$N(x)$	The neighbors of x
D	The set of dead integers
D^i	The set of dead integers at instant i
A^i	The set of integers that are not dead at instant i
$d(x)$	The time of death of integer x
σ	A sequence of integers, infinite at both ends
$s(x)$	The successor of integer x in σ
W	A Boolean matrix
μ	A string on the alphabet $\{ (,), * \}$
$\langle_f$	An open parenthesis with weight f
χ	A string on the alphabet $\{ \langle_f, \rangle, * : f \in Z \}$
$c(i)$	The cost associated with element i
Σ_q	The set of well-formed strings on $\{ (,) \}$ of length $2q$
$\Upsilon_{p,q}$	The set of all well-formed strings on $\{ \langle_f, \rangle \}$ of length p which contains exactly q open parentheses
Ξ_{g_k}	The set of all well-formed strings on $\{ (,), * \}$ of length g_k
Ω_{g_k}	The set of all well-formed strings on $\{ \langle_f, \rangle, * \}$ of length g_k

Γ_G	The number of catastrophic fault patterns for G
$\mathcal{P}$	The set of nodes in a ring network
$\mathcal{E}$	The set of edges in a ring network
C	The chordal ring
C^+	The chordal ring whose links have clockwise orientation
C^-	The chordal ring whose links have counter clockwise orientation
$\mathcal{F}$	A decomposition of the fault pattern F
$\delta(F)$	The size of the smallest blocking decomposition of F
S_i	A set of consecutive non-faulty nodes
F_i	A set of consecutive faulty nodes
$\mathcal{X}_{a,b}$	A subsequence $S_a, F_{a+1}, S_{a+1}, \dots, F_b, S_b$ in ring C
$\pi(\mathcal{X}_{a,b})$	The number of fault-free nodes in $\mathcal{X}_{a,b}$
Π	The largest possible subsequence of type $\mathcal{X}_{a,b}$
Φ_i	The number of blocks of length g_i
Φ_{max}	The number of blocks of length g_k or more
$d(x, y)$	The distance between nodes x and y in ring C

Chapter 1

Introduction

1.1 The Framework

Fault tolerance is the survival attribute of computer systems; when a system is able to recover automatically from internal faults without suffering an externally perceivable failure, the system is said to be fault-tolerant. A common approach for achieving fault tolerance in VLSI-based systolic architectures is through the incorporation of redundancy. The popularity of this approach rests on the fact that, with modern technology, it is now possible to incorporate a large degree of redundant processing elements (PEs) and additional circuitry into a single chip. The redundant PEs are used to replace any faulty PE(s); the redundant links are used to bypass the faulty PEs and reach the redundant PEs used as a replacement. Most of the decisions made at design time with regards to fault tolerance are therefore focused on two particular aspects: amount of redundancy and reconfiguration technique. In the following, the VLSI design will be used as the leading example.

Intuitively, a system incorporating a large number of redundant PEs and “long” redundant links should be able to tolerate a large number of PE failures. A long redundant link can bypass a large block of consecutive faulty PEs. However, long wires are not always possible in such systems due to layout constraints. A long wire will introduce larger propagation delays which in turn might create synchronization problems and become the limiting factor in the performance of the system. Furthermore, note that to increase the number of redundant PEs in a chip requires an extra

overhead of interconnections and switching circuitry which implies a higher likelihood of failure. Provided that these technical difficulties can be successfully overcome, it still seems natural to assume that a very high degree of tolerance can be achieved by simply providing a sufficiently large number of spare PEs with a large number of long connections.

The effectiveness of the approach of using structural redundancy to increase fault-tolerance clearly does not depend solely on the amount of redundancy. In fact, the availability of a large number of redundant PEs is useless if these PEs cannot be successfully employed to replace the faulty ones. Thus, a main measure of fault-tolerance in such redundant arrays is the *reconfiguration capability* (or reconfiguration effectiveness); that is, the ability to map faulty elements to spares (using bypass links) while preserving the high degree of regularity and locality of reference required by the system to perform correctly.

It is therefore not surprising that a large amount of research has been devoted to the design of reconfiguration algorithms for redundant arrays as well as proposing redundant architectures which facilitate the reconfiguration process [12,15,19,24,33,42,43,55,57,67,71,74,75,92,94,97,108,111,115,120,133]. Related work has focused on the study of low yield problem [27,28,47,56,129], i.e., on improving spare utilization.

The assessment of a design should obviously be done with respect to both fault tolerance of the system and (degraded) performance of the system in the presence of failures. Typically, the only evaluation of the designs proposed in the literature is carried in terms of statistical measures (e.g., availability, mean time between failures, etc.) of the reliability. No measurement of the effectiveness of the fault tolerance scheme (i.e., redundancy and reconfiguration) used in the design is typically given, nor is any measurement of the degradation in computational effectiveness of the resulting system once failures occur. An important point to be raised here is that no claim is made, for any of the designs, on guaranteed fault tolerance such as "this design will always tolerate up to k faults regardless of how and where the faults occur". In fact,

such a claim would be untrue for most designs even for very small values of k . The issue of guaranteed fault tolerance will be addressed at great length in the rest of the thesis.

An important (but often neglected) consideration is that the knowledge about the application field of the system can be usefully incorporated in the design process to obtain improved fault tolerance and performance. In particular, knowledge of the types and distribution of the faults, if properly employed, can make the design resilient to these faults in the application domain; furthermore, the knowledge of the application field can significantly reduce the amount of testing needed to access the design. It is therefore important to acquire and integrate in the design the knowledge of the faults that occur, the fault distribution, the fault density, the uniformity of the fault distribution, etc. [85]. These data should be used (together with the circuit details) to assess fault tolerance.

A computer architecture (or network) can be characterized by its topological properties. It is called regular if the underlying graph is regular; that is, it is connected and every node has the same degree. Because of their properties, the regular architectures are very common (as described in Chapter 2) and amenable to analysis unlike the irregular ones. The focus of this thesis will be on the regular architectures.

The effectiveness of using redundancy to increase fault tolerance in a regular architecture clearly depends on both the amount of redundancy and the reconfiguration capability of the system. It does however depend also on the distribution of the faults in the system. In fact, faults occurring at strategic locations in a regular architecture may have catastrophic effect on the entire structure and cannot be overcome by any amount of clever design. For a given design, it is not difficult to identify a set of elements whose failure will have catastrophic consequence.

Consider, for example, an array of PEs. For a linear array with no link redundancy, a single PE fault in any location is sufficient to stop the flow of information from one side to the other. This fact holds irrespective of the amount of PE redundancy.

One can make similar observations in situations where there is some form of link redundancy. It is again obvious that a linear array with bypass links $\{g_1, g_2, \dots, g_k\}$ cannot tolerate g_k PE faults if they occur in a block (or cluster). In order to tolerate a block fault of size g_k , one must have some bypass links of length at least $g_k + 1$. This can be illustrated by the following simple example.

Example 1: Figure 1.1 shows a linear array in which each PE is connected to its immediate neighbors and to the PEs at distance 2 and 4. The faulty PEs are shadowed. Clearly, there is no single link or a combination of links that would allow us to bypass the faulty area. In other words, the structure is logically separated for all practical purposes.

Figure 1.1: A block fault of size 4 for links $\{1, 2, 4\}$

It is interesting to observe that the number of faults sufficient to partition the array is a function only of the length of the longest bypass connection, g_k , but not of any intermediate bypass connections. In practice, the probability of block faults of size g_k or higher is relatively small. A natural question is whether the bypass links $\{g_2, \dots, g_k\}$ will be sufficient to cope with g_k faults if they do not occur in a block. Unfortunately, there exist certain fault distributions involving g_k faults, not all in a block, for which the structure will be unusable. The following example demonstrates that.

Example 2: Once again consider the same linear array as in the previous example with the same number of faults, but a different fault distribution (see Figure 1.2). In this case, it is not difficult to verify that any link from a non-faulty PE outside the faulty region either leads to a faulty PE directly or to a non-faulty PE within the faulty region which then eventually leads to a faulty PE.

Figure 1.2: A fault pattern ($g_k = 4$) for links $\{1, 2, 4\}$

These examples illustrate dramatically the fact that, regardless of the amount of redundancy used, there are inherent limits to the fault tolerance of every design. Therefore, the knowledge of these limits is essential in the assessment of the fault tolerance effectiveness of any given design.

The main objective of this thesis is to study the characteristics of catastrophic fault patterns; that is, patterns of faults whose occurrence has catastrophic effects on the system and cannot be overcome regardless of how “intelligent” is the employed reconfiguration algorithm. This study sheds some light on the inherent limits to reconfigurability of various regular architectures as well as on the factors limiting the effectiveness of the redundancy approach. The main theoretical results of this thesis are discussed in the next section; the practical implication within the framework discussed earlier as well as in different application domains will be presented in Section 1.4.

1.2 The Results

The main contribution of this thesis is a complete characterization of catastrophic fault patterns for linear arrays. These results, in turn, are used to study reliability and limits to reconfigurability of regular networks such as one-dimensional systolic arrays, two-dimensional systolic arrays, and ring topologies.

The catastrophic fault patterns for linear arrays are studied by treating the linear array as a set of consecutive positive integers. The results are obtained by reducing the problem of characterization of catastrophic fault patterns to the problem of identifying a special family of sets of integers, called deadly sets. Several important properties of deadly sets are established which, in turn, express important properties of catastrophic faults. For example, it is shown that the cardinality of a catastrophic fault pattern does not depend on the number of bypass links available but only on the length of the longest one; achievable bounds are established on the width of catastrophic fault patterns; results expressing the sensitivity of the width to the amount of bypass links available are proved. Special classes of catastrophic fault patterns are analyzed in detail and seen to have an interesting underlying algebraic structure.

In this thesis, the catastrophic fault patterns are characterized also in a constructive way. Algorithms are presented for generating and enumerating such patterns for any given link configuration, both in the unidirectional and in the bidirectional case. The exact number of catastrophic fault patterns for bidirectional links is determined; both upper and lower bounds are established in the case of unidirectional links. To prove these results, the original problem is mapped into a more general (and not previously studied) version of the well-known problem of counting the number of well-formed strings of parenthesis.

All the established properties of catastrophic fault patterns and their windows are used to study limits to reconfigurability of one- and two-dimensional arrays. The catastrophic fault patterns are employed as means to derive fault-tolerant attributes

in terms of the minimum number of faults sufficient to cause a catastrophic failure. The minimum number of faults required to have an impact on reconfigurability is derived for different modes of computations. In particular, bounds for impossibility of reconfiguration are established for systems with 1) unidirectional links, 2) bidirectional links where the I/O can be at either side of the array, and 3) even for the case where each PE can act as an I/O port. The negative results hold even if faults are detectable, and transient faults never occur: they draw an impossibility map for reconfigurability in both one- and two-dimensional systolic architectures.

In the case of redundant ring topologies, both positive and negative results are established. The positive results refer to conditions for the reconfigurability of the structure in the presence of faults. Unlike for array topologies, very little is known about this problem; the only result appearing in the literature [132] is actually shown to follow as a corollary of one of the results presented here. All the proofs are constructive and, hence, reconfiguration techniques can be easily derived. The negative results are analogous to the ones for linear arrays; the bounds are obviously different. Once again, these results describe some of the inherent limits to reconfigurability for such topologies.

The results of this thesis, some of which have appeared in a preliminary form in [2,3,87,88,89], sheds light on understanding of the real fault tolerance (or better yet, fault intolerance) of redundant regular architectures.

1.3 Importance & Applicability of the Results

The contributions of the thesis are mostly of theoretical nature, and they all stem from a mathematical characterization of minimal catastrophic fault patterns. There are both theoretical and practical importance of the results presented here.

From a theoretical viewpoint, the results express precise bounds on the computability in the presence of failures in large classes of regular architectures employing

redundancy and reconfiguration techniques. In particular, limits to reconfigurability are established for linear arrays, two- dimensional arrays, and ring networks under several models of communication. These results are either the first known (as in the case of one- and two-dimensional arrays) or extend and generalize the existing results (as in the case of ring networks) for the considered class of systems.

Furthermore, in addition to the analysis of the class of "deadly integer sets", an interesting and unsuspected relationship between catastrophic fault patterns and well- formed sequences of weighted parentheses has been brought to light. The results obtained on these sequences extend in a non-trivial way the existing results for the unweighted case. Besides its theoretical interest, this correspondence has led to the design of optimal algorithms for determining whether a fault pattern is catastrophic, and for generating all catastrophic fault patterns for a given link configuration.

From a practical viewpoint, the results allow to prove some answers to questions about the "guaranteed" level of fault tolerance of a design, something not previously considered. Guaranteed fault tolerance indicates positive answers to questions such as

- a) will the system withstand up to k faults always regardless of how and where they occur?
- b) will the system withstand up to k faults with probability ϵ regardless of how and where the faults occur?

The results in this thesis provide a set of tools which can be employed in

1. assessing the fault tolerance effectiveness of a design; this can be done by specifying the minimum number of faults which the design cannot be guaranteed to withstand,
2. testing whether a design meets the specified fault tolerance requirements; this can be achieved by comparing the requirements with the ones derived using the properties of the catastrophic fault patterns, and

3. determining redundancy requirement for the designer to meet a desired level of fault tolerance; this can be done by determining the minimal link configuration for which no catastrophic fault patterns exist below the specified amount of failure.

Furthermore, the results presented here can help to usefully incorporate knowledge of the application field into the design process. In particular, knowledge of the type and distribution of faults occurring in the application field can be used to determine for which design those patterns are catastrophic; thus, the designer can remove those designs from further consideration (even though, without that knowledge, they might have been viable choices).

Other possible practical application is in post-manufacturing assessment phase. Since in VLSI-based systems faults are introduced by the manufacturing process, the guaranteed level of fault tolerance of the resulting system might differ from that of the original design; the results presented here could be useful in assessing the new level of fault tolerance and thus aiding in the decision making process of whether to accept or reject a fabricated chip.

The results for arrays apply to a large variety of commercially available array processors such as Geometric Arithmetic Parallel Processor (GAPP) [16] of NCR, Distributed Array Processor (DAP) [109] of ICL, England, NASA's Massively Parallel Processor (MPP) [11], and Connection Machines [130] of Thinking Machines Corporation. These array processors are basically single instruction multiple data (SIMD) computers with very simple (bit serial) processing elements.

The results presented in this thesis also apply to a large number of WSI-based and VLSI-based processor arrays which include the Systolic Arrays [61], Reconfigurable Array of Processors ELSA (European large SIMD array) [122], and a variety of special-purpose VLSI and experimental WSI devices for applications such as signal processing, image processing, and numerical computations. Furthermore, the results are equally applicable to the memory chips. Memory chips are the most obvious

candidates since the underlying architecture is highly regular and has a large number of identical cells with spare cells which can be switched in to replace faulty ones. Memory chips are particularly densely packed and therefore vulnerable to defects. As the complexities of the interconnections increase, either an increasing amount of the fault-tolerant circuitry has to be dedicated to routing around the faulty cells, or a less flexible use of the spares has to be accepted [35,85].

It should be stressed that the results established in this thesis rely only on the underlying topological structure (e.g., ring, array) of the system; thus, they hold regardless of whether the basic processing elements are hardware registers, transputers, chips, workstations, software modules, etc., as long as the connectivity of the resulting system (network) is regular and uniform. The results apply directly to systems whose topologies fall in one of the classes considered here in which there are only PE failures.

1.4 Outline of the Thesis

The thesis is organized as follows. In Chapter 2, a comprehensive background on some common regular architectures is given; also given in this chapter is an overview of how fault tolerance is achieved in each of these architectures.

Chapter 3 contains a complete characterization of catastrophic fault patterns. In this chapter, characteristics of catastrophic patterns are described; bounds on the width of fault patterns are given; results expressing sensitivity of the width to the amount of bypass links available are given; and the properties of a special class of catastrophic fault patterns are described.

Chapter 4 deals with the problem of generation and enumeration of catastrophic fault patterns. For both unidirectional and bidirectional links, algorithms are given for constructing catastrophic fault patterns that are contained in the largest possible window; bounds are established on the number of catastrophic fault patterns; it is

shown how to generate all such patterns systematically and determine if a given pattern is catastrophic for a given link configuration.

Fault-tolerant attributes in terms of the minimum number of faults sufficient to cause a catastrophic failure in one- and two-dimensional systolic arrays for different models of computation are studied in Chapter 5.

In Chapter 6, limits to reconfigurability in ring topologies are studied; bounds are established on the degree of fault tolerance achievable; bounds are derived on the size of the problems whose solution can be guaranteed in the presence of a given number of faults.

The major results are summarized in Chapter 7. Some open problems are listed, and a few conclusions that might serve as guidelines for future research are given.

Chapter 2

Background

2.1 Introduction

Several regular structures such as loops, trees, full connections and hypercubes, etc. have been proposed to interconnect a network of computers in a message/packet switching environment and have also been implemented [4,98,99,134]. These are special purpose architectures to solve special class of computationally intensive problems. Fault tolerance is an important factor in these architectures and has received considerable attention in recent years. This chapter provides an overview of some of the concepts and terminologies of fault tolerance, gives an overview of some of the regular architectures, and describes how fault tolerance is achieved in each of these regular architectures.

This chapter is organized as follows. In Section 2.2, an overview of the fault tolerance terminologies and techniques is given. A general background on some known regular architectures like systolic arrays, ring networks and tree topologies is described in Section 2.3. Section 2.4 outlines how the fault tolerance techniques are used in each of the aforementioned architectures.

2.2 Background on Fault Tolerance Techniques

2.2.1 Fault Tolerance Terminologies

A *fault* is an erroneous behavior of a system's hardware or software. Hardware faults are attributed to the component failures and external disturbances such as electromagnetic interference and radiation. Software faults are the result of incomplete design specifications and incorrect implementation of the specifications. These faults introduce errors in the execution of programs and corrupt the information in data structures.

There are three major techniques for improving or maintaining a system's normal performance: fault avoidance, fault masking, and fault tolerance. *Fault avoidance* is the prevention of the causes of faults. Good examples of fault avoidance techniques include design review, use of high-reliability components, derating, screening, and system testing. *Fault masking* is the process that prevents the propagation of a system's faults into the data of the system. An example is the use of redundancy and voting. Finally, *fault tolerance* is the ability of the system to continue to perform its specified tasks in the presence of faults. One example of fault tolerance is the use of "sparing": if a fault appears in an operational unit, a spare is brought on line to prevent the occurrence of a system malfunction.

Fault tolerance can be broadly defined as applying redundancy to a system. It sounds simple enough, but implementing the technique successfully requires careful tailoring to a system's application requirements; a properly designed fault-tolerant approach will correct any kind of fault – permanent, intermittent, or transient. Fault tolerance involves the following four major issues: fault detection, fault location, fault containment, and fault recovery. *Fault detection* is the ability of a system to recognize that a fault has occurred. Fault detection is often required before any recovery mechanism can be implemented. *Fault location* is a system's ability to determine where a fault has occurred so that an appropriate recovery can be implemented. *Fault containment* is the process of isolating a fault and preventing its effects from propagating

throughout the system. Finally, *fault recovery* is a system's ability to remain operational or to regain operational status via reconfiguration even in the presence of faults.

2.2.2 Techniques of Fault Tolerance

The key ingredient in all fault tolerance techniques is redundancy. Redundancy is simply the addition of *information, resources, or time* beyond what is needed for normal system operation. The redundancy may take several forms, including information redundancy, hardware redundancy, software redundancy, and time redundancy.

Information redundancy is the addition of information beyond what is required to implement a function. A good example of information redundancy is an error-detecting code. Perhaps the simplest form of error detection coding is the single-bit parity check. Other forms of error detection coding are: checksums and arithmetic codes [125]. Checksums are most applicable when blocks of data are to be transferred from one point to another. The primary advantage of the arithmetic codes is that they allow hardware that performs arithmetic operations to be easily checked. Once the location of the error is known, the data bit can be corrected by a single-parity code. Error-correcting codes not only detect errors but also correct them. Possibly the most common extension of parity is the Hamming error-correcting code. Hamming codes are designed to detect double errors and correct single errors. Other forms of single and multiple error-correcting codes also exist and can be found in [29,100].

Hardware redundancy is the physical replication of hardware for the purpose of detecting and tolerating faults. A common form of this redundancy is the triple modular redundancy, or TMR [125]. The purpose of TMR is to mask single faults by triplicating hardware and voting on the results. The general form of TMR is the N-Modular redundancy (NMR). Other form of hardware redundancy involves incorporating fault detection and fault recovery into the system at the expense of eliminating fault-masking capability. Example of this is the standby replacement. In this configuration, one unit is operational while one or more units are standbys.

Various error detection schemes are used to determine when the on-line unit has failed; if a failure is detected, the on-line unit is removed from operation and replaced with a standby.

Software redundancy is simply the addition of extra software to provide some fault tolerance features. The type of redundancy may range from complete duplication of software to the addition of small programs to perform validity checks. Probably the most common form of software redundancy is the validity, or reasonableness check. A second type of software redundancy is the periodic self- test. A third example of software redundancy is the use of multiple copies of programs. This version of software redundancy is typically referred to as "N-version programming" [9]. In this approach, multiple versions of a software run simultaneously on multiple processors or sequentially on a single processor; the results are compared to provide a means of fault detection. The N-version, independently written programs provide protection against both hardware and software faults. The hardware equivalent of N-version programming is the N-modular redundancy.

Time redundancy uses additional time to provide fault detection and, sometimes, fault tolerance. It can be employed to distinguish between permanent and transient failures. The processor performs the computations one or more times after detecting first error; if the error condition clears, the processor can assume that fault was transient. In another form of time redundancy, the same processor performs the same computations multiple times using different coding schemes in each case with the assumption that the fault may manifest itself in different ways depending upon the particular code being used. Unfortunately, the time redundancy technique cannot be used to detect permanent failures.

Concurrent error detection is receiving a lot of attention in recent years in the context of transient errors. Concurrent error detection is believed to be the only feasible way to diagnose transient faults [86]. Since a good percentage of faults occurring in a system are transient in nature, having a concurrent error detection scheme has

become a necessity, rather than an optional feature, for fault-tolerant systems. The algorithm-based fault tolerance technique [5,45,114] is a widely applicable concurrent error detection scheme, where data elements are encoded at a high level, and the redundancy in the encoded output produced by modified algorithms is utilized to detect errors [86].

All the fault tolerance techniques mentioned above can be separated into two major categories: static redundancy and dynamic redundancy. *Static redundancy* techniques instantly correct errors and do not require higher system-level intervention to do so. The use of TMR or error correcting codes are some examples of the static redundancy technique. *Dynamic redundancy*, on the other hand, requires interruption of system availability while the fault is isolated. Once the fault is isolated, the redundant resource is switched on (reconfigured), and the system is recovered (i.e., restarted). For a real-time system, if the error occurs in the middle of an operation, some program rollback is necessary to discard the bad data and to recover as much good data as possible. The primary advantage of dynamic redundancy techniques is that they can be implemented with fewer resources; indeed, static redundancy can use as much as 100 percent more resources than the dynamic schemes.

Mechanisms to achieve fault tolerance are usually incorporated in all levels of computer systems to improve their quality and continuity of performance. Fault-tolerant systems have taken one of two architectural directions: *loosely-coupled* systems have relied primarily on software to ensure reliable operation, while *tightly-coupled* systems have provided a hardware solution. A great deal of work in the area of fault-tolerant computing has been done in recent years, much of it in the support of the U.S. space program [126] and special applications such as electronic telephone switching systems [131]. The use of various fault tolerance techniques in commercial computer systems are well documented in [68,110,126,131].

2.3 Background on Regular Architectures

Regular architectures [8] are structures in which a number of identical cells (processors) are interconnected by means of a regular pattern. Memories are basic instances of such regular structures. Fast, dedicated computer architectures such as systolic arrays, mesh-connected multiprocessors, loop topologies, binary tree structures, and hypercubes are the examples of regular architectures. Each structure possesses some unique advantages and disadvantages compared to others. For example, the loop topology allows for very simple routing, but may have too large internode distance. On the other hand, the binary tree structures possess small internode distance but poor fault tolerance. In this section, certain regular architectures to be considered in this thesis are discussed in reasonable details.

2.3.1 Systolic Arrays

A systolic system is a network of processors which rhythmically compute and pass data through the system. Systolic arrays are readily amenable to VLSI implementation. It is particularly suitable to a special class of computation bound algorithms and takes advantage of their regular, localized data flow. The basic principle of systolic design is that all the data, while being pumped regularly and rhythmically across the array, can be effectively used in all the PEs. The systolic array features the important properties of modularity, regularity, local interconnection, a high degree of pipelining, and highly synchronized multiprocessing.

Systolic arrays are effective means to achieve speedups in execution time for computation-bound problems by exploiting parallel processing and pipelining [61]. A large number of simple homogeneous processing elements (PEs) can be connected in a regular fashion to give one or higher dimensional structures. Systolic arrays are characterized by their regular data flow, with streams of data items flowing through the structure synchronously in a fixed regular pattern. All communications take place between nearest neighbors. The uniformity of the individual components and

their regular interconnection make it attractive to consider such arrays for wafer-scale integration. An example of a systolic array is given in Figure 2.1.

Figure 2.1: A systolic array of processors

Array processors have been proposed in real-time control systems, signal and image processing systems to solve computation-intensive problems such as fast Fourier transformation, convolution, and matrix operations, which consist of high volume and iterative (or recursive) computational steps [61]. In today's VLSI technologies, it is economically feasible to use a massively parallel processing array, which consists of a large number of fine-grained cells (i.e., processing elements) for achieving high performance in such applications. Systolic arrays and massive parallel processor (MPP) are examples of such VLSI arrays. For a very large array, faults are very likely to occur during normal operations. Therefore, on-line fault-tolerance processes, which include fault detection, diagnosis, reconfiguration, and rollback error recovery are crucial for such systems.

2.3.2 Ring Networks

A ring network is defined as a set of processors connected into a Hamiltonian cycle by oriented links. Due to its simple structure, easy implementation, and expandability, the ring network is one of the most popular topologies for local computer networks [107].

The loop topology is a regular network without central node. Loop networks are attractive because they require simple control software and permit effective operation on data rates much higher than broadcast buses as they do not suffer the carrier sense limitation of the latter [73,132]. Interprocessor communication problems are similar in tightly coupled architectures and local computer networks, although of a different scale. In large loop networks it is necessary to provide fault tolerance against node and link failures.

In a loop network, host computers are connected via loop interface hardware; each node is connected to its neighbor to form a complete loop, and a message is passed from node to node along unidirectional links. There are no routing decisions to be made in such a network. A node simply transmits its message to the next node in the loop, and the message circulates around the network until it reaches the destination node. The interface hardware must be able to identify messages intended for its host. An example of a ring network is shown in Figure 2.2.

Ring structures possess some unique advantages and disadvantages compared to one another. For example, a bi-directional single loop (ring) structure has only two I/O ports per node, but the diameter (maximum number of hops between any two nodes along the shortest path) is $N/2$ in a system with N nodes. Any two non-adjacent faulty nodes or links disconnect the loop. On the other hand, a completely connected structure has $(N - 1)$ I/O ports per node, but the distance between any two nodes is unity. The structure is highly fault tolerant, but due to its high cost the structure is unsuitable for systems with large number of nodes.

Figure 2.2: A ring network of processors

2.3.3 Tree Topologies

A large number of computational problems have the property that they start slowly, build up to a crescendo of activity, and then collapse to produce a relatively small result. This has led to an interest in tree structured networks [31]. A tree is a commonly used interconnection structure in computing systems. Typically the nodes of the tree corresponds to computing facilities (processors or I/O devices) while the edges represent communication links between the facilities. See Figure 2.3 for an example of a tree.

Tree structures are useful for a variety of applications, such as real-time control and measurement systems, divide-and-conquer strategies, sorting, and database operations [21,22,41,44,66,77,104,127].

Figure 2.3: A tree of processors

2.4 Fault Tolerance in Regular Architectures

All of the architectures described in the previous section have the major drawback that a single processor or edge failure may render the entire network unusable if the algorithm running on the network requires that the topology of the network does not change. The failure of a single processor or the failure of a link between two processors would destroy the topology of these architectures. Thus, some form of fault tolerance must be incorporated into these architectures in order to make the network of processors more reliable. The fact that these structures are highly regular makes it easy to achieve a high degree of fault tolerance through the use of redundant processors and links. This section reviews different approaches known for making the regular architectures fault-tolerant.

2.4.1 Fault-Tolerant Systolic Arrays

Using VLSI technology, systolic arrays are now being manufactured cost effectively due to their highly dense and efficient layout [46,70,79,80,81,103]. In one hand, it is now possible to incorporate a large number of processing elements and their necessary interconnections into a single VLSI/WSI chip. On the other hand, low yield of VLSI/WSI has been considered of a major concern. The low yield comes from both complex manufacturing processes and very high circuit density in VLSI/WSI components [81]. Therefore, it is necessary to build redundancy and reconfiguration mechanisms into VLSI/WSI device for improving the yield.

Two different approaches to enhance low yield has been suggested [54,84,85]. One uses proper design rules and advanced manufacturing techniques to reduce failures during manufacturing. The other approach uses modular redundancy in the chip/wafer design so that the desired target system can be implemented by interconnecting the good modules on the chip/wafer. The use of redundant modules in the chip/wafer to enhance yield is popular because the first approach is almost impossible to achieve in VLSI/WSI components with current manufacturing technology. Thus, redundancy and reconfiguration are two fundamental aspects considered in achieving fault tolerance. A lot of research has been done in the design [32,48,49,50,53,54,58,59,60,63,64,65,72,84,85,91,93,95,105,112,113,136] and implementation [23,25,30,40,46,52,62,69,70,79,80,81,82,96,103,117,118,121,128] of fault-tolerant systolic architectures.

Redundancy Schemes

Structure redundancy is implemented by adding redundant PEs and redundant interconnections to replace faulty PEs. A simple way to provide redundancy in systolic arrays is by adding spare rows and/or spare columns of PEs. The spare PEs need not be arranged that way; there are, however, other ways to arrange spare PEs in the array. An arrangement of spare PEs can be judged by many factors which include spare utilization, effectiveness of tolerating various fault distributions, the

length of connecting wires after reconfiguration, etc. Some of the popular schemes will be described here.

In a “local (or interstitial) redundancy scheme,” [40,136], spare PEs are distributed uniformly into an array, and a small group of PEs (called a block) share the spare PEs which are in the block. In this approach, a faulty PE can be replaced only by a “local” spare PE, i.e., a spare in the same block. Since spare PEs are partitioned and distributed into each block of the array, maintaining the required interconnection structure becomes much simpler. It requires a simple reconfiguration control algorithm and circuit, and low hardware and time overhead. The length of connecting wires after reconfiguration is limited by the width of a block. The drawback of this scheme is that the array may fail due to the lack of spare PEs in one block when other blocks contain spare PEs. An example of a systolic structure using interstitial redundancy is shown in Figure 2.4.

Figure 2.4: Interstitial redundancy scheme

In a “global redundancy scheme,” there is no restriction on the usage of spares to replace faulty PEs. In principle, any faulty PE can be replaced by any spare PE. The extent to which this is true depends on the flexibility and therefore the complexity of the adopted redundancy scheme.

In a “two-level redundancy scheme” scheme [133], the array is partitioned into identical blocks with each block containing a fixed number of spare PEs. This provides the first level of redundancy, called local redundancy. The local redundancy is effective for the distributed faults but it is not sufficient to overcome clustered faults. A second level of redundancy, called the global redundancy, is therefore provided by adding spare blocks to the array for tolerating failures of blocks induced by clustered faults. This redundancy scheme is called two-level redundancy. The two-level redundancy can be viewed as a compromise between the pure local and pure global redundancy schemes.

Some of the advantages of the two-level redundancy scheme over the local and the global redundancy scheme are the following [133]:

1. The two-level redundancy scheme has better spare utilization than that of global whole row (and/or column) bypass redundancy schemes.
2. The local redundancy is effective for distributed faults; the global redundancy is effective for clustered faults. A pure local redundancy cannot tolerate clustered faults.

Reconfiguration Schemes

Reconfiguration is defined as the operation of replacing faulty components with spares while maintaining the original interconnection structure. The important criteria for evaluating a reconfiguration scheme are as follows [133]:

1. reconfiguration effectiveness - the probability that an array with a given number of faulty PEs is reconfigurable.

2. hardware (silicon area) overhead - switches, connecting wires, and control unit for reconfiguration.
3. length of connecting wires after reconfiguration procedure.
4. overall yield and reliability.

The reconfiguration effectiveness represents the ability of a reconfiguration and redundancy scheme to tolerate a given number of faults in an array. It also indicates the utilization of spare PEs under a redundancy and reconfiguration scheme. The performance degradation of the system is directly proportional to the length of the longest connecting wires after reconfiguration. For a high- performance system, it is important to limit the length of connecting wires. For a real-time system, the time for reconfiguration should be kept within a limit to assure that the hard deadline will be met [133]. If a real-time system misses the hard deadline, a catastrophic failure will occur. The reconfiguration effectiveness and hardware overhead are functions of the redundancy and reconfiguration scheme. The yield and reliability are also important criteria. They are strongly related to the reconfiguration effectiveness and the hardware overhead.

Several reconfiguration strategies have been suggested to restore fault-free operation in regular processor arrays with faulty elements [12,14,15,19,24,33,42,43,55, 57,62,67,71,74,75,76,92,94,97,108,111,115,120,123,133,135]. Two of these approaches will be briefly reviewed here. One is due to Rosenberg [111] and the other is due to Kung and Lam [62]. Rosenberg's approach, called the Diogenes approach, provides alternate paths between PEs. When a PE is diagnosed to be faulty, it is bypassed by using the alternate paths. The main feature of the Diogenes approach is its simplicity. However, a potential disadvantage is that, in the presence of consecutive faulty PEs, logically connected PEs may be far apart, requiring the reduction of clock speed and thus reducing throughput of the array.

An alternate approach called systolic fault tolerance is due to Kung and Lam [62]. This approach uses delay registers in the alternate paths. When a PE is diagnosed

to be faulty, it is bypassed through the bypass registers. Use of memories or registers in bypass paths helps to avoid increase in inter-PE wire length due to reconfiguration. When applied to 2-D arrays, the scheme required, in addition to various delay registers, varying number of extra data links between PEs for reconfiguration. The two reconfiguration methods described above make the assumption that the circuits in the alternate paths remain fault-free.

Since it is possible to map an arbitrary large algorithm to fixed-size array architecture, fault-tolerant schemes that provide graceful degradation should be considered. The idea is to dynamically reconfigure so that, after removal of faulty PEs, a fully operational, reduced size array is available. There are two types of reconfiguration schemes for using spare PEs in mesh arrays with global redundancy:

- a) Whole row (and/or column) bypass (WRB/WCB) [52]: In this type, a faulty PE causes its whole row or column to be bypassed. There are many different reconfiguration schemes that are based on this type. They have the advantage of requiring simple circuitry to bypass rows and/or columns. Their main drawback is poor utilization of spares. The problem of choosing the minimum number of spare rows and/or column that cover all the faulty PEs is NP-complete [60]. Thus, various heuristic reconfiguration algorithms are used.
- b) Single-cell bypass (SCB) [91,117]: In this reconfiguration scheme, only faulty PEs are bypassed (see Figure 2.5). This scheme has better utilization of spare PEs. After the array is reconfigured, each PE of the array is mapped to a fault-free PE in the wafer. In its most flexible form, mapping in the required structure depends only on the number of fault-free PEs and not on the location of the PEs in the wafer. Depending on the flexibility of the mapping, the reconfiguration algorithm can become very complex. In that case, many interconnecting lines and complicated switches are needed. In addition, there is a possibility of long interconnection wires after reconfiguration.

Figure 2.5: Single cell bypass (SCB) scheme

An entirely different approach to reconfiguration is based on the utilization of time redundancy rather than space redundancy. In time redundancy [78,90,119,138], the number of spare processors is kept to a minimum, while the operational speed of the reconfigured array is commonly degraded. This limitation can severely affect the application of time redundancy to systems which are used in high throughput applications [6].

2.4.2 Fault-Tolerant Ring Networks

Reliability is a severe problem with single-loop networks, particularly when network size increases. Any single node or link failure can disrupt communication. One approach to improving reliability is to use very reliable loop interfaces and links. However, a simple solution is to bring all the interfaces to a central point. This results

in a star shaped ring with a wire center, allowing easy maintenance and reconfiguration. Even though this scheme allows distributed loop control, there is a physical central point containing the switch hardware; therefore, the reliability of the network depends on the reliability of this central facility. Further, in a very large network, there may be frequent node and link failures, thus making the network less reliable.

A common approach to provide higher reliability in loop networks is by introducing redundancy. The idea here is to design regular multiconnected loop networks so that several alternative paths for communication exist between pair of nodes. In other words, each node is connected to two or more nodes in the network. With alternative paths between nodes, the network can sustain several node or link failures. Liu [73] has proposed a distributed double-loop computer network (DDL CN) as a fault-tolerant distributed system. In this network each node is connected to its two neighboring nodes, forming a complete forward loop and a complete backward loop. With redundant links the network can sustain loop interface and link failures. The loop interface hardware is implemented with a microprocessor, and the network uses a fairly simple routing algorithm to bypass failed links or nodes.

For a network with a large number of nodes, delays will still be excessive because the diameter of this network is $N/2$, and the average delay is on the order $O(N/4)$. A highly reliable loop network architecture, called the daisy-chain loop has been studied in [34] (see Figure 2.6). This is a double-loop architecture with a basic forward loop connecting all neighboring nodes and a node-skipping, daisy-chain backward loop. Analysis has shown that this network to have less delay, better throughput performance, and higher reliability than DDL CN [20,34,83,101].

Figure 2.6: A daisy loop network

These double-loop networks are classified as forward-loop-backward-hop (FLBH) networks. In FLBH networks each node has a forward link to its neighbor and a backward link to a node at a distance h , where h is called the skip distance. That is, a node is connected to node $x + 1$ and node $(x - h) \bmod N$. Different topologies can be obtained for different values of h ; $h = 1$ results in the single-loop network. In the optimal FLBH network, the parameter h is selected such that the diameter is minimized.

The optimal FLBH network is described and analyzed by Raghavendra *et al* [102, 106, 107]; the network consists of a forward loop connecting all neighboring nodes and a backward loop whose interconnection pattern depends on N , the number of

nodes in the network (See Figure 2.7). It is shown that for optimum performance and reliability the skip distance of the backward loop is $\sqrt{N}$. Except for the fact that skip distance is a function of N , all other features (including the interface architecture) are similar to those of the DDLCN and the daisy-chain loop.

Figure 2.7: 20-node optimal forward loop backward hop network

A ring network is also known to have long connections between pairs of processors. To be more specific, define the distance from processor i to processor j as the minimum number of links a message has to traverse from i to j , and define the diameter of a ring network as the maximum distance between any two processors. Then a large diameter signifies a connection of two processors through many intermediate processors and is clearly undesirable.

A ring network of size n has diameter $n - 1$, which is the distance from processor i to processor $i - 1$ (assuming that the links in the ring network are $i \rightarrow i + 1 \pmod{n}$, $i =$

$0, 1, \dots, n - 1$. Furthermore, a common measure of reliability is the connectivity of a network. The connectivity of a directed graph is defined to be k if every node has a directed path to every other node after removal of any $d < k$ nodes. It is easily seen that the connectivity of the ring network is one.

One way to shorten the diameter and increase the connectivity is by adding links to a ring network. The following two general principles are usually followed in adding links: 1) as few links as possible are added since links can be costly and more links per processor increase the switching cost at processors and 2) the links are added homogeneously to the processors so that standardized processors can be used to build the network.

2.4.3 Fault-Tolerant Tree Topologies

In a binary tree architecture there is a certain probability that a node or link will fail. The tree structure is physically static; if any node or link fails, the tree structure no longer exists. Also, many processor may not be able to communicate with each other. It is, therefore, necessary to maintain communication paths among pairs of nodes at all times. Redundant nodes and links can be employed for providing fault tolerance against node and link failures.

Fault tolerance in tree architectures has been studied by several researchers [21, 22, 39, 41, 44, 66, 77, 127]. Hayes [39] has introduced a graph model for fault-tolerant computing systems, presenting design procedures for k -fault-tolerant systems; in particular, optimal one-fault-tolerant tree structures. The system is designed so that it requires the minimum number of spare nodes and redundant links for tolerating any single failure. However, the one-fault-tolerant system cannot tolerate multiple failures. Another study, by Kwan and Toida [66], also considers the design of optimal k -fault-tolerant tree structures. Extending the work of Hayes, Kwan and Toida study procedures for designing optimal fault-tolerant systems for a class of hierarchical tree structures. Their work assumes that there are mechanisms for fault diagnosis, reconfiguration, and recovery.

The fault-tolerant tree architecture, proposed by Raghavendra [104], provides protection against any single failure in each level of the tree structure. The system will tolerate multiple failures as long as they are in different levels of the tree. The logical structure of the redundant binary tree is shown in Figure 2.8. There is a spare node for each level in the tree (indicated by a circle within a circle), and there are redundant links (indicated by dashed lines). With this arrangement any single failure in a level can be tolerated. This scheme tolerates several failures if they are in different levels of the tree. If more protection against failures is needed, additional spare nodes can be used at lower levels of the tree where the number of nodes increases rapidly. In this redundant binary tree structure, each node has between three and five extra links to provide redundant connections to other nodes. Reconfiguration upon node failure involves selecting a spare node and appropriate links for communication.

Figure 2.8: A fault-tolerant hyper tree

2.5 Chapter Summary

This chapter gives an overview of various fault tolerance concepts, terminologies, and approaches. A background on some of the commonly known regular architectures such as systolic arrays, ring and tree topologies are described. Fault tolerance schemes in each of these architectures are also discussed. The commonly proposed approach for achieving fault tolerance was found to be the process of incorporating redundancy - redundant processors and redundant links. The incorporation was rather easy due to the regularity of these architectures. None of the fault tolerance schemes takes into account catastrophic faults or guarantee full functional operation all the time. In other words, there seems to be no fully effective fault tolerance scheme.

Chapter 3

Characterization of Catastrophic Fault Patterns

3.1 Introduction

In this chapter, fault patterns which are catastrophic for linear systolic arrays are studied. A characterization of catastrophic fault patterns is derived by treating the array as a special set of positive integers (called a *deadly integer set*); it is shown that the cardinality of a catastrophic fault pattern does not depend on the number of bypass links available but *only* on the length of the longest one; bounds on the width of catastrophic fault patterns are established; results expressing the sensitiveness of the width to the amount of bypass links available are proved. A special class of catastrophic fault patterns is studied in detail and is shown to have interesting underlying algebraic properties.

The organization of this chapter is as follows. In Section 3.2, basic concepts and terminologies are introduced. The characterization of catastrophic fault patterns is presented in Section 3.3. Section 3.4 is devoted to the study of a particular class of catastrophic fault patterns and their underlying algebraic properties.

3.2 Preliminaries

The systems being studied are one-dimensional (or linear) systolic arrays as shown in Figure 3.1. The basic components of such a system are the *processing elements* (PEs) and the *links*. The links can be either unidirectional or bidirectional. There are two kind of links in redundant arrays: *regular* or *bypass*. Regular links exist between neighboring PEs; bypass links connect non-neighboring PEs. The bypass links are used strictly for reconfiguration purposes when a fault is detected; in the absence of faults, the bypass links are redundant. Bypass links are not shown in Figure 3.1 for simplicity.

Figure 3.1: A linear systolic array of PEs

A *fault* in a PE can be best described as its inability to perform correct operation on correct inputs, possibly due to some elementary circuit or gate-level failures. The failures could be either permanent or transient. In this thesis, the focus of the investigation is on permanent faults. Any non-faulty PE which is not active during a computation is said to be a *spare* for that computation; thus, there is no a priori designation distinguishing spares from other PEs. The location of spares is irrelevant for this discussion although it may play a significant role in real life situation.

Let $A = \{x_1, x_2, \dots, x_N\}$ denote one-dimensional systolic array (shortly, array) of N PEs, where each $x \in A$ represents a processing element and there exists a direct link between x_i and x_{i+1} , $1 \leq i < N$. Any link connecting x_i and x_j where $j > i + 1$ is said to be a bypass link.

Definition 1 The length of a bypass link, connecting x_i and x_j , is the distance in the array between x_i and x_j ; i.e., $|j - i|$.

Definition 2 Given an integer $g \in (1, N]$ and an array A of size N , A is said to have link redundancy g , if for every $x_i \in A$ with $i \leq N - g$ there exists a link between x_i and x_{i+g} .

The above definition can be generalized as follows:

Definition 3 The array A has link redundancy $G = \{g_1, g_2, \dots, g_k\}$ where $g_j < g_{j+1}$ and $g_j \in (1, N]$, if A has link redundancy $g_1, g_2, \dots, g_k$.

In the following, it will be assumed that $g_1 = 1$ and no other link except the ones specified exists in the array; that is, G totally defines the *link structure* of A .

Given a linear array A of size N , a *fault pattern* for A is a set of integers $F = \{f_1, f_2, \dots, f_m\}$ where $m \leq N$, $f_j < f_{j+1}$ and $f_j \in (1, N)$.

An assignment of a fault pattern F to A means that for every $f \in F$, x_f is faulty. Examples of fault pattern of size 4 is given in Figure 3.2(a) and Figure 3.2(b).

Figure 3.2(a): A fault pattern of size 4

Figure 3.2(b): A different fault pattern of size 4

Definition 4 *A fault pattern F is catastrophic for an array A with link redundancy G if the array cannot be reconfigured in the presence of such assignment of faults. That is, the removal of the faulty elements and their incident links will cause the array to become disconnected.*

Example 1: The fault pattern given in Figure 3.2(b) is catastrophic since the removal of the faulty PEs and their incident links disconnect the array as shown in Figure 3.3.

Figure 3.3: A disconnected array

3.3 Deadly Integer Sets and Catastrophic Fault Patterns

In this section, the fault patterns for linear arrays are characterized by treating the linear array as a set of consecutive positive integers. The results are obtained by reducing the problem of characterization of catastrophic fault patterns to the problem of identifying a special family of sets of integers, called deadly sets. Properties of deadly sets are established. The minimum number of faults required to construct a catastrophic fault pattern is a function of the length of the longest bypass link, but not the number of bypass links as it might seem at a first glance. It is also necessary that the fault pattern be located within a window of certain size for it to have any impact on reconfiguration.

Consider two finite sets $F = \{f_1, f_2, \dots, f_m\}$ and $G = \{g_1, g_2, \dots, g_k\}$ of positive integers. Without loss of generality, let $f_i < f_{i+1}$ and $g_j < g_{j+1}$ for $1 \leq i \leq m$ and $1 \leq j \leq k$. Here, G can be visualized as the set of links including the bypass links, and F corresponds to a distribution of faults (fault pattern). The definition of the deadly integer set now follows.

Definition 5 *An integer $x \in Z$, the set of integers, is "dead" if at least one of the following conditions hold:*

1. $x \in F$,
2. $\forall g \in G, x + g$ is dead,
3. $\forall g \in G, x - g$ is dead.

Let D be the set of dead integers with respect to F and G .

Definition 6 *F is deadly for G if $D = Z$. That is, F is deadly for G if all integers are dead.*

Given $x \in Z$, let $R(x) = \{y \in Z : \exists g \in G \text{ such that } y = x + g\}$ be the right neighbor of x and $L(x) = \{y \in Z : \exists g \in G \text{ such that } y = x - g\}$ be the left neighbor of x . Also let $N(x) = R(x) \cup L(x)$ be the neighbors of x . Obviously, if $y \in R(x)$ then $x \in L(y)$.

The set of dead integers for a given F and G is now characterized with respect to their "time" of death as follows:

1). $D^0 = F$ and $A^0 = Z - D^0$; that is, initially only the "faulty" integers are dead and all others are "alive".

2a). $D_R^i = \{x \in A^{i-1} : R(x) \subseteq \bigcup_{j < i} D^j\}$

b). $D_L^i = \{x \in A^{i-1} : L(x) \subseteq \bigcup_{j < i} D^j\}$

c). $D^i = D_R^i \cup D_L^i$

d). $A^i = A^{i-1} - D^i$

3). $D = \bigcup_i D^i$

Note that 1, 2, and 3 correspond to the equally numbered conditions in Definition 5.

Now define the time of death, $d(x)$, of an integer x to be

$$d(x) = \begin{cases} 0 & \text{if } x \in F \\ i & \text{if } x \in A^{i-1} \cap D^i \\ \infty & \text{otherwise (i.e., } x \notin D) \end{cases}$$

Definition 5 can now be rephrased as follows:

Definition 7 F is deadly for G if $d(x) \neq \infty, \forall x \in Z$

Now consider the set $\Psi = \{\sigma\}$ of all sequences σ of integers, "infinite at both ends", where each integer in

$$\sigma = \dots x_{i-1} x_i x_{i+1} \dots$$

is a right neighbor of the preceding one, i.e., $x_{i+1} \in R(x_i)$.

Given σ and $x \in \sigma$, let $s(x)$ denote the successor of x in σ .

Lemma 1 *If F is deadly then $\forall \sigma \in \Psi$, $\sigma \cap F \neq \emptyset$. That is, if F is deadly then any “doubly-ended” infinite sequence of neighbors must contain an element of F .*

Proof: Let $x \in \sigma$. Since F is deadly for G , $d(x) = i \neq \infty$ (by Definition 6). If $i = 0$ then the lemma is proved. Suppose $i > 0$. Then $d(x) = i$ implies $x \in D_R^i \cup D_L^i$.

Case 1: $x \in D_R^i$

This, in turn, implies that $R(x)$ is a subset of $\bigcup_{j < i} D^j$. Since $s(x) \subseteq R(x)$, $d(s(x)) = j$ and $s(x) \in D^j$ where $j < i$.

Claim: $s(x) \in D^j$ for $x \in D_R^i$ and $j < i$

Proof: Suppose $s(x) \in D_L^j$. This implies $L(s(x)) \in \bigcup_{p < j} D^p$. Since $x \in L(s(x))$, $d(x) < j$ contradicting the fact that $d(x) = i > j$. Since $s(x) \in D^j = D_R^j \cup D_L^j$, the claim follows.

If $d(s(x)) = 0$, the lemma is proved; otherwise, define for an integer $k \geq 0$, $s^k(x) = s(s^{k-1}(x))$ where $s^0(x) = x$. What is shown that if neither x nor $s(x)$ is in F , then $d(x) > d(s(x)) > 0$. Furthermore, $s(x) \in D_R^{d(s(x))}$. Let this hold for a contiguous subsequence of σ ; that is, for $1 \leq p \leq k$,

$$1. d(s^{p-1}(x)) > d(s^p(x)) > 0,$$

$$2. s^p(x) \in D_R^{d(s^p(x))}.$$

It will now be shown that

$$(i) d(s^k(x)) < d(s^{k-1}(x)) \text{ and}$$

$$(ii) \text{ if } d(s^k(x)) > 0 \text{ then } s^k(x) \in D_R^{d(s^k(x))}.$$

Proof of (i): $s^{k-1}(x) \in D_R$ implies $R(s^{k-1}(x))$ is a subset of $\bigcup_{p < d(s^{k-1}(x))} D^p$. Since $s^k(x) \in R(s^{k-1}(x))$, $d(s^k(x)) < d(s^{k-1}(x))$.

Proof of (ii): Let $d(s^k(x)) > 0$. Then if $L(s^k(x))$ is a subset of $\bigcup_{p < d(s^{k-1}(x))} D^p$, $s^k(x) \in D_R^{d(s^k(x))}$.

Therefore, either $s^k(x) \in F$ or its “death time” is smaller than the preceding element in the sequence. If the latter case holds, the same process can be applied to $s^{k+1}(x)$ and so on until an element $s^r(x)$ can be found such that $d(s^r(x)) = 0$ which must be the case since the sequence

$$d(x), d(s(x)), \dots, d(s^i(x)), \dots$$

is decreasing.

Case 2: $x \in D_L^i$.

The argument is similar by substituting L for R.

This now completes the proof of Lemma 1. $\square$

For a given G , not all fault patterns are catastrophic. For example, the fault pattern given in Figure 3.2(a) is not catastrophic whereas the fault pattern in Figure 3.2(b) is catastrophic. Certain additional conditions must hold that would make the distinction easy. Obvious questions that could be asked are the following:

- how many faults would it take to form a catastrophic fault pattern for a given link configuration G ?
- how does a catastrophic fault pattern spread out?
- how sensitive is a catastrophic fault pattern to the variation in the number of bypass links?
- what properties does a catastrophic fault pattern have?
- how can catastrophic fault patterns be constructed?
- how many catastrophic fault patterns are there for a given link configuration G ?

Questions regarding construction and enumeration of catastrophic fault patterns will be discussed in Chapter 4. Rest of the questions are addressed in the following sections of this chapter.

3.3.1 Cardinality of Catastrophic Fault Patterns

The following theorem establishes condition on the minimum number of faults that a fault pattern F must consist of in order for it to be catastrophic for a given link configuration G .

Theorem 1 F is deadly with respect to G implies that the cardinality of F , $\|F\| \geq g_k$.

Proof: Define g_k sequences σ_i ($1 \leq i \leq g_k$) as follows:

$$\sigma_i = \{i + jg_k : j \in \mathbb{Z}^+\} \cup \{i - jg_k : j \in \mathbb{Z}^+\}$$

It is easy to verify that

$$1) \sigma_i \cap \sigma_j = \emptyset \text{ if } i \neq j \text{ for } 1 \leq i, j \leq g_k,$$

$$2) \bigcup_{1 \leq i \leq g_k} \sigma_i = \mathbb{Z}^+.$$

Let $f(i) = \|\sigma_i \cap F\|$. Thus, from 1) and 2) above, it follows that

$$3) \sum f(i) = \|F\|.$$

If $\|F\| < g_k$ then by 1), 2), and 3), it follows that $\exists \sigma_i$ such that $\sigma_i \cap F = \emptyset$. That is, there exists a double-ended infinite sequence which does not contain any element of F . By Lemma 1, it follows that F would not then be deadly. That is, $\|F\| < g_k$ implies F is not deadly. $\square$

This theorem gives the necessary condition on the minimum number of faults required for blocking a linear array. This also tells that fewer than g_k faults occurring in the linear array will not be catastrophic. In the case where all the faults occur in a block, there is always a bypass link of size greater than the size of the block to go around the faulty area; on the other hand, if the faults are scattered, it is always possible to use a combination of some regular and bypass links for reconfiguration. An important point to be emphasized here is that the minimum number of faults to form a catastrophic fault pattern depends solely on the length of the longest bypass link, g_k , but no other intermediate link nor the number of bypass links.

In the following, all discussions will be restricted to the case where there are at least g_k faults, and catastrophic fault patterns containing exactly g_k faults will be characterized.

3.3.2 Width of Catastrophic Fault Patterns

In this section, it is shown how a catastrophic fault pattern, consisting of g_k faults, is spread out. In particular, bounds on the width of these patterns are established. The width of a catastrophic fault pattern depends on two quantities: the link configuration G , and the orientation of G . Obviously, the lower bound on the width of a catastrophic fault pattern, consisting of g_k number of faults, is g_k . This is the case when all faults occur in a cluster. The upper bound on the window size must be established.

Definition 8 *The width W_F of a fault pattern F (or fault window) is the number of PEs between and including the first and the last fault in F . That is, if $F = \{f_1, \dots, f_m\}$ then $W_F = f_m - f_1 + 1$.*

In the following, the width of the largest possible fault window is given for both unidirectional and bidirectional links.

Unidirectional Links

Theorem 2 *Let $\|F\| = g_k$ and $G = \{g_1, g_k\}$ be a set of unidirectional links. Then the width of the fault window for G is given by $W_F = (g_k - 1)^2 + 1$.*

Proof: Partition the linear array of PEs into blocks of g_k elements and list the blocks as the rows of a matrix. Pick a block arbitrarily and mark the first element in the block by an "x" (denoting the first fault). This indicates the start of the window. Observe that going from left to right in a row is same as traversing the array using the link of length 1. Similarly, moving along a column in the matrix is same as traversing the array using the link of length g_k . The construction of a catastrophic fault pattern in this matrix be such that the following properties must hold.

- Each column of the matrix must be covered, i.e., it must have an element marked "x"; otherwise, if there is a column which does not contain an "x" then there is always an escape through that column.
- Each column must have exactly one element marked "x". This is because if there are two or more elements marked "x" in one column, then there will be at least one column which will not have any element marked "x". This implies that it is possible to escape through the column that is not covered.
- Each element below (i.e., the corresponding element in the next row) an element marked "x" must be inaccessible; otherwise, if the element is accessible, there is an escape through that element using the link of length g_k . Notice that for this property to hold, the last element in the first block must be marked "x".

Now the objective is to find a distribution of "x"s in the matrix that meets the above requirements. The obvious one is, of course, the block (row) in which all g_k elements are marked "x" as $(x, x, \dots, x)$. This implies that $W_F = g_k$, the case of the smallest window. There are clearly other patterns that can give a larger window size than g_k . The only distribution of "x"s in the matrix that meets all the requirements and at the same time gives the largest window is the one shown in Figure 3.4.

Figure 3.4: Distribution of "x"s for unidirectional $G = \{1, g_k\}$

Altogether, there are $(g_k - 1)$ blocks (rows) which contains elements marked "x". Each block is of size g_k . The distance between the first and the last "x" in the distribution is W_F . It is clear that

$$W_F = g_k(g_k - 2) + 2 = g_k^2 - 2g_k + 2 = (g_k - 1)^2 + 1. \quad \square$$

It should be observed that the bound established on W_F in this theorem is the upper bound for any unidirectional link configuration $G' \supseteq G = \{1, g_k\}$. This is true because the inclusion of any additional link in G will likely shrink the window size. An algorithm to construct a catastrophic fault pattern that is contained in the largest window is outlined in Chapter 4 for unidirectional links.

Bidirectional Links

Theorem 3 *Let $\|F\| = g_k$ and $G = \{g_1, g_k\}$ be a set of bidirectional links. Then the width of the fault window for G is*

$$W_F = (\lceil \frac{g_k}{2} \rceil - 1)g_k + \lfloor \frac{g_k}{2} \rfloor + 1.$$

Proof: The proof is similar to the proof of Theorem 2. Like before, partition the linear array of PEs into blocks of g_k elements and list the blocks as the rows of a matrix. Pick a block arbitrarily and mark the first element in the block by an "x" (denoting the first fault). The construction of a catastrophic fault pattern in this matrix be such that all the previous properties (in Theorem 2) must hold. In addition, since the links are bidirectional, each element below (i.e., the corresponding element in the next row) an element marked "x" must now be inaccessible from left as well as from right. The only distribution of "x"s in the matrix that meets all the requirements and at the same time gives the largest window is the one shown in Figure 3.5.

Figure 3.5: Distribution of “x”s for bidirectional $G = \{1, g_k\}$

Altogether, there are $\lceil g_k/2 \rceil$ blocks (rows) which contains elements marked “x”. Each block is of size g_k . The distance between the first and the last “x” in the distribution is W_F . From Figure 3.5, it follows that

$$W_F = (\lceil \frac{g_k}{2} \rceil - 1)g_k + \lfloor \frac{g_k}{2} \rfloor + 1. \quad \square$$

It should be observed that the bound established on W_F in Theorem 3 is the upper bound for any bidirectional link configuration $G' \supseteq G = \{1, g_k\}$. An algorithm to construct a catastrophic fault pattern that is contained in the largest window is outlined in Chapter 4 for bidirectional links.

3.3.3 Effect of G on W_F

In this section, the effect of the number of bypass links on the size of the fault window is studied. Obviously, the width of the fault window varies with the size of G . Consider k , the size of G . When $k = 1$, there are no bypass links; hence, any single fault is catastrophic. When $k = g_k$ (a case which is very much impractical from a circuit manufacturing point of view) only a cluster of g_k faults can be catastrophic; the probability of having a cluster of g_k faults is quite small. Thus, the focus of attention is on the case where $k < g_k$. The following corollary shows that the window size decreases as the size of G increases.

Corollary 1 *Let $G' \subseteq G$ and W'_F, W_F be the corresponding widest fault windows. Then $W'_F \geq W_F$.*

Proof: There are two cases to be considered here.

- Case 1: G' is unidirectional. Theorem 2 gives the width of the largest window for $G' = \{1, g_k\}$. Since it is the upper bound on the size of W'_F , when an additional link is introduced in G' , the corresponding W'_F will not increase. Thus, for any $G \supseteq G'$, $W_F \leq W'_F$.
- Case 2: G' is bidirectional. The argument is similar and follows from Theorem 3. $\square$

Some of the fundamental results which show the sensitiveness of the width of the largest fault window to the variation in the size of G in both unidirectional and bidirectional links are presented below.

Unidirectional Links

Corollary 2 *Let $\|F\| = g_k$ and $G = \{1, 2, g_k\}$. Then the width of the widest fault window W_F is given by*

$$W_F = \begin{cases} (\frac{g_k}{2} - 1)g_k + 2 & \text{if } g_k \text{ is even} \\ (\lfloor \frac{g_k}{2} \rfloor - 1)g_k + 3 & \text{if } g_k \text{ is odd} \end{cases}$$

The corollary follows from Algorithm 1, given in Chapter 4, for the construction of the largest fault window. Figure 3.6 gives an example of a catastrophic fault pattern for $G = \{1, 2, 12\}$ that is contained in the largest window.

Figure 3.6: A catastrophic fault pattern for $G = \{1, 2, 12\}$

Corollary 3 *Let $\|F\| = g_k$ and $G \supseteq \{g_k - 1, g_k\}$ in which $3 \leq \|G\| \leq g_k - 1$. Then the width of the largest fault window $W_F = g_k$.*

The corollary follows from Algorithm 1, given in Chapter 4, for the construction of the largest fault window. This corollary states that when the link set G includes two links, one of size g_k and other of size $g_k - 1$, then any cluster of g_k faults can be catastrophic for G . Now one can ask what happens when there is a link of length $g_k - 2$ in place of $g_k - 1$. The next corollary provides an upper bound on the width of the fault window.

Corollary 4 *Let $\|F\| = g_k$ and $G \supseteq \{g_k - 2, g_k\}$ in which $\|G\| \leq g_k - 1$ but the link of length $g_k - 1$ is not available. Then the width of the fault window $W_F \leq 2g_k - 1$.*

Equality in Corollary 4 is achievable. Figure 3.7 gives an example of a G for which this is true.

Figure 3.7: A catastrophic fault pattern for $G = \{1, 7, 9\}$

Corollary 5 *Let $\|F\| = g_k$ and $G = \{1, \lceil g_k/2 \rceil, g_k\}$. Then the width of the fault window is*

$$W_F = \begin{cases} (\frac{g_k}{2} - 1)g_k + 2 & \text{if } g_k \text{ is even} \\ (\lfloor \frac{g_k}{2} \rfloor - 1)g_k + \lceil \frac{g_k}{2} \rceil + 1 & \text{if } g_k \text{ is odd} \end{cases}$$

This corollary gives the size of the fault window when bypass links of length $\lceil g_k/2 \rceil$ and g_k are available in the linear array. Figure 3.8 shows an example of a fault window when $G = \{1, \lceil g_k/2 \rceil, g_k\}$.

Figure 3.8: A catastrophic fault pattern for $G = \{1, 5, 10\}$

The following corollary is the generalization of Corollaries 2, 3, 4, and 5.

Corollary 6 *Let $\|F\| = g_k$ and $G = \{1, \alpha, g_k\}$. Also let $\beta = \alpha \bmod (g_k - \alpha)$. Then the width of the fault window is given by*

$$W_F \leq \begin{cases} (g_k - \alpha)g_k & \text{if } \beta = 0 \\ \lfloor \frac{g_k - \alpha}{\beta} \rfloor g_k + (\beta - 1)g_k & \text{if } \beta \neq 0 \end{cases}$$

In Corollary 6, equality is achievable. The example in Figure 3.9 demonstrates the case.

Figure 3.9: A catastrophic fault pattern for $G = \{1, 7, 12\}$

Bidirectional Links

In the case of bidirectional links, the results which show the sensitiveness of the width of the largest fault window to the variation of the size of G are now shown.

Corollary 7 *Let $\|F\| = g_k$ and $G = \{1, 2, g_k\}$. Then the width of the largest fault window W_F is given by*

$$W_F = \lfloor \frac{g_k}{4} \rfloor (g_k - 2) + g_k$$

The corollary follows from Algorithm 2, given in Chapter 4, for the construction of the largest fault window. Figure 3.10 gives an example of a catastrophic fault pattern for $G = \{1, 2, 12\}$ that is contained in the largest window.

Figure 3.10: A catastrophic fault pattern for bidirectional $G = \{1, 2, 12\}$

Corollary 8 *Let $\|F\| = g_k$ and $G \supseteq \{g_k - 1, g_k\}$ in which $\|G\| \leq g_k - 1$. Then the width of the fault window $W_F = g_k$.*

This corollary states that when the link set G includes two links, one of length g_k and other of length $g_k - 1$, then any cluster of g_k faults can be catastrophic for G .

Corollary 9 *Let $\|F\| = g_k$ and $G \supseteq \{g_k - 2, g_k\}$ in which $\|G\| \leq g_k - 1$ but the link of length $g_k - 1$ is not available. Then the width of the fault window $W_F \leq 2g_k - 1$.*

Note that Corollaries 8 and 9 are identical to Corollaries 3 and 4 respectively. This means that the results in Corollaries 3 and 4 hold in the case when links are used in bidirectional sense with backtracking.

3.4 Properties of Catastrophic Fault Patterns

In this section, certain properties of catastrophic fault patterns when all the faults are contained in an interval of size $2g_k$ are described.

3.4.1 Translation Property

Property 1 *Let F be blocking for G . Then, for all integers c , $F + c$ is also blocking for G .*

This tells us that if a fault pattern is blocking then the occurrence of this pattern anywhere within the array will still be blocking. That is, the window can be translated along the integer axis.

This property holds regardless of the window size.

From this translation property it follows that without loss of generality one can restrict to the case in which the g_k faults are integers in $Z[2g_k]$, where $Z[x] = \{1, 2, \dots, x\}$ for $x \in Z^+$, the set of positive integers.

3.4.2 Symmetric Property

Let $S = Z[2g_k] - F$ be the set of non-faulty or sane integers in the "window".

Theorem 4 *F is blocking for G if and only if $i + g \in F, \forall i \in S \cap Z[g_k]$ and $g \in G$ such that $i + g > g_k$.*

Proof. (only if part) By contradiction assume that there exist i in $Z[g]$ and $g \in G$ such that $i + g > g_k$ and both i and $i + g$ are in S . It is easy to see that the infinite sequence of integers $\dots, i - j * g_k, \dots, i, i + g, i + g + g_k, \dots, i + g + j * g_k, \dots$ is not in $D(j = 0, 1, 2, \dots)$ contradicting the fact that F is blocking for G .

(if part)

It will be shown by induction that, if the conditions stated in the theorem hold, all integers in S are in D . Let i be the largest integer in S . By definition of D , the

conditions of the theorem imply that i is in D . Let all the p largest integers in S be also in D , and let i be the $p + 1$ largest. Consider $i + g$, for all g in G . If $i + g$ is in F then it is also in D . Let $i + g$ be not in F . If $i + g$ is greater than g_k then by assumption $i + g$ is in D ; otherwise, it is in D by induction. The condition that all integers in S to be in D clearly implies Z^+ is in D which in turn implies that all integers are in D . Thus F is blocking G . $\square$

The theorem can be explained further as follows. Consider the non-faulty or sane elements in the first half of the window; i.e., in $Z[g_k]$. For the fault-pattern F to be blocking for G , every non-faulty element in the first half of the window must lead directly to a faulty element in the second half of the window and vice versa.

Corollary 10 *Let $i \in Z[g_k]$. Then, $i \in S$ if and only if $i + g_k \in F$.*

This corollary is a direct consequence of the above theorem. The corollary shows the symmetric property of the window and tells that when an element in the first half of the window is non-faulty its corresponding element in the second half of the window will be faulty. Conversely, a non-faulty element in the second half of the window will have a corresponding element which is faulty.

Example 2: Figure 3.11 shows two fault patterns that are contained in an interval of size $2g_k$ for link configuration $G = \{1, 3, 7, 10\}$. Both fault patterns are symmetric. That is, if an element in the first half of the window (interval) is faulty then the corresponding element in the same location of the second half of the window is non-faulty.

Figure 3.11 (a): A symmetric fault pattern for $G = \{1, 3, 7, 10\}$

Figure 3.11 (b): A symmetric fault pattern not catastrophic for $G = \{1, 3, 7, 10\}$

It should be noted that the symmetric property is a necessary condition but it is not a sufficient condition. That is, a symmetric fault pattern is not necessarily catastrophic. For example, the fault pattern given in Figure 3.11 (b) is *not* catastrophic although it is symmetric.

The previous theorem and the corollary imply that a blocking set F can be completely characterized by the non-faulty integers in the first half of the window; i.e., by $S \cap Z[g_k]$.

3.4.3 Closure Property

Definition 9 Given a subset, W , of $Z[g_k]$, let $F[W] = (Z[g_k] - W) \cup (W + g_k)$. W is called a solution for G if $F[W]$ is blocking for G . $F[W]$ is essentially the set of faulty elements in the entire window.

That is, to each solution corresponds a unique fault pattern and vice versa. Thus, by knowing all possible solutions, one can generate all possible fault patterns. Additional property given in the following corollary must hold in order for an integer set to be a solution.

Corollary 11 W is a solution if and only if $\forall i \in W$ and $\forall g \in G$ such that $i + g > g_k, i + g \bmod g_k \in W$.

This corollary illustrates the property that the elements in a solution are closed under modulus; this property can be used to verify if an integer set is a solution. Later, it will be shown that the solutions are themselves closed under union.

Example 3: Consider the set of links $G = \{1, 5, 7, 9\}$ and two sets of integers $W1 = (2, 4, 6, 8)$ and $W2 = (1, 2, 5, 7, 9)$. Use of Corollary 11 yields that $W1$ is a solution for G , but $W2$ is not.

As a consequence of the corollary the presence of integer i in a solution W "forces" other integers to be in W ; i.e., the integers $i + g \bmod g_k$ whenever $i + g > g_k$.

Definition 10 Let $W(i)$ be the set of all integers forced by i in W .

It is not difficult to verify that $W(i)$ is a solution. Denote the base set solution by $\{W(i), i = 1, 2, \dots, g_k\}$.

Presented below is an algorithm for the construction of $W(i)$. The algorithm is very simple and finds a solution forced by some integer element i .

Algorithm for construction of $W(i)$

Input : G and i

Output : $W(i)$

Begin

$W \leftarrow \text{empty}$

$P \leftarrow i$

 While nonempty P do

$W \leftarrow j \leftarrow P$

 For all g in G such that $g + j > g_k > g$ do

$P \leftarrow g + j - g_k$

 EndFor

 EndWhile

End

Repeated executions of the algorithm with $i = 1, 2, \dots, g_k$ will construct the base set. The construction can actually be simplified by using the following relation:

$$W(i) = W(g_k) - * (g_k - i)$$

where $A - * a$ denotes the set $\{x - a : x \in A \text{ and } x > a\}$ for an integer set A and a scalar a . Thus, it is sufficient to construct $W(g_k)$ using the algorithm, and then derive all other elements of the base set using a simple shift.

Example 4: Consider a linear array which has link structure $G = \{1, 3, 7, 10\}$. In this example, g_k is 10 and the window is of size 20. By applying the above algorithm, one can find $W(10)$, the solution which is forced by 10. $W(10)$ in this case turns out to be the set $(10, 7, 4, 3, 1)$. The corresponding fault pattern is $(2, 5, 6, 8, 9, 11, 13, 14, 17, 20)$ for that window. By using the relation, $W(i) = W(g_k) - * (g_k - i)$, one can find other $W(i)$, for $i = 1, 2, \dots, 9$. The following table shows the complete base set solutions.

i	$W(i)$	$F[W(i)]$
10	(10,7,4,3,1)	(2,5,6,8,9,11,13,14,17,20)
9	(9,6,3,2)	(1,4,5,7,8,10,12,13,16,19)
8	(8,5,2,1)	(3,4,6,7,9,10,11,12,15,18)
7	(7,4,1)	(2,3,5,6,8,9,10,11,14,17)
6	(6,3)	(1,2,4,5,7,8,9,10,13,16)
5	(5,2)	(1,3,4,6,7,8,9,10,12,15)
4	(4,1)	(2,3,5,6,7,8,9,10,11,14)
3	(3)	(1,2,4,5,6,7,8,9,10,13)
2	(2)	(1,3,4,5,6,7,8,9,10,12)
1	(1)	(2,3,4,5,6,7,8,9,10,11)

Table 3.1: Base set solutions for $G = \{1, 3, 7, 10\}$

The following properties illustrate the operations that can be performed on the base set solutions to derive other solutions.

Property 2 *If W is a solution then $W - *c$ is a solution.*

This is another way of stating the translation property as described in Section 3.4.1. It says that starting with a solution and performing simple shifts, one can obtain several other solutions.

Property 3 *If $W1$ and $W2$ are solutions then $W1 \cup W2$ is a solution. That is, the set of solutions is closed under union.*

Example 5: $W1 = (1, 3, 4, 7, 10)$ and $W2 = (1, 2, 5, 8)$ are two solutions for $G = \{1, 3, 7, 10\}$. $W1 \cup W2 = (1, 2, 3, 4, 5, 7, 8, 10)$ is also a solution for G .

Property 4 *Every solution can be expressed as the union of elements of the base set.*

This property indicates that every solution can be expressed as the union of elements of the base set solutions. The union of two or more elements in the base set may produce a solution that is already in the base set. Therefore, only the distinct solutions should be considered when generating all solutions in this manner.

3.5 Chapter Summary

In this chapter, fault patterns which are catastrophic for linear systolic arrays are studied. A characterization of catastrophic fault patterns is derived by treating the array as a special set of positive integers (called a *deadly integer set*). The requirement on the minimum number of faults in a fault pattern is shown to be a function of the length of the longest bypass link available, and not the total number of bypass links. Achievable bounds on the width of catastrophic fault patterns are established; results expressing the sensitiveness of the width to the amount of bypass links available are given. A special class of catastrophic fault patterns is studied in detail and is shown to have several interesting algebraic properties.

Chapter 4

Generation and Enumeration of Catastrophic Fault Patterns

4.1 Introduction

A characterization for the fault patterns that are catastrophic for linear systolic arrays has been given in the previous chapter. It is shown that for a given link configuration in the array, it is possible to identify all catastrophic fault patterns. In this chapter, the problem of generation and enumeration of catastrophic fault patterns for a given link configuration in a linear array is studied. For both unidirectional and bidirectional links, algorithms are given for constructing catastrophic fault patterns that are contained in the largest possible fault window as established in the previous chapter. Bounds are established on the number of catastrophic fault patterns for both unidirectional and bidirectional links. It is also shown how to generate all catastrophic fault patterns systematically. An algorithm for testing if a given fault pattern is catastrophic for a known link configuration is outlined in this chapter.

This chapter is organized as follows. In Section 4.2, algorithms for constructing a catastrophic fault pattern, contained in the largest fault window, are given for both unidirectional and bidirectional links. Enumeration of catastrophic fault patterns is studied in Section 4.3. In Section 4.4, algorithms are given for generating all catastrophic fault patterns for both unidirectional and bidirectional links. Finally, the problem of determining if a given pattern is catastrophic is discussed in Section 4.5.

4.2 Construction of Widest Fault Pattern

In this section, algorithms for constructing a catastrophic fault pattern, contained in the largest possible fault window, are given for both unidirectional and bidirectional links.

Some of the terminologies of interest, defined and used in Chapter 3, are now reviewed here prior to the algorithms, since these terminologies will be used throughout this chapter.

It can be recalled that

- the set $F = \{f_1, f_2, \dots, f_m\}$, $f_i < f_{i+1}$ for $1 \leq i \leq m$, corresponds to a distribution of faults (fault pattern),
- the set $G = \{g_1, g_2, \dots, g_k\}$, $g_j < g_{j+1}$ for $1 \leq j \leq k$ of positive integers is the set of links including the bypass links with $g_1 = 1$, and
- The width W_F of a fault pattern F (or fault window) is the number of PEs between and including the first and the last fault in F . That is, if $F = \{f_1, \dots, f_m\}$ then $W_F = f_m - f_1 + 1$.

4.2.1 Unidirectional Links

Algorithm 1: Construction of a Catastrophic Fault Pattern for Unidirectional Links

Input: A link configuration G

Output: A catastrophic fault pattern F

Step 1: Partition the linear array of PEs into blocks of g_k elements and list the blocks as rows of a matrix. Select a block arbitrarily, and mark the first element in the block by an “x”.

Step 2: Repeat

- Select the next block.
- Moving from left to right in the selected block, pick the unmarked element directly below the element that is marked “x” or “y” in the previous block. Mark it by a “y”. If the element in the previous block is marked by an “x” then mark all unmarked elements that are to the left at distance $g_1, g_2, \dots, g_{k-1}$ of the current element by an “x”. Note that this process can span the current as well as the previous block.

Step 3: Until all elements in the current block are marked.

Note that the elements marked “x” represent faulty PEs. The distribution of all the “x”s constitutes a catastrophic fault patten. The algorithm assigns exactly g_k number of “x”s. The number of PEs between and including the first “x” and the last “x” is the width of the fault pattern (window).

Example 1: Figure 4.1 shows a catastrophic fault pattern consisting of 8 faults corresponding to the unidirectional link configuration $G = \{1, 2, 4, 8\}$ as generated by Algorithm 1. The width of this pattern is 18.

Figure 4.1: A catastrophic fault pattern for unidirectional $G = \{1, 2, 4, 8\}$

Theorem 1 *Algorithm 1 is correct, i.e., it generates a catastrophic fault pattern.*

Proof: The following simple observations can be made on the algorithm:

- Any PE marked with an y is inaccessible, i.e., there is no way to reach this PE using any of the bypass links.
- There is only one x per column. Thus, there can be only g_k x 's.
- For each column the x occurs before the y 's.

Clearly, if the algorithm terminates then the entire row is marked with x and y , i.e., either the PE is faulty or inaccessible. Thus the algorithm is correct if it ever terminates.

Observe that location 1 and g_k of the first block have x 's. Consequently, on the second row, the last element is marked with y , so location $g_k - 1$ must be an x (because of the 1 link) unless it is already a y , i.e., location $g_k - 1$ in row 1 already has an x . Continuing row by row, it can be seen that each location from the right is gradually marked with an x . Consequently, all the PEs in the column under the x are also marked with y . Thus, eventually there is a row that is all marked. $\square$

Intuitively, by examining Algorithm 1, it can be seen that there are more x 's and y 's for G' in each row, so the algorithm terminates sooner.

Property 1 *Algorithm 1 has time complexity $O(W_F + kg_k)$ and space complexity $O(W_F)$.*

Proof: The algorithm will consider and mark elements only in the first $\lceil W_F/g_k \rceil + 1$ rows of the matrix (denote it as M) as defined by the algorithm; in particular, the last row will be entirely marked (either with "x" or "y"). This marking is performed by a linear scan of these rows together with a backtracking operation. The backtracking operation involves checking (and possibly marking) all the elements in the array at distance g_l ($1 \leq l < k$) from any element $M[i, j]$ such that $M[i - 1, j]$ is marked "x". Since there are exactly g_k elements $M[i, j]$ such that $M[i - 1, j]$ is marked "x", the backtracking will require exactly $(k - 1)g_k$ operations in addition to the scan which requires exactly $(\lceil W_F/g_k \rceil + 1)g_k$ operations. Hence, the algorithm requires $g_k(k + \lceil W_F/g_k \rceil)$ operations. The space complexity is a linear function of the size of the fault window W_F . This is because the algorithm operates on $\lceil W_F/g_k \rceil + 1$ rows, each of size g_k , hence the space complexity of $O(W_F)$. It is important to note that W_F is the width of the fault window which could, in the worst case, be of $O(g_k^2)$. $\square$

Theorem 2 *The catastrophic fault pattern generated by Algorithm 1 for unidirectional links has the widest window, W_F .*

Proof: Suppose there is a catastrophic fault pattern with a wider window W_F . Position the pattern into blocks of size g_k with f_1 at location 1 of the first block. Note that since there are only g_k faults, each column can only contain one faulty PE. This means that each column that contains a faulty PE (x) must have the rest of the column be inaccessible (y). This, in turn, forces the PEs $g_1, \dots, g_k$ distance back to be faulty or inaccessible. In other words, F must contain the same pattern as that in the algorithm. $\square$

This theorem gives a framework for achieving specific upper bounds and in some cases exact bounds on the size of the the largest window for a given link configuration. Given a link configuration G , one can obtain, by applying the above algorithm, a catastrophic fault pattern F which is contained in the largest window; that is, W_F is the maximum value possible.

4.2.2 Bidirectional Links

In the following, an algorithm is outlined for the construction of a catastrophic fault pattern F from a given set G when links are bidirectional, and backtracking is allowed in attempts to bypass faulty PEs.

Algorithm 2: Construction of a Catastrophic Fault Pattern for Bidirectional Links

Input: A link configuration G

Output: A catastrophic fault pattern F

Step 1: Partition the linear array of PEs into blocks of g_k elements and list the blocks as rows of a matrix. Select a block arbitrarily, and mark the first element in the block by an "x".

Step 2: Repeat

- Select the next block.
- Moving from left to right in the selected block, pick the unmarked element directly below the element that is marked “x” or “y” in the previous block. Mark it by a “y”. If the element in the previous block is marked by an “x” then consider all unmarked elements that are to the left at distance $g_1, g_2, \dots, g_{k-1}$ of the current element: mark each of them “x” or “y” depending on whether the corresponding element in the row above is unmarked or marked respectively. Now mark those unmarked elements that are to the right in the same row at distance $g_1, g_2, \dots, g_{k-1}$ of the current element by an “x”, ignoring the ones which appear in a column that already has an “x” above it.

Step 3: Until all elements in the current block are marked.

Example 2: Figure 4.2 shows a catastrophic fault pattern consisting of 9 faults corresponding to $G = \{1, 3, 9\}$ when the links are bidirectional using the construction outlined in Theorem 2. The width of the widest catastrophic fault pattern is 23.

Figure 4.2: A catastrophic fault pattern for bidirectional $G = \{1, 3, 9\}$

Theorem 3 *Algorithm 2 is correct, i.e., it generates a catastrophic fault pattern.*

The proof of this theorem is similar to the proof of Theorem 1, so it is omitted.

Theorem 4 *The fault pattern generated by Algorithm 2 for bidirectional links is catastrophic and has the widest window, W_F .*

The proof of this theorem is similar to the proof of Theorem 2, so it is omitted.

Property 2 *Algorithm 2 has time complexity $O(W_F + kg_k)$ and space complexity $O(W_F)$.*

Proof: The only difference between this algorithm and the one for the unidirectional case is that in addition to scanning and backtracking it also checks (and possibly marks) the elements at distance g_l ($1 \leq l \leq k$) to the right of each element $M[i, j]$ such that $M[i - 1, j]$ is marked "x". This requires an additional kg_k operations. The time complexity still remains at $O(W_F + kg_k)$. The space complexity for Algorithm 2 is comparable to that of Algorithm 1 since both algorithms operate on exactly the same number of rows of the matrix. $\square$

4.3 Enumeration of Catastrophic Fault Patterns

In the previous section, it is shown how to construct the fault pattern that is contained in the largest possible window for unidirectional and bidirectional links. For a given link configuration it is possible to identify all catastrophic fault patterns. This section establishes bounds on the number of catastrophic fault patterns for a given link configuration.

The problem of counting the number of catastrophic fault patterns is reformulated in terms of a combinatorial problem, called "parenthesized expressions" with underlying properties identical to those seen in a catastrophic fault patterns. This reformulation makes it particularly easy to establish a closed form formula for the number of catastrophic fault pattern. The link configuration considered here is $G = \{1, g_k\}$; that is, the one in which only the shortest and the longest bypass link are available. The formula for $G = \{1, g_k\}$ serves as the upper bound on the total number catastrophic fault patterns for any superset of G .

4.3.1 Bidirectional Links

The total number of catastrophic fault patterns for $G = \{1, g_k\}$ can be obtained using the following formulation. Without loss of generality, let $f_1 = 1$; that is, the first faulty element occurs in the first position in the array. Let $W_F^+ = \lceil W_F / g_k \rceil g_k$. Now consider a Boolean matrix W of size $(W_F^+ \times g_k)$ defined as follows:

$$W[i, j] = \begin{cases} 1 & \text{if } ((i-1)g_k + j) \in F \\ 0 & \text{otherwise} \end{cases}$$

In other words, W is a fault pattern as seen before. For F to be catastrophic for G , it implies that, if $W[i, j] = 1$, then one of $W[i, j+1]$, $W[i-1, j+1]$, or $W[i+1, j+1]$ is equal to 1 (over valid indices).

Construct now a sequence $S_1, S_2, \dots, S_{\|F\|}$ of non-zero elements of W as follows:

- $S_1 = W[1, 1]$ and $S_{\|F\|} = W[1, g_k]$,
- If $S_l = W[i, j]$ then S_{l+1} is the sole non-zero element among $W[i, j+1]$, $W[i+1, j+1]$, and $W[i-1, j+1]$.

Associate with the above sequence $S_1, S_2, \dots, S_{\|F\|}$, a string μ on the alphabet $\{ (,), * \}$ as follows:

- $\mu(S_1) = "("$ and $\mu(S_{\|F\|}) = ")"$,
- given $S_l = W[i, j]$, $l < \|F\| - 1$, $\mu(S_{l+1})$ is $"*"$, $"("$, or $)"$ depending on whether $S_{l+1} = W[i, j+1]$, $W[i+1, j+1]$, and $W[i-1, j+1]$ respectively.

The resulting string μ for a catastrophic F has the property of being *well-formed* with respect to the symbols $"("$, and $)"$: that is,

1. for any j , $1 \leq j \leq \|F\|$, the number of open parentheses, denoted by $\|L_j\|$, in the substring $S_1, S_2, \dots, S_j$ is always greater than or equal to the number of closed parentheses, denoted by $\|R_j\|$, in the same subsequence; and
2. the total number of $"("$, denoted by $\|L\|$, in μ is equal to the total number of $)"$, denoted by $\|R\|$, in μ .

The “*” symbol can occur in any place in μ except in $\mu(S_1)$ and $\mu(S_{\|F\|})$, which by definition are “(” and “)”, respectively. It is not difficult to show that, conversely, any such well-formed string corresponds to a catastrophic fault pattern. Hence, the total number of catastrophic fault patterns for G can be determined by counting the number of well-formed strings where $\|L\| = \|R\| \leq \lfloor g_k/2 \rfloor$ and $\|L\| + \|R\| + \|Z\| = g_k$, where $\|Z\|$ denotes the number of “*” in the string.

Some additional terminologies must now be introduced.

Let Σ_q denote the set of well-formed strings on $\{ (,) \}$ of length $2q$; obviously, every $\mu \in \Sigma_q$ contains exactly q “(” and q “)” symbols.

Let $\Xi_{p,q}$ denote the set of well-formed strings of length p on $\{ (,), * \}$ symbols; obviously, if $\mu \in \Xi_{p,q}$, then it contains exactly $p - 2q$ “*” symbols. $\Xi_{p,q}$ is the set of interest, and the objective is to find the cardinality of the set $\Xi_{p,q}$.

The set $\Xi_{p,q}$ can be generated without repetition from Σ_q by inserting in all possible ways $p - 2q$ “*” symbols in each element of Σ_q . Hence, the cardinality of the set $\Xi_{p,q}$ is equal to the cardinality of Σ_q times the number of ways $p - 2q$ indistinguishable balls can be inserted into $2q - 1$ distinguishable baskets. The number of ways in which $p - 2q$ indistinguishable balls can be inserted into $2q - 1$ distinguishable baskets is

$$\binom{2q - 1 + p - 2q - 1}{p - 2q} = \binom{p - 2}{p - 2q}$$

The size of Σ_q is known to be the q -th Catalan number [51]:

$$\|\Sigma_q\| = \frac{1}{q + 1} \binom{2q}{q}$$

Hence,

$$\|\Xi_{p,q}\| = \frac{1}{q + 1} \binom{2q}{q} \binom{p - 2}{p - 2q}$$

The total length of the string under consideration is equal to g_k ; that is, $p = g_k$. Let Ξ_{g_k} denote the set of all such well-formed strings on $\{ (,), * \}$, of length g_k . Now, in each of these strings there is at least one open and one closed parenthesis. Therefore,

$$\|\Xi_{g_k}\| = \sum_{i=1}^{\lfloor g_k/2 \rfloor} \|\Xi_{g_k,i}\| = \sum_{i=1}^{\lfloor g_k/2 \rfloor} \frac{1}{i+1} \binom{2i}{i} \binom{g_k-2}{g_k-2i}$$

which proves the following

Theorem 5 *Let $G = \{1, g_k\}$. Then the number of catastrophic fault patterns Γ_G is given by*

$$\Gamma_G = \sum_{i=1}^{\lfloor g_k/2 \rfloor} \frac{1}{i+1} \binom{2i}{i} \binom{g_k-2}{g_k-2i}$$

Example 3: Consider $G = \{1, 5\}$. In this case, by using, Theorem 5, the value of $\Gamma_G = 9$. All nine catastrophic fault patterns for $G = \{1, 5\}$ are shown in Figure 4.3.

Theorem 6 *Given an arbitrary G' , the number of catastrophic fault patterns $\Gamma_{G'}$ is given by*

$$\Gamma_{G'} \leq \sum_{i=1}^{\lfloor g_k/2 \rfloor} \frac{1}{i+1} \binom{2i}{i} \binom{g_k-2}{g_k-2i}$$

Proof: By Corollary 1 in Chapter 3, if $G' \supseteq G$ with $W'_{F'}, W_F$ being their corresponding widest fault windows, then $W'_{F'} \leq W_F$; in other words, the window size decreases as the size of G increases. By Theorem 2 in Chapter 3, $G = \{1, g_k\}$ has the widest window. Therefore, $\Gamma_{G'} \leq \Gamma_G$. By Theorem 5 above, the claim follows. $\square$

Figure 4.3: All catastrophic fault patterns for bidirectional $G = \{1, 5\}$

4.3.2 Unidirectional Links

The formulation of in this case is similar to the case of bidirectional links except for some exceptions described below.

The definition for the Boolean matrix W of size $(W_F^+ \times g_k)$ is same as before. Now, for F to be catastrophic for an unidirectional G , it implies that, if $W[i, j] = 1$, then one of $W[i - 1, j + 1]$, $W[i, j + 1]$, or $W[k, j + 1]$, $k > i$, is equal to 1 (over the valid indices).

As before, construct the sequence $S_1, S_2, \dots, S_{\|F\|}$ of non-zero elements of W . Associate with this sequence, a string χ on the alphabet $\{(f,), *\}$ as follows. Given $S_l = W[i, j]$, $l < \|F\|$, define

$$\chi(S_{l+1}) = \begin{cases} "*" & \text{if } S_{l+1} = W[i, j+1] \\ ")" & \text{if } S_{l+1} = W[i-1, j+1] \\ "(f" & \text{if } S_{l+1} = W[i+f, j+1], f \geq 1 \end{cases}$$

Notice that the open parenthesis has a weight associated with it. A "(_f" denotes a block of f open parentheses corresponding to the modified definitions of the Boolean matrix W . Now define the cost $c(S_i)$ of an element in the sequence to be

$$c(S_i) = \begin{cases} f & \text{if } \chi(S_i) = "(f" \\ 0 & \text{if } \chi(S_i) = "*" \\ -1 & \text{if } \chi(S_i) = ")" \end{cases}$$

The resulting string χ associated to a catastrophic fault pattern F has the property of being *well-formed* in the following sense:

1. for any subsequence $S_1, S_2, \dots, S_j$, $1 < j \leq \|F\|$, starting with S_1 ,

$$\sum_{i=1}^j c(\chi(S_i)) \geq 0 \text{ i.e., } \|L\| > \|R\|, \text{ and}$$

2. for the entire sequence $S_1, S_2, \dots, S_{\|F\|}$,

$$\sum_{i=1}^{\|F\|} c(\chi(S_i)) = 0.$$

As before, the "*" symbol can occur in any place in χ except in $\chi(S_1)$ and $\chi(S_{\|F\|})$. It is not difficult to show that, conversely, any such well-formed string corresponds to a catastrophic fault pattern. Hence, the total number of catastrophic fault patterns for G can be determined by counting the number of well-formed strings as defined above.

Example 4: An example of a well-formed string on $\{(f,) : f \in Z\}$ is "(₄))(₂*))". In this case, the sum of the weights on all open parentheses is at least the number of close parentheses at any point in the string, and the sum of the weights on all open parentheses is equal to the number of close parentheses for the entire string. An example of a string on $\{(f,) : f \in Z\}$ which is not well-formed is "(₂))) * (₃*)*)".

Let $\Upsilon_{p,q}$ denote the set of all well-formed strings on $\{(f,) : f \in Z\}$ of length p and which contains exactly q open parentheses, $2q \geq p \geq q + 1$. A well-formed string of length g_k on the alphabet $\{(f,), *\}$ can be easily constructed from an element in $\Upsilon_{p,q}$ by throwing $g_k - p$ indistinguishable “*” in between the p elements of the original string. Therefore, the total number of well-formed strings on the alphabet $\{(f,), *\}$ is

$$\|\Omega_{g_k}\| = \sum_{q=1}^{g_k-1} \sum_{p=q+1}^{\text{Min}\{2q, g_k\}} \|\Upsilon_{p,q}\| \binom{g_k-1}{g_k-p}$$

Unlike the case of bidirectional links, the exact value of $\|\Upsilon_{p,q}\|$ is not known. On the other hand, a lower bound on $\|\Omega_{g_k}\|$ can be obtained by observing that any catastrophic fault pattern for bidirectional link is also catastrophic for the unidirectional case. Hence, the results of the previous section are lower bounds for the case considered here.

4.4 Generation of Catastrophic Fault Patterns

In Section 4.2, algorithms are given for the construction of a catastrophic fault pattern that is contained in the largest possible window. Starting with the largest window it is possible to generate all catastrophic fault patterns that are contained in windows of smaller sizes. A more general algorithm is shown in this section which generates all catastrophic fault patterns systematically for a given link configuration G without making any reference to the largest fault window.

4.4.1 Bidirectional Case

Algorithm 3: Generation of all Catastrophic Fault Patterns for Bidirectional Links

Step 1: Generate all possible well-formed expressions on $\{(,)\}$ alphabets containing i open parentheses and i closed parentheses ($1 \leq i \leq \lfloor g_k/2 \rfloor$) of total length g_k .

Step 2: For each expression formed in Step 1, throw in $g_k - 2i$ “*” symbols in between the $2i$ parentheses in all possible ways.

Each string generated in Step 2 corresponds to a new catastrophic fault pattern for the bidirectional case.

Property 3 *Algorithm 3 generates each catastrophic fault pattern for bidirectional links in time complexity $O(g_k)$ and storage complexity $O(g_k)$.*

Proof: Each well-formed expression on $\{ (,) \}$ can be generated in time linear in the size of the expression [51]. A catastrophic fault pattern is generated by throwing the “*” symbols into a well-formed expression. The process of throwing “*” symbols is done in time which is a linear function of the number of “*” symbols. Hence, the generation of a catastrophic fault pattern can be achieved in time which is a linear function of the size of the pattern since the size is equal to the number of parentheses and “*” symbols. With this time complexity, it is not necessary to store all previously computed well-formed expressions on $\{ (,) \}$ nor it is required to store the number of ways of throwing “*” symbols into expressions of a given size. Hence, the entire execution of the algorithm will use a storage of size $O(g_k)$. $\square$

4.4.2 Unidirectional Case

Algorithm 4: Generation of all Catastrophic Fault Patterns for Unidirectional Links

Step 1: Generate all possible well-formed expressions on $\{“(f”, “)” : f \in Z\}$ alphabets containing q open parentheses ($1 \leq q \leq g_k - 1$) and of total length p ($q + 1 \leq p \leq \text{Min}\{2q, g_k\}$).

Step 2: For each expression formed in Step 1, throw $g_k - p$ “*” in between the p symbols in all possible ways.

Each string generated in Step 2 corresponds to a new catastrophic fault pattern for the unidirectional case.

Property 4 *Algorithm 4 generates each catastrophic fault pattern for unidirectional links in $O(g_k)$ time and $O(g_k)$ space.*

Proof: Similar to Property 3.

4.5 Determining if a Pattern is Catastrophic

Determining if a given fault pattern F is catastrophic for a link configuration G is a trivial one. It can be done as follows.

- Form a Boolean matrix W of size $(W_F^+ \times g_k)$, where $W_F^+ = \lceil W_F / g_k \rceil g_k$, in which all elements are marked 0,
- Let $W[1, 1]$ be marked 1 to correspond to the first fault f_1 in F . Starting with f_2 onwards, mark the corresponding entries in W by a 1, so that the entire fault pattern is mapped onto the Boolean matrix W ,
- Verify if each entry in W satisfies the properties defined in Section 4.3 for the corresponding link configuration G . If there is no violation, then the given fault pattern F is catastrophic; otherwise, F is not catastrophic.

The entire process can obviously be done using a number of operations and storage linear in the pattern (i.e., in $O(\|F\|)$ time and space).

Example 5: Consider a fault pattern given in Figure 4.4. To test if the fault pattern is catastrophic for the link configuration $G = \{1, 10\}$ in which the links are bidirectional, the first step is to map the fault pattern onto a Boolean matrix W as shown in Figure 4.5. The corresponding string of weighted parentheses is “(((*)()))” which is well-formed. Therefore, the given fault pattern in Figure 4.4 is catastrophic for the link configuration $G = \{1, 10\}$ in which the links are bidirectional.

Figure 4.4: Fault pattern #1

	f_1						f_2		f_3	
	1	0	0	0	0	0	0	1	0	1
	0	1	0	0	0	0	1	0	1	0
	0	0	1	0	1	1	0	0	0	0
	0	0	0	1	0	0	0	0	0	0
	f_{10}									

Figure 4.5: Boolean matrix for fault pattern #1

Example 6: Figure 4.6 shows a fault pattern which is to be tested if it is catastrophic for the link configuration $G = \{1, 10\}$ in which the links are unidirectional. the fault pattern is mapped onto a Boolean matrix as shown in Figure 4.7. The corresponding string of weighted parentheses is “ $((()(*))_3)$ ” which is *not* well-formed. Therefore, the given fault pattern in Figure 4.6 is *not* catastrophic for the link configuration $G = \{1, 10\}$ in which the links are unidirectional.

Figure 4.6: Fault pattern #2

$$\begin{array}{cccccccccc}
 & f_1 & & & & & & f_2 & & f_3 \\
 \left[\begin{array}{cccccccccc}
 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
 \end{array} \right] \\
 & & & & & & & & & f_{10}
 \end{array}$$

Figure 4.7: Boolean matrix for fault pattern #2

4.6 Chapter Summary

In this chapter, the problem of generation and enumeration of catastrophic fault patterns for a given link configuration in a linear array has been studied. For both unidirectional and bidirectional links, algorithms are given for constructing catastrophic fault patterns that are contained in the largest possible fault window as established in Chapter 3. Bounds are established on the number of catastrophic fault patterns. Algorithms are presented for generating all catastrophic fault patterns systematically for both unidirectional and bidirectional links and for determining if a given fault pattern is catastrophic for a given link configuration.

Chapter 5

Limits to Reconfigurability of Systolic Architectures

5.1 Introduction

In systolic arrays, the peripheral PEs are normally responsible for I/O operations. Faults occurring at strategic locations can prevent the computation to proceed, by disrupting I/O operations or even segmenting the structure into pieces which can no longer be suitable for any practical purpose. In this chapter, the properties of catastrophic fault patterns and their windows are used to study limits to reconfigurability of one- and two-dimensional arrays. The catastrophic fault patterns are employed as means to derive fault-tolerant attributes in terms of the minimum number of faults sufficient to cause a catastrophic failure. The minimum number of faults required to have an impact on reconfigurability is derived for different modes of computations, requiring a minimum number of connected fault-free processors. All negative results established in this chapter hold even in the most benign situation; i.e., even if faults are detectable, reconfiguration is instantaneous, only correct information is generated, no transient failures occur, etc.

The chapter is organized as follows. Section 5.2 gives basic definitions and assumptions. Models of computation are also discussed in this section. Sections 5.3 and 5.4 outline the fault-tolerant characteristics of one- and two-dimensional systolic arrays respectively for each of the specified model.

5.2 Preliminaries

This chapter deals with two fundamental systolic structures: one-dimensional (or linear) and two-dimensional (or mesh) arrays of homogeneous processing elements (PEs) as shown in Figure 5.1 and 5.2 respectively. Each PE includes a multiplier, an adder, some control registers, and some logic and fault detection circuits.

The connections (or links) can be either unidirectional in which case information flows from one end to another in the direction of the arrows or bidirectional in which case the links can be used in either direction. Input and output PEs (normally the peripheral ones) are distinguishable in the unidirectional case unlike in the bidirectional case where there is no such clear distinction. The connections can be of type *regular* or *bypass*. Regular connections exist between neighboring PEs while the bypass connections are assumed to exist between non-neighbors for bypassing faulty PEs or links. The bypass connections are considered to be the redundant links.

The PEs not active in the absence of faults are called *spares*. They have no specific designation that distinguishes them from the active PEs. The amount of PEs normally available in a single chip far outnumbers the number required for solving a problem, and that leaves sufficient number of spares for arrays of any dimension.

The basic sources of chip-related failures for VLSI can also be grouped as a) process related, and b) design related. The process related defects are those which happen during the integrated circuit (IC) processing while the design related defects are those which show up during the design phase of the chip. As a result of these defects, individual components in random or components in form of clusters (or blocks) with no definite pattern may develop within a chip. A fault in a PE can be described as its inability to perform correct operation on correct inputs, possibly due to some elementary circuit or gate-level faults. The failures (defects) could be either permanent or transient. In this chapter, the focus will be only on process related permanent random and block failures at PE level.

A complete characterization of catastrophic fault patterns has been given in Chapter 3. The characterization has revealed certain properties that influence the study of reconfigurability of one- and two-dimensional systolic arrays. Some of the important properties mentioned in Chapter 3 that are relevant to the discussion are summarized below. Let F and G be a catastrophic fault pattern and a link configuration respectively as defined in Chapter 3.

1. $\|F\| \geq g_k$. That is, a necessary condition on the minimum number of faulty nodes to form a blocking pattern for a given link configuration G is g_k . Note that a fault pattern is said to be catastrophic with respect to a link configuration when the structure is logically separated.
2. If the number of faulty nodes is restricted to g_k (i.e., $\|F\| = g_k$), then all g_k faulty nodes must be located within a region (fault "window") with width $W_F \geq g_k$. Also, the size of the largest fault window is given by

$$W_F \leq \begin{cases} (g_k - 1)^2 + 1 & \text{for unidirectional links} \\ ([g_k/2] - 1)g_k + [g_k/2] + 1 & \text{for bidirectional links} \end{cases}$$

3. The window size, W_F varies with the size of G (i.e., the number of bypass links). In other words, the window size tends to shrink as the number of bypass links increases.

Figure 5.1: 1-Dimensional array of PEs

Figure 5.2: 2-Dimensional array of PEs

Some definitions now follow.

Definition 1 *Given a problem X , let $\|X\|$ denote the minimum number of connected PEs required to compute X . $\|X\|$ is called the size of the problem.*

Definition 2 *A “ (k, m) – Fault Tolerant” system is one, where k PEs have failed, but it is still possible to solve problems of all size n , for which $n < m < N$ (total number of PEs).*

Definition 3 *The system is operating normally if it is $(0, N)$ – Fault Tolerant.*

Definition 4 *A “catastrophic” failure is said to have occurred if the system is $(k, 0)$ – Fault Tolerant.*

The analysis will be based on the direction of information flow in and out of one- and two-dimensional systolic arrays. As special case, the systolic structures where it may be possible to download data and upload result in some miraculous way are considered. The three models are the following (see Figure 5.3):

- a) Systems with unidirectional links
- b) Systems with bidirectional links (I/O can be in one side or both)
- c) Systems with bidirectional links as in (b), and also has miracle I/O facilities

Figure 5.3: Three basic models

Only the PE failures are considered here. In this case, isolated PE faults could occur at random (scattered), or they could occur in a block (a row/column, part of a row/column, or portion of an area). All these faults occur mostly at run-time environment. All negative results established in this chapter hold even in the most benign situation; i.e., even if faults are detectable, reconfiguration is instantaneous, only correct information is generated, no transient failures occur, etc.

5.3 One-Dimensional Systolic Arrays

In this section, fault-tolerance aspects in one-dimensional systolic arrays with both unidirectional and bidirectional connections are studied. Block faults of varying sizes are considered, and the relationship between the size of the block faults and the bypass connections are established. Bypass connections are assumed to exist to and from each PE, but are not shown in subsequent figures for simplicity.

Theorem 1 *For a problem of size, $\|X\| = n$, a one-dimensional array with N PEs and $G = \{1, 2, \dots, g_k\}$ unidirectional links is $(g_k, 0)$ -Fault Tolerant.*

Proof: Since $G = \{1, 2, \dots, g_k\}$, the only fault patterns that can be catastrophic are the clusters of g_k faults. Consider a block fault of size g_k , as shown in Figure 5.4, occurring somewhere in a linear array. The faulty region is shadowed. Let x and y be two PEs occurring in immediate left and right of the faulty region respectively. There is no bypass link from x to y since the longest bypass link from x takes to a PE in the faulty region. In this case, flow of data from x cannot reach y and hence the output. Therefore, occurrence of a block fault of size g_k (i.e., g_k faults) is enough to disconnect the entire array when the bypass links are of size g_k or less.

The result still holds if the problem size is specified, i.e., $\|X\| = n$. By Definition 1, n fault-free connected PEs are needed to solve the problem. But, the fault block could just happen to be in the input side (see Figure 5.5), not leaving enough fault-free PEs for the computation to proceed. $\square$

Figure 5.4: 1-D array of PEs with a block fault

Figure 5.5: 1-D array with block fault in input side

Theorem 2 For a problem of size, $\|X\| = n$, a one-dimensional array with N PEs and $G = \{1, 2, \dots, g_k\}$ bidirectional links is $(\text{Min}\{2g_k, N - 2(n - 1)\}, 0)$ - Fault Tolerant.

Proof: Two cases will be considered here.

- Case 1: $\{N \geq 2(n - 1 + g_k)\}$. Position two fault blocks, each of size g_k , in both side of the array (as shown in Figure 5.6) such that there are less than n fault-free PEs in either end. By the previous theorem (Theorem 1), these two blocks of faults would stop the flow of data beyond the faulty region into the array. In addition, fault-free PEs in either end, by Definition 1, are insufficient for the problem. The total number of faults sufficient to disconnect the array is equal to $2g_k$.

- **Case 2:** $\{N < 2(n - 1 + g_k)\}$ In this case less than $2g_k$ faults, i.e., $N - 2(n - 1)$, occurring in one large block, leaving fewer than n non-faulty PEs (see Figure 5.7) in either side of the array which are insufficient for a problem of size, $\|X\| = n$. Therefore, the total number of faults is simply $N - 2(n - 1)$. $\square$

Figure 5.6: 1-D bidirectional array with blocked I/O

Figure 5.7: 1-D bidirectional array with $< 2g_k$ faults

The result given in Theorem 2 will still hold even if faults do not occur in blocks of size g_k . This can be seen in the following corollary.

Corollary 1 Let $G' \subset G = \{1, 2, \dots, g_k\}$. For a problem of size $\|X\| = n$, a one-dimensional array with N PEs and the bidirectional link configuration G' is $(\text{Min}\{2g_k, N - 2(n - 1)\}, 0)$ - Fault Tolerant.

Proof: Since $G' \subset G = \{1, 2, \dots, g_k\}$, there are other fault patterns besides the block (or cluster) faults that can be catastrophic. The only difference in the case of G' is that the number of fault patterns is at least one, and each fault pattern has the window size $W_F \geq g_k$. The window containing the fault pattern is equivalent to a

Definition 4 A fault pattern F is catastrophic for C if $C - F$ is not connected.

In other words, the ring is disconnected when the faulty nodes along with their incident links are removed.

Definition 5 A blocking decomposition of F is a set $\mathcal{F} = \{F_1, F_2, \dots, F_p\}$, where

- 1) each F_i is blocking,
- 2) $\forall j, k \ F_j \cap F_k = \emptyset$, and
- 3) $\bigcup_i F_i = F$.

Definition 6 The blocking dimension of F , denoted by $\delta(F)$, is the size of the smallest blocking decomposition of F .

Theorem 1 F is catastrophic for $C = (G, N)$ if and only if $\delta(F) \geq 2$.

Proof: (if part) It will be proved that if $\delta(F) = 1$, then $C - F$ is connected and, thus, F is not catastrophic. Consider two arbitrary non-faulty nodes x and y in C . Since F is blocking, it is possible that there is no path from x to y in $C^+ - F$; should this be the case, such a path trivially exists in $C^- - F$. Hence, x and y are connected in $C - F$. Since x and y are arbitrary non-faulty nodes, it follows that $C - F$ is connected if $\delta(F) = 1$.

(only if) It suffices to show that if $\delta(F) = 2$ then F is catastrophic. Let $\mathcal{F} = \{F_1, F_2\}$ be a blocking decomposition of F . Let S_1 (S_2) be the set of non-faulty nodes between F_1 and F_2 (F_2 and F_1) according to the clockwise direction as shown in Figure 6.2. Notice that $S_1 \neq \emptyset \wedge S_2 \neq \emptyset$; otherwise, $\delta(F) = 1$. Let $x \in S_1$ and $y \in S_2$. Since F_1 and F_2 are blocking, there is no path from x to y in $C^+ - F$ nor in $C^- - F$; conversely, there is no path from y to x in $C^+ - F$ nor in $C^- - F$. That is, there is no path between x and y in $C - F$. Since x and y are arbitrary nodes in S_1 and S_2 respectively, it follows that S_1 and S_2 are not connected; that is, $C - F$ is disconnected. Hence F is catastrophic. $\square$

fault block in the sense that both contain the same number of faulty PEs. This, in turn, does not change the minimum number of faults required to logically disconnect the array. $\square$

The last remaining model is the case of miraculous I/O. In this model it is assumed that each PE is accessible and can provide I/O facilities. The number of faults required in this case to make such a structure fail will certainly be higher than in the case of normal I/O. Furthermore, the number of faults also depends on the size of the fault window W_F . The following theorem demonstrates that.

Theorem 3 *Given a problem of size, $\|X\| = n$, an one-dimensional array with a total of N PEs, $G = \{1, 2, \dots, g_k\}$ bidirectional links, and provision of miraculous I/O facilities is $(\alpha, 0)$ -Fault Tolerant, where $\alpha = \lfloor N/(n-1+g_k) \rfloor g_k + \text{Min}\{0, [N \bmod (n-1+g_k)] - n + 1\}$.*

Proof : Consider the linear array given in Figure 5.8. In this structure, first $(n-1)$ PEs are fault-free, and the next g_k PEs are faulty. Let the pattern of $(n-1)$ fault-free and g_k faulty PEs continue until $\lfloor N/(n-1+g_k) \rfloor$ such patterns are seen. Out of the remaining $[N \bmod (n-1+g_k)]$ PEs, if this number is greater than $(n-1)$, there should be a block of $(n-1)$ fault-free PEs with the remaining as a block of faulty PEs. The structure now has blocks of $(n-1)$ fault-free PEs that are disconnected from other fault-free blocks. By Definition 1, it is not possible to compute $f(X)$ for which $\|X\| = n$. The total number of faults is, therefore, $\lfloor N/(n-1+g_k) \rfloor g_k + \text{Min}\{0, [N \bmod (n-1+g_k)] - n + 1\}$. $\square$

Figure 5.8: 1-D array with alternating block faults

Now what remains for investigation is the the case where $G' \subset G = \{1, 2, \dots, g_k\}$; that is, the link configuration does not contain all the bypass links up to size g_k . Let W_F and W'_F be the largest fault window for G and G' respectively. It has been shown in Chapter 3 that $W'_F \geq W_F$. Consider the case where $W'_F > W_F$, and let F_i be the catastrophic fault pattern consisting of g_k faults contained in the window W'_F . Since $W'_F > g_k$, there exists at least one non-faulty PE in F_i . Let l and r be the number of non-faulty nodes that can be accessed in F_i from left side and right side respectively. Obviously, $l, r \geq 0$.

Corollary 2 *Let $G' \subset G = \{1, 2, \dots, g_k\}$. For a problem of size, $\|X\| = n$, an one-dimensional array with a total of N PEs, bidirectional link configuration G' , and provision of miraculous I/O facilities is $(\beta, 0)$ - Fault Tolerant, where*

$$\beta = \lfloor \frac{N - n + l + 1}{n - 1 + g_k} \rfloor g_k + \text{Min}\{0, [(N - n + l + 1) \bmod (n - 1 + g_k)] - n + r + 1\}.$$

Proof: From left to right of the one-dimensional array, label the array as follows. Mark the first $n - l - 1$ PEs be marked non-faulty. In the remaining array of size $N - n + l + 1$ mark alternately the pattern in W'_F and $n + g_k - 1$ fault-free PEs until $\lfloor (N - n + l + 1) / (n - 1 + g_k) \rfloor$ such patterns are seen (see Figure 5.9). Out of the remaining $[(N - n + l + 1) \bmod (n - 1 + g_k)]$ PEs, if this number is greater than $(n - r + 1)$, there should be a block of $(n - r + 1)$ fault-free PEs with the remaining as a block of faulty PEs. The array now has blocks of $(n - 1)$ fault-free PEs that are disconnected from other fault-free blocks. By Definition 1, it is not possible to compute $f(X)$ for which $\|X\| = n$. Counting the number of faults in the labelled array carefully, it is clear that the total number of faults is

$$\lfloor \frac{N - n + l + 1}{n - 1 + g_k} \rfloor g_k + \text{Min}\{0, [(N - n + l + 1) \bmod (n - 1 + g_k)] - n + r + 1\}. \quad \square$$

Figure 5.9: 1-D array with alternating fault patterns

5.4 Two-Dimensional Systolic Arrays

In this section, fault-tolerance attributes of two-dimensional systolic arrays with both unidirectional and bidirectional connections have been studied. Block faults of varying sizes are considered, and the relationship between the size of the block faults and the bypass connections are established. Bypass connections are assumed to exist to and from each PE, but are not shown in subsequent figures for simplicity.

In one-dimensional array of PEs, it was seen that the length of the longest bypass link must be at least g_k if there is a cluster of g_k faults. A fault cluster in one-dimensional array of PEs spans just one dimension whereas it can span both dimensions (row and column) in a two-dimensional array of PEs. The following theorem shows what the length of the longest bypass link must be in order to reconfigure around the faulty region in a two-dimensional array of PEs.

Theorem 4 *For a fault cluster of size K faults or any small fraction of K , the length of the longest bypass link has a lower bound $\Omega(\sqrt{K})$; that is, no optimal configuration scheme will be able to tolerate upto K faults if the the longest bypass link is $\sqrt{K}$ long, regardless of the number of spares and the size of the array.*

Proof: Consider the case where the faults occur in a contiguous area (cluster) at one corner of the processing array. In particular, consider the cluster having a square shape as shown in Figure 5.10. In such a case, for any active PE to simulate the function of the faulty PE at the extreme corner, the bypass distance cannot be less than or equal to $\sqrt{K}$. The lower bound is exact in the sense that there are worst case reconfiguration algorithms for which such lower bound performance results. $\square$

Figure 5.10: $\sqrt{K} \times \sqrt{K}$ cluster faults in 2-D array

Theorem 5 A two-dimensional systolic array of $M \times N$ PEs with $G = \{1, 2, \dots, g_k\}$ unidirectional links is $(\text{Min}\{Mg_k, Ng_k\}, 0)$ – Fault Tolerant regardless of the problem size.

Proof: Consider the worst case situation where g_k contiguous columns (or rows) are faulty as shown in Figure 5.11. The PEs in two sides of the g_k contiguous faulty columns (or rows) will be $g_k + 1$ distance apart. Hence, there is no connection that takes from one side of the faulty area to another without passing a faulty PE. Clearly, g_k contiguous faulty columns (or rows) can disconnect the entire array. In order to maintain the locality of reference, the length of longest bypass link must be at least $g_k + 1$; the number of faults is therefore $\text{Min}\{Mg_k, Ng_k\}$. $\square$

The result still holds even when not all g_k continuous columns (or rows) are faulty. In Figure 5.12, there are g_k non-contiguous faulty columns that corresponds to a fault

pattern. In other words, there is a fault pattern in each row of the two-dimensional array. It is clear that each row is useless for the flow of data going from left to right.

Figure 5.11: Cluster faults in 2-D array

Figure 5.12: g_k non-contiguous faulty columns

In the case where the links are bidirectional, the peripheral PEs can be used for both input and output functions. To prevent the array from solving a problem of certain size, it would require at least all four sides be blocked. Consider the two-dimensional array shown in Figure 5.13 in which all four sides are blocked. Although all four sides are blocked, it still might be possible to solve the problem with the remaining non-faulty PEs adjacent to any side. It, therefore, has to be ensured that there does not exist sufficient number of connected PEs to solve the problem of a given size. The following theorem establishes the number of faults needed to render a two-dimensional array useless for this purpose.

Figure 5.13: 2-D array with both faulty rows and columns

Theorem 6 Given a problem of size, $\|X\| = n \times n$, a two-dimensional array with a total of $N \times N$ PEs and a bidirectional link configuration $G = \{1, 2, \dots, g_k\}$ is $(\gamma, 0)$ -Fault Tolerant, where

$$\gamma = \left\lceil \frac{N}{n + g_k - 1} \right\rceil 2g_k + 4(N - 2n - g_k - 1)g_k.$$

Proof: The objective is to block all four sides of the two-dimensional array and at the same time not leaving sufficient connected PEs adjacent to an I/O side. When fault clusters are well-placed in the array to meet this objective, the resulting pattern takes the shape as shown in the Figure 5.14. The width of the fault cluster is simply g_k since $G = \{1, 2, \dots, g_k\}$. Each blocked square near the periphery (I/O side) contains $(n - 1) \times (n - 1)$ fault-free PEs which is insufficient for a problem of size $n \times n$. By counting the number of faulty PEs in the shaded area carefully, it is easy to show that the total number of faults γ in this case is

$$\gamma = \left\lceil \frac{N}{n + g_k - 1} \right\rceil 2g_k + 4(N - 2n - g_k - 1)g_k.$$

Therefore, the theorem follows. $\square$

Figure 5.14: Fault bands in a bidirectional 2-D array

5.5 Chapter Summary

The effects of processor failures in one- and two-dimensional systolic architectures on computations requiring a fixed number of processors are described in this chapter. Three different models of computations are analyzed. Catastrophic fault patterns are used to derive fault-tolerant characteristics in terms of the minimum number of faults required to cause a catastrophic failure in each of these structures. All results established in this chapter are negative, and they hold even in the most benign situation; i.e., even if faults are detectable, reconfiguration is instantaneous, only correct information is generated, no transient failures occur, etc.

Chapter 6

Limits to Reconfigurability of Ring Topologies

6.1 Introduction

A common technique to improve reliability of loop (or ring) networks is by introducing link redundancy; that is, by providing several alternative paths for communication between pairs of nodes. With alternate paths between nodes, the network can now sustain several node and link failures by bypassing the faulty components. In this chapter, fault patterns that are catastrophic for loop networks with a given degree of redundancy are identified. Bounds are established on the degree of fault tolerance achievable in a redundant loop network when performing a computation that requires a minimal number of operational nodes. Also the bounds on the size of the problems one can be guaranteed to solve in the presence of a given number of faults in the network are derived.

The organization of this chapter is as follows. In Section 6.2, basic concepts are introduced and definitions are provided. Fault-tolerance analysis of general chordal ring networks is given in Section 6.3. In Section 6.4, limits to reconfiguration of loop topologies are studied followed by the chapter summary in Section 6.5.

6.2 Preliminaries

The systems being studied in this chapter are the redundant ring networks, particularly the chordal rings as shown in Figure 6.1. The basic components of such a system are the *processing elements* (PEs) and the *links*. From a topological point of view, a chordal ring is a circular arrangement of PEs with the addition of extra links (or chords) connecting some of the PEs. There are two kind of links in redundant ring networks: *regular* or *bypass*. The distinction between regular links (edges of the ring) and bypass links (the chords) is a functional one: the bypass links are used strictly for reconfiguration purposes when a fault is detected; in the absence of faults, the bypass links are redundant and only regular links are used. The links can be either unidirectional or bidirectional, but in the following only the bidirectional links are considered.

Figure 6.1: A chordal ring

A formal definition of generalized chordal rings now follows.

Definition 1 Given a set $\mathcal{P} = \{P_0, P_1, \dots, P_{N-1}\}$ of N nodes and a set $G = \{g_1, g_2, \dots, g_k\}$ of positive integers, where $g_i < g_{i+1}$, a generalized chordal ring on $\mathcal{P}$ with link configuration G is an undirected graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ where the set $\mathcal{E}$ of edges is

$$\mathcal{E} = \{(P_i, P_j) : \exists g \in G, (i = j + g \bmod N) \vee (i = j - g \bmod N)\}$$

In the following, a node P_i will be denoted simply by i . The generalized chordal ring (or simply, ring) on $\mathcal{P}$ with link configuration G will be denoted by the couple (G, N) , where $N = \|\mathcal{P}\|$. Observe that in a chordal ring (G, N) , every node has the same degree $d = 2\|G\|$. Since the focus is on redundant ring structures, without loss of generality, it will be always assumed that $g_1 = 1$. It is also assumed that the number of nodes N is much larger than the degree of the network.

Two well known examples of generalized chordal rings are the simple ring $R(N) = (\{1\}, N)$ and the Fully Connected Network $K(N) = (\{1, 2, 3, \dots, N/2\}, N)$.

Definition 2 Given a ring $C = (G, N)$, a fault pattern for C is a set of integers $F = \{f_1, f_2, \dots, f_m\}$ where $m \leq N$, $f_j < f_{j+1}$ and $f_j \in [0, N)$.

The set $F \subseteq \mathcal{P}$ represents the set of faulty PEs in C .

A clockwise (counter clockwise) orientation of $C = (G, N)$, denoted by C^+ (C^-), is the directed graph obtained by orienting all arcs in C in a clockwise (counter clockwise) direction with respect to the ring $(\{1\}, N)$.

Let $C^+ - F$ ($C^- - F$) denote the graph obtained by removing from C^+ (C^-) all nodes in F and their incident edges.

Definition 3 A fault pattern F is blocking for C if neither $C^+ - F$ nor $C^- - F$ is strongly connected.

In other words, the occurrence of such a fault pattern in C blocks the flow of information from one side of the pattern to the other through the faulty region in any fixed direction.

Definition 4 A fault pattern F is catastrophic for C if $C - F$ is not connected.

In other words, the ring is disconnected when the faulty nodes along with their incident links are removed.

Definition 5 A blocking decomposition of F is a set $\mathcal{F} = \{F_1, F_2, \dots, F_p\}$, where

- 1) each F_i is blocking,
- 2) $\forall j, k \ F_j \cap F_k = \emptyset$, and
- 3) $\bigcup_i F_i = F$.

Definition 6 The blocking dimension of F , denoted by $\delta(F)$, is the size of the smallest blocking decomposition of F .

Theorem 1 F is catastrophic for $C = (G, N)$ if and only if $\delta(F) \geq 2$.

Proof: (if part) It will be proved that if $\delta(F) = 1$, then $C - F$ is connected and, thus, F is not catastrophic. Consider two arbitrary non-faulty nodes x and y in C . Since F is blocking, it is possible that there is no path from x to y in $C^+ - F$; should this be the case, such a path trivially exists in $C^- - F$. Hence, x and y are connected in $C - F$. Since x and y are arbitrary non-faulty nodes, it follows that $C - F$ is connected if $\delta(F) = 1$.

(only if) It suffices to show that if $\delta(F) = 2$ then F is catastrophic. Let $\mathcal{F} = \{F_1, F_2\}$ be a blocking decomposition of F . Let S_1 (S_2) be the set of non-faulty nodes between F_1 and F_2 (F_2 and F_1) according to the clockwise direction as shown in Figure 6.2. Notice that $S_1 \neq \emptyset \wedge S_2 \neq \emptyset$; otherwise, $\delta(F) = 1$. Let $x \in S_1$ and $y \in S_2$. Since F_1 and F_2 are blocking, there is no path from x to y in $C^+ - F$ nor in $C^- - F$; conversely, there is no path from y to x in $C^+ - F$ nor in $C^- - F$. That is, there is no path between x and y in $C - F$. Since x and y are arbitrary nodes in S_1 and S_2 respectively, it follows that S_1 and S_2 are not connected; that is, $C - F$ is disconnected. Hence F is catastrophic. $\square$

Figure 6.2: A blocking decomposition $\mathcal{F} = \{F_1, F_2\}$

Observe that any F which is blocking for a ring $C = (G, N)$ is a catastrophic fault pattern for a linear array A (denoted by $A = (G, N')$) on $N' \geq N$ nodes with link configuration G and vice versa. In particular, each F_i in the blocking decomposition $\mathcal{F}$ is a catastrophic fault pattern for a linear array $A = (G, N')$. Therefore, the results established in Chapter 3 on catastrophic fault patterns for linear array $A = (G, N')$ can be applied to the elements of a blocking decomposition of the corresponding ring $C = (G, N)$ with $N \leq N'$. Some of the important properties stated in Chapter 3 and which are relevant to the discussion here are summarized below.

1. $\|F_i\| \geq g_k$. That is, a necessary condition on the minimum number of faulty nodes to form a blocking pattern for a given link configuration G is g_k .
2. If the number of faulty nodes is restricted to g_k (i.e., $\|F_i\| = g_k$), then all g_k faulty nodes must be located within a region, called fault "window"; the size of the largest fault window is given by

$$W_F \leq \begin{cases} (g_k - 1)^2 + 1 & \text{for unidirectional links} \\ (\lceil g_k/2 \rceil - 1)g_k + \lfloor g_k/2 \rfloor + 1 & \text{for bidirectional links} \end{cases}$$

3. The window size, W_F varies with the size of G (i.e., the number of bypass links).

In particular, the window size tends to shrink as the number of bypass links increases.

Algorithms to generate and enumerate catastrophic fault patterns systematically for a given link configuration in a processor array were given in Chapter 4. Figure 6.3 shows a fault pattern of size four which is catastrophic for a linear array with $G = \{1, 4\}$ bidirectional links, occurring as a blocking pattern in the ring $(G, 20)$.

Figure 6.3: A blocking fault pattern

6.3 Fault-Tolerance Analysis of Chordal Rings

As discussed in the previous section, the occurrence of a set of more than one disjoint blocking patterns will logically disconnect a ring structure. However, disconnection of a ring does not mean that the entire structure is unusable. In fact, it still might be possible to find one or more subrings or linear arrays in the original ring in spite of the failure of some nodes. These subrings and linear arrays can be utilized in computations requiring fewer processing nodes. This section deals with the degree of fault tolerance that can be achieved in a ring in the presence of node failures. The degree of fault tolerance is characterized in terms of bounds on the sizes of the subrings and linear arrays it might still be possible to obtain in the presence of faults.

The analysis focuses on complete chordal rings; that is, rings in which the specified link configuration $G = \{1, 2, \dots, g_k\}$. This case has been studied in the literature, and the existing results follow as corollary of the results presented here.

As discussed in Chapter 3, the only pattern that can be blocking for $C = (G, N)$ with link configuration $G = \{1, 2, \dots, g_k\}$ is a cluster of at least g_k faulty nodes. In this section, the focus is, therefore, on block (or cluster) of faults.

It should be recalled that the degree d of $C = (G, N)$ is $2\|G\|$ by definition; obviously, for $G = \{1, 2, \dots, g_k\}$, $d = 2g_k$. For this case, Tyszer [132] has studied the problem of constructing a linear array in the ring when $\|F\| < d = 2g_k$. Tyszer's result is formally described by the following lemma.

Lemma 1 *If $\|F\| < 2g_k - 2$, it is possible to create a linear array $A = (G, N - \|F\|)$.*

The lemma is proved in [132]. Nothing beyond this result is known. In particular, no result (e.g., characterization of the sizes of subrings or connected components) is available where there are more than $(d - 1)$ faults in the system. In this section, an extensive characterization of fault-tolerance of chordal rings with $G = \{1, 2, \dots, g_k\}$ is provided, and several properties are revealed. As shown later, Tyszer's result follows as a simple corollary to one of the results established here.

A decomposition $\mathcal{F} = \{F_1, F_2, \dots, F_p\}$ of F with link configuration $G = \{1, 2, \dots, g_k\}$ is a partition of F into clusters of faults having at least one non-faulty element between any two of them. Note that $\mathcal{F}$ is not necessarily a blocking decomposition of F .

Let $\mathcal{F}$ be a decomposition of F . Choose the clockwise direction and partition the nodes of the ring $C = (G, N)$ into sets of consecutive nodes

$$F_1, S_1, F_2, S_2, \dots, F_p, S_p$$

where all nodes in F_i are to the left of the nodes in S_i (which consists solely of non-faulty nodes), and all nodes in S_i are to the left of the nodes in F_{i+1} ; all operations are modulo p . Let $l_i = \|F_i\|$ and $s_i = \|S_i\|$. For all $j < g_k$, let $\Phi_j = \|\{F_i : l_i = g_j\}\|$ and $\Phi_{\max} = \|\{F_i : l_i \geq g_k\}\|$. For example, $\Phi_i =$ the number of blocks of length g_i .

Theorem 2 For a ring $C = (G, N)$, $\Phi_{\max} = 0$ implies that there exists a subring $C' = (G, N - \|F\|)$ regardless of the number of faults $\|F\|$.

Proof: $\Phi_{\max} = 0$ implies that there is no fault cluster of size g_k or larger. That is, $l_i < g_k \forall F_i \in \mathcal{F}$. Since $G = \{1, 2, \dots, g_k\}$, there exists a link of length $l_i + 1$ to bypass the fault cluster F_i . It is clear that a ring can be formed on the remaining $N - \|F\|$ nodes by using links of length 1 and links of length $l_i + 1$, where $l_i = \|F_i\|$ and $F_i \in \mathcal{F}$. $\square$

Obviously, if there exists a ring on $(N - \|F\|)$ nodes, there exists a linear array on $(N - \|F\|)$ nodes. From now on, only the general case where $\Phi_{\max} \geq 0$ will be considered.

Theorem 3 For a ring $C = (G, N)$, there exists a linear array $A = (G, \alpha)$ regardless of the number of faults $\|F\|$, where

$$\alpha = \frac{N - \|F\|}{\text{Max}\{1, \Phi_{\max}\}}.$$

Proof: If $\Phi_{max} = 0$, the theorem follows from Theorem 2. Let $\Phi_{max} > 0$. Let $F_{i1}, F_{i2}, \dots, F_{iz}$ be the faulty clusters of size g_k or larger, where $z = \Phi_{max}$. Removal from C of these clusters will disconnect the ring into z connected components, $C_1, C_2, \dots, C_z$. Let c_i denote the number of non-faulty elements in C_i . Without loss of generality, let $c_1 \geq c_i$, $1 < i \leq z$; that is, C_1 is the connected component containing the largest number of non-faulty elements. Then, $c_1 \geq (N - \|F\|)/z$. Without loss of generality, let C_1 be

$$S_1, F_2, S_2, F_3, \dots, S_{q-1}, F_q, S_q.$$

An array on these S_i 's can be constructed as follows:

- 1) Connect every node in S_i using regular links (length = 1).
- 2) Connect S_i to S_{i+1} using a bypass link of length $l_{i+1} + 1$; this is clearly possible since $l_{i+1} = \|F_{i+1}\| < g_k$. $\square$

Note that $\|F\| < 2g_k - 2$ implies $\Phi_{max} = 1$. Thus, by Theorem 3, there exists a linear array $A = (G, N - \|F\|)$. In other words, the result by Tyszer [132], expressed by Lemma 1, follows as a simple corollary of Theorem 3.

In the following, necessary and sufficient conditions for constructing subrings in a given ring $C = (G, N)$ in the presence of faults are described. The conditions hold regardless of the number of faults $\|F\|$.

Let $\mathcal{X}_{a,b}$ be a subsequence $S_a, F_{a+1}, S_{a+1}, \dots, F_b, S_b$ (all operations are modulo N) in C . Let the number of fault-free nodes in the $\mathcal{X}_{a,b}$ be denoted by $\pi(\mathcal{X}_{a,b})$; that is,

$$\pi(\mathcal{X}_{a,b}) = \sum_{i=a}^b \|S_i\|.$$

$\mathcal{X}_{a,b}$ is permissible if

1. $\|S_i\| \geq 2$ for $a < i < b$,
2. $\|F_i\| + 1 < g_k$ for $a < i \leq b$, and

3. $\pi(\mathcal{X}_{a,b}) > 2$.

Let Π be the largest permissible sequence; i.e., $\pi(\Pi) \geq \pi(\mathcal{X}_{a,b})$, and let $\pi = \pi(\Pi)$ for simplicity. Π is the largest string of alternating fault-free and faulty clusters (starting and ending with a fault-free cluster) in which each faulty cluster has fewer than g_k faulty nodes, and each fault-free cluster (except for the ones at both ends) has at least two fault-free nodes.

Theorem 4 *There exists a subring $C' = (G, \pi)$ regardless of the number of faults $\|F\|$.*

Proof: The theorem will be proved by showing that there exists a construction for $C' = (G, \pi)$. The outline of the construction is the following. Starting from the leftmost element of S_j , label all the elements in Π alternatively R (for "right") and L (for "left"). Connect each R (L) element to the nearest R (L) element to the right as follows. The R (L) elements within the same S_i are linked using links of length 2; the last R (L) element of S_i is linked to the first R (L) element of S_{i+1} with a link of length $\|F_{i+1}\| + 2$. This can always be done since $\|F_{i+1}\| \leq g_k - 2$.

To form a ring, it is sufficient to connect the first R with the first L element in S_j , and the last R with the last L element in S_q . Since all the fault-free elements in Π are labeled R or L , a subring on π nodes therefore exists in $C = (G, N)$. $\square$

Corollary 1 *Let $\Phi_{\max} = 1$. If $\|\Pi\| = 2p - 1$, then there exists a subring $C' = (G, N - \|F\|)$ regardless of the number of faults $\|F\|$.*

Proof: It follows from the Theorem 4 where

$$\pi = \sum_{i=1}^p \|S_i\| = N - \|F\|. \quad \square$$

The case where only the first condition for Π (i.e., $\|S_i\| \geq 2$) is met will now be considered.

Corollary 2 *Let $\Phi_{\max} + \Phi_{k-1} = \alpha$. If $\|S_i\| \geq 2$, $1 \leq i \leq p$, then there exists a subring $C'' = (G, (N - \|F\|)/\alpha)$ regardless of the number of faults $\|F\|$.*

Proof: Let $F_{i1}, F_{i2}, \dots, F_{i\alpha}$ be the fault clusters of size $g_k - 1$ or greater. Let S_{ij} denote the union of all blocks of non-faulty elements between F_{ij} and $F_{i(j+1)}$. Every S_j in S_{ij} contains at least two elements. Therefore, by Theorem 4, it is possible to construct a subring of size $\|S_{ij}\|$. By the *pigeon hole* principle, there exists S_{ij} such that $\|S_{ij}\| \geq (N - \|F\|)/\alpha$, and the claimed property follows. $\square$

The above theorem and corollaries establish sufficient conditions for a subring to be formed. These conditions are in general not the necessary ones. It is however possible to construct examples where these conditions do not hold, and no subring of non-trivial size can be constructed. Observe that, given any $\mathcal{X}_{a,b}$, if $\forall i$ ($a \leq i \leq b$) $\|F_i\| + 1 \geq g_k$ (that is, the second condition of the permissibility is violated everywhere in $\mathcal{X}_{a,b}$), then the only possible subring which can be formed solely within $\mathcal{X}_{a,b}$ is the trivial one of size $\max\{\|S_i\| : a \leq i \leq b\}$. Analogously, given any $\mathcal{X}_{a,b}$, if $\forall i$ $\|S_i\| = 1$ (that is, the first condition for permissibility is violated everywhere in $\mathcal{X}_{a,b}$), then no subring can be formed solely within $\mathcal{X}_{a,b}$.

A different sufficient condition can be established even in the cases in which single non-faulty elements are enclosed between faulty ones (i.e., $\|S_i\| = 1$), a case not considered by the previous results. Let Λ be the largest subsequence

$$S_j, F_{j+1}, S_{j+1}, \dots, F_q, S_q$$

in C such that $\|F_i\| + \|F_{i+1}\| + 1 < g_k$, for $j \leq i < q$. Let the number of fault-free nodes in the sequence Λ be denoted by λ ; that is,

$$\lambda = \sum_{i=j}^q \|S_i\|.$$

Theorem 5 *There exists a subring $C'' = (G, \lambda)$ regardless of the number of faults $\|F\|$.*

Proof: The theorem will be proved by showing that there exists a construction for $C'' = (G, \lambda)$. The outline of the construction is the following. Starting from the leftmost element of S_j , label all the elements in Λ alternatively R (for "right") and

L (for “left”). This is the same labelling technique that is used in proving Theorem 4. Connect each R (L) element to the nearest R (L) element to the right. It is now shown that this is always possible. Consider three arbitrary consecutive non-faulty elements a, b , and c ; without loss of generality, let them be labelled R , L , and R respectively.

- Case 1: $a, b, c \in S_i$. If this is the case, connect a and c using link of length 2.
- Case 2: $a \in S_i$ and $c \in S_{i+1}$. There is the fault cluster F_{i+1} in between S_i and S_{i+1} . It is possible to connect a and c , since $d(a, c)$, the distance between a and c , is $\|F_{i+1}\| + 1$ which is clearly less than g_k . Thus, a and c can be connected.
- Case 3: $a \in S_i, b \in S_{i+1}$ and $c \in S_{i+2}$. This implies that $\|S_{i+1}\| = 1$. Since S_{i+1} is in between F_{i+1} and F_{i+2} , $d(a, c) = \|F_{i+1}\| + \|F_{i+2}\| + 1$ which is again less than g_k . Thus, a and c can be connected.

To form a ring, it is sufficient to connect the first R with the first L element in S_j , and the last R with the last L element in S_q . Since all the fault-free elements in Λ are labeled R or L , a subring on λ nodes therefore exists in $C = (G, N)$. $\square$

Corollary 3 *Let $\Phi_{max} = 1$. If $\|\Lambda\| = 2p - 1$, then there exists a subring $C'' = (G, N - \|F\|)$ regardless of the number of faults $\|F\|$.*

Proof: It follows from the Theorem 5 where

$$\lambda = \sum_{i=1}^p \|S_i\| = N - \|F\|. \quad \square$$

6.4 Limits to Reconfigurability

In the previous section, bounds on the size of subrings as well as linear arrays possible in a ring of processors in the presence of known number of faults were established. This section takes into consideration computations on rings as well as linear arrays requiring a fixed number of fault-free nodes and studies the inherent limits to reconfigurability.

Some definitions now follow.

Definition 7 A ring-problem X of size $\|X\| = n$ is a problem whose solution requires a ring of size at least n .

Definition 8 An array-problem X of size $\|X\| = n$ is a problem whose solution requires a linear array of size at least n .

Definition 9 A chordal ring $C = (G, N)$ is " (k, m) - Ring Intolerant" if in the presence of k faults, it is not possible to solve all ring-problems of size m or greater.

Definition 10 A chordal ring $C = (G, N)$ is " (k, m) - Array Intolerant" if in the presence of k faults, it is not possible to solve all array-problems of size m or greater.

An obvious but useful property linking ring intolerance and array intolerance is the following.

Lemma 2 If a chordal ring $C = (G, N)$ is (α, β) - Array Intolerant then it is also (α, β) - Ring Intolerant.

Theorem 6 A ring $C = (G, N)$ is $(\alpha g_k, \beta)$ - Array Intolerant, where $\alpha \geq 1$ and $\beta = \lceil N/\alpha \rceil - g_k + 1$.

Proof: Let F be a blocking fault pattern with the widest fault window W_F , consisting of g_k faults, for a link configuration G in $C = (G, N)$. From the properties of the width of a fault pattern as stated in Chapter 3, $W_F \geq g_k$. Without loss of generality,

let $W_F > g_k$; this means, there is at least one non-faulty elements in F . Let l and r be the number of non-faulty elements that can be accessed in F from left and right of F respectively. Obviously, $l, r \geq 0$ and $l + g_k + r = W_F$. Let $F_1, F_2, \dots, F_\alpha$, be α blocking fault patterns, each consisting of exactly g_k faults. There are a total of αg_k number of faults in $C = (G, N)$. Now place α blocking patterns $F_1, F_2, \dots, F_\alpha$ in C in such a way that F_i 's are equally spaced as shown in Figure 6.4. The number of non-faulty elements between any two blocking patterns F_i and F_{i+1} , is

$$\left\lceil \frac{N - \alpha W_F}{\alpha} \right\rceil + l + r = \left\lceil \frac{N - \alpha W_F}{\alpha} \right\rceil + W_F - g_k = \left\lceil \frac{N}{\alpha} \right\rceil - W_F + W_F - g_k = \left\lceil \frac{N}{\alpha} \right\rceil - g_k.$$

This implies that in the occurrence of αg_k faults in C , a linear array can be formed on $\lceil N/\alpha \rceil - g_k$ non-faulty elements, but not on $\lceil N/\alpha \rceil - g_k + 1$ non-faulty nodes. Therefore, it follows that C is $(\alpha g_k, \lceil N/\alpha \rceil - g_k + 1)$ - Array Intolerant. $\square$

Figure 6.4: Ring showing α blocking patterns

In other words, that it is always impossible to solve an array-problem of size $\|X\| = n$ in the presence of αg_k faults if $n > \lceil N/\alpha \rceil - g_k$.

Corollary 4 *A ring $C = (G, N)$ with $G = \{1, 2, \dots, g_k\}$ is $(\xi g_k, \tau)$ -Array Intolerant, where $\xi \geq 1$ and $\tau = \lceil (N - \xi g_k) / \xi \rceil + 1$.*

Proof: It follows from Theorem 6 since the width W_F of a blocking pattern for the link configuration $G = \{1, 2, \dots, g_k\}$ is g_k . $\square$

It can be observed that Corollary 4 also follows from Theorem 3, where $\|F\| = \xi g_k$ and $\Phi_{max} = \xi$. In the previous section, the problem of constructing subrings in $C = (G, N)$ has been addressed. It is also shown what sizes of subrings can be formed in the presence of faults when $G = \{1, 2, \dots, g_k\}$. Therefore, the intolerance of a ring can be seen and rephrased in terms of the largest subring possible in C .

6.5 Chapter Summary

An extensive characterization of fault-tolerance in ring topologies is given in this chapter. This characterization augments the results known in the literature to date. In fact, it is shown that some of the existing results follow as simple corollaries to the ones established in this chapter. The characterization has revealed several properties which describe the problem of constructing subrings and linear arrays in the presence of node failures in the original ring for a specified link configuration. Also in this chapter, bounds on the degree of fault tolerance that can be achieved in a redundant ring network when performing a computation (on a linear array as well as on a ring) that requires a fixed number of operational nodes are established. Bounds on the size of the problems one can be guaranteed to solve in the presence of a given number of faults in the network are also derived.

Chapter 7

Summary and Conclusions

7.1 Summary of Main Contributions

The results of this thesis sheds some light on the understanding of real fault tolerance (or better yet, fault intolerance) of redundancy-based regular architectures. The results emphasized the fact that the effectiveness of a redundancy-based fault tolerance scheme could not simply be measured by the amount of redundancy built into the system, but also by the ability of the system to reconfigure in the presence of faults and carry on with normal operation. Therefore, redundancy and reconfigurability are two key aspects of fault tolerance.

The reconfigurability of a large class of redundancy-based regular architectures is studied in this thesis. In particular, the limits to reconfigurability by identifying and characterizing fault patterns that have catastrophic consequence in the regular architectures are derived. The work in this thesis was motivated by the observation that faults occurring at strategic locations in a regular architecture can be catastrophic and cannot be overcome by any clever design nor by any intelligent reconfiguration mechanism.

The main contribution of this thesis was a complete characterization of minimal catastrophic fault patterns for a linear array. The characterization revealed several important properties. One of the important properties emphasized the point that the number of faults needed to construct a catastrophic pattern is the function of the

length of the longest bypass link but not of the total number of bypass links. Bounds were established on the width of catastrophic fault patterns, and results expressing the sensitiveness of the width to the amount of bypass links available were presented. Algorithms were provided for constructing a catastrophic fault pattern contained in the widest window and for generating and enumerating all catastrophic patterns for a given link configuration, both in unidirectional and bidirectional case. Also, a simple algorithm was provided to test whether a given fault pattern is catastrophic for a given link configuration.

The properties of the catastrophic fault patterns were used to study reliability and limits to reconfigurability of regular architectures such as one-dimensional systolic arrays, two-dimensional systolic arrays, and ring topologies. In the case of one- and two-dimensional systolic arrays, bounds of impossibility for reconfiguration were established for systems with 1) unidirectional links, 2) bidirectional links, and 3) even for the case where each PE could act as an I/O port. The impossibility results would hold even if the faults were detectable, and transient faults never occurred.

In the case of redundant ring topologies, both positive and negative results were established. The positive results refer to the conditions for the reconfigurability of the structure in the presence of node failures. The proofs were constructive and, hence, reconfiguration algorithms could easily be derived. The negative results were analogous to the ones for linear array. These results described some of the inherent limits to reconfigurability for such topologies.

7.2 Open Problems

The following two problems are left as open problems:

1. Recall that the problem of counting the exact number of catastrophic fault patterns was mapped into a more general version of the well-known problem of counting the number of well-formed parenthesized expressions. It was able to derive a closed form formula for the number of catastrophic fault patterns for

the case of bidirectional links, but not for the case of unidirectional links. It still remains as an unsolved problem.

2. In this thesis, the characterization of catastrophic fault pattern was essentially the problem of finding a set of PEs that would be catastrophic for a given link configuration. This leads to the problem (which was not in the scope of this thesis) of finding a link configuration for which a given fault pattern would be catastrophic. This problem is nontrivial.

7.3 Applications

The contributions of the thesis can be judged from its theoretical importance, practical importance, and its applicability to the existing architectures.

From a theoretical viewpoint, the results expressed precise bounds on the computability in the presence of failures in large classes of regular architectures employing redundancy and reconfiguration techniques. In particular, limits to reconfigurability were established for linear arrays, two-dimensional arrays, and ring networks under several models of communication. These results are either the first known (as in the case of one- and two-dimensional arrays) or extend and generalize the existing results (as in the case of ring networks) for the considered class of systems.

Furthermore, the thesis revealed an interesting and unsuspected relationship between catastrophic fault patterns and well-formed sequences of weighted parentheses. The results obtained on these sequences extended in a non-trivial way the existing results for the unweighted case. Besides its theoretical interest, this correspondence made the design of optimal algorithms possible for determining whether a fault pattern is catastrophic, and for generating all catastrophic fault patterns for a given link configuration.

From a practical viewpoint, the results carry potential to answer questions about the "guaranteed" level of fault tolerance of a design, something which was not possible

before. Guaranteed fault tolerance indicates positive answers to questions such as i) will the system withstand up to k faults always regardless of how and where they occur? ii) will the system withstand up to k faults with probability ϵ regardless of how and where the faults occur? The results in this thesis provided a set of tools which could be employed in partially addressing the following issues: 1) assessing the fault tolerance effectiveness of a design, 2) testing whether a design meets the specified fault tolerance requirements, and 3) determining redundancy requirement for the designer to meet a desired level of fault tolerance. A complete design tool will require consideration of link failures which is beyond the scope of this thesis.

The results presented in this thesis apply to a large variety of commercially available array processors such as Geometric Arithmetic Parallel Processor (GAPP) [16] of NCR, Distributed Array Processor (DAP) [109] of ICL, England, NASA's Massively Parallel Processor (MPP) [11], and Connection Machine [130] of Thinking Machines Corporation in identifying the answers to questions regarding the guaranteed level of fault tolerance. The results also apply to a large number of WSI-based and VLSI-based processor arrays which include the Systolic Arrays [61], Reconfigurable Array of Processors ELSA (European large SIMD array) [122], and a variety of special-purpose VLSI and experimental WSI devices for applications such as signal processing, image processing, and numerical computations. Furthermore, the results are equally applicable to the design of fault-tolerant memory chips [35].

Furthermore, the results in this thesis could facilitate incorporation of the knowledge of application field into the design process. In particular, knowledge of the type and distribution of faults occurring in the application environment could be used to determine appropriate design. Other possible practical application could be in the post-manufacturing assessment phase. Since in VLSI-based systems faults are introduced by the manufacturing process, the guaranteed level of fault tolerance of the resulting system might differ from that of the original design; the results could be useful in assessing the new level of fault tolerance and thus aiding in the decision making process of whether to accept or reject a fabricated chip.

A more detailed description of these as well as other practical implications of the results in this thesis can be found in [89].

7.4 Areas of Future Work

Only the PE failures are considered in this thesis. The results can still be directly applied in the presence of link failures (e.g., by inscribing the failure of a link to one of its incident PEs, by considering the dual graph of the network topology, etc.). A future direction for research is to extend and generalize the model so to include link failures explicitly.

The results in this thesis assumed the underlying topology of the network to be regular. A direction for future research is to extend the results to other regular architectures and various kinds of irregular architectures. It should be noted that as link failures are considered in conjunction with PE failures in a regular architecture, the resulting topology could become irregular. In regular architectures, the regularity depends on the connectivity; therefore, in this work, the structures remain regular when the PEs fail.

Removing these two restrictions will be the focus of the future research which will give a new dimension to understanding the inherent limits to reconfigurability of any computer architectures, regular or irregular. Furthermore, the future work should also focus on solving the open problems suggested in the thesis.

Glossary

(α, β) —Array Intolerant

A system is said to be (α, β) —Array Intolerant if in the presence of α faults, it is not possible to solve all array-problems that require β connected fault-free processing elements.

Array Problem

An array-problem is a problem whose solution requires an array.

Blocking

A fault pattern is called blocking for a link configuration if it is catastrophic.

Blocking Decomposition

A decomposition of a fault pattern into disjoint clusters of consecutive faults where each cluster is blocking.

Blocking Dimension

The number of clusters in a blocking decomposition.

Bypass Link

A link that is used to bypass a faulty processing element.

Catastrophic Fault Pattern

A fault pattern is said to be catastrophic for a link configuration, if the removal of the faults and the incident links disconnects the structure.

Deadly

A fault pattern is said to be deadly for a link configuration if the fault pattern is catastrophic for that link configuration.

Fault Pattern

An ordered set that contains a distribution of faults is called a fault pattern.

(α, β) -Fault Tolerant

A (α, β) -Fault Tolerant system is one, where α processing elements have failed, but it is still possible to solve problems that require β connected fault-free processing elements.

Fault Window

A fault window is the interval that contains all the faults of a fault pattern. The fault window starts with the first fault and ends with the last fault in the fault pattern.

Link Configuration

An ordered set of links including all the bypass links for each processing element.

Parenthesized Expression

An expression on the alphabet $\{ (,), * \}$.

(α, β) -Ring Intolerant

A system is said to be (α, β) -Ring Intolerant if in the presence of α faults, it is not possible to solve all ring-problems that require β connected fault-free processing elements.

Ring Problem

A ring-problem is a problem whose solution requires a ring.

Solution

A solution corresponds to a fault pattern.

Spare

A non-faulty processing element which is not active during a computation is called a spare for that computation.

Weighted Parenthesized Expression

An expression on the alphabet $\{(/,), *\}$ in which each open parenthesis is associated with a positive weight (≥ 1).

Well-formed Parenthesized Expression

A parenthesized expression in which the number of open parentheses is at least the number of close parentheses at any point in the expression, and the number of open parentheses is equal to the number of close parentheses for the entire expression.

Bibliography

- [1] J. A. Abraham and K. Fuchs, "Fault and Error Models for VLSI," *Proc. of the IEEE*, Vol. 74, No. 5, May 1986, pp. 639-654.
- [2] V. Acciaro, A. Nayak, and N. Santoro, "On the Fault-Tolerance of Reconfigurable VLSI Systolic Arrays," *Congressus Numerantium*, Vol. 68, May 1989, pp. 197-210.
- [3] V. Acciaro, A. Nayak, and N. Santoro, "Catastrophic Faults for Reconfigurable Systolic Arrays," Tech. Report, Center for Parallel and Distributed Computing, Carleton University, Ottawa, Canada, 1989.
- [4] G. A. Anderson and E. D. Jenson, "Computer Interconnection Structures : Taxonomy, Characteristics and Examples," *ACM Computing Surveys*, Dec. 1975, pp. 197-213.
- [5] C. J. Anfinson and F. T. Luk, "A Linear Algebraic Model of Algorithm-based Fault Tolerance," in *Proc. Int'l Conf. on Systolic Arrays*, 1988, pp. 483-493.
- [6] A. Antola, R. Negrini, M. Sami, and N. Scarabottolo, "Policies for Fault-Tolerance Through Mixed Space and Time Redundancy in Semi-Systolic FFT Arrays," in *Proc. Int'l Conf. on Systolic Arrays*, 1988, pp. 565-576.
- [7] B. W. Arden and H. Lee, "Analysis of Chordal Ring," *IEEE Trans. on Computers*, Vol. C-30, No. 4, April 1981, pp. 291-295.
- [8] B. W. Arden and H. Lee, "A Regular Network for Multicomputer Systems," *IEEE Trans. on Computers*, Vol. C-31, No. 1, Jan. 1982, pp. 60-69.

- [9] A. Avizienis, "The N-Version Approach for Fault-Tolerant Systems," *IEEE Trans. on Software Engineering*, Vol. SE-11, No. 12, Dec. 1985, pp. 1491-1501.
- [10] A. E. Barbour and A. S. Wojcik, "A General, Constructive Approach to Fault-Tolerant Design Using Redundancy," *IEEE Trans. on Computers*, Vol. C-38, No. 1, Jan. 1989, pp. 15-29.
- [11] K. E. Batcher, "Design of a Massively Parallel Processor," *IEEE Trans. on Computers*, Vol. C-29, No. 9, Sept. 1980, pp. 836-840.
- [12] K. P. Belkhale and P. Banerjee, "Reconfiguration Strategies in VLSI Processor Arrays," in *Proc. Int'l Conf. on Computer Design*, 1988, pp. 418-421.
- [13] S. Chakravarty and S. J. Upadhyaya, "A Unified Approach to Designing Fault-Tolerant Processor Ensembles," in *Proc. Int'l Conf. on Parallel Processing*, 1988, pp. 339-342.
- [14] M. Chean and J. A. B. Fortes, "A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays," in *IEEE Computer*, Vol. 23, No. 1, Jan. 1990, pp. 55-69.
- [15] C. Chen, A. Feng, Y. Takada, T. Kikuno, and K. Torii, "Spare Processor Assignment for Reconfiguration of Fault-Tolerant Arrays," *The Trans. of the IEICE*, Vol. E 73, No. 8, Aug. 1990, pp. 1247-1256.
- [16] E. L. Cloud, "The Geometric Arithmetic Parallel Processor," in *Proc. 2nd Symp. on the Frontiers of Massively Parallel Processing*, Fairfax, VA, Oct. 1989.
- [17] J. R. Day, "A Fault-Driven, Comprehensive Redundancy Algorithm," *IEEE Design and Test*, June 1985, pp. 35-44.
- [18] A. Despain and D. Patterson, "X-Tree: A Tree Structured Multiprocessor Computer Architecture," in *Proc. Symp. Computer Architecture*, 1978, pp. 144-151.

- [19] F. Distanté, F. Lombardi, and D. Sciuto, "Array Partitioning: A Methodology for Reconfigurability and Reconfiguration Problems," in *Proc. Int'l Conf. on Computer Design*, 1988, pp. 564-567.
- [20] D. Z. Du, D. F. Hsu, and F. K. Hwang, "Doubly Linked Ring Networks," *IEEE Trans. on Computers*, Vol. C-34, No. 9, Sept. 1985, pp. 853-855.
- [21] S. Dutt and J. P. Hayes, "Design and Reconfiguration Strategies of Near-Optimal k -Fault-Tolerant Tree Architecture," in *Proc. 18th Int'l Conf. on Fault-Tolerant Computers*, 1988, pp. 328-333.
- [22] S. Dutt and J. P. Hayes, "On Designing and Reconfiguring k -Fault-Tolerant Tree Architectures," *IEEE Trans. on Computers*, Vol. C-39, No. 4, Apr. 1990, pp. 490-503.
- [23] R. Evans, "A Self-Organizing Fault-Tolerant 2-Dimensional Array," *VLSI-85, Tokyo, Japan, North-Holland*, Aug. 1985, pp. 239-248.
- [24] G. E. Farr and H. Schröder, "An Evaluation of Reconfiguration Algorithms in Fault Tolerant Processor Arrays," in *Proc. 5th MIT Conf. on VLSI*, MIT Press, March 1988, pp. 131-148.
- [25] J. A. B. Fortes and C. S. Raghavendra, "Dynamically Reconfigurable Fault-Tolerant Array Processors," in *Proc. 14th Int'l Conf. on Fault-Tolerant Computers*, June 1984, pp. 386-392.
- [26] P. Franzon, "Interconnect Strategies for Fault Tolerant 2D VLSI Arrays," in *Proc. Int'l Conf. on Computer Design*, 1986, pp. 230-233.
- [27] P. Franzon, "Yield Modeling For Fault Tolerant VLSI Arrays," in *Proc. Int'l Conf. on Systolic Arrays*, 1986, pp. 207-216.
- [28] P. Franzon and K. Eshraghian, "Achieving ULSI Through Defect Tolerance," *Int'l Journal of Computer Aided VLSI Design*, Vol. 1, No. 1, 1989, pp. 73-90.

- [29] E. Fujiwara and D. K. Pradhan, "Error-Control Coding in Computers," in *IEEE Computers*, Vol. 23, No. 7, July 1990, pp. 63-72.
- [30] D. Fussell and P. Varmon, "Fault-Tolerant Wafer-Scale Architectures for VLSI," in *Proc. Symp. Computer Architecture*, 1982, pp. 190-198.
- [31] J. R. Goodman and C. H. Sequin, "Hypertree: A Multiprocessor Interconnection Topology," *IEEE Trans. on Computers*, Vol. C-30, No. 12, Dec. 1981, pp. 923-933.
- [32] J. Greene and A. E. Gamal, "Area and Delay Penalties in Restructurable Wafer-Scale Arrays," in *3rd Caltech Conf. on VLSI*, Jan. 1983, pp. 165-184.
- [33] J. W. Greene and A. E. Gamal, "Configuration of VLSI Arrays in the Presence of Defects," *Journal of the ACM*, Vol. 31, No. 4, Oct. 1984, pp. 694-717.
- [34] A. Grnarov, L. Kleinrock, and M. Gerla, "A Highly Reliable Distributed Double Loop Network Architecture," *Proc. Int'l Symp. Fault-Tolerant Computing*, Kyoto, Japan, Oct. 1980, pp. 319-324.
- [35] R. W. Haddad, A. T. Dahbura and A. B. Sharma, "Increased Throughput for the Testing and Repair of RAM's with Redundancy," *IEEE Trans. on Computers*, Vol. C-40, No. 2, Feb. 1991, pp. 154-166.
- [36] N. Hasan and C. L. Liu, "Minimum Fault Coverage in Reconfigurable Arrays," in *Proc. 18th Int'l Conf. on Fault-Tolerant Computers*, 1988, pp. 348-353.
- [37] A. S. M. Hassan and V. K. Agarwal, "A Modular Approach to Fault-Tolerant Binary Tree Architectures," in *Proc. 15th Int'l Conf. on Fault-Tolerant Computers*, 1985, pp. 344-349.
- [38] A. S. M. Hassan and V. K. Agarwal, "A Fault-Tolerant Modular Architecture for Binary Trees," *IEEE Trans. on Computers*, Vol. C-35, No. 4, April 1986, pp. 356-361.

- [39] J. P. Hayes, "A Graph Model for Fault-Tolerant Computing Systems," *IEEE Trans. on Computers*, Vol. C-25, No. 9, Sept. 1976, pp. 875-884.
- [40] K. S. Hedlund and L. Snyder, "Systolic Architectures - A Wafer Scale Approach," in *Proc. Int'l Conf. on Computer Design*, 1984, pp. 604-610.
- [41] S. H. Hosseini, J. G. Kuhl, and S. M. Reddy, "Distributed Fault-Tolerance of Tree Structures," *IEEE Trans. on Computers*, Vol. C-36, No. 11, Nov. 1987, pp. 1378-1382.
- [42] S. H. Hosseini, "On Fault-Tolerant Structure, Distributed Fault-Diagnosis, Reconfiguration, and Recovery of the Array Processors," in *Proc. Int'l Conf. on Computer Design*, 1988, pp. 554-559.
- [43] S. H. Hosseini, "On Fault-Tolerant Structure, Distributed Fault-Diagnosis, Reconfiguration, and Recovery of the Array Processors," *IEEE Trans. on Computers*, Vol. C-38, No. 7, July 1989, pp. 932-942.
- [44] M. C. Howells and V. K. Agarwal, "Yield and Reliability Enhancement of Large Area Binary Tree Architectures," in *Proc. 17th Int'l Conf. on Fault-Tolerant Computers*, 1987, pp. 290-295.
- [45] K. H. Huang and J. A. Abraham, "Algorithm-based Fault Tolerance for Matrix Operations," *IEEE Trans. on Computers*, Vol. C-33, No. 6, June 1984, pp. 518-528.
- [46] J. H. Hwang and C. S. Raghavendra, "VLSI Implementation of Fault-Tolerant Systolic Arrays," in *Proc. Int'l Conf. on Computer Design*, 1986, pp. 110-113.
- [47] A. Jain and J. Rajski, "Probabilistic Analysis of Yield and Area Utilization of Reconfigurable Processor Arrays," in *Int'l Workshop on Defect and Fault Tolerance in VLSI Arrays*, Univ. of Massachusetts, Oct. 1988, pp. 7.1.1-7.1.12.
- [48] J. H. Kim and S. M. Reddy, "A Fault-Tolerant Systolic Array Design Using TMR Method," in *Proc. Int'l Conf. on Computer Design*, 1985, pp. 769-773.

- [49] J. H. Kim, "On the Design of Easily Testable and Reconfigurable Systolic Arrays," in *Proc. Int'l Conf. on Systolic Arrays*, 1988, pp. 505-513.
- [50] J. H. Kim and S. M. Reddy, "On the Design of Fault-Tolerant Two-Dimensional Systolic Arrays for Yield Enhancement," *IEEE Trans. on Computers*, Vol. C-38, No. 4, April 1989, pp. 515-525.
- [51] D. E. Knuth, *The Art of Computer Programming*, Second Edition, Addison-Wesley Publishing Company, 1977.
- [52] I. Koren, "A Reconfigurable and Fault-Tolerant VLSI Multiprocessor Array," in *Proc. Symp. Computer Architecture*, 1981, pp. 425-442.
- [53] I. Koren and D. K. Pradhan, "Introducing Redundancy into VLSI Designs for Yield and Performance Enhancement," in *Proc. 15th Int'l Conf. on Fault-Tolerant Computers*, 1985, pp. 330-335.
- [54] I. Koren and D. K. Pradhan, "Yield and Performance Enhancement Through Redundancy in VLSI and WSI Multiprocessor Systems," *Proc. of the IEEE*, Vol. 74, No. 5, May 1986, pp. 699-711.
- [55] I. Koren and I. Pomeranz, "Distributed Structuring of Processor Arrays in the Presence of Faulty Processors," in *Proc. Int'l Conf. on Systolic Arrays*, 1986, pp. 239-248.
- [56] I. Koren and D. K. Pradhan, "Modeling the Effect of Redundancy on Yield and Performance of VLSI Systems," *IEEE Trans. on Computers*, Vol. C-36, No. 3, March 1987, pp. 344-355.
- [57] E. F. Kouka and J. L. Patry, "Restructuring WSI Arrays: An Empirical Approach," *Int'l Journal of Computer Aided VLSI Design*, Vol. 1, No. 1, 1989, pp. 91-112.

- [58] R. H. Kuhn, "Interstitial Fault Tolerance — A Technique for Masking Systolic Arrays Fault Tolerant," in *Proc. 16th Annual Hawaii Int'l Conf. on System Sciences*, 1983, pp. 215-224.
- [59] R. Kumar and F. G. Gray, "Fault Tolerant 1-Dimensional Cellular Structure," in *1984 Conf. on Distributed Computing*, May 1984, pp. 472-483.
- [60] S. Kuo and W. K. Fuchs, "Efficient Spare Allocation for Reconfigurable Arrays," *IEEE Design and Test*, Feb. 1987, pp. 24-31.
- [61] H. T. Kung, "Why Systolic Architecture?," *IEEE Computer*, Vol. 15, No. 1, Jan. 1982, pp. 37-46.
- [62] H. T. Kung and M. Lam, "Fault-Tolerant VLSI Systolic Arrays and Two-Level Pipelining," *Journal of Parallel and Distributed Processing*, Aug. 1984, pp. 32-63.
- [63] S. Y. Kung, D. D'Souza, and J. T. Juhl, "On Fault-Tolerance in Array Processors," in *Proc. Int'l Conf. on Computer Design*, 1985, pp. 764-768.
- [64] S. Y. Kung, C. W. Chang, and C. W. Jen, "Fault-Tolerance Design in Real-Time VLSI Array Processors," in *Proc. Conf. on Computers and Communications*, Phoenix, 1987, pp. 110-115.
- [65] S. Y. Kung, S. N. Jean, and C. W. Chang, "Fault-Tolerant Array Processors Using Single-Track Switches," *IEEE Trans. on Computers*, Vol. C-38, No. 4, April 1989, pp. 501-514.
- [66] C. L. Kwan and S. Toida, "Optimal Fault-Tolerant Realizations of Some Classes of Hierarchical Tree Systems," *Proc. 11th Int'l Conf. on Fault-Tolerant Computers*, 1981, pp. 176-178.
- [67] C. W. H. Lam, H. F. Li, and R. Jayakumar, "A Study of Two Approaches for Reconfiguring Fault-Tolerant Systolic Arrays," *IEEE Trans. on Computers*, Vol. C-36, No. 6, June 1989, pp. 833-844.

- [68] J.C. Laprie, J. Arlat, C. Béounes, and K. Kanoun, "Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures," in *IEEE Computers*, Vol. 23, No. 7, July 1990, pp. 39-51.
- [69] M. S. Lee and G. Frieder, "Massively Fault-Tolerant Cellular Array," in *Proc. Int'l Conf. on Parallel Processing*, Aug. 1986, pp. 343-350.
- [70] T. Leighton and C. E. Leiserson, "Wafer-Scale Integration of Systolic Arrays," *IEEE Trans. on Computers*, Vol. C-34, No. 5, May 1985, pp. 448-461.
- [71] H. F. Li, R. Jayakumar, and C. Lam, "Restructuring for Fault-Tolerant Systolic Arrays," *IEEE Trans. on Computers*, Vol. C-38, No. 2, Feb. 1989, pp. 307-311.
- [72] G. Liao, Q. Tong, and W. Lei, "Fault Tolerance Design of VLSI Two-Dimensional Systolic Processor Array," in *Proc. Conf. on Computers and Communications*, Phoenix, 1988, pp. 57-61.
- [73] M. T. Liu, "Distributed Loop Computer Networks," *Advances in Computers*, Vol. 17, Academic Press, New York, 1978, pp. 163-221.
- [74] F. Lombardi, R. Negrini, M. Sami, and R. Stefanelli, "Reconfiguration of VLSI Arrays: A Covering Approach," in *Proc. 17th Int'l Conf. on Fault-Tolerant Computers*, 1987, pp. 251-256.
- [75] F. Lombardi and W. K. Huang, "Approaches for Repair of VLSI/WSI RRAMs by Row/Column Deletion," in *Proc. 18th Int'l Conf. on Fault-Tolerant Computers*, 1988, pp. 342-347.
- [76] F. Lombardi, "Reconfiguration of Hexagonal Arrays by Diagonal Deletion," *The VLSI Journal*, Vol. 6, 1988, pp. 263-290.
- [77] M. B. Lowrie and W. K. Fuchs, "Reconfigurable Tree Architectures Using Sub-tree Oriented Fault Tolerance," *IEEE Trans. on Computers*, Vol. C-36, No. 10, Oct. 1987, pp. 1172-1182.

- [78] A. Majumdar, C. S. Raghavendra and M. A. Breuer, "Fault Tolerance in Linear Systolic Arrays Using Time Redundancy," *IEEE Trans. on Computers*, Vol. C-39, No. 2, Feb. 1990, pp. 269-276.
- [79] T. E. Mangir and A. Avizienis, "Effect of Interconnect Requirements on Yield Improvement of VLSI Design," *IEEE Trans. on Computers*, C-31, No. 7, July 1982, pp. 609-615.
- [80] T. E. Mangir and C. S. Raghavendra, "Issues in the Implementation of Fault-Tolerant VLSI and WSI Systems," in in *Proc. Int'l Conf. on Computer Design*, 1984, pp. 95-100.
- [81] T. E. Mangir, "Sources of failures and yield improvement for VLSI and re-structurable interconnects for RVLSI and WSI: Part I - Sources of failures and yield improvement for VLSI," *Proc. of the IEEE*, Vol. 72, No. 6, June 1984, pp. 690-709.
- [82] W. Marwood and A. P. Clarke, "Fault Tolerant Systolic Architectures," in *3rd National Workshop on Fault Tolerant Computing*, Melbourne, Australia, July 1985.
- [83] H. Masuyama and T. Ichimori, "Tolerance of Double-Loop Computer Networks to Multinode Failures," *IEEE Trans. on Computers*, Vol. C-38, No. 5, May 1989, pp. 738-741.
- [84] W. R. Moore and M. J. Day, "Yield-Enhancement of a Large Systolic Array Chip," *Microelectronic Reliability*, Vol. 24, No. 3, 1984. pp. 511-526.
- [85] W. R. Moore, "A Review of Fault-Tolerant Techniques for the Enhancement of Integrated Circuit Yield," *Proc. of the IEEE*, Vol. 74, No. 5, May 1986, pp. 684-698.

- [86] V. S. S. Nair and J. A. Abraham, "Hierarchical Design and Analysis of Fault-Tolerant Multiprocessor Systems using Concurrent Error Detection," in *20th Int'l Conf. on Fault-Tolerant Computers*, 1990, pp. 130-137.
- [87] A. Nayak, N. Santoro and R. Tan, "Fault-Intolerance of Reconfigurable Systolic Arrays," in *20th Int'l Conf. on Fault-Tolerant Computers*, 1990, pp. 202-209.
- [88] A. Nayak and N. Santoro, "Fault-Tolerance Analysis of some Regular Networks," in *Int'l Conf. on Parallel Processing Technology (PARCOM-90)*, Pune, India, Dec. 1990, pp. 67-76.
- [89] A. Nayak, M. Niktash and N. Santoro, "How to Avoid Catastrophic Failure and Guarantee Fault Tolerance," in preparation.
- [90] R. Negrini and R. Stefanelli, "Time Redundancy in WSI Arrays of Processing Elements," in *Proc. Int'l Conf. on Supercomputing Systems*, 1985, pp. 429-438.
- [91] R. Negrini, M. Sami, and R. Stefanelli, "Fault-Tolerance Approaches for VLSI/WSI Arrays," in *Proc. Conf. on Computers and Communications*, Phoenix, 1985, pp. 460-468.
- [92] R. Negrini and R. Stefanelli, "Algorithms for Self Reconfiguration of Wafer-Scale Regular Arrays," in *Proc. Int'l Conf. on Circuits and Systems*, 1985, pp. 190-196.
- [93] R. Negrini, M. G. Sami, and R. Stefanelli, "Fault-Tolerance Techniques for Array Structures used in Supercomputing," *IEEE Computer*, Vol. 19, No. 2, Feb. 1986, pp. 78-87.
- [94] S. Pateras and J. Rajski, "A Self-Reconfiguration Scheme for Fault-Tolerant VLSI Processor Arrays," in *Proc. Int'l Conf. on Computer Design*, 1988, pp. 560-563.

- [95] V. Piuri and R. Stefanelli, "Use of Redundant Binary Representation for Fault-Tolerant Array Processors," in *Proc. Int'l Conf. on Computer Design*, 1988, pp. 496-501.
- [96] S. P. Popli and M. A. Bayoumi, "A Reconfigurable VLSI Array for Reliability and Yield Enhancement," in *Proc. Int'l Conf. on Systolic Arrays*, 1988, pp. 631-642.
- [97] S. P. Popli and M. A. Bayoumi, "Fault Diagnosis and Reconfiguration for Reliable VLSI Arrays," in *Proc. Conf. on Computers and Communications*, Phoenix, 1988, pp. 69-73.
- [98] D. K. Pradhan, "Interconnection Topologies for Fault-tolerant Parallel and Distributed Architectures," in *Proc. Int'l Conf. on Parallel Processing*, Aug. 1981, pp. 238-242.
- [99] D. K. Pradhan and S. M. Reddy, "A Fault-Tolerant Communication Architecture for Distributed Systems," *IEEE Trans. on Computers*, Vol. C-31, No. 10, Oct. 1982, pp. 863-870.
- [100] D. K. Pradhan, Ed., *Fault-Tolerant Computing, Theory and Techniques*, Vol. 1 and 2, Englewood Cliff, NJ: Prentice-Hall, 1986.
- [101] C. S. Raghavendra and M. Gerla, "Optimal Loop Topologies for Distributed Systems," in *Proc. 7th Data Communication Symp.*, Mexico City, Oct. 1981, pp. 218-223.
- [102] C. S. Raghavendra, M. Gerla, and D. S. Parker, "Multi-Connected Loop Topologies for Local Computer Networks," in *Proc. INFOCOM 82*, Mar. 1982.
- [103] C. S. Raghavendra and T. E. Mangir, "On the VLSI Implementation of Fault Tolerant Architectures," in *Proc. Int'l Conf. on Computer Design*, 1983, pp. 744-747.

- [104] C. S. Raghavendra, A. Avizienis, and M. D. Ercgovac, "Fault Tolerance in Binary Tree Architectures," *IEEE Trans. on Computers*, Vol. C-33, No. 6, June 1984, pp. 568-572.
- [105] C. S. Raghavendra, "Fault Tolerance in Regular Network Architectures," *IEEE Micro*, Vol. 4, No. 6, Dec. 1984, pp. 44-53.
- [106] C. S. Raghavendra, M. Gerla, and A. Avizienis, "Reliable Loop Topologies for Large Local Computer Networks," *IEEE Trans. on Computers*, Vol. C-34, No. 1, Jan 1985, pp. 46-55.
- [107] C. S. Raghavendra and J. A. Silvester, "A Survey of Multi-Connected Loop Topologies for Local Computer Networks," *Computer Networks and ISDN Systems*, Vol. 11, No. 1, Jan. 1986, pp. 29-42.
- [108] C. V. Ramamoorthy and Y. W. Eva Ma, "Optimal Reconfiguration Strategies for Reconfigurable Computer Systems with no Repair," *IEEE Trans. on Computers*, Vol. C-35, No. 3, Mar. 1986, pp. 278-280.
- [109] S. F. Reddaway, "DAP-A Distributed Array Processor," in *Proc. Symp. Computer Architecture*, 1973, pp. 61-65.
- [110] D. A. Rennels, "Fault-Tolerant Computing — Concepts and Examples," *IEEE Trans. on Computers*, Vol. C-33, No. 12, Dec. 1984, pp. 1116-1129.
- [111] A. L. Rosenberg, "The Diogenes Approach to Testable Fault-Tolerant Arrays of Processors," *IEEE Trans. on Computers*, Vol. C-32, No. 10, Oct. 1983, pp.902-910.
- [112] A. L. Rosenberg, "On Designing Fault-Tolerant VLSI Processor Arrays," *Advances in Computer Research*, 1984, pp.181-204.
- [113] A. L. Rosenberg, "Graph Theoretical Approach to Fault-Tolerant Processor Arrays," in *Int'l Workshop on Wafer Scale Integration*, Southampton, 1985, pp.1-16.

- [114] D. J. Rosenkrantz and S. S. Ravi, "Improved Bounds on Algorithm-Based Fault Tolerance," in *Proc. 26th Allerton Conf. on Communication, Control, and Computing*, Sept. 1988, pp. 388-397.
- [115] V. P. Roychowdhury, J. Bruck and T. Kailath, "Efficient Algorithms for Reconfiguration in VLSI/WSI Arrays," *IEEE Trans. on Computers*, Vol. C-39, No. 4, Apr. 1990, pp. 480-489.
- [116] J. H. Saltzer, D. D. Clark, and K. T. Pograd, "Why a Ring?," in *Proc. 7th Data Communication Symp.*, Mexico City, Oct. 1981, pp. 211-217.
- [117] M. G. Sami and R. Stefanelli, "Reconfigurable Architectures for VLSI Processing Arrays," in *Proc. NCC 83*, Los Angeles, May 1983, pp. 567-577.
- [118] M. G. Sami and R. Stefanelli, "Self-Testing Array Structures," in *Proc. Int'l Conf. on Computer Design*, 1984, pp. 677-682.
- [119] M. G. Sami and R. Stefanelli, "Fault-Tolerance of VLSI Processing Arrays: The Time Redundancy Approach," in *Proc. Real-Time Systems Symp.*, Dec. 1984, pp. 200-207.
- [120] M. G. Sami and R. Stefanelli, "Fault-Stealing: An Approach to Fault-Tolerance of VLSI Array Structures," in *Proc. Int'l Conf. on Circuits and Systems*, 1985, pp. 205-210.
- [121] M. Sami and R. Stefanelli, "Reconfigurable Architectures for VLSI Processing Arrays," *Proc. of the IEEE*, Vol. 74, No. 5, May 1986, pp. 712-722.
- [122] G. Saucier, J. L. Patry and E. F. Kouka, "A Reconfigurable Wafer Scale Array for Image Processing," in *Proc. Int'l Conf. on Wafer Scale Integration*, 1989, pp. 277-288.
- [123] L. A. Shombert and D. P. Siewiorek, "Using Redundancy for Concurrent Testing and Repairing of Systolic Arrays," in *Proc. 17th Int'l Conf. on Fault-Tolerant Computers*, 1987, pp. 244-249.

- [124] M. J. Shute, *Fifth Generation Wafer Architecture*, Prentice Hall Int'l (UK) Ltd., 1988.
- [125] D. P. Siewiorek and R. S. Swartz, *The Theory and Practice of Reliable System Design*, Bedford, MA: Digital, 1982.
- [126] D. P. Siewiorek, "Fault Tolerance in Commercial Computers," in *IEEE Computers*, Vol. 23, No. 7, July 1990, pp. 26-37.
- [127] A. D. Singh, "A Reconfigurable Modular Fault Tolerant Binary Tree Architecture," in *Proc. 17th Int'l Conf. on Fault-Tolerant Computers*, 1987, pp. 298-304.
- [128] L. Snyder, "Introduction to Configurable, Highly Parallel Computer," *IEEE Computer*, Vol. 15, No. 1, Jan. 1982, pp. 47-56.
- [129] C. H. Stapper, F. M. Armstrong, and K. Saji, "Integrated Circuit Yield Statistics," *Proc. of the IEEE*, Vol. 71, No. 4, April 1983, pp. 453-470.
- [130] Thinking Machines Corporation, *Connection Machine Model CM-2 Technical Summary*, Thinking Machines Corporation, Technical Report Series HA87-4, 1987.
- [131] W. Toy and B. Zee, *Computer Hardware/Software Architecture*, Englewood Cliff, NJ: Prentice-Hall, 1986.
- [132] J. Tyszer, "A Multiple Fault-Tolerant Processor Network Architecture for Pipeline Computing," *IEEE Trans. on Computers*, Vol. C-37, No. 11, Nov. 1988, pp. 1414-1418.
- [133] M. Wang, M. Cutler, and S. Y. H. Su, "Reconfiguration of VLSI/WSI Array Processors with Two-Level Redundancy," *IEEE Trans. on Computers*, Vol. C-38, No. 4, April 1989, pp. 547-554.
- [134] L. D. Wittie, "Communication Structures for a Large Multimicrocomputer Systems," *IEEE Trans. on Computers*, Vol. C-33, No. 4, April 1984, pp. 264-273.

- [135] S. Yalamanchili and J. K. Aggarwal, "Reconfiguration Strategies for Parallel Architectures," *IEEE Computer*, Vol. 18, No. 12, Dec. 1985, pp. 44-61.
- [136] H. Y. Youn and A. D. Singh, "A Highly Efficient Design for Reconfiguring the Processor Array in VLSI," in *Proc. Int'l Conf. on Parallel Processing*, 1988, pp. 375-382.
- [137] H. Y. Youn and A. D. Singh, "On Implementing Large Binary Tree Architectures in VLSI and WSI," *IEEE Trans. on Computers*, Vol. C-38, No. 4, April 1989, pp. 526-537.
- [138] S. Yurttas and F. Lombardi, "New Approaches for the Reconfiguration of Two-Dimensional VLSI Arrays Using Time Redundancy," in *Proc. Real-Time Systems Symp.*, 1988, pp. 212-221.

END

0/2·03·92

FIN