TURUN YLIOPISTON JULKAISUJA
ANNALES UNIVERSITATIS TURKUENSIS

SARJA - SER. A I OSA - TOM. 465

ASTRONOMICA - CHEMICA - PHYSICA - MATHEMATICA

ADAPTIVE ROUTING APPROACHES FOR NETWORKED MANY-CORE SYSTEMS

by

Masoumeh Ebrahimi

TURUN YLIOPISTO
UNIVERSITY OF TURKU
Turku 2013

From the Laboratory of Embedded Computer and Electronic Systems, ECES
Department of Information Technology
University of Turku
Turku, Finland
&
Graduate School in Electronics, Telecommunications and Automation, GETA
Aalto University
Helsinki, Finland

Supervisors

Professor Hannu Tenhunen
Associate Professor Juha Plosila
Associate Professor Pasi Liljeberg
Department of Information Technology
University of Turku
Turku, Finland

Reviewers

Professor Nader Bagherzadeh
Department of Electrical Engineering and Computer Science
University of California
Irvine, US

Professor Thomas Hollstein
Department of Computer Engineering
Tallin University of Technology
Tallin, Estonia

Opponent

Professor Paolo Ernesto Prinetto
Department of Control and Computer Engineering
Polytechnic University of Turin
Turin, Italy

ISBN 978-951-29-5453-7
ISSN 0082-7002
Painosalama Oy-Turku, Finland 2012

Abstract

Through advances in technology, System-on-Chip design is moving towards integrating tens to hundreds of intellectual property blocks into a single chip. In such a many-core system, on-chip communication becomes a performance bottleneck for high performance designs. Network-on-Chip (NoC) has emerged as a viable solution for the communication challenges in highly complex chips. The NoC architecture paradigm, based on a modular packet-switched mechanism, can address many of the on-chip communication challenges such as wiring complexity, communication latency, and bandwidth. Furthermore, the combined benefits of 3D IC and NoC schemes provide the possibility of designing a high performance system in a limited chip area. The major advantages of 3D NoCs are the considerable reductions in average latency and power consumption.

There are several factors degrading the performance of NoCs. In this thesis, we investigate three main performance-limiting factors: network congestion, faults, and the lack of efficient multicast support. We address these issues by the means of routing algorithms.

Congestion of data packets may lead to increased network latency and power consumption. Thus, we propose three different approaches for alleviating such congestion in the network. The first approach is based on measuring the congestion information in different regions of the network, distributing the information over the network, and utilizing this information when making a routing decision. The second approach employs a learning method to dynamically find the less congested routes according to the underlying traffic. The third approach is based on a fuzzy-logic technique to perform better routing decisions when traffic information of different routes is available.

Faults affect performance significantly, as then packets should take longer paths in order to be routed around the faults, which in turn increases congestion around the faulty regions. We propose four methods to tolerate faults at the link and switch level by using only the shortest paths as long as such path exists. The unique characteristic among these methods is the toleration of faults while also maintaining the performance of NoCs. To the best of our knowledge, these algorithms are the first approaches to bypassing faults prior to reaching them while avoiding unnecessary misrouting of packets.

Current implementations of multicast communication result in a significant performance loss for unicast traffic. This is due to the fact that the routing rules of multicast packets limit the adaptivity of unicast packets. We present an approach in which both unicast and multicast packets can be efficiently routed within the network. While suggesting a more efficient multicast support, the proposed approach does not affect the performance of unicast routing at all. In addition, in order to reduce the overall path length of multicast packets, we present several partitioning methods along with their analytical models for latency measurement. This approach is discussed in the context of 3D mesh networks.

Acknowledgements

It has been a great privilege to make my doctoral studies at the Department of Information Technology, university of Turku, Finland since March 2009. I would like to express my deepest appreciation to all those who have made some contributions in this way. The valuable experience that I have gained in this period is tightly connected to all people around me as my wonderful supervisors, helpful staff, friendly colleagues, and my lovely husband-colleague.

Special gratitude I give to my supervisors, Professor Hannu Tenhunen, Associate professor Juha Plosila, and Associate professor Pasi Liljeberg for their excellent guidance, continuous support, patience against any issues, and providing me with an excellent atmosphere for doing research. There are much more appreciations but I fall short of words. I would really mean by saying that I could have ever imagined having better supervisors than them! Beside my supervisors, I would also like to acknowledge the crucial role of Doctor Masoud Daneshtalab for his valuable comments, advice, and insight throughout my work. This thesis would not have been possible without his stimulating suggestions and encouragement throughout these years.

I wish to thank the pre-examiners, Professor Nader Bagherzadeh and Professor Thomas Hollstein for their constructive comments and suggestions on the thesis. Special thanks go to Professor Paolo Prinetto for accepting to act as my opponent.

The research was mainly funded by the graduate school in Electronics, Telecommunications, and Automation (GETA). I would like to thank GETA and its staff for the financial, academic, and technical support. In addition, I greatly appreciate the financial support by Nokia and Elisa Foundation. Furthermore, I would like to acknowledge the Business and Innovation Development (BID) and European Institute of Innovation and Technology (EIT ICT Labs) for their support to follow the MBA program along with the doctoral studies and being involved in ICT innovation and entrepreneurship activities.

I have had an opportunely to do a research visit at the University of Toronto, Canada. I wish to thank Professor Natalie Jerger for the invitation and valuable discussions.

I wish to thank my parents and family for their continuous love and inspiration. I could say the most difficult part of this thesis was to be far from them! They are the most valuable things to me. I wish I could express how much I love and appreciate them.

I would like to thank Masoud again, but this time as my beloved one. He has supported me throughout the entire process not only by his academic guidance and encouragement but also spiritually by his pure love! Masoud is the one who made my dreams reality and constantly giving me a wonderful feeling of unlimited love!

Last but not the least, I want to thank my lovely friends Anne-Marie Tuikka and Fahime Farahnakian for having delightful coffee breaks, helping me to refresh my mind!

Contents

1 Introduction 1

1.1 Thesis Contributions 2
1.1.1 Congestion-aware Techniques 3
1.1.2 Fault-Tolerant Techniques 4
1.1.3 Collective Communication 5

1.2 Thesis Organization 6

2 Networks-on-Chip 11

2.1 Networks-on-Chip Characteristics 11
2.1.1 Network Dimension 11
2.1.2 Network Topology 12
2.1.3 Switch Architecture 13
2.1.4 Virtual Channel 13
2.1.5 Switching Techniques 14
2.1.6 Flow Control Mechanisms 14

2.2 Routing Algorithms 15
2.2.1 Deterministic and Adaptive Routing 15
2.2.2 Minimal and Non-minimal Routing 15
2.2.3 Congestion-aware and Congestion-oblivious Routing 16
2.2.4 Unicast and Multicast Routing 16
2.2.5 Fault-Tolerant Routing 17
2.2.6 Starvation, Deadlock, and Livelock 17
2.2.7 Turn Models 18

3 Congestion-Aware Routing Algorithms for a 2D Mesh Network 19

3.1 Traditional Approaches 19
3.1.1 Dynamic XY (DyXY) 20
3.1.2 Enhanced Dynamic XY (EDXY) 20
3.1.3 Neighbor-on-Path (NoP) 21
3.1.4 Regional Congestion Awareness (RCA) 22
3.1.5 Destination-Based Adaptive Routing (DBAR) 23
3.1.6 Q-Learning Approach with Reduced Table Sizes (C-Routing) 24
3.1.7 Summary of Traditional Methods 25

3.2 The Proposed Cluster-based Approaches 25
3.2.1 Agent-based Routing Algorithm (AgRA) 26

3.2.2 Trapezoid-based Routing Algorithm (TRA) 29
3.2.3 Results and Discussion 40
3.3 The Proposed Non-Minimal and Learning-based Approaches 43
3.3.1 Highly Adaptive Non-Minimal Routing Algorithm (HARA) 44
3.3.2 Q-Learning-based Approach using HARA (HARAQ) 50
3.3.3 Results and Discussion 54
3.4 The Proposed Fuzzy-based Approach 57
3.4.1 Non-Fuzzy Routing Algorithm (NFRA) 58
3.4.2 Fuzzy-based Routing Algorithm (FRA) 60
3.4.3 Results and Discussion 69
3.5 Summary of the Proposed Methods 72
4 Fault-Tolerant Routing Algorithms for a 2D Mesh Network 75
4.1 Traditional Approaches 76
4.1.1 A Ring-based Fault-Tolerant Routing (Extended X-Y) 76
4.1.2 Reconfigurable Routing for Tolerating Faulty Switches (ReRS) 77
4.1.3 Reconfigurable Routing for Tolerating Faulty Links (RAFT) 78
4.1.4 Bidirectional Fault-Tolerant NoC (BFT-NoC) 78
4.1.5 Summary of Traditional Methods 78
4.2 The Proposed Approaches for Tolerating Faulty Links 79
4.2.1 Any Single Faulty Link (MD) 79
4.2.2 Multiple Faulty Links (MAFA) 86
4.2.3 Results and Discussion 96
4.3 The Proposed Approaches for Tolerating Faulty Switches 99
4.3.1 Any Single Faulty Switch (HiPFaR) 99
4.3.2 Multiple Faulty Switches (MiCoF) 105
4.3.3 Results and Discussion 112
4.4 Summary of the Proposed Methods 116
5 Unicast and Multicast Routing Algorithms for a 3D Mesh Network 119
5.1 Traditional Approaches 120
5.1.1 Virtual Circuit Tree Multicasting (VCTM) 120
5.1.2 Dual-Path Multicast Routing (DP) 121
5.1.3 Multi-Path Multicast Routing (MP) 123
5.1.4 Hamiltonian Adaptive Multicast and Unicast Model (HAMUM) 124
5.1.5 Summary of Traditional Methods 124
5.2 The Proposed Partitioning Methods for a 3D Mesh Network 125
5.2.1 Hamiltonian Path in a 3D Mesh Network 125
5.2.2 Two-Block Partitioning Method (TBP) 126
5.2.3 Vertical Block Partitioning (VBP) 130

5.2.4 Recursive Partitioning (RP) 133
5.3 The Proposed Adaptive Routing Algorithm 136
5.3.1 Minimal Adaptive Routing (MAR) 136
5.3.2 Deadlock Avoidance 139
5.4 Results and Discussion 140
5.4.1 Analytical Results 140
5.4.2 Simulation Results 143
5.5 Summary of the Proposed Methods 150
6 Conclusion 151

List of Figures

1.1: (a) Bus (b) Network-on-Chip 1
2.1: (a) 2D NoC (b) 3D NoC 12
2.2: (a) Mesh topology (b) Torus topology 12
2.3: A switch architecture 13
2.4: A switch in (a) XY (b) double-XY (c) double-Y network 14
2.5: Multicast approaches 17
2.6: (a) Clockwise and counter-clockwise turns (b) XY routing (c) Negative-First (d) West-First (e) North-Last (Solid lines indicate the allowable turns and dash lines indicate the prohibited turns) 18
3.1: An example of the DyXY method 20
3.2: An example of the EDXY method 21
3.3: An example of the NoP method 22
3.4: An example of the RCA method 23
3.5: An example of the DBAR method 24
3.6: An example of the learning-based approach 25
3.7: Agent-based Networks-on-Chip 27
3.8: The congestion-aware selection algorithm in the agent-based approach 28
3.9: An example of the agent-based selection method 29
3.10: Passing-probability of packets through intermediate switches 31
3.11: The output selection function of TRA 32
3.12: The congestion information of the highlighted regions is needed at the switch 24 to perform the output selection function of TRA 33
3.13: Input selection function of TRA 34
3.14: The congestion information of the highlighted regions is needed at the switch 24 to perform the input selection function of TRA 34
3.15: Assigning a name to each switch 35
3.16: Assigning congestion values to packets 36
3.17: The pseudo code of the priority-based input selection function of TRA 36
3.18: A general example of combining the input and output selection functions of TRA 38
3.19: The required information at the switch 24 to perform the input and output selection functions 38
3.20: Information from (a) west- (b) east- (c) north-, and (d) south-ward regions to the switch 24 39
3.21: The required number of bits to propagate the congestion information 40
3.22: Performance analysis in an 8×8 mesh network under uniform traffic profile 41
3.23: Performance analysis in an 8×8 mesh network under hotspot traffic profile with H=10% 41
3.24: Performance analysis under different application benchmarks normalized to NoP 43

3.25: (a) A switch in a double-Y network (b) 0-degree-ch (c) 0-degree-vc (d) 90-degree (e) 180-degree-vc (f) 180-degree-ch 45
3.26: (a) 90-degree turns in vc1 (b) 90-degree turns in vc2 (c) 0-degree-ch (d) 0-degree-vc 45
3.27: Channel numbering in the Mad-y method 46
3.28: Allowable 180-degree turns in the HARA method 47
3.29: The numbering mechanism of HARA 47
3.30: All eligible turns in HARA 47
3.31: Determining all eligible output channels by HARA 49
3.32: An example of HARA 50
3.33: The process of updating the Q-Tables 52
3.34: Performance analysis in an 8×8 mesh network under the uniform traffic profile 55
3.35: Performance analysis in an 8×8 mesh network under hotspot traffic profile with H=10% 56
3.36: Performance analysis under different application benchmarks normalized to DBAR 56
3.37: Non-optimal decision by the DyXY routing algorithm 58
3.38: The pseudo code of NFRA 60
3.39: Two examples of non-optimal routing decisions in NFRA 60
3.40: (a) General fuzzy system (b) Fuzzy routing algorithm 61
3.41: An example of triangular membership function 62
3.42: (a) OccupiedSlots_Input (b) OccupiedSlots_Switch (c) Cost membership functions 63
3.43: (a) OccupiedSlots_Input (b) OccupiedSlots_Switch as a part of two membership functions 65
3.44: Cost for (a) rule1 (b) rule2 (c) rule3 (d) rule4 66
3.45: Composition of the Cost membership function of all rules 66
3.46: The degree of membership function for the input parameters at the switch 9 67
3.47: The degree of membership function for the input parameters at the switch 6 68
3.48: Performance analysis in an 8×8 mesh network under the uniform traffic profile 70
3.49: Performance analysis in an 8×8 mesh network under hotspot traffic profile with H=10% 70
3.50: Simulation results under two multimedia traffic profiles: MPEG and VOPD 71
3.51: (a) VOPD block diagram, with communication BW annotated (in MB/s) (b) its mapping onto a mesh topology [19] 71
3.52: (a) MPEG4 decoder block diagram, with communication BW annotated (in MB/s) (b) its mapping onto a mesh topology [19] 72
4.1: (a) Two examples of the Extended X-Y routing algorithm (b) The required fault information 77
4.2: (a) Two examples of the ReRS routing algorithm (b) The required fault information . 77
4.3: (a) Two examples of the RAFT routing algorithm (b) The required fault information 78
4.4: (a) Statuses of twelve links are needed by RAFT (b) The statuses of eight links are needed by MD 79
4.5: Permitted and prohibited turns of MD similar to Mad-y 80
4.6: The numbering mechanism of MD similar to Mad-y 80

4.7: Bypassing faulty links when the destination is located in the northeast position of the source switch (Note that numbers determine the priority of selecting among different routes) 82
4.8: Bypassing faulty links when the destination is located in the (a) east (b) west (c) north (d) south positions of the source switch 82
4.9: Northward and southward packets are strictly belonging to the second virtual channel (a) and (b) show the cases where the fault does not occur in the left borderline (c) and (d) show how the fault can be tolerated when the fault occur in the left borderline 83
4.10: MD routing algorithm..... 84
4.11: Alternative paths from the source switch S to the destination D..... 85
4.12: The first fault occurs in (a) borderline link (b) central link 86
4.13: Fault distribution mechanism..... 87
4.14: (a) west-last (b) east-last (c) all permitted transactions between vc1 and vc2..... 88
4.15: Tolerating one faulty link by MAFA when the destination is in the (a) northeast (b) southeast (c) northwest (d) southwest of the current switch 89
4.16: Tolerating one faulty link by MAFA when the destination is in the (a) north (b) south (c) east (d) west positions of the current switch..... 89
4.17: Priority of selecting among different routes when the destination is in the (a) northeast (b) southeast (c) northwest (d) southwest positions of the current switch 90
4.18: Different positions of two faulty links for a northeast packet 91
4.19: Different positions of two faulty links for a southwest packet..... 91
4.20: Priority of different routes for (a) northward (b) southward (c) eastward (d) westward packets 92
4.21: MAFA routing algorithm for northeast, northwest, southeast, and southwest packets 93
4.22: MAFA routing algorithm for north-, south-, east-, and west-ward packets..... 94
4.23: Examining all possible paths by Enhanced MAFA..... 95
4.24: Non-minimal choices offered by Enhanced-MAFA 95
4.25: Performance analysis of MD and RAFT in an 8×8 mesh network under uniform traffic profile 96
4.26: Performance analysis of MAFA and RAFT in an 8×8 mesh network under uniform traffic profile 97
4.27: Performance analysis of MD and RAFT in an 8×8 mesh network under hotspot traffic profile..... 97
4.28: Performance analysis of MAFA and RAFT in an 8×8 mesh network under hotspot traffic profile 98
4.29: Reliability evaluation of Enhanced-MAFA in a 6×6 mesh network under uniform traffic profile 98
4.30: The statues of four neighboring switches are required by the HiPFaR routing algorithm..... 100
4.31: Permitted and prohibited turns of MD similar to Mad-y [10] 100
4.32: The basic rules for selecting among the neighboring switches when a packet gets close to the destination switch 102
4.33: Different positions of current, destination and a faulty switch..... 102
4.34: Tolerating a faulty switch by (a) eastward packets (b) westward packets..... 103

4.35: Tolerating a faulty switch for northward and southward packets 103
4.36: HiPFaR routing algorithm 104
4.37: Switch architecture using MiCoF 106
4.38: Five fautly switches in a 4×4 mesh topology (b) the resulted network using MiCoF 106
4.39: Tolerating any single faulty switch using only the shortest paths 107
4.40: Tolerating two faulty switches by the MiCoF approach 109
4.41: A couple indicates a diagonal position 110
4.42: Three faulty switches in the network, which are located close to each other 111
4.43: MiCoF routing algorithm 112
4.44: Performance analysis of HiPFaR and ReRS in an 8×8 mesh network under uniform traffic profile 113
4.45: Performance analysis of MiCoF and ReRS in an 8×8 mesh network under uniform traffic profile 113
4.46: Performance analysis of HiPFaR and ReRS in an 8×8 mesh network under hotspot traffic profile 114
4.47: Performance analysis of MiCoF and ReRS in an 8×8 mesh network under hotspot traffic profile 114
4.48: Reliability measurement based on the first metric 115
4.49: Reliability measurement based on the second metric 115
5.1: (a) VCTM (b) the number of hops can be reduced 121
5.2: (a) A physical channel (b) high channel subnetwork (c) low channel subnetwork 122
5.3: Dual-path multicast approach 123
5.4: Multi-path multicast approach 123
5.5. An example of HAMUM for (a) unicast packets (b) multicast packets 124
5.6: (a) A 3×3×3 mesh network with the label assignment (b) high channel (c) low channel subnetworks. The solid lines indicate the Hamiltonian path and dashed lines indicate the links that could be used to reduce the path length 126
5.7: The TBP method (a) balanced (b) unbalanced partitions 127
5.8: The pseudo code of the TBP method 127
5.9: Measuring MML for the TBP method 129
5.10: The pseudo code of the VBP method 130
5.11: The VBP method (a) balanced partitions (b) unbalanced partitions 131
5.12: Measuring MML for the TBP method 132
5.13: The pseudo code of the RP method 134
5.14: RP when the source switch is at (a) the switch 26 (b) the switch 7 134
5.15: The pseudo code of the MAR algorithm 137
5.16: (a) An example of the MAR algorithm for a unicast packet (b) showing all possible paths between the source 6 and destination 48 (c) An example of the MAR algorithm for a multicast packet 138
5.17: Packet length can affect the startup latency 141
5.18: Performance analysis in a 4×4×3 mesh network using deterministic routing with 8 destinations 144
5.19: Performance analysis in a 4×4×3 mesh network using deterministic routing with 16 destinations 144

5.20: Performance analysis in a 4×4×3 mesh network using adaptive routing with 8 destinations 145
5.21: Performance analysis in a 4×4×3 mesh network using adaptive routing with 16 destinations 145
5.22: Performance analysis in a 4×4×3 mesh network using deterministic routing with 8 destinations under mixed traffic (30% multicast and 70% unicast); Unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and H=10%146
5.23: Performance analysis in a 4×4×3 mesh network using deterministic routing with 16 destinations under mixed traffic (30% multicast and 70% unicast); Unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and H=10%146
5.24: Performance analysis in a 4×4×3 mesh network using adaptive routing with 8 destinations under mixed traffic (30% multicast and 70% unicast); Unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and H=10%146
5.25: Performance analysis in a 4×4×3 mesh network using adaptive routing with 16 destinations under mixed traffic (30% multicast and 70% unicast); Unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and H=10%147
5.26: Performance analysis under different application benchmarks normalized to TBP . 148
5.27: Average power dissipation results in a 4×4×3 mesh network under multicast traffic profile 149

List of Tables

3.1: System configuration parameters 42
3.2: Hardware implementation details 43
3.3: Potential output channels offered by HARA 49
3.4: Potential output channels offered by Mad-y 49
3.5: The area overhead 53
3.6: Hardware implementation details 57
3.7: FRA inference rules 64
3.8: The cost membership function of the switch 9 68
3.9: The cost membership function of the switch 6 68
3.10: Hardware implementation details 72
3.11: Summarized characteristics of different congestion-aware approaches 73
4.1: Robustness analysis of two faulty links in different network sizes 86
4.2: Hardware implementation details 99
4.3: Hardware implementation details 116
4.4: Summarized characteristics of different fault-tolerant approaches 117
5.1: UL: Unicast Latency; SP: Startup packets; SL: Startup Latency; 141
5.2: MML, MXML, and total latency in TBP, VBP, and RP methods 143
5.3: System configuration parameters 148
5.4: Performance gain of ARP over the other presented schemes 148

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
AgRA	Agent-based Routing Algorithm
ANoC	Agent-based Network-on-Chip
ARP	Adaptive Recursive Partitioning
ATBP	Adaptive Two-Block Partitioning
AUL	Average Unicast Latency
AVBP	Adaptive Vertical-Block Partitioning
BFT-NoC	Bidirectional Fault-Tolerant NoC
BW	Bandwidth
CDG	Channel Dependency Graph
CL	Congestion Level
CoG	Center of Gravity
CPU	Central Processing Unit
CS	Congestion Status
C-Routing	Cluster based Adaptive Routing
DBAR	Destination-Based Adaptive Routing method
DP	Dual-Path
DRAM	Dynamic Random Access Memory
DyAD	Dynamic Adaptive Deterministic
DyXY	Dynamic XY
EDXY	Enhanced Dynamic XY
FIFO	First In, First Out
FRA	Fuzzy-based Routing Algorithm
GB	Giga Byte
GEMS	General Electric Manufacturing Simulator
GHz	Gigahertz
HAMUM	Hamiltonian Adaptive Multicast and Unicast Model
HARA	Highly Adaptive Non-Minimal Routing Algorithm
HARAQ	Highly Adaptive Non-Minimal Routing Algorithm based on Q-Learning
HiPFaR	High Performance Fault-tolerant Routing
KB	Kilo Byte
IC	Integrated Circuit
IP	Intellectual Property
ISA	Instruction Set Architecture
Mad-y	Maximally Fully Adaptive Routing
MAFA	Minimal and Adaptive Fault-Tolerant Algorithm

MAR	Minimal Adaptive Routing
MB	Mega Byte
MD	Minimal and Defect-resilient routing algorithm
MESI	Modified, Exclusive, Shared, Invalid
MF	Membership Function
MiCoF	Minimal-path Connection-retaining Fault-tolerant approach
MML	Mean Multicast Latency
MoM	Mean of Maxima
MP	Multi-Path
MPEG	Moving Picture Experts Group
MPSoC	Multi-Processor System-on-Chip
MxML	Maximum Multicast Latency
NFRA	Non-Fuzzy Routing Algorithm
NoC	Network-on-Chip
NoP	Neighbor-on-Path
PE	Processing Element
RAFT	Reconfigurable, Adaptive and Fault-Tolerant
RCA	Regional Congestion Awareness
ReRS	Reconfigurable Routing for Tolerating Faulty Switches
RP	Recursive Partitioning
SL	Startup Latency
SP	Startup Packets
SNUCA	Static Non-Uniform Cache Architecture
SoC	Systems-on-Chip
SPARC	Scalable Processor ARChitecture
SPLASH	Stanford Parallel Applications for Shared Memory
TBP	Two-Block Partitioning
TRA	Trapezoid-based Routing Algorithm
TSV	Through-Silicon-Via
UMC	United Microelectronics Corporation
VBP	Vertical-Block Partitioning
VC	Virtual Channel
VCTM	Virtual Circuit Tree Multicasting
VHDL	VHSIC hardware description language
VOPD	Video Object Plane Decoder

Chapter 1

Introduction

Traditionally, System-on-Chip (SoC) designers employ buses or hierarchical bus structures to interconnect Intellectual Property (IP) blocks. The advances in semiconductor technologies make it possible to integrate billions of gates and hundreds of processing units into a single chip [1]. This technology trend implies the need for a structured, scalable, reusable, and high performance communication platform which cannot be offered by bus infrastructures (Figure 1.1(a)). Therefore, Networks-on-Chip (NoCs) have emerged as a solution to address the communication demands of future SoC designs. The scalability and reusability characteristics of NoCs may result in the reduction of design time and the shortening of the time taken to reach the market for new products. An NoC consists of an interconnection of many switches to enable a large number of cores to communicate with each other.[1] Figure 1.1(b) shows a mesh-based NoC where each core is connected to a switch by a local network interface [2]. Cores can communicate with each other by propagating packets through switches in the network. Each switch is connected to its neighbors through bidirectional links. Typically in NoCs, the resources are scarce regarding the ever-increasing demand for a high performance communication among cores [3], [4], [5]. Therefore, a major challenge in this domain is achieving a high performance system using the limited available resources [1]. Routing algorithms can perform an important role in fulfilling this gap, by realizing these challenges.

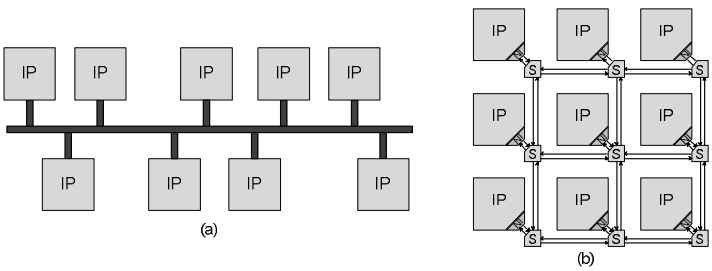

Figure 1.1: (a) Bus (b) Network-on-Chip

[1] Throughout this thesis, the term “switch” may refer to both the switch and the core connected to it.

As the size of the network is scaled up, the transmission delay between distant switches is significantly increased, which results in lower performance and higher power consumption. In addition, a 2D IC design imposes a very large chip area as the number of cores increases. Considering 2D design bottlenecks, the technology is moving toward the concept of 3D integrated circuits where multiple active silicon layers are vertically stacked. Combining the benefits of 3D IC and NoC schemes provides a significant performance gain for SoCs. Layers, stacked on top of each other, are connected via vertical interconnects tunneling through them. Wire bonding, micro-bump, contactless (capacitive or inductive), and through-silicon-via (TSV) interconnections are some of the vertical interconnect technologies that have been used in stacked structures [6], [7]. The TSV interconnection approaches have the potential to offer the greatest vertical interconnect density and therefore are the most promising among the other vertical interconnect techniques [6], [8], [9]. The major advantages of 3D ICs are the considerable reduction in the average wire length and wire delay, resulting in lower power consumption and better performance [10], [11], [12], [13]. Nevertheless, if the number of IP cores and memories increases in each layer, more TSVs are required to handle the inter-layer communication. Inasmuch as each TSV employs a pad for bonding, the area footprint in each layer is augmented significantly [14]. Modern SoC designs can benefit from 3D integration by placing memory blocks on top of a processing core in different layers [15]. This introduces a higher bandwidth and a shorter critical path [16].

Routing protocols have a significant impact on the latency and power consumption of NoC-based systems. In order to avoid the blocking of packets within NoCs, some rules should be defined. These rules are determined by routing algorithms. Some routing algorithms are very limiting regarding the adaptivity of packets, yet some others are not. In other words, routing algorithms may force packets to be routed through specific paths, while in others the packets can be distributed over all the possible routes. Routing algorithms may not be able to tolerate faults and thus the network may stop functioning after the occurrence of a single fault. Fault-tolerant routing algorithms make the network resilient against faults by allowing packets to choose among alternative non-faulty routes. The support of collective communication has a great impact on the overall performance of the system. The reason for this impact is that if collective communication is not supported, then multiple unicast packets should be delivered into the network, which imposes a significant amount of traffic on the network. On the other hand, routing algorithms should be designed based on the limitations and requirements of the system. In SoC designs, these limitations include the area overhead, the power budget, the temperature, and the required level of performance. In sum, moving towards more efficient communication protocols for NoCs considering the constraints of SoC designs forms the main contribution of this thesis.

1.1 Thesis Contributions

The underlying routing algorithms in current systems are mostly taken from the supercomputer concepts where energy consumption, design specialization, resource limitation, and the area overhead are not big constraints. This implies that the current

routing algorithms cannot satisfy the requirements of NoCs well. As a result, high-speed processing elements and memories are developed while the interconnection between them might be underutilized. In this thesis, we provide in-depth studies on routing algorithms to discover the key problems in the current and next generation of many-core SoCs. Different aspects of routing algorithms are discussed in this thesis as congestion, fault-tolerant, and collective communication.

1.1.1 Congestion-aware Techniques

Congestion occurs frequently in NoCs when the demands of the packets exceed the capacity of the network resources [17], [18]. Congestion may lead to increased transmission delay and power consumption. Performance can be improved by routing packets through less congested regions and distributing traffic over the network. In contrast with supercomputers, in NoCs the congestion information can be easily propagated over the network, and thus by using this knowledge, the traffic can be balanced over the network [19]. Based on this characteristic of NoCs, we propose three different approaches to alleviate congestion over a network. These algorithms can be divided into cluster-based, learning-based, and fuzzy-based approaches.

- *Cluster-based approach*

  In traditional congestion-aware techniques, congestion is measured at a switch level and delivered to other switches, either local or non-local. One of the contributions of this thesis is to show that performance can be improved if the congestion level is measured for a group of switches and propagated over the network, rather than considering the congestion level of single switches. We have proposed two algorithms: the Agent-based Routing Algorithm (AgRA) and the Trapezoid-based Routing Algorithm (TRA). In AgRA, a lightweight clustering structure is built upon a mesh network to propagate the congestion information over the different regions of the network. This approach presents an efficient solution for providing a better view of the network traffic condition. In TRA, we investigate the impact of both the routing unit and switch arbitration unit in distributing the traffic load over a network. In the proposed method, the congestion information is gathered from a group of switches which are more likely chosen as intermediate switches and can provide up-to-date information for a given switch.

- *Learning-based approach*

  Almost all of the existing methods are based on using the shortest paths in the network. However, in high traffic conditions a longer and a less congested path may result in a lower latency for a packet. On the other hand, gathering the information from all minimal and non-minimal routes may not be possible, and would need an intelligent method in order to cope. Another contribution of this thesis is to propose a non-minimal routing algorithm for on-chip networks, called the Highly Adaptive Non-Minimal Routing Algorithm (HARA), providing a wide range of alternative paths between each pair of source and destination switches. In addition to this, for

selecting a less congested path, an optimized and scalable learning method is utilized. The learning method is based on local and global congestion information and can estimate the latency from each output channel to the destination region. The Q-Learning-based approach using HARA is called HARAQ.

- *Fuzzy-based approach*

  In traditional methods, the comparison between the congestion values of candidate output ports is very strict, meaning that a single free buffer slot may change the routing decision toward a more congested region. To address this problem, we propose a Fuzzy-based Routing Algorithm (FRA) to estimate the latency of each candidate direction. Fuzzy systems avoid arbitrary rigid boundaries by giving a level of confidence to a data. Thus the use of fuzzy-logic algorithms in the routing decision unit leads to a systematic comparison among the candidates of output ports.

1.1.2 Fault-Tolerant Techniques

Faults can occur in NoCs like in any other electrical system. Traditional techniques tolerate faults by forcing packets to route around the fault. However, we show that faults can be tolerated in NoCs without taking unnecessary, longer paths. Short wires are utilized to inform the surrounding switches about the location of faults. Using this information, the proposed routing algorithms avoid sending packets through faulty components. The proposed methods are divided into two main groups; tolerating faulty links and tolerating faulty switches. In each group, two algorithms are presented. The distinguishing characteristic of these algorithms is the toleration of faults while also maintaining the performance of the network.

- *Tolerating faulty links*

  The first algorithm in this group, called the Minimal and Defect-resilient routing algorithm (MD), targets the addressing of a total link failure where the key ideas are twofold. First, it can tolerate all one-faulty links using the shortest path between each pair of source and destination switches, if such path exists. Therefore, unnecessary longer paths are avoided when tolerating faults. Second, unlike traditional fault-tolerant routing algorithms which are based on deterministic algorithms, the proposed method is nearly fully adaptive. So, in addition to tolerating faults, traffic can be balanced by distributing packets over the network. MD takes advantage of one and two virtual channels along the X and Y dimensions, respectively. This idea of MD is extended in the second approach, called Minimal and Adaptive Fault-Tolerant Algorithm (MAFA), to tolerate two faulty links in the network. Increasing reliability comes at the cost of using an extra virtual channel along the Y dimension. MAFA uses two virtual channels along both X and Y dimensions and is able to tolerate a wide range of multiple faulty links without affecting the performance of the network.

- *Tolerating faulty switches*

  The algorithms in this group, called the High Performance Fault-tolerant Routing (HiPFaR) and the Minimal-path Connection-retaining Fault-tolerant (MiCoF) approaches, are able to tolerate faulty switches in the network and to avoid rerouting packets around faults. In other words, a proper non-faulty route is chosen for a packet prior to it, reaching the fault. Similar to MD, HiPFaR uses one and two virtual channels along the X and Y dimensions but it is able to tolerate all single faulty switches in the network using only the shortest paths, if such path exists. A non-minimal route is necessitated when the source and destination switches are located in the same row or column with a faulty switch between them. In MiCoF, the same amounts of virtual channels are used. However, it is able to tolerate a wide range of faults by using only the shortest paths even if the source switch is in the same row or column as the destination switch. This is achieved by a simple modification in the switch architecture.

1.1.3 Collective Communication

The lack of hardware-level multicast support may result in decreased performance. Multicast can be easily implemented with no hardware overhead by assuming that a multicast packet is replicated and every instance is sent to a particular destination (this is termed unicast-based multicast). However, this implementation is inefficient. This inefficiency arises because sending multiple copies of the same packet into the network not only imposes a significant amount of traffic into the network and increases the overall power consumption, but also introduces a large serialization delay at the injection point of the packets into the network. Based on this fact, the support of multicast communication results in a significant performance gain, even if a small percentage of the traffic is multicast. However, current hardware-level multicast support limits the adaptivity of unicast packets. This means that all unicast packets have to be routed through a single path for each pair of source and destination switches. Taking into account that unicast communication forms the vast majority of almost all traffic, performance can be significantly improved if the multicast support does not limit the adaptivity of unicast packets. In this thesis, we propose a unicast/multicast communication protocol to maintain the performance of unicast communication while supporting the multicast communication efficiently. This method consists of two main parts; in the first part several partitioning methods are introduced in order to reduce the overall path length of multicast packets while the second part presents an adaptive routing algorithm for both unicast and multicast packets. These approaches are discussed in both 2D and 3D mesh networks.

- *Partitioning methods*

  Partitioning methods try to reduce latency and increase performance through an efficient partitioning of destinations into disjoint subsets. In this thesis, we propose several partitioning methods, called Two-Block Partitioning (TBP), Vertical-Block Partitioning (VBP), and Recursive Partitioning (RP), for 3D mesh NoCs. In TBP,

destinations are divided into two groups and a multicast packet is responsible for delivering the packet to all destinations within its group. In VBP, the network is vertically partitioned and destinations are divided based on the partitions to which they belong. Thereby, latency can be decreased in comparison with TBP but more packets should be delivered into the network to cover all destinations of a multicast message. RP tries to have a comparable number of switches within each partition, offering balanced partitions while keeping the number of delivered packets low.

- *Adaptive routing algorithm for both unicast and multicast communication*

  In addition to these partitioning methods, and in order to distribute the unicast and multicast packets more efficiently, we present a minimal and adaptive routing algorithm, called the Minimal Adaptive Routing (MAR), which is based on the Hamiltonian path. Using MAR, congestion can be alleviated by routing both unicast and multicast packets adaptively in the network. Furthermore, MAR is relatively simple and does not require any virtual channel.

1.2 Thesis Organization

The thesis is organized as follows. Chapter 2 reviews the Network-on-Chip architectures and different types of routing algorithms. In Chapter 3, three different congestion-aware approaches are proposed. In the first section of this chapter, Section 3.1, traditional methods are reviewed and the advantages and disadvantages of each method are discussed. In Section 3.2, the proposed cluster-based approaches are described in which congestion is measured at a region level rather than at individual switches. Section 3.3 introduces the learning-based method. Within this section, it is explained how the traffic condition is learned at run time and a less congested path is selected to route packets. Section 3.4 explains how to provide a reliable routing decision by employing the fuzzy-based technique. Section 3.5 gives a summary of the proposed methods. Chapter 4 introduces four high performance fault-tolerant routing algorithms. Section 4.1 reviews different well-known fault-tolerant approaches along with their advantages and shortcomings. In Section 4.2, we explain the proposed fault-tolerant approaches to tolerate faulty links in the network. Tolerating faults at the switch level are investigated in Section 4.3. These algorithms are designed to tolerate faults with an emphasis on performance. A short summary of the proposed methods are provided in Section 4.4. Chapter 5 presents an efficient unicast and multicast communication approach for 3D NoCs. In Section 5.1, the traditional methods in a 2D mesh network are described. Different partitioning methods and their analytical formulas for a 3D mesh network are provided in Section 5.2. The adaptive routing algorithm for these partitioning methods is explained in Section 5.3. Analytical and simulation results are investigated in Section 5.4 while a summary of the proposed methods are given in Section 5.5. The goal of this chapter is to show that the multicast communication can be efficiently supported without any negative impact on the performance of unicast communication. Finally, Chapter 6 concludes the thesis.

This thesis is written based on my research during the period from April 2009 to May 2013 at the University of Turku, Finland. The following publications are the main references of this thesis:

- *Journal Publications*

1. **M. Ebrahimi**, Hannu Tenhunen, "Fuzzy-based Adaptive Routing Algorithm for Networks-on-Chip," *Journal of Systems Architecture*, 2013.
2. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, J. Plosila, J. Flich, and H. Tenhunen, "Path-based Partitioning Methods for 3D Networks-on-Chip with Minimal Adaptive Routing," *IEEE Transaction on Computers (IEEE TC), Special issue on NOCS*, 2013.
3. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Cluster-based Topologies for 3D Networks-on-Chip Using Advanced Inter-layer Bus Architecture," *Elsevier Journal of Computer and System Sciences (JCSS-elsevier)*, Vol 79, No. 4, pp. 2013.
4. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, J. Plosila, and H. Tenhunen, "A systematic reordering mechanism for on-chip networks using efficient congestion-aware method," *Elsevier Journal of Systems Architecture (JSA-Elsevier)*, 2012.
5. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, J. Plosila, and H. Tenhunen, "Memory-Efficient On-Chip Network with Adaptive Interfaces," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems (IEEE-TCAD)*, Vol. 31, No. 1, pp. 146-159, 2012.
6. M. Daneshtalab, M. Kamali, **M. Ebrahimi**, S. Mohammadi, A. Afzali-Kusha, and J. Plosila, "Adaptive Input-output Selection Based On-Chip Router Architecture," *Journal of Low Power Electronics (JOLPE)*, Vol. 8, No. 1, pp. 11-29, 2012.
7. M. Daneshtalab, **M. Ebrahimi**, T. C. Xu, P. Liljeberg, and H. Tenhunen, "A generic adaptive path-based routing method for MPSoCs," *Journal of Systems Architecture (JSA-elsevier)*, Vol. 57, No. 1, pp. 109-120, 2011.
8. M. Daneshtalab, **M. Ebrahimi**, S. Mohammadi, A. Afzali-Kusha, "Low distance path-based multicast algorithm in NOCs," *IET (IEE) Special issue on NoC*, Vol. 3, No. 5, pp. 430-442, 2009.

- *Conference Publications*

9. **M. Ebrahimi**, M. Daneshtalab, J. Plosila, H. Tenhunen, "Minimal-Path Fault-Tolerant Approach Using Connection-Retaining Structure in Networks-on-Chip," *in Proceedings of 7th International Symposium on Networks-on-Chip (NOCS)*, 2013.
10. M. Daneshtalab, **M. Ebrahimi**, J. Plosila, H. Tenhunen, "CARS: Congestion-Aware Request Scheduler for Network Interfaces in NoC-based Manycore Systems," *in Proceedings of 16th ACM/IEEE Design, Automation, and Test in Europe (DATE)*, pp. 1048-1052, 2013.
11. **M. Ebrahimi**, M. Daneshtalab, J. Plosila, "Fault-Tolerant Routing Algorithm for 3D NoC Using Hamiltonian Path Strategy," *in Proceedings of 16th ACM/IEEE Design, Automation, and Test in Europe (DATE)*, pp. 1601-1605, 2013.
12. **M. Ebrahimi**, M. Daneshtalab, J. Plosila, "High Performance Fault-Tolerant Routing Algorithm for NoC-based Many-Core Systems," *in Proceedings of 21th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, pp. 463-469, 2013.

13. **M. Ebrahimi**, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, H. Tenhunen, "DyXYZ: Fully Adaptive Routing Algorithm for 3D NoCs," *in Proceedings of 21th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, pp. 499-503, 2013.

14. **M. Ebrahimi**, M. Daneshtalab, J. Plosila, F. Mehdipour, "MD: Minimal path-based Fault-Tolerant Routing in On-Chip Networks", *in Proceedings of 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 35-40, 2013.

15. **M. Ebrahimi**, M. Daneshtalab, J. Plosila, H. Tenhunen, "MAFA: Adaptive Fault-Tolerant Routing Algorithm for Networks-on-Chip," *in Proceedings of 15th IEEE Euromicro Conference on Digital System Design (DSD)*, pp. 201-206, 2012.

16. **M. Ebrahimi**, M. Daneshtalab, J. Plosila, "GLB - Efficient Global Load Balancing Method for Moderating Congestion in On-Chip Networks," *in Proceedings. of 7th IEEE International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pp. 1-5, 2012.

17. X. Chang, **M. Ebrahimi**, M. Daneshtalab, T. Westerlund, J. Plosila, "PARS – An Efficient Congestion-Aware Routing Method for Networks-on-Chip," *in Proceedings of 16th IEEE International Symposium on Computer Architecture and Digital Systems (CADS)*, pp. 166-171, 2012.

18. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, P. Liljeberg, J. Plosila, M. Palesi, and H. Tenhunen, "HARAQ: Congestion-Aware Learning Model for Highly Adaptive Routing Algorithm in On-Chip Networks," *in Proceedings of 6th ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pp. 19-26, 2012.

19. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "CATRA- Congestion Aware Trapezoid-based Routing Algorithm for On-Chip Networks," *in Proceedings of 15th ACM/IEEE Design, Automation, and Test in Europe (DATE)*, pp. 320-325, 2012.

20. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "LEAR – A Low-weight and Highly Adaptive Routing Method for Distributing Congestions in On-Chip Networks," *in Proceedings of 20th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, pp. 520-524, 2012.

21. M. Daneshtalab, **M. Ebrahimi**, J. Plosila, "HIBS-Novel Inter-layer Bus Structure for Stacked Architectures," *in Proceedings of IEEE International 3D Systems Integration Conference (3DIC)*, pp. 1-7, 2011.

22. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, J. Plosila, and H. Tenhunen, "Memory-Efficient Logic Layer Communication Platform for 3D-Stacked Memory-on-Processor Architectures," *in Proceedings of IEEE International 3D Systems Integration Conference (3DIC)*, pp. 1-8, 2011.

23. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Agent-based On-Chip Network Using Efficient Selection Method," *in Proceedings of 19th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 284-289, 2011.

24. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, et al., "High-Performance On-Chip Network Platform for Memory-on-Processor Architectures," *in Proceedings of IEEE International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pp.1-6, 2011.

25. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Efficient Congestion-Aware Selection Method for On-Chip Networks," *in Proceedings of IEEE International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pp.1-4, 2011.

26. M. Dehyadegari, M. Daneshtalab, **M. Ebrahimi**, J. Plosila, and S. Mohammadi, "An Adaptive Fuzzy Logic-based Routing Algorithm for Networks-on-Chip," *in Proceedings of 13th IEEE/NASA-ESA International Conference on Adaptive Hardware and Systems (AHS)*, pp. 208-214, 2011.

27. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, J. Plosila, and H. Tenhunen, "Cluster-based Topologies for 3D Stacked Architectures," *in Proceedings of ACM International Conference on Computing Frontiers (CF)*, No. 14, 2011.
28. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Exploring Partitioning Methods for 3D Networks-on-Chip Utilizing Adaptive Routing Model," *in Proceedings of 5th ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pp. 73-80, 2011.
29. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, H. Tenhunen, "Performance Evaluation of Unicast and Multicast Communication in Three-Dimensional Mesh Architectures," *in Proceedings of 15th IEEE International Symposium on Computer Architecture & Digital Systems(CADS)*, pp.181-182, 2010.
30. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, J. Plosila, and H. Tenhunen, "Pipeline-Based Interlayer Bus Structure for 3D Networks-on-Chip," *in Proceedings of 15th IEEE International Symposium on Computer Architecture & Digital Systems(CADS)*, pp.41-47, 2010.
31. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, S. Mohammadi, A. Afzali-Kusha, and J. Plosila, "Input-Output Selection Based Router for Networks-on-Chip," *in Proceedings of 9th IEEE International Symposium on VLSI (ISVLSI)*, pp. 92-97, 2010.
32. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, J. Plosila, and H. Tenhunen, "A Low-Latency and Memory-Efficient On-Chip Network", *in Proceedings of 4th IEEE/ACM International Symposium on Network-on-Chip (NOCS)*, pp.99-106, 2010.
33. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, H. Tenhunen, "Partitioning Methods for Unicast/Multicast Traffic in 3D NoC Architecture", *in Proceedings of 13th IEEE International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS)*, PP. 127-132, 2010.
34. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, H. Tenhunen, "HAMUM – A Novel Routing Protocol for Unicast and Multicast Traffic in MPSoCs", *in Proceedings of 18th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, pp. 525-532, 2010.
35. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, J. Plosila, H. Tenhunen, "A High-Performance Network Interface Architecture for NoCs Using Reorder Buffer Sharing", *in Proceedings of 18th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, pp. 547-550, 2010.
36. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, H. Tenhunen, "An Efficient Unicast/Multicast Routing Protocol for MPSoCs", *in Proceedings of 12th IEEE Euromicro Conference On Digital System Design (DSD)*, pp. 203-206, 2009.
37. **M. Ebrahimi**, M. Daneshtalab, S. Mohammadi, A. Afzali-Kusha, H. Tenhunen, "An Efficent Dynamic Multicast Routing Protocol for Distributing Traffic in NOCs", *in Proceedings of 12th IEEE/ACM International Conference on Design, Automation, and Test in Europe (DATE)*, pp. 1064-1069, 2009.
38. **M. Ebrahimi**, M. Daneshtalab, N. Sreejesh, P. Liljeberg, H. Tenhunen, "Efficient Network Interface Architecture for Network-on-Chips", *in Proceedings of 27th IEEE Norchip*, pp. 1-4, 2009.

- *PhD Forum and Workshops*

39. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, J. Plosila, and H. Tenhunen, "High-Performance TSV Architecture for 3-D ICs", *in Proceedings of 9th IEEE International Symposium on VLSI (ISVLSI)*, PhD-Forum, pp. 467-468, 2010.
40. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, H. Tenhunen, "Performance Analysis of 3D NoCs Partitioning Methods", *in Proceedings of 9th IEEE International Symposium on VLSI (ISVLSI)*, PhD-Forum, pp. 467-468, 2010.

41. **M. Ebrahimi**, M. Daneshtalab, P. Liljeberg, H. Tenhunen, "Partitioning Methods for Unicast/Multicast Routings in 3D Mesh NOCs", *3D Integration Workshop, The Design, Automation, and Test in Europe (DATE),* 2010.

42. M. Daneshtalab, **M. Ebrahimi**, P. Liljeberg, H. Tenhunen, "A Novel Interlayer Bus Architecture for Three-Dimensional Network-on-Chips", *3D Integration Workshop, The Design, Automation, and Test in Europe (DATE),* 2010.

Chapter 2

Networks-on-Chip

The Network-on-Chip (NoC) technology has emerged as a solution to address the communication demands of future many-core Systems-on-Chip (SoCs) due to its reusability, scalability, and higher bandwidth compared with traditional approaches [20]. NoCs have been increasing in popularity in both academia and industry (e.g. Tile64 [21] and Polaris [22]). The technology enables the integration of a large number of Intellectual Property (IP) cores into a single chip [23], [24]. The performance and efficiency of NoCs largely depends on the underlying routing technique which decides the routes of packets between the source and destination switches. In this chapter, we review the characteristics of NoCs and the basics of routing algorithms with further emphasis on the factors which are employed in this thesis.

2.1 Networks-on-Chip Characteristics

In this section, we take a look at NoC designs in general. An NoC is defined by many characteristics such as the network dimension, topology, switch architecture, switching technique, and flow control. These characteristics have a direct impact on performance, latency, and power consumption.

2.1.1 Network Dimension

The NoC design is commonly discussed in the form of two-dimensional (2D) and three-dimensional (3D) architectures [25]. As shown in Figure 2.1(a), in 2D NoCs all switches are laid down in a single layer and connected to each other via intra-layer connections. In 3D NoCs (Figure 2.1(b)), layers are stacked on top of each other via inter-layer connections instead of being spread across a 2D plane [26]. Each layer can use different technologies, topologies, clock frequencies, etc. In recent years, through-silicon-via (TSV) has attracted a lot of attention to be employed for the inter-layer connections (vertical channels). TSVs enable faster and more power efficient inter-layer communication across multiple stacked layers. Figure 2.1 illustrates a 2D and 3D network with an almost similar number of cores.

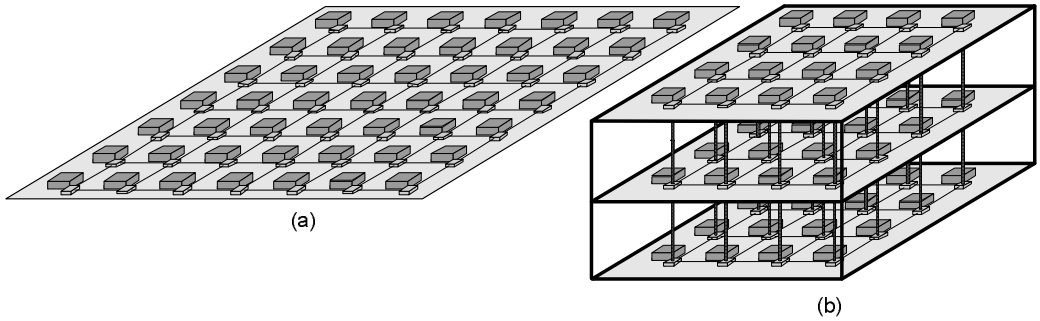

Figure 2.1: (a) 2D NoC (b) 3D NoC

2.1.2 Network Topology

NoCs consist of a set of switches and links. A topology determines the arrangement of these switches and links. The selection of a topology can affect latency and power consumption. Lots of topologies have been proposed so far, such as Ring, Mesh, Torus, and Butterfly [24], [27]. Two popular topologies used in NoCs are mesh and torus. As shown in Figure 2.2(a), in the mesh topology, each switch is connected to its four neighboring switches. Although mesh topology offers the same wire length between the connected switches, it suffers from the long distances between distant switches that cause a negative effect on the communication latency. In the torus topology, the switches on the edges are connected to the switches on the opposite edges through wrap-around channels [28] (Figure 2.2(b)). This results in a shorter distance between distant switches. However, the long wrap-around connections may result in excessive delay. Among mesh and torus topologies, the mesh topology has gained more attention and has been widely used in NoCs, mainly because of its simplicity and symmetrical characteristic. Some other topologies are also defined in 3D networks such as a cluster-based topology [29]. In this thesis, our focus is on mesh topology in both 2D and 3D networks.

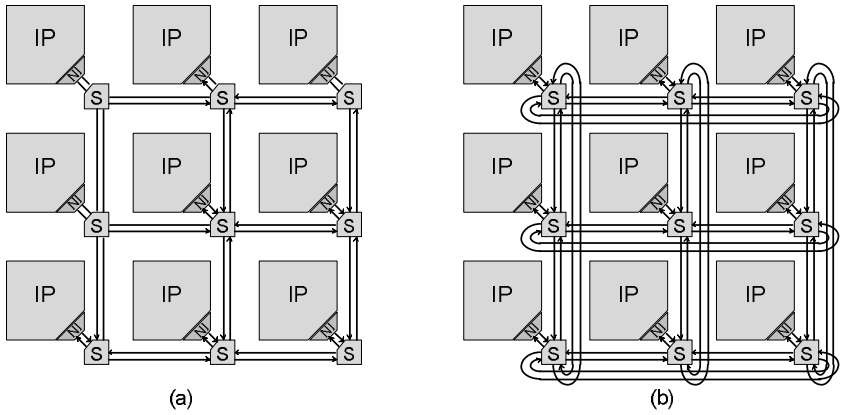

Figure 2.2: (a) Mesh topology (b) Torus topology

2.1.3 Switch Architecture

A generic switch architecture is illustrated in Figure 2.3. It consists of a routing unit, a virtual channel (vc) allocator, a switch allocator, a crossbar, and input buffers. The routing unit determines the output port and the virtual channel for an incoming packet. Multiple packets may request the same output port and/or virtual channel. However, an output port or a virtual channel should be granted to at most one packet at a time. The switch allocator grants a packet to access the output port among all requestors while the virtual channel allocator chooses a packet to get access to the requested virtual channel. When granted, packets on input ports should be connected to the corresponding output ports. The crossbar unit is responsible for making this connection.

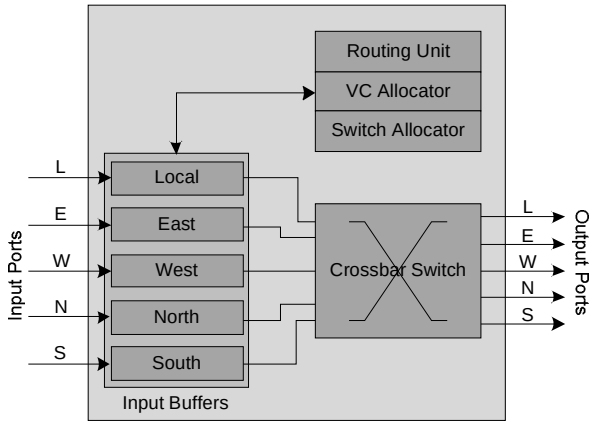

Figure 2.3: A switch architecture

The switch-based and bus-based [30], [31] organizations are the two dominant architectures for 3D NoCs. In the switch-based architecture, each switch has 7 input/output ports, a natural extension from a 5-port 2D switch by adding two ports to make connections to the upper and lower layers [16]. Thereby, the obtained 3D switch requires additional buffers with some modifications on the virtual channel allocator, switch allocator, and crossbar switch. This architecture is called 3D symmetric NoC [16]. In the bus-based architecture, however, each switch has 6 input/output ports where 1 input/output port is connected to the bus architecture. Both approaches have some disadvantages. The bus-based approach suffers from the poor scalability and deteriorates performance at high injection rates, while the switch-based design consumes more area and power. In this thesis, the default architecture is chosen as a simple switch-based design.

2.1.4 Virtual Channel

Packet latency and network throughput can be improved by dividing the buffer associated with each physical channel into several virtual channels [32]. In this way, packets can make progress by sharing the physical channel rather than remain blocked to get a free channel.

Figure 2.4(a) shows a typical switch in the XY network. In this figure, each input channel is paired with a corresponding output channel. By adding two virtual channels per physical channel, a double-XY network is obtained (Figure 2.4(b)). The virtual channels in each dimension are differentiated by vc1 and vc2. Figure 2.4(c) shows the double-Y network in which one and two virtual channels are used along the X and Y dimensions, respectively. Note that one virtual channel simply refers to the physical channel while two virtual channels represent the physical channel shared by two different flows. All these types of switches are used throughout the thesis.

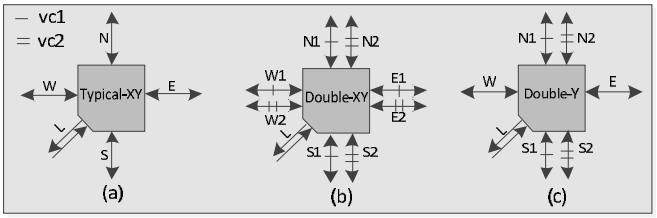

Figure 2.4: A switch in (a) XY (b) double-XY (c) double-Y network

2.1.5 Switching Techniques

Two common types of switching techniques are circuit switching and packet switching. In circuit switching, a physical path is reserved between the source and destination switches prior to sending a packet. The main disadvantage of this technique is in underutilization of resources as some parts of a path might be idle for a significant period of time [33]. In packet switching, the path is not established between the source and destination switches and the routing decision is made at each intermediate switch. At intermediate switches, packets are temporarily stored in input buffers to receive a proper output channel. In packet switching, communication links are better utilized as they can be simultaneously used by different packets. In NoCs, the packet switching technique is more preferred which is also employed in this thesis.

2.1.6 Flow Control Mechanisms

Packet switching technique can be implemented using three schemes as store-and-forward, virtual cut-through and wormhole switching. In store-and-forward, the whole packet should be stored in the input buffer before proceeding to the next one. In virtual cut-through, the packet can be forwarded to the next switch before it is completely received by the current switch. However, there should be enough space in the next switch when delivering the packet. Obviously, virtual cut-through results in a lower latency than the store-and-forward scheme. In both approaches, the buffer sizes must be large enough to be able to accommodate the largest possible packet in the network. In wormhole switching, packets are divided into flits traversing through the network in a pipelined fashion. This approach eliminates the need to allocate large buffers in intermediate switches along the path [34]. However, a packet waiting to be allocated to an outgoing channel may prohibit other

packets from using the channels and thereby wasting channel bandwidth and increasing latency. Wormhole switching is the mostly used technique in the realm of NoCs and it is chosen as the default technique in this thesis.

2.2 Routing Algorithms

Routing algorithms can be classified into different categories such as deterministic vs. adaptive, minimal vs. non-minimal, and congestion-aware vs. congestion-oblivious. Moreover, routing algorithms may be designed to support collective communication or to tolerate faults. When designing routing algorithms, it is extremely important to remove deadlocks, livelocks, and starvation. Turn models are used to guarantee deadlock freeness in the network. This section gives the overall view about these issues.

2.2.1 Deterministic and Adaptive Routing

Routing algorithms can be classified as deterministic and adaptive algorithms. The simplest deterministic routing method is dimension-order routing which is known as XY in a 2D network and XYZ in a 3D network. Using dimension-order routing algorithms, packets are routed by crossing dimensions in strictly increasing order, reducing to zero the offset in one direction before proceeding to the next one. Implementations of deterministic routing algorithms are simple but they are unable to balance the load across the links in a non-uniform or bursty traffic [35].

In adaptive routing algorithms, a packet can traverse from a source to a destination switch through multiple paths. Adaptive routing has been used in interconnection networks to improve network performance and to tolerate link or switch failures. Specifically, adaptive routing algorithms can be used to avoid congestion by adapting the routing decision with the network status [36]. Thereby, they can decrease the probability of routing packets through congested regions. Adaptive routing algorithms can be either partially adaptive or fully adaptive. In partially adaptive routing algorithms, packets are limited to choose among the shortest paths, while in fully adaptive methods, packets are allowed to take any minimal paths available between the source and destination pair [33], [34], [35].

2.2.2 Minimal and Non-minimal Routing

In minimal routing algorithms, the shortest paths are used between each pair of source and destination switches. In non-minimal routing, packets may take longer paths to reach the destination switch. Non-minimal routing algorithms are usually used to tolerate faults rather than avoiding congestion in the network. Employing non-minimal routing algorithms as a congestion-aware technique may result in a severe deterioration of performance. This is due to the fact that packets may take longer paths and meanwhile passing through congested regions. Moreover, non-minimal methods suffer from a more complex switch structure than minimal schemes. It is worth mentioning that, in this thesis we show that non-minimal routing algorithms can be used to alleviate congestion in the network when coupling with an intelligent method to estimate the traffic of different routes.

2.2.3 Congestion-aware and Congestion-oblivious Routing

Adaptive routing algorithms can be decomposed into routing and selection functions [37]. The routing function suggests a set of output channels to deliver a packet. The selection function selects an output channel from the set of channels supplied by the routing function. The selection function can be classified as either congestion-oblivious or congestion-aware schemes. In congestion-oblivious algorithms, routing decisions are independent of the congestion condition of the network. This policy may disrupt the load balance since the network status is not considered. In congestion-aware routing algorithms, the path a packet traverses from a source to a destination is determined by the network condition which can be based on local or global information. In approaches considering local traffic conditions, the routing decision is made only based on the congestion statuses of adjacent neighbors. These methods provide a limited view of the network condition. Routing algorithms based on global information provide a better distribution of the traffic load. However, the congestion information should be collected, distributed, and utilized in an efficient way. Congestion-aware algorithms can take advantages of different metrics such as the number of free buffer slots [38], [39], [40], [41], [42], available virtual channels [43], crossbar demand, or combinations of these factors [44].

2.2.4 Unicast and Multicast Routing

Communication in NoCs may be in form of unicast or multicast messages [45]. In the unicast communication, a packet is sent from a source switch to a single destination switch, while in the multicast communication, a packet is delivered from one source switch to an arbitrary number of destination switches. Multicast communication can be implemented by sending a single unicast packet per destination of a multicast message (Figure 2.5(a)). However, it increases the network congestion due to sending multiple copies of the same message into the network. This results in decreasing performance significantly.

Hardware-based multicast schemes can be broadly classified into path-based [46], [47] and tree-based methods [48]. In the tree-based method, a spanning tree is built at the source switch and a single multicast packet is sent down the tree (Figure 2.5(b)). The source switch is considered as the root while destinations are the leaves of this tree. The packet is replicated along its route at switches and forwarded along multiple outgoing channels reaching to disjoint subsets of destinations [27]. With tree-based multicast, care must be taken to avoid deadlock. When using wormhole switching, dependencies between branches of different multicast trees may create deadlock. Typically more resources are needed to avoid such situations (e.g. using virtual channels or implementing virtual cut-through switching). Also, replicating a packet through different output ports leads to larger contention within the network (due to the temporal blocking of a branch). The situation is even more difficult in 3D designs, because the probability of forming deadlock and blockage is higher in such systems.

In the path-based multicast method (Figure 2.5(c)), a source switch prepares a packet for delivery to a set of destinations by placing the list of destinations in the header of the packet. The packet is routed along the path until it reaches the first destination. The packet

is delivered both to the local core and to the corresponding output channel toward the next destination in the list. In this way, the packet is eventually delivered to all specified destinations. Notice that the path-based approach does not replicate packets within the network, thus not increasing packet contention. However, the path visiting all switches can become large.

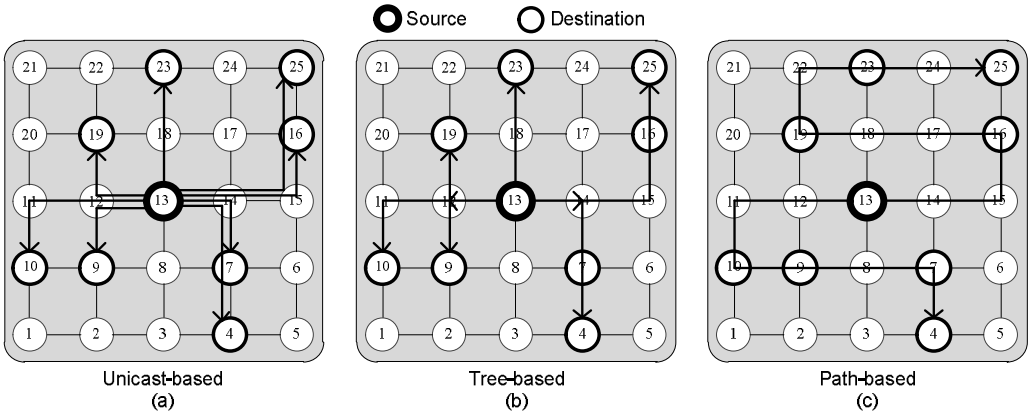

Figure 2.5: Multicast approaches

2.2.5 Fault-Tolerant Routing

Two different types of faults that can occur in NoCs are transient and permanent. Transient faults have unpredictable causes (e.g. power grid fluctuations and particle hits) and they are often difficult to be detected and corrected. Permanent faults are caused by physical damages such as manufacturing defects and device wear-out.

NoC, inherently, has a robust infrastructure by providing alternative routes between different switches. Fault-tolerant routing algorithms realize this potential and enable the NoC to be a fault-tolerant system. Fault-tolerant algorithms attempt to tolerate faulty links or switches by routing packets through alternative paths. Although fault-tolerant algorithms can bring a high degree of robustness for NoCs, on the other hand, they may result in huge performance loss. This is because of sending packets through longer paths and creating congestion around the fault.

2.2.6 Starvation, Deadlock, and Livelock

Starvation is the situation in which low priority packets cannot get access to output ports as high priority packets always win the arbitration. The arbitration policy can be based on the round-robin or the packet priority scheme. Round-robin provides a fair scheduling by serving packets in a circular manner, so that it is starvation-free. However, packet prioritization might be needed in the network for the quality of service or it can be used for increasing the overall performance. On the other hand, prioritization schemes may lead to

starvation in the network. A proper resource assignment can avoid starvation while giving packets different priorities.

Deadlock is a situation where a set of packets are permanently waiting for each other and no progress is made in the network. Deadlock is formed by a circular wait between packets in which each packet holds a channel needed by the next one. One solution to break cyclic dependencies is to drop a couple of packets involved in a deadlocked configuration. However, this is not a good approach as the deadlock situation might occur occasionally in the network. A better solution is to prevent deadlock by the means of routing algorithms in the routing unit which can be based on turn models or adding virtual channels.

A different situation, called livelock, arises when some packets are not able to reach their destinations, even if they are not blocked permanently [33]. In this situation, packets keep moving indefinitely in the network without any progress toward their destinations. Similar to the deadlock issue, livelock can be prevented if it is properly being considered in the routing algorithm. Livelock may arise only when non-minimal routing is employed.

2.2.7 Turn Models

Turn models are firstly proposed by Glass and Ni [129] to provide a systematic approach for deadlock-free adaptive routing [33], [49]. There are two types of complete cycles that can be formed in the network, known as clockwise and counter-clockwise cycles (Figure 2.6(a)). The creation of a cycle may lead to deadlock in the network and thus it should be avoided. In turn models, certain turns are prohibited from each cycle in order to break all cyclic dependencies and thus avoiding deadlock. In the XY routing algorithm, for example, packets are routed along the X dimension before proceeding to the Y dimensions. As shown in Figure 2.6(b), in this algorithm, two turns are avoided from each abstract cycle and there is no possibility of forming a complete cycle among the remaining turns. Deadlock can be avoided by prohibiting fewer turns than in the XY routing algorithm. Negative-First (Figure 2.6(c)), West-First (Figure 2.6(d)) and North-Last (Figure 2.6(e)) prohibit only two turns to avoid deadlock.

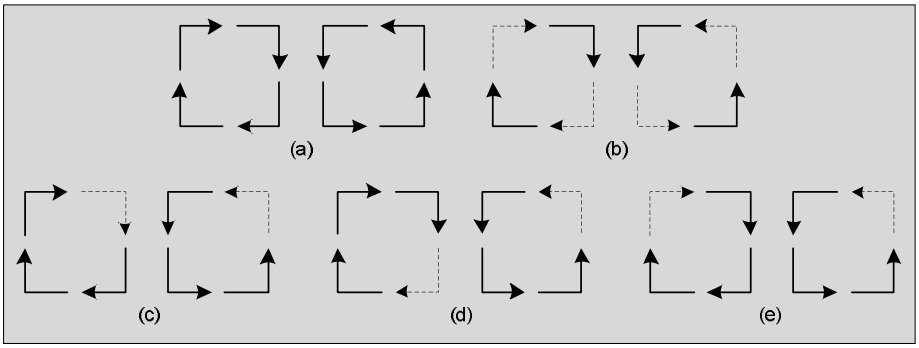

Figure 2.6: (a) Clockwise and counter-clockwise turns (b) XY routing (c) Negative-First (d) West-First (e) North-Last (Solid lines indicate the allowable turns and dash lines indicate the prohibited turns)

Chapter 3

Congestion-Aware Routing Algorithms for a 2D Mesh Network

The performance and efficiency of NoCs largely depends on the underlying routing technique. Congestion occurs frequently in NoCs when the demands of packets exceed the capacity of the network's resources. Congestion may lead to increased transmission delay, thus limiting the performance of NoC. Efficient routing algorithms can address this issue by routing packets through less congested regions and balancing traffic over the network. The routing decision can be based on local or non-local congestion information. Methods based on local congestion information are generally simple but they are unable to balance the traffic load efficiently. On the other hand, methods using non-local congestion information are more complicated while providing better traffic distribution.

In this chapter, we first investigate different well-known congestion-aware routing methods. The routing decision in some of them is made based on purely local congestion information, while in others non-local congestion information is also taken into account. We discuss the advantages and disadvantages of each method and the impact on routing decisions and traffic balancing. After that, we propose three main approaches to alleviate congestion in the network, which forms the main contributions of this chapter. The first approach is based on a clustering scheme in which congestion in the network is relevant to regions rather than switches. The second method takes advantage of learning approaches to estimate latency values from each source switch to different destinations at run time. The third approach employs the basic concept of fuzzy-logic. This scheme eliminates rigid boundaries on congestion metrics and provides a more reliable comparison mechanism to select between output directions. Based on the design characteristics such as area, performance, and power budget, the three proposed approaches (i.e. cluster-based, learning-based, and fuzzy-based approaches) can be combined together as desired.

3.1 Traditional Approaches

In this section, we investigate several traditional approaches to alleviate congestion in the network. Some of them are based on more local congestion information while others take more global information into account.

3.1.1 Dynamic XY (DyXY)

An adaptive routing algorithm called Dynamic XY (DyXY) is proposed in [38]. In this algorithm, which is based on the static XY algorithm, a packet is sent either to the X or Y dimension depending on the congestion condition. It uses local information, which is the number of free buffer slots in an input buffer of the neighboring switches, to decide on the next hop. One of the drawbacks of DyXY is that the use of local information in the routing decision may forward a packet to a switch which is not locally congested while the surrounding regions of this switch are highly congested. Such non-optimal routing decisions increase the network latency in NoC.

Figure 3.1 shows an example of the DyXY method where the routing decision based on local congestion information leads to deliver a packet through a congested region. In this figure, colors show the congestion level at each switch such that the darker color indicates the more congested switch. In this example, the switches 0 and 15 are the source and destination of the packet, respectively. In the DyXY method, the source switch 0 compares the occupied slots at the west input buffer of the switch 1 and that of the south input buffer of the switch 4. Since the switch 1 is less congested, the packet is sent to this switch. When the packet arrives at the switch 1, it has to be delivered through the switch 2 or 5. According to the congestion condition shown in Figure 3.1, the switch 2 is less congested and thus the packet is delivered to the switch 2. At the switch 2, the packet has to pass through the most congested region (i.e. the switches 6, 7, 10, and 11) in the network to reach the destination switch. This non-optimal path is as a result of making routing decisions based on local information. The packet could pass through the less congested region (i.e. the switches 8, 9, 12, and 13) if non-local information was considered.

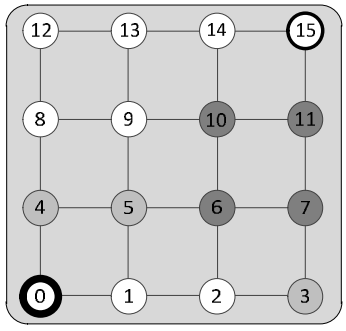

Figure 3.1: An example of the DyXY method
(Note that the darker color indicates the more congested switch)

3.1.2 Enhanced Dynamic XY (EDXY)

In the Enhanced Dynamic XY (EDXY) method [39], the network is provided with auxiliary wires along each row and column to carry the buffer statuses located in that row or column. This information is propagated to the switches in the adjacent row and column. In this method, every switch first looks at the destination address of the packet. If the destination

switch is not located in the adjacent row or column, the packet is routed similar to the DyXY method. However, if the destination address is just one hop apart from the switch in either the X or Y dimension, not only the queue length of the buffer in the neighboring switches are considered, but also congestion wires are used for the decision making.

Consider the example of Figure 3.2 where a packet is delivered from the source switch 0 to the destination 15 and it is already at the switch 2. Based on the DyXY method, since the switch 3 is less congested than the switch 6, the switch 3 is selected as the next hop. However, by this decision the packet has to pass the switches 7 and 11 which are highly congested. In contrast, in the EDXY method, in a similar situation (i.e. when a packet is at the switch 2), the congestion conditions of the third and fourth columns are compared together; since the third column is less congested, the packet is sent to the switch 6, thus avoiding packets to be routed through the highly congested switches.

In a similar example as in Figure 3.1, when the switches 6, 7, 10, and 11 are congested, EDXY behaves similar to DyXY and the packet has to be routed through the congested region due to the lack of global congestion information.

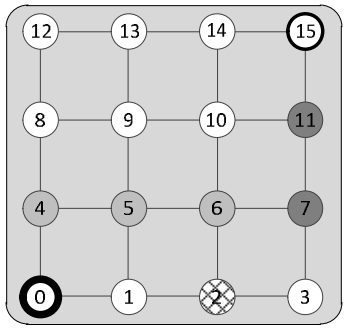

Figure 3.2: An example of the EDXY method

3.1.3 Neighbor-on-Path (NoP)

In the Neighbor-on-Path (NoP) [40] approach, the locality decision is based on 2-hop neighbors. An example of NoP is shown in Figure 3.3 where a packet is sent from the source switch 0 to the destination switch 15. At the source switch 0, the packet can be sent either to the switch 1 or the switch 4. In NoP, the congestion value in the X dimension is computed by considering the free buffer slots at the west input buffer of the switch 2 and south input buffer of the switch 5 (i.e. these switches are located in the minimal routing path to the destination). Similarly, the congestion value in the Y dimension is measured by using the number of free buffer slots at the south input buffer of the switch 8 and west input buffer of the switch 5. By comparing the obtained values in two directions, a packet is sent to the switch 1 or the switch 4. The congestion status of the switch 5 may not affect the routing decision. The reason for this is that, the number of free buffer slots in the south and west input buffers of the switch 5 is compared at the switch 0. On the other hand, since both of these input buffers belong to the same switch, their congestion values would be in a

similar rang as they are largely affected by the contention of output ports (i.e. north and east output ports in this example). Therefore, the comparison of these two values cannot have a considerable impact on the routing decision. The NoP method also suffers from the recursive nature of the routing algorithm, resulting in increased hardware overhead and switch complexity. This method cannot be extended to look at the congestion of 3-hop neighbors due to its significant hardware overhead.

Following the example of Figure 3.3, the packet is sent to the switch 1 since the congestion status of the switch 2 is less than the switch 8. At the switch 1, the packet is sent to the switch 2 as the switch 3 is less congested than the switch 9. As the result, the packet has to pass through the highly congested region (i.e. the switches 6, 7, 10, and 11).

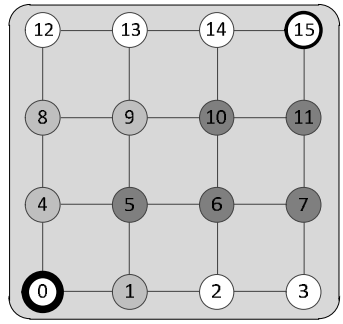

Figure 3.3: An example of the NoP method

3.1.4 Regional Congestion Awareness (RCA)

A well-known method called Regional Congestion Awareness (RCA-quadrant) is proposed in [44] to utilize non-local congestion information in the routing decision. In the RCA method, in order to collect global congestion value in switches, the locally computed congestion value of the corresponding input buffers of a switch is combined with those global signals propagated from upstream switches and the newly-aggregated value is transmitted to the downstream switches and so on. In this method, non-local congestion information is used to send a packet through a less congested direction. Even though RCA collects global information, in fact this information cannot be efficiently used in routing decisions.

Figure 3.4 shows an example of the RCA method where the switch 0 wants to communicate with the switch 15. Firstly, the packet should be sent to either the switch 1 or the switch 4 depending on global congestion information received from X and Y dimensions. According to the RCA method, the congestion value in the Y dimension is calculated by weighting sum (i.e. 50-50 weighting of local and propagated congestion values) of the values of the corresponding input buffers of all switches above the first row. Similarly, for the X dimension, the congestion value is determined by aggregating the values of corresponding input buffers of all switches except the first column. As can be seen in Figure 3.4 both calculated congestion values in the X and Y dimensions contain the congestion information of several common switches, i.e. 5, 6, 7, 9, 10, 11, 13, 14, and 15.

In as much as the comparison is made between the values in X and Y dimensions and both values include the congestion information of common switches, only the values of the switches in the row 0 and column 0 have a strong impact on routing decisions. Another shortcoming of the RCA method is that it considers the statuses of all switches along each direction whether they are located in the minimal path or not. Therefore, this method introduces excessive information, which may degrade performance, especially when traffic is mostly local [43].

According to the above explanation, the packet is sent to the X dimension from the source switch since the congestion statuses of the switches 1, 2, and 3 are less than those of the switches 4, 8, and 12. The congestion statuses of the other switches (i.e. 5, 6, 7, 9, 10, 11, 13, 14 and 15) are compared with each other, thus not affecting the comparison result considerably. On the other hand, for an instance, the routing decision is the same wherever the destination is located in the northeast position of the source switch (i.e. the switches 5, 6, 7, 9, 10, 11, 13, 14, and 15). Therefore, when the destination is at the switch 5, the routing unit not only compares the congestion values of the switches 1 and 4, but also considers the congestion values of all the other switches which are not located between the source and destination switches.

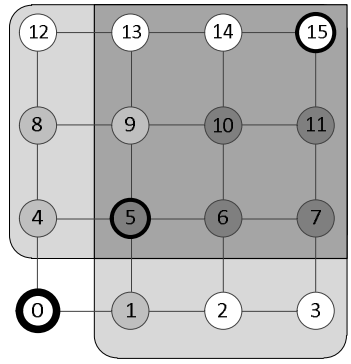

Figure 3.4: An example of the RCA method

3.1.5 Destination-Based Adaptive Routing (DBAR)

The Destination-Based Adaptive Routing method (DBAR) is described in [43] under the assumption of an 8×8 mesh network. In this method, each switch has two 9-bit registers, called congestion-X and congestion-Y. These registers are used to store the congestion information received from the X and Y dimensions. Each entry in the congestion-X (congestion-Y) register is associated with one switch in that row (column) so that the register represents the statuses of all input buffers along that row (column). The congestion information of an input buffer is transferred through a wire to all the other switches along that row or column. In an 8×8 mesh network, eight congestion wires are used along each row and column. The DBAR method only considers the congestion value of the switches that are located in the minimal path between source and destination switches.

DBAR has several shortcomings as follows: 1- As the network size increases, the register size in each switch should also be changed, indicating that the DBAR method is not a scalable approach. 2- Comparison is unfair when a packet has different distances along the X and Y dimensions to the destination switch. For an instance in Figure 3.5, the problem of unfair decision can be seen when a packet is sent from the source 0 to the destination 13. DBAR considers the congestion values of the switches along the X and Y dimensions that reside in the minimal path to the destination switch. However, as shown in Figure 3.5, the congestion condition of a single switch (i.e. the switch 1) in the X dimension is compared with the congestion values of three switches (i.e. the switches 4, 8, and 12) in the Y dimension. Thereby, in most cases, the packet is sent to the switch 1. This unfair comparison results in limiting the packet adaptivity in the remaining path to the destination. In other words, the packet has to be routed through one path from the switch 1 to the switch 13 and it has no alternative routing choices.

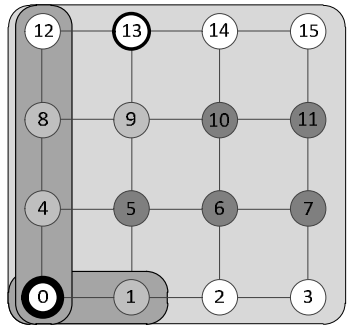

Figure 3.5: An example of the DBAR method

3.1.6 Q-Learning Approach with Reduced Table Sizes (C-Routing)

Learning-based models have been studied in several works such as [50] and [51], but they have rarely been investigated in the context of on-chip networks [52]. In learning-based approaches, the network condition is learned at run time which is utilized in the routing decision. Each switch has a routing table maintaining the latency value from the current switch to each destination switch in the network. Therefore, a large table is required at each switch which may not be an efficient solution for NoCs. An example is shown in Figure 3.6, where a packet is sent from the source switch 0 to the destination switch 10. The table at the switch 0 maintains the latency values to reach from this switch to different destination switches (i.e. the switch 1 to the switch 15) through the possible output channels (i.e. E, W, N, or S output channel). Let us assume the packet is delivered to the east direction as it shows a smaller latency. The packet has to be waited in the input buffer of the switch 1 to receive an output channel. In this example, the packet is sent to the north direction. At this time, the waiting time of the packet in the input buffer of the switch 1 and the estimated latency from the switch 1 to the destination switch 10 are summed up and the obtained value is sent to the switch 0 where the table 0 is updated with this new

information. This procedure is repeated until the packet reaches its final destination. This example shows that as more packets are propagated inside the network, the values of the tables are more up-to-date.

C-Routing [53] is introduced for NoCs with smaller tables. In this method, all switches are divided into several clusters. Tables maintain the latency values from the current switch to other switches within its cluster and from the current switch to different clusters. This approach is not scalable and the area overhead is still high in comparison with other congestion-aware methods.

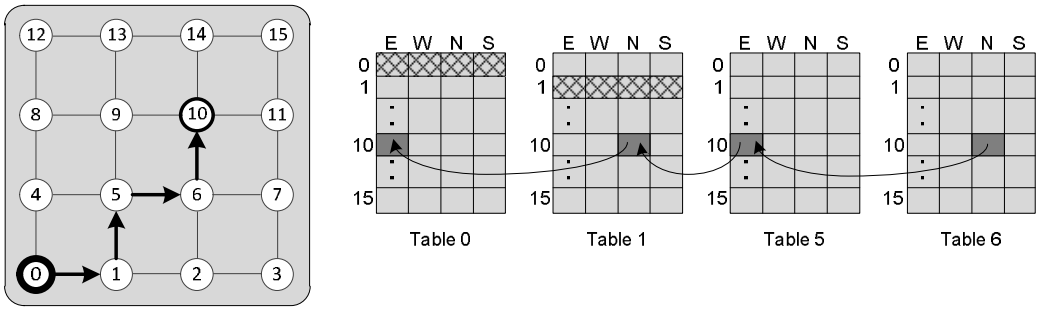

Figure 3.6: An example of the learning-based approach

3.1.7 Summary of Traditional Methods

Among the mentioned algorithms, the routing decision in DyXY is based on purely local information. EDXY, NoP, and DBAR take more global information into account. RCA collects the congestion information throughout the network, but this information cannot be efficiently used in the routing unit and may result in non-optimal decisions. C-Routing is a learning-based approach which imposes a large area overhead.

3.2 The Proposed Cluster-based Approaches

A common assumption among the mentioned traditional algorithms is that congestion is measured at a switch level. In this section, we show that the performance can be improved if the congestion level is obtained for a group of switches, called a cluster, rather than a single switch. We propose two algorithms based on cluster-based schemes, called Agent-based Routing Algorithm (AgRA) and Trapezoid-based Routing Algorithm (TRA).

In the agent-based method, a lightweight clustering structure is built upon the mesh network to propagate the congestion information over the different regions of the network. It employs an efficient selection method to choose between the output channels provided by the routing function. This approach suggests an efficient solution to provide a better view of the network traffic and to utilize this knowledge in the routing decision. TRA investigates the impact of both the routing unit and the switch arbitration unit in distributing the traffic load over the network. In the proposed method, the congestion information about the remaining path is used in the routing unit while the information about the traversed path is utilized in the switch arbitration unit. The non-local information about the remaining path

is gathered from the switches that not only are more likely to be chosen as intermediate switches in the routing path but also provide up-to-date information for a given switch. Moreover, to collect and deliver the non-local information, a distributed propagation mechanism is presented.

A basic requirement to distribute traffic over the network is to give packets some degree of freedom in choosing among different routes. Fully adaptive routing algorithms satisfy this condition well by providing the maximum degree of adaptiveness for packets. The two presented cluster-based methods are based on the fully adaptive method, enabling to select between all minimal directions at each switch. For this purpose one and two virtual channels are used along the X and Y dimensions, respectively. To avoid deadlock, similar to the methods in [38] and [39], eastward and westward packets use the first and second virtual channel, respectively, along the Y dimension while northward and southward packets can take either virtual channel.

3.2.1 Agent-based Routing Algorithm (AgRA)

Figure 3.7(a) shows the Agent-based Network-on-Chip (ANoC) structure in a 2D mesh network [54]. The network is divided into several overlapped clusters in which a cluster contains four switches and a cluster agent. The design consists of two separate mesh networks: data network and lightweight agent network. The data network connects the switches to each other to propagate packets over the network while in the agent network, cluster agents communicate with each other to spread the congestion information. Each cluster agent performs three simple tasks. First, it collects and aggregates the congestion information from the local switches; second, it distributes the information to the neighboring cluster agents; third, the cluster agent transfers the information to the local switches (i.e. the information includes both local cluster information and those received from the neighboring clusters). In addition to this information, a switch receives 1-bit information from each neighboring switch about the buffer availability at the corresponding input port. Accordingly, each switch is aware of the congestion condition of the switches located in the local and neighboring clusters and the congestion condition of the input buffer in the neighboring switches.

As illustrated in Figure 3.7(b), the Congestion Level of a switch (CL) is computed using the Congestion Statuses of input buffers (CS). In each flit event (i.e. flit_tx or flit_rx), if the number of occupied slots of an input buffer is larger (smaller) than a threshold value, the threshold signal is assigned to one, otherwise zero. A history-based scheme is used to capture the threshold signal of an input buffer. For this purpose a 4-bit shift register is adopted to store the threshold signal whenever a new flit enters or leaves the buffer. The CS signal is asserted if all bits of the shift register are one. Finally, the CL value of a switch is computed by summing up the CS signals received from the input buffers. In fact, the CL value of each switch indicates its load level. For example, if only the east and local input buffers of a switch are congested (CS(local)=1 and CS(east)=1), then the CL value of the switch will be two. By using one and two virtual channels along the X and Y dimensions, three bits are enough to code the congestion level of a switch having the maximum of seven

input buffers (i.e. L, E, W, N1, N2, S1, and S2 where numbers refer to virtual channels). As the channel width of cluster agents is identical to the CL value, CL can be transferred within one clock cycle.

To explain the propagation strategy, let us consider the central cluster (i.e. cluster 4) in Figure 3.7(a). The cluster agent 4 receives the 3-bit CL value from each of the four local switches (i.e. the switches 5, 6, 9, and 10). The obtained 12-bit information should be transferred to the local switches and neighboring clusters, but it does not require transferring the whole information to all of them. For example, a local switch knows about its CL value and it only needs to be informed about the CL values of the other three local switches. In another example, when transferring the information from the cluster agent 4 to the cluster agent 5, only the CL values of the switches 5 and 9 are delivered since the cluster 5 receives the CL values of the switches 6 and 10 by its local connections. The similar principle is applied to other directions when transferring the information from the cluster agent 4 to cluster agents 7, 3, and 1.

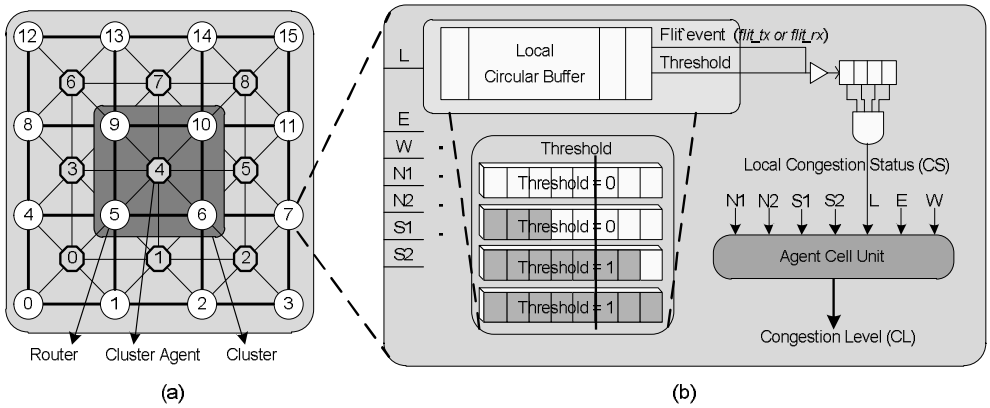

Figure 3.7: Agent-based Networks-on-Chip

By distributing the congestion information over the network, the routing decision can be assisted by the local and non-local congestion information received from the neighboring cluster agents. For example, the switch 0 in Figure 3.7 not only knows the CL value of the switches within its cluster (i.e. the switches 1, 4 and 5) but also have the information about the switches in the clusters C1 and C3 (i.e. the switches 2, 6, 8, and 9). The congestion-aware selection algorithm in a 2D mesh network is shown in Figure 3.8. The routing decision is made based on the relative position of the source and destination switches using the following rules.

- **Rule1-** If the source and destination are located in the same row or column *(delta-X=0 or delta-Y=0)*, the packet has no adaptivity and it is delivered through the only possible direction. For instance, in Figure 3.9(a) when the packet is currently at the switch 0 and the destination is located at the switch 1, 2, or 3, the packet has to be routed along the X dimension while the packet is sent through the Y dimension when the destination is located at the switch 4, 8, or 12.

- **Rule2-** If the distance along both directions is one *(delta-X=1 and delta-Y=1)*, the free buffer slots in the neighboring input buffers (*FreeSlots_Input*) are compared together and the packet is sent to the less congested direction. For example, in Figure 3.9(b) when the destination is at the switch 5, the number of free slots at the west input buffer of the switch 1 is compared with the south (virtual channel 1) input buffer of the switch 4.

- **Rule3-** If the distance along one dimension is one and along the other dimension is greater than one *((delta-X=1 and delta-Y>1) or (delta-X>1 and delta-Y=1))*, at first the *FreeSlots_Input* value along both directions is compared together. In a case that the congestion values in both directions are the same, the CL value of 1- and 2-hop neighbors (pair) is taken into consideration. For example, when the destination is located at the switch 9 or 13 in Figure 3.9(c), at first the *FreeSlots_Input* values of the switches 1 and 4 are compared together. If they are the same, the congestion value of the pair (1,5) is compared with the pair (4,8). The congestion of each pair is called Pair-Congestion. Similarly, Rule3 is applied in Figure 3.9(d) when the destination is located at the switch 6 or 7.

- **Rule4-** If the distances along both directions are greater than or equal to two *(delta-X>=2 and delta-Y>=2)*, after checking the congestion values of the input buffers in the neighboring switches, the congestion level of the neighboring clusters are compared together. The congestion on these clusters is called Cluster-Congestion. For instance in Figure 3.9(e), at first the input buffers of the switches 1 and 4 are compared, then the congestion values of clusters 1 and 3 are checked.

```
ALGORITHM: Congestion-aware selection method in the agent-based approach
                    **-------------------------------------------------**
Definitions:    FreeSlots_Input: The number of free buffer slots in the input buffer
                Cluster-Congestion: Congestion value of a cluster
                Pair-Congestion: Congestion in 1- and 2-hop switches
                delta_x, delta_y: Distance between source and destination
                    **-------------------------------------------------**
if (delta_x=0 and delta_y=0) then
        select the local port;
elseif (delta_x=0 or delta_y=0) then
        select the only possible direction;
elseif (delta_x=1 and delta_y=1) then
        select based on FreeSlots_Input;
elseif (delta_x=1 and delta_y>1) or (delta_x>1 and delta_y=1) then
        select based on FreeSlots_Input and then Pair-Congestion;
elseif (delta_x>=2 and delta_y>=2) then
        select based on FreeSlots_Input and then Cluster-Congestion;
end if;
```

Figure 3.8: The congestion-aware selection algorithm in the agent-based approach

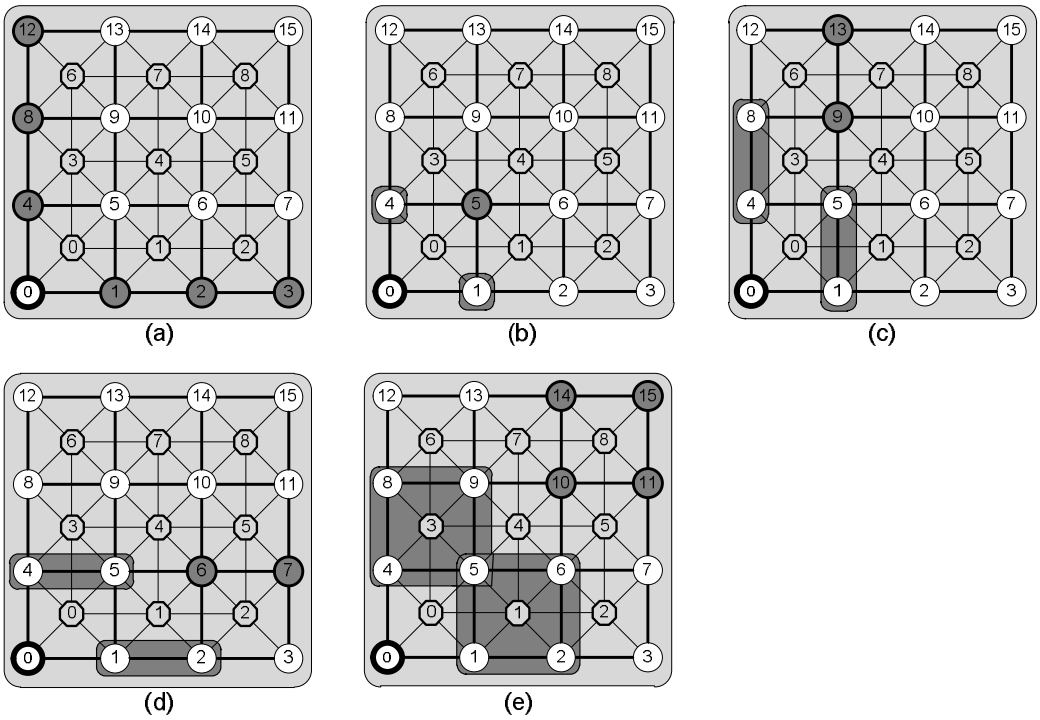

Figure 3.9: An example of the agent-based selection method

3.2.2 Trapezoid-based Routing Algorithm (TRA)

The agent-based approach is a scalable method which can reduce the latency of packets significantly compared with traditional methods. The reason of this reduction is that the agent-based approach has a better overall view regarding the congestion at the neighboring regions while traditional methods have a more limited view as they consider the congestion at the switch level. However, the question is that how the number of switches within each cluster is defined (e.g. four switches within a cluster in the agent-based approach). On the other hand, a major part of research works as well as the presented agent-based method try to alleviate congestion by considering the traffic condition in the forward paths and delivering packets through the less congested paths. In other words, none of them consider the impact of the switch arbitration in traffic distribution.

In this section, we propose a method called Trapezoid-based Routing Algorithm (TRA) [55]. TRA tries to find the optimal number of switches within each cluster. In detail, the basic ideas of TRA can be divided into two parts: input selection function and output selection function [56].

The input selection function of TRA uses a priority-based approach to prioritize packets coming from the congested regions. There are two solutions to estimate the congestion condition of the path from where the packet is coming. The first solution is to carry the

information by the means of packets, so that as a packet moves toward its destination, the information of the traversed path is stored in the header flit of the packet. In this way, a packet not only is responsible for delivering data but also carrying the experienced congestion value along its path to the destination. The main drawback of this approach is the increased per-hop latency due to updating the header flit with the new congestion information whenever a packet leaves an intermediate switch.

The second solution is to propagate the information of nearby switches by using distinct wires. Since packets do not carry the path information, the traversed path should be estimated according to the position of the source and current switches. This estimation is rather impractical considering all possible paths between the source and the current switch. However, it is enough to consider the limited area residing in the packet quadrant rather than the switches that have been passed by the packet. The imposed wiring overhead is the shortcoming of this approach. In TRA, the second solution is employed. To reduce the area overhead, TRA utilizes a distributed propagation mechanism to collect and distribute the traffic information of distant switches. This suggests a relatively low wiring overhead, latency and power consumption required for propagating traffic information.

The output selection function of TRA utilizes the most validated and up-to-date congestion information in the routing decision unit. The aim of the output selection function of TRA is to gather the congestion information from the switches that not only are more likely to be chosen as intermediate switches in the routing path but also provide up-to-date information for a given switch. By calculating the passing probability of packets through the switches, we find out that the congestion information of the switches located in the trapezoid-based region is sufficient to make optimal routing decisions.

3.2.2.1 Output Selection Function

For a given source and destination pair in an adaptive routing, some intermediate switches are passed by more number of packets than others. In general, central switches forward significantly a larger number of packets than edge switches since central switches are accessible through more paths. One of the aims of the TRA method is to make the routing decision based on the congestion conditions of the switches through which more packets are forwarded toward destinations. For example, consider a case in Figure 3.10 where a packet is sent from the switch 0 to the destination 24 when the network is not congested. At the source switch, the packet can be sent through the switch 1 or 5 with 50% probability each. When the packet arrives at the switch 1, there are two choices for the next hop, the switch 2 or 6. The probabilities that a packet passes through them are 25% each (the switch 6 is accessible through the switches 1 and 5, so the passing probability of packets through this switch is 50%). The arrived packet at the switches 2 and 6 can be delivered to the switch 7, thereby with a probability of 37.5% the packet passes through the switch 7, and so on. Similar to the DBAR technique, the routing decision can be assisted by the statuses of the switches explicitly in the row or column (i.e. the switches 1, 2, 3, and 4 in the row; and the switches 5, 10, 15, and 20 in the column), while the passing probabilities of packets through the switches along a row or column are 50%, 25%, 12.5%, and 6.25%,

respectively. Obviously, these switches are in the 1-, 2-, 3-, and 4-hop distances from the source switch. By continuing along a direction, not only the passing probability of packets through the switches approaches zero but also the congestion statuses of the switches would be out-of-date for the source switch.

In general, traditional methods do not efficiently improve the load balance due to the lack of knowledge about the congestion statuses of the switches to be likely passed in the routing path. By calculating the passing probability of packets through the switches, in the TRA method, the congestion statuses of the switches with the high passing probability (trapezoid positions) are analyzed. Trapezoid positions for the X and Y dimensions (X-trapezoid and Y-trapezoid) are shown in Figure 3.10. The delivered packets from the switch 0 may pass the switches in the X-trapezoid positions, the switches 1, 2, 7, 8, and 3 with a probability of 50%, 25%, 37.5%, 25%, and 12.5%, respectively. The distances of these switches from the source switch are 1, 2, 3, 4, and 3. So, the probability of the switches 7 and 8 being passed by the packets from the switch 0 is higher than either the switch 3 or the switch 4 in the axis. Although the switches 6 and 12 can be used in the routing path with a probability of 50% and 37.5%, respectively, their congestion statuses cannot affect the comparison result at the switch 0 as they have similar effect on the obtained congestion values in both directions. The other switches are either far from the source switch or have a little chance to be intermediate switches through which packets are forwarded.

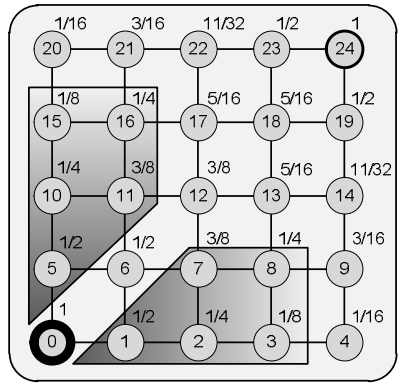

Figure 3.10: Passing-probability of packets through intermediate switches

Two 5-bit registers, called CR_x and CR_y, are used to store the congestion values of the switches in the trapezoid positions. Each bit of these registers is allocated to one switch in the trapezoid position and the orders (i.e. most significant bit to least significant bit) are determined based on the passing probability of packets through the switches and their distances from the source switch. For example, in Figure 3.10 the bits of CR_x and CR_y are assigned as CR_x=(switch 1; switch 2; switch 7; switch 8; switch 3) and CR_y=(switch 5; switch 10; switch 11; switch 16; switch 15). This suggests that for a given source and destination pair, the congestion statuses of nearby switches with large passing probabilities influences the routing decision more than those of long-distance switches.

The pseudo code of the output selection function is presented in Figure 3.11. According to this algorithm, when a packet is in the same row or column as the destination switch, the packet is sent to the appropriate direction. However, if the packet is one hop apart from the destination row or column, only the congestion conditions of the adjacent switches are used in the routing decision. In this situation, if the congestion statuses of both neighboring switches are the same, the packet is sent to the neighboring switch that probably is not located in the destination row or column. This selection retains the packet adaptivity in the next hop. When the packet is two hops away from the destination row or column, the algorithm ignores the statuses of the switches that are not located in the minimal path to the destination switch. In other cases, CR_x and CR_y are compared with each other. Since, in all cases, the congestion conditions of the switches in the minimal paths are considered, this algorithm offers an efficient approach to application's isolation, when different applications are mapped to multiple regions of the network. By integrating all the other trapezoid positions, the congestion information of the highlighted regions in Figure 3.12 is needed by a switch in a central part of the network.

```
ALGORITHM: Output selection function of TRA
                        **-------------------------------------------------**
Definitions:    CRx: Congestion in the X-trapezoid Position (CRx=register(4 downto 0))
                CRy: Congestion in the Y-trapezoid Position (CRy=register(4 downto 0))
                Xs,Ys: X and Y coordinates of the source switch
                Xd,Yd: X and Y coordinates of the destination switch
                        **-------------------------------------------------**
delta_x=Xd-Xs; delta_y=Yd-Ys;
if delta_x=0 or delta_y=0) then
        if (delta_x=0 and delta_y=0) then select <= local;
        elsif (delta_x=0) then select <= y_dir;
        else select <= x_dir;
        end if;
elsif (delta_x=1 or delta_y=1) then
        if (CRx(4)>CRy(4)) then select <= y_dir;
        elsif (CRx(4)<CRy(4)) then select <= x_dir;
        elsif (CRx(4)=CRy(4)) then select<= greater-distance dimension;
        end if;
elsif (delta_x=2 or delta_y=2) then
        if (CRx(4 downto 2)>CRy(4 downto 2)) then select <= y_dir;
        elsif (CRx(4 downto 2)<CRy(4 downto 2)) then select <= x_dir;
        elsif (CRx(4 downto 2)=CRy(4 downto 2)) then select<= greater-distance dimension;
        end if;
else
        if (CRx>CRy) then select <= y_dir;
        elsif (CRx<CRy) then select <= x_dir;
        elsif (CRx=CRy) then select<= greater-distance dimension;
        end if;
end if;
```

Figure 3.11: The output selection function of TRA

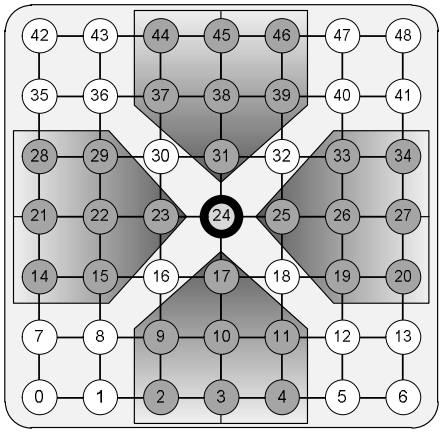

Figure 3.12: The congestion information of the highlighted regions is needed at the switch 24 to perform the output selection function of TRA

3.2.2.2 Input Selection Function

Let us explain the idea of the input selection function through the example of Figure 3.13(a) where a darker color indicates a higher congestion level. Three packets reach the switch 12; two generated at the switch 0 and one at the switch 4. These packets arrive from the west, south and east input buffers of the switch 12 and request the north output channel. If a fair arbitration mechanism is used at the switch 12, the packet from the congested region C1 has the same chance to win switch arbitration as the packets from the less congested regions C2 and C3. Accordingly, the switch 12 can be a bottleneck for the packets leaving the congested region. This bottleneck problem can be alleviated by giving a higher priority to the packets arrived from the region C1 to access the output port of the switch 12. In contrast, packets arriving from the regions C2 and C3 should wait in the input buffers for an extended period of time before accessing the output channel, which slightly increases the congestion condition of the switches in the regions C2 and C3. In other words, traffic of the highly congested regions is steered to low congested ones. Figure 3.13(b) depicts the spread of congestion when packets arriving from the congested regions (e.g. C1) get higher chance to win the arbitration than the other packets (e.g. C2 and C3). Such priority-based arbitration is performed similarly in all switches. In sum, the input selection function of the TRA method allows the congested switches to forward their buffered packets rapidly. The input selection of TRA requires the congestion information of static groups of four switches. In Figure 3.14, the highlighted region around the central switch 24 indicates the switches whose congestion information is required by the input selection function of the switch 24 when considering all possible positions of the source switch.

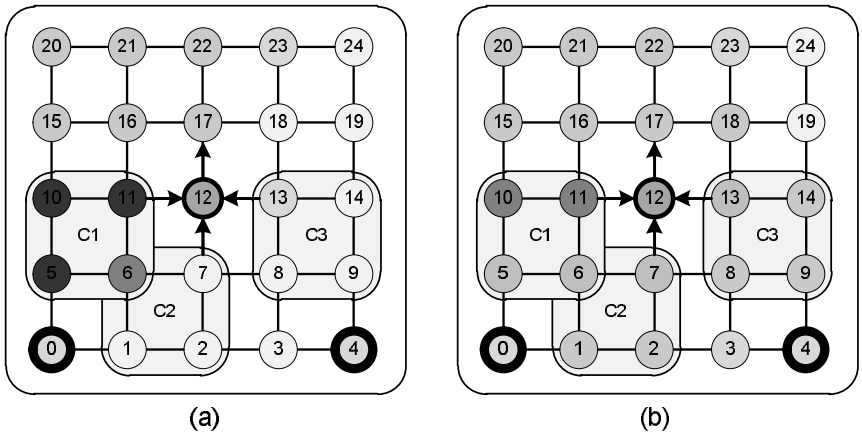

Figure 3.13: Input selection function of TRA

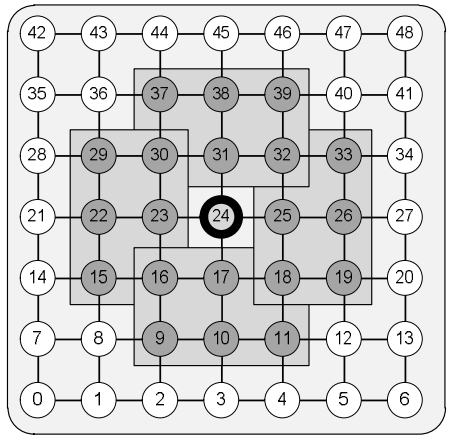

Figure 3.14: The congestion information of the highlighted regions is needed at the switch 24 to perform the input selection function of TRA

According to the source switch position, the congestion values of some specific switches are combined together and associated with the packet. This value can be considered the priority of the packet in the switch arbitration unit. For the ease of understanding, each switch is assigned a label shown in Figure 3.15, for example, the label *nnw* means that the switch is located in the north-north-west position of the current switch. In the algorithm of Figure 3.16, congestion values are determined and assigned to packets.

The priority of a packet is determined based on the congestion values of the nearby switches located between the source and the current switches. For example, if the packet is generated at the north neighboring switch *(delta-X=0* and *delta-Y=1)*, only the congestion value of the north neighboring switch is considered for the packet. Similarly, when the distance from the current to the source switch along the X and Y dimensions is zero and

two, respectively *(delta-X=0* and *delta-Y>=2)*, the congestion values of the switches in 1- and 2-hop neighbors in the north direction are considered for the packet. When the distance along the Y dimension is one and along the X dimension is greater than one *(delta-X>=1* and *delta-Y=1)*, the congestion values of the switches at north and north-east positions are used for the packet (similarly, north and north-west positions when *delta-X<=-1* and *delta-Y=1*). Finally, when the distance along the X dimension is equal or greater than one and along the Y dimension is equal or greater than two (*delta-X>=1* and *delta-Y>=2*), the congestion values of four switches located at north, north-north, north-east, and north-north-east positions are considered for the packet (similarly, north, north-north, north-west, and north-north-west positions when *delta-X<=-1* and *delta-Y>=2*). A similar principle is applied to assign a congestion value to packets coming from the east, west, and south input ports.

Obviously, depending on the source and current switch positions, the congestion information of a various number of switches contributes to the calculation of packets' priority. To have a consistent value for the comparison purposes, the calculated value should be divided by the number of switches involved to obtain that value. As shown in Figure 3.16, the number of contributing switches is limited to one, two and four, and hence the division function can be easily implemented by a shift register.

The input selection function examines the priority value of all incoming packets and gives a grant to a packet with the highest congestion value. In order to prevent starvation, each time after finding a packet with the highest value, the priorities of other packets are incremented. Figure 3.17 shows the pseudo code of the priority-based input selection function.

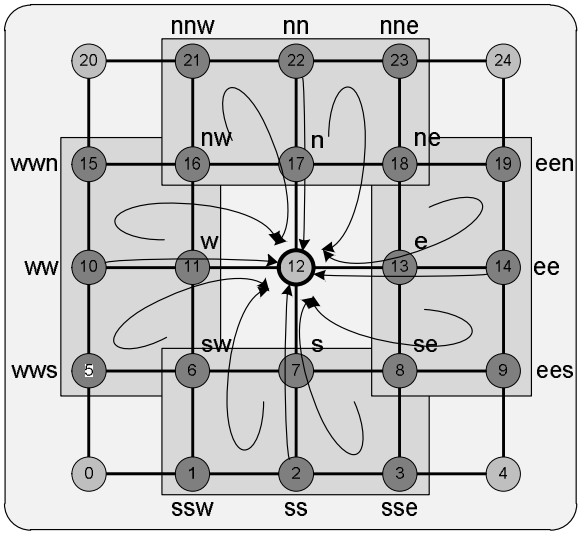

Figure 3.15: Assigning a name to each switch

```
ALGORITHM: Assigning congestion value to packets
                    **-------------------------------------------------**
Definitions:    Xc,Yc, Xs,Ys: X and Y coordinates of the current and source switches
                pc, rc: Packet Congestion, Region Congestion
                n,s,e,w: North, South, East, West
                    **-------------------------------------------------**
delta_x=Xc-Xs; delta_y=Yc-Ys;
if (inputBuffer=north) then
        if (delta_x= 0 and delta_y= 1) then pc <= rc(n);
        elseif (delta_x= 0 and delta_y>= 2) then pc <= rc(n,nn)/2;
        elseif (delta_x>= 1 and delta_y= 1) then pc <= rc(n,ne)/2;
        elseif (delta_x<= -1 and delta_y= 1) then pc <= rc(n,nw)/2;
        elseif (delta_x>= 1 and delta_y>= 2) then pc <= rc(n,nn,ne,nne)/4;
        elseif (delta_x<= -1 and delta_y>= 2) then pc <= rc(n,nn,nw,nnw)/4; end if;
elseif (inputBuffer=south) then
        if (delta_x= 0 and delta_y= -1) then pc <= rc(s);
        elseif (delta_x= 0 and delta_y<= -2) then pc <= rc(s,ss)/2;
        elseif (delta_x>= 1 and delta_y= -1) then pc <= rc(s,se)/2;
        elseif (delta_x<= -1 and delta_y= -1) then pc <= rc(s,sw)/2;
        elseif (delta_x>= 1 and delta_y<= -2) then pc <= rc(s,ss,se,sse)/4;
        elseif (delta_x<= -1 and delta_y<= -2) then pc <= rc(s,ss,sw,ssw)/4; end if;
elseif (inputBuffer=east) then
        if (delta_x= 1 and delta_y= 0) then pc <= rc(e);
        elseif (delta_x>= 2 and delta_y= 0) then pc <= rc(e,ee)/2;
        elseif (delta_x= 1 and delta_y>= 1) then pc <= rc(e,en)/2;
        elseif (delta_x= 1 and delta_y<= -1) then pc <= rc(e,es)/2;
        elseif (delta_x>= 2 and delta_y>= 1) then pc <= rc(e,ee,en,een)/4;
        elseif (delta_x>= 2 and delta_y<= -1) then pc <= rc(e,ee,es,ees)/4; end if;
elseif (inputBuffer=west) then
        if (delta_x= -1 and delta_y= 0) then pc <= rc(w);
        elseif (delta_x<= -2 and delta_y= 0) then pc <= rc(w,ww)/2;
        elseif (delta_x= -1 and delta_y>= 1) then pc <= rc(w,wn)/2;
        elseif (delta_x= -1 and delta_y<= -1) then pc <= rc(w,ws)/2;
        elseif (delta_x<= -2 and delta_y>= 1) then pc <= rc(w,ww,wn,wwn)/4;
        elseif (delta_x<= -2 and delta_y<= -1) then pc <= rc(w,ww,ws,wws)/4; end if;
end if;
```

Figure 3.16: Assigning congestion values to packets

```
ALGORITHM: Priority-based input selection function
                    **-------------------------------------------------**
Definitions:    i: i(th) input channel; c: congestion; w: weight
                    **-------------------------------------------------**
for i=0 to all input buffers loop
        if packet in input(i) is newly arrived then w(i) <= 0;
        else    w(i) <= w(i) + 1;
        end if;
        if w(i) + c(i) > maxPriority then
                maxPriority <= w(i) + c(i);
                select  <= i;
        end if;
end loop;
```

Figure 3.17: The pseudo code of the priority-based input selection function of TRA

3.2.2.3 A General Example

Figure 3.18 presents a general example of TRA when a packet is routed from the source switch 24 to the destination switch 6. In this figure, the congestion values of dotted and highlighted regions are used for distinguishing between input and output selection functions, respectively. In the input selection function of Figure 3.18(a), the generated packet at the switch 24 is in a competition with other packets to access an output port. The priority of this packet is determined based on the congestion level of the switch 24. In the output selection function, the west and south neighbors are detected as the possible choices to forward the packet. Since the packet is far from the destination switch, the congestion information of all switches in the trapezoid positions is used in the routing function (CR_x,CR_y). Let us assume that the packet is delivered to the switch 23 (Figure 3.18(b)).

In the input selection function of the switch 23, the priority of the packet is determined again based on the congestion value of the switch 24. In the output selection function, since the switches 15 and 20 are located outside of the minimal path, their congestion values are ignored in the routing decision, and for a fair comparison, the congestion statuses of the switches 7 and 8 are also disregarded (CR_x(4 downto 2), CR_y(4 downto 2)). The purpose of this policy is to compare the congestion information of the same number of switches in each direction. When the packet reaches the switch 18 (Figure 3.18(c)), the similar comparison mechanism is used (CR_x(4 downto 2), CR_y(4 downto 2)). In the input selection function, since the packet arrives from the north direction and the distances from the source switch are one hop along the X and Y dimensions, according to the algorithm shown in Figure 3.16, the congestion values of the switches in the north (the switch 23) and northeast (the switch 24) positions are considered. When a packet reaches the switch 17 (Figure 3.18(d)), the priority value of the packet is determined based on the congestion values of four switches located at the east (the switch 18), east-east (the switch 19), east-north (the switch 23), east-east-north (the switch 24) positions. In the output selection function, as the packet is currently one hop away from the destination column, only the congestion value of the adjacent switches 12 and 16 are compared together (CR_x(4), CR_y(4)). If the congestion conditions are the same in both the X and Y dimensions, the packet is delivered to the switch 12. This decision is made since the packet loses alternative choices in the remaining path if the packet is sent to the switch 16. The input and output selection functions of Figure 3.18(e) are similar to Figure 3.18(d). In Figure 3.18(f), the priority of the packets is determined based on the congestion information of four switches while the output selection function delivers the packet to destination.

3.2.2.4 Propagation Mechanism

To perform the input selection and output selection functions of TRA, the congestion information of some group of switches should be available at the current switch. Figure 3.12 and Figure 3.14 show the switches whose congestion information should be available at the switch 24 to perform input and output selection functions, respectively. In total, the congestion information of the highlighted switches in Figure 3.19 should be provided for the switch 24 according to the TRA method.

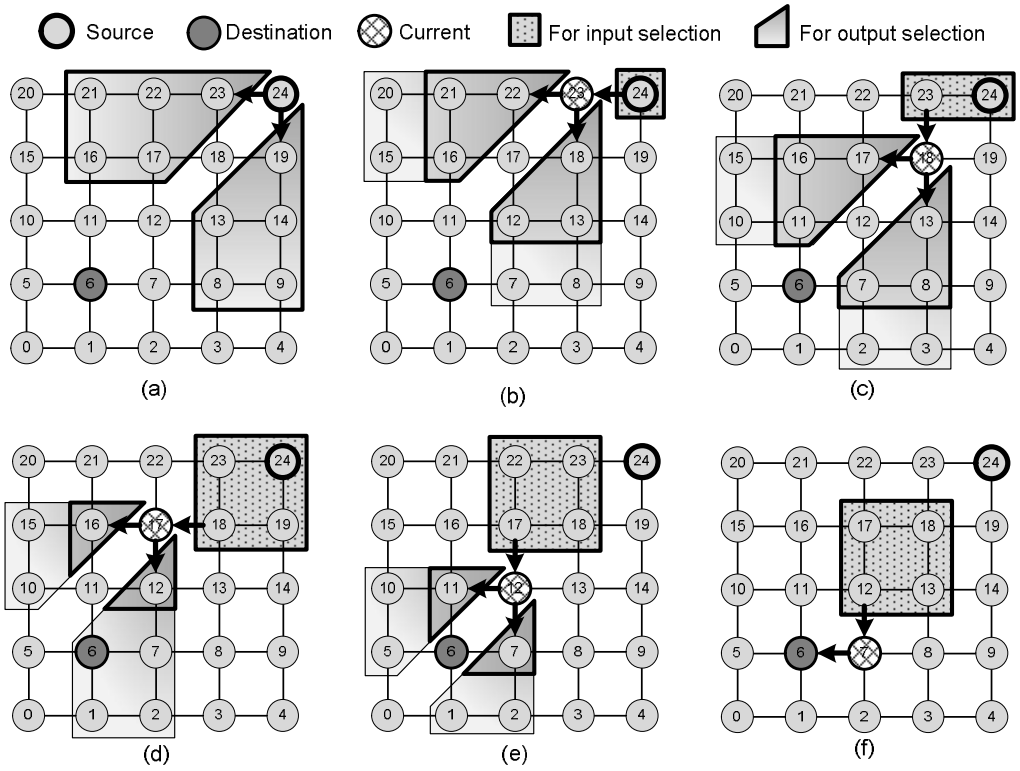

Figure 3.18: A general example of combining the input and output selection functions of TRA

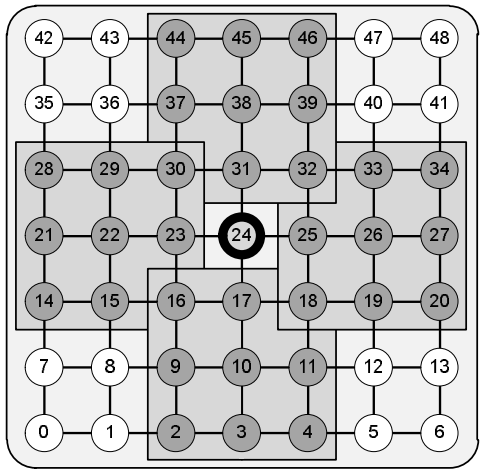

Figure 3.19: The required information at the switch 24 to perform the input and output selection functions

A propagation mechanism is needed to provide the congestion information of the required regions for every switch in the network. To do this, as shown in Figure 3.20(a) and Figure 3.20(b), each switch calculates its congestion value and aggregates it with two 1-bit values of the neighboring switches in the Y dimension and then propagates the whole information along the X dimension. This information should be transferred at most three hops away from where it is generated. A similar methodology is applied to transfer the information along the Y dimension (Figure 3.20(c) and Figure 3.20(d)).

Let us consider an example in Figure 3.20(a) where the congestion values of the switches in the westward region are delivered to the switch 24. To transfer the information, the congestion values of the switches 14 and 28 are sent to the switch 21. The congestion values of the switches 14 and 28 are aggregated with the congestion value of the switch 21 before transmitting the whole information to the switch 22. At the switch 22, the received 3-bit congestion information is combined with those of values from the switches 15, 22, and 29 and the 6-bit information is delivered to the switch 23. Finally, arrived information at the switch 23 is merged with the values of the switches 16, 23, and 30 and the resulted 9-bit information is transferred to the switch 24. In this way, the switch 24 is informed about the congestion values of the switches 14, 15, 16, 21, 22, 23, 28, 29, and 30. Figure 3.20(b), Figure 3.20(c), and Figure 3.20(d) show the similar situations when the information is gathered from eastward, northward, and southward regions. Figure 3.21 illustrates the required number of bits to propagate the congestion information throughout a 4×4 mesh network. The minimum number of bits per unidirectional link is 3 bits while the maximum is 9 bits. As the network size enlarges, the average number of bits per link approaches 9 bits.

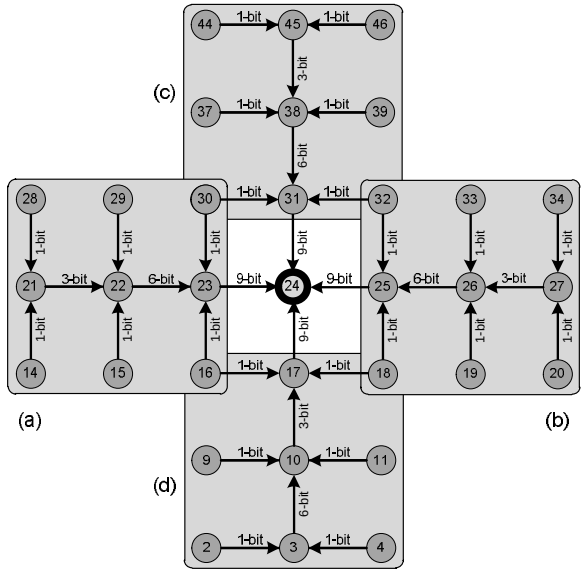

Figure 3.20: Information from (a) west- (b) east- (c) north-, and (d) south-ward regions to the switch 24

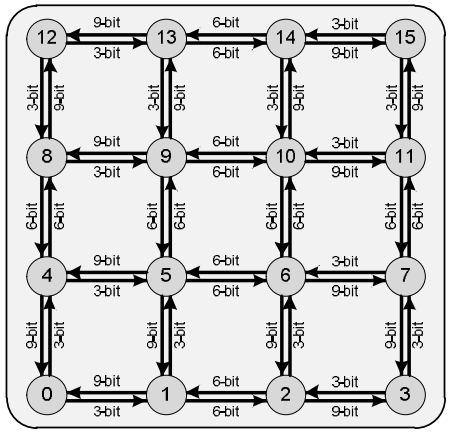

Figure 3.21: The required number of bits to propagate the congestion information

3.2.3 Results and Discussion

The efficiency of the proposed routing schemes (i.e. AgRA and TRA) are compared with two other proposals NoP [40] and DBAR [43] selected from the traditional methods (Section 3.1.3 and 3.1.5). For fairness, all of these methods utilize a fully adaptive routing function based on the DyXY method [57] using one and two virtual channels along the X and Y dimensions, respectively. A NoC simulator is developed with VHDL to model all major components of the on-chip network. Simulations are carried out to determine the latency characteristic of each network. As a performance metric, we use latency defined as the number of cycles between the initiation of a packet by a Processing Element (PE) and the time when the packet is completely delivered to the destination PE. The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles. For all switches, the data width is set to 32 bits. Each input buffer contains 8 slots. Moreover, the packet length is uniformly distributed between 1 and 10 flits. The congestion threshold value is set to 6. It is worth mentioning that in this thesis, wire delay and power consumption due to wires are not included in the performance and power estimates.

3.2.3.1 Performance Analysis under Uniform Traffic Profile

In the uniform traffic profile, each processing element generates data packets and sends them to another processing element using a uniform distribution [49]. The mesh size is considered 8×8. In Figure 3.22, the average communication latency as a function of the packet injection rate is plotted. As observed from the results, TRA leads to the lowest latency. This is due to the fact that TRA takes advantage of both input and output selection functions in distributing packets while the performance gain in AgRA, DBAR and NoP methods are limited to the output selection function. AgRA performs better than DBAR and NoP as it provides a better view of the network traffic utilizing the clustering approach.

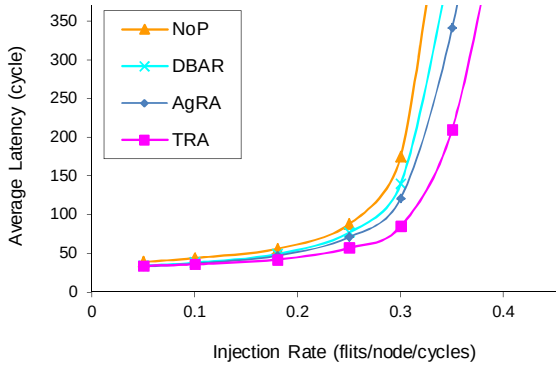

Figure 3.22: Performance analysis in an 8×8 mesh network under uniform traffic profile

3.2.3.2 Performance Analysis under Hotspot Traffic Profile

Under the hotspot traffic pattern, one or more switches are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In simulations, given a hotspot percentage of *H*, a newly generated packet is directed to the hotspot switch with an additional *H* percent probability. We simulate the hotspot traffic with a single hotspot switch at (4,4) in an 8×8 mesh network. The performance with *H=10%* is illustrated in Figure 3.23. Comparing average latency of all methods, it is obtained that TRA performs the best. The reason for this improvement is that traffic can be distributed more efficiently as the routing decision unit is based on the congestion statuses of the switches through which more packets may pass toward destinations. In addition, TRA utilizes an efficient input selection function whereas the AgRA, DBAR and NoP methods are lacking it.

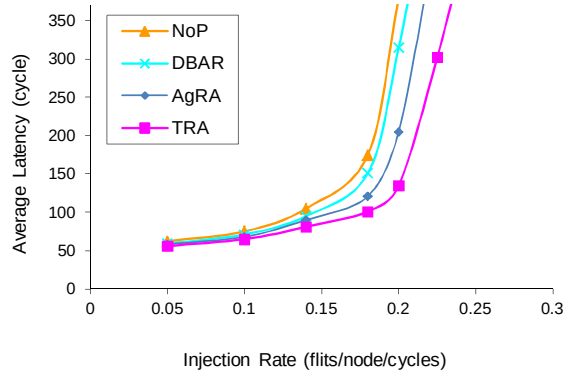

Figure 3.23: Performance analysis in an 8×8 mesh network under hotspot traffic profile with *H=10%*

3.2.3.3 Performance Analysis under Application Traffic Profile

Application traces are obtained from the GEMS simulator [58] using some application benchmark suites selected from SPLASH-2. As listed in Table 3.1, we use a 64-node network configuration: 20 processors and 44 L2-cache memory modules. For the CPU, we assume a core similar to Sun Niagara and use SPARC ISA [59]. Each L2 cache core is 512KB, and thus, the total shared L2 cache is 22MB. The memory hierarchy is governed by a two-level directory cache coherence protocol. Each processor has a private write-back L1 cache (split L1 I and D cache, 64KB, 2-way, 3-cycle access). The L2 cache is shared among all processors and split into banks (44 banks, 512KB each for a total of 22MB, 6-cycle bank access), connected via on-chip switches. The L1/L2 block size is 64B. Our coherence model includes a MESI-based protocol with distributed directories, with each L2 bank maintaining its own local directory. The simulated memory hierarchy mimics SNUCA while the off-chip memory is a 4GB DRAM with a 220-cycle access time.

Figure 3.24 shows the average packet latency across five benchmark traces, normalized to NoP. TRA provides lower latency than other schemes and it shows the greatest performance gain in Radix with 29% reduction in latency. The average performance gain of TRA is up to 23% across all benchmarks vs. NoP and 16% vs. DBAR.

Table 3.1: System configuration parameters

Processor Configuration	
Instruction set	SPARC
Processor, memory	64-node network configuration: 20 processors and 44 L2-cache memory modules
L1 cache	split L1 I and D cache, 64KB, 2-way, 3-cycle access
L2 cache	Shared, 44 banks, 512KB each for a total of 22MB, 6-cycle bank access
Cache coherence protocol	MESI
Cache hierarchy	SNUCA
Size	4GB DRAM
Access latency	220 cycles
Requests per processor	16 outstanding
Benchmarks	SPLASH-2
Network configuration	
Switch scheme	wormhole
Flit size	32 bits
Workloads	
SPLASH-2	Barnes, Radix, Ocean, fft, Cholesky

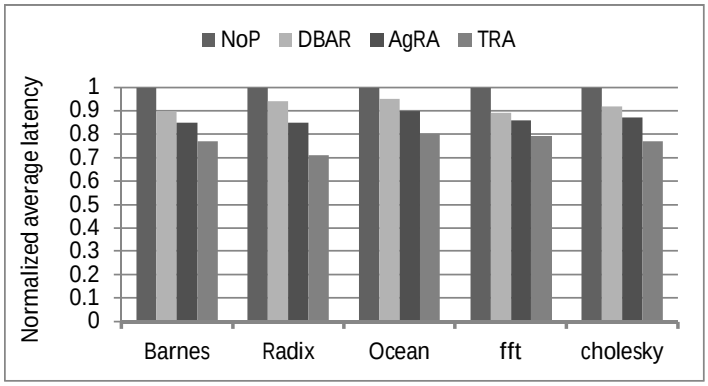

Figure 3.24: Performance analysis under different application benchmarks normalized to NoP

3.2.3.4 Hardware Analysis

To assess the area overhead and power consumption of the proposed scheme, the whole platform of each scheme is synthesized by Synopsys Design Compiler. Each scheme includes network interfaces [2], switches, congestion wires, and communication channels. For synthesis, we use the TSMC 65nm technology at the operating frequency of 500MHz and supply voltage of 1V. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation of each scheme is calculated under the hotspot traffic using Synopsys PrimePower in an 8×8 mesh network. The layout area and power consumption of each platform are shown in Table 3.2. According to this table, the NoP platform consumes more power and has a higher area overhead than other methods. The area overhead and power consumption of DBAR and AgRA are in a same range. The TRA platform occupies more area than both the DBAR and AgRA platforms while its power consumption is comparable with them.

Table 3.2: Hardware implementation details

Network platforms	Area (mm^2)	Power (mw)
NoP	2.611	1.413
DBAR	2.578	1.316
AgRA	2.581	1.277
TRA	2.590	1.268

3.3 The Proposed Non-Minimal and Learning-based Approaches

All of the mentioned traditional methods along with the proposed cluster-based methods are based on using only the shortest paths in the network. In low traffic loads, methods based on the shortest paths can achieve optimized performance, while they are inefficient in avoiding hotspots when the network load increases. The reason for this inefficiency is that

they can deliver packets through at most two minimal directions and they cannot reroute packets around congested regions. The routing policy (output selection) of these methods can be based on local, non-local, or mix of local and non-local congestion information. However, minimal routing algorithms suffer from a low degree of adaptiveness, which are inefficient in distributing traffic over the network even if they have accurate knowledge of the network condition.

The wormhole switching technique eliminates the need to allocate large buffers in intermediate switches along the path. However, a packet waiting to be allocated to an outgoing channel may prohibit other packets from using the channels and buffers and thereby wasting channel bandwidth and increasing latency. Adding virtual channels can alleviate this problem, but it is an expensive solution. Non-minimal methods can partially overcome this blocking problem and reduce the waiting time of packets in input buffers by delivering them through alternative paths. In contrast, performance can severely deteriorate in non-minimal methods due to the uncertainty in finding an optimal path as they may choose longer paths and meanwhile delivering packets through congested regions. Moreover, non-minimal methods can suggest minimal and non-minimal paths between a source and destination but this flexibility is at the cost of a more complex switch structure or additional virtual channels. An output selection function should choose a single channel from a set of predetermined channels to forward a packet to the next hop. As the number of minimal and non-minimal paths might be very large, one of the main challenges involved in designing an efficient non-minimal method is to select a less congested path from a set of alternative paths. The decision for an output channel should not be based on local information as it may route packets through paths which are not only longer but also highly congested. On the other hand, even if a global knowledge of the network is provided, due to a large number of alternative paths, finding a less congested path is questionable which demands an intelligent method to cope with.

In this section, we present a non-minimal routing algorithm for on-chip networks that provides a wide range of alternative paths between each pair of source and destination switches. Initially, the algorithm determines all permitted turns in the network including 180-degree turns on a single channel without creating cycles. The implementation of the algorithm provides the best usage of all allowable turns to route packets more adaptively in the network. On top of that, for selecting a less congested path, an optimized and scalable learning method is utilized. The learning method is based on local and global congestion information and can estimate the latency from each output channel to the destination region.

3.3.1 Highly Adaptive Non-Minimal Routing Algorithm (HARA)

The proposed non-minimal routing algorithm, called Highly Adaptive Non-Minimal Routing Algorithm (HARA) [60], is based on the Mad-y method [57] which has been introduced by Glass and Ni. Mad-y utilizes a double-Y network where the X and Y dimensions have one and two virtual channels, respectively (Figure 3.25(a)). In a 2D mesh network, three types of turns can be taken: 0-degree, 90-degree, and 180-degree turns (U

turns). By taking a 0-degree turn, a packet transmits in a same direction with a possibility of switching between virtual channels. The turn is called 0-degree-ch if in a turn neither the direction nor the virtual channel changes (Figure 3.25(b)) while it is a 0-degree-vc turn if the virtual channel changes (Figure 3.25(c)). By taking a 90-degree turn, a packet transmits between the switches in perpendicular directions (Figure 3.25(d)). By taking a 180-degree turn, a packet is transferred to a channel in the opposite direction. If the virtual channel is changed, the turn is called 180-degree-vc (Figure 3.25(e)); otherwise, it is represented as 180-degree-ch (Figure 3.25(f)). In all figures, the vc1 and vc2 are differentiated by – and = respectively.

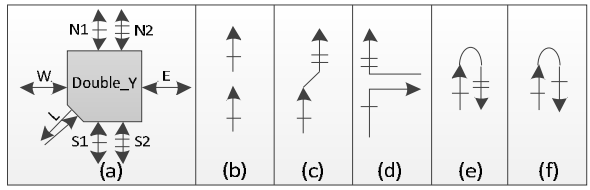

Figure 3.25: (a) A switch in a double-Y network (b) 0-degree-ch (c) 0-degree-vc (d) 90-degree (e) 180-degree-vc (f) 180-degree-ch

In order to avoid deadlock, the Mad-y method [57] prohibits some turns in the double-Y network. As shown in Figure 3.26(c) and Figure 3.26(d), the 0-degree-vc turns from vc1 to vc2 are permitted, also all 0-degree-ch turns are allowable, however 0-degree-vc turns from vc2 to vc1 may cause deadlock in the network and are prohibited. As illustrated in Figure 3.26(a) and Figure 3.26(b), out of sixteen 90-degree turns that can be potentially taken in a network, four of them cannot be taken in Mad-y. Finally, 180-degree turns are not allowed in Mad-y. To prove the deadlock freeness, a two-digit number (a,b) is assigned to each output channel of a switch in a $n \times m$ mesh network. According to the numbering mechanism, a turn connecting the input channel (I_a,I_b) to the output channel (O_a,O_b) is called an ascending turn when ($O_a > I_a$) or (($O_a = I_a$) and ($O_b > I_b$)). Figure 3.27 shows the numbers assigned to each channel for a switch at the position (X,Y). Since this numbering mechanism causes the packets to take the permitted turns in strictly increasing order, so that Mad-y is deadlock-free.

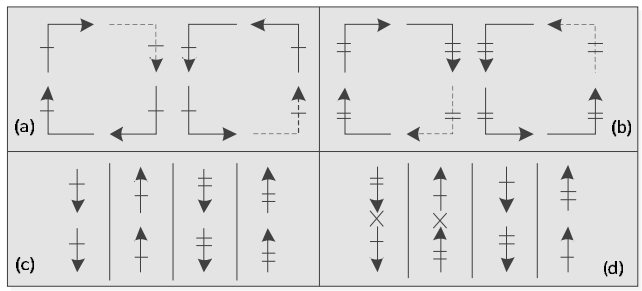

Figure 3.26: (a) 90-degree turns in vc1 (b) 90-degree turns in vc2 (c) 0-degree-ch (d) 0-degree-vc

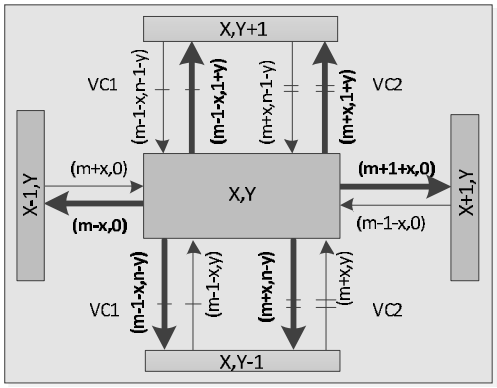

Figure 3.27: Channel numbering in the Mad-y method

As Mad-y is a minimal and adaptive routing method, it cannot fully utilize the eligible turns to route packets through less congested regions. The aim of HARA is to enhance the capability of the existing virtual channels in Mad-y to reroute packets around congested regions and hotspots. Since the Mad-y and HARA methods combine two virtual channels with different prohibited turns, they diminish the drawbacks of turn models prohibiting certain turns at all locations. In minimal routings, (e.g. Mad-y), 180-degree turns are prohibited but they can be incorporated in non-minimal routings (e.g. HARA). One way to incorporate 180-degree turns is to examine the turns one by one to see whether the turn causes any cycle. After determining all allowable turns, in order to prove deadlock freeness, the numbering mechanism is utilized.

In HARA, however, we use the numbering mechanism of the Mad-y method to learn all 180-degree turns that can be taken in the ascending order, and then we modify the numbering mechanism to meet our requirements. According to the numbering mechanism shown in Figure 3.27, among 180-degree-vc turns, those from vc1 to vc2 are taken in the ascending order (Figure 3.28(a), Figure 3.28(b)), so that it is safe to employ them in the network. As all 180-degree-vc turns from vc2 to vc1 take place in the descending order, thereby they cannot be used in the network (Figure 3.28(c), Figure 3.28(d)). Now, let us examine the 180-degree-ch turn connecting the first virtual channel of the north output port to the same virtual channel of the north input port (Figure 3.28(e)). As shown in Figure 3.27, the label on the output channel of the north direction along vc1 is *(m-1-x,1+y)* and the label on the input channel of the north direction along the same virtual channel is *(m-1-x,n-1-y)*. The turn takes place in ascending order if and only if *n-1-y* is greater than *1+y*. Therefore, this turn can be safely added to a set of allowable turns if the Y coordinate of a switch is less than *(n-2)/2*. Similarly, in Figure 3.28(f), the 180-degree-ch turn on the vc2 of the north direction is permitted if the Y value of a switch is less than *(n-2)/2*. 180-degree-ch turns on the vc1 and vc2 of the south direction are permitted if and only if the Y coordinate of a switch is greater than *n/2* (Figure 3.28(g) and Figure 3.28(h)). Finally, the 180-degree-ch turn on the west direction is always permitted (Figure 3.28(i)) while the 180-degree-ch turn on the east direction is prohibited in the network (Figure 3.28(j)).

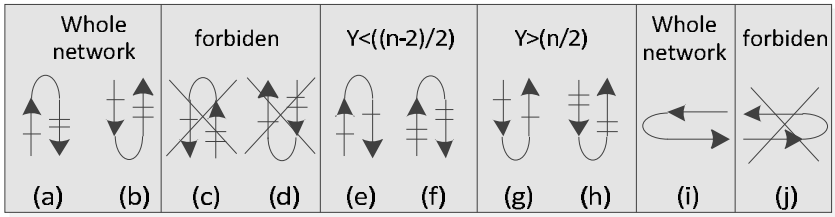

Figure 3.28: Allowable 180-degree turns in the HARA method

As shown in Figure 3.28, there are four conditional 180-degree turns. Two of those are allowable only in the northern part of the network and two others in the southern part of the network. This not only increases the complexity of the routing function but also imposes heterogeneous routing function for switches. To overcome this issue, we modify the numbering mechanism such that two turns are permitted in the whole network (Figure 3.28(g) and Figure 3.28(h)) and two other are prohibited in the whole network (Figure 3.28(e) and Figure 3.28(f)). The numbering mechanism of HARA and all permitted turns in the network are shown in Figure 3.29 and Figure 3.30. As can be observed from this figure, all allowable turns are taken in the ascending order.

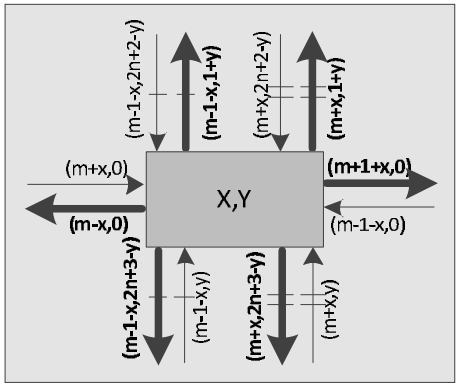

Figure 3.29: The numbering mechanism of HARA

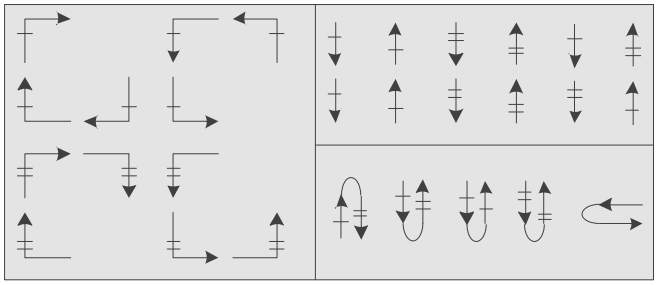

Figure 3.30: All eligible turns in HARA

In the following we prove that HARA is deadlock-free and livelock-free.

Theorem 1: *HARA is deadlock-free*

Proof: If the numbering mechanism ensures that all eligible turns are ordered in the ascending order (descending order), no cyclic dependency can occur between channels. As can be observed from Figure 3.30, all connections between input channels and output channels to form eligible turns in HARA take place in the ascending order and thus HARA is deadlock-free.

Theorem 2: *HARA is livelock-free*

Proof: According to the turn model used in HARA, whenever a packet transmits in the east direction, it can never be routed back to the west direction. Therefore, in the worst case, the packet may reach to the leftmost column and then starts moving in the east direction toward the destination column. Therefore, after a limited number of hops, the packet reaches the destination, and Theorem 2 is proved.

In non-minimal routing, only eligible turns can be employed at each switch but it is not sufficient to avoid blocking in the network. In fact, there is no possibility of creating cycles but packets might be blocked forever. The reason for this blocking is that by using the allowable turns a packet may not be able to find a path to the destination from the next hop and is blocked. On the other hand, one of the aims of HARA is to fully utilize all eligible turns to present a low-restrictive adaptive method in the double-Y network. To achieve the maximal adaptiveness without the blocking issue, for each combination of the input channel and the destination switch position, we examined all eligible 0-degree, 90-degree, and 180-degree turns, separately. The output channels are selected in a way that not only the turn is allowable but also it is guaranteed that there is a path from the next switch to the destination switch. When a packet arrives through one of the input channels, the routing unit determines one or several potential output channels to deliver the packet. The routing decision is based on the relative position of the current and the destination switch which is within one of the following eight cases: north (N), south (S), east (E), west (W), northeast (NE), northwest (NW), southeast (SE), and southwest (SW).

All permissible output channels of HARA, for each pair of the input channels (inCh) and destination positions (pos) are shown in Table 3.3. The adaptivity provided by Mad-y is illustrated in Table 3.4. As can be obtained from both tables, HARA offers a large degree of adaptiveness to route packets. For an instance when the packet arrives from the local input channel (L) and the destination is to the east of the current switch (E), according to the HARA algorithm, the packet can be delivered through all possible output channels (N1, N2, S1, S2, E, and W). However, in a similar condition MAR offers only the east output channel (E).

One of the drawbacks of non-minimal methods is in their complexity due to considering different conditions in the routing decisions. However, as shown in Figure 3.31 (i.e. that is extracted from Table 3.3), the implementation of HARA is very simple.

Table 3.3: Potential output channels offered by HARA

pos / inCh	N	S	E	W	NE	NW	SE	SW
L	N1,N2,S1, W	N1,S1,S2, W	N1,N2,S1,S2, E,W	N1,S1, W	N1,N2,S1, S2,E,W	N1,S1, W	N1,N2,S1,S2, E,W	N1,S1, W
N1	N2,S1, W	S1,S2, W	N2,S1,S2, E,W	S1, W	N2,S1,S2, E,W	S1, W	N2,S1,S2, E,W	S1, W
N2	-	S2	S2, E	-	S2, E	-	S2, E	-
S1	N1,N2,S1, W	N1,S1,S2, W	N1,N2,S1,S2, E,W	N1,S1, W	N1,N2,S1,S2, E,W	N1,S1, W	N1,N2,S1,S2, E,W	N1,S1, W
S2	N2	-	N2, E	-	N2, E	-	N2, E	-
E	N1,N2,S1, W	N1,S1,S2, W	N1,N2,S1,S2, E,W	N1,S1, W	N1,N2,S1,S2, E,W	N1,S1, W	N1,N2, S1,S2, E,W	N1,S1, W
W	N2	S2	N2,S2, E	-	N2,S2, E	-	N2,S2, E	-

Table 3.4: Potential output channels offered by Mad-y

Pos / inCh	N	S	E	W	NE	NW	SE	SW
L	N1,N2	S2	E	W	N1,N2,E	N1,W	S1,S2,E	S1,W
N1	-	S1,S2	E	W	-	-	S1,S2,E	S1,W
N2	-	S2	E	-	-	-	S2,E	-
S1	N1,N2	-	E	W	N1,N2,E	N1,W	-	-
S2	N2	-	E	-	N2,E	-	-	-
E	N1,N2	S2	-	W	N1,N2	N1,W	S1,S2	S1,W
W	N2	S2	E	-	N2,E	-	S2,E	-

```
ALGORITHM: Determining all eligible output channels in HARA
                **----------------------------------------------**
Definitions:    inCh: input Channel
                outCh: output Channel
                pos: destination Position
                **----------------------------------------------**
if pos={L} then outCh(L) <= '1';
if pos={E or NE or SE} then outCh(E) <=' 1';
if inCh={L or N1 or S1 or E} then outCh(W) <= '1';
if inCh={L or S1 or E} then outCh(N1) <= '1';
if inCh={L or N1 or E or S1} then outCh(S1) <= '1';
if (inCh/={N2}) and (pos={N or E or NE or SE}) then outCh(N2) <= '1';
if (inCh/={S2}) and (pos={S or E or NE or SE}) then outCh(S2) <= '1';
end if;
```

Figure 3.31: Determining all eligible output channels by HARA

Figure 3.32 shows an example of the HARA method in a 5×5 mesh network in which the source switch 7 sends a packet to the destination switch 14. According to Table 3.3, the packet arriving from the local channel and delivering toward the destination in the northeast position has six alternative choices (i.e. N1, N2, S1, S2, E, and W); among them, the output channels N1, N2, and E introduce the minimal paths and S1, S2, and W indicate the non-minimal paths. Since the neighboring switches in the shortest paths are in the congested region, the packet is sent to a non-minimal direction that is not congested. Again, at the switch 2, all the minimal paths are congested, so the packet is sent to the switch 1 which is not congested. The same strategy is used until the packet reaches the destination switch. This example shows the ability of the HARA method to reroute packets around the congested regions.

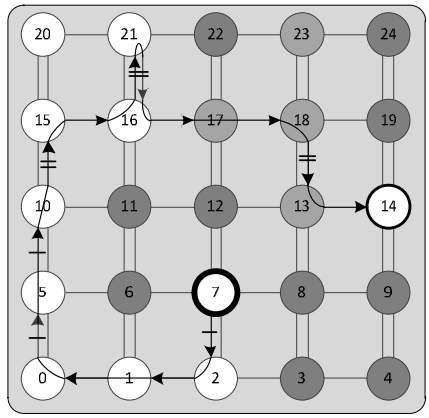

Figure 3.32: An example of HARA

3.3.2 Q-Learning-based Approach using HARA (HARAQ)

Reinforcement learning provides an effective model for problems where optimal solutions are analytically unavailable or difficult to obtain. The learning methodology is based on the common sense that if an action is followed by a satisfactory state or by an improvement, then the tendency to produce that action is strengthened, i.e., reinforced. On the other hand, if the state becomes unsatisfactory, then that particular action should be suitably punished [61]. Q-Learning [51] is one of the algorithms in the reinforcement learning family of machine learning. In the Q-Learning approach, the learning agent first learns a model of the environment on-line and then utilizes this knowledge to find an effective control policy for the given task. Q-Routing [50] is a network routing method based on Q-Learning models which learn a routing policy to minimize the delivery time of packets to reach their destinations. Q-Routing methods are implemented by allowing each switch to maintain a table of Q-values (called Q-Table), where each value is an estimate of how long it takes for a packet to be delivered to a destination from the source switch, if it sends through a neighboring switch [51]. In this method, a routing table is updated whenever a packet is

delivered to the next switch. Q-Routing methods allow a network to be continuously adopted to load changes. Although these schemes make the most optimal routing decisions, due to employing large tables, they are not cost-efficient approaches for NoCs.

We utilize an optimized Q-Routing model for the selection function of HARA to estimate the latency of sending a packet from each output channel to the destination switch. As the output selection function of HARA is inspired by the Q-Routing model, the proposed routing method is called HARAQ (HARA using Q-Routing) [62]. Let us explain the idea of HARAQ using the example of Figure 3.33 where a packet is generated at the source switch S for the destination D. According to HARA, when a packet arrives from the local input channel and destined for a destination switch in the northeast position, six output channels can be selected to forward the packet (i.e. N1, N2, S1, S2, E, and W). At Figure 3.33(a), suppose that the colored entry of the Q-Table indicates the estimated latencies of a packet from each possible output channel to the northeast region. Since the output channel N1 has the lowest estimated latency, the packet is delivered through this channel.

At the switch X, the packet is received by the input channel S1 (Figure 3.33(b)). Using the information in Table 3.3, multiple output channels can be used to forward the packet (i.e. N1, N2, S1, S2, E, and W). Among eligible output channels, the output channel E has the lowest latency, and thus it is selected for sending the packet to the switch Y. At this time, the local and global congestion values should be returned to the switch S. The time the packet waited in the input buffer of the switch X before transmission to the switch Y is counted as the local information (i.e. B_X=1). The minimum estimated latency of routing packets from the switch X to the destination region through the neighboring switch Y is considered as the global latency which is extracted from the Q-Table of the switch X (i.e. $\min Q_X(D,Y)$=4). The summed value of the local and global information provides a new latency estimation of the path from the switch S to the destination D. Finally, the corresponding entry of the Q-Table at the switch S (i.e. row: NE; column: N1) should be updated with the new value. This is done by taking the average of the old and new latency estimations (Figure 3.33(a)).

At the switch Y, the packet is received through the west input channel (Figure 3.33(c)). The output channel with the lowest latency is selected among the three possible output channels (i.e. N2, S2, and E). Upon connecting the input channel to the output channel of the switch Y, local and global information are returned to the switch X. The local congestion shows the waiting time of the packet at the input buffer of the switch Y (i.e. B_Y=3) while the global congestion indicates the estimated latency from the switch Y to the destination switch D through the neighboring switch Z (i.e. $\min Q_Y(D,Z)$=5). The sum of the local and global values is a new latency estimation from the switch X to the destination switch. As shown in Figure 3.33(b), the corresponding entry of the Q-Table at the switch X is updated taking an average of the new estimated value (i.e. $B_Y+\min Q_Y(D,Z)$) and an existing estimation ($Q_X(D,Y)$).

Finally, the packet arrives at the switch Z from the input channel S2 (Figure 3.33(d)). This packet can reach the destination by delivering it through the N2 or E output channel.

The output channel E has the lowest value and selected for delivering the packet. The local latency (i.e. the waiting period at the input buffer of the switch Z) is 3 while the global latency to the destination is equal to 0 as the packet reaches the destination in the next hop. Similarly, the latency values are returned to the switch Y and consequently the corresponding entry of the Q-Table is updated (Figure 3.33(c)). Hence, as packets are propagated inside the network, Q-Tables gradually incorporate more global information [63].

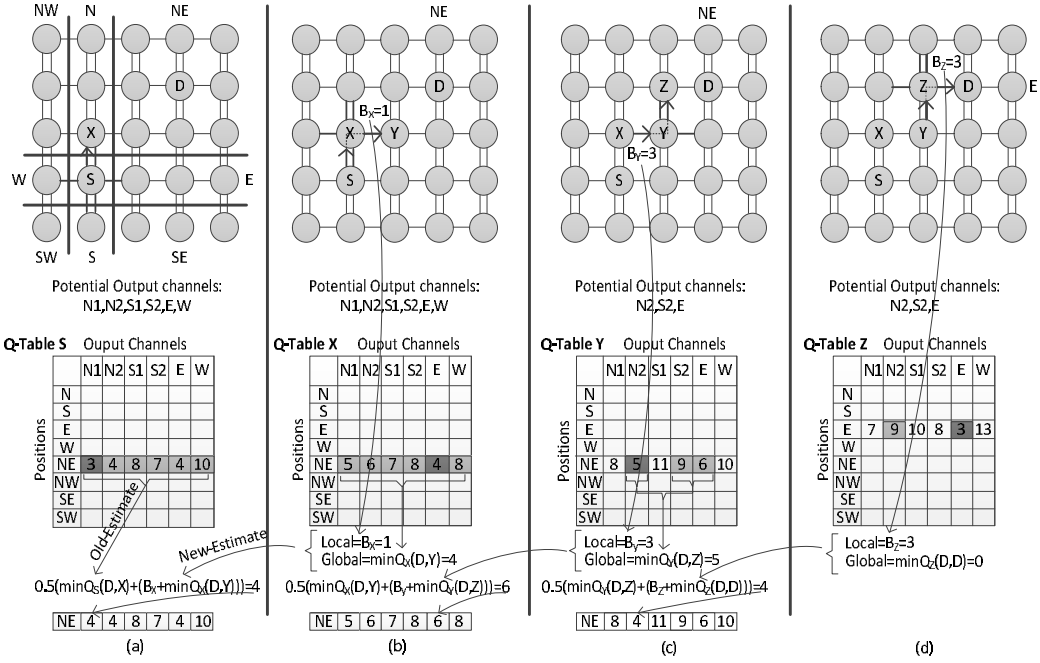

Figure 3.33: The process of updating the Q-Tables

3.3.2.1 Q-Table Format

Q-Routing models learn the network condition at run time and based on the obtained information, a packet is sent through the path that has the lowest estimated latency to the destination switch [50]. Generally, each switch maintains a Q-Table to store the estimated latencies of routing packets to a destination switch through each output channel. Two typical types of Q-Table, called Q-Routing and C-Routing tables, are investigated in [53]. The size of a Q-Routing table is $n \times m \times k$ where n is the number of switches in the network, m is the number of output channels per switch, and k is the size of each entry in the Q-Table. The required area of Q-Routing tables not only is very large but also increases as the network size enlarges. C-Routing tables can decrease the size of tables by taking advantages of the clustering approach. The size of C-Routing tables is $(l+c) \times m \times k$ consisting of two parts: the

cluster part having a size of $c \times m \times k$ where c is the number of clusters, and the local part having a size of $l \times m \times k$ where l is the number of switches within each cluster. The size of Q-Tables can be reduced by using C-Routing tables, but this model still suffers from the scalability issue since the size of C-Routing tables can become rather large as the network scales up. There are some other issues regarding the clustering model such as determining the size of each cluster for different network sizes or partitioning the network when the network size is not a multiple of the cluster size.

The Q-Table in our model is called Region-based Routing (R-Routing); each row of this table corresponds to one of the eight different positions of the destination switch (i.e. N, S, E, W, NE, NW, SE, and SW) and each column indicates output channels (i.e. N1, N2, S1, S2, E, and W). Regardless of the network size, the size of R-Routing tables is $8 \times 6 \times k$ that is considerably smaller than Q-Routing and C-Routing tables. The required size for each approach is given in Table 3.5 where $l=4$ and $k=4$. Note that the reported areas for Q-Routing and C-Routing tables are based on using no virtual channel in the network while the R-Routing table is based on utilizing an extra virtual channel.

While our approach reduces the size of Q-Tables such that they can be applicable in NoCs, someone might think that the accuracy of the estimated latency toward each destination diminishes by our model. In fact, under real traffic conditions, each entry of the R-Routing table is inherently influenced by the switches which are in more communication with them at that period. Therefore, it is not necessary to allocate a row for each specific switch in the network. Moreover, R-Routing tables are updated more occasionally than Q-Routing and C-Routing tables since packets designated for the same regions can be used to update R-Routing tables while in two other models, each entry is updated only by the packets for the same destination.

Table 3.5: The area overhead

Size\method	Q-Routing	C-Routing	R-Routing
8×8	128 bytes	40 bytes	24 bytes
16×16	512 bytes	64 bytes	24 bytes
32×32	2048 bytes	160 bytes	24 bytes

3.3.2.2 Transferring Local and Global Information

In HARAQ, a 4-bit congestion wire is used between each two neighboring switches to propagate local and global congestion information. The local congestion information is a 2-bit value indicating the congestion level of an input buffer. The global congestion information is a 4-bit value which provides a global view of the latency from the output channel of the current switch to the region of the destination switch. This global information is extracted from the corresponding entry of the R-Routing table.

The Q-Values are updated whenever a packet is propagated between two neighboring switches. Suppose that a packet is sent from the switch X toward the destination switch D by

passing through the neighboring switch Y and then the switch Z with the lowest estimated latencies. At the switch Y, upon connecting the input channel to the output channel, 2-bit local and 4-bit global values are aggregated into a 4-bit value (with the maximum value of "1111") and then it is transferred to the switch X. This value is a new estimation of the latency from the selected output channel of the switch X to the destination D. The corresponding entry of the Q-Table at the switch X is updated taking an average of the new estimated value (i.e. B_Y+minQ_Y(D,Z)) and an existing estimation (Q_X(D,Y)). Commonly, in Q-Routing models, the following formula is used:

$$Q_X(D,Y) = (1-\propto)Q_X(D,Y)+\propto (B_Y + minQ_Y(D,Z))$$

In this formula, α represents the learning rate at which newer information overwrites the older one. With the factor of 0, no learning is performed while a factor of 1 considers only the most recent information [52]. In our simulation, a 50-50 weight is assigned to the old and new information so that $\alpha=0.5$.

3.3.2.3 Table Initialization

Q-Routing models have an initial learning period during which it performs worse than minimal schemes. The reason for this inefficiency is that there is a possibility of choosing non-minimal paths even if the network is not congested. To cope with this problem, in the initialization phase, all entries of Q-Tables are initialized such that minimal output channels are set to "0000" and non-minimal output channels are set to "1000" and never can be less than it. Accordingly, in a low traffic condition, only minimal paths are selected while non-minimal paths are used to distribute traffic when the network gets congested.

3.3.3 Results and Discussion

To evaluate the efficiency of HARAQ, two other schemes are also implemented, DBAR [43] and C-Routing [53]. The former is an adaptive routing algorithm using local and non-local congestion information; while the latter is an adaptive and cluster-based routing using the Q-Learning technique (DBAR and C-Routing methods are reviewed in Section 3.1.5 and Section 3.1.6). For fairness, DBAR and C-Routing utilize a fully adaptive routing function based on Mad-y [57]. A wormhole-based NoC simulator is developed with VHDL to model all major components of the on-chip network and simulations are carried out to determine the latency characteristic of each network. The packet length is uniformly distributed between 1 and 10 flits. For all switches, the data width is set to 32 bits and each input channel has the buffer (FIFO) size of 8 flits. The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles. Two synthetic traffic profiles including uniform random and hotspot, and five application benchmarks from SPLASH-2 [64] are selected.

3.3.3.1 Performance Evaluation under Uniform Traffic Profile

In Figure 3.34, the average communication delay as a function of the average packet injection rate is plotted for an 8×8 mesh network. As observed from the results, in low traffic loads, the Q-Routing schemes (HARAQ and C-Routing) behave as efficiently as DBAR. As load increases, DBAR is unable to tolerate the high load condition, while the Q-Routing schemes learn an efficient routing policy. HARAQ leads to the lowest latency due to the fact that it can distribute traffic more efficiently than the other two schemes. In fact, in DBAR and C-Routing, packets use minimal paths so that under this traffic they are routed through the very center of the network which creates large permanent hotspots in the network. Correspondingly, packets traversing through the center of the network will be delayed much more than they would use any non-minimal paths.

Due to the fact that the HARAQ method can reroute packets through non-minimal paths, it alleviates the congestion in the network and performs considerably better than other schemes. Using minimal and non-minimal routes along with the intelligent selection policy reduces the average network latency of HARAQ in an 8×8 network (near the saturation point) about 18% and 37%, compared with C-Routing and DBAR, respectively.

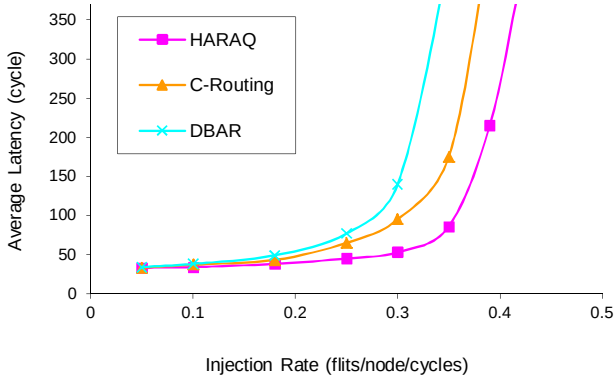

Figure 3.34: Performance analysis in an 8×8 mesh network under the uniform traffic profile

3.3.3.2 Performance Evaluation under Hotspot Traffic Profile

In simulations, given a hotspot percentage of *H*, a newly generated packet is directed to each hotspot switch with an additional *H* percent probability. We simulate the hotspot traffic with a single hotspot switch at (4,4) in an 8×8 mesh network. The performance of each network with *H=10%* is illustrated in Figure 3.35.

As can be observed from the figure, the proposed routing scheme achieves better performance compared with the other schemes. In an 8×8 mesh network, the performance gain near the saturation point is about 31% and 42%, compared with C-Routing and DBAR, respectively. The results reveal that using the non-minimal scheme along with the Q-Learning policy can distribute the traffic efficiently.

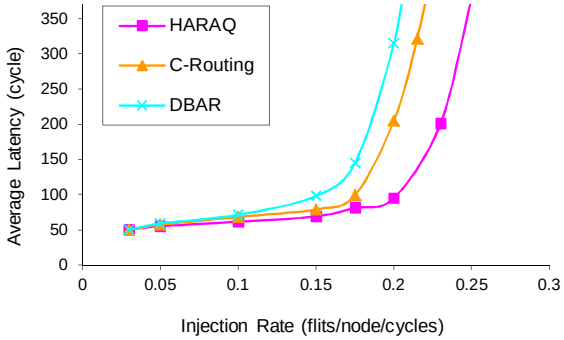

Figure 3.35: Performance analysis in an 8×8 mesh network under hotspot traffic profile with *H=10%*

3.3.3.3 Performance Analysis under Application Traffic Profile

Application traces are obtained from the GEMS simulator using some application benchmark suites selected from SPLASH-2. The configuration is the same as in Section 3.2.3 which has been listed in Table 3.1. Figure 3.36 shows the average packet latency across four benchmark traces, normalized to DBAR. HARAQ provides lower latency than other schemes and it shows the greatest performance gain in Radix with 27% reduction in latency (vs. C-Routing). The average performance gain of HARAQ across all benchmarks is up to 22% vs. C-Routing and 33% vs. DBAR.

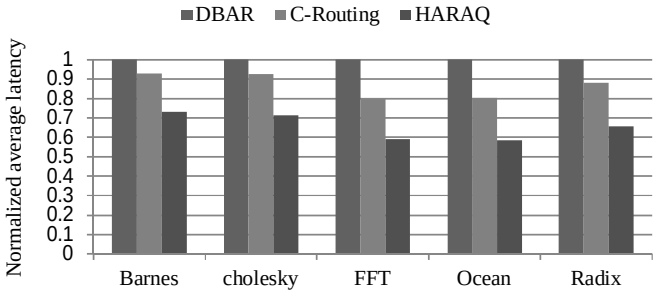

Figure 3.36: Performance analysis under different application benchmarks normalized to DBAR

3.3.3.4 Hardware Analysis

To assess the area overhead and power consumption of HARAQ, the whole platform of each scheme is synthesized by Synopsys Design Compiler. Each scheme includes switches, communication channels, and congestion wires. For synthesis, we use the TSMC 65nm technology at the operating frequency of 500MHz and supply voltage of 1V. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power consumption of each scheme is calculated under the hotspot traffic profile near the saturation point using Synopsys PrimePower in an 8×8 mesh network. The layout area and

power consumption of each platform are shown in Table 3.6. The area overheads of both C-Routing and HARAQ approaches are higher than DBAR while the area overhead of HARAQ is smaller than C-Routing due to using smaller table sizes. Comparing power consumption indicates that the HARAQ platform consumes more power than DBAR but less power than C-Routing.

Table 3.6: Hardware implementation details

Network platforms	Area (mm^2)	Power (mw)
DBAR	2.579	1.316
C-Routing	2.912	1.677
HARAQ	2.765	1.594

3.4 The Proposed Fuzzy-based Approach

Cluster-based and learning-based approaches can greatly improve performance compared with traditional methods. However, they have yet another drawback. The comparison between the congestion values of candidate output ports are very strict, meaning that a single free buffer slot in one direction would change the routing decision toward a more congested region. Fuzzy systems avoid arbitrary rigid boundaries by giving a level of confidence to a data. They are commonly used to improve performance or to resolve ambiguities in complex problems that are difficult to tackle mathematically. Since control problems in communication systems become increasingly complex (due to their characteristics of having multiple performance criteria), the use of fuzzy and adaptive algorithms is indeed well suited to increase performance. Most applications using fuzzy-logic can be regarded as systems with numerical inputs and outputs [65]. The linguistic descriptions are used to define the relationship between input(s) and output(s).

In this section, at first, we investigate the problem of non-optimal routing decisions because of defining rigid boundaries on input parameters. The investigation is done on the DyXY method as a representative of traditional methods. After defining the problem, we try to solve it by modifying the DyXY method. Although, by using the new approach, called Non-Fuzzy Routing Algorithm (NFRA) [66], [67], the routing decision is improved, the problem still persists and needs deeper analysis. To address this problem, we present a Fuzzy-based Routing Algorithm (FRA), utilizing the fuzzy-logic controller [66], [67]. The fuzzy system is employed to estimate the latency of each candidate direction. This cost is determined based on the crisp (non-fuzzy) value of two metrics, the number of occupied buffer slots at the corresponding input buffer of the next switch (*OccupiedSlots_Input*) and the congestion level of that switch (*OccupiedSlots_Switch*). At each switch, the output direction with the lowest cost is chosen as to deliver the packet. The use of fuzzy-logic algorithms in the path decision making leads to a systematic comparison among the candidates of output ports.

3.4.1 Non-Fuzzy Routing Algorithm (NFRA)

In the DyXY method, switches employ a pre-port selection unit. Based on the number of occupied buffer slots in the instant input buffer of the neighboring switches, the pre-port selection unit selects the best candidate between two minimal directions (i.e. North vs. East for northeast packets; North vs. West for northwest packets; South vs. East for southeast packets; and South vs. West for southwest packets) and makes a routing decision based on this information. Although DyXY is simple, in many cases it leads to non-optimal decisions.

Let us consider the example of Figure 3.37 when the switch 5 has to decide whether to send a packet to the switch 6 or 9. Since the number of occupied buffer slots in the south input buffer of the switch 9 (i.e. 5 occupied buffer slots) is more than the west input buffer of the switch 6 (i.e. 4 occupied buffer slots), the packet is sent to the switch 6. The decision is made because of an extra free buffer slot at the switch 6. Now, by looking at the overall congestion level at the switches 6 and 9, we notice that the total number of occupied buffer slots in the switch 6 (i.e. 28 occupied buffer slots) is considerably larger than that of the switch 9 (i.e. 15 occupied buffer slots). In other words, the contention in the switch 6 is high and thus packets entering this switch from the west input port will be in competition with other packets to receive the desired output channel while this contention situation is mild at the switch 9. In this example, obviously it was better to deliver a packet to the switch 9 rather than the switch 6. DyXY selects a direction by random in the case of the same congestion values in both directions.

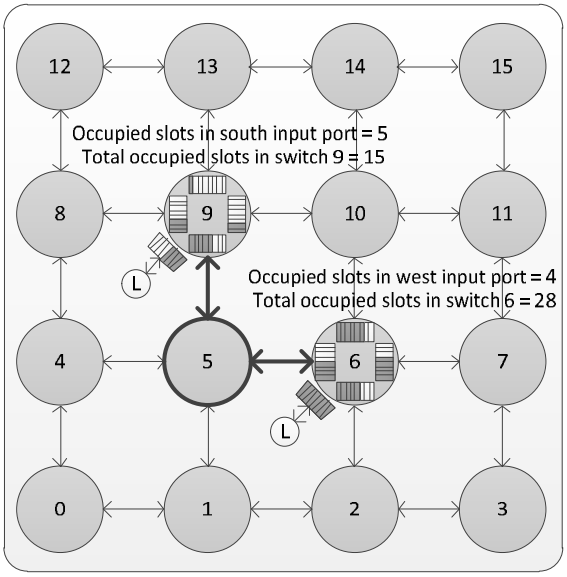

Figure 3.37: Non-optimal decision by the DyXY routing algorithm

This example shows that DyXY may lead to a non-optimal decision when the number of occupied buffer slots is comparable in two directions. In order to improve the performance of DyXY, we proposed a method, called Non-Fuzzy Routing Algorithm (NFRA), which utilizes two parameters for choosing among output directions. These parameters are the number of occupied buffer slots in an instant input buffer of the neighboring switch (*OccupiedSlots_Input*) and the total number of occupied buffer slots in the neighboring switch (*OccupiedSlots_Switch*). To exchange the congestion information of *OccupiedSlots_Input* and *OccupiedSlots_Switch* between adjacent switches, 4-bit and 6-bit wires are required. In NFRA, when the differences between the values of *OccupiedSlots_Input* in two directions are less than or equal to 2, the values of *OccupiedSlots_Switch* are checked, so that the packet is sent to a direction which has the lowest *OccupiedSlots_Switch*.

OccupiedSlots_Input has been considered as the main factor in NFRA since it has the information about the available slots in an instant input buffer to accommodate a packet rather than the overall switch congestion condition (*OccupiedSlots_Switch*). Considering the same example as in Figure 3.37, since the values of *OccupiedSlots_Input* are comparable, the values of *OccupiedSlots_Switch* are compared together and thus the packet is sent to the switch 9, resulting in a better routing decision. The NFRA routing algorithm is shown in Figure 3.38. Although NFRA can alleviate the shortcoming of the DyXY routing algorithm, it is still suffering from non-optimal routing decisions due to using rigid boundaries in input parameters. Regardless of the metrics used, this is a common drawback of traditional methods.

In the following example, we explain this problem in the case of the NFRA routing algorithm. In Figure 3.39(a) suppose that the switch 5 has two options (i.e., the switch 6 or the switch 9) to forward a packet toward the desired destination. At the switch 5, the congestion statuses of the west buffer of the switch 6 is compared with the south buffer of the switch 9. Since the switch 6 has five free slots more than that of the switch 9, the packet is delivered to the switch 6. However, it might not be a good decision as well, as the packet at the switch 9 may get access to its desired output channel earlier than at the switch 6.

As another example in Figure 3.39(b), when the values of *OccupiedSlots_Input* are three and five in two directions, the decision will depend on the values of *OccupiedSlots_Switch*. This may not lead to an optimal decision as the values of *OccupiedSlots_Switch* are nearly similar. In this case it is better to send a packet to the switch 6 which is able to accommodate more flits of the arriving packet. In sum, the inability to find a proper output direction is the weakness of almost all traditional methods. On top of it, solutions vary for different buffer sizes and different metrics. These problems are due to the fact that the decision making is based on rigid boundaries on input variables. A possibility to solve this problem is to equip the routing algorithms by a fuzzy-logic system that allows a flexible and controllable routing process.

```
ALGORITHM: Non-fuzzy routing algorithm
                    **-------------------------------------------------**
Definitions:    Xc,Yc: X and Y coordinates of the current switch
                Xd,Yd: X and Y coordinates of the destination switch
                X_dir: candidate port in the X dimension
                Y_dir: candidate port in the Y dimension
                    **-------------------------------------------------**
if (Xd=Xc) and (Yd=Yc) then select <= local;
elsif (Xd=Xc) then select <= Y_dir;
elsif (Yd=Yc) then select <= X_dir;
else
    if (ABS (OccupiedSlots_Input(X_dir) - OccupiedSlots_Input(Y_dir)) <=2) then
        if OccupiedSlots_Switch(X_dir) >= OccupiedSlots_Switch(Y_dir) then
            select <= Y_dir;
        else
            select <= X_dir;
        end if;
    elsif (OccupiedSlots_Input(X_dir) > OccupiedSlots_Input(Y_dir)) then
        select <= Y_dir;
    else
        select <= X_dir;
    end if;
end if;
```

Figure 3.38: The pseudo code of NFRA

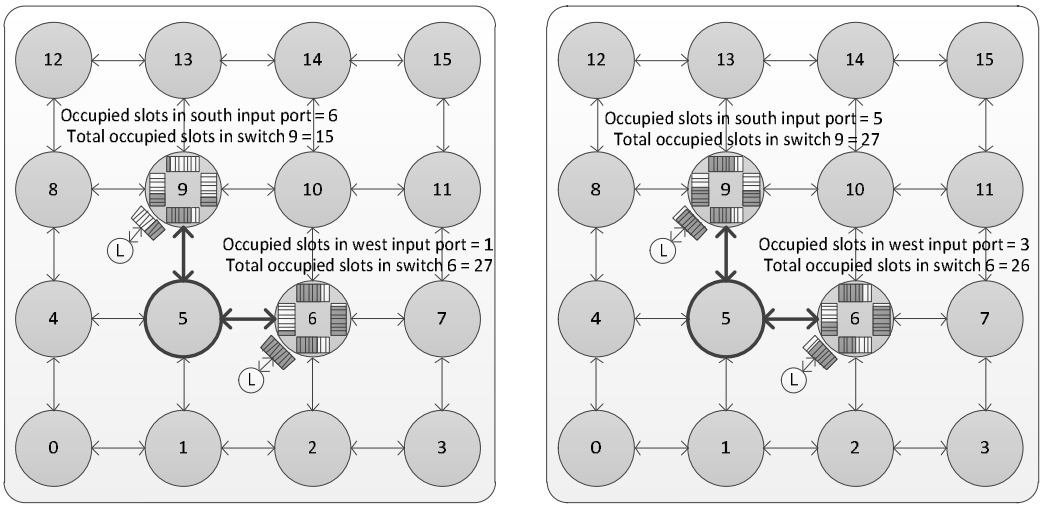

Figure 3.39: Two examples of non-optimal routing decisions in NFRA

3.4.2 Fuzzy-based Routing Algorithm (FRA)

Fuzzy controllers are widely used in many different fields nowadays, ranging from control applications, robotics, image and speech processing to biological and medical systems [65].

Fuzzy controllers have already received much attention in Ad Hoc, wireless and interconnection networks [21], [22]. For example in [68], a fuzzy controller is used to instructing cache decisions and to optimize routing selection, so that only high quality links are used between source and destination switches. In [69], the authors present a fuzzy controller-based QoS routing algorithm in mobile Ad Hoc networks in order to dynamically evaluate the route expiry time. In [70], hop-count, bandwidth, and mobile speed are considered for routing decision based on a fuzzy-logic system to satisfy the required QoS. In [71], using fuzzy rules, the link cost is dynamically determined depending on the link delay and the number of packets waiting in the queue. The analysis of control problems in communication systems can be perfectly done using the fuzzy-logic mechanism, due to their characteristics of having multiple performance criteria [72]. In this section, we propose a fuzzy-based routing algorithm for NoCs, called FRA.

As illustrated in Figure 3.40(a), a Fuzzy Inference System (FIS) consists of an input stage (fuzzification), an inference system, a composition unit, and an output stage (defuzzification). Fuzzification is a process of converting crisp input values to fuzzy values. The fuzzy inference system uses the collection of linguistic rules to convert the fuzzy inputs into fuzzy outputs. In the composition stage, the fuzzy outputs of all rules are combined together to obtain a single fuzzy output. Defuzzification converts the fuzzy output into crisp output value. Figure 3.40(b) demonstrates employing a fuzzy-logic system in a switch. A packet can be sent through at most two directions toward the destination switch. The cost is calculated over two candidate minimal directions using fuzzy-logic. The packet is sent to a direction with the lowest cost.

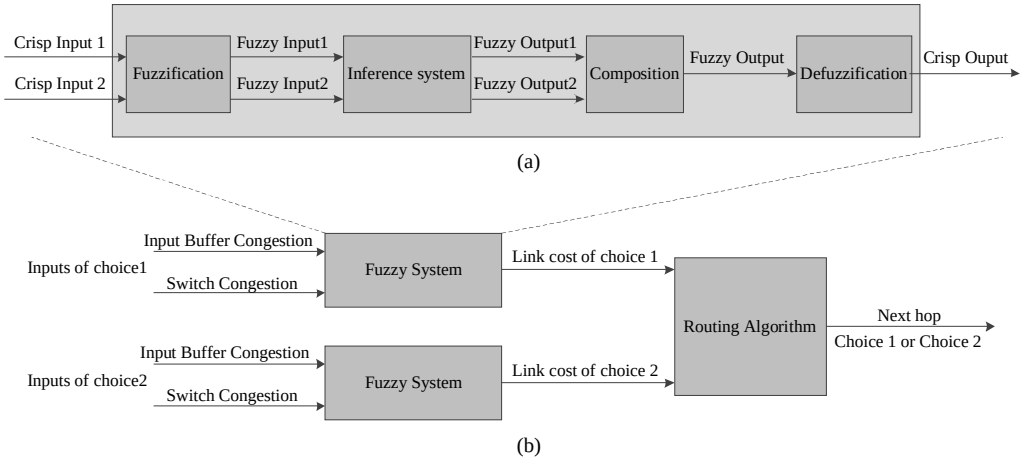

Figure 3.40: (a) General fuzzy system (b) Fuzzy routing algorithm

3.4.2.1 Fuzzification

Like NFRA, FRA has two input variables (i.e. *OccupiedSlots_Input* and *OccupiedSlots_Switch*) and one output (i.e. *Cost*). In the fuzzification stage, the fuzzy

controller accepts the crisp inputs and maps them into their membership functions, known as fuzzy set. Fuzzification determines the degree of membership for a crisp input χ being applied to appropriate fuzzy set μ. The degree of membership is a number between 0 and 1.

$$\mu: \chi \rightarrow [0,1]$$

The value 0 meaning that χ is not a member of the fuzzy set; the value 1 means that χ is fully a member of the fuzzy set. The values between 0 and 1 characterize fuzzy members, which partially belong to the fuzzy set.

A *membership function (MF)* is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the *universe of discourse* [73]. The most commonly used shapes for membership functions are triangular, trapezoidal, and Gaussian. Among them, the *triangular* membership function is the simplest and the most frequently used [73], [74]. In the proposed FRA, the assigned membership functions to input and output variables are chosen as triangular. The triangular edges can be identified by a triple (a, b, c) (with a < b < c). The parameters {a,b,c} determine the X coordinates of the three corners of the underlying triangular function. Figure 3.41 illustrates a triangular membership function defined by the triangle (2,4,6). The point 4 has the largest value in the membership function while points 2 and 6 have the lowest values.

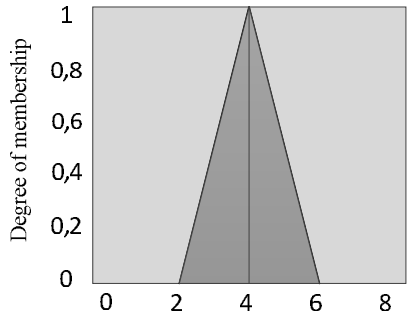

Figure 3.41: An example of triangular membership function

In the following, we have defined a fuzzy membership function for *OccupiedSlots_Input* and *OccupiedSlots_Switch* as the input metrics and *Cost* as the output parameter. From three to seven curves are generally appropriate to cover the required range of an input value, or the universe of discourse in a fuzzy region. We have examined the fuzzy system using three, five, and seven curves. The performance gain of considering five and seven curves was considerably larger in comparison with three curves while the difference between five and seven curves was negligible. Thereby, we select the fuzzy set with five states as “zero (Z)”, “very small (VS)”, “small (S)”, “medium (M)”, and “large (L)”.

- **Membership Function of *OccupiedSlots_Input***

Considering the number of occupied buffer slots in the input buffer (*OccupiedSlots_Input*), the universe of discourse includes the numbers between 0 and 8. The value of 0 indicates that the input buffer is empty while the value of 8 indicates that the buffer is full. The triangular membership function maps the number of occupied buffer slots in the input buffer ranged from 0 to 8 (*OccupiedSlots_Input*) to five fuzzy sets (Z, VS, S, M, and L) by a degree of membership. The assignment is illustrated in Figure 3.42(a). According to this figure, the fuzzy sets are {Z: triangle (0,0,2)}, {VS: triangle (0,2,4)}, {S: triangle (2,4,6)}, {M: triangle (4,6,8)}, and {L: triangle (6,8,8)}.

- **Membership Function of *OccupiedSlots_Switch***

OccupiedSlots_Switch can be a number between 0 and 40, where the value of 0 means that all the input buffers of the switch are empty while the value of 40 indicates that all input buffers are full. This variable can be divided into five fuzzy sets (Z, VS, S, M, and L). As shown in Figure 3.42(b), the crisp values are mapped into the sets associated with the degree of membership by defining fuzzy sets as {Z: triangle (0,0,10)}, {VS: triangle (0,10,20)}, {S: triangle (10,20,30)}, {M: triangle (20,30,40)}, and {L: triangle (30,40,40)}.

- **Membership Function of *Cost***

We have defined *Cost* as a value between 0 and 40. The fuzzy set includes the states as Z, VS, S, M, and L. The triangular membership function maps the input element to a certain fuzzy set by a degree of membership. As illustrated in Figure 3.42(c), the fuzzy sets are {Z: triangle (0,0,10)}, {VS: triangle (0,10,20)}, {S: triangle (10,20,30)}, {M: triangle (20,30,40)}, and {L: triangle (30,40,40)}.

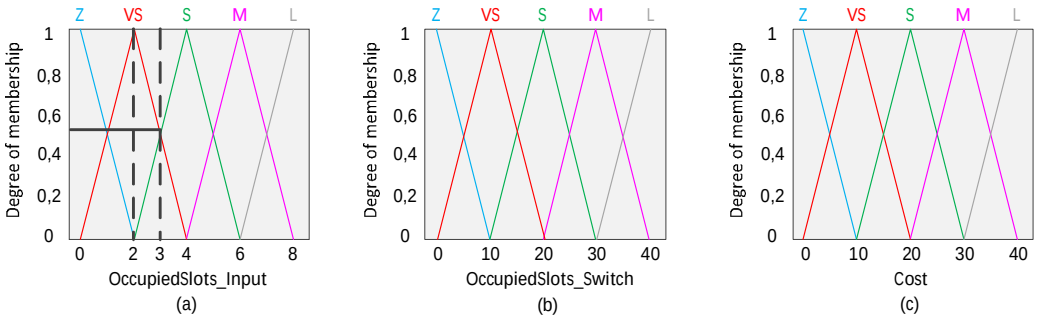

Figure 3.42: (a) *OccupiedSlots_Input* (b) *OccupiedSlots_Switch* (c) *Cost* membership functions

With these schemes, the states of input variable are no longer changed abruptly from one state to the next. Instead, as the input changes, it loses a value in one membership function

while gaining a value in the next. In other words, an input variable with some degree is part of two membership functions. In Figure 3.42(a), for example, when *OccupiedSlots_Input* is 2, the input fully belongs to the membership function VS. However, when *OccupiedSlots_Input* is 3, the input is partially (0.5 each) part of two membership functions VS and S. In general, a fuzzy system is constructed based on human expertise and expert knowledge. The boundaries of the states are also defined in the same manner. The knowledge can be obtained by experiments.

3.4.2.2 Fuzzy Inference System

An inference engine is equipped with fuzzy rules to make a decision for an output channel based on the current condition of the network. The inference engine is characterized by a set of linguistic statements to describe the system by using a number of conditional "IF-THEN" rules where the IF part is called the "antecedent" and the THEN part is called the "consequent". Expert knowledge is usually used to form the rules of a fuzzy inference system. Table 3.7 contains the rules used in FRA with two fuzzy inputs and one fuzzy output. The table provides various ranges of the output for different ranges of inputs. Filling a data table with fuzzy attributes (scaling) is subjective. The table is filled based on the basic knowledge on the impact of each metric in the overall performance of the network. Based on our experiments, small changes in the table have negligible impact on performance.

Table 3.7: FRA inference rules

Rules		OccupiedSlots_Switch				
		Z	VS	S	M	L
Occupied Slots-Input	Z	Z	Z	VS	S	M
	VS	Z	VS	VS	S	M
	S	VS	VS	S	M	M
	M	S	S	M	L	L
	L	M	M	L	L	L

Fuzzy rule sets usually have several antecedents that are combined using fuzzy operators, such as fuzzy intersection (AND) and fuzzy union (OR). If the rule uses an AND relationship for mapping of two input variables, the minimum of those values is used as the output while for the OR relationship, the maximum is used. In FRA, the AND operator is utilized to combine the fuzzy inputs. Let us consider an example in Figure 3.43 where the *OccupiedSlots_Input* and *OccupiedSlots_Switch* have the values of 5 and 18, respectively. As shown in Figure 3.43(a), *OccupiedSlots_Input* is a part of membership functions S and M while the portion of each membership function is 0,5. The input *OccupiedSlots_Switch* is a part of membership functions VS and S as illustrated in Figure 3.43(b). In this case, the degree of membership for membership functions VS and S is 0,2 and 0,8, respectively.

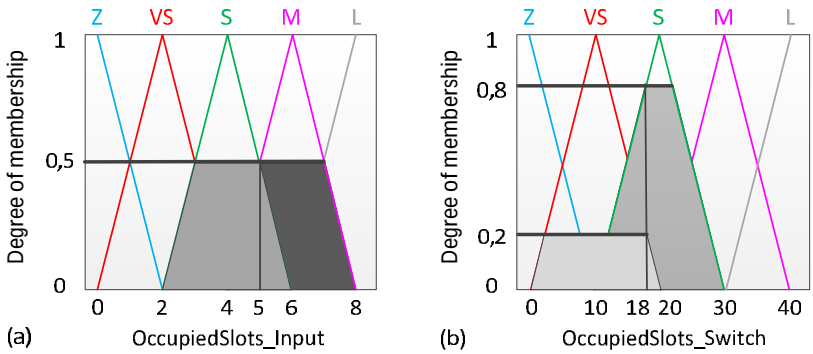

Figure 3.43: (a) *OccupiedSlots_Input* (b) *OccupiedSlots_Switch* as a part of two membership functions

As shown in Figure 3.44, there are four combinations between *OccupiedSlots_Input* and *OccupiedSlots_Switch* as:

- Figure 3.44(a): *OccupiedSlots_Input: S and OccupiedSlots_Switch: VS*
- Figure 3.44(b): *OccupiedSlots_Input: S and OccupiedSlots_Switch: S*
- Figure 3.44(c): *OccupiedSlots_Input: M and OccupiedSlots_Switch: VS*
- Figure 3.44(d): *OccupiedSlots_Input: M and OccupiedSlots_Switch: S*

Based on these combinations, four following rules are fired according to Table 3.7:

Rule1-if *(OccupiedSlots_Input is S)* and *(OccupiedSlots_Switch is VS)* then *(Cost is VS)*
Rule2-if *(OccupiedSlots_Input is S)* and *(OccupiedSlots_Switch is S)* then *(Cost is S)*
Rule3-if *(OccupiedSlots_Input is M)* and *(OccupiedSlots_Switch is VS)* then *(Cost is S)*
Rule4-if *(OccupiedSlots_Input is M)* and *(OccupiedSlots_Switch is S)* then *(Cost is M)*

3.4.2.3 Composition and Defuzzification

Defuzzification is the process of producing a quantifiable result in fuzzy-logic and converts the fuzzy control action into a crisp value. The outputs of all rules should be aggregated and converted into a single output. Two methods for defuzzification are widely used:

1- The Center-of-Gravity method (CoG). This method finds the geometrical center. It favors the rule with the output of the greatest area.
2- The Mean-of-Maxima method (MoM). This method finds the value which has the maximum membership degree according to the fuzzy membership function.

MoM is simpler but it loses useful information while CoG is the commonly used method as it is more efficient. In FRA, the CoG defuzzification method is used to produce a crisp value.

In the defuzzification stage, the four obtained cost values (Figure 3.44) are combined together and by using the Center-of-Gravity method, a single cost value is extracted. As shown in Figure 3.45, the fuzzy outputs of the same cost membership function are summed together while the values in different membership functions are united (i.e. the maximum value is considered).

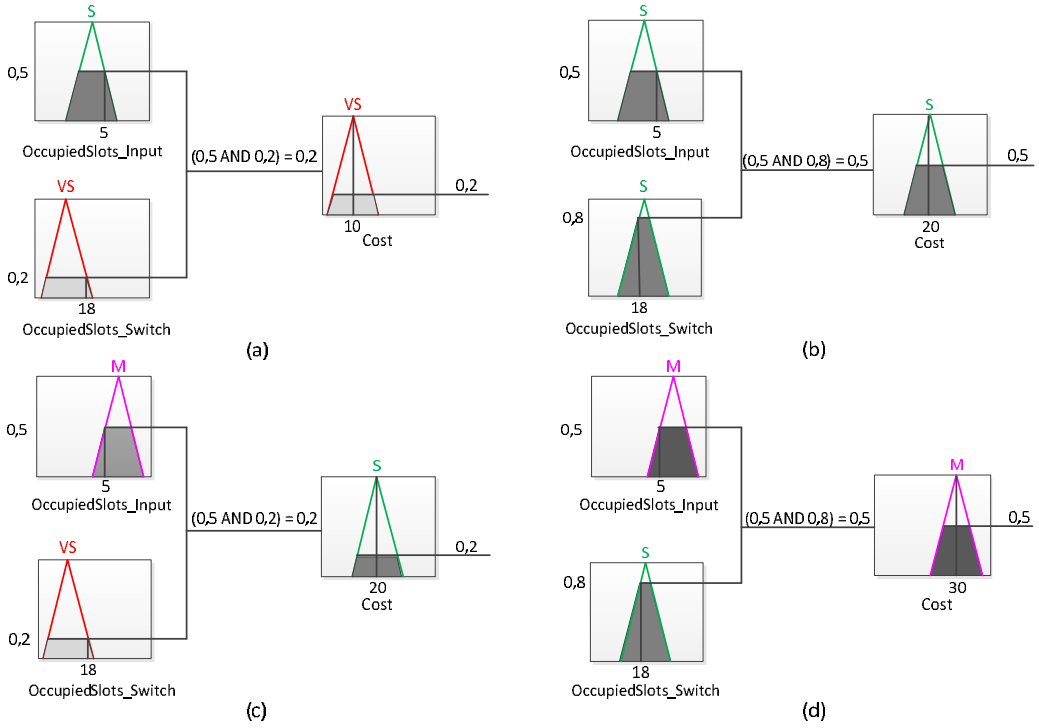

Figure 3.44: *Cost* for (a) rule1 (b) rule2 (c) rule3 (d) rule4

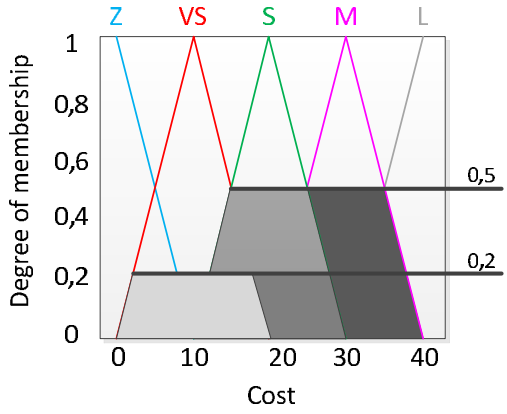

Figure 3.45: Composition of the *Cost* membership function of all rules

In this case, the cost value can be calculated from the following formula:

$$ObtainedCost = \frac{ObtainedCosts}{degree\ of\ membership\ functions}$$

$$= \frac{(10 * 0{,}2) + (20 * 0{,}5) + (20 * 0{,}2) + (30 * 0{,}5)}{0{,}2 + 0{,}5 + 0{,}2 + 0{,}5} \approx 22$$

According to this formula, the degree of membership function of each rule multiplies to the cost value associated with the maximum value in the membership function. This procedure is performed for both candidate directions and the packet is delivered to a direction with a smaller *Cost* value. Although in FRA, the congestion metric parameters of the number of occupied slots (*OccupiedSlots_Input*) and congestion level of a switch (*OccupiedSlots_Switch*) are used, the proposed approach is generic and can be easily extended to different routing metrics. Now, let us employ the proposed fuzzy-logic system in the example of Figure 3.39(b) where a packet should be delivered either through the switch 6 or switch 9. The conditions of these switches are as follows:

$$Switch9 = \begin{Bmatrix} OccupiedSlots\_Input = 5 \\ OccupiedSlots\_Router = 27 \end{Bmatrix}$$

$$Switch6 = \begin{Bmatrix} OccupiedSlots\_Input = 3 \\ OccupiedSlots\_Router = 26 \end{Bmatrix}$$

We first measure the cost of selecting the switch 9 and then the switch 6. Considering the switch 9, the degree of membership function for the input parameter *OccupiedSlots_Input* is shown in Figure 3.46(a) and the input parameter *OccupiedSlots_Switch* is shown in Figure 3.46(b). According to these figures, the degree of membership in four triangles is non-zero: S and M from the input parameter *OccupiedSlots_Input* and S and M from the input parameter *OccupiedSlots_Switch*. The combinations of these membership functions result in a new membership function called *Cost*. The type of the membership function is extracted from Table 3.7 while the degree of membership function is achieved by using an AND operator between the input parameters. The resulted *Cost* and the degree of membership are shown in Table 3.8.

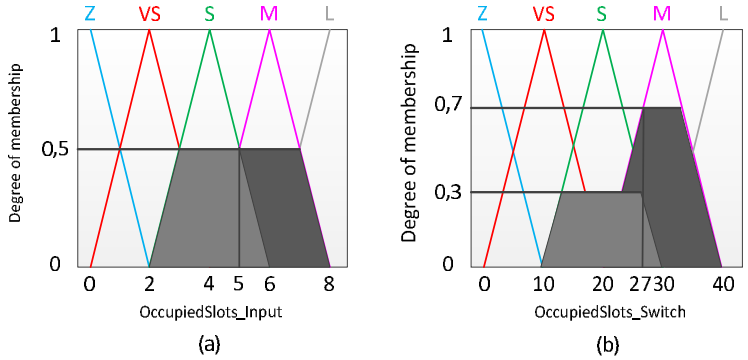

Figure 3.46: The degree of membership function for the input parameters at the switch 9

Table 3.8: The cost membership function of the switch 9; MF stands for the membership function

Occupied Slots_Input	Degree of MF	Occupied Slots_Switch	Degree of MF	Cost	MaxValue of MF	Degree of MF
S	0,5	S	0,3	Rule(S,S)=S	20	(0,5 AND 0,3)=0,3
S	0,5	M	0,7	Rule(S,M)=M	30	(0,5 AND 0,7)=0,5
M	0,5	S	0,3	Rule(M,S)=M	30	(0,5 AND 0,3)=0,3
M	0,5	M	0,7	Rule(M,M)=L	40	(0,5 AND 0,7)=0,5

Finally, the cost of the switch 9 is calculated by:

$$ObtainedCost_{Switch9} = \frac{ObtainedCosts\ \text{Values}}{degree\ of\ membership\ functions}$$

$$= \frac{(20 * 0{,}3) + (30 * 0{,}5) + (30 * 0{,}3) + (40 * 0{,}5)}{0{,}3 + 0{,}5 + 0{,}3 + 0{,}5} \approx 31$$

Similarly, the degree of membership function for each input parameter at the switch 6 is shown in Figure 3.47.

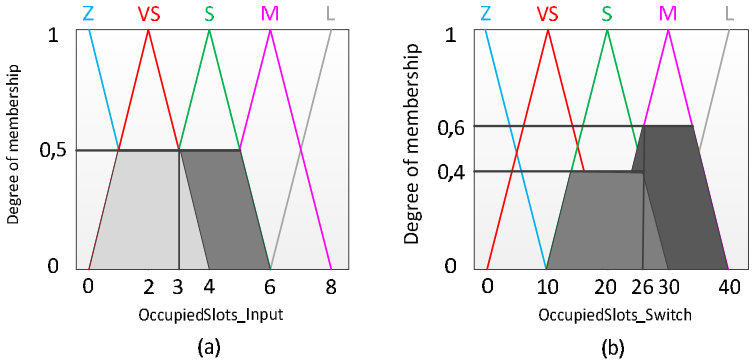

Figure 3.47: The degree of membership function for the input parameters at the switch 6

The information on the *Cost* membership function is listed in Table 3.9.

Table 3.9: The cost membership function of the switch 6

Occupied Slots_Input	Degree of MF	Occupied Slots_Switch	Degree of MF	Cost	MaxValue of MF	Degree of MF
VS	0,5	S	0,4	Rule(VS,S)=VS	10	(0,5 AND 0,4)=0,4
VS	0,5	M	0,6	Rule(VS,M)=S	20	(0,5 AND 0,6)=0,5
S	0,5	S	0,4	Rule(S,S)=S	20	(0,5 AND 0,4)=0,4
S	0,5	M	0,6	Rule(S,M)=M	30	(0,5 AND 0,6)=0,5

Finally, the cost of the switch 6 is measured as follow:

$$ObtainedCost_{Switch6} = \frac{ObtainedCosts}{degree\ of\ membership\ functions}$$

$$= \frac{(10 * 0{,}4) + (20 * 0{,}5) + (20 * 0{,}4) + (30 * 0{,}5)}{0{,}4 + 0{,}5 + 0{,}4 + 0{,}5} \approx 20$$

According to these formulas, the cost of sending a packet from the switch 9 or 6 is 31 and 20, respectively, so the packet is sent toward the destination through the switch 6 which is less congested. This choice is reasonable as the switch 9 has more occupied buffer slots in the input buffer than the switch 6 while the overall congestion conditions of both switches are similar. If the final result is not satisfactory (e.g. the switch 6 has a higher cost value than the switch 9), it means that the fuzzy rules are not well defined and should be modified.

3.4.3 Results and Discussion

To assess the efficiency of the proposed adaptive routing algorithms, NFRA and FRA, we compare them with DyXY under synthetic and multimedia traffic profiles. All of them are based on a fully adaptive routing algorithms using one and two virtual channels along the X and Y dimensions, respectively. We have developed a synthesizable NoC simulator implemented in VHDL to evaluate the efficiency of NFRA and FRA. This simulator is based on wormhole switching in a 2D mesh configuration. The simulator inputs include the array size, the routing algorithm, the link width, the buffer size, and the traffic type. For all switches, the data width was set to 32 bits and each input port has a buffer size of 8 flits. For the performance metric, we use latency defined as the number of cycles between the initiation of the packet and the time when the tail of the packet reaches the destination. The proposed schemes are evaluated for various traffic loads in an 8×8 mesh network. The packet size is uniformly distributed between 1 and 10 flits.

The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles. For evaluating performance, two synthetic traffic profiles including uniform random and hotspot, and multimedia traffic are selected.

3.4.3.1 Performance Evaluation under Uniform Traffic Profile

In the uniform traffic model, each PE core sends a packet to any other core with equal probability. As illustrated in Figure 3.48, NFRA has better performance than DyXY while FRA performs the best under the uniform traffic profile. This performance improvement of NFRA is due to the fact that it makes a better local decision than DyXY. On other hand, FRA performs the best as the routing decision based on it leads to a better distribution of packets over the network.

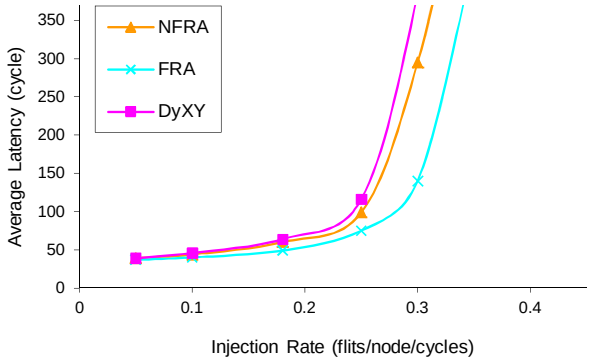

Figure 3.48: Performance analysis in an 8×8 mesh network under the uniform traffic profile

3.4.3.2 Performance Evaluation under Hotspot Traffic Profile

In the hotspot traffic model, a PE receives an extra portion (*H*) of traffic more than the other switches (here we assume the switch (4,4) receives *H=10%* more traffic). As illustrated in Figure 3.49, FRA performs the best and then NFRA and DyXY, respectively. This performance improvement of FRA over NFRA and DyXY confirms the fact that employing fuzzy-logic mechanism results in a better routing decision which in turn reduces average latency of packets.

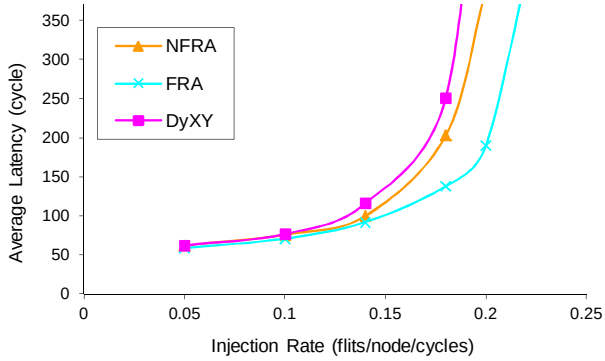

Figure 3.49: Performance analysis in an 8×8 mesh network under hotspot traffic profile with *H=10%*

3.4.3.3 Multimedia Traffic

NFRA and FRA are also evaluated under two realistic case studies mapped onto a 3×4 mesh topology. We selected two different video processing applications: Video Object Plane Decoder (VOPD) and MPEG4 decoder [75]. Figure 3.51 and Figure 3.52 depict the VOPD and MPEG4 Decoder block diagrams mapped onto 3×4 mesh topologies, respectively. Figure 3.50 shows the latency values normalized to DyXY. According to this figure, FRA decreases latency considerably where the performance gain is up to 24% and

25% under the MPEG and VOPD traffic profiles, respectively, compared with the DyXY routing algorithm. The performance gain of NFRA over the DyXY method is around 4% and 6% under the MPEG and VOPD traffic profiles, respectively.

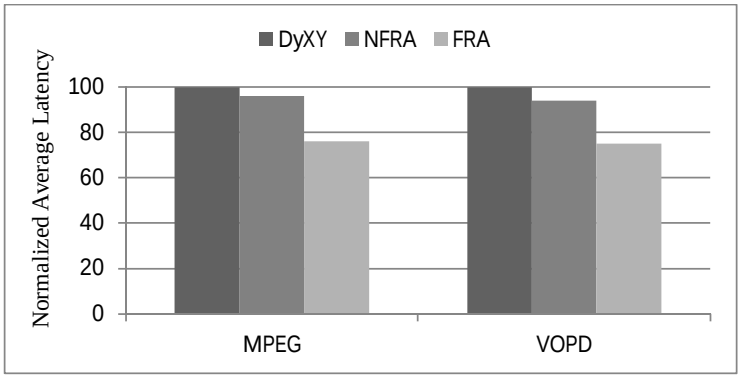

Figure 3.50: Simulation results under two multimedia traffic profiles: MPEG and VOPD

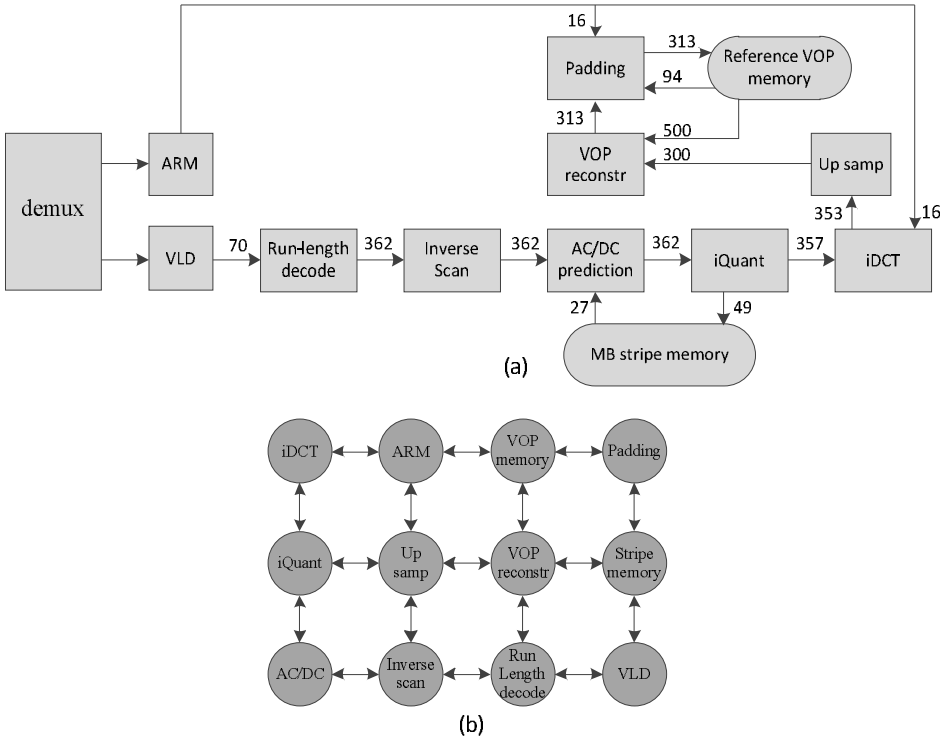

Figure 3.51: (a) VOPD block diagram, with communication BW annotated (in MB/s) (b) its mapping onto a mesh topology [19]

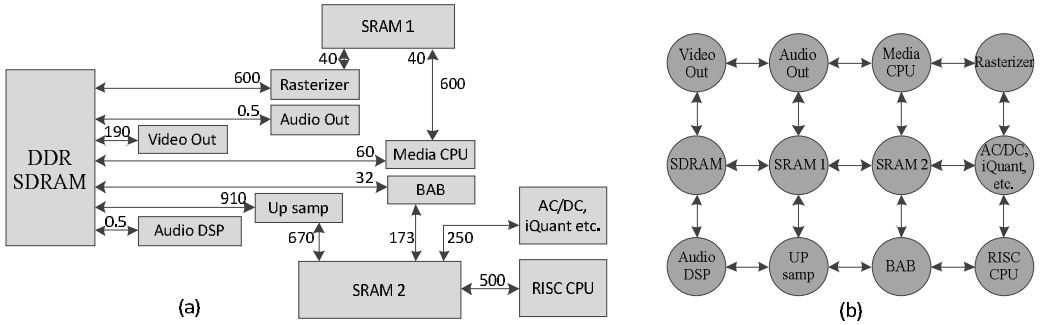

Figure 3.52: (a) MPEG4 decoder block diagram, with communication BW annotated (in MB/s) (b) its mapping onto a mesh topology [19]

3.4.3.4 Hardware Overhead

For appraising the area overhead of the switch utilizing the proposed fuzzy-logic, each scheme was synthesized by Synopsys Design Compiler using the TSMC 65nm technology with an operating point of 500MHz and supply voltage of 1V. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation of each scheme is calculated under the uniform traffic profile near the saturation point using Synopsys PrimePower in an 8×8 mesh network. The layout area and power consumption of each platform are shown in Table 3.10. Comparing the area cost and power consumption of the switches using DyXY and NFRA and the one employing FRA, indicates that the hardware overhead of implementing a switch using the fuzzy-logic is about 1.5% larger than two other methods. Although FRA seems to be a complicated method, its implementation is easy and straightforward. The power consumption is improved in both NFRA and FRA in comparison with DyXY. This is due to the fact that these methods perform better in sending packets over less congested areas.

Table 3.10: Hardware implementation details

Network platforms	Area (mm2)	Power (mw)
DyXY	2.563	1.355
NFRA	2.566	1.342
FRA	2.604	1.333

3.5 Summary of the Proposed Methods

In this chapter, we proposed three main congestion-aware routing approaches as cluster-based, learning-based and fuzzy-based routing algorithms. AgRA and TRA are the proposed cluster-based approaches, HARAQ is a leaning-based and FRA is a fuzzy-based method. These methods are mainly compared with three conventional methods, called DyXY, NoP and DBAR. Among all the algorithms, HARAQ leads to the best performance

but at the cost of employing routing tables. Agent-based methods do not use any routing tables while utilize the congestion information of the surrounding regions. This information leads to near-optimal routing decisions. The fuzzy-based algorithm can be applied on top of every congestion-aware method (except learning methods) to guarantee the best routing decision based on the collected congestion information. The characteristics of different approaches are summarized in Table 3.11.

Table 3.11: Summarized characteristics of different congestion-aware approaches

Methods	Routing	Congestion Propagation Strategy	Wiring overhead per link	Congestion view	Use of Tables
DyXY	Minimal	Congestion wires	3 bits	1-hop neighbor	No
NoP	Minimal	Congestion wires	12 bits	2-hop neighbors	No
DBAR	Minimal	Congestion wires	n bits in n×n mesh network	Switches in rows and columns	No
AgRA (proposed)	Minimal	Lightweight congestion network	9 bits orthogonal + 3 bits diagonal	Group of four switches in two directions	No
TRA (proposed)	Minimal	Congestion wires	From 3 to 9 bits	5 switches from each direction (Trapezoid positions)	No
HARAQ (proposed)	Non-minimal	Congestion wires	4 bits	Whole network (global view)	Yes 24 bytes
FRA (proposed)	Minimal	Congestion wires	10 bits	1-hop neighbor	Yes ~ 6 bytes

Chapter 4

Fault-Tolerant Routing Algorithms for a 2D Mesh Network

On-chip interconnects implemented with a deep submicron semiconductor technology, running at GHz clock frequencies are prone to failures [76], [77]. Due to extreme device scaling, the likelihood of failures increases [78]. These failures may have architectural level ramifications as it may cause an entire on-chip network to fail. Since on-chip communication reliability is a crucial factor in many-core systems, the NoC paradigm should address these reliability issues. Fault-tolerant routing algorithms play an important role in this domain by bypassing faults in the network and allowing the system to continue functioning.

In NoCs, faults may occur in cores (such as processing elements and memory modules), links or switches. When a core is faulty, it can be deactivated while the connected switch and links can continue functioning. Therefore, a faulty core does not affect transmitting packets between the other cores. However, once a link or switch has failed, the faulty component cannot be simply discarded as it results in the blocking of other packets inside the network. Fault-tolerant routing algorithms can be classified into two groups: addressing faulty links, and addressing faulty switches. Usually, the algorithms within each group can be modified to tolerate faults from the other group.

In this chapter, we review some well-known traditional methods tolerating faulty links and switches in the network. A common behavior in fault-tolerant approaches is that packets are routed normally in the network until they are faced with a fault. At this point, turn models or other techniques are used to reroute packets around the faults in a way that no cycles will be created in the network. The performance analysis in [79] indicates that in a 4×4 mesh network, the average packet latency can be increased by almost double when there is a single faulty switch in the network. Faults might exist forever or they might be recovered after a long period. Now, imagine how many packets should be rerouted around the fault and how it may affect performance. What if there are several faults in the network, existing for an extended period of time?

We present four fault-tolerant routing algorithms, called MD (Minimal and Defect-resilient), MAFA (Minimal and Adaptive Fault-Tolerant Algorithm), HiPFaR (High

Performance Fault-tolerant Routing), and MiCoF (Minimal-path Connection-retaining Fault-tolerant). MD and MAFA address faulty links and HiPFaR and MiCoF deal with faulty switches. The focus of these methods is to tolerate faults using the shortest paths in the network as long as such path exists. Using these algorithms, packets are possibly routed through minimal and non-faulty paths, which avoids facing faulty components and making unnecessary rerouting around them.

In sum, the idea is based on a common idiom: “prevention is better than cure”. Based on the failure rate and the desired reliability at link or switch levels, these algorithms can be combined together to tolerate both faulty switches and links in the network. On the other hand, most of the presented fault-tolerant algorithms are limited to deterministic routing algorithms, resulting in considerable performance loss. However, the proposed methods are based on fully adaptive algorithms, enabling to distribute packets over the network.

4.1 Traditional Approaches

Usually, the implementations of fault-tolerant algorithms are very complex due to various issues, such as the location of faults, the number of faults, turn model rules, etc. In the following, we review different attempts to tolerate faults either in off-chip or on-chip network.

4.1.1 A Ring-based Fault-Tolerant Routing (Extended X-Y)

A wide range of fault-tolerant algorithms tolerates faults by rerouting packets around faulty regions which have special convex or concave shapes. To form these shapes, some healthy switches should be deactivated. For example, Extended X-Y is a well-known routing algorithm presented in [80]. This algorithm is designed based on XY routing and the odd-even turn model [81]. Similar to the odd-even turn model, this algorithm is deadlock-free without using any virtual channels by prohibiting certain turns in odd and even columns. In fault-free cases and depending on the position of the source and destination switch, this algorithm may perform the same way as the XY routing algorithm (minimal routing), or take a longer path (non-minimal routing). In more details, if the source switch is located in an even column, the packet is sent to the Y dimension; otherwise it has to take a hop to the west direction to reach an even column before making a turn toward the Y dimension. The packet follows YX routing until it reaches the destination switch. When the packet faces a fault along its path, it has to be routed around the fault based on some specific rules.

Two examples of Extended X-Y are shown in Figure 4.1(a). As can be seen in this figure, packets have to take a very long path while they could be simply routed through the shortest paths. It is worth mentioning that, this algorithm has many restrictions on the location of faults (e.g. faults cannot be tolerated on borderline switches and there should be enough distance between faulty regions). Extended X-Y only knows about the fault statuses of its direct neighbors to make its routing decision (Figure 4.1(b)).

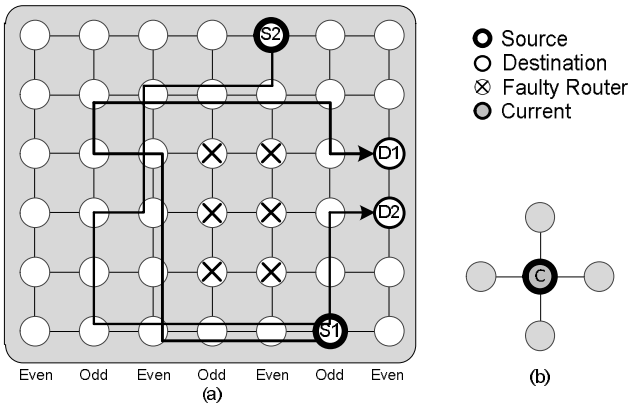

Figure 4.1: (a) Two examples of the Extended X-Y routing algorithm (b) The required fault information

4.1.2 Reconfigurable Routing for Tolerating Faulty Switches (ReRS)

Z. Zhang et al. presented a reconfigurable routing algorithm [79] to tolerate any single faulty switch in a mesh network without using virtual channels and disabling healthy switches. We call this reconfigurable routing scheme, ReRS. This algorithm provides the possibility of routing packets through a cycle free contour surrounding a faulty switch. To tolerate more number of faulty switches, the contours must not be overlapped and thus faulty switches should be located far away from each other. In other words, ReRS can tolerate a single faulty switch in the network or multiple faulty switches if their contours do not overlap. This algorithm is deterministic and does not make any effort toward alleviating congestion in the network. This method shows that cycles can naturally be avoided in borderline switches. Two examples of this method are shown in Figure 4.2(a). Packets are routed normally inside the network using a deterministic method and they have to turn around a fault when facing to it. Each switch should be informed about the fault statuses of eight direct and indirect neighboring switches (Figure 4.2(b)).

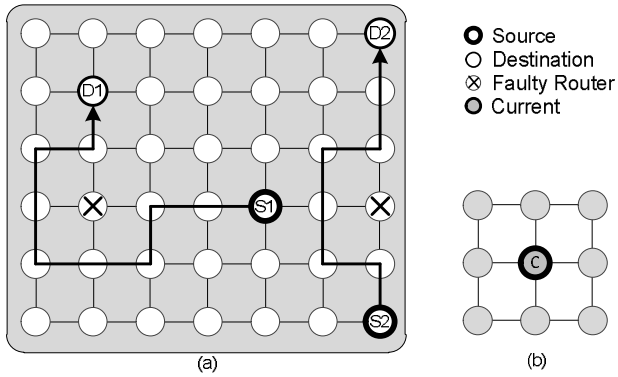

Figure 4.2: (a) Two examples of the ReRS routing algorithm (b) The required fault information

4.1.3 Reconfigurable Routing for Tolerating Faulty Links (RAFT)

ReRs has been extended in the RAFT method [82] to tolerate two faulty links. For this purpose, RAFT requires two virtual channels along both X and Y dimensions. This algorithm investigates a cycle free contour for all combinations of one and two faulty links. This algorithm is very complicated and has many exceptional rules. Two examples of this algorithm are illustrated in Figure 4.3(a). RAFT is an adaptive method and is able to deliver packets through multiple paths. This reduces latency as the probability of sending packets through the shortest paths increases. In RAFT, each switch needs to know the fault statuses of twelve surrounding links (Figure 4.3(b)).

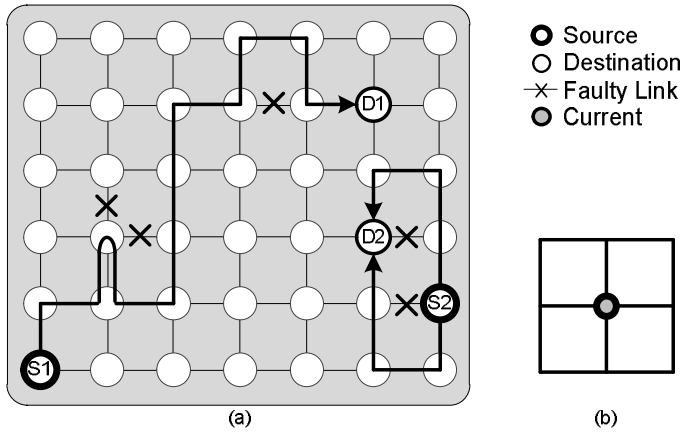

Figure 4.3: (a) Two examples of the RAFT routing algorithm (b) The required fault information

4.1.4 Bidirectional Fault-Tolerant NoC (BFT-NoC)

A different fault-tolerant approach, called BFT-NoC, is presented in [76]. Normally, two channels are used for transmitting and receiving packets between two adjacent switches. In fault-free cases, one channel is dedicated to transmitting packets and another one for receiving packets. The idea of BFT-NoC is to share a channel for both transmitting and receiving packets when one of the channels is faulty. This method reduces packet latencies by avoiding costly rerouting of packets. By sharing the resources, the link performs as a bottleneck in high traffic loads. To reduce the load on the partially functioning links, it might be better to deliver packets through alternative routes. This is the solution that can be offered by routing algorithms. Obviously, BTF-NoC cannot be used to address faulty switches or a total link failure in both unidirectional directions.

4.1.5 Summary of Traditional Methods

Most of the traditional fault-tolerant routing algorithms reroute packets around faulty regions, either convex or concave, so that the selected paths are not always the shortest ones. However, rerouting is an expensive solution and affects performance significantly not only by taking longer paths but also by creating hotspot around a fault. On the other hand,

most of the presented fault-tolerant algorithms are limited to deterministic routing algorithms, resulting in considerable performance loss. Traditional fault-tolerant algorithms are relatively complex due to considering different fault models and location of faults. Fault-tolerant algorithms may use virtual channels [78], [82] or do not use any virtual channels [80], [83]. The virtual channel based fault-tolerant routing algorithms provide better fault-tolerant characteristics than those without virtual channels, but they are not usually cost efficient. The methods that do not use any virtual channel are mainly based on the turn models [49].

4.2 The Proposed Approaches for Tolerating Faulty Links

In this section, we first start by introducing a method called MD, Minimal and Defect-resilient routing algorithm [84]. This method targets addressing a total link failure where the key ideas are twofold: First, it can tolerate any single-link faults using the shortest path between each pair of source and destination switches, if a path exists. Second, to avoid congestion, output channels can be adaptively chosen whenever the distance from the current to the destination switch is greater than one hop along both dimensions.

MD takes advantages of one and two virtual channels along the X and Y dimensions. This idea of MD is extended in the second approach, called Minimal and Adaptive Fault-Tolerant Algorithm (MAFA) [85], to make a network resilient against two faulty links. Increasing reliability is at the cost of using an extra virtual channel along the Y dimension in MAFA than MD. In total, MAFA uses two virtual channels along both dimensions.

4.2.1 Any Single Faulty Link (MD)

MD is able to tolerate any single faulty link in the network. It means that there might be only one fault in the network or it can be multiple faults with enough distances from each other such that faults are handled independently. The required distance between faults varies in different methods.

4.2.1.1 Fault Distribution Mechanism

The RAFT approach [82] requires the statuses of twelve links to make its routing decision (Figure 4.3(b) or Figure 4.4(a)). MD needs to know the statuses of fewer number of links (i.e. eight links as shown in Figure 4.4(b)). Using this information, MD knows the faulty conditions of the following paths: east, west, north, south, northeast, northwest, southeast, and southwest.

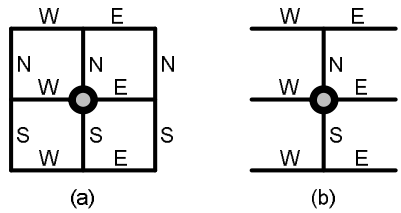

Figure 4.4: (a) Statuses of twelve links are needed by RAFT (b) The statuses of eight links are needed by MD

4.2.1.2 Turn Model in MD

MD utilizes one and two virtual channels along the X and Y dimensions. The turns to be prohibited in each virtual channel are taken from the Mad-y method [57]. To prove deadlock-freeness, we use a numbering mechanism similar to the Mad-y method [57] (Note that the proof has been also given in Section 3.3.1). This numbering mechanism shows that all the turns have occurred only in ascending order, and thus no cycle can be formed in the network. A two-digit number (a,b) is assigned to each output channel of a switch in an $n \times m$ mesh network. According to the numbering mechanism, a turn connecting the input channel (I_a,I_b) to the output channel (O_a,O_b) is called an ascending turn when ($O_a > I_a$) or (($O_a = I_a$) and ($O_b > I_b$)). Figure 4.6 shows how the channels of a switch at the position (x,y) are numbered. By using this numbering mechanism, it is guaranteed that all allowable turns in Figure 4.5 are taken in the strictly increasing order, so that the MD routing algorithm is deadlock-free. For instance, if the E-N1 turn (i.e. a packet moving to the east direction makes a turn to the north direction using the first virtual channel) is taken into consideration, the west input channel with label (I_a=m+x, I_b=0) is connected to the first virtual channel of the north output port having the label (O_a=m+x, O_b=1+y). This turn takes place in an ascending order since (($O_a = I_a$) and ($O_b > I_b$)). Similarly, all the other turns allowed by MD are taken in an ascending order.

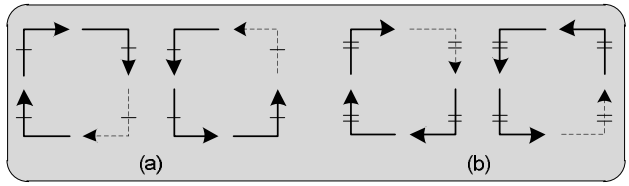

Figure 4.5: Permitted and prohibited turns of MD similar to Mad-y

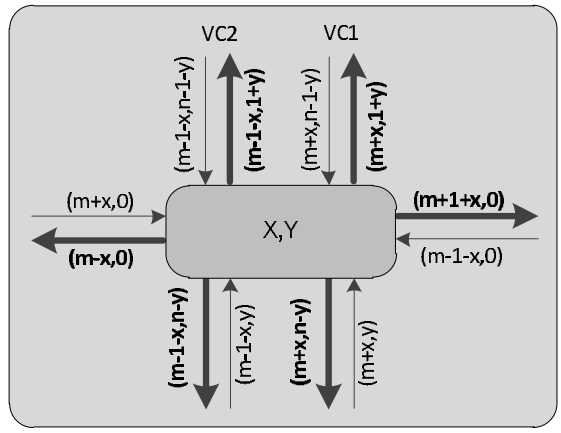

Figure 4.6: The numbering mechanism of MD similar to Mad-y

4.2.1.3 Tolerating Faulty Links

A destination switch might be located in eight different positions of a source switch as north, south, east, west, northeast, northwest, southeast, and southwest. In MD, packets are able to use only the shortest paths in the presence of a faulty link when the destination is in the northeast, northwest, southeast, and southwest directions of a source switch. This is achieved by knowing about the statues of eight links and taking advantage of a fully adaptive routing algorithm. Therefore, no rerouting takes place in these cases and the algorithm remains deadlock-free. However, for east-, west-, north-, and south-ward packets, non-minimal paths must be taken if a faulty link exists in the path.

Using a fully adaptive routing algorithm, all the shortest paths in the east direction are valid for eastward packets. Similarly, westward packets can utilize all shortest paths in the west direction. When the destination is in the northeast position of the current switch, the packet can be delivered in either the north or east direction. As illustrated in Figure 4.7(a), the distances along both east and north directions are one. On the other hand, the current switch knows about the fault statuses of the links in E and NE paths. Using this information, if a link is faulty in the NE path, the packet is routed through the east direction and could safely reach the destination (assuming that there is no other fault in the path). As a result, the packet is routed through a minimal path to the destination switch.

An example in Figure 4.7(b) shows the case where the distance along the X dimension reaches one while the distance along the Y dimension is greater than one. The current switch knows about the fault statues of E and N links. In MD, the packet is sent to the Y dimension as long as it is non-faulty; otherwise the X dimension is selected. The reason for this decision is that if the distance along the X dimension reaches zero, the packet has to take the Y direction in the remaining path toward the destination switch. Thereby, if there is a faulty link in the Y dimension, the packet must take a non-minimal route to bypass the fault. This is not an optimal solution which is addressed by MD. MD avoids reducing the distance into zero in one direction when the distance along the other direction is greater than one. Therefore, when the distance between the current and destination switches reaches one in at least one dimension, at first the possibility of sending the packet to the greater-distance dimension is checked. The packet is sent along the greater-distance dimension if the link toward this direction is non-faulty; otherwise the smaller-distance dimension is examined. Consequently, in Figure 4.7(b) the availability of the N link is checked before than that of the E link, and the packet is sent to the N link if it is non-faulty. In the next hop, the packet faces the similar situation as in Figure 4.7(a), and thus only the shortest paths are selected by MD so far. Similarly, in Figure 4.7(c) the condition of the E link is examined earlier than that of the N link.

Finally, in Figure 4.7(d), the east and north links have the same priority to be selected, so the congestion values are checked to choose between the non-faulty directions. By these choices, the packet faces a similar situation as in Figure 4.7(b) or Figure 4.7(c) and thereby only the shortest paths are taken by MD in all cases. The idea can be simply extended to the northwest, southeast, and southwest packets in the network.

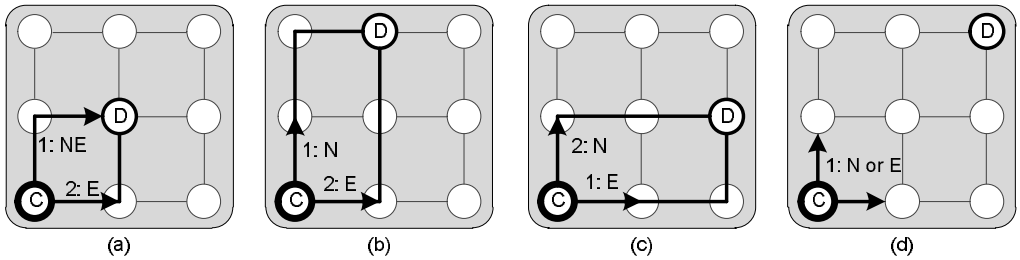

Figure 4.7: Bypassing faulty links when the destination is located in the northeast position of the source switch (Note that numbers determine the priority of selecting among different routes)

As it is already mentioned, when a packet is east-, west-, north-, or south- bounded and there is a faulty link in the path, the packet must be rerouted through a non-minimal path around the faulty link. As illustrated in Figure 4.8(a), for the eastward packet, at first the east link is checked and if it is non-faulty, the packet is sent through this direction. However, if the link is faulty, the packet is delivered to the north or south direction. The situation is similar for the westward packet (Figure 4.8(b)). In this case, the fault information in the west direction is checked before those of the north and south directions. For a northward packet facing a faulty link in the north direction (Figure 4.8(c)), the west direction is checked earlier than the east direction. It means that the east direction is used only when the faulty link is located in the left borderline. A similar perspective is applied to southward packets (Figure 4.8(d)).

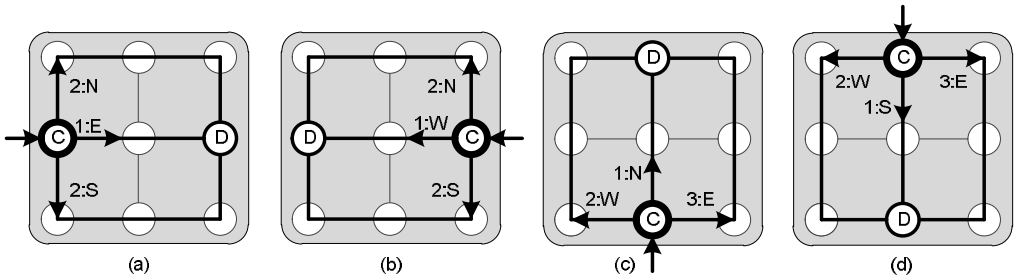

Figure 4.8: Bypassing faulty links when the destination is located in the (a) east (b) west (c) north (d) south positions of the source switch

Now, we need to show that all the required turns for bypassing faults are in the set of allowable turns. As shown in Figure 4.5(a), all required turns toward the east direction are allowable, so that all cases of faults can be tolerated for northeast-, southeast-, and east-ward packets by routing through the first virtual channel. Similarly, northwest-, southwest-, and west-ward packets are routed through the second virtual channel where all of the required turns toward the west direction are available. We only need to prove that the northward and southward packets are routed in the network using the allowable turns. As shown in Figure 4.9(a) and Figure 4.9(b), the northward and southward packets use the allowable turns of the second virtual channel to bypass the faults. In Figure 4.9(c) and

Figure 4.9(d), the packet has to make a turn to the right where the unallowable turns E-N2 and E-S2 must be taken. However, this cannot result in deadlock. This is due to the fact that a cycle cannot be formed around the faulty borderline links or switches (as indicated in [79]), thus MD remains deadlock-free.

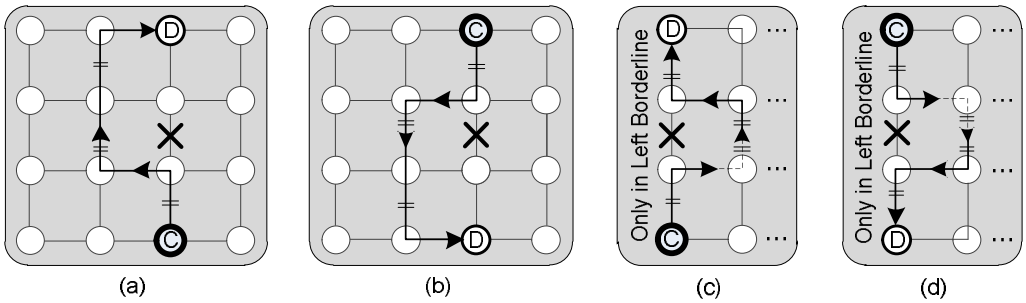

Figure 4.9: Northward and southward packets are strictly belonging to the second virtual channel (a) and (b) show the cases where the fault does not occur in the left borderline (c) and (d) show how the fault can be tolerated when the fault occur in the left borderline

4.2.1.4 MD Routing Algorithm

A deterministic routing algorithm is a common used method in traditional fault-tolerant methods. However, MD uses the deterministic routing only when a packet gets close to the region of the faulty link. Based on MD, if the distance from the current to the destination switch is greater than one hop along both dimensions, packets can adaptively choose among the non-faulty links without any restrictions. The output selection function of MD is similar to DyXY with some modifications. According to DyXY, a packet should be sent to a direction which has more number of free buffer slots in the corresponding input buffer of the neighboring switch. When both directions have the same number of free buffer slots, a direction is chosen by random. On the other hand, MD tries not to reduce the distance in one dimension to zero when the distance along the other dimension is greater than one. To step toward this goal, we try to keep the distances along both directions as equally as possible when routing packets. However, forcing a packet to choose a specific direction is against the adaptive principles.

Our solution to this issue is to send a packet to the desired direction (greater-distance dimension) whenever the number of free buffer slots are nearly equal in both directions. The number of free buffer slots does not need to be exactly the same. For instance, four or five free buffer slots out of eight available slots can be seen similar in the term of affecting performance. Therefore, when the difference between the number of free slots in two input buffers is less than or equal to two slots, the packet is sent to the greater-distance dimension. The MD routing algorithm is shown in Figure 4.10.

```
ALGORITHM: MD routing algorithm
                        **--------------------------------------------------**
Definitions:    Xs,Ys: X and Y coordinates of the source switch
                Xd,Yd: X and Y coordinates of the destination switch
                Xc,Yc: X and Y coordinates of the current switch
                vc: Virtual Channel;
                        **--------------------------------------------------**
position <= {NE,NW,SE,SW,E,W,N, or S} according to the source and destination positions

x_dir <= E when Xd > Xc else W;
y_dir <= N when Yd > Yc else S;

delta_x <= Xd - Xc when Xd>Xc else Xc-Xd;
delta_y <= Yd - Yc when Yd>Yc else Yc-Yd;

vc <= vc1 when position={E,NE,SE} else
      vc2 when position={W,N,S,NW,SW};

if position={NE, NW, SE, or SW} then
        if (delta_x>=1 and delta_y=0) then select <= x_dir;
        elsif (delta_x=0 and delta_y>=1) then select <= y_dir(vc);
        elsif (delta_x/=1 and delta_y/=1) then
                if link(y_dir)=faulty then select <= x_dir; else select <= y_dir(vc); end if:
        elsif (delta_x=1 and delta_y=1) then
                if position={NE } then
                        if link(NE)=faulty then select <= x_dir; else select <= y_dir(vc); end if:
                elsif position={NW } then
                        if link(NW)=faulty then select <= x_dir; else select <= y_dir(vc); end if:
                elsif position={SE} then
                        if link(SE)=faulty then select <= x_dir; else select <= y_dir(vc); end if:
                elsif position={SW} then
                        if link(SW)=faulty then select <= x_dir; else select <= y_dir(vc); end if:
                end if;
        end if;
elsif position={E or W} then
        if delta_y = 0 then
                if link(x_dir)=faulty then select <= N(vc) or S(vc); else select <= x_dir; end if;
        else
                if link(y_dir)=dest then select <= y_dir(vc); else select <= x_dir; end if;
        end if;
elsif position={N or S} then
        if delta_x = 0 then
                if link(y_dir)=faulty then
                        if Xc/=0 then select <= west; else select <= east; end if;
                else select <= y_dir(vc); end if;
        else
                if inPort/= {E,W} and link(x_dir)=non-faulty then select <= x_dir;
                else select <= y_dir(vc); end if;
        end if;
end if;
```

Figure 4.10: MD routing algorithm

An example of MD is shown in Figure 4.11(a), where the source and destination are located at the switches S and D while the link (11,D) is faulty. As can be seen in this figure, the packet is routed adaptively inside the network until the distance along one dimension reach one. The number of free buffer slots is used to select among the output channels at each switch. The degree of adaptiveness of MD in faulty cases (Figure 4.11(a)) is nearly equal to fault-free cases (Figure 4.11(b)).

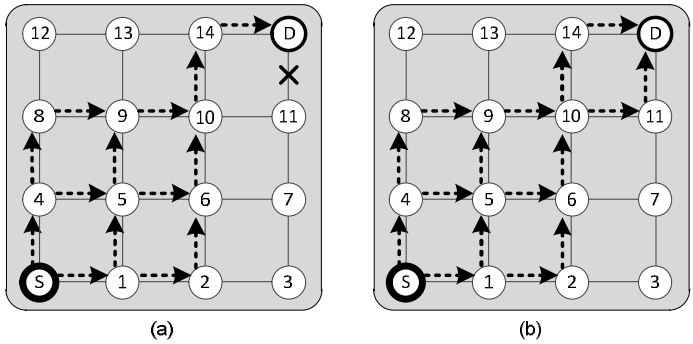

Figure 4.11: Alternative paths from the source switch S to the destination D

4.2.1.5 Analytical Analysis of Two Faulty Links

Since MD cannot tolerate all positions of two faulty links, we analyze the reliability of MD when two faults occur in the network. Let us assume that the location of faults is chosen randomly. We divide the problem into two cases: the first fault occurs on the border links or the first fault occurs in central links. If the first fault occurs on one of the borderline links, the second fault must not happen on some specific locations. For example, in Figure 4.12(a) if the link 2 is faulty, the locations marked with cross-cycle signs must be healthy. A similar situation exists for all the other borderline links. The probability that the first faulty link occurs on the border links and the second one does not occur in specific locations is calculated as follow:

$$R_{fault-border} = \frac{4(n-1)}{2n(n-1)} * \left(1 - \frac{3(n-1) + (n-2)}{2n(n-1)}\right) = \frac{2}{n} * (1 - \frac{4n-5}{2n(n-1)})$$

where the terms *2n(n-1)* and *4(n-1)* indicate the total number of links and the number of borderline links, respectively, in an $n \times n$ mesh network. The second term measures the number of locations in which the second fault can be tolerated.

Similarly, when the first fault occurs in a central link (Figure 4.12(b)), the robustness is obtained by the following formula:

$$R_{fault-center} = \frac{2(n-1)(n-2)}{2n(n-1)} * \left(1 - \frac{4 * (n-1) + (n-2)}{2n(n-1)}\right) = \frac{(n-2)}{n} * (1 - \frac{5n-6}{2n(n-1)})$$

where the term *2(n-1)(n-2)* indicates the number of central links. The second term measures the number of locations in which the second fault can be tolerated.

According to these formulas, Table 4.1 shows the robustness in different network sizes when two faults occur in the network. These values are based on this assumption that all packets reach their destinations. Thereby, if only one packet cannot reach the destination, the network is tagged as unreliable.

Table 4.1: Robustness analysis of two faulty links in different network sizes

Network size	Robustness against two faulty links
4×4	48%
6×6	63%
8×8	71%
16×16	85%

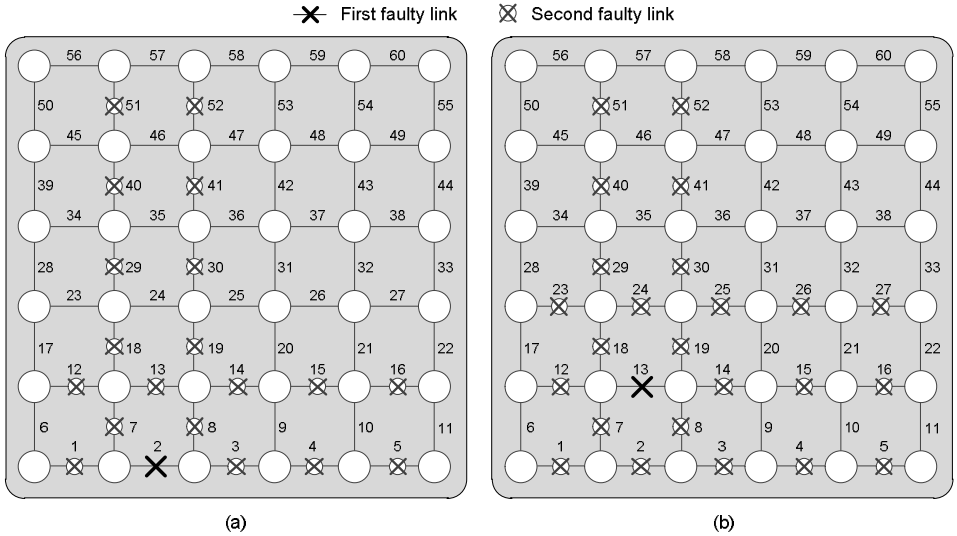

Figure 4.12: The first fault occurs in (a) borderline link (b) central link

4.2.2 Multiple Faulty Links (MAFA)

MD is a good candidate when an algorithm is targeted for a high performance design while some degree of reliability is valued. MD works nearly similar to a fully adaptive method with a negligible performance loss when the network becomes faulty. In faulty conditions, MD possibly uses minimal paths while it is able to tolerate any single faulty link and some multiple faulty links. However, if the design was targeted for high reliability, MD would not be a good candidate. In this section, we present an algorithm, called Minimal and Adaptive Fault-Tolerant Algorithm (MAFA), which is able to tolerate all one and two faulty links in the network using two virtual channels in both X and Y dimensions. This

algorithm is designed to retain the adaptivity in the network while tolerating a higher number of faults.

The basic version of the proposed method is able to tolerate all two faulty links by using the shortest paths as long as a path exists. Then we improve the method to address multiple faulty links by taking advantages of non-minimal paths. In this method, we find out all possible transactions between two virtual channels, such that a packet being routed in one virtual channel can switch to the next one. Similar to the MD approach, MAFA avoids reducing the distance into zero in one direction when the distance along the other direction is greater than one.

4.2.2.1 Fault Distribution Mechanism

MAFA utilizes a new fault distribution mechanism. As shown in Figure 4.13(d), fault information is distributed in a way that each switch is informed about the faulty links of its 2-hop distance. To collect this information, it is enough that each switch transfers the faulty information on its links to the neighbors. In Figure 4.13(a), the neighboring switch in the north of the current switch (C) transfers the fault information on its links in N, E, and W directions to the current switch. Accordingly, the current switch would be informed about the fault information in its N, NN, NE, and NW paths. In Figure 4.13(b), by receiving the fault information from the east neighboring switch, the current switch knows about the fault information in E, EE, EN, and ES paths as well. In addition, the statuses of ENW and NES paths are also obtained. Similarly, in Figure 4.13(c), this knowledge is extended to know about the fault information in S, SS, SE, SW, ESW, and SEN paths. Finally, as shown in Figure 4.13(d) by receiving the information from the west neighboring switch, the current switch has the information about the links in E, W, N, S, EE, EN, ES, WW, WN, WS, NN, NE, NW, SS, SE, SW, ENW, ESW, WNE, WSE, NES, NWS, SEN, and SWN paths in total.

For routing a packet to the northeast direction, a switch uses the fault information on the links located in either minimal paths (i.e. EE, EN, NE, and NN) or non-minimal paths (e.g. SE, WN, WW, and SS). Similarly, for a northward packet, the fault statues on some links (e.g. N, E, W, NN, NE, NW, ENW, and WNE) are beneficial for making a reliable routing decision.

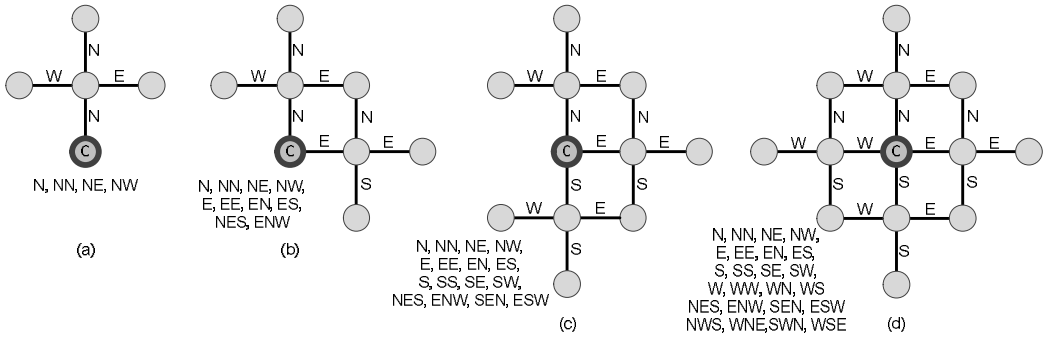

Figure 4.13: Fault distribution mechanism

4.2.2.2 Turn Model in MAFA

MAFA uses two virtual channels along both dimensions. The turns to be prohibited in each virtual channel are inspired from the method in [86]. According to this method, the routing algorithm used in the vc1 is west-last where the west-to-north (W1-N1) and west-to-south (W1-S1) turns cannot be taken by the packets. In the vc2, the east-last routing algorithm is utilized where the east direction cannot be taken earlier than the other directions. The allowable and unallowable turns in vc1 and vc2, are shown in Figure 4.14(a) and Figure 4.14(b), respectively. Eastward packets are routed in the first virtual channel and they are able to use all minimal routes in the east direction. Moreover, the remaining turns are used for non-minimal routing if needed. Similarly, westward packets are routed in the second virtual channel to utilize all alternative paths to send packets in the west direction and the remaining turns are utilized for non-minimal purposes. Since the algorithms are deadlock-free within each virtual channel, so that the whole network is deadlock-free. We have modified this method allowing transactions between two virtual channels without forming any cycles. The simple idea is that a cycle cannot be formed if packets could switch from vc1 to vc2 but not vice versa. Based on this idea, many other turns (0-degree, 90-degree, 180-degree) are added into the list of allowable turns as shown in Figure 4.14(c). Now, we have an extensive set of allowable turns in the network to be used for rerouting packets and tolerating faults. According to MAFA, all packets must be routed in the first virtual channel for as long as possible. They switch to the second virtual channel only when no minimal option is available in the first virtual channel. This situation may happen when a packet faces a faulty link.

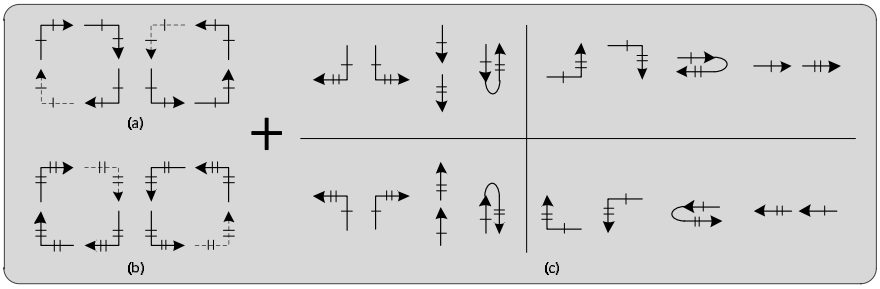

Figure 4.14: (a) west-last (b) east-last (c) all permitted transactions between vc1 and vc2

4.2.2.3 Tolerating One Faulty Link by MAFA

MAFA is able to tolerate any single faulty link. Figure 4.15 shows the cases where the destination is located in the northeast, southeast, northwest, and southwest of the current switch and the distances along both X and Y dimensions are one. All the turns used in this figure are in the set of allowable turns offered by MAFA. As shown in Figure 4.15(a) and Figure 4.15(b), different fault locations are supported using the first virtual channel when the destination switch is in the east position of the current switch. When the destination is in the west position of the current switch, packets may switch from the first to the second virtual channel as indicated in Figure 4.15(c) and Figure 4.15(d).

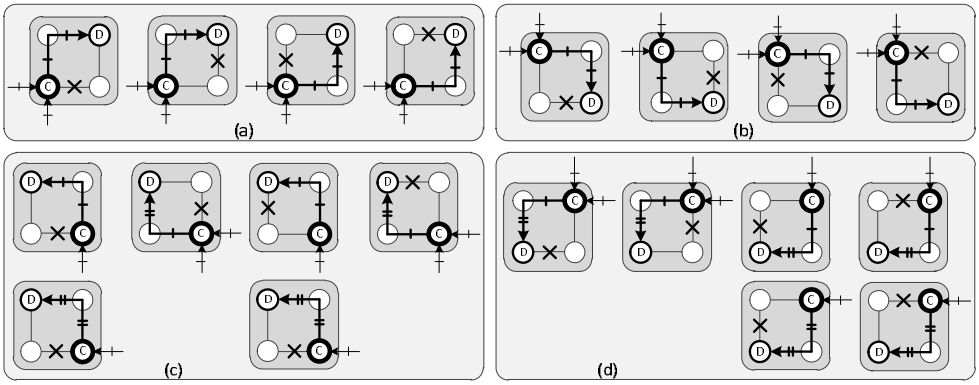

Figure 4.15: Tolerating one faulty link by MAFA when the destination is in the (a) northeast (b) southeast (c) northwest (d) southwest of the current switch

Figure 4.16 covers the cases when the destination is in the north, south, east, and west of the current switch. As shown in Figure 4.16(a), normally northward packets are routed around the fault from the east side using the first virtual channel. However, if the fault is located in the right borderline, packets have to make a turn to the west direction. This rerouting can be done by using the first virtual channel even if the west-north turn is prohibited. In fact, a cycle cannot be formed in borderline cases. Similar perspective is applied to Figure 4.16(b) for tolerating faults in the south direction. Eastward packets can turn around the fault either through the north or south direction, and all the required turns are allowable. Westward packets can bypass the fault by rerouting to the north or south direction while switching to the second virtual channel. All turns are in a set of allowable turns and can be safely taken.

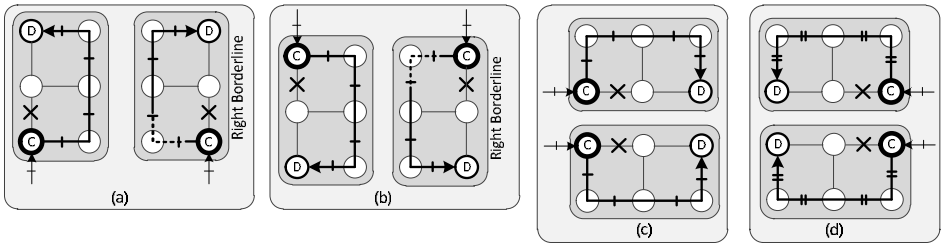

Figure 4.16: Tolerating one faulty link by MAFA when the destination is in the (a) north (b) south (c) east (d) west positions of the current switch

4.2.2.4 Tolerating Multiple Faulty Links by MAFA

We start by investigating the cases of multiple faulty links when the destination is in the northeast, southeast, northwest, and southwest positions of the current switch. If the distance from the current to the destination switch is one hop along both the X and Y dimensions, at maximum, two minimal paths exist. If both of them are faulty, non-minimal routes should be checked. The priorities of different routes (either minimal or non-minimal) are illustrated in Figure 4.17 for northeast, southeast, northwest, and southwest packets

when the distance from the current to the destination switch is one hop along both dimensions. For instance, when the destination is in the northeast of the current switch, at first the availability of NE and EN paths are checked, if both of them are faulty, the statuses of NN and EE paths are examined. If the links in these paths are also faulty, the possibility of sending the packet through the SE path is considered. In addition to the SE path, the availability of the south link of the destination switch should be also checked. Finally, if all of the mentioned options were unsuccessful, the packet is delivered to the west direction.

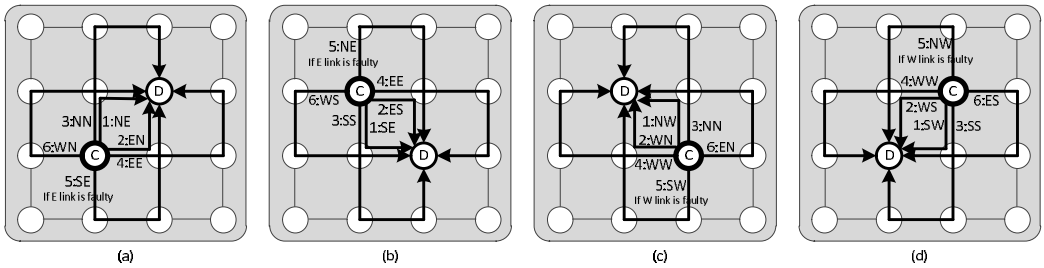

Figure 4.17: Priority of selecting among different routes when the destination is in the (a) northeast (b) southeast (c) northwest (d) southwest positions of the current switch

Now, we should prove that all paths can be taken without creating cycles in the network. In other words, all of the required turns should be selected from the set of allowable turns. The first row in Figure 4.18 shows all the six different positions of two faulty links regarding the current and destination switches for a northeast packet. As shown in Figure 4.17(a), the priority of selecting among different paths for a northeast packet is as follow: NE, EN, NN, EE, SE (if the E link is faulty), and WN. In Figure 4.18(a), the NE path is non-faulty and the packet is sent to the north direction while in Figure 4.18(b), the east direction is a suitable path for delivering the packet. In Figure 4.18(c), both the NE and EN paths are faulty, and thus the availability of the NN path is examined. If the NN path is not available (e.g. the destination is located in the top borderline), the EE path is checked. As shown in Figure 4.18(d), faults are already bypassed by the packet prior to reaching the position of the current switch unless the packet is generated at the current switch (i.e. current=source). In this case, there are two options to forward the packet to the destination switch. The possibility of sending the packet to the south direction is checked earlier than the west direction. In Figure 4.18(e), the NE and EN paths are faulty while the NN path might be available, so the packet may be sent to the north direction. If the NN path is not available, the next option is to check the SE path. If the packet is coming from the west direction through the first virtual channel, it is routed to the south direction within the same virtual channel to reach the destination switch. If the packet arrives from the south input port, it should switch to the second virtual channel. Doing this, the packet has to take the unallowable turn E2-N2. However, this turn can be safely taken as the cycle cannot be completed in borderline cases. Finally, in Figure 4.18(f), after checking the NE, EN, and NN paths, the EE path is found as a non-faulty path and the suitable choice for sending the packet. If the EE path is not available, the next priority is the SE path. In this example, as

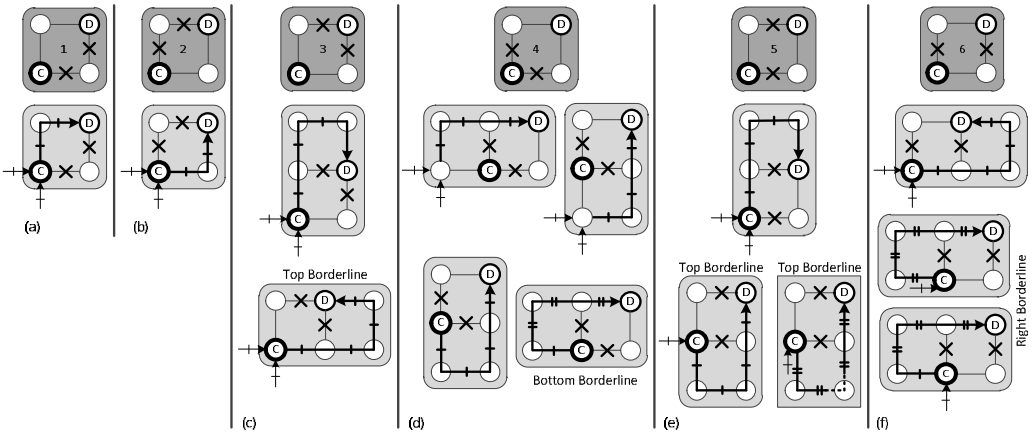

Figure 4.18: Different positions of two faulty links for a northeast packet

the south link of the destination switch is faulty, the packet cannot reach the destination switch. Therefore, the availability of the WN path is checked. Faults can be handled similarly for southeast packets.

Westward packets are normally routed in the first virtual channel and then they may switch to the second virtual channel when facing a fault. Figure 4.19 shows a southwest packet facing faulty links in its path. As packets can always switch from the first to the second virtual channel, for simplicity we assume that the packet has just switched to the second virtual channel at the current switch. Even under this worst-case assumption, all the required turns are in the set of allowable turns. Faults can be similarly tolerated for northwest packets.

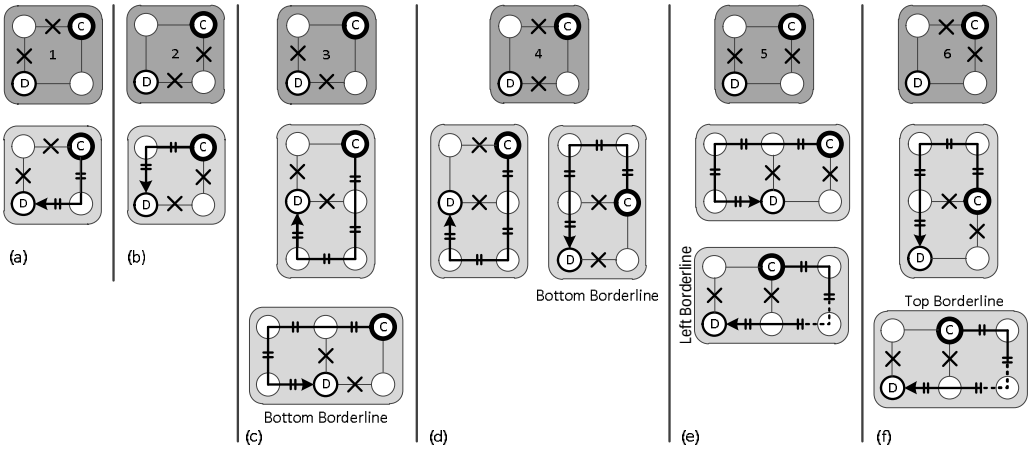

Figure 4.19: Different positions of two faulty links for a southwest packet

Figure 4.20 shows the priority of different routes when the destination is in the north, south, east and west positions of the current switch. The first row covers the cases where the packet is in the 1-hop distance from the destination switch while the second row covers the rest of the cases.

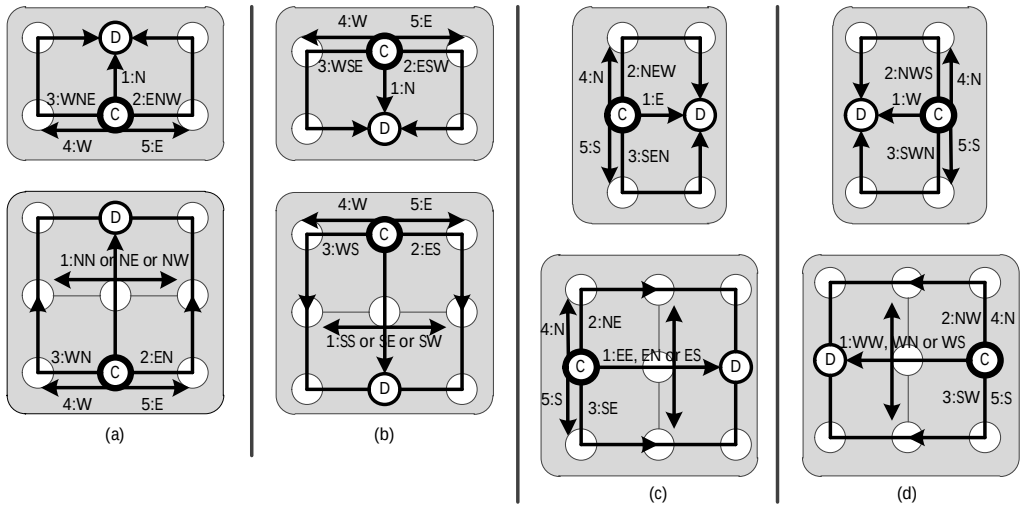

Figure 4.20: Priority of different routes for (a) northward (b) southward (c) eastward (d) westward packets

4.2.2.5 MAFA Routing Algorithm

The MAFA routing algorithm for northeast, northwest, southeast, and southwest packets is shown in Figure 4.21. As long as the packet has not reached close to the destination switch, it is sent to the greater-distance dimension. If all the minimal paths in 2-hop distance along this direction are faulty, the availability of the minimal options along the other dimension is examined. If all the minimal routes along this dimension are also faulty, the packet is sent to a non-minimal direction based on the priorities shown in Figure 4.17.

In fault-free cases, MAFA performs similar to a fully adaptive routing algorithm. In faulty cases, the fully adaptiveness is limited to eastward packets while westward packets are routed deterministically. The reason for this limitation is that all packets are routed in the first virtual channel for as long as possible where the turns toward the west direction are limited to deterministic routing.

The MAFA routing algorithm for north-, south-, east-, and west-ward packets is shown in Figure 4.22. In fault-free cases, packets follow the shortest paths. However, in faulty situations, packets have to be rerouted around faults based on the priorities defined in Figure 4.20.

```
ALGORITHM: Tolerating faults by MAFA for northeast, northwest, southeast, and southwest packets
                        **-------------------------------------------------**
Definitions:    Xd,Yd: X and Y coordinates of the destination switch
                Xc,Yc: X and Y coordinates of the current switch
                in, vc: input channel; virtual channel
                dLoc: Location of the destination switch regarding the current switch
                        **-------------------------------------------------**
deltaX <= (Xd - Xc) when Xd > Xc else (Xc - Xd);
deltaY <= (Yd - Yc) when Yd > Yc else (Yc - Yd);
vc <= '0' when (((select = east) and (in = W1,N1,S1, or L)) or
                ((select = west) and (in = E1,N1,S1, or L)) or
                ((select = north) and (in = W1,S1, or L)) or
                ((select = south) and (in = W1,N1, or L))) else '1';

if (dLoc=local) then select <= local;
elsif (dLoc=northeast) then
        if not(DeltaX=1 and DeltaY=1) and ((N='1') or (E='1')) then
                if ((DeltaY>=DeltaX) and (NE='1' or NN='1')) then select <= north;
                elsif (EE='1' or EN='1') then select <= east; else select <= north; end if;
        else
                if (NE='1') or (EN='0' and NN='1') then select <= north;
                elsif (EN='1') or (EE='1') then select <= east;
                elsif (SE='1') and (E='0') then select <= south; else select <= west; end if;
        end if;
elsif (dLoc=northwest) then
        if not(DeltaX=1 and DeltaY=1) and ((N='1') or (W='1')) then
                if ((DeltaY>=DeltaX) and (NW='1' or NN='1')) then select <= north;
                elsif (WW='1' or WN='1') then select <= west; else select <= north; end if;
        else
                if (NW='1') or (WN='0' and NN='1') then select <= north;
                elsif (WN='1') or (WW='1') then select <= west;
                elsif (SW='1') and (W='0') then select <= south; else select <= east; end if;
        end if;
elsif (dLoc=southeast) then
        if not(DeltaX=1 and DeltaY=1) and ((S='1') or (E='1')) then
                if ((DeltaY>=DeltaX) and (SE='1' or SS='1')) then select <= south;
                elsif (EE='1' or ES='1') then select <= east; else select <= south; end if;
        else
                if (SE='1') or (ES='0' and SS='1') then select <= south;
                elsif (ES='1') or (EE='1') then select <= east;
                elsif (NE='1') and (E='0') then select <= north; else select <= east; end if;
        end if;
elsif (dLoc=southwest) then
        if not(DeltaX=1 and DeltaY=1) and ((S='1') or (W='1')) then
                if ((DeltaY>=DeltaX) and (SW='1' or SS='1')) then select <= south;
                elsif (WW='1' or WS='1') then select <= west; else select <= south; end if;
        else
                if (SW='1') or (WS='0' and SS='1') then select <= south;
                elsif (WS='1') or (WW='1') then select <= west;
                elsif (NW='1') and (W='0') then select <= north; else select <= east; end if;
        end if;
end if;
```

Figure 4.21: MAFA routing algorithm for northeast, northwest, southeast, and southwest packets

```
ALGORITHM: Tolerating faults by MAFA for north-, south-, east-, and west-ward packets
                        **--------------------------------------------------**
Definitions:    Xd,Yd: X and Y coordinates of the destination switch
                Xc,Yc: X and Y coordinates of the current switch
                in,vc: Input channel; virtual channel
                dLoc: Location of the destination switch regarding the current switch
                        **--------------------------------------------------**
vc <= '0' when (((select = east) and (in = W1,N1,S1, or L)) or
               ((select = west) and (in = E1,N1,S1, or L)) or
               ((select = north) and (in = W1,S1, or L)) or
               ((select = south) and (in = W1,N1, or L))) else '1';

if (dLoc=north) then
        if (N='1') and ((NN='1' or NE='1' or NW='1') or (ly+1=dy)) then select <= north;
        elsif (ENW='1') and (in=W1,N1,S1, or L) then select <= east;
        elsif (WNE='1') and (in=E1,N1,S1, or L) then select <= west;
        elsif (E='1') then select <= east;
        elsif (W='1') then select <= west; else select <= south;
        end if;
elsif (dLoc=south) then
        if (S='1') and ((SS='1' or SE='1' or SW='1') or (ly-1=dy)) then select <= south;
        elsif (ESW='1') and (in=W1,N1,S1, or L) then select <= east;
        elsif (WSE='1') and (in=E1,N1,S1, or L) then select <= west;
        elsif (E='1') then select <= east;
        elsif (W='1') then select <= west; else select <= north;
        end if;
elsif (dLoc=east) then
        if (E='1') and ((EE='1' or EN='1' or ES='1') or (lx+1=dx)) then select <= east;
        elsif (NES='1') and (in=W1,S1, or L) then select <= north;
        elsif (SEN='1') and (in=W1,N1, or L) then select <= south;
        elsif (N='1') then select <= north;
        elsif (S='1') then select <= south; else select <= west;
        end if;
elsif (dLoc=west) then
        if (W='1') and ((WW='1' or WN='1' or WS='1') or (lx-1=dx)) then select <= west;
        elsif (NWS='1') and (in=W1,S1, or L) then select <= north;
        elsif (SWN='1') and (in=W1,N1, or L) then select <= south;
        elsif (N='1') then select <= north;
        elsif (S='1') then select <= south; else select <= east;
        end if;
end if;
```

Figure 4.22: MAFA routing algorithm for north-, south-, east-, and west-ward packets

4.2.2.6 Enhanced-MAFA

MAFA is designed to tolerate all situations of two faulty links in the network, but it is not the maximum reliability that can be provided by this method. For example, in Figure 4.23(a) the statuses of the NE, EN, NN, EE, SE, and WN links are enough to cover all situations of two faulty links for a northeast packet. However, all of these routes might not be available when there are more faults in the network. The aim of Enhanced-MAFA is to utilize almost all the possible paths for rerouting packets.

When a packet enters a switch through one of the input channels (i.e. L, N1, N2, S1, S2, E1, E2, W1, and W2), the routing unit determines one or several potential output channels to deliver the packet. A switch receiving a packet needs to check for the faulty links and eligible turns prior to connecting the input channel to the output channel. In Enhanced-MAFA, the potential output channels are selected in a way that the existence of at least one path from the next switch to the destination switch is guaranteed. For example, in Figure 4.23(b) if the current switch is the source switch, packets can be routed through the NN, NE, NW, WW, WN, WS, EE, EN, ES, SS, SE, or SW path without creating any deadlock. In Figure 4.23(c), input channel is E1 and the destination is in the northeast of the current switch. In this case, packets can be safely sent through the NE, NN, NW, WW, WN, WS, SS, or SW path. Figure 4.24 presents the choices of output channels allowed by Enhanced-MAFA. The main priorities are always checked before the other options.

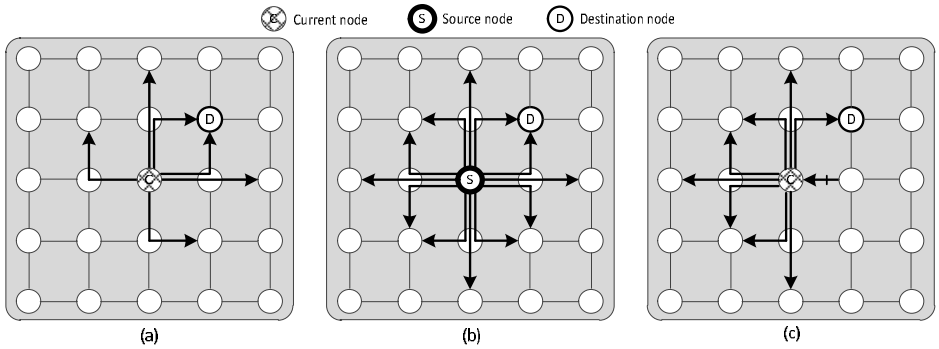

Figure 4.23: Examining all possible paths by Enhanced MAFA

```
ALGORITHM: Output choices offered by Enhanced-MAFA
                **----------------------------------------------**
Definitions:    inPort: input Port;  dLoc: destination Location
                **----------------------------------------------**
if (dLoc/={e} and inPort/={w2,n2}) or (dLoc={e} and inPort={l,w1,s1}) then
        select <= nn or ss;
if (dLoc/={ne} and inPort={l,w1,s1}) or (dLoc={ne} and inPort/={w2,n2}) then
        select <= ne;
if (dLoc/={e} and inPort/={w2,n2}) then
        select <= nw;
if (dLoc/={e} and inPort={l,w1,n1,s1}) or (dLoc={e} and inPort/={n2,s2}) then
        select <= ee;
if (inPort ={l,w1,n1,s1}) then
        select <= en,es;
if (dLoc/={e} and inPort/={w2}) then
        select <= ww,wn,ws;
if (dLoc/={e} and inPort/={w2,s2}) then
        select <= sw;
if (dLoc/={se} and inPort={l,w1,n1}) or (dLoc={se} and inPort/={w2,s2}) then
        select <= se;
```

Figure 4.24: Non-minimal choices offered by Enhanced-MAFA

4.2.3 Results and Discussion

To evaluate the efficiency of the proposed routing schemes, a NoC simulator is developed with VHDL to model all major components of the on-chip network. For all switches, the data width is set to 32 bits. The congestion threshold value is set to 5, meaning that a buffer is considered as a congested one when 5 out of 8 buffer slots are occupied. Moreover, the packet length is uniformly distributed between 5 and 10 flits. As a performance metric, we use latency defined as the number of cycles between the initiation of a packet issued by a Processing Element (PE) and the time when the packet is completely delivered to the destination PE. The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles.

For evaluating performance, MD, MAFA, and RAFT methods are compared together. As we discussed in section 4.1.3, RAFT is a traditional method designed for tolerating faulty links in the network [82]. MAFA and RAFT methods require two virtual channels along both dimensions while MD utilizes one and two virtual channels along the X and Y dimension. To have a fair comparison, an extra virtual channel is added to the MD approach which is utilized for improving performance.

4.2.3.1 Performance Analysis under Uniform Traffic Profile

In the uniform traffic profile, each processing element generates data packets and sends them to another processing element using a uniform distribution [49]. The mesh size is considered 8×8. In Figure 4.25, the average communication latencies of the MD and RAFT are measured for fault-free and a single faulty link cases. As observed from the results, in fault-free cases, RAFT works slightly better than MD. The reason is that RAFT is a fully adaptive routing algorithm while in MD the adaptivity is limited when packets get close to their destination switch. In one-faulty cases, MD maintains performance at the similar level while the performance of RAFT drops significantly with a single faulty link in the network. This is due to the fact that MD can route packets through minimal paths while in RAFT, packets may take longer paths when facing a faulty link. MAFA is able to tolerate two faulty links, so that we compare it with RAFT under fault-free and two faulty cases. As can be seen in Figure 4.26, MAFA maintains performance under two faulty links.

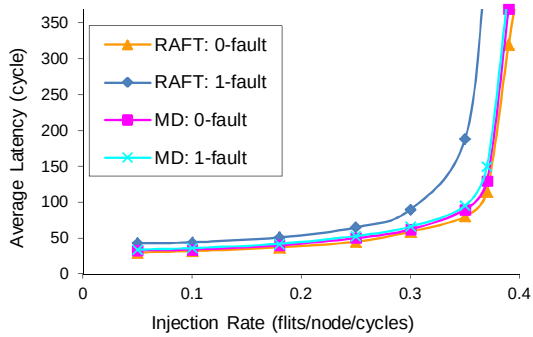

Figure 4.25: Performance analysis of MD and RAFT in an 8×8 mesh network under uniform traffic profile

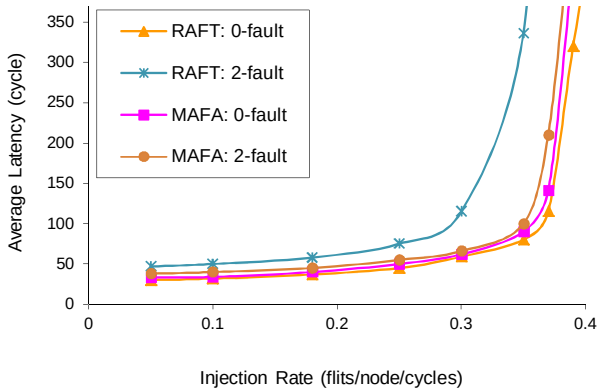

Figure 4.26: Performance analysis of MAFA and RAFT in an 8×8 mesh network under uniform traffic profile

4.2.3.2 Performance Analysis under Hotspot Traffic Profile

Under the hotspot traffic pattern, one or more switches are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In simulations, given a hotspot percentage of *H*, a newly generated packet is directed to each hotspot switch with an additional *H* percent probability. We simulate the hotspot traffic with a single hotspot switch at (4,4) in an 8×8 mesh network. The performance of MD and RAFT is measured for fault-free and one-faulty link cases while MAFA and RAFT are compared with each other under fault-free and two faulty links. The performance analysis with *H=10%* are illustrated in Figure 4.27 and Figure 4.28. In all the faulty configurations, MD and MAFA outperform the RAFT approach.

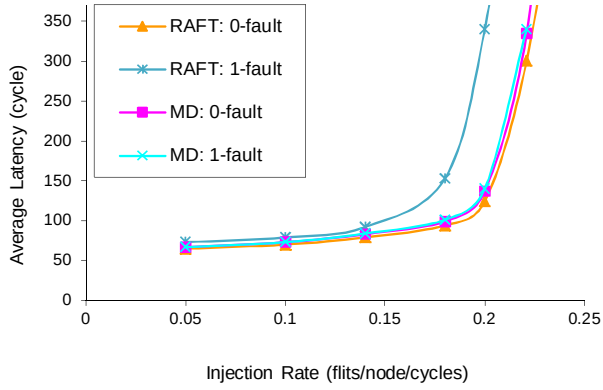

Figure 4.27: Performance analysis of MD and RAFT in an 8×8 mesh network under hotspot traffic profile

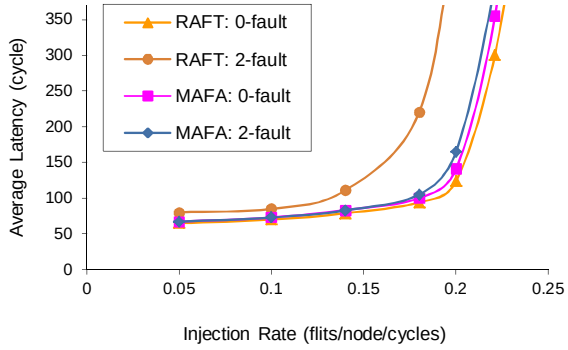

Figure 4.28: Performance analysis of MAFA and RAFT in an 8×8 mesh network under hotspot traffic profile

4.2.3.3 Reliability Evaluation under Uniform Traffic Profile

It is worth mentioning that, MAFA and Enhanced-MAFA perform similarly in the term of performance. However, Enhanced-MAFA offers higher reliability and can tolerate more number of faults than MAFA. In the reliability evaluation, we take MD, Enhanced-MAFA, and RAFT into consideration when the number of faulty links increases from 1 to 6. All faulty links are selected using a random function. The results are obtained using 10,000 iterations in a 6×6 mesh network when the traffic is uniform random. A network is reliable if all the injected packets reach their destinations. As shown in Figure 4.29, Enhanced-MAFA can tolerate up to 6 faulty links by more than 97% probability.

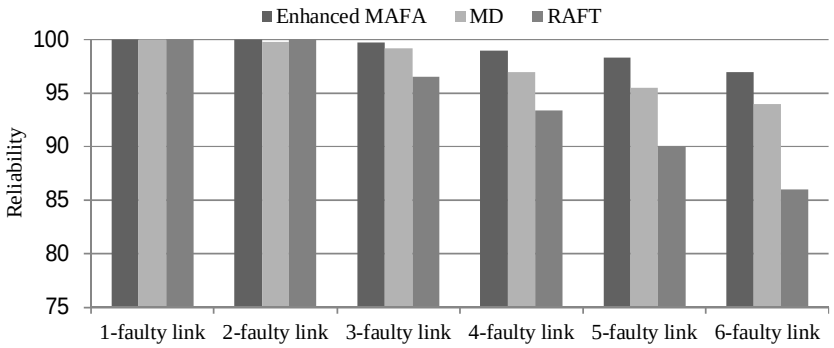

Figure 4.29: Reliability evaluation of Enhanced-MAFA in a 6×6 mesh network under uniform traffic profile

4.2.3.4 Hardware Analysis

To assess the area overhead and power consumption, the whole platform of each method is synthesized by Synopsys Design Compiler. We measured the area overhead and power consumption of the MD, Enhanced-MAFA, and RAFT methods. MD utilizes one and two virtual channels along the X and Y dimensions while Enhanced-MAFA and RAFT use two

virtual channels in both dimensions. Power consumption is measured under a one-faulty link case. Each scheme includes network interfaces, switches, and communication channels. All methods are synthesized using the TSMC 65nm technology at the operating frequency of 500MHz and supply voltage of 1V. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation is calculated using Synopsys PrimePower in an 8×8 mesh network.

The layout area and power consumption of each platform are shown in Table 4.2. As indicated in the table, Enhanced-MAFA and RAFT have larger area overheads than MD due to using an extra virtual channel. Enhanced-MAFA consumes less power than RAFT. It is because of avoiding hotspots in the network in the presence of faults. Moreover, Enhanced-MAFA uses a simpler routing unit than RAFT.

Table 4.2: Hardware implementation details

Network platforms	Area (mm^2)	Power (mw)
MD	2.571	1.563
Enhanced_MAFA	2.903	1.657
RAFT	2.924	1.791

4.3 The Proposed Approaches for Tolerating Faulty Switches

The two proposed methods MD and MAFA are able to tolerate faulty links in the network. MD is able to tolerate a small number of faulty links while performing similar to a fully adaptive routing algorithm. MAFA tolerates a large number of faulty links with limited adaptivity. Faults may occur at the switch level in which MD and MAFA are not able to address them in their basic forms. In this section, we present two fault-tolerant approaches to tolerate faulty switches. These methods are called High Performance Fault-tolerant Routing (HiPFaR) [87] and Minimal-path Connection-retaining Fault-tolerant approach (MiCoF) [88]. Similar to MD, HiPFaR uses one and two virtual channels along the X and Y dimensions and is able to tolerate any single faulty switch in the network using only the shortest path as possible. A non-minimal route is necessitated when the source and destination switches are located in the same row or column with a faulty switch between them. In MiCoF, the same numbers of virtual channels are used. However, it is able to address a wide range of faults by using only the shortest paths even if the source is in the same row or column as the destination switch. This is achieved by a simple modification in the switch architecture.

4.3.1 Any Single Faulty Switch (HiPFaR)

As was already mentioned, there are many fault-tolerant approaches presented both in the off-chip and on-chip networks. Some approaches disable healthy components in order to form a specific shape while others do not. Regardless of all varieties, there has always been a common assumption among them. Most of all traditional fault-tolerant methods are based on rerouting packets around faults. These approaches affect performance significantly not only by taking longer paths but also by creating hotspot around a fault.

The goal of HiPFaR is to tolerate faulty switches without affecting performance or disabling healthy components. HiPFaR uses the minimum number of virtual channels to provide adaptiveness. Moreover, it requires only the fault statuses of the neighboring switches which are the minimum knowledge for making a routing decision in a fault-tolerant approach.

4.3.1.1 Fault Distribution Mechanism

Some proposals require the knowledge about the fault statuses of all switches or links in the network. To collect this knowledge, online or offline techniques are employed. In some other proposals, for making routing decisions, the fault statuses of direct and indirect neighboring switches are needed. In few approaches, the routing decision is based on the minimum knowledge about the faults in the network (i.e. four neighboring switches). As the fault might occur on additional resources, the simpler approach is always preferred.

In HiPFaR, it is enough that each switch is informed about the fault statues of its neighboring switches (Figure 4.30). Based on this knowledge, HiPFaR is able to deliver packets through the available shortest paths.

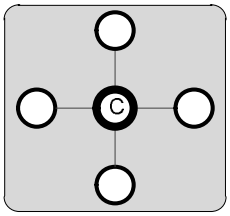

Figure 4.30: The statues of four neighboring switches are required by the HiPFaR routing algorithm

4.3.1.2 Turn Model in HiPFaR

HiPFar uses the same number of virtual channels as the MD routing algorithm which is one and two virtual channels along the X and Y dimensions, respectively. HiPFaR also utilizes the same turn model as MD, shown in Figure 4.31. This turn model is proved to be deadlock-free in Section 4.2.1.

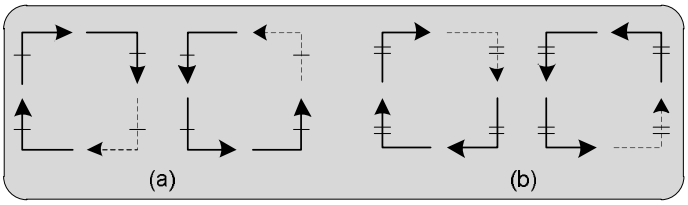

Figure 4.31: Permitted and prohibited turns of MD similar to Mad-y [10]

4.3.1.3 Tolerating Faulty Switches

Let us assume that there is a faulty switch in the network. The fault is bypassed using non-minimal paths when the source and destination switches are located in the same row or column. In other cases, only the shortest paths are taken from the source to the destination switch. In Figure 4.32, for the ease of understanding, we investigate HiPFaR for a northeast packet. However, as it will be shown, the HiPFaR routing algorithm is general and can be applied to the northwest, southeast, and southwest packets without any implementation differences.

As illustrated in Figure 4.32(a) and Figure 4.32(b), the packet is one hop away from the destination switch in both the X and Y dimensions (*X-dir=1 and Y-dir=1*). By default, the packet is sent to the Y direction (Figure 4.32(a)). However, when the north neighboring switch is faulty, the packet is delivered to the X direction (Figure 4.32(b)). Figure 4.33(a) shows the possible cases where the distances along both the X and Y directions are one hop. As illustrated in this figure, in positions 1 and 2, the packet is delivered to the Y dimension since the north neighboring switch is non-faulty. In positions 3, the packet is sent to the X dimension as the north neighboring switch is faulty. Thereby, in all three positions, the packet can reach the destination using the shortest paths.

In Figure 4.32(c) and Figure 4.32(d), the distance is one and two (or greater than two) hops along the X and Y directions (*X-dir=1 and Y-dir>=2*), respectively. The rule is similar to the previous case such that the packet is sent to the Y direction unless the north neighboring switch is faulty (i.e. in this case, the packet is delivered to the X direction). According this rule, in positions 1, 2, and 3 of Figure 4.33(b), the packet is sent to the Y direction. In the next hop, the packet stands in one of the positions of Figure 4.33(a) (It is already shown that packets can reach the destination switch using the shortest paths in the presence of fault). In position 4, the north neighboring switch is faulty, and thus the packet is sent to the X direction. This packet reaches the destination switch using the shortest path as the faulty switch is already bypassed.

Figure 4.32(e) and Figure 4.32(f) indicate the cases where the distances are two hops (or greater than two) and one hop along the X and Y dimensions (*X-dir>=2 and Y-dir=1*). The rule is as simple as avoiding to send the packet to the Y direction when the east neighboring switch is non-faulty. Figure 4.33(c) shows the different positions of the current, destination and a faulty switch. In positions 1, 2, and 3, the packet is sent to the X direction as the east neighboring switch is non-faulty. In the next hop, the packet stands in one of the positions of Figure 4.33(a). If the east neighboring switch is faulty (position 4), the packet is delivered to the destination through the north neighboring switch.

When the distances along both directions are two or greater than two hops (*X-dir>=2 and Y-dir>=2*), the packet is sent to a non-faulty neighboring switch (Figure 4.33(d)). By routing under this policy, the packet reaches one of the positions of Figure 4.33(b) or Figure 4.33(c). In sum, in all faulty cases, the packet is routed to the destination switch through the shortest paths. In a case where both neighboring switches are non-faulty, one direction is chosen based on the congestion information. Therefore, HiPFaR is fully adaptive as long as the remaining distance along both directions is equal or greater than two

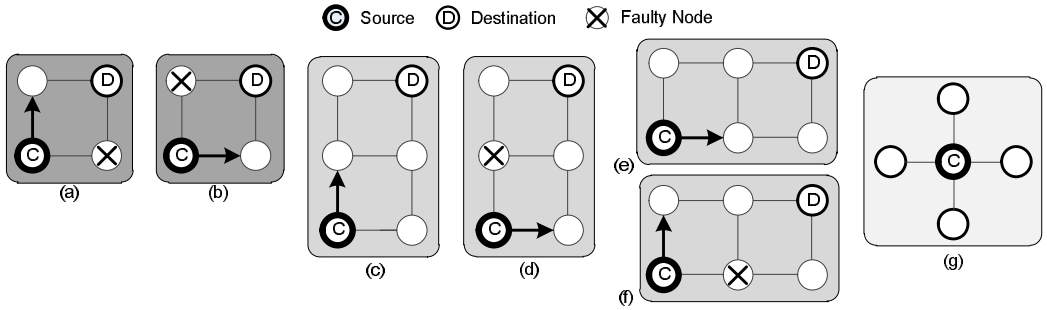

Figure 4.32: The basic rules for selecting among the neighboring switches when a packet gets close to the destination switch

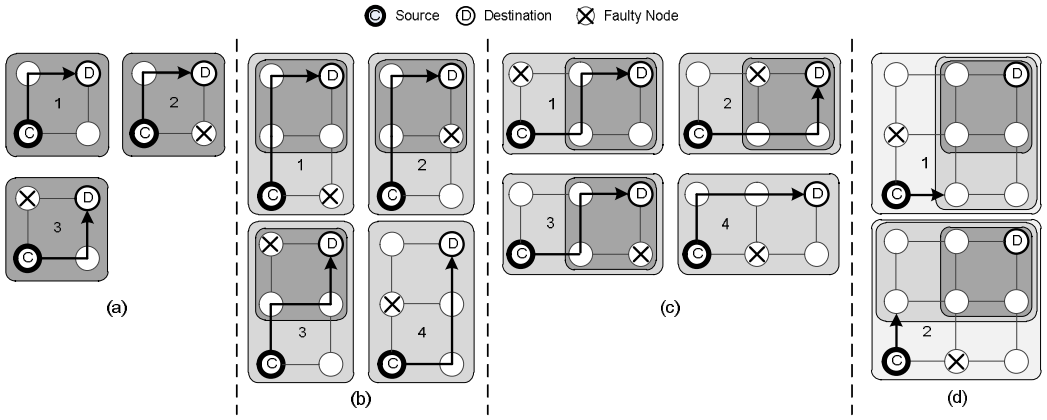

Figure 4.33: Different positions of current, destination and a faulty switch

hops. According to these rules, each switch needs to know the fault information of only four neighboring switches shown in Figure 4.32(g).

HiPFaR is a fully adaptive routing algorithm supporting all the required turns to route packets through the shortest paths. The northeast, northwest, southeast, and southwest packets do not take any non-minimal routes for tolerating faults, so that, all the required turns are supported by HiPFaR. East-, west-, north-, and south-ward packets have to be rerouted around the fault when facing to it. As shown in Figure 4.34(a), eastward packets use the E-N1, N1-E, E-S1, E-S1, and S1-E turns to bypass the faults in which all of them are in the set of allowable turns. Similarly, all the required turns by the westward packets are allowable (i.e. W-N2, N2-W, W-S2, W-S2, and S2-W). As illustrated in Figure 4.35, normally, the northward and southward packets use the allowable turns as N2-W, W-N2, N2-E, S2-W, W-S2, S2-E. However, when the source and destination switches are located in the left borderline, the required turns are N2-E, E-N2, N2-W, S2-E, E-S2, and S2-W. Among them, E-N2 and E-S2 are unallowable according to the turn model of HiPFaR, but a cycle cannot be formed in borderline cases and these unallowable turns can be safely taken.

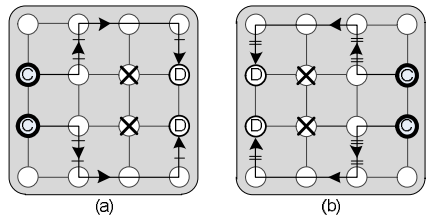

Figure 4.34: Tolerating a faulty switch by (a) eastward packets (b) westward packets

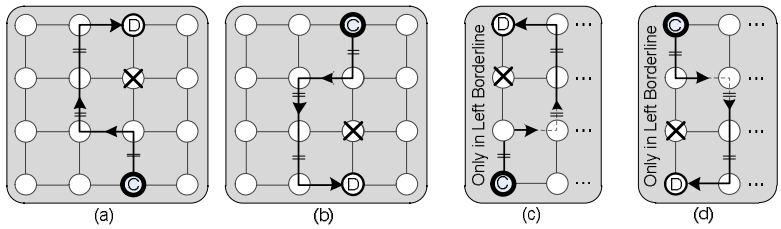

Figure 4.35: Tolerating a faulty switch for northward and southward packets

4.3.1.4 HiPFaR Routing Algorithm

The HiPFaR routing algorithm is illustrated in Figure 4.36. HiPFaR can be explained by two parts: when the destination is in the east, west, north and south directions of the source switch and when the destination is in the northeast, northwest, southeast, and southwest directions of the source switch. In this algorithm, the position is specified based on the source and destination position. The *X-dir* and *Y-dir* parameters determine the minimal directions toward the destination switch regarding the current switch. The *delta-X* and *delta-Y* parameters maintain the remaining distances from the current to the destination switch along the X and Y dimensions. Finally, in the algorithm, *vc* indicates the appropriate virtual channel. According to HiPFaR, the first virtual channel of the Y dimension is used when the destination is toward the east, northeast, and southeast directions of the source switch. Similarly, the second virtual channel is utilized for west-, northwest-, and southwest-ward packets. The north- and south-ward packets utilize the second virtual channel.

In Figure 4.34 when the packets are either east- or west-ward and *delta-Y* is equal to zero, there are two options to bypass the fault: making a turn to the north or to the south direction. However, when the packet is already rerouted to the north or south direction, it has to be routed along the *X-dir* until *delta-X* becomes zero. At this point, the packet can be delivered to the destination switch by turning to the Y direction. As illustrated in Figure 4.35, the north- and south-ward packets are routed in the second virtual channel as long as they do not face with a fault. To bypass the fault, normally the turn to the west direction is done. In a case where the faulty switch is located in the left borderline, the turn to the east direction should be done. Packets are routed along this direction until *delta-Y* becomes zero. At this point, a turn to the east or west direction is needed to deliver the packet to the destination switch.

A minimal routing algorithm is used for northeast-, northwest-, southeast-, and southwest-ward packets. Packets can be adaptively routed within the network when they are two hops away along each direction from the destination switch. At each intermediate switch, the congestion information is used to send a packet to one of the non-faulty neighboring switches which are located in the minimal path. HiPFaR is designed to tolerate any single faulty switch in the network using only the shortest paths. However, there are some cases in which non-minimal paths are required in order to bypass multiple faulty switches. Non-minimal routes can be supported by similar approaches as Enhanced-MAFA.

```
ALGORITHM: HiPFaR routing algorithm
                    **-------------------------------------------------**
Definitions:  Xs,Ys,Xd,Yd,Xc,Yc: X and Y coordinates of the source, destination and current switches
              vc: virtual channel;
                    **-------------------------------------------------**
position <= {NE,NW,SE,SW,E,W,N, or S} according to the source and destination positions

x_dir <= E when Xd > Xc else W;
y_dir <= N when Yd > Yc else S;
vc <= vc1 when position={E,NE,SE} else vc2 when position={W,N,S,NW,SW};
delta_x <= Xd - Xc when Xd>Xc else Xc-Xd;
delta_y <= Yd - Yc when Yd>Yc else Yc-Yd;

if position={NE, NW, SE, or SW} then
        if (delta_x>=1 and delta_y=0) then select <= x_dir;
        elsif (delta_x=0 and delta_y>=1) then select <= y_dir(vc);
        elsif (delta_x=1 and delta_y>=1) then
                if neighbor(y_dir)=faulty then select <= x_dir; else select <= y_dir(vc); end if;
        elsif (delta_x>1 and delta_y=1) then
                if neighbor(x_dir)=faulty then select <= y_dir(vc); else select <= x_dir; end if;
        else
                if neighbor(x_dir)=faulty then select <= y_dir(vc);
                elsif neighbor(y_dir)=faulty then select <= x_dir;
                else select <= x_dir or y_dir(vc); end if;
        end if;
elsif position={E or W} then
        if delta_y = 0 then
                if neighbor(x_dir)=faulty then select <= N(vc) or S(vc); else  select <= x_dir; end if;
        else
                if neighbor(y_dir)=dest then select <= y_dir(vc); else  select <= x_dir; end if;
        end if;
elsif position={N or S} then
        if delta_x = 0 then
                if neighbor(y_dir)=faulty then
                        if Xc/=0 then select <= west; else select <= east; end if;
                else select <= y_dir(vc); end if;
        else
                if inPort/= {E,W} and neighbor(x_dir)=non-faulty then select <= x_dir;
                else select <= y_dir(vc); end if;
        end if;
end if;
```

Figure 4.36: HiPFaR routing algorithm

4.3.2 Multiple Faulty Switches (MiCoF)

HiPFaR is able to tolerate faulty switches using the shortest paths as long as the source and destination switches are not located in the same row or column. It has many advantages over traditional methods as: 1- it maintains the performance level of NoC by choosing only the shortest paths for each pair of source and destination switches in the presence of faults. 2- the routing unit only requires the fault statuses of the adjacent switches, that is the minimum knowledge needed by a fault-tolerant routing algorithm. 3- the algorithm is very simple such that it can be implemented in few lines of code. This small piece of code covers all positions of faulty switches. 4- it requires only one and two virtual channels along the X and Y dimensions, which is the minimum number of virtual channels to design a fully adaptive routing algorithm. Moreover, it does not require any routing table.

In this section, we present a method called MiCoF (Minimal-path Connection-retaining Fault-tolerant approach). The number of virtual channels, the fault distribution mechanism and the turn model are similar to the HiPFaR approach. The main goal of MiCoF is to use only the shortest paths in the presence of faulty switches. In this approach, packets are never rerouted around the faulty switches. To keep the connectivity and avoid rerouting packets, the switch architecture is slightly modified such that when a switch becomes faulty, the involved links will be connected in appropriate directions. In other words, a faulty switch can be seen as a wire, connecting the surviving switches to each other. Based on this architecture, a routing algorithm based on the shortest paths is presented. This architecture along with the shortest-path routing algorithm provides a highly resilient network. The interesting point is that the average packet latencies are decreased as the number of faults increases in the network.

MiCoF has the whole characteristics of HiPFaR while it offers more benefits as: 1- packets take the shortest paths even if the source and destination switches are located in the same row or column with a faulty switch between them. 2- when a switch becomes faulty, its involved links can still be utilized. 3- it does not have any exceptional rules for borderline switches. 3- it is highly reliable such that on average 99.5% percentages of packets successfully reach their destinations when there are six faulty switches in an 8×8 mesh network. By another metric of reliability, when six faults occur in the network, with the probability of more than 50%, the network functions normally without any packet loss. Reachability is another highlighting point of this method as a switch with all neighbors faulty is still reachable.

4.3.2.1 MiCoF Architecture

A faulty switch has a severe impact on the performance of NoCs. When a switch becomes faulty, not only the connected core cannot send or receive packets, but also packets from the other cores cannot be transmitted through this switch. In other words, the connected core and links of a faulty switch are also tagged as faulty and become unusable. The core cannot start working until the fault is recovered. However, by using the MiCoF architecture, we show that the links can still be used to retain performance.

Figure 4.37 shows the switch architecture based on MiCoF. Normal switch architecture includes input buffers, a routing unit, a virtual channel allocator, a switch allocator and a crossbar switch. In our modified architecture, in a case of faults, the east input channel is directly connected to the west output channel while the west input channel is connected to the east output channel. Similarly, the packets coming from the north and south input channels are directly connected to the south and north output channels, respectively. So, no processing takes place in the switch and packets are not stored in input buffers. In the MiCoF architecture, the whole faulty switch acts as a wire, connecting the input channels to output channels in specific directions. Compared with normal switch architecture, MiCoF needs a few multiplexers and de-multiplexers at input and output ports plus a small wiring overhead.

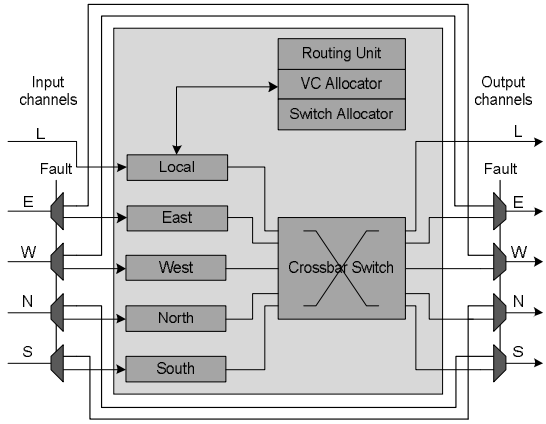

Figure 4.37: Switch architecture using MiCoF

Figure 4.38(a) shows a 4×4 mesh topology with five faulty switches. A faulty switch itself and the core connected to it are disconnected from the network while the links are used to connect the neighboring switches in appropriate directions. Using the MiCoF architecture, the resulted network is illustrated in Figure 4.38(b).

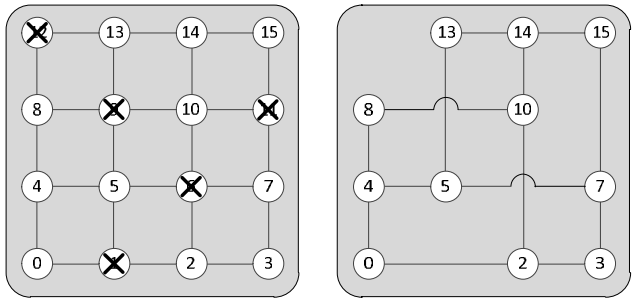

Figure 4.38: Five fautly switches in a 4×4 mesh topology (b) the resulted network using MiCoF

4.3.2.2 Tolerating Single Faulty Switches by MiCoF

We start introducing the MiCoF approach in a case when there is a single faulty switch in the network. To do this, we follow the path of a northeast packet as shown in Figure 4.39. When the destination is in the east or north position of the current switch (Figure 4.39(a)), the packet can bypass the faulty switch and reach the destination without taking any non-minimal path. This is possible as the faulty switch can be considered as a wire connecting the links in the horizontal and orthogonal directions.

In Figure 4.39(b), the packet is one hop away from the destination switch in both the X and Y dimensions (*X-dir=1 and Y-dir=1*). By default, the packet is sent to the Y direction (patterns 1 and 2). However, when the neighboring switch in the Y direction is faulty, the packet is delivered to the X direction instead (pattern 3).

Figure 4.39(c) indicates the cases where the distances are two (or greater than two) hops and one hop along the X and Y dimensions (*X-dir>=2 and Y-dir=1*), respectively. Similar to the previous case, the packet is sent to the Y direction (patterns 1, 2, and 3) unless the neighboring switch in this direction is faulty. If the north neighboring switch is faulty, the packet is delivered to the X direction (pattern 4). In the next hop, the packet stands in one of the positions of Figure 4.39(a) or Figure 4.39(b). Therefore, in all positions of a faulty switch, the packet could reach the destination by using the shortest paths.

In Figure 4.39(d), the distance is one and two (or greater than two) hops along the X and Y directions (*X-dir=1 and Y-dir>=2*), respectively. The rule is as simple as avoiding to send the packet to the Y direction when the neighboring switch in the X direction is non-faulty. Using this rule, all positions of faults can be covered by patterns 1, 2, 3, and 4. In the next hop, the packet stands in one of the positions of Figure 4.39(a) or Figure 4.39(b). When the distances along both directions are two or greater than two hops (*X-dir>=2 and Y-dir>=2*), the packet should be sent to a non-faulty neighboring switch (Figure 4.39(e)). By routing the packet with this policy, the packet reaches one of the positions of Figure 4.39(c) or Figure 4.39(d). In a case where both neighboring switches are healthy, the

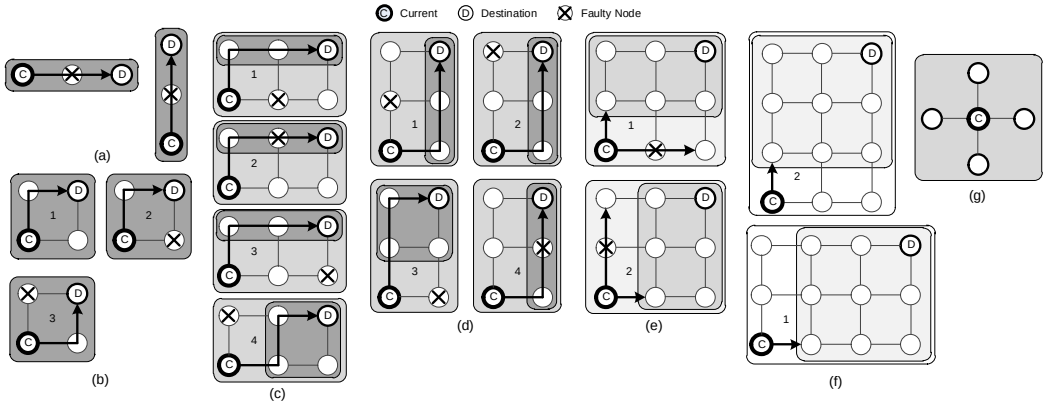

Figure 