**PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT**

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file / Votre référence

Our file / Notre référence

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.

Reconfiguration of Chordal Ring Networks

by

Quanhu Xue

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

School of Computer Science

**Carleton University
Ottawa, Ontario
April 12, 1994**

© Copyright
1994, Quanhu Xue

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

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.

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.

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 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 0-315-92935-9

Canada

Name Quonluc Kue

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Computer Science

SUBJECT TERM

0984

SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0317
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language 0679
General 0289
Ancient 0290
Linguistics 0291
Modern 0291
Literature 0401
General 0294
Classical 0295
Comparative 0297
Medieval 0298
Modern 0316
African 0591
American 0305
Asian 0352
Canadian (English) 0355
Canadian (French) 0593
English 0311
Germanic 0312
Latin American 0315
Middle Eastern 0313
Romance 0314
Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion 0318
General 0321
Biblical Studies 0319
Clergy 0320
History of 0322
Philosophy of 0469
Theology 0323

SOCIAL SCIENCES

American Studies 0323
Anthropology 0324
Archaeology 0326
Cultural 0327
Physical 0310
Business Administration 0272
General 0770
Accounting 0454
Banking 0338
Management 0385
Marketing 0501
Canadian Studies 0503
Economics 0505
General 0508
Agricultural 0509
Commerce Business 0510
Finance 0511
History 0358
Labor 0366
Theory 0351
Folklore 0578
Geography 0351
Gerontology 0578
History 0578
General 0578

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0373
United States 0337
History of Science 0585
Law 0398
Political Science 0615
General 0616
International Law and Relations 0617
Public Administration 0814
Recreation 0452
Social Work 0626
Sociology 0627
General 0938
Criminology and Penology 0631
Demography 0629
Ethnic and Racial Studies 0630
Individual and Family Studies 0700
Industrial and Labor Relations 0344
Public and Social Welfare 0709
Social Structure and Development 0999
Theory and Methods 0453
Transportation 0453
Urban and Regional Planning 0453
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture 0473
General 0285
Agronomy 0475
Animal Culture and Nutrition 0476
Animal Pathology 0359
Food Science and Technology 0478
Forestry and Wildlife 0479
Plant Culture 0480
Plant Pathology 0817
Plant Physiology 0777
Range Management 0746
Wood Technology 0306
Biology 0287
General 0308
Anatomy 0309
Biostatistics 0379
Botany 0329
Cell 0353
Ecology 0369
Entomology 0793
Genetics 0410
Limnology 0307
Microbiology 0317
Molecular 0416
Neuroscience 0433
Oceanography 0821
Physiology 0778
Radiation 0472
Veterinary Science 0786
Zoology 0760
Biophysics 0786
General 0760
Medical 0760

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0366
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences 0566
General 0300
Audiology 0992
Chemotherapy 0567
Dentistry 0350
Education 0769
Hospital Management 0758
Human Development 0982
Immunology 0564
Medicine and Surgery 0347
Mental Health 0569
Nursing 0570
Nutrition 0380
Obstetrics and Gynecology 0354
Occupational Health and Therapy 0381
Ophthalmology 0571
Pathology 0419
Pharmacology 0572
Pharmacy 0382
Physical Therapy 0573
Public Health 0574
Radiology 0575
Recreation 0575

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences 0485
Chemistry 0749
General 0486
Agricultural 0487
Analytical 0488
Biochemistry 0738
Inorganic 0490
Nuclear 0491
Organic 0494
Pharmaceutical 0495
Physical 0754
Polymer 0405
Radiation 0605
Mathematics 0986
Physics 0606
General 0608
Acoustics 0748
Astronomy and Astrophysics 0607
Atmospheric Science 0799
Atomic 0759
Electronics and Electricity 0609
Elementary Particles and High Energy 0610
Fluid and Plasma 0752
Molecular 0756
Nuclear 0611
Optics 0463
Radiation 0346
Solid State 0984
Statistics 0984

Applied Sciences

Applied Mechanics 0346
Computer Science 0984

Engineering 0537
General 0538
Aerospace 0539
Agricultural 0540
Automotive 0541
Biomedical 0542
Chemical 0543
Civil 0544
Electronics and Electrical 0348
Heat and Thermodynamics 0545
Hydraulic 0546
Industrial 0547
Marine 0794
Materials Science 0548
Mechanical 0743
Metallurgy 0551
Mining 0552
Nuclear 0549
Packaging 0765
Petroleum 0554
Sanitary and Municipal System Science 0790
Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

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

Reconfiguration of Chordal Ring Networks

submitted by

Quanhui Xue

in partial fulfilment of the requirements
for the degree of M.C.S

Chairman, School of Computer Science

Thesis Supervisor

Thesis Co-Supervisor

Carleton University

April 12, 1994

Abstract

Reconfiguration is a crucial aspect for network fault tolerance using link redundancy and has drawn a great deal of research. In the presence of node failures, the problem of how to reconfigure the non-faulty nodes into smaller network with the same topological structure is very important if the algorithms running on the network require that particular structure.

In this thesis, we consider redundant ring networks with link configuration $\{g_1, g_2, \dots, g_k\}$, where $g_1 = 1$. That is, rings where each node is also connected to the nodes at distances $g_2, g_3, \dots, g_k$, where $1 < g_2 < g_3 < \dots < g_k$. These additional links are used to bypass failed nodes. The goal of a reconfiguration scheme is to construct the largest subring among the non-faulty nodes. If the faults 'cut' the network into disconnected components, such a subring will be entirely within a component. The set of faulty nodes delimiting a component is called blocking.

We present several reconfiguration algorithms and analyze the size of the resulting subrings. We show that the size of the subring depends not only on the size of the component, but also on the 'shape' of the delimiting sets of faults.

In the literature, reconfiguration of redundant rings has been studied for the special case where all the bypass links at distances $2, 3, \dots, g_k$ exist. Our results apply to an arbitrary link configuration $\{g_1, g_2, \dots, g_k\}$.

Acknowledgements

I would like to give my sincere thanks to my thesis supervisor, Prof. Nicola Santoro, this work would not have been possible without the ingenious guidance and tireless support from him. His inspiration, enthusiasm, and encouragement has been a constant resource for me. His feedback over the years influenced greatly on me both as a researcher and as a writer.

Sincere gratitude should also go to my thesis co-supervisor, Dr. Amiyaranjan Nayak, whose friendship, wise advice and support knew no bounds, even extending over vacations! His valuable suggestions and great encouragement throughout the development of this thesis, make the development of this thesis a pleasure.

I am grateful to all my thesis committee members, Prof. Danny Krizanc, Prof. Jorge Urrutia for their suggestions for the improvement of this thesis.

Most of all, the understanding, encouragement, and support from my parents and my family have been a great driving force behind me. Especially, I thank my wife, Jinrui Zhang, not only for the unending hours that should have been spent with her, but also for her help in more ways than I can list here throughout this adventure. It is to them that I dedicate it, as they are the ones who paid the highest price for its completion.

Contents

Abstract	iii
Acknowledgements	iv
List Of Figures	viii
1 Introduction	1
1.1 The Framework	1
1.1.1 Fault Tolerance	1
1.1.2 Fault Tolerant Ring Networks	2
1.2 The Problem and The Results	3
1.2.1 The Problem	3
1.2.2 The Results	4
1.3 Outline of the Thesis	5
2 Background	7
2.1 Introduction	7
2.2 Background on Fault Tolerance Techniques	8
2.2.1 Fault Tolerance Terminologies	8
2.2.2 Techniques for Fault Tolerance	9
2.3 Background on Ring Networks	10
2.4 Catastrophic Fault Patterns in Linear Arrays	13

2.5	Known Results on Ring Reconfiguration	17
2.6	Chapter Summary	19
3	Ring Representation and Manipulation	20
3.1	Introduction	20
3.2	Preliminaries	20
3.3	Representation and Operations of Ring	26
3.3.1	Matrix Representation	27
3.3.2	Operations on the Matrix	29
3.3.3	Minimal Blocking Patterns	31
3.4	Problem Reformulation	35
3.5	Chapter Summary	40
4	Reconfiguration Algorithms For $H > 2$	42
4.1	Introduction	42
4.2	Strategy and Terminology for Ring Formation	43
4.3	A Simple Line Segment Formation Algorithm	46
4.4	Another Line Segment Formation Algorithm	57
4.5	A General Line Segments Formation Algorithm	61
4.6	Ring Formation	68
4.6.1	Connecting Line Segments to a Line in S_i	68
4.6.2	Connecting the Three Parts to Form A Ring	71
4.7	Chapter Summary	79

5	Reconfiguration Algorithms For $H \leq 2$	81
5.1	Introduction	81
5.2	Necessary Conditions	82
5.3	Subrings with wrap-around link	89
5.4	Subrings without using wrap-around link	98
5.5	Chapter Summary	106
6	Summary and Conclusions	107
6.1	Summary of Main Contributions	107
6.2	Open Problems	108
6.3	Areas of Future Work	109
	Glossary	110
	References	113

List of Figures

2.1	Examples of unidirectional and bidirectional ring networks	11
2.2	A daisy loop network	13
2.3	A 12-node optimal forward loop backward hop network	14
2.4	A linear array of processors	14
3.1	A chordal ring	21
3.2	Two equivalent operational configurations	22
3.3	A blocking fault pattern for a chordal ring ($\{1,3\}, 12$)	25
3.4	A ring C has a toroidal shape	26
3.5	A matrix representation of a ring network based on node 0	28
3.6	A 'LSR by 3' operation	29
3.7	An example of non-blocking minimal fault pattern with $C'DP$	32
3.8	a) L is a line segment; b) L is not a line segment	34
3.9	An example of the out-layer of a line segment	35
3.10	Three blocking fault patterns decompose a ring into 3 regions	36
3.11	A ring with a continuous blocking fault pattern	37
3.12	Decomposing a matrix into distinct regions	37
3.13	Decomposing a ring with one fault pattern	38
3.14	Normalization of a region	39
4.1	Divide S into S_u , S_m , and S_l	44
4.2	Illustration of transferring S_u	45

4.3	The matrix representation of S_i	47
4.4	Illustration of the result of Algorithm FB	50
4.5	Illustration of the result of Algorithm FBS	55
4.6	Illustration of the SB algorithm	58
4.7	The missed nodes of the SB algorithm	60
4.8	Illustration of the CB operation	63
4.9	Illustration of Algorithm FBB	64
4.10	Illustration of the result of Algorithm $BLRS$	68
4.11	Illustration of the result of Algorithm LFORM	70
4.12	Illustration of case 1 and case 2	77
4.13	Illustration of case 3 and case 4	77
4.14	Illustration of case 5 and case 6	78
4.15	Illustration of case 7 and case 8	78
5.1	Illustration of the row distances	82
5.2	Illustration of a cut of type 1	84
5.3	Illustration of a cut of type 2	84
5.4	Illustration of a cut of type 3	85
5.5	Illustration of division of S	88
5.6	Two through paths	89
5.7	A subring generated by BW	96
5.8	The two possible patterns of $LS(S)$ for case (2)	99
5.9	The two possible patterns of $LS(S)$ for case (3)	99
5.10	Illustration for the case when $x_i - x'_{i+2} = 2$	100
5.11	A subring generated by $FBBT$	105

Chapter 1

Introduction

1.1 The Framework

1.1.1 Fault Tolerance

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 regular architectures, such as linear arrays, rings, and tree structures, is through the incorporation of redundancy. 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.

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; moreover, 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 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.

However, 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 systems is the *reconfiguration capability* (or *reconfiguration effectiveness*); that is, the ability to map faulty elements to spares by using bypass links while preserving the original interconnection structure of the system.

It is therefore not surprising that a large amount of research has been devoted to the design of reconfiguration algorithms for redundant structures as well as proposing redundant architectures which facilitate the reconfiguration process [4, 9, 14, 23, 32, 37, 49]. Related work has focused on the study of low yield problem, i.e. on improving spare utilization.

1.1.2 Fault Tolerant Ring Networks

The ring network is one of the most popular topologies for local computer networks due to its simple structure and easy implementation. In a ring network, each node is connected to its neighbor to form a complete loop. Reliability is a severe problem in single-loop ring networks: any single node failure can disrupt communication, and if the algorithms running on the network requires a ring topology, the entire network will become useless for the computation.

It is clearly necessary to provide fault tolerance against node and link failures in large ring networks. A common practice to provide such fault tolerance in this kind of networks is by introducing redundancy. The idea here is to design multiconnected loops in the original rings so that several alternative paths for communication exists

between nodes. In other words, each node is connected to two or more nodes in the network. With alternate paths between nodes, the network can tolerate several node and link failures. Several ring networks, suggested in [2, 8, 11, 27, 41], are based on this principle.

1.2 The Problem and The Results

1.2.1 The Problem

The effectiveness of using redundancy to increase fault tolerance in a regular architecture clearly depends on the distribution of the faults in the system. In fact, faults occurring at strategic locations may have catastrophic effect on the entire structure and cannot be overcome by any clever design.

A large amount of research [1, 29, 30, 31, 34, 35, 7, 44] has been done to characterize catastrophic fault patterns for linear arrays. These studies have shed some light on the inherent limits to reconfigurability as well as on the factors limiting the effectiveness of the redundancy approach.

In the case of redundant ring networks, it is possible that faults occurring in strategic locations in the network logically disconnect the ring structure [29]. However, disconnection of a ring does not necessarily mean that the entire structure is unusable. In fact, it still might be possible to find, among the non-faulty elements, one or more subrings (or linear arrays) on which to perform smaller tasks.

The problem of constructing linear arrays and subrings in a ring network in the presence of faulty nodes has been studied in [32, 33, 48]. In [48], the author gives a bound on the size of the linear array which can be constructed when the number of faults is less than the degree of the network. In [32, 33], the authors extensively studied the problem of constructing linear arrays or subrings in the presence of faults in ring networks with bypass links $\{2, \dots, g_k\}$ and established the bounds on the degree of the fault tolerance that could be achieved; the results of [48] follow as corollaries of the ones they established.

The main objective of this thesis is to develop reconfiguration algorithms in general bidirectional ring networks; that is, rings with bypass links $\{g_2, g_3, \dots, g_k\}$. The focus is on constructing subrings in the presence of sets of "blocking" faults, i.e., sets of fault patterns which are catastrophic for linear arrays.

1.2.2 The Results

The subring construction problem in the presence of one or more sets of blocking fault patterns are studied by mapping a given ring network into a Boolean matrix, in which the ring is divided into regions. The ring networks under consideration are bidirectional ring networks with link configuration $\{1, g_2, g_3, \dots, g_k\}$, where $g_2, g_3, \dots, g_k$ are pre-designed parameters of the network. By a region, we mean a connected component delimited by two blocking fault patterns. Then we study the subring construction problem for a region, since any subring will be entirely within one region.

In our investigation, we concentrate on the special link configuration $\{1, g_k\}$. Since this configuration is contained in any other link configuration, it provides lower bound for the general case.

Depending on the distance H between the two delimiting blocking fault patterns, we study the problem in two cases: $H > 2$ and $H \leq 2$.

When $H > 2$, we present several algorithms to suit various situations of the region based on the "line segment" approach. In this approach, we first connect the non-faulty nodes near the two blocking patterns and form line segments, then we connect all the line segments together to form a subring.

These algorithms are not applicable when $H \leq 2$. A different but similar approach is presented, what we call the "through path" approach. In this case, since it is possible that some columns are not usable for any subring, the region may need to be divided again. Thus, we develop two subring construction algorithms for two different cases: using or without using wrap-around links. The number of nodes in the region they will miss is $\frac{3}{2}g_k + 2$ and $2g_k + 3$ in the worst case, respectively, for the two cases.

For both cases, the algorithms developed are given in a semi-distributed fashion. That is, we describe the algorithms in terms of the states and behavior of individual non-faulty nodes.

The main contribution of this thesis is an extensive study of the subring construction problem for ring networks with arbitrary link configuration i.e. the presence of sets of blocking fault patterns. Two sets of subring construction algorithms for two different cases of S , that is, $H > 2$ and $H \leq 2$, are developed. They are highly efficient in the sense that they capture most of the non-faulty nodes. They are also given in a semi-distributed fashion so that it is easy to implement them in a distributed environment. These results extend the knowledge on the general reconfiguration problem for redundant ring networks; a knowledge which, until now, was restricted to very special case [32, 33, 48].

1.3 Outline of the Thesis

The rest of this thesis is organized as follows.

The basic fault tolerance terminologies and techniques are reviewed in Chapter 2. Chapter 2 also contains some background knowledge on ring networks, a discussion on the fault tolerance principles employed for making ring networks fault-tolerant, and the summary of the characterization of catastrophic fault patterns in linear arrays known in the literature. Some existing results on ring networks, relating to the size of the largest subring possible in the presence of a given set of faulty nodes, are also given in Chapter 2.

Formal definitions of redundant ring networks and blocking fault patterns are given in Chapter 3. A Boolean matrix representation for ring networks and the corresponding operations for manipulating a Boolean matrix are also described there. Furthermore, it is also shown how to reformulate the problem of constructing a subring in terms of finding a connected closed path in the corresponding Boolean matrix of the given ring.

In Chapter 4, a set of algorithms for forming a subring in a given region S

with at least three rows of non-faulty elements are developed. A general strategy for forming the largest subring in S is described. Following the general strategy, a given S is decomposed into three parts (upper part, middle part, and lower part). Line segments in both upper and lower parts are then formed and connected with the help of the middle part to get a ring. The number of missed nodes for all these algorithms is no more than $g_k + 1$.

In Chapter 5, the reconfiguration problem is studied for regions S when S can have fewer than three rows of non-faulty elements. By examining the necessary conditions for a column to be included in a subring, we derive a scheme to divide the region into sub-regions such that any subring will be entirely within one of the sub-regions. For each sub-region, we develop subring formation algorithms for two cases: sub-region with wrap-around link and sub-region without wrap-around link. The number of missed nodes for the algorithm using wrap-around link and the one without using wrap-around links are $\frac{3}{2}g_k + 2$ and $2g_k + 3$ in the worst case, respectively.

The major results are summarized in Chapter 6. Some open problems are also listed, and some conclusions that may serve as guidelines for future research are included.

Chapter 2

Background

2.1 Introduction

Ring networks have been widely used to interconnect computers in message passing and packet switching environment. Reliability is a serious problem for single-loop ring networks; thus, fault tolerance of ring networks has received considerable attention in recent years. In this chapter, we will provide some background information on ring networks and fault tolerance of ring networks.

Notice that, on one hand, any portion of a ring topology is a linear systolic array; on the other hand, if we connect the two ends of any linear systolic array, we will get a ring topology. That means some results established on arrays can be applied to rings and vice versa. Therefore, it is necessary to examine related aspects of linear arrays in this chapter.

This chapter is organized as follows. An overview of fault tolerance terminologies and techniques is given in Section 2.2. In Section 2.3, we give a background on ring networks and the fault tolerance principles employed to make ring networks fault-tolerant. Some known results about catastrophic fault patterns in linear arrays are summarized in Section 2.4. In Section 2.5, we review some known results on ring reconfiguration. Finally, Section 2.6 contains a summary of the Chapter.

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. *Fault masking* is the process that prevents the propagation of a system's faults into the data of the system. Finally, *fault tolerance* is the ability of the system to continue to perform its specified tasks in the presence of faults.

Fault tolerance can be broadly achieved by 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 for Fault Tolerance

The key ingredient in practically 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. 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.

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 [47]. 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.

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" [3]. 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 the 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.

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 being isolated, the redundant resource is switched on (reconfigured), and the system is recovered (i.e., restarted).

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. The use of various fault tolerance techniques in commercial computer systems are well documented in [45].

2.3 Background on Ring Networks

A ring network (also called a loop network) is a regular network without central node. In a ring network, host computers are connected via loop interface hardware; each node is connected to its neighbor(s) to form a complete loop, and a message is passed from node to node along links. Links can be either unidirectional or bidirectional, and the ring networks are called unidirectional or bidirectional rings.

Figure 2.1: Examples of unidirectional and bidirectional ring networks

accordingly. Some examples of unidirectional and bidirectional ring networks are shown in Figure 2.1

Ring networks possess some unique advantages and disadvantages. For example, a bidirectional single loop (ring) network has only two I/O ports per node for any network size which means it has very good expandability; on the other hand, the diameter (maximum number of hops between any two nodes along the shortest path) is $N/2$ in a system with N nodes and any two non-adjacent faulty nodes or links disconnect the loop, which gives the network very poor reliability. In contrast, a completely connected loop network has $(N - 1)$ I/O ports per node, but the distance between any two nodes is unity, which means that the structure is highly fault tolerant, but due to its high cost the structure is unsuitable for systems with large number of nodes.

Reliability is a severe problem with single-loop networks, particularly when the network size increases. Any single node or link failure can disrupt communication. A common approach to provide higher reliability in loop networks is by introducing redundant links. 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 al-

ternative paths between nodes, the network can sustain several node or link failures.

A variety of specific ring architectures have been explored [6] that use redundant network topology and suitable reconfiguration techniques to recover from link or node failures. Reconfiguration has predominantly been viewed as a corrective mechanism that is invoked in direct response to a previously diagnosed fault. Typically, normal network operation relies on a conventional method of medium access control (such as token passing), originally designed for single-loop networks; breaks in the ring are detected and located by specialized diagnostic frames or signals, and then a suitable reconfiguration is performed if necessary.

Liu [24] has proposed a distributed double-loop computer network (DDLCN) 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 the DDLCN 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 [11] (see Figure 2.2). 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 DDLCN [8, 11, 27, 40].

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*

Figure 2.2: A daisy loop network

[11, 12, 13]: 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. 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 DDLN and the daisy-chain loop (See Figure 2.3).

2.4 Catastrophic Fault Patterns in Linear Arrays

Since the characteristics of catastrophic fault patterns in a linear array is crucial for the reconfiguration problem in ring networks, we will briefly summarize some known results about catastrophic fault patterns in linear arrays in this section.

Let $A = \{p_1, p_2, \dots, p_N\}$ denote a linear array of N processors, where each $p \in A$ represents a processing element and there exists a direct link between p_i and p_{i+1} , $1 \leq i < N$. An example of such a linear array is shown in Figure 2.4.

To describe the concept of information flow from one side to the other side, we introduce two special processors, called I (for input) and O (for output), at the

Figure 2.3: A 12-node optimal forward loop backward hop network

Figure 2.4: A linear array of processors

two ends of the network. They are responsible for the I/O functions of the system. It is clear that one faulty element in any location is sufficient to stop the flow of information from one side to the other side of the linear array. A common approach for achieving fault tolerance is through the incorporation of redundancy, both in terms of redundant processing elements and redundant links.

Definition 2.1 Given an integer $g \in [1, N-1]$ and an array A of size N , A is said to have link redundancy g , if for every $p_i \in A$ with $i \leq N-g$ there exists a link between p_i and p_{i+g} ; if $g > 1$, such a link will be called a bypass link.

The following is a general definition of link redundancy for a set of bypass links:

Definition 2.2 The array A has link redundancy $G = \{g_1, g_2, \dots, g_k\}$ where $g_j < g_{j+1}$ and $g_j \in [1, N-1]$, if A has link redundancies $g_1, g_2, \dots, g_k$.

It has been assumed that no other links exist in the array except the ones specified by G . That is, G totally defines the *link structure* of A . Notice that $g_1 = 1$ is the regular link, while all other g_i 's correspond to bypass links.

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 $m \leq N$, $f_i < f_{i+1}$ and $f_i \in [1, N]$. An assignment of a fault pattern F to A means that for every $f \in F$, p_f is faulty.

Definition 2.3 A fault pattern F is catastrophic for an array A with link redundancy G if I and O are not connected in the presence of such an assignment of faults.

In other words, F is catastrophic if the removal of the faulty elements and their incident links will cause the I/O 's to become disconnected.

It has been shown that a catastrophic fault pattern for a link redundancy $G = \{g_1, g_2, \dots, g_k\}$ must have at least g_k number of faults [29]. A catastrophic fault pattern with exactly g_k faulty PEs is called a *minimal fault pattern*.

The authors in [29] have introduced the Boolean matrix representation of fault patterns and is reviewed below.

Consider an arbitrary fault pattern $F = \{f_1, f_2, \dots, f_{j_k}\}$, consisting of g_k faults for an arbitrary link configuration $G = \{g_1, g_2, \dots, g_k\}$. Without loss of generality, assume that $f_1 = 1$. The links can be either unidirectional or bidirectional. We represent F by a Boolean matrix M of size $(w_F \times g_k)$, where $w_F = f_{j_k} - f_1 + 1$, and $w_F = [w_F/g_k]$, defined as follows:

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

In the matrix representation, each $f_i \in F$ is mapped into $M[x_i, y_i]$ where $x_i = [f_i/g_k]$ and $y_i = f_i \bmod g_k$. Notice that $M[1, 1] = 1$ which indicates the location of the first fault.

Notice that any minimal catastrophic fault pattern satisfies the necessary condition that $\forall j$, there is only one i for which $M[i, j] = 1$.

Let M be the matrix representation of a minimal fault pattern F . The *row coordinates* of F is the ordered set $\{x_0, x_1, \dots, x_{g_k-1}\}$ of the row indices of M corresponding to the faults f_i ($0 \leq i \leq g_k - 1$). The *interior*, *exterior*, and *border* elements in the matrix representation of a fault pattern are defined as follows:

Definition 2.4 Let $M[x_i, y_i]$ be the location of fault f_i . The location $M[i, y]$, with respect to f_i , is interior if $i < x_i$, border if $i = x_i$, and exterior if $i > x_i$.

The definition of interior, border, and exterior has been extended from element to regions as follows.

Definition 2.5 For a given fault pattern F , $I(F)$ (i.e., interior of F) is the set of all interior elements, $B(F)$ (i.e., border of F) is the set of all border elements, and $E(F)$ (i.e., exterior of F) is the set of all exterior elements.

Now with respect to the matrix representation of F , the definition for a catastrophic fault pattern can be rephrased in terms of interior and exterior regions as follows:

Theorem 2.1 A fault pattern F is catastrophic for an array A with link redundancy G if it is not possible to reach any exterior element from any interior element using the links in G .

According to this definition, a fault in a specific position in the array will force the existence of other faults for the fault pattern to be catastrophic. To characterize this fact, the concept of *effect* of a faulty node in a fault pattern has been introduced and accordingly, a necessary and sufficient condition for catastrophic fault patterns has been developed in [30]. Here we state them as follows:

Definition 2.6 Let $f \in F$ and let $W[i, j]$ be the corresponding entry in the Boolean representation of F . The effect of f is the set $\varphi(f) = \{f + g_k \pm g : g \in G\}$.

Theorem 2.2 [30] F is catastrophic for G if and only if $\varphi(f) \subseteq E(F) \cup B(F)$ for every $f \in F$.

2.5 Known Results on Ring Reconfiguration

In this section, we introduce some known results on ring reconfiguration. In literature, the reconfiguration problem on rings has been studied on *complete chordal rings*, that is, rings in which the special link configuration $G = \{1, 2, \dots, g_k\}$. The first result is due to J. Tyszer [48], which is stated in the following lemma:

Lemma 2.1 [48] *If $|F| < 2g_k - 2$, it is possible to create a linear array $A = (G, N - |F|)$.*

This result shows that, if the total number of faults in C is less than $2g_k - 2$, there is a way to connect all the non-faulty nodes to form a linear array: that is, it is possible to construct a linear array of size $N - |F|$. This fact is not surprising since, in case of $|F| < 2g_k - 2$, the number of blocking fault patterns is at most 1; thus, by using the regular links to connect the consecutive non-faulty nodes and the bypass links to pass the cluster of faults of size less than g_k , clearly we can get all the non-faulty nodes to form a linear array.

A new approach to the problem has been considered in [32]. In this approach, the ring was decomposed into successive clusters of faults and non-faulty nodes, say $\{F_1, S_1, F_2, S_2, \dots, F_p, S_p\}$, 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 . The following result has been proved with the following notations: $l_i = |F_i|$, $s_i = |S_i|$, for all $j < g_k$, $\Phi_j = |\{F_i : l_i = g_j\}|$ and $\Phi_{max} = |\{F_i : l_i \geq g_k\}|$, that is, Φ_i is the number of clusters of faults of length g_i .

Theorem 2.3 [32] *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 $|F|$.*

This result illustrates a general property which indicates that if there is no blocking fault pattern formed in F , then by using the links of G , it is able to reconstruct a subring with all the non-faulty nodes. An important point made by this result

is that the possibility of reconstructing a subring or linear array depends on the distribution of the faults, but not the total number of faults.

In the presence of blocking fault patterns (i.e., $\Phi_{max} > 0$), the ring will be cut into Φ_{max} connected components. In this case, consider the component containing the largest number of non-faulty elements, which is at least $(N - |F|)/\Phi_{max}$; however, it is possible to construct a linear array within the connected component by connecting all the non-faulty nodes in the component. This fact is illustrated in the following theorem:

Theorem 2.4 [32] *For a ring $C = (G, N)$, there exists a linear array $A = (G, \alpha)$ regardless of the number of faults $|F|$ where*

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

To construct a subring in the presence of blocking fault patterns, we need some more restrictions on the distribution of the faults. In [30], the concept of *permissible* subsequence $\chi_{a,b} = S_a, F_{a+1}, S_{a+1}, \dots, F_b, S_b$ in C was defined as follows:

Definition 2.7 $\chi_{a,b}$ 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. $\pi(\chi_{a,b}) > 2$.

where $\pi(\chi_{a,b}) = \sum_{i=a}^b |S_i|$.

Let Π be the largest permissible sequence; i.e., $\pi(\Pi) \geq \pi(\chi_{a,b})$. The authors developed the following strategy to build a subring in each permissible sequence $\chi_{a,b}$, where, for simplicity, $\pi(\Pi)$ is denoted by π .

- 1 : Starting from the leftmost element of S_a , label all the non-faulty nodes in $\chi_{a,b}$ alternatively R and L.

- 2 : Connect each R (or L) node to the nearest R (or L) node to the right according to the following rules: the R (or L) nodes within the same S_i are linked using links of length 2; the last R (or L) node of S_i is linked to the first R (or L) node of S_{i+1} with a link of length $|F_{i+1}| + 2$.
- 3 : Finally, form the ring by connecting the first R with the first L in S_k and the last R with the last L in S_k .

Notice that the above theorem requires at least 2 non-faulty nodes in each non-faulty cluster. In case single non-faulty nodes are enclosed between faulty ones (i.e., there is a i such that $|S_i| = 1$), it is still possible to build a subring under certain conditions.

In [33], the authors also studied the limits to reconfigurability of complete chordal rings in general. They established some bounds on the degree of the fault tolerance that could be achieved in a redundant ring network when performing a computation that requires a fixed number of the size of operational nodes.

2.6 Chapter Summary

The fault tolerance terminologies and techniques were reviewed in this chapter. Background knowledge on ring networks and fault tolerance principles employed in ring networks to make them fault-tolerant were also discussed. The characterization of catastrophic fault patterns known in the literature was summarized. Some existing results on ring networks, relating to the size of the largest subring possible in the presence of a given set of faulty nodes, were also given in this chapter.

Chapter 3

Ring Representation and Manipulation

3.1 Introduction

A ring network can be represented in form of a Boolean matrix, and any ring operation can be shown to have a corresponding matrix operation. Since it is fairly easy to work with a Boolean matrix, we study the problem of constructing a subring in the presence of faulty nodes in a given redundant ring network purely in matrix terms. That is, we first represent the ring as a Boolean matrix and then work on the matrix in building subrings for the original ring.

This chapter is organized as follows. In Section 3.2, we give a formal definition of a redundant ring network and define the blocking fault patterns in ring networks. In Section 3.3, we describe in detail a Boolean matrix representation of ring networks and give the corresponding operations for manipulating a Boolean matrix. The problem of subring construction is then rephrased in Section 3.4 in terms of finding connected closed paths in the Boolean matrix. Finally, Section 3.5 contains a summary of the chapter.

3.2 Preliminaries

Most of the terminology, given in Chapter 2 in the context of catastrophic fault patterns for linear arrays, still apply to ring networks; however, some of the definitions

Figure 3.1: A chordal ring

needs to be modified. In this section, we introduce the additional notations and terminology necessary for the reconfiguration of redundant ring networks.

Redundant ring networks with bidirectional links are considered in this thesis. The basic components of such a network are the *processing elements* (PEs) and the links. The links can be either *regular* or *bypass*. The distinction between regular links (edges of the ring) and bypass links (the chords) is their functionality: the bypass links are strictly for reconfiguration purposes when faults are detected; in the absence of faults, the bypass links are redundant and only regular links are used (see Figure 3.1). In the following, we will use the term PE and node interchangeably. These redundant ring networks are regular and highly symmetrical. They are called *chordal rings* or *circulant graphs* in the literature.

Practically, each node in the ring network has a unique physical identifier (or address) and an operational status (*faulty* or *non-faulty*). At any given time, the set of faulty nodes in the network defines an operational configuration of the network, and a different set of faulty nodes can define a different operational configuration. However, in this thesis, we are interested in the topological aspect of the network and do not distinguish nodes according to their functionalities; thus, two or more operational configurations will be considered to be the same if they result in the same topological structure.

Figure 3.2: Two equivalent operational configurations

Example 3.1: Consider a ring network consisting of 6 nodes, namely A, B, C, D, E, L, as shown in Figure 3.2. The operational configurations defined by the two sets of faulty nodes $\{A, D\}$ and $\{B, E\}$ are two different physical configurations, but they will be considered to be equivalent because of identical topological structure.

We assign logical identifiers to the nodes, usually consecutive integers increasing in one of the orientations (either clockwise or counter-clockwise) starting from a given node. We call the operational configuration of the network with these logical identifiers the “logical operational configuration”. For the same network, if by changing starting nodes two logical operational configurations become the same, then we say that they are *equivalent*. Equivalent configurations will be treated as the same in future discussions.

Observe that the choice of an orientation and the location of the starting node do not effect the structure of the network since we are dealing with rings with bidirectional links.

A formal definition of generalized chordal rings now follows.

Definition 3.1 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 $C = (\mathcal{P}, \mathcal{E})$ 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, we denote node P_i by i if there is no ambiguity. 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 d of the ring.

Two well known examples of generalized chordal rings are the simple ring $R(N) = (\{1\}, N)$ and the fully connected graph $K(N) = (\{1, 2, 3, \dots, \lfloor N/2 \rfloor\}, N)$.

Definition 3.2 A clockwise (resp. counter-clockwise) base ring of $C = (G, N)$, denoted by C^+ (resp. C^-), is the directed graph obtained by orienting all arcs in $(\{1\}, N)$ in a clockwise (resp. counter-clockwise) direction.

The definition of a fault pattern for rings can now be extended and modified as follows:

Definition 3.3 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 $m \leq N$ and $f_j \in [0, N)$. The set $F \subseteq \mathcal{P}$ represents the set of faulty PEs in C .

Notice that unlike the case for linear arrays, we did not specify any order of f_j s in F . In fact, due to the wrapped around characteristic of the ring topology, the concept of "order" might still be unclear. In the following, we define the concept of order for rings:

Definition 3.4 Given any three nodes in C , say $x, y, z \in [0, N)$, we say that $x <_+ y <_+ z$, if in C^+ , the path from x to z contains y . Similarly, $x <_- y <_- z$ will denote that y is in the path from x to z in C^- .

Definition 3.5 A fault pattern $F = \{f_1, f_2, \dots, f_m\}$ is said to be in a clockwise ordered form if $f_{j-1} <_+ f_j <_+ f_{j+1}$ for $1 < j < m$.

Analogously, we can define the counter-clockwise ordered form.

Definition 3.6 Let $F = \{f_1, f_2, \dots, f_m\}$ be in ordered form (either clockwise or counter-clockwise). The cyclic shift of F is $\lambda^{(1)}(F) = \{f_2, f_3, \dots, f_m, f_1\}$. Furthermore, let $\lambda^{(0)}(F) = F$, for any integer $1 \leq k \leq m$, the k-th cyclic shift of F is $\lambda^{(k)} = \lambda^{(1)}(\lambda^{(k-1)})$.

Obviously, if F is in clockwise (resp. counter-clockwise) ordered form, then $\lambda^{(k)}$, $1 \leq k \leq m$, will also be in clockwise (resp. counter-clockwise) ordered form. We will denote the set of cyclic shifts of F by $\Lambda(F) = \{\lambda^{(i)}(F) : 0 \leq i < m\}$.

Definition 3.7 $F = \{f_1, f_2, \dots, f_m\}$ is said to be in cardinal form, if $f_j < f_{j+1}$ for $1 \leq j < m$.

Clearly, if the logical numbers of the nodes are assigned according to C^+ (resp. C^-), then fault patterns in cardinal form are obviously in clockwise (resp. counter-clockwise) ordered form, but the converse is not necessarily true.

In the following, without loss of generality, we will assume that C^+ is the orientation for C , and all the fault patterns are given in clockwise ordered form, or shortly, ordered form.

Definition 3.8 The window W_F of a fault pattern $F = \{f_1, f_2, \dots, f_m\}$ (in ordered form) is the set of PEs between and including the first and the last fault in F . That is, the set $\{f_1, f_1 + 1, \dots, f_m - 1, f_m\}$, where all operations are modulo N .

Definition 3.9 The width w_F of a fault pattern $F = \{f_1, f_2, \dots, f_m\}$ is the size of W_F . The height h_F of F is $\lceil w_F/g_k \rceil - 1$.

Definition 3.10 Given a fault pattern $F = \{f_1, f_2, \dots, f_m\}$ in C , we define the cardinalization of F to be $\mathcal{C}(F) = F + N - f_1 = \{0, f_2 + N - f_1, \dots, f_m + N - f_1\}$, where all operations are modulo N .

Notice that the cardinalization of F is a translation of F by $N - f_1$ such that the first fault is 0 in the renumbered ring. Clearly, $\mathcal{C}(F)$ is in cardinal form if F is in ordered form.

Denote by $\mathcal{N}(F) = \{\mathcal{C}(\lambda(F))\}$ the set of all the cardinalized cyclic shifts of F .

Figure 3.4: A ring C has a toroidal shape

Definition 3.14 The blocking dimension of F , denoted by $\delta(F)$, is the size of the largest blocking decomposition of F .

Recall that any F is blocking for a ring $C = (G, N)$ if and only if it is a catastrophic fault pattern for linear array $A = (G, N')$ where $N' \geq N$. Therefore, the results established on catastrophic fault patterns for linear arrays can be applied to the blocking fault patterns for rings.

3.3 Representation and Operations of Ring

In this section, we will represent the ring in form of a Boolean matrix and establish various operations on this matrix relating to the renumbering of the nodes of the ring.

Let us assume for the moment that $N = Lg_k$ for some L (we will show in next section that, this restriction can be removed easily). In this case, it is interesting to observe that with $G = \{1, g_k\}$, the ring C has a toroidal shape of size $L \times g_k$. For example, the ring $C = (\{1, 4\}, 16)$ has the toroidal shape of size 4×4 shown in Figure 3.4.

3.3.1 Matrix Representation

A matrix representation for fault patterns in linear arrays has been described in [29] for studying various properties of catastrophic fault patterns. In this representation, a fault pattern $F = \{f_1, f_2, \dots, f_m\}$ is mapped into a Boolean matrix M of size $(\lceil w_F/g_k \rceil \times g_k)$. It is also possible to represent a ring as a Boolean matrix in a similar way. This process is described in this section.

Due to the wrap-around property of ring, we have to specify a starting node before we can give the matrix representation of the ring; we will denote this starting node as the *base node* of the matrix representation.

Definition 3.15 For a given ring $C = (G, N)$, a base node b and a fault pattern F , we define the matrix form of C based on b , denoted by $M_b(C)$, to be the Boolean matrix of size $(L \times g_k)$ given by the following relation:

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

for $1 \leq i \leq L, 1 \leq j \leq g_k$.

In this representation, the node b is mapped into $M_b(C)[1, 1]$, node $b+1$ is mapped into $M_b(C)[1, 2]$, and so on. For simplicity, when no ambiguity arises, in the following, we will use (i, j) to represent $M_b(C)[i, j]$ as well as the corresponding node in C .

Example 3.2: Consider the ring $C = (\{1, 3\}, 12)$ with $F = \{3, 5, 7\}$ as shown in Figure 3.3. The matrix form $M_0(C)$ of C based on node 0 is shown in Figure 3.5.

Given a node P in C and a matrix form M of C , let $P = (i, j)$ in M ; then we will use the following notation through out this thesis: $P(x) = i$ and $P(y) = j$.

Consider the matrix form M of $C = (G, N)$ for a link configuration G . Notice that $\{1, g_k\} \subseteq G$; we have the following observations regarding the links in M :

Figure 3.5: A matrix representation of a ring network based on node 0

- 1): there is a link between (i, j) and $(i + 1, j)$ for any i and j such that $1 \leq j \leq g_k$, where the addition is modulo L ;
- 2): there is a link between (i, j) and $(i, j + 1)$ for any i, j such that $1 \leq i \leq L, 1 \leq j < g_k$; and
- 3): there is a link between (i, g_k) and $(i + 1, 1)$ for any i , where the addition is modulo L . We call this kind of links the wrap-around link.

Definition 3.16 A live path (or shortly path) π between two non-faulty locations (u, v) and (w, z) in matrix M is a set of locations $\pi = \{(x_0, y_0), (x_1, y_1), (x_2, y_2), \dots, (x_p, y_p)\}$, where $(x_0, y_0) = (u, v), (x_p, y_p) = (w, z)$ such that:

- 1) (x_i, y_i) is non-faulty, $0 \leq i < p$;
- 2) there is a link between (x_i, y_i) and (x_{i+1}, y_{i+1}) , $0 \leq i < p$; and
- 3) $(x_i, y_i) \neq (x_j, y_j)$ if $i \neq j$.

Example 3.3. $\pi_1 = \{(1, 3), (1, 2), (1, 1), (4, 1)\}$ and $\pi_2 = \{(1, 3), (1, 2), (4, 2), (4, 1)\}$ are two paths between $(1, 3)$ and $(4, 1)$ in Figure 3.5. It is clear that there may be more than one path between two locations.

In most cases, we will need to represent ring C based on the first faulty node f_1 of $F = \{f_1, f_2, \dots, f_m\}$; thus, for convenience, we define the following:

Definition 3.17 For a given fault pattern $F = \{f_1, f_2, \dots, f_m\}$ in C , we define the matrix form of C based on F , denoted by $M_F(C)$, to be $M_{f_1}(C)$, the matrix of C based on f_1 .

Figure 3.6: A LSR by 3 operation

3.3.2 Operations on the Matrix

In this section, we define some operations to manipulate these Boolean matrices.

Definition 3.18 Given two $L \times g_k$ Boolean matrices M and M' , we say M' is a left-shift-rotate by 1 of M , denoted by $M' = LSR(M, 1)$, if the following relation holds:

$$M'[i, j] = \begin{cases} M[i, j-1] & \text{if } j < g_k \\ M[1, 1] & \text{if } i = L, j = g_k \\ M[i+1, 1] & \text{otherwise} \end{cases}$$

Definition 3.19 Given two $L \times g_k$ Boolean matrices M and M' let t be a integer such that $0 \leq t < N$, we say M' is a left-shift-rotate by t of M , denoted by $M' = LSR(M, t)$, if $M' = LSR(LSR(M, t-1), 1)$.

Example 3.4: In Figure 3.6, we show a LSR by 3 operation.

Similarly, we can define right-shift-rotate operation as follows:

Definition 3.20 Given two $L \times g_k$ Boolean matrices M and M' , we say M' is a right-shift-rotate by 1 of M , denoted by $M' = RSR(M, 1)$, if the following relation holds:

$$M'[i, j] = \begin{cases} M[i, j+1] & \text{if } j > 1 \\ M[L, g_k] & \text{if } i = 1, j = 1 \\ M[i-1, g_k] & \text{otherwise} \end{cases}$$

Definition 3.21 Given two $L \times g_k$ Boolean matrices M and M' , let t be a integer such that $0 \leq t < N$, we say M' is a left-shift-rotate by t of M , denoted by $M' = RSR(M, t)$, if $M' = RSR(RSR(M, t-1), 1)$.

We can easily verify the following properties associated with LSR and RSR operations:

Property 3.1 For any given matrix M , and any s and t such that $0 \leq s, t < N$:

- 1) $LSR(M, 0) = M; RSR(M, 0) = M$
- 2) $LSR(RSR(M, t), t) = M; RSR(LSR(M, t), t) = M$
- 3) $LSR(M, t) = RSR(M, N - t)$
- 4) $LSR(LSR(M, t), s) = LSR(M, (s + t))$
- 5) $RSR(RSR(M, t), s) = RSR(M, (s + t))$

where the $+$ and $-$ are modulo N .

Proof:

Follows from Definition 3.19 and Definition 3.21. □

Definition 3.22 For any two Boolean matrices M and M' of size $L \times g_k$, we say that M and M' are equivalent, denoted by $M \equiv M'$, if $M' = LSR(M, t)$ (or $M' = RSR(M, t)$) for some t , $0 \leq t < N$.

Theorem 3.1 The relation $\equiv$ is an equivalence relation; that is, it is reflexive, symmetric, transitive.

Proof:

- 1) Reflexivity: For any M , we have $M \equiv M$

It is clear that $M = LSR(M, 0)$, thus $M \equiv M$.

2) Symmetry: if $M \equiv M'$, then $M' \equiv M$.

Since $M \equiv M'$, according to definition, there must be a t such that $M' = LSR(M, t)$, then

$$M = RSR(LSR(M, t), t) = RSR(M', t) = LSR(M', N - t)$$

thus $M' \equiv M$.

3) Transitivity: If $M \equiv M'$ and $M' \equiv M''$, then $M \equiv M''$

Since $M \equiv M'$ and $M' \equiv M''$, there exist a t and a s , such that $M' = LSR(M, t)$ and $M'' = LSR(M', s)$, then

$$M'' = LSR(LSR(M, t), s) = LSR(M, (s + t))$$

that is, $M'' \equiv M$.

□

Notice that if we represent a given ring C based on two different base nodes, we will get two different matrices; however, they both represent the same ring.

Theorem 3.2 For a given ring C and any two base nodes p and q in the ring, $M_p(C) \equiv M_q(C)$.

Proof:

Without loss of generality, let us assume $p \geq q$. Then it is easy to see that $M_p(C) = LSR(M_q(C), p - q)$. Thus, $M_p(C) \equiv M_q(C)$. □

3.3.3 Minimal Blocking Patterns

Consider the matrix form of C based on a given minimal blocking fault pattern $F = \{f_1, f_2, \dots, f_{jk}\}$. Let us denote the column coordinates of F by the ordered set $\{y_1, y_2, \dots, y_{jk}\}$. We have the following necessary condition according to [30]:

Figure 3.7: An example of non-blocking minimal fault pattern with *C'DP*

Property 3.2 [30] For any minimal blocking fault pattern $F = \{f_1, f_2, \dots, f_{g_k}\}$, the following property holds: there is exactly one fault in each column: that is, $y_i \neq y_j$ for any pair of i and j such that $1 \leq i, j \leq g_k$ and $i \neq j$.

We will call this property the Column Distribution Property (*C'DP*).

Theorem 3.3 For any given minimal blocking fault pattern F in C , let M be the matrix form of C based on F and x_i be the row coordinates of the faulty node of F in column i , where $1 \leq i \leq g_k$. Then $x_j - 1 \leq x_i \leq x_j + 1$ if $|i - j| = 1$.

Proof:

By Property 3.2, there is one fault of F in each column of M . Furthermore, by Theorem 2.2, if $|i - j| = 1$, then x_j has only three possible values: x_i , $x_i - 1$, and $x_i + 1$; thus, the result follows. $\square$

Notice that it is not necessary for every minimal fault pattern which satisfies *C'DP* to be a blocking fault pattern for C . An example of such a minimal fault pattern F which satisfies *C'DP* but is not a blocking fault pattern is shown in Figure 3.7.

For any given minimal fault pattern $F = \{f_1, f_2, \dots, f_m\}$ which satisfies *C'DP*, let x_i be the row coordinates of the faulty node of F in column i , where $1 \leq i \leq g_k$. Now, we define the interior, exterior and border elements in $M_F(C)$ for the window W_F of F .

Definition 3.23 Let $a = (i, j) \in W_F$. a is said to be an interior element if $i < x_j$, a border element if $i = x_j$, and an exterior element if $i > x_j$, respectively.

The definition of interior, border, and exterior can be extended from element to regions as follows:

Definition 3.24 For a given minimal fault pattern F satisfying CDP in C , the interior $I(F)$, border $B(F)$, and exterior $E(F)$ of F , with respect to the matrix form of C based on F , are the set of all interior, border, and exterior elements, respectively.

Theorem 3.4 Given a fault pattern $F = \{f_1, f_2, \dots, f_{g_k}\}$ in C and the matrix form of C based on F , F is blocking if and only if the following two conditions are satisfied:

- 1): F satisfies CDP;
- 2): for any pair of interior (u, v) and exterior (s, t) , there is no live path π between (u, v) and (s, t) such that $\pi \subseteq W_F$.

Proof:

The proof follows from the definition of blocking and the definition of catastrophic pattern for linear arrays in [30]. $\square$

Corollary 3.1 Given a minimal blocking fault pattern $F = \{f_1, f_2, \dots, f_{g_k}\}$ and the matrix form M based on F of C , $(1, g_k) \in F$.

Definition 3.25 Assume that $F = \{f_1, f_2, \dots, f_{g_k}\}$ is a minimal blocking fault pattern of C and M is a matrix of C such that $f_1 = (r, 1)$ for some $r > 1$. For any two non-faulty nodes $p = (s, 1)$ and $q = (s, g_k)$ with $s < r$, a line segment L between p and q is a live path between p and q , which satisfies the following conditions:

- 1): L does not contain any wrap-around links;
- 2): for any node e in L , it will be below row s ; that is, if $e \in L$, then $e(x) \geq s$.

Figure 3.8: a) L is a line segment; b) L is not a line segment

We can see that a line segment is a special kind of live path between the two nodes: the two end nodes are on the same row s , one on the first column, the other on the last column; furthermore, all the other nodes in the path are on the rows below row s .

Example 3.5: In Figure 3.8, we show an example of a path which is a line segment in a) and an example of a path which is not a line segment in b).

Lemma 3.1 *Given a line segment L between p and q , let $\Omega_j = \{e \in L | e(y) = j\}$ be the set of all the nodes in L which are in column j for $1 \leq j \leq g_k$.*

$$(1) : L = \cup_{j=1}^{g_k} \Omega_j;$$

$$(2) : \text{for all } j, \Omega_j \neq \emptyset.$$

Proof:

(1): Notice that $\Omega_1, \Omega_2, \dots, \Omega_{g_k}$ is a decomposition of L according to the column coordinate, thus, the result follows.

(2): Since p and q are in the first and the last column, respectively, any path that does not contain wrap-around links must span all the columns, thus, the result follows. $\square$

Definition 3.26 *Given a line segment l , the out-layer $OL(l)$ of l is the set $\{\epsilon_1, \epsilon_2, \dots, \epsilon_{g_k}\}$, where $\epsilon_j = \min_i \{(i, j) \in \Omega_j\}$.*

Figure 3.9: An example of the out-layer of a line segment

Example 3.6: Figure 3.9 shows the out-layer of a line segment L , which is the same line segment as shown in Figure 3.8 a). We use shadowed circles to indicate the nodes in the out-layer.

Furthermore, we can define the following relationship for two line segments:

Definition 3.27 Two line segments L_1 and L_2 are said to be non-crossing if $L_1 \cap L_2 = \emptyset$.

3.4 Problem Reformulation

In this thesis, we focus on the reconfiguration problem of the chordal rings given a minimal blocking decomposition $\{F_1, F_2, \dots, F_p\}$ of a blocking fault pattern F . That is, our objective is to construct the largest subring using the non-faulty nodes in the presence of such a fault pattern.

Notice that in the presence of such a minimal decomposition, the ring is partitioned into p connected components of non-fault nodes and the largest subring will be entirely within one of these components.

In this section, we will reformulate the problem in the context of matrix representation of rings.

Let $\{F_1, F_2, \dots, F_p\}$ be a blocking decomposition of F , where each $F_i = \{f_{i1}, f_{i2}, \dots, f_{ig_k}\}$ is a minimal blocking fault pattern. Furthermore, without loss of generality, let us assume that $f_{i1} = 0$ (that is, the first fault of F_i is the first node of C ; otherwise, we can always renumber the ring to be in such situation) and $f_{i1} < f_{(i+1)1}$ for $1 \leq i < p$. Let M be the Boolean matrix of ring C based on F_1 .

Figure 3.10: Three blocking fault patterns decompose a ring into 3 regions

Due to the fact that each F_i is a minimal blocking pattern by itself, each F_i satisfies the CDP. Let us denote the row coordinate of fault in F_i on column j , $1 \leq j \leq g_k$, by x_{ij} and also assume that $x_{ij} < x_{(i+1)j}$ for $1 \leq i < p, 1 \leq j \leq g_k$, which implies that $F_i < F_{i+1}$. The ordering $(F_i < F_{i+1})$ ensures that F_i will appear above F_{i+1} in the corresponding matrix form M based on F_1 .

These p minimal blocking fault patterns partition C into p connected regions as shown in Figure 3.10 and therefore, each connected region is uniquely specified by two minimal blocking fault patterns. We denote the region specified by F_i and F_{i+1} by S_i for $i < p - 1$; the region specified by F_p and F_1 is S_p .

Regardless of the choice of the base node, one region (e.g., S_3 in Figure 3.10) will be split into two parts in the matrix representation, except for the special case when the first block pattern is a cluster of faults as shown in Figure 3.11. However, since each region is totally independent of the others in terms of connectivity, we can decompose the matrix into p independent regions and treat each region individually.

Example 3.7: Consider a minimal decomposition $\{F_1, F_2\}$, the matrix form of C has two connected regions: one is S_1 and the other is S_2 . We can decompose the matrix as shown in Figure 3.12.

Figure 3.11: A ring with a continuous blocking fault pattern

Figure 3.12: Decomposing a matrix into distinct regions

Figure 3.13: Decomposing a ring with one fault pattern

For the case $p = 1$, there is only one blocking fault pattern, say F_1 . In this case, there exists only one region which we denote by S_1 . We still can decompose it as shown in Figure 3.13.

Recall that we had assumed $N = Lg_k$ for some L when we represented the ring in form of a Boolean matrix. In fact, as we see in the decomposition process of M , each region is independent of the others in terms of reconfiguration. Thus, when one particular region is considered, we can represent the concerned region in a matrix form in a similar fashion, which means we can always represent a ring in a matrix form even if N is not divisible by g_k .

Consider a region S specified by the $F_l = \{f_{l1}, f_{l2}, \dots, f_{lg_k}\}$ and $F_u = \{f_{u1}, f_{u2}, \dots, f_{ug_k}\}$, where $F_l < F_u$. If f_{l2} is not in the first column of the matrix, we perform the so-called “normalization” of S with respect to F_l . That is, we perform a series of shift-rotate operations which ensures that f_{l1} is placed in the first column. During the process of “normalization”, it is quite likely that the original shape of F_u in the matrix representation is distorted (i.e., f_{u1} is no longer in the column where is used to be before the normalization).

Clearly, we can also perform the normalization operation with respect to F_u ; the two choices are symmetric in terms of the development of reconfiguration algorithms

Figure 3.14: Normalization of a region

in S . In the rest of the thesis, we will assume that S under discussion is always normalized with respect to F_l .

Example 3.8: In Figure 3.14, we show an example of the normalization operation. In this example, f_l was located in column 2 before normalization and in column 1 afterwards.

In building a subring in S , we are particularly interested in the size $|C'|$ of C' we built. Clearly, there is an upper bound for $|C'|$, that is, $|C'| \leq |S|$. If we denote the subring generated by a reconfiguration algorithm $\mathcal{A}$ by $C'(\mathcal{A})$, then for any two given subring generating algorithms $\mathcal{A}$ and $\mathcal{B}$, we will say that $\mathcal{A}$ is better than $\mathcal{B}$ if $|C'(\mathcal{A})| > |C'(\mathcal{B})|$.

For any given subring generating algorithm $\mathcal{A}$, we define a node $P \in S$ to be a missed node of $\mathcal{A}$ as follows:

Definition 3.28 For any node $P \in S$, P is said to be a missed node of a subring generating algorithm $\mathcal{A}$ if $P \notin C'(\mathcal{A})$.

Let $m(\mathcal{A})$ be the set of missed nodes for a given algorithm $\mathcal{A}$. Clearly, we have an alternative criteria for comparing two given subring generating algorithms since $|C'(\mathcal{A})| = |S| - |m(\mathcal{A})|$ for any $\mathcal{A}$:

Definition 3.29 For any two given subring generating algorithms $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A}$ is better than $\mathcal{B}$ if and only if $|m(\mathcal{A})| < |m(\mathcal{B})|$.

Consider two link configurations G_1 and G_2 for the same number of nodes N and assume that $G_2 \subseteq G_1$. Then, clearly, any minimal blocking fault pattern F of $C_1 = (G_1, N)$ is a minimal blocking fault pattern of $C_2 = (G_2, N)$. Suppose that there is a region S in C_1 , then S would also be a region in C_2 . In terms of the subring generating algorithms in S , we have the following observation.

Property 3.3 Let $G_2 \subseteq G_1$. $\mathcal{A}$ be a subring generation algorithm for S in $C_2 = (G_2, N)$, then $\mathcal{A}$ is also a subring generation algorithm for S in $C_1 = (G_1, N)$. In particular, a subring generation algorithm for S in ring $(\{1, g_k\}, N)$ is also one for S in ring $(\{1, g_2, \dots, g_k\}, N)$.

In the rest of the thesis, we will focus on the development of subring generation algorithms for the case of $G = \{1, g_k\}$, which can be viewed as generating a lower bound on the size of subring for the general case.

Regarding the relative positions of the two fault patterns F_l and F_u in S , we use the term height of S defined below as a measurement.

Definition 3.30 The height of S , denoted by H_S or simply H , to be the number of rows of non-faulty nodes. To be precise, $H = \lfloor (f_{l1} - f_{ug_k})/g_k \rfloor$.

In Chapter 4, we will deal with the case when $H > 2$, that is, there are at least three rows of non-faulty nodes separating the two fault patterns F_l and F_u . In Chapter 5, we will deal with the case when $H \leq 2$.

3.5 Chapter Summary

A formal definition of a redundant ring network and the blocking fault patterns in ring networks were given in this chapter. A Boolean matrix representation for ring networks and the corresponding operations for manipulating a Boolean matrix were

also described. It was also shown how to look at the problem of constructing a subring in a given ring in terms of finding a connected closed path in the corresponding Boolean matrix of the given ring.

Chapter 4

Reconfiguration Algorithms For $H > 2$

4.1 Introduction

In this chapter, we will develop a family of subring construction algorithms in a given region S when $H > 2$; recall that S is delimited by two minimal blocking fault patterns $F_l = \{f_{l1}, f_{l2}, \dots, f_{lg_k}\}$ and $F_u = \{f_{u1}, f_{u2}, \dots, f_{ug_k}\}$ and $H = \lfloor (f_{u1} - f_{lg_k}) / g_k \rfloor$. That is, there are at least three rows of non-faulty nodes separating the two fault patterns F_l and F_u .

To form a subring in the given region S , we use the notation of *line segment* introduced in Section 3.3. The general strategy, which will be described in details in Section 4.2, consists of first decomposing the region S into three parts (the upper part, the middle part, and the lower part), then forming line segments in the upper and lower parts respectively, and finally connecting those line segments using the middle part to form a ring.

The organization of this chapter is as follows: a basic technique for constructing line segments is described in Section 4.3; in Section 4.4, a different method for forming line segments is introduced which also can be used as a line extension technique. Both techniques are integrated in a new algorithm described in Section 4.5. The line segments are then joined to form a unique line using the method introduced in Section 4.6.1. In Section 4.6.2, we show how to join the lines to form a ring. Finally, Section 4.7 contains a summary of the the chapter.

The following notations will be used throughout this chapter: $f_1 = (r, 1)$, $f_{r-1} = (r + H_{F_1}, s)$, $f_{r,k} = (r', s')$, where $r' = r + H + 1$. Clearly, due to the nature of the region S , the following relation holds: $r' + H + H_{F_1} + 1, s \neq 1$.

4.2 Strategy and Terminology for Ring Formation

In this section, we will provide the overall view of the general strategy used in this chapter.

All the algorithms developed in this chapter uses the notion of line segment introduced in Section 3.3. The idea is that we try to appropriately divide S into three parts, namely the upper part, the lower part and the middle part. If the middle part is large enough, we can construct a set of line segments in both the upper part and the lower part respectively, and use the middle part to connect them to form the ring. This scheme, called *LSRI*, consists of the following 4 steps:

- Step 1:** Decompose S into three parts S_u , S_l , and S_m appropriately
- Step 2:** Form a set of non-crossing line segments in S_l and in S_u , respectively
- Step 3:** Connect the line segments to form a single line in S_l and in S_u , respectively
- Step 4:** Connect the two lines formed in Step 3 using S_m to form a ring

A detailed explanation of these four steps is given below:

- Step 1:** Decompose S into three parts S_u , S_l , and S_m appropriately

Intuitively, S consists of three parts: $S_{F_1} = S \cap W_{F_1}$, $S_{F_2} = S \cap W_{F_2}$ and $M_S = S - S_{F_1} - S_{F_2}$; that is, $M_S = \{(i, j) | (r', s') < (i, j) < (r, 1)\}$ with respect to the lexicographical ordering. (Recall that $r = f_{11}(x)$, $s = f_{12k}(y)$, $r' = f_{r,k}(x)$ and $s' = f_{r,k}(y)$). Since $H \geq 2$, $|M_S| \geq 3g_k$. For any decomposition of $H = h_u + h_m + h_l$, where h 's are positive integers, we can further divide M_S into three subsets as follows: $M_u = \{(i, j) | (i, j) \leq (r' + h_u, s')\}$, $M_l = \{(i, j) | (r - h_l, 1) \leq (i, j) \leq (r - 1, g_k)\}$ and $S_m = M_S - M_u - M_l$; accordingly, we can divide S into three parts, namely: the

Figure 4.1: Divide S into S_u , S_m , and S_l

upper part $S_u = S_{F_u} \cup M_u$, the lower part $S_l = S_{F_l} \cup M_l$ and the middle part S_m . A decomposition of S is illustrated in Figure 4.1 with $H = 6$, $h_u = 2$, $h_m = 1$, $h_l = 3$.

Which decomposition of S will be chosen depends on the values of H , H_{F_l} and H_{F_u} , and will be discussed with each algorithm later in this chapter.

Step 2: Form a set of non-crossing line segments in S_l and S_u , respectively

Recall that S is normalized with respect to F_l ; thus, in S_l , F_l has the following characteristics: 1) the faults in the first column and in the last column are always in the same row, say row r ; and 2) all the faults of F_l are located in or below row r . Based on this property, an algorithm to form a set of non-crossing line segments is developed.

Due to the existence of S_m , we can view S_l and S_u as two separated entities in terms of line segments formation. Furthermore, we can always translate S_u by the *shift rotate* operations so that F_u has the following properties: 1) the faults of F_u in the first column and in the last column are in the same row, say row r' ; and 2) all the faults of F_u are in or above row r' (see Figure 4.2). In other words, the

Figure 4.2: Illustration of transferring S_u

scheme developed for S_l apply symmetrically to S_u by using the counter-clockwise orientation. Thus, without loss of generality, we will consider S_l in the rest of the discussion.

Step 3: Connect the line segments to form a single line in S_l and in S_u , respectively

A unique line L is formed by connecting all the line segments formed in S_l in step 2 such that the two end nodes of L have links to nodes in S_m . A unique line is formed in S_u analogously.

Step 4: Use S_m to connect the two lines formed in Step 3 to form a ring

Since the two lines formed in Step 3 have links to nodes in S_m , we can finally use the nodes in S_m to connect the two lines to form the ring.

The following terminology will be used throughout this chapter.

Recall that S_l consists of S_{F_l} and M_l ; also notice that S_{F_l} spans over $H_{F_l} + 1$ rows in matrix M , namely, rows $r, r+1, r+2, \dots, r+H_{F_l}$, where all the nodes of L_S in row $r+H_{F_l}$ are faulty ones. Let R_i denotes the set of nodes in $I(F_l)$ on row i with $r \leq i \leq r+H_{F_l}-1$; then R_i can be decomposed into n_i (for some n_i) clusters $N_{i,j}$ of consecutive non-faulty nodes, where all nodes in $N_{i,j}$ are on the right of the nodes in $N_{i(j-1)}$.

Definition 4.1 The odd-set $odd(F_l)$ (resp. even-set $even(F_l)$) of F_l is the set of all $N_{i,j}$ such that $|N_{i,j}|$ is odd (resp. even).

Clearly, $S_{F_l} = odd(F_l) \cup even(F_l)$.

Definition 4.2 For row i , $r \leq i \leq r + H_{F_l} - 1$, the valley-width v_i of R_i is the largest size of all N_{ij} , that is,

$$v_i = \max_j \{|N_{ij}|\}.$$

Theorem 4.1 If $i < j$, then $v_i \geq v_j + 2$ for all $r \leq i, j \leq r + H_{F_l} - 1$.

Proof:

Let $N_{jt} = \{(j, t), (j, t+1), \dots, (j, t+v_j-1)\}$ be the largest clusters of consecutive non-faulty nodes in row j ; according to the definition of N_{jt} , both $(j, t-1)$ and $(j, t+v_j)$ are in F_l . Consider the facts that $i < j$ and F_l has the CDP; then both $(i, t-1)$ and $(i, t+v_j)$ are in $I(F_l)$. Furthermore, since $N_{jt} \subset I(F_l)$, $\{(i, t), (i, t+1), \dots, (i, t+v_j-1)\} \subseteq I(F_l)$. Thus, $\{(i, t-1), (i, t), \dots, (i, t+v_j)\} \subseteq R_i \subseteq I(F_l)$; that is, $v_i \geq v_j + 2$. $\square$

Definition 4.3 The valley-width of F_l , $v(F_l)$ or simply v , is $v_{r+H_{F_l}-1}$.

In Sections 4.3, 4.4, and 4.5, we are going to develop a set of line segment formation algorithms. For simplicity, let us denote the fault pattern in S_l by $F = \{f_1, f_2, \dots, f_{g_k}\}$ in the following three sections.

4.3 A Simple Line Segment Formation Algorithm

Assume for the moment that there is an unbounded number of non-faulty rows above row r in S_l (see Figure 4.3). This assumption is only made to simplify the presentation and will be lifted later.

Definition 4.4 For any given node $P = (i, j)$, its neighbor below $B(P)$, neighbor above $A(P)$, right neighbor $R(P)$ and left neighbor $L(P)$ are nodes $(i+1, j)$, $(i-1, j)$, $(i, j+1)$, $(i, j-1)$, respectively.

Definition 4.5 For any given set of nodes D , its neighbors below $B(D)$, neighbors above $A(D)$, right neighbors $R(D)$ and left neighbors $L(D)$ are the set of nodes $\{B(P)|P \in D\}$, $\{A(P)|P \in D\}$, $\{R(P)|P \in D\}$, $\{L(P)|P \in D\}$, respectively.

Figure 4.3: The matrix representation of S_l

Notice that in S_l , if a non-faulty node is surrounded by 3 faulty neighbors, it will be impossible for any ring generation algorithm to include it in the resulting ring.

Definition 4.6 For a given fault pattern F , a non-faulty node $P \in A(F)$ is a singleton of F if $L(P), B(P), R(P) \in F$.

We will denote the set of all the singletons of F by $SGT(F)$. Clearly, $SGT(F) \subseteq odd(F)$.

It is clear that no singleton can appear in the middle of a line segment since its degree is only one once we remove all its links to the faulty neighbors. Accordingly, we define the following terms:

Definition 4.7 A node is said to be connectable if it is neither a faulty node, nor a singleton. It is said to be a non-connectable node, otherwise.

Now we present an algorithm which takes a minimal blocking fault pattern F as input and produces as output a line segment between node $(r-1, 1)$ and node $(r-1, g_k)$. We will denote this algorithm by FB (for "Follow the Border") in the following discussion.

Algorithm FB:

INPUT: A minimal blocking fault pattern F

OUTPUT: A line segment L

Begin

$L \leftarrow \emptyset$

```

 $r \leftarrow f_1(r)$ 
current-node  $P \leftarrow (r - 1, 1)$ 
While  $P(y) \neq g_k$  do
  add  $P$  to  $L$ 
  if  $B(P)$  is connectable then
     $P \leftarrow B(P)$ 
  elseif  $R(P)$  is connectable then
     $P \leftarrow R(P)$ 
  elseif  $A(P)$  is connectable then
     $P \leftarrow A(P)$ 
  Endif
Endwhile
add  $P$  to  $L$ 
output  $L$ 
End

```

The execution of this algorithm is best described in terms of a control token being passed along the nodes of the line segment. The holder of the token performs the following tasks:

1. add itself to the current result L ;
2. check if it is the last node (in column g_k) of this process. If it is, then terminate the process and consume the token; otherwise, try to pass the token to the first connectable neighbor according to the following priority: the neighbor below, the right neighbor, the neighbor above.

Initially, the token is held by node $(r - 1, 1)$.

Definition 4.8 For a node P , the connections from P to $R(P)$, $L(P)$, $B(P)$, $A(P)$ are the right connection, left connection, down connection, and up connection, respectively.

It is clear that there is no left connection in algorithm FB since we choose the connections from left to right. Also observe that there is no up connection followed immediately by a down connection, and vice versa.

Furthermore, considering the fact that F is a minimal blocking fault pattern, we have the following results:

Property 4.1 *There are no two consecutive down (resp. up) connections generated by algorithm FB.*

Proof:

We will only prove the case for down connections; the case for the up connections is similar and omitted.

To the contrary, suppose there are $n \geq 2$ down connections in column j ; without loss of generality, let $\{(i, j), (i+1, j), \dots, (i+n, j)\} \subseteq L$ be the n down connections, then (i, j) must have gained the control from $(i, j-1)$ by a right connection at $(i, j-1)$. Since down connection has the highest priority, $(i+1, j-1)$ must be non-connectable. Moreover, since $n \geq 2$, both $(i+2, j)$ and $(i+1, j)$ are in L , which means $(i+1, j-1) \in F$; that is, $(i, j-1) \in I(F)$ and $(i+2, j-1) \in E(F)$. However, there is a live path $\pi = \{(i, j-1), (i, j), (i+1, j), (i+2, j), (i+2, j-1)\}$ between $(i, j-1) \in I(F)$ and $(i+2, j-1) \in E(F)$, which contradicts the definition of F . $\square$

From the above observations, we can see easily make the following conclusion:

Corollary 4.1 *For any two consecutive connections in FB, there are only three possible combinations; that is, down - right, up - right, right - *, where * can be any one of the following: up, down, right.*

According to the above properties, FB can be reformulated as follows:

Algorithm FB:

INPUT: A minimal blocking fault pattern F

OUTPUT: A line segment

Begin

$L \leftarrow \emptyset$

$r \leftarrow f_1(x)$

current-node $P \leftarrow (r-1, 1)$

While $P(y) \neq g_k$ **do**

 add P to L

if $B(P)$ is connectable **then**

 add $B(P)$ to L

$P \leftarrow R(B(P))$

if $R(P)$ is connectable **then**

Figure 4.4: Illustration of the result of Algorithm FB

```

     $P \leftarrow R(P)$ 
    if  $A(P)$  is connectable then
        add  $A(P)$  to  $L$ 
         $P \leftarrow R(A(P))$ 
    Endif
Endwhile
add  $P$  to  $L$ 
output  $L$ 
End

```

Example 4.1: In Figure 4.4, we show an example of the result of algorithm FB.

Property 4.2 Let L be the output of Algorithm FB with input fault pattern F , then:

- (1) : L is a path from $(r-1, 1)$ to a node in column g_k ; that is, L spans over all the columns;
- (2) : let $\Omega_j = \{P \mid P \in L, P(y) = j\}$, then $1 \leq |\Omega_j| \leq 2$ for all j ;
- (3) : for any $P \in S_i$, if P is non-connectable and $A(P)$ is connectable, then $A(P) \in L$;
- (4) : for any $P \in L$, $P(x) \geq r-1$.

Proof:

- (1) : It is obvious since, in the algorithm, each connection is only between two neighboring nodes and the algorithm ends when it reaches column g_k .
- (2) : It is a direct result from Corollary 4.1.

- (3) : Considering the fact that L spans over all columns, particularly column $P(y)$, then according to the algorithm, P will be connected since the down connection has the highest priority.
- (4) : Suppose to the contrary that there is some $P \in L$ such that $P(x) < r - 1$; let $\{P_1, P_2, \dots, P_s\}$ be the set of all such nodes, $Q = \max_{P(x)} \{\min_{P(y)} \{P_i; 1 \leq i \leq s\}\}$, and $j = Q(y)$. Clearly, $j \neq 1$. Since there are three type of connections in the Algorithm, let us examine them individually.

Case 1 : If Q has gained the control by a down connection (i.e., from $A(Q)$), then by Property 4.1, $A(A(Q)) \notin L$. Furthermore, since $j = Q(y) \neq 1$, $L(A(Q)) \in L$, which contradicts the definition of Q .

Case 2 : If Q has gained the control by an up connection (i.e., from $B(P)$), then by the definition of Q , $B(Q)(x) = r - 1$; that is, $R(B(Q)) \notin F$. Thus, according to the algorithm, we should have a right connection at $B(Q)$, instead of an up connection since the right connection has higher priority. Contradiction.

Case 3 : If Q has gained the control by a right connection (i.e., from $L(Q)$), then $L(P)(y) < P(y)$ and $L(P)(x) = P(x)$, which contradicts the definition of P .

□

Theorem 4.2 *Algorithm FB is correct, i.e., it terminates in finite time and generates a line segment between $(r - 1, 1)$ and $(r - 1, g_k)$.*

Proof:

By Corollary 4.1, it follows that, for any two consecutive connections, there will be at least one right connection. That is, within at most $2(g_k - 1)$ connections, $P(y)$ will become g_k and the algorithm thus terminates.

By Property 4.2 (1), Algorithm FB generates a path between $(r - 1, 1)$ and a node P in column g_k ; since $P \in I(F)$ and $(r, g_k) \in F$, $P(x) \leq r - 1$. On the other

Chapter 4. Reconfiguration Algorithms For $H > 2$

hand, by Property 4.2 (4), $P(x) \geq r - 1$, thus $P = (r - 1, g_k)$. That is, algorithm FB generates a line segment between $(r - 1, 1)$ and $(r - 1, g_k)$. $\square$

If it takes one time unit for a node to determine if any one of its neighbors is connectable, then we have the following

Theorem 4.3 *Algorithm FB has time complexity $O(g_k)$.*

Proof:

In Algorithm FB, in the worst case, 3 tries will be needed to establish a connection from the current node. From the proof of Theorem 4.2, we can see that the algorithm needs at most $2(g_k - 1)$ connections to form a line segment, thus the total time complexity is $6(g_k - 1) = O(g_k)$. $\square$

In the following discussion, let L be the line segment produced by Algorithm FB with fault pattern F and $\bar{F}$ be $OL(L)$ (the out-layer of L as defined in Section 3.3).

Theorem 4.4 *For any given minimal blocking fault pattern F , $\bar{F}$ is a minimal blocking fault pattern.*

Proof:

According to the definition of OL, $|\bar{F}| = g_k$ and $\bar{F}$ satisfies the CDP; thus, we need only to show that $\bar{F}$ is a blocking fault pattern. To show this, for any location $\bar{f} = (i, j) \in \bar{F}$, we have to show that $\psi(\bar{f}) \subseteq E(\bar{F}) \cup B(\bar{F})$ according to Theorem 2.2.

This clearly holds for $j = 1$ and $j = g_k$. For $1 < j < g_k$, we have $\psi(\bar{f}) = \{(i, j), (i + 1, j - 1), (i + 1, j + 1)\}$. Clearly $(i, j) \in B(\bar{F})$. Next, we will show that $(i + 1, j + 1) \in E(\bar{F}) \cup B(\bar{F})$; the case for $(i + 1, j - 1)$ will be similar.

Suppose to the contrary that $(i + 1, j + 1) \notin E(\bar{F}) \cup B(\bar{F})$; then, $(i + 1, j + 1) \in I(\bar{F})$. Let us denote the row number of the faulty element of $\bar{F}$ in column $j + 1$ by x_{j+1} ; then $x_{j+1} > i + 1$. Since $(x_{j+1}, j + 1)$ is the first node in column $j + 1$ of L , then $(x_{j+1}, j) \in L$; that is, FB must have at least two consecutive down connections $(i, j), (i + 1, j), \dots, (x_{j+1}, j)$, which contradicts Property 4.1. Thus $(i + 1, j + 1) \in E(\bar{F}) \cup B(\bar{F})$. $\square$

Definition 4.9 A node P is said to be a missed node of FB if the following holds: $P \in I(F)$ and $P \notin E(\bar{F})$. We denote the set of missed nodes of FB by $m(FB, F)$.

Recall that we defined in the previous section a decomposition of R_i for F , say $R_i = \{N_{ij} | 1 \leq j \leq n_i\}$ for some n_i , with $r \leq i \leq r + H_F - 1$; similarly, we denote by $\bar{R}_i$ the set of nodes in $I(\bar{F})$ on row i with $r - 1 \leq i \leq r + H_F - 2$. Then we have the following relation between R_i and $\bar{R}_{i-1}$:

Theorem 4.5 Let $N_{ij} = \{P_{j1}, P_{j2}, \dots, P_{jm_j}\}$, where $r \leq i \leq r + H_F - 1$ and $1 \leq j \leq n_i$, then

$$\bar{R}_{i-1} = \bigcup_{j=1}^{n_i} A(N_{ij} - \{P_{j1}, P_{jm_j}\})$$

Proof:

The proof will proceed in two steps.

Step 1 : we show that $\bar{R}_{i-1} \supseteq \bigcup_{j=1}^{n_i} A(N_{ij} - \{P_{j1}, P_{jm_j}\})$.

If $m_j = 1, 2$, then $N_{ij} - \{P_{j1}, P_{jm_j}\} = \emptyset$; thus, $A(N_{ij} - \{P_{j1}, P_{jm_j}\}) = \emptyset \subseteq \bar{R}_{i-1}$.

If $m_j > 2$, according to the definition of N_{ij} , both $L(P_{j1})$ and $R(P_{jm_j})$ are in F ; furthermore, since F is a minimal blocking fault pattern, then both $B(P_{j1})$ and $B(P_{jm_j})$ are in F . By Property 4.2 (3), $\{A(L(P_{j1})), A(P_{j1}), P_{j1}, P_{j2}, \dots, P_{jm_j}, A(P_{jm_j}), A(R(P_{jm_j}))\} \subset L$; thus,

$$\{A(P_{j2}), A(P_{j3}), \dots, A(P_{jm_j-1})\} = A(N_{ij} - \{P_{j1}, P_{jm_j}\}) \subseteq \bar{R}_{i-1}.$$

Step 2 : we show that $R_{i-1} \subseteq \bigcup_{j=1}^{n_i} A(N_{ij} - \{P_{j1}, P_{jm_j}\})$.

Let $P \in \bar{R}_{i-1}$, then $B(P) \in R_i$; without loss of generality, let $B(P) \in N_{ij}$ for some $1 \leq j \leq n_i$; that is, $P \in A(N_{ij})$. Also, since $P \in \bar{R}_{i-1}$, and $P \notin L$, $B(P) \neq P_{j1}, P_{jm_j}$. Thus, $\bar{R}_{i-1} \subseteq \bigcup_{j=1}^{n_i} A(N_{ij} - \{P_{j1}, P_{jm_j}\})$.

Hence, the result follows. $\square$

Corollary 4.2 Let $v_i(F)$ and $v_i(\bar{F})$ be the valley-width of R_i and $\bar{R}_i$ respectively, then, for each i with $r \leq i \leq r + H_F + 1$:

- 1) : For each $r \leq i \leq r + H_F - 2$, $v_{i-1}(\tilde{F}) = v_i(F) - 2$;
- 2) : For $i = r + H_F - 1$, if $v(F) > 2$ then $v(\tilde{F}) = v(F) - 2$ and $H_F = H_F$;
otherwise $H_F = H_F - 1$ and $v(\tilde{F}) = v_{i-1}(F) - 2$.

Now we can get a set of non-crossing line segments by applying Algorithm FB repeatedly. More precisely, we start with the fault pattern F in S_l as the input to Algorithm FB, and then we apply Algorithm FB to the OL of the previous resulting line segment, and so on, until we get a line segment which contains a complete row of S_l .

Algorithm FBS: Forming a Set of Line Segments with FB

INPUT: A minimal blocking fault pattern F

OUTPUT: A set of non-crossing line segment

Begin

$L \leftarrow \emptyset$

While L does not include a complete row **do**

$L \leftarrow FB(F)$

output L

$F \leftarrow OL(L)$

Endwhile

End

Example 4.2: In Figure 4.5, we show an example of the result of algorithm FBS in which $g_k = 22$, $H_F = 4$; the total number of iterations is 10.

Theorem 4.6 *For any given minimal blocking fault pattern F , Algorithm FBS terminates in at most $g_k H_F / 2$ iterations and produces a set of non-crossing line segments in F .*

Proof:

It is clear that if the algorithm ever terminates, then it will produce a set of non-crossing line segments. To prove that Algorithm FBS terminates in at most $g_k H_F / 2$ iterations, we present a proof by induction on H_F .

Step 1 : $H_F = 1$.

Figure 4.5: Illustration of the result of Algorithm FBS

If $v(F) \leq 2$, then according to Corollary 4.2 $H_F = 0$; that is, the algorithm terminates in one iteration.

If $v(F) > 2$, then according to Corollary 4.2 $v(\tilde{F}) = v(F) - 2$; thus, after $\lceil v(F)/2 \rceil$ iterations, the height of the resulting fault pattern will become 0. That is, the algorithm terminates in $\lceil v(F)/2 \rceil$ iterations in the worst case. Since $v(F) < g_k$, the algorithm will terminate in at most $g_k/2$ iterations if $H_F = 1$.

Step 2 : Suppose the statement is true for any minimal fault pattern of height $h \geq 1$.

Given a minimal blocking fault pattern F with height $h + 1$, by the same argument as above, after $\lceil v(F)/2 \rceil$ iterations, the input fault pattern of the algorithm will have height h . Thus the algorithm will terminate in at most $v(F)/2 + g_k h/2 < g_k(h + 1)/2 = g_k H_F/2$ iterations.

Thus, Algorithm FBS terminates in at most $g_k H_F/2$ iterations in all cases. $\square$

Corollary 4.3 *Algorithm FBS has time complexity $O(g_k^2 H_F)$.*

Lemma 4.1 *For any given minimal blocking pattern F , let $F_1 = F$, $L_1 = FB(F_1)$, $F_{i+1} = OL(L_i)$, and $m_i = m(FB, F_i)$ in FBS; then $m(FBS, F) = \cup_i m_i$.*

Proof:

It is an immediate result from the definition of the missed nodes of both FBS and FB. $\square$

Theorem 4.7 *For each odd of F , there will be exactly one missed node and there is no other missed node for Algorithm FBS; that is, $|m(FBS, F)| = |\text{odd}(F)|$.*

Proof:

The proof consists of the following four steps.

Step 1: We claim that $SGT(F_i) = m_i$.

On one hand, we have $SGT(F_i) \subseteq m_i$ by the definition of singleton.

On the other hand, we will show that $m_i \subseteq SGT(F_i)$ in the following. That is, if any node $P \in I(F_i)$ and $P \notin SGT(F_i)$, then $P \notin m_i$:

If $B(P)$ is non-connectable with respect to F_i , then by Property 4.2

(3) $P \in L_i$;

If $B(P)$ is connectable, but $B(B(P))$ is non-connectable, then by Property 4.2 (2) and (3) $P \in I(F_{i+1}) \cup L_i$;

If both $B(P)$ and $B(B(P))$ are connectable, then by Property 4.2 (2) $P \in I(F_{i+1})$.

Thus, $SGT(F_i) = m_i$.

Step 2: We claim that $|\text{odd}(F_{i+1})| = |\text{odd}(F_i)| - |SGT(F_i)|$.

This follows immediately from Theorem 4.5.

Step 3: We claim that $|\text{odd}(F)| = \sum_i |SGT(F_i)|$.

By step 2, $|\text{odd}(F_i)| = |\text{odd}(F_{i+1})| + |SGT(F_i)|$, thus, $|\text{odd}(F)| = |\text{odd}(F_1)| = \dots = \sum_i |SGT(F_i)|$.

Step 4: we have $|m(FBS, F)| = \sum_i |m_i| = \sum_i |SGT(F_i)| = |\text{odd}(F)|$ as desired. $\square$

4.4 Another Line Segment Formation Algorithm

As we have seen in the last section, it may require up to $g_k H / 2$ rows of non-faulty nodes above row r in S_l for *FBS* to run to completion; this is a very strong or even impossible condition in some cases. The main reason for this is that it requires $\lceil r/2 \rceil$ iterations to reduce the height of the fault patterns by 1 in Algorithm *FBS*. In this section, we introduce a different scheme to form line segments in S_l , namely, the "simple bounce" algorithm, or shortly *SB*.

Intuitively, all the nodes in row $r - 1$ are non-faulty, that is, the straight line $\{(r - 1, 1), (r - 1, 2), \dots, (r - 1, g_k)\}$ between $(r - 1, 1)$ and $(r - 1, g_k)$ is a trivial line segment which is also a single line as specified in step 3. However, if we use this "straight line" scheme (or shortly *SL*) as the line segments formation algorithm, then all the non-fault nodes in $I(F)$ below row r are missed; that is, $m(SL(F)) = I(F)$.

Let $P = (r - 1, j)$, where $1 \leq j \leq g_k$. In this section, we will take this simple *SL* algorithm as the main structure and try to improve its performance. The idea is to connect as many nodes from $I(F)$ as possible while keeping $SL(F)$ as part of the new line segment.

To be precise, we will take P_i as the starting node; for any two consecutive connections in L , say $\{P_{i-1}, P_i\}$ and $\{P_i, P_{i+1}\}$, we will keep the first connection $\{P_{i-1}, P_i\}$ and extend the second one $\{P_i, P_{i+1}\}$ by connecting the non-faulty nodes in column i and $i + 1$. If we denote the row coordinate of the faults of F in column i and $i + 1$ by x_i and x_{i+1} , respectively, and $x = \min\{x_i, x_{i+1}\}$, then the extension of $\{P_i, P_{i+1}\}$ will be $\{(r - 1, i), (r, i), \dots, (x, i), (x, i + 1), \dots, (r - 1, i + 1)\}$.

Analogously, we can extend the first connection and keep the second one. In the following discussion, we will use the first one for demonstration. We call this algorithm the "simple bounce" algorithm, or shortly, the *SB* algorithm.

The formal description of the *SB* algorithm is given below.

SB: Simple Bounce

INPUT: S_l and F

OUTPUT: a line segment between the two input nodes

Figure 4.6: Illustration of the *SB* algorithm

```

Begin
   $L \leftarrow \emptyset$ 
  For  $i = 1$  to  $g_k$ 
    add  $P_i$  to  $L$ 
    if  $i$  is even then
       $P \leftarrow B(P)$ 
      While  $B(P) \notin F$ 
        add  $P$  to  $L$ 
         $P \leftarrow B(P)$ 
      ENDWhile
      if  $R(P) \notin F$  then
        add  $P, R(P)$  to  $L$ 
      Endif
       $P \leftarrow A(R(P))$ 
      While  $P \neq P_{i+1}$  do
        add  $P$  to  $L$ 
         $P \leftarrow A(P)$ 
      EndWhile
    Endif
  EndFor
  return  $L$ 
End

```

Example 4.3: In Figure 4.6, an example of the *SB* algorithm is shown with $g_k = 18$. We use 'm' to indicate the missed nodes.

Theorem 4.8 *Algorithm SB is correct; that is, it terminates in finite time and produces a line segment between $(r-1, 1)$ and $(r-1, g_k)$.*

Proof:

Similar to the proof of Theorem 4.2. $\square$

Theorem 4.9 *The line segment constructed by algorithm SB includes all the nodes in row $r - 1$.*

Proof:

By construction. $\square$

To analyze the performance SB, we will examine the size of $m(SB, F)$ for any given F .

Definition 4.10 *For any minimal blocking fault pattern F , if there exist i, j such that $(i, j), (i+1, j+1) \in F$, then $DM_i(F) = \{(i, j), (i+1, j+1)\}$ is a down movement of F on column j . We denote by $DM(F)$ the set $DM(F) = \cup_{i=1}^{g_k} DM_i(F)$.*

Similarly, we define

Definition 4.11 *For any minimal blocking fault pattern F , if there exist i, j such that $(i, j), (i, j+1) \in F$, then $FM_i(F) = \{(i, j), (i, j+1)\}$ is a flat movement of F on column j . We denote by $FM(F)$ the set $FM(F) = \cup_{i=1}^{g_k} FM_i(F)$.*

Definition 4.12 *For any minimal blocking fault pattern F , if there exist i, j such that $(i, j), (i-1, j+1) \in F$, then $UM_i(F) = \{(i, j), (i-1, j+1)\}$ is a up movement of F on column j . We denote by $UM(F)$ the set $UM(F) = \cup_{i=1}^{g_k} UM_i(F)$.*

Property 4.3 *For any minimal blocking fault pattern F , $|DM(F)| = |UM(F)| \leq g_k/2$.*

Proof:

Let r be the first row of S_l containing a faulty node of F , consider the movement of F ; due to the property of minimal blocking fault pattern, F has one of the three movements between every two adjacent columns. Furthermore, since both $(r, 1)$ and (r, g_k) are in F ; that is, F starts from and ends at row r . Thus, the result follows.

$\square$

Figure 4.7: The missed nodes of the SB algorithm

Theorem 4.10 *In the worst case, $|m(SB, F)| \leq g_k/2$.*

Proof:

Consider any sequence of consecutive down movements of F , say $\{(i, j), (i+1, j+1), \dots, (i+l, j+l)\}$. If j is even, then SB will miss one node on each of the columns $j+1, j+3, \dots$; thus, the number of missed nodes of SB from column j to column $j+l$ is $\lfloor l/2 \rfloor$. If j is odd, then the missed nodes will occur on columns $j+2, j+4, \dots$; again, the number of missed nodes on these columns is $\lfloor l/2 \rfloor$. That is, for any l consecutive down movements, SB will have at most $l/2$ missed nodes. These situations are illustrated in Figure 4.7.

Similarly, for any l consecutive up movements, SB will have at most $l/2$ missed nodes.

Finally, for any sequence of consecutive flat movements of F , SB will have at most 2 missed nodes at the ends.

Hence, by Property 4.3, $|m(SB, F)| \leq |DM(F)|/2 + |UM(F)|/2 = |DM(F)| \leq g_k/2$. $\square$

Observe that, in the case when $H_F = 1$ and $v(F) < 2$, FBS and SB will become the same. Normally, we can view the two algorithms as two extreme cases: FBS , on

one extreme, requires up to $C(g_k H_F)$ rows above row r and $|m(FBS, F)| = |odd(F)|$; SB , on the other extreme, requires just 1 row above row r and $|m(SB, F)| = O(g_k)$.

Clearly, for each odd of F , SB will have at least one missed node; thus, we have $|m(SB, F)| \geq |m(FBS, F)|$. That is, FBS is better than SB , providing that we have enough rows above row r for FBS . However, if there is only one row available above F , we are forced to use SB .

In next section, we will consider the cases when the number of rows of non-faulty nodes are between these two extreme cases.

4.5 A General Line Segments Formation Algorithm

In this section, we will consider the line segments formation algorithm for more general situations: when the number of rows of non-faulty nodes h is larger than 1. We will combine the first two algorithms FBS and SB to get an algorithm which works for any $h \geq 1$, namely the "follow the border and bounce" algorithm, or shortly FBB .

Recall the comparison between FBS and SB : FBS has a better performance than SB , but it is more restricted than SB , since it is applicable only when $h \geq g_k H_F$. It is reasonable and desirable to have an algorithm applicable to any h while the number of missed nodes is a non-increasing function of h .

Also recall that in the SB algorithm, we refined the straight line by extending it with the connections to the non-faulty nodes in the same column. Now we can apply a similar technique to modify FB .

Given a minimal blocking fault pattern F and S_l , for any $1 \leq u \leq H_F + 1$, let $i = r + H_F - u + 1$ and $N_{ij} = \{(i, l), (i, l+1), \dots, (i, l+d)\}$ be a cluster of non-faulty nodes in R_i . Consider the straight line between $(i-1, l-1)$ and $(i-1, l+d+1)$: we can apply to this line the same extension technique as we did in SB . We call this operation the cluster bounce CB with the last u rows from column $l-1$ to column $l+d+1$, denoted by $CB(u, l, d, F)$.

More precisely, the CB operation can be formally given as follows, where $P_j = (i-1, j)$.

CB: Cluster Bounce

INPUT: u, l, d and F

OUTPUT: a line segment

Begin

$L \leftarrow \emptyset$

For $j = -1$ to $d+1$

add P_{i+j} to L

if j is even then

$P \leftarrow B(P)$

While $B(P) \notin F$

add P to L

$P \leftarrow B(P)$

Endwhile

if $R(P) \notin F$ then

add $P, R(P)$ to L

Endif

$P \leftarrow A(R(P))$

While $P \neq P_{j+1}$ do

add P to L

$P \leftarrow A(P)$

Endwhile

Endif

EndFor

return L

End

Example 4.4 In Figure 4.8, we show an example of the CB operation. In this example, $g_k = 22, r = 4, H_F = 4, u = 4, i = 5$ and $l = 3, d = 6$.

Observe that if $u = 1$ then $i = r + H_F$. Since $f_{g_k}(x) = r + H_F$, then $R_i = \emptyset$. Thus, there is no CB operation at all; or, in other words, $CB(F) = \emptyset$. Also observe that if $u = H_F + 1$ then $i = r$; that is, the CB operation will connect nodes starting and ending at row $r-1$, and the resulting line segment will be part of $SB(F)$; in other words, $CB \subseteq SB$.

Figure 4.8: Illustration of the CB operation

For any given $2 \leq u \leq H_F$, we can use the CB operation to modify the FB algorithm to obtain a new algorithm, the FBB algorithm. The main idea is that for the connections between row $r-1$ and row $i-1$, we use FB ; for the connections below row $i-1$, we use the CB operation on each cluster of non-faulty nodes in row i .

For any pair of inputs u and F , we use the following scheme to detect the occurrence of a cluster of non-faulty nodes: once the current node P is on row $i-1$, according to the property of minimal blocking fault pattern, we know that $L(P), B(P) \in F$; furthermore, if $B(R(P))$ is connectable, we can conclude that we have encountered a cluster of R_i ; otherwise, we repeat this process with $R(P)$ as the new current node.

The algorithm FBB can be formalized as follows:

Algorithm FBB: Follow the border and bounce

INPUT: F and u

OUTPUT: A line segment

Begin

$L \leftarrow \emptyset$

$i = r + H_F - u$

current-node $P \leftarrow (r-1, 1)$

While $P(y) \neq g_k$ **do**

if $P(x) = i-1$ **then**

While $B(R(P))$ is non-connectable **do**

 add P to L

$P \leftarrow R(P)$

Figure 4.9: Illustration of Algorithm *FBB*

```

Endwhile
   $Q \leftarrow \text{find.NextEnd}(P, F)$ 
  add  $CB(i, P(y), Q(y), F) - \{Q\}$  to  $L$ 
   $P \leftarrow Q$ 
Endif
add  $P$  to  $L$ 
if  $B(P)$  is connectable then
  add  $B(P)$  to  $L$ 
   $P \leftarrow R(B(P))$ 
if  $R(P)$  is connectable then
   $P \leftarrow R(P)$ 
if  $A(P)$  is connectable then
  add  $A(P)$  to  $L$ 
   $P \leftarrow R(A(P))$ 
Endif
Endwhile
add  $P$  to  $L$ 
output  $L$ 
End

```

Function $\text{find.NextEnd}(P, F)$ finds the next end point for CB : that is, if f is the next faulty node of F in row i from column $P(y)$, it returns $A(f)$.

Example 4.5: In Figure 4.9, we show the resulting line segment generated by *FBB* with the same minimal blocking fault pattern used for Examples 4.2 and 4.4. Here $u = 3$.

Using the same structures of the proof of the corresponding theorems for algorithm *FB*, we have the following results for *FBB*.

Theorem 4.11 *For any minimal blocking fault pattern F and any given $1 \leq u \leq H_F$, let $L = FBB(u, F)$; then $P(x) \geq r - 1$ for any $P \in L$, where $r = f_1(x)$.*

Theorem 4.12 *Algorithm FBB terminates in finite time and generate a line segment between $(r - 1, 1)$ and $(r - 1, g_k)$.*

Theorem 4.13 *For any given minimal blocking fault pattern F and u , the out-layer $\bar{F}$ of the line segment L generated by FBB will be a minimal blocking fault pattern.*

The following theorem has a slightly different proof compared to the corresponding one for FB .

Theorem 4.14 *For any given $2 \leq u \leq H_F + 1$, let $\bar{F}$ be the out-layer of the output of algorithm FBB with input u and F ; then $H_{\bar{F}} = H_F - u + 1$.*

Proof:

According to FBB , the first and the last fault of $\bar{F}$ will be on row $r - 1$ and row $i - 1 = r + H_F - u$, respectively. Thus, $H_{\bar{F}} = H_F - u + 1$. $\square$

For any given F and any row i , $r \leq i \leq r + H_F - 1$, denote the down movements, the flat movements and the up movements of F below row i by $DM^i(F)$, $FM^i(F)$ and $UM^i(F)$, respectively.

Theorem 4.15 *Let F be a minimal blocking fault pattern, $2 \leq u \leq H_F + 1$, $i = r + H_F - u$, $m(FBB(u, F)) = |DM^i(F)|$.*

Proof:

Similar to the proof of Theorem 4.10. $\square$

As an immediate consequence of Theorem 4.15, we have the following corollary:

Corollary 4.4 *For any given two integers u_1, u_2 with $2 \leq u_1 \leq u_2 \leq H_F + 1$, $|m(FBB(u_1, F))| \leq |m(FBB(u_2, F))|$.*

A new line segments generation algorithm *FBBS* is formed by repeatedly using *FBB* with an appropriate u at each iteration. According to Theorem 4.4, we want to make u as small as possible to minimize the number of missed nodes; on the other hand, we want the algorithm to be completed in at most h iterations.

For any given $h \geq 1$ and a minimal fault pattern F , let $u_1, u_2, \dots, u_h$ be a set of integers where $2 \leq u_i \leq H_F + 1$, and $\sum_i u_i = h + H_F$. we can describe *FBBS* as follows:

Algorithm FBBS: Line Segments Generation by *FBB*

INPUT: $h, u_1, u_2, \dots, u_h, F$

OUTPUT: A set of non-crossing line segments

Begin

$L \leftarrow \emptyset$

For $i = 1$ to h **do**

$L \leftarrow FBB(u_i, F)$

output L

$F \leftarrow OL(L)$

Endfor

End

Theorem 4.16 *FBBS generates h non-cross line segments and the last one contains a complete row of S .*

Proof:

According to Theorem 4.14, the heights of the fault pattern after iteration $1, 2, \dots, h$ are $H_F - u_1 + 1, H_F - u_1 - u_2 + 2, \dots, H_F - (\sum_i u_i) + h$. Since $\sum_i u_i = h + H_F$, the height of the fault pattern after iteration h will be 0; that is, the h -th line segment contains a complete row of S as desired. $\square$

Notice a very interesting situation for *FBBS*: when $h = H_F$ and $u_i = 2, 1 \leq i \leq h$. In this case, we only apply the bounce technique to the last row; so we call this type of *FBBS* algorithm the "bounce to the last row only", or simply the *BLRS*.

Theorem 4.17 *For each odd of F , there will be exactly one missed node and there is no other missed node for Algorithm BLRS; that is, $|m(BLRS, F)| = |odd(F)|$.*

Proof:

Observe that all the singletons of F will be missed after one iteration in *BLRS*. Furthermore, since the *CB* operation is only applied to the last row, according to Corollary 4.2, there will be exactly one odd of the resulting fault pattern F for each non-singleton odd of F above the last row. However, for each non-singleton odd of F on the last row, according to the *CB* operation, we will miss one node after this iteration. That is, if we denote the set of odds on the last row of F by $oddl(F)$, then we have the following facts:

- 1): $|oddl(F_{i+1})| = |oddl(F_i)| - |(SGT(F_i) \cup oddl(F_i))|$ for $1 \leq i < H_F$;
- 2): $oddl(F_{H_F}) = SGT(F_i) \cup oddl(F_{H_F}) = oddl(F_{H_F})$; and
- 3): $m_i(BLRS, F) = SGT(F_i) \cup oddl(F_i)$ for $1 \leq i \leq H_F$.

Thus,

$$\begin{aligned}
 |m(BLRS, F)| &= \sum_{i=1}^{H_F} |m_i(BLRS, F)| \\
 &= \sum_{i=1}^{H_F} |SGT(F_i) \cup oddl(F_i)| \\
 &= \sum_{i=1}^{H_F-1} |oddl(F_i)| - |oddl(F_{i+1})| + |oddl(F_{H_F})| \\
 &= |oddl(F_1)| = |oddl(F)|
 \end{aligned}$$

□

Example 4.6: In Figure 4.10, we show the result of Algorithm *BLRS* with the same minimal blocking fault pattern used for Algorithm *FBS* in Examples 4.2, 4.4, 4.5. As we can see, *BLRS* uses only 4 iterations instead of the 10 iterations required by *FBS*, while the number of missed nodes still be 3.

Clearly, if $h \geq H_F$, $u_i = 2$ is the best choice for *FBBS*. If $h < H_F$, this set of u_i 's is not a valid choice for *FBBS*. In this case, we can take any valid combination of u_i 's for *FBBS*. For simplicity, we choose $u_1 = u_2 = \dots = u_{h-1} = 2$, and $u_h = H_F - h + 2$. That is, we apply the algorithm *BLRS* for $h - 1$ iterations and then apply *SB* to the last iteration.

A similar variation of the above scheme is the following: we apply *FBB* with $u = H_F - h + 2$ for the first line segment, then we apply *BLRS* for the remaining iterations.

Figure 4.10: Illustration of the result of Algorithm *BLRS*

The number of missed nodes for these two approaches is comparable, although for some fault patterns, one may perform better than the other.

4.6 Ring Formation

Recall from Section 4.2 that, in step 2 of the subring formation algorithm, we want to build a set of non-crossing line segments in S_l and in S_u , respectively. By observing the similarity between S_l and S_u , we developed a set of common algorithms for both S_l and S_u in Sections 4.3, 4.4, and 4.5.

In this section, we present an algorithm for step 3 and 4, that is, to connect the line segments to form a line in S_l and in S_u , respectively, and finally to connect the two lines to form a ring by using S_m . Observe that, once we applied the shift rotate operations in S_u , the line segments formed in S_u will be similar to those in S_l ; thus, we can also develop common algorithms for step 3 as described in the following.

4.6.1 Connecting Line Segments to a Line in S_l

In this section, we present an algorithm to connect the set of line segments, produced by any line segments formation algorithm described before, to form one line in S_l , such that both its end points are on the top of all the line segments in S_l . By "top of all the line segments" we mean that, if $f_1 = (r, 1)$ and there are h line segments, the two end points for the line are both on row $r - h$. When constructing this line, we

want to keep the structure of the line segments as much as possible while including in the resulting line all the nodes of these line segments.

Let the n line segments be $L_1, L_2, \dots, L_h$ and each L_i have d_i elements, say $L_i = \{P_{i1}, P_{i2}, \dots, P_{id_i}\}$. Without loss of generality, assume that $P_{i1}(x) > P_{j1}(x)$ if $i > j$. Considering the way each line L_i is constructed, we have the following property:

Property 4.4 *For each L_i , where $1 \leq i \leq h$, there is exactly one node in each of columns 1, 2, $g_k - 1$, and g_k .*

Proof:

By algorithms *FB*, *SB*, and *FBB*. □

Since the end points of the resulting line have to be on row $r - h$ in S_i , we have to start and end the resulting line at line segment L_1 ; thus, it is impossible to completely keep the structure of L_1 . By Property 4.4, without loss of generality, we choose the first node P from the left in L_1 as one end point of the resulting line. To construct this line, we start from P and perform the following tasks: connect all the nodes in the first column; go along L_h until the last column is reached; connect the last node in L_{h-1} ; then go along L_{h-1} in reversed order until the node in column 2 is reached; connect the line L_{h-2} on column 2 and go along L_{h-2} ; and so on, until all the elements in L_1 is included. More precisely:

Algorithm LFORM: Form A Line From Some Line Segments

INPUT: $L_1, L_2, \dots, L_h$, where $L_i = \{P_{i1}, P_{i2}, \dots, P_{id_i}\}$

OUTPUT: a line L

Begin

For $i = 1$ to h **do**

 add P_{i1} to L

Endfor

For $i = 2$ to d_h **do**

 add P_{hi} to L

Endfor

$j \leftarrow h - 1$

While $j > 0$ **do**

Figure 4.11: Illustration of the result of Algorithm LFORM

```

if  $(n - j)$  is odd then
  For  $i = d_j$  to 2 do
    add  $P_{ji}$  to  $L$ 
  Endfor
else
  For  $i = 2$  to  $d_j$  do
    add  $P_{ji}$  to  $L$ 
  Endfor
Endif
decrement  $j$ 
Endwhile
output  $L$ 
End

```

Example 4.7: In Figure 4.11, we show the result of Algorithm LFORM with some line segments produced by BLRS. In this example, we use dashed lines to represent the line segments and solid line for the resulting line.

Property 4.5 *The line formed by LFORM contains all the nodes of the input line segments $L_1, L_2, \dots, L_h$ in S_1 .*

Proof:

By construction. □

Notice that in Algorithm LFORM, if h is even, then the two end points of the resulting line will be $(r - h, 1)$ and $(r - h, 2)$; otherwise, they will be $(r - h, 1)$ and $(r - h, g_k)$.

4.6.2 Connecting the Three Parts to Form A Ring

In this section, we will connect the two lines obtained by applying the line formation algorithm *LFORM* in S_u and in S_l ; the connection will use the nodes in S_m .

Recall that the first step of the subring formation algorithm is to decompose S based on a decomposition $H = h_l + h_m + h_u$. For any specific decomposition, the two end points of the line L formed by algorithm *LFORM* in S_u are: $(r - h_l, 1)$ and $(r - h_l, 2)$, if h_l is even; and $(r - h_l, 1)$ and $(r - h_l, g_k)$, if H_l is odd.

Similarly, for S_l , by applying *LFORM* in S_l , we get another line L' . Let $f_{u,k} = (r', s')$, where $r' = r - H - 1$; then the two end points of L' are: $(r' + h_u, s')$ and $(r' + h_u, s' - 1)$, if h_u is even and $s' \neq 1$; $(r' + h_u, s')$ and $(r' + h_u - 1, g_k)$, if h_u is even and $j = 1$; $(r' + h_u, s')$ and $(r' + h_u - 1, s' - 1)$, if h_u is odd and $s' \neq g_k$; $(r' + h_u, s')$ and $(r' + h_u, g_k)$, if h_u is odd and $j = g_k$.

The main objective in this section is to form a ring by connecting the end points of the two lines L and L' . The major concern is to include as many nodes from S_m as possible and maintain the structures of L and L' .

If $h_m > 1$, we have h_m complete rows of non-faulty nodes in S_m , which are row $r' + h_u + 1, r' + h_u + 2, \dots, r - h_l - 1$. We can always reduce this case to one of the cases for $h_m = 1$ by extending one of the two lines, say L , in a fashion similar to the one used by algorithm *LFORM*. That is, on column 1, we can connect $(r' + h_u + 2, 1), (r' + h_u + 3, 1), \dots, (r - h_l - 1, 1)$ to $(r - h_l, 1)$; on the other end, by using column 2 and column g_k , we can connect all the remaining nodes in row $r' + h_u + 2, \dots, r - h_l - 1$ to L at the other end point of L . Clearly, after this extension, the two end points of L will be either $(r' + h_u + 2, 1)$ and $(r' + h_u + 2, 2)$ or $(r' + h_u + 2, 1)$ and $(r' + h_u + 2, g_k)$.

Let us now consider the problem of choosing the appropriate decomposition of H . As discussed above, any such choice would have an impact on the performance of the algorithm. A very important observation is that, in step 1 of the subring formation algorithm, we can choose $h_m = 1$ to improve the performance of step 2, without degrading the performance of step 4. That is, it is possible to decrease the number of missed nodes in S_l and in S_u by increasing h_l and H_u , respectively.

In general, if $H \geq H_{F_L} + H_{F_u} + 1$, then we will choose $h_u = H_{F_u}$, $h_l = H_{F_L}$, and $h_n = H - h_l - h_u$.

Without loss of generality, let us assume that $h_n = 1$. Denote by P, Q and P', Q' the two end points of L and L' , respectively, where $P = (r' + h_u + 2, 1)$, $P' = (r' + h_u, s')$, $Q = (r' + h_u + 2, 2)$ or $(r' + h_u + 2, g_k)$, and

$$Q' = \begin{cases} (r' + h_u, s' - 1) & \text{if } h_u \text{ is even; } s' \neq 1 \\ (r' + h_u - 1, g_k) & \text{if } h_u \text{ is even; } s' = 1 \\ (r' + h_u - 1, s' + 1) & \text{if } h_u \text{ is odd; } s' \neq g_k \\ (r' + h_u, 1) & \text{otherwise} \end{cases}$$

It is clear that there are exactly eight different cases depending on the positions of Q and Q' . We will examine each case individually and describe how to connect the two lines L and L' to form a ring:

Case 1 $Q = (r' + h_u + 2, 2)$ and $Q' = (r' + h_u, s' - 1)$

Begin

$C \leftarrow L = \{P, \dots, Q\}$

add $A(Q), R(A(Q)), \dots, B(Q')$ to C

add $L' = \{P'\} = \{Q', \dots, P'\} - \{P'\}$ to C

add P' to C

$T \leftarrow P'$

While $T \neq (r' + h_u, g_k)$ **do**

add $B(T), R(B(T)), R(T)$ to C

If $R(T) \neq (r' + h_u, g_k)$ **then**

add $R(R(T))$ to C

$T \leftarrow R(R(T))$

Else

$S \leftarrow R(T)$

Endif

Endwhile

add $A(P)$ to C

End

Case 2 $Q = (r' + h_u + 2, 2)$ and $Q' = (r' + h_u - 1, s' + 1)$ with $s' \neq q_k$

Begin

$C \leftarrow L = \{P, \dots, Q\}$

add $A(Q), R(A(Q)), \dots, B(P')$ to C

add $L' = \{P', \dots, Q'\}$ to C

add $B(Q')$ to C

$T \leftarrow B(Q')$

While $T \neq (r' + h_u, g_k)$ **do**

add $B(T), R(B(T)), R(T)$ to C

If $R(T) \neq (r' + h_u, g_k)$ **then**

add $R(R(T))$ to C

$T \leftarrow R(R(T))$

Else

$T \leftarrow R(T)$

Endif

Endwhile

add $A(P)$ to C

End

Case 3 $Q = (r' + h_s + 2, 2)$ and $Q' = (r' + h_s + 1)$

Begin

$C \leftarrow L = \{P, \dots, Q\}$

add $A(Q), R(A(Q)), \dots, B(P')$ to C

add $L' = \{P', \dots, Q'\}$ to C

add $B(Q')$ to C

End

Case 4 $Q = (r' + h_s + 2, 2)$ and $Q' = (r' + h_s + 1, g_k)$

Begin

$C \leftarrow L = \{P, \dots, Q\}$

add $A(Q), A(A(Q))$ to C

$T \leftarrow A(A(Q))$

If $T \neq (r' + h_s, g_k)$ then

$T \leftarrow R(T)$

add T to C

Endif

While $T \neq (r' + h_s, g_k)$ do

add $B(T), R(B(T)), R(T)$ to C

If $R(T) \neq (r - H_{F_1} - h - 1, g_k)$ then

add $R(R(T))$ to C

$T \leftarrow R(R(T))$

Else

$T \leftarrow R(T)$

Endif

Endwhile

add $L' = \{Q', \dots, P'\}$ to C

add $A(P)$ to C

End

Case 5 $Q = (r' + h_x + 2, g_x)$ and $Q' = (r' + h_x, s' - 1)$ with $s' \neq 1$

Begin

$C \leftarrow L = \{P, \dots, Q\}$

add $A(Q)$ to C

$T \leftarrow A(Q)$

While $T \neq R(B(P'))$ **do**

add $A(T), L(A(T)), L(T)$ to C

If $L(T) \neq R(B(P'))$ **then**

add $L(L(T))$ to C

$T \leftarrow L(L(T))$

Else

$T \leftarrow L(T)$

Endif

Endwhile

add $B(P')$ to C

add $L' = \{P', \dots, Q'\}$ to C

add $B(Q'), L(B(Q')), \dots, A(P)$ to C

End

Case 6 $Q = (r' + h_x + 2, g_x)$ and $Q' = (r' + h_x, 1)$

Begin

$C \leftarrow L = \{P, \dots, Q\} - \{Q\}$

add $A(L(Q)), L(A(L(Q))), \dots, B(Q')$ to C

add $\bar{L}' = \{Q', \dots, P'\}$ to C

add $B(P')$ to C

End

Case 7 $Q = (r' + h_u + 2, g_k)$ and $Q' = (r' + h_u - 1, s' + 1)$ with $s' \neq g_k$

Begin

$C \leftarrow L = \{P, \dots, Q\}$

add $A(Q), A(A(Q))$ to C

$T \leftarrow A(A(Q))$

If $T \neq B(Q')$ **then**

$T \leftarrow L(T)$

Endif

While $T \neq B(Q')$ **do**

add $B(T), L(B(T)), L(T)$ to C

If $L(T) \neq R(B(P'))$ **then**

add $L(L(T))$ to C

$T \leftarrow L(L(T))$

Else

$T \leftarrow L(T)$

Endif

Endwhile

add $\hat{L}' = \{Q', \dots, P'\}$ to C

add $B(P'), L(B(P')), \dots, A(P)$ to C

End

Case 8 $Q = (r' + h_u + 2, g_k)$ and $Q' = (r' + h_u - 1, g_k)$

Begin

$C \leftarrow L = \{P, \dots, Q\}$

add $A(Q), L(A(Q)), \dots, B(R(P'))$ to C

add $R(P'), R(P'), \dots, B(Q')$ to C

add $\hat{L}' = \{Q', \dots, P'\}$ to C

add $A(P)$ to C

End

Figure 4.12: Illustration of case 1 and case 2

Figure 4.13: Illustration of case 3 and case 4

The connections are illustrated in Figure 4.12, Figure 4.13, Figure 4.14, Figure 4.15.

Notice that in the worst case, we may miss up to 1 node in S_m in cases 1, 3, 4, 5, 7. In cases 2 and 8, we do not lose any; in case 6, we lose the node $Q = (v' + h_u + 2, g_k)$. Thus, we have the following result:

Theorem 4.18 *For any given S , there is at most one missed node in step 4 of the subring formation algorithm LSRF.*

Proof:

By construction. □

Theorem 4.19 *For any given S , the number of missed nodes in the subring formed by using the LSRF scheme with any of the line segments formation algorithms developed before is at most $g_k + 1$.*

Proof:

Clearly, it does not lose any node in Steps 1 and 3. By Theorem 4.18, in Step 1, it loses at most one node.

Consider Step 2 now. In the previous sections, we developed three line segments formation algorithms, i.e., *FBS*, *SB*, and *FBBS*. We examine them individually as follows:

If *FBS* is used as the line segments formation algorithm in Step 2, by Theorem 4.7, the number of missed nodes for S_u and S_l is at most $\text{odd}(F_u)$ and $\text{odd}(F_l)$, respectively; furthermore, both $\text{odd}(F_u)$ and $\text{odd}(F_l)$ are at most $g_k/2$. Thus, the total number of missed nodes in Step 2 is no more than g_k .

If *SB* is used as the line segments formation algorithm in Step 2, by Theorem 4.10, the number of missed nodes for both S_u and S_l is at most $g_k/2$; thus, the total number of missed nodes in Step 2 is no more than g_k .

If *FBBS* is used as the line segments formation algorithm in Step 2, by Corollary 4.4, the number of missed nodes for both S_u and S_l is at most $g_k/2$; thus, the total number of missed nodes in Step 2 is no more than g_k .

Hence, the result follows. □

4.7 Chapter Summary

In this Chapter, a set of algorithms for forming a subring in a given region S with at least three rows of non-faulty elements were developed. A general strategy for

forming the largest subring in S was described. Following the general strategy, a given S was firstly decomposed into three parts (upper part, middle part, and lower part). Line segments in both upper and lower parts were then formed and were connected with the help of the middle part to get a ring. It was shown that the number of missed nodes for all the algorithms is $g_k + 1$ in the worst case.

Chapter 5

Reconfiguration Algorithms For $H \leq 2$

5.1 Introduction

In this chapter, we will deal with the more restricted case when $H \leq 2$. In this case, a given region S does not contain three rows of non-faulty nodes and we can not apply the algorithms developed in Chapter 4. Therefore, we need a different strategy to handle this case.

We firstly present a necessary condition for the existence of a subring which spans all the columns in S . In case this condition is not satisfied, we will divide S into subregions and examine the subring formation problem in each one of them. Finally, we develop subring formation algorithms for a region in which this condition is satisfied.

This chapter is organized as follows: in Section 5.2, we introduce some terminology and present a necessary condition for the existence of a subring which spans all the columns in S ; in Section 5.3, we show that the necessary condition discussed in Section 5.2 is also a sufficient condition for any region S which contains wrap-around links. Furthermore, an algorithm to form a subring using wrap-around link is developed. In Section 5.3, we focus on the subring formation algorithm when wrap-around links cannot be used. Finally, Section 5.4 contains a summary of the chapter.

Figure 5.1: Illustration of the row distances

5.2 Necessary Conditions

Recall that S is bounded by two minimal blocking fault patterns F_l and F_u and normalized with respect to F_l . Through out this chapter, we will denote by x_i and x'_i the row coordinates of the two faulty nodes of F_l and F_u in column i , respectively, where $1 \leq i \leq g_k$.

Definition 5.1 For any column i , $1 \leq i \leq g_k$, the row distance $rd_i(S)$ (or simply rd_i) of S on column i is $x_i - x'_i$. The row distance $rd(S)$ of S is $\min_i \{rd_i(S)\}$.

Definition 5.2 For any column i , $1 \leq i < g_k$, the diagonal row distance $drd_i(S)$ (or simply drd_i) of S on column i is $\min\{x_i - x'_{i+1}, x_{i+1} - x'_i\}$. The diagonal row distance $drd(S)$ of S is $\min_i \{drd_i(S)\}$.

Definition 5.3 For any column i , $1 \leq i < g_k - 1$, the second diagonal row distance $sdrd_i(S)$ (or simply $sdrd_i$) of S on column i is $\min\{x_i - x'_{i+2}, x_{i+2} - x'_i\}$. The second diagonal row distance $sdrd(S)$ of S is $\min_i \{sdrd_i(S)\}$.

Example 5.1: In Figure 5.1, we show some row distances, diagonal row distances, and second diagonal row distances. In this figure, in column 4, $rd_4(S) = 3$, $drd_4(S) = 2$, $sdrd_4(S) = 1$; in column 5, $rd_5(S) = 3$, $drd_5(S) = 2$, $sdrd_5(S) = 2$.

Property 5.1 For any region S , the following results hold:

- (1): $rd_i \geq 1$, $drd_i \geq 0$, and $sdrd_i \geq -1$;

$$(2): drd_i - 1 \leq rd_i \leq drd_i + 1;$$

$$(3): drd_i - 1 \leq rd_{i+1} \leq drd_i + 1;$$

$$(4): rd_i - 2 \leq rd_{i+1} \leq rd_i + 2;$$

$$(5): sdrd_i - 2 \leq rd_i \leq sdrd_i + 2, \text{ where } i \leq j \leq i + 2.$$

$$(6): \text{if } drd_i = 0, \text{ then } rd_i = rd_{i+1} = 1; \text{ and if } sdrd_i = -1, \text{ then } rd_i = rd_{i+1} = rd_{i+2} = 1.$$

Proof:

(1): Clearly $rd_i \geq 1$, since $F_i \cap F_{i+1} = \emptyset$ and $F_i \cap F_{i+2}$. Furthermore, by Theorem 3.3, $x'_{i+1} \leq x'_i + 1 \leq x_i$ and $x'_i \leq x'_{i+1} + 1 \leq x_{i+1}$; that is, $drd_i \geq 0$. Finally, by Theorem 3.3 again, $x_i \geq x'_i + 1 \geq x'_{i+1} \geq x'_{i+2} - 1$; that is, $x_i - x'_{i+2} \geq -1$. Moreover, $x'_i \leq x_i - 1 \leq x_{i+1} \leq x_{i+2} + 1$; that is, $x_{i+2} - x'_i \geq -1$. Hence, $sdrd_i \geq -1$.

(2): Let $drd_i = a$, then either $x_i - x'_{i+1} = a$ or $x_{i+1} - x'_i = a$. If $x_i - x'_{i+1} = a$, by Theorem 3.3, we have $x_i - x'_i \leq x_i - (x'_{i+1} - 1) = a + 1$; that is, $rd_i \leq drd_i + 1$. Similarly, if $x_{i+1} - x'_i = a$, then we have $x_i - x'_i \leq x_{i+1} + 1 - x'_i = a + 1$, that is, $rd_i \leq drd_i + 1$.

In a similar fashion, it follows that $drd_i - 1 \leq rd_i$.

(3), (4), and (5): similar to the proof of (2).

(6): They are special cases of (2), (3), and (5), respectively.

□

Now, let us examine the necessary condition for a node of S in column i to be included in a subring within S .

In Section 4.3, we introduced the concept of singleton and the connectability of a node in S_l and S_u . We can extend these terms to nodes in S as follows:

Figure 5.2: Illustration of a cut of type 1

Figure 5.3: Illustration of a cut of type 2

Definition 5.4 For any node P in S , P is said to be a singleton if it is non-faulty and has degree 1. Furthermore, P is said to be connectable if it is neither a faulty node, nor a singleton; it is said to be non-connectable, otherwise.

Clearly, no singleton can appear in any subring in S , since its degree is 1.

Definition 5.5 Column g_k is said to be a cut of S of type 1 if $rd_{g_k}(S) = 2$.

This situation is shown in Figure 5.2.

Definition 5.6 For $1 \leq i \leq g_k - 1$, column i in S is said to be a cut of S of type 2 if $drd_i(S) = 1$.

This situation is illustrated in Figure 5.3.

Definition 5.7 A column i in S is said to be a cut of S of type 3 if $rd_i = 1$, and $drd_i(S) > 1$.

Figure 5.4: Illustration of a cut of type 3

This situation is illustrated in Figure 5.4.

Some properties of these cuts are given below

Lemma 5.1 *Let g_k be a cut of S of type 1; then no subring of S will contain nodes in columns 1 and g_k .*

Proof:

Since S is normalized with respect to F_l and since F_l is a minimal blocking fault pattern, we have $rd_1(S) \leq 2$. Thus, any node of S in columns 1 and g_k is either faulty or singleton, that is, no subring of S will contain nodes in columns 1 and g_k .

□

Lemma 5.2 *Let column i be a cut of S of type 2; then no subring of S will contain nodes in column i and $i + 1$.*

Proof:

If $drd_1(S) = 1$, then the only non-faulty nodes of S in columns i and $i + 1$ are singletons. □

Lemma 5.3 *Let column i be a cut of S of type 3; then no subring of S will contain node in column i .*

Proof:

By definition, there is no non-faulty node in column i for S . □

Lemma 5.4 *Let column g_k be a cut of S of type 1; then,*

OF/DE

2

PM-1 3 1/2"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

- (1): for any sequence of l consecutive columns $2, 3, \dots, l+1$, such that $rd_i(S) = 2$ for $2 \leq i \leq l+1$, no subring of S will contain nodes in these columns.
- (2): for any sequence of m consecutive columns $g_k - m, \dots, g_k - 1$, such that $rd_i(S) = 2$ for $g_k - m \leq i \leq g_k - 1$, no subring of S will contain nodes in these columns.

Proof:

- (1): Assume first that there is no cut among these l columns. In this case, there is only one non-faulty node in each of these columns and all of them are in the same row. Since g_k is a cut of S of type 1, any node of S in column 1 is non-connectable; this implies that the non-faulty node of S in column 2 will be non-connectable once we remove its links to the non-connectable nodes. Thus, no subring will contain node in column 2. By repeating this argument, we can see easily that no subring will contain nodes in any column i , $2 \leq i \leq l+1$.

If there is a cut among these l consecutive columns, say column i , $2 \leq i \leq l+1$, then i must be a cut of S of type 2, since $rd_i(S) = 2$. By Lemma 5.2, no subring of S will contain node in columns i and $i+1$. Recursively apply the above two arguments both to the columns 2 to $i-1$ and to the columns $i+2$ to $l+1$.

- (2): Similar to (1).

□

Lemma 5.5 *Let column i be a cut of S of type 2; then:*

- (1): for any sequence of l consecutive columns $i+2, i+3, \dots, i+l+1$, such that $rd_j(S) = 2$ for $i+2 \leq j \leq i+l+1$, no subring of S will contain nodes in these columns.
- (2): for any sequence of m consecutive columns $i-m, \dots, i-1$, such that $rd_j(S) = 2$ for $i-m \leq j \leq i-1$, no subring of S will contain nodes in these columns.

Proof:

Similar to Lemma 5.4 □

Lemma 5.6 *If column i is a cut of S of type 3, then $rd_{i+1}(S) = 3$.*

Proof:

By definition, $drd_i(S) > 1$. Thus, $rd_{i+1}(S) = 3$, according to Theorem 3.3. □

In the following, when no ambiguity arises, we will use the term cut to indicate any one of the three types of cut. In general, assume that there are q cuts of S . According to the above analysis, we can divide S into q sub-regions and any subring in S will be entirely within one of the sub-regions. This division is as follows:

- Step 1: if g_k is a cut of S of type 1, delete column 1 and column g_k in S ; furthermore, delete any columns satisfying the conditions in Lemma 5.4.
- Step 2: for each cut j of type 2, delete columns j and $j + 1$; furthermore, delete any columns satisfying the conditions in Lemma 5.5.
- Step 3: for each cut i of type 3, delete column i .
- Step 4: if neither column 1 nor column g_k is deleted, then we shift rotate S such that the sub-region including column 1 and column g_k are put together.

This division operation is illustrated in Figure 5.5.

Regarding to above division of S , we have the following observations:

- (1) : If $q = 0$, that is, there is no cut in S , then S remains normalized with respect to F_l and there are some wrap-around links in S . In this case, we will call S a region with wrap-around link(s).
- (2) : If $q > 0$, then none of the resulting sub-regions has wrap-around link; moreover, these sub-regions are not necessarily normalized any more. In this case, we will call each sub-region a region without wrap-around link.

Figure 5.5: Illustration of division of S

Notice that a sub-region is a portion of a region S . it contains faults of F_u and F_l in some consecutive columns; we call such a portion of a minimal blocking fault pattern a minimal blocking segment.

Clearly, any minimal blocking segment F satisfies Theorem 3.3.

In the rest of this Chapter, a region S (either with or without wrap-around link) will indicate a sub-region delimited by two minimal blocking segments, F_u and F_l . Clearly, $rd(S) \geq 2$ and $drd(S) \geq 2$.

For simplicity, through out this chapter, we will assume that for both cases, a region S spans over g_k columns and denote the row numbers of the faults of F_l and F_u in column i by x_i, x'_i , respectively, where $x_i > x'_i$.

Figure 5.6: Two through paths

5.3 Subrings with wrap-around link

In this section, we will focus on the regions with wrap-around link(s). Our goal is to construct a subring which spans all the columns in S and uses wrap-around link(s).

Definition 5.8 A through path of a region S is a live path π between two nodes P and Q in S , such that $P(y) = 1$, $Q(y) = g_k$, and there is no wrap-around link in π .

Example 5.2: In Figure 5.6, we show an examples of two through paths. In this example, we used a solid line and a dash line to show the two different through paths of S .

Observe that a line segment (if there is any) in S is a through path.

Theorem 5.1 Given a through path L of S , let $\Omega_j(L) = \{e \in L | e(y) = j\}$ be the set of all elements of L which are in column j for $1 \leq j \leq g_k$.

$$(1) : L = \cup_{j=1}^{g_k} \Omega_j(L);$$

$$(2) : \text{for all } j, \Omega_j \neq \emptyset.$$

Proof:

Identical to the proof of Theorem 3.1. □

Similar to the concept of *out-layer* for a line segment, we introduce the following terms for through path:

Definition 5.9 Given a through path L of S , let $t_j = \min_i \{(i, j) \in \Omega_j(L)\}$, and $b_j = \max_i \{(i, j) \in \Omega_j\}$, the top surface $TS(L)$ and the bottom surface $BS(L)$ of L are the sets $\{t_1, t_2, \dots, t_{g_k}\}$ and $\{b_1, b_2, \dots, b_{g_k}\}$, respectively.

Definition 5.10 Two through paths L_1 and L_2 are said to be non-crossing if $L_1 \cap L_2 = \emptyset$.

Given two through paths L_1 and L_2 , we say $L_1 > L_2$ if $TS(L_1) > BS(L_2)$. Clearly, if $L_1 > L_2$, then they are non-crossing.

Lemma 5.7 For any Region S , if $rd(S) \geq 2$ and $drd(S) \geq 2$, then S contains a through path L .

Proof:

The proof is by construction as follows:

Firstly, since $rd(S) \geq 2$, $(x_1 - 1, 1)$ is non-faulty in S ; thus, we can take $(x_1 - 1, 1)$ as the first node in the resulting through path.

Secondly, we will show that algorithm FB (Follow the border), presented in Section 4.3, can be applied to F_l and generates a through path in S . To show this, we have to show that if P is the current node of FB and P is not in column g_k , then at least one of its three neighbors – neighbor below, right neighbor, and neighbor above – must be connectable.

Let P be the current node of $FB(F_l)$ in column i . Consider node $R(P)$:

1. If $R(P)$ is connectable, then we can connect P to $R(P)$ in L .
2. If $R(P) \in F_l$, then both $A(P)$ and $R(A(P))$ are connectable, since $drd(S) \geq 2$ and $rd(S) \geq 2$. Thus, we can connect P to $A(P)$ and $R(A(P))$ in L .
3. If $R(P) \in F_u$, then both $B(P)$ and $R(B(P))$ are connectable, since $drd(S) \geq 2$ and $rd(S) \geq 2$. Thus, we can connect P to $B(P)$ and $R(B(P))$ in L .

Hence, S contains a through path L as desired. $\square$

By Lemma 5.7, there is at least one through path in S . Let $\mathcal{L}$ be the set of all the through paths of S . Within $\mathcal{L}$, there are two through paths which are especially interesting.

In the proof of Lemma 5.7, we have shown that FB ("follow the border") algorithm can be applied to F_l and generates a through path of S . We call this through path the lower surface $LS(S)$ of S .

Analogously, let P' and Q' be the faults of F_u in column 1 and g_k , respectively. We can show easily that the following scheme applies to F_u and generates a through path of S :

Starting from node $B(P')$, make the connections according to the following priority: the neighbor above, the right neighbor, the neighbor below; repeat this process until column g_k has been reached. We call this through path the upper surface of S , denoted by $US(S)$.

Similar to the properties for a line segment generated by FB , $LS(S)$ has the following properties:

Property 5.2 (1): for any column i , $|\Omega_i(LS(S))| = 1$ or 2 ;

(2): if $|\Omega_i(LS(S))| = 1$, let P be the only node of $LS(S)$ in column i , then $\{L(P), P, R(P)\} \subset LS(S)$;

(3): if $|\Omega_i(LS(S))| = 2$, let $P, B(P)$ be the two nodes of $LS(S)$ in column i , then either $\{L(P), P, B(P), R(B(P))\} \subset L$, or $\{L(B(P)), B(P), P, R(P)\} \subset LS(S)$.

(4): if $\{L(P), P, B(P), R(B(P))\} \subset LS(S)$, then both $L(B(P))$ and $B(B(P))$ are in F_l ;

(5): if $\{L(B(P)), B(P), P, R(P)\} \subset LS(S)$, then both $B(B(P))$ and $R(B(P))$ are in F_l .

Proof:

(1) : Similar to Property 4.2.

(2) , (3), (4), and (5): By Theorem 3.3 and the definition of FB .

□

Analogously, $US(S)$ has the following properties:

Property 5.3 *For any region S , the following results hold:*

(1): for any column i , $|\Omega_i(US(S))| = 1$ or 2 ;

(2): if $|\Omega_i(US(S))| = 1$, let P be the only node of $US(S)$ in column i , then $\{L(P), P, R(P)\} \subset US(S)$;

(3): if $|\Omega_i(US(S))| = 2$, let $P, B(P)$ be the two nodes of $US(S)$ in column i , then either $\{L(P), P, B(P), R(B(P))\} \subset US(S)$, or $\{L(B(P)), B(P), P, R(P)\} \subset US(S)$.

(4): if $\{L(P), P, B(P), R(B(P))\} \subset US(S)$, then both $A(P)$ and $R(P)$ are in F_u ;

(5): if $\{L(B(P)), B(P), P, R(P)\} \subset US(S)$, then both $L(P)$ and $A(P)$ are in F_u .

It is clear that if there are wrap-around links in S , then there is a subring in S , since $rd(S) \geq 2$ and $drd(S) \geq 2$.

Theorem 5.2 *For any region S with wrap-around links, if $rd(S) \geq 2$ and $drd(S) \geq 2$, then S contains a subring C' which spans all the columns of S .*

Proof:

By Lemma 5.7, S contains a through path L . Let P_1 and Q_1 be the two end nodes of L in column 1 and column g_k , respectively. Next, we will show that for any given wrap-around link (P, Q) in S , where P in column 1 and node Q in column g_k , there are always two nodes P' and Q' in L , where P' in column 1 and Q' in column g_k , such that there is a through path L' of S between P' and Q' ; furthermore, the

connections from P' to P and from Q' to Q in columns 1 and g_k do not contain any node of L' .

Due to the similarity, we will only consider the connection in column 1.

If $P_1 = P$, then take $P' = P_1$.

If $P_1 \neq P$, since both P_1 and P are non-faulty nodes in column 1 in S , there is a direct path π by connecting all the nodes in column 1 between them. If $\pi \cap (L - \{P\}) = \emptyset$, then take $P' = P_1$. Otherwise, let P' be the last node in L which is also contained in π .

Clearly, L' between P' and Q' satisfies the condition. $\square$

In the rest of this section, we will present an algorithm for forming a subring by using a wrap-around link in S .

First of all, recall from Section 5.2 that a region S with wrap-around link(s) has the following properties: S is normalized with respect to F_l ; $rd(S) \geq 2$, and $d_{rd}(S) \geq 2$; and $rd_{g_k}(S) \geq 3$. Let $(x_1, 1)$ and (x_1, g_k) be the two faults of F_l in columns 1 and g_k , respectively, then clearly, $(x_1 - 1, 1)$, $(x_1 - 1, g_k)$, and $(x_1 - 2, g_k)$ are non-faulty. Thus, the link between $(x_1 - 2, g_k)$ and $(x_1 - 1, 1)$ is always a wrap-around link in S .

Secondly, by the proof of Lemma 5.7, S always contains $LS(S)$ as a through path. Furthermore, by Property 4.4, $LS(S)$ will be in the following pattern: $\{(x_1 - 1, 1), (x_1 - 1, 2), \dots, (x_1 - 1, g_k - 1), (x_1 - 1, g_k)\}$. Thus, a simple subring in S can be formed by connecting $(x_1 - 2, g_k)$ to the both ends of $LS(S)$. We will refer this scheme by "Follow the Border and Wrap-up", or simply by FBW .

As for the size of the subring generated by FBW , we have the following result.

Theorem 5.3 *For any region S with wrap-around link, the size of the subring generated by FBW , $|C'(FBW)|$, is at most $2g_k + 1$.*

Proof:

Since we formed the subring $FBW(S)$ by connecting $(x_1 - 2, g_k)$ to both $(x_1 -$

1, 1) and $(x_1 - 1, g_k)$, thus, $|C'(FBW)| \leq |LS(S)| + 1$ by Property 5.2 (1), and the result follows. $\square$

Clearly, according to Theorem 5.3, regardless of the size of S , FBW will generate a subring of size at most $2g_k + 1$. That is, the larger $|S|$ is, the more FBW misses.

In the following, we present another subring formation algorithm, "Bouncing and Wrap-up", or simply BW , which is an improvement of FBW : it uses the bouncing operation developed in Chapter 4.

Algorithm BW:

INPUTS:

$$F_l = \{(x_1, 1), (x_2, 2), \dots, (x_{g_k}, g_k)\}$$

$$F_u = \{(x'_1, 1), (x'_2, 2), \dots, (x'_{g_k}, g_k)\}$$

OUTPUT: A subring C''

Begin

$$C'' \leftarrow \emptyset$$

$$\text{current-node } P \leftarrow (x_1 - 1, 1)$$

For $i = 1$ to $g_k - 2$ **do**

If i is odd **then**

$$x' \leftarrow \max\{x'_i, x'_{i+1}\}$$

 add $P, A(P), \dots, (x' + 1, i)$ to C''

$$P \leftarrow (x' + 1, i + 1)$$

elseif

$$x \leftarrow \min\{x_i, x_{i+1}\}$$

 add $P, B(P), \dots, (x - 1, i)$ to C''

$$P \leftarrow (x - 1, i + 1)$$

Endif

Endfor

If g_k is even **then**

$$x' \leftarrow \max\{x'_{g_k-1}, x'_{g_k}\}$$

 add $P, A(P), \dots, (x' + 1, g_k - 1)$ to C''

 add $(x' + 1, g_k), (x' + 2, g_k), \dots, (x_1 - 2, g_k)$ to C''

elseif

If $P(x) < x_1 - 3$ and $A(P), R(A(P))$ are connectable **then**

 add $P, A(P), R(A(P)), R(P), B(R(P)), B(P)$ to C''

$$P \leftarrow B(B(P))$$

Endif

If $P(x) < x_1 - 3$ and $R(P)$ is non-connectable **then**

 add P to C''

$$P \leftarrow B(P)$$

```

Endif
while  $P(x) < x_1 - 3$  then
    add  $P, R(P), B(R(P)), B(P)$  to  $C''$ 
     $P \leftarrow B(B(P))$ 
Endwhile
If  $P(x) = x_1 - 1$  then
    add  $P, B(P), B(B(P)), (x_1 - 1, g_k), (x_1 - 2, g_k)$  to  $C''$ 
else
    add  $P, B(P), (x_1 - 1, g_k), (x_1 - 2, g_k)$  to  $C''$ 
Endif
Endif
output  $C''$ 
End

```

The above algorithm can be briefly described in words as follows: It keeps the structure of FBW , and extends the connections between two adjacent columns by using the concept of bouncing.

The execution of this algorithm is best described in terms of a control token being passed along the nodes. The holder P of the token performs the following tasks, where i is its column number:

1. : If $1 \leq i \leq g_k - 2$, then:
 - (a) : If i is odd, then the node P keeps adding nodes above it to C'' until $(x' + 1, i)$ has been added, where $x' = \max\{x'_i, x'_{i+1}\}$; it then passes the control token to node $(x' + 1, i + 1)$.
 - (b) : If i is even, then P keeps adding nodes below it to C'' until $(x - 1, i)$ has been added, where $x = \min\{x_i, x_{i+1}\}$; it then passes the control token to node $(x - 1, i + 1)$.
2. : If $i = g_k - 1$, then:
 - (a) : If g_k is even, then P will make connections as in 1. until $(x_1 - 2, g_k)$ is added;
 - (b) : Otherwise, since $(x_1 - 2, g_k)$ is the last node in the subring, P tries to connect as many nodes above $(x_1 - 2, g_k)$ from column g_k as possible, while connecting nodes below it in column $g_k - 1$ as in 1.

Figure 5.7: A subring generated by BW

Initially, the token is held by node $(x_1 - 1, 1)$.

Example 5.3: In Figure 5.7, we show two subgrids generated by BW . In example (a), $g_k = 22$, and in example (b), $g_k = 23$.

Theorem 5.4 For any region S with wrap-around link, $|m(BW)| \leq \frac{3}{2}g_k + 2$.

Proof:

Let $u_i = \max\{x'_i, x'_{i+1}\}$, and $l_i = \min\{x_i, x_{i+1}\}$.

By the construction of $BW(S)$, there is at most one missed node, $(x'_1, 1)$; moreover, any missed node Q in column i , $2 \leq i \leq g_k - 2$, will satisfy the following condition: $Q(x) = u_i$ or $Q(x) = l_i$. Let us call a missed node Q in column i .

$2 \leq i \leq g_k - 2$, a missed node on the top (resp. on the bottom) if $Q(x) = u_i$ (resp. $Q(x) = l_i$).

By contradiction, suppose that there are more than 2 missed nodes in column i , $2 \leq i \leq g_k - 1$; then there are at least two missed nodes either on the top or on the bottom.

Assume there are two missed nodes Q_1 and Q_2 on the top in column i . Since $x'_i < Q_1(x)$ and $x'_i < Q_2(x)$, and by the definition of u_i , we have $x'_{i+1} > x'_i + 1$, contradicting the fact that F_u is a minimal blocking fault pattern.

Similarly, we can show that it is impossible to have more than 1 missed node on the bottom in column i . Thus, there are no more than 2 missed nodes in each of the columns $1, 2, \dots, g_k - 2$.

Suppose that there are exactly 2 missed nodes in column i , $2 \leq i \leq g_k - 2$; then we claim that there is at most 1 missed node in both columns $i - 1$ and $i + 1$. In fact, if i is even, then clearly there is no missed node on the top in column $i - 1$ and no missed node on the bottom in column $i + 1$; if i is odd, then there is no missed node on the bottom in column $i - 1$ and no missed node on the top in column $i + 1$. Thus, in the worst case, the total number of missed nodes for BW in columns $1, 2, \dots, g_k - 2$ is no more than $\frac{3}{2}(g_k - 2)$.

Consider now columns $g_k - 1$ and g_k . If g_k is even, then in the worst case BW can miss the following four nodes: $(x_{g_k-1} - 1, g_k - 1)$, $(x'_{g_k-1} + 1, g_k - 1)$, $(x'_{g_k} + 1, g_k)$, and $(x_1 - 1, g_k)$.

If g_k is odd, let $x' = \max\{x'_{g_k-2}, x'_{g_k-1}\}$. If $x' < x_1 - 3$, then in the worst case, BW can miss the following three nodes: $(x_{g_k-1} - 1, g_k - 1)$, $(x'_{g_k} + 1, g_k)$, and $(x_1 - 3, g_k)$. Otherwise, BW can miss at most the following five nodes: $(x_{g_k-1} - 1, g_k - 1)$, $(x'_{g_k-1} + 1, g_k - 1)$, $(x'_{g_k} + 1, g_k)$, $(x_1 - 4, g_k)$, and $(x_1 - 3, g_k)$.

Thus, $|m(BW)| \leq \frac{3}{2}(g_k - 2) + 5 = \frac{3}{2}g_k + 2$. □

5.4 Subrings without using wrap-around link

In this section, we will focus on the regions without wrap-around link(s). Our goal is to build a subring which spans all the columns in S and contains no wrap-around link. Since for a region with wrap-around link(s), we can still build subrings without using them, the schemes developed in this section applies to regions for both cases.

Lemma 5.8 *For any S , $TS(LS(S))$ (resp. $BS(US(S))$) is a minimal blocking segment.*

Proof:

Similar to Theorem 4.4. □

Lemma 5.9 *Given a region S , where $rd(S) \geq 3$, $drd(S) \geq 3$, and $sdrd(S) \geq 3$, let S' be the new region bounded by F_u and $TS(LS(S))$; then $rd(S') \geq 2$ and $drd(S') \geq 2$.*

Proof:

As shown in the proof of Lemma 5.7, S always contains the through path $LS(S)$. Furthermore, for any column i in S , by Property 5.2 (1), there are four possible cases depending on the combinations of $|\Omega_i(LS(S))|$ and $|\Omega_{i+1}(LS(S))|$. These cases are examined below:

(1) : $|\Omega_i(LS(S))| = 1$ and $|\Omega_{i+1}(LS(S))| = 1$.

Let P be the node of $LS(S)$ in column i . By Property 5.2 (2) $\{L(P), P, R(P), R(R(P))\} \subset LS(S)$. Thus, both $B(P)$ and $B(R(P))$ are non-connectable. If $B(P) \in F_l$, then $x'_i \leq P(x) - 2$ and $x'_{i+1} \leq P(x) - 2$, since $rd(S) \geq 3$ and $drd(S) \geq 3$. Thus, we have $rd_i(S') = P(x) - x'_i \geq 2$ and $rd_{i+1}(S') = P(x) - x'_{i+1} \geq 2$; furthermore, $drd_i(S') = \min\{P(x) - x'_{i+1}, P(x) - x'_i\} \geq 2$. If $B(P)$ is a singleton, then $B(R(P)) \in F_l$; similarly, we can show that $rd_i(S') \geq 2$, $rd_{i+1}(S') \geq 2$, and $drd_i(S') \geq 2$.

(2) : $|\Omega_i(LS(S))| = 1$ and $|\Omega_{i+1}(LS(S))| = 2$.

Let P be the node of $LS(S)$ in column i . In this case, there are two possible

Figure 5.8: The two possible patterns of $LS(S)$ for case (2)

Figure 5.9: The two possible patterns of $LS(S)$ for case (3)

connection patterns for $LS(S)$: (a) $\{L(P), P, R(P), A(R(P)), R(A(R(P)))\}$ and (b) $\{L(P), P, R(P), B(R(P)), R(B(R(P)))\}$, shown in Figure 5.8.

In case (a), by the definition of $LS(S)$, both $B(R(P))$ and $R(R(P))$ are in F_l . Since $sdrd(S) \geq 3$, we have $x'_i \leq P(x) - 3$ and $x'_{i+1} \leq P(x) - 3$. Thus, $rd_i(S') \geq 2$, $rd_{i+1}(S') \geq 2$, and $drc'_i(S') \geq 2$.

A similar argument will also work for case (b).

: $|\Omega_i(LS(S))| = 2$, and $|\Omega_{i+1}(LS(S))| = 1$.

Let P and $B(P)$ be the two nodes of $LS(S)$ in columns i . There are two possible patterns for $LS(S)$ for this case: $\{L(P), P, B(P), R(B(P))\}$ and $\{I(B(P)), B(P), P, R(P)\}$, as shown in Figure 5.9.

By a similar argument to the one for case (2), the result follows for this case.

: $|\Omega_i(LS(S))| = 2$ and $|\Omega_{i+1}(LS(S))| = 2$.

Similar to the proof of case (3).

□

Figure 5.10: Illustration for the case when $x_i - x'_{i+2} = 2$

Lemma 5.10 *There are two non-crossing through paths in S if and only if the following conditions hold: $rd(S) \geq 3$, $d_{rd}(S) \geq 3$, and $sdrd(S) \geq 3$.*

Proof:

The 'only if' part:

Let L_1 and L_2 be the two non-crossing through paths in S . Clearly, there are at least two non-faulty nodes on each column, that is, $rd(S) \geq 3$.

Let P_i and Q_i be the last nodes on column i for L_1 and L_2 , respectively. Then, P_i and Q_i will connect to their neighbors on column $i+1$. That is, $R(P_i) \in L_1$ and $R(Q_i) \in L_2$, which implies that $d_{rd,i}(S) \geq 3$ for $1 \leq i < g_k$; that is, $d_{rd}(S) \geq 3$.

Since $rd_i(S) \geq 3$ and $rd_{i-1}(S) \geq 3$, $x_{i+1} - x'_i \geq 3$ and $x_{i+1} - x'_{i+2} \geq 3$. Thus, $x_i - x'_{i+2} \geq x_{i+1} - 1 - x'_{i+2} \geq 2$. Similarly, $x_{i+2} - x'_i \geq x_{i+1} - 1 - x'_i \geq 2$. That is, $sdrd(S) \geq 2$.

Finally, to show that $sdrd(S) > 2$, assume to the contrary that $sdrd_i(S) = 2$ for some column i . Then either $x_i - x_{i+2} = 2$ or $x_{i+2} - x_i = 2$.

If $x_i - x'_{i+2} = 2$, since $x_i - x'_{i+2} \geq x_{i+1} - 1 - x'_{i+2} \geq 2$, we have $x_i = x_{i+1} - 1$. In this case, it is easy to see that any path from column i to column $i+2$ includes node $(x_i - 1, i+1)$, as shown in Figure 5.10. That is, there is at most one non-crossing through path in S , which is a contradiction.

If $x_{i+2} - x'_i = 2$, similarly, we can show that there is at most one through path in S contradicting the fact that both L_1 and L_2 are through paths of S .

Hence, $sdrd(S) \geq 3$.

the 'if' part:

Since $rd(S) \geq 3$, $drd(S) \geq 3$, and $sdrd(S) \geq 3$, by Lemma 5.7, $LS(S)$ is a through path in S . Furthermore, by Lemma 5.8, $TS(LS(S))$ is a minimal blocking segment; thus, F_u and $TS(LS(S))$ define a new region S' . By Lemma 5.9, $rd(S') \geq 2$ and $drd(S') \geq 2$. Thus, by applying Lemma 5.7 to S' , there is a through path $LS(S')$ in S' . Clearly, $LS(S')$ is also a through path in S which does not cross $LS(S)$. $\square$

Lemma 5.11 *There is a subring C' , which spans all the columns of S and does not contain any wrap-around link, if and only if there are two non-crossing through paths L_1 and L_2 in S .*

Proof:

The 'if' part:

Let L_1 and L_2 be two non-crossing through paths in S . Then, by an argument similar to the proof of Theorem 5.2, we can always form a subring by appropriately connecting the nodes of L_1 in columns 1 and g_k to the nodes of L_2 in columns 1 and g_k , respectively.

The 'only if' part:

If there is a subring C' in S , which spans all the columns and does not contain any wrap-around link, then there are at least two connectable nodes from each column in C' . Furthermore, since there is no wrap-around link in C' , there must be at least one connection within columns 1 and g_k in C' . That is, C' contains a connection from a node in column 1 to another node in column 1 as well as a connection from a node in column g_k to another node in column g_k .

Without loss of generality, let (P_1, P_2) and (Q_1, Q_2) be connections of C' in column 1 and in column g_k , respectively, according to a fixed orientation (say clockwise orientation). Now, subring C' can be partitioned into two linear arrays as follows:

L_1 from P_2 to Q_1 , L_2 from Q_2 to P_1 . Clearly they are two non-crossing through paths of S as desired. $\square$

Theorem 5.5 *There is a subring C' , which spans all the columns of S and does not contain any wrap-around link, if and only if the following conditions hold: $rd(S) \geq 3$, $drd(S) \geq 3$, and $sdrd(S) \geq 3$.*

Proof:

This is an immediate consequence of Lemma 5.10 and Lemma 5.11. $\square$

According to Theorem 5.5, if there is any column i , such that $rd_i(S) < 3$ or $drd_i(S) < 3$ or $sdrd_i(S) < 3$ in a region S , then there will be no subring in S can span over all columns of S without using wrap-around link. If any of these cases occurs, we have to cut the region by an operation similar to the one we used in Section 5.2.

In the rest of this section, without loss of generality, we assume that $rd(S) \geq 3$, $drd(S) \geq 3$, and $sdrd(S) \geq 3$ for S . In the following, we will develop an algorithm for forming a subring without using wrap-around links.

Recall that, in the proof of Lemma 5.10, we have shown that S always contains two non-crossing through paths $LS(S)$ and $LS(S')$. Furthermore, as shown in the proof of Theorem 5.11, we can form a subring by connecting the end points appropriately. We will call this scheme "Follow the Border Twice", or simply by *FBT*.

As for the size of the subring generated by *FBT*, we have the following result.

Theorem 5.6 *For any region S without wrap-around link, the size of the subring generated by *FBT*, $|C'(FBT)|$, is no larger than $4g_k$.*

Proof:

Since we formed the subring by connecting $LS(S)$ and $LS(S')$, $|C'(FBT)| = |LS(S)| + |LS(S')|$. By Property 5.2 (1), $|LS(S)| \leq 2g_k$ and $|LS(S')| \leq 2g_k$, thus the result follows. $\square$

Clearly, according to Theorem 5.6, regardless of the size of S , FBT will generate a subring of size at most $4g_k$. That is, the larger $|S|$ is, the more FBT misses.

In the following, we present another subring formation algorithm, "Follow the Border on the Bottom and Bouncing on the Top", or simply $FBBT$, which is an improvement of the FBT algorithm by combining the techniques of both BW and FBW . In the algorithm, we use the following notation: (t_i, i) is the node of $TS(LS(S))$ (the top surface of $LS(S)$) in column i ; $\bar{L}$ is the reverse ordered through path of L .

Algorithm FBBT:

INPUTS:

$$F_l = \{(x_1, 1), (x_2, 2), \dots, (x_{g_k}, g_k)\}$$

$$F_u = \{(x'_1, 1), (x'_2, 2), \dots, (x'_{g_k}, g_k)\}$$

OUTPUT: A subring C'

Begin

$$C' \leftarrow \emptyset$$

$$L \leftarrow FB(F_l)$$

$$P \leftarrow (x_1 - 1, 1)$$

If $t_1 = x_1 - 2$ **then**

$$L \leftarrow L - \{P\}$$

Endif

add $\bar{L}$ to C'

$$P \leftarrow (t_1 - 1, 1)$$

For $i = 1$ to $g_k - 2$ **do**

If i is odd **then**

$$x' \leftarrow \max\{x'_i, x'_{i+1}\}$$

add $P, A(P), \dots, (x' + 1, i)$ to C'

$$P \leftarrow (x' + 1, i + 1)$$

else

$$t \leftarrow \min\{t_i, t_{i+1}\}$$

add $P, B(P), \dots, (t - 1, i)$ to C'

$$P \leftarrow (t - 1, i + 1)$$

Endif

Endfor

If g_k is even **then**

$$x' \leftarrow \max\{x'_{g_k-1}, x'_{g_k}\}$$

add $P, A(P), \dots, (x' + 1, g_k - 1)$ to C'

add $(x' + 1, g_k), (x' + 2, g_k), \dots, (t_{g_k} - 1, g_k)$ to C'

else

```

If  $P(x) < t_{g_k} - 2$  and both  $A(P), R(A(P))$  are connectable then
    add  $P, A(P), B(A(P)), R(P), B(R(P)), B(P)$  to  $C''$ 
     $P \leftarrow B(B(P))$ 
Endif
If  $P(x) < t_{g_k} - 2$  and  $R(P)$  is non-connectable then
    add  $P$  to  $C''$ 
     $P \leftarrow B(P)$ 
Endif
while  $P(x) < t_{g_k} - 2$  then
    add  $P, R(P), B(R(P)), B(P)$  to  $C''$ 
     $P \leftarrow B(B(P))$ 
Endwhile
If  $P(x) = t_{g_k} - 2$  then
    add  $P, R(P), (t_{g_k} - 1, g_k)$  to  $C''$ 
else
    add  $P, (t_{g_k} - 1, g_k)$  to  $C''$ 
Endif
Endif
output  $C''$ 
End

```

The above algorithm can be briefly described as follows: a subring is formed by connecting two non-crossing through paths appropriately. One through path is $LS(S)$, the other one is the "extended version" of $LS(S')$. By extended version, we mean the through path resulting from applying a technique similar to the one of algorithm *BW*.

Example 5.3: In Figure 5.11, we show two subrings generated by *BW*. In example (a), $g_k = 22$, and in example (b), $g_k = 23$.

Theorem 5.7 For any region S with wrap-around link, $|m(FBBT)| \leq 2g_k + 3$.

Proof:

The missed nodes of *FBBT* can be divided into three cases:

- 1 : nodes missed when constructing $LS(S)$;
- 2 : nodes missed when constructing the extended version of $LS(S')$;
- 3 : nodes missed when connecting the two through paths.

(a)

(b)

Figure 5.11: A subring generated by *FBBT*

By the step 1 of the proof of Theorem 4.7, the set of missed nodes for case 1 is $SGT(F_l)$, which is at most $g_k/2$. An argument similar to the one used in the proof of Theorem 5.4 will show that the number of missed nodes for case 2 is at most $\frac{3}{2}g_k + 2$. Finally, for the connection of the two through paths, according to the algorithm, we may or may not lose $(x_1 - 1, 1)$. Thus, the result follows. $\square$

5.5 Chapter Summary

In this chapter, the reconfiguration problem was studied for regions S with height $H \leq 2$. By examining the necessary conditions for a column to be included in a subring, we derived a scheme to divide the region into subregions such that any subring will be entirely within one of the subregions. For each subregion, we developed subring formation algorithms for two cases: subregion with wrap-around link and subregion without wrap-around link. It was shown that the number of missed nodes for the algorithm using wrap-around link and the one without using wrap-around links are $\frac{3}{2}g_k + 2$ and $2g_k + 3$ in the worst case, respectively.

Chapter 6

Summary and Conclusions

6.1 Summary of Main Contributions

In the presence of $p \geq 2$ sets of blocking faulty patterns in a redundant ring network, the ring is broken into p connected components. Although the ring is disconnected, it is still possible to construct subrings in some or all connected components.

The main contribution of this thesis was the development of reconfiguration algorithms for redundant ring networks in the presence of several blocking fault patterns. The reconfiguration algorithms are semi-distributed algorithms; furthermore, they are efficient in the sense that they capture most of the non-faulty nodes.

The algorithms we have developed work for ring networks with just one bypass link in addition to the regular link. Thus, they can be applied to any chordal ring networks. These results extend the knowledge on the general reconfiguration problem for redundant ring networks: a knowledge which, until now, was restricted to very special case.

The contribution of the thesis can be summarized as follows:

1. : We have reviewed the basic fault tolerance terminologies and techniques, background knowledge on ring networks, fault tolerance principles employed in ring networks to make them fault-tolerant, existing results on catastrophic fault patterns in linear arrays, and existing reconfiguration results on ring networks.

2. : We have introduced a formal definition of a redundant ring network, the blocking fault patterns, and a Boolean matrix representation for ring networks. We have shown how to look at the problem of constructing a subring in a given ring in terms of finding a connected closed path in the corresponding Boolean matrix of the given ring.
3. : We have developed a set of algorithms for forming a subring in a given region S with at least three rows of non-faulty elements were developed. They all follow the same general strategy of first decomposing S into three parts (upper part, middle part, and lower part), then forming line segments in both upper and lower parts, and finally connecting the line segments with the help of the middle part to get a ring. The number of missed nodes for all the algorithms is $g_k + 1$ in the worst case.
4. : We have then studied the reconfiguration problem for regions S with fewer than three rows of non-faulty elements. By examining the necessary conditions for a column to be included in a subring, we derived a scheme to divide the region into subregions such that any subring will be entirely within one of the subregions. For each subregion, we developed subring formation algorithms for two cases: subregion with wrap-around link and subregion without wrap-around link. The number of missed nodes for the algorithm using wrap-around link and the one without using wrap-around links were no more than $\frac{3}{2}g_k + 2$ and $2g_k + 3$, respectively.

6.2 Open Problems

The following problems are still open:

- 1 : For any given region S , what is the largest possible subring within the region? In particular, are the algorithms developed in this thesis optimal in the sense that they form the largest possible subring? The first question is a particularly interesting one; we do have the following conjecture:

Conjecture 6.1 *For any given S , the problem of determining the largest possible subring in S is NP-hard.*

- 2 : We have assumed that the minimal blocking fault patterns forming the blocking decomposition of the set of faults are given. An interesting open problem is the one of developing an efficient algorithm to derive the minimal blocking decomposition given the set of blocking faults.

6.3 Areas of Future Work

Only the case when there are exactly p minimal blocking fault patterns presented in the ring was studied in this thesis. A future direction for research is to extend and generalize the results to allow for any blocking decomposition.

The results in this thesis assumed the minimal blocking fault patterns are non-crossing, that is, there is no common faulty nodes within any two patterns. In fact, it is possible that two blocking fault patterns cross each other. In this case, we can extend our work by looking at the areas delimited by various segments in the matrix. It should not be very difficult to extend the through path approach to solve this problem. However, this case still should be examined in details in the future.

Note that the results in this thesis are established for ring networks with link configuration $\{1, g_k\}$, and it serves as a lower bound for general link configuration $\{1, g_2, g_3, \dots, g_k\}$. A direction of future work is to study the subring construction problem for the general link configuration directly to obtain better results.

Glossary

Cluster

A sequence of consecutive non-faulty nodes in a row within a region bounded by two faulty nodes.

Blocking Decomposition

A decomposition of a blocking fault pattern into several smaller blocking fault patterns.

Blocking Fault Pattern

A fault pattern that blocks the information flow from one side of the faulty zone to the other side of the faulty zone for any fixed orientation.

Bouncing

An operation on the Boolean matrix form of a ring, which connects non-faulty nodes in a repeatedly (up, right, down, right) fashion between the two ends of a cluster.

Bypass link

A link from a node to another node at distance larger than 1; it is only used to bypass failed nodes.

Line Segment

A live path in which the two end nodes are in column 1 and column g_k , and they are located in the same row. Furthermore, all the nodes within the line segment is either below or above the direct line between the two end nodes.

Link Configuration

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

Live Path

A path between two non-faulty nodes in which every node is non-faulty.

Lower Surface

A set of nodes including one node in each column which is the one with the largest row coordinate in that column in a given through path.

Minimal Blocking Fault Pattern

A blocking fault pattern with exactly g_k faults, where g_k is the distance by the longest bypass link.

Minimal Blocking Segment

A portion a minimal blocking fault pattern between two columns.

Missed Node

A non-faulty node that is not included in the resulting subring of an algorithm.

Normalization

A region such that the first fault of one of the two minimal blocking fault patterns delimiting it is in column 1.

Fault Pattern

An ordered set that specifies the distribution of the faulty nodes.

Follow the Border

A line segment generation algorithm in which the non-faulty nodes are connected according to the following priority: down, right and up.

Out-Layer

A set of nodes including one node in each column which is the node with the smallest row coordinate in the column within the given line segment.

Region

A connected component in a given region delimited by two minimal blocking fault patterns.

Singleton

A non-faulty node that has degree 1 once the connections to the faulty nodes have been removed.

Through Path

A live path between two nodes in column 1 and column g_k in a given region.

Top Surface

A set of nodes including one node in each column which is the node with the largest row coordinate in the column within the given line segment.

Wrap-around Link

A link that connects a node in column g_k to a node in column 1.

References

- 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.
- B. W. Arden and H. Lee, "Analysis of Chordal Ring", *IEEE Trans. on Computers*, Vol. C-30, No. 4, April 1981, pp. 291-295.
- 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.
- K. P. Belkhale and P. Banerjee, "Reconfiguration Strategies in VLSI Processor Arrays", *Proc. Int'l Conf. on Computer Design*, Rye Brook, NY, 1988, pp. 418-421.
- M. Chean and J. A. B. Fortes, "A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays", *IEEE Computer*, Vol. 23, No. 1, Jan. 1990, pp. 55-69.
- I. N. Dallas, and E. B. Spratt (eds), *Ring Technology Local Area Networks*, North Holland, 1984.
- R. De Prisco and A. Monti, "On Reconfigurability of VLSI Linear Arrays", *Proc. 3rd Workshop on Algorithms and Data Structures*, Montreal, Aug. 1993. In *Lecture Notes in Computer Science*, Vol. 709, Springer-Verlag, 1993, pp. 553-564.

- [8] 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.
- [9] G. E. Farr and H. Schröder, "An Evaluation of Reconfiguration Algorithms in Fault Tolerant Processor Arrays", *Proc. 5th MIT Conf. on VLSI*, Cambridge, Mass., March 1988, MIT Press, 1988, pp. 131-148.
- [10] E. Fujiwara and D. K. Pradhan, "Error-Control Coding in Computers", *IEEE Computers*, Vol. 23, No. 7, July 1990, pp. 63-72.
- [11] 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.
- [12] J. H. Kim, "On the Design of Easily Testable and Reconfigurable Systolic Arrays", *Proc. Int'l Conf. on Systolic Arrays*, San Diego, Calif., 1988, pp. 505-513.
- [13] J. H. Kim and S. M. Reddy, "A Fault-Tolerant Systolic Array Design Using FMR Method", *Proc. Int'l Conf. on Computer Design*, Port Chester, NY., Oct. 1985, pp. 769-773.
- [14] 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.
- [15] J. H. Hwang and C. S. Raghavendra, "VLSI Implementation of Fault-Tolerant Systolic Arrays", *Proc. Int'l Conf. on Computer Design*, Port Chester, NY., Oct. 1986, pp. 110-113.
- [16] I. Koren and I. Pomeranz, "Distributed Structuring of Processor Arrays in the Presence of Faulty Processors", *Proc. Int'l Conf. on Systolic Arrays*, Oxford, July 1986, pp. 239-248.

- [17] R. H. Kuhn, "Interstitial Fault Tolerance — A Technique for Masking Systolic Arrays Fault Tolerant", *Proc. 16th Annual Hawaii Int'l Conf. on System Sciences*, Hawaii, 1983, pp. 215-224.
- [18] H. T. Kung, "Why Systolic Architecture?", *IEEE Computer*, Vol. 15, No. 1, Jan. 1982, pp. 37-46.
- [19] 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.
- [20] S. Y. Kung, C. W. Chang, and C. W. Jen, "Fault-Tolerance Design in Real-Time VLSI Array Processors", *Proc. Conf. on Computers and Communications*, Phoenix, 1987, pp. 110-115.
- [21] S. Kuo and W. K. Fuchs, "Efficient Spare Allocation for Reconfigurable Arrays", *IEEE Design and Test*, Feb. 1987, pp. 24-31.
- [22] 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.
- [23] 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.
- [24] M. T. Liu, "Distributed Loop Computer Networks", *Advances in Computers*, Vol. 17, Academic Press, 1978, pp. 163-221.
- [25] T. E. Mangir, "Sources of Failures and Yield Improvement for VLSI and Restructurable 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.
- [26] T. E. Mangir and C. S. Raghavendra, "Issues in the Implementation of Fault-Tolerant VLSI and WSI Systems", *Proc. Int'l Conf. on Computer Design*, Port Chester, Oct. 1984, pp. 95-100.

- [27] 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.
- [28] 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.
- [29] A. Nayak, "On Reconfigurability of Some Regular Architectures", Ph.D. Thesis, Carleton University. 1991.
- [30] A. Nayak, L. Pagli, and N. Santoro, "Efficient Construction of Catastrophic Patterns for VLSI Reconfigurable Arrays", *Integration - the VLSI journal*, Vol. 15, No. 2, Oct. 1993, pp. 133-150.
- [31] A. Nayak, L. Pagli, and N. Santoro, "Recognition of Catastrophic Faults", *Proc. Int'l Workshop on Defect and Fault Tolerance in VLSI Systems*, Dallas, 1992, pp. 70-79.
- [32] A. Nayak and N. Santoro, "Fault-Tolerance Analysis of Redundant Ring Networks", *Proc. Int'l Conf. on Parallel Computing*, London, 1991; in *Advances in Parallel Computing*, Vol. 4, North-Holland, 1992, pp. 311-322.
- [33] A. Nayak and N. Santoro, "Limits to Reconfigurability of Loop Topologies", *Proc. Int'l Symp. on Parallel and Distributed Computing in Engineering Systems*, Corfu, 1991; in *Parallel and Distributed Computing in Engineering Systems* (S. Tzafestas, P. Borne, L. Grandinetti eds.), North-Holland, 1992, pp. 453-458.
- [34] A. Nayak and N. Santoro, "Bounds on Performance of VLSI Processor Arrays", *Proc. 5th Int'l Parallel Processing Symposium*, Anaheim, California, May 1991, pp. 453-458.

- [35] A. Nayak, N. Santoro and R. Tan, "Fault-Intolerance of Reconfigurable Systolic Arrays", *Proc. 20th Int'l Conf. on Fault-Tolerant Computers*, Newcastle upon Tyne, June 1990, pp. 202-209.
- [36] R. Negrini, M. Sami, and R. Stefanelli, *Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays*, The MIT Press, 1989.
- [37] S. Pateras and J. Rajski, "A Self-Reconfiguration Scheme for Fault-Tolerant VLSI Processor Arrays", *Proc. Int'l Conf. on Computer Design*, Rye Brook, NY., 1988, pp. 560-563.
- [38] D. K. Pradhan, Ed., *Fault-Tolerant Computing, Theory and Techniques*, Vol. 1 and 2, Prentice-Hall, 1986.
- [39] C. S. Raghavendra, "Fault Tolerance in Regular Network Architectures", *IEEE Micro*, Vol. 4, No. 6, Dec. 1984, pp. 44-53.
- [40] C. S. Raghavendra and M. Gerla, "Optimal Loop Topologies for Distributed Systems", *Proc. 7th Data Communication Symp.*, Mexico City, Oct. 1981, pp. 218-223.
- [41] 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.
- [42] C. S. Raghavendra, M. Gerla, and D. S. Parker, "Multi-Connected Loop Topologies for Local Computer Networks", *Proc. INFOCOM 82*, Las Vegas, March 1982, pp. 184-190.
- [43] 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.
- [44] J. Ren, "Geometric Characterizations of Fault Patterns in Linear Systolic Arrays", M.C.S. Thesis, Carleton University, Jan. 1994.

- [45] D. A. Rennels, "Fault-Tolerant Computing — Concepts and Examples", *IEEE Trans. on Computers*, Vol. C-33, No. 12, Dec. 1984, pp. 1116-1129.
- [46] A. L. Rosenberg, "Graph Theoretical Approach to Fault-Tolerant Processor Arrays", *Int'l Workshop on Wafer Scale Integration*, Southampton, 1985, pp. 1-16.
- [47] D. P. Siewiorek and R. S. Swartz, *The Theory and Practice of Reliable System Design*, Digital Press, 1982.
- [48] 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.
- [49] 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.
- [50] H. Y. Youn and A. D. Singh, "A Highly Efficient Design for Reconfiguring the Processor Array in VLSI", *Proc. Int'l Conf. on Parallel Processing*, St. Charles, Aug. 1988, Vol. 1, pp. 375-382.

END

0 8-1 1-94

FIN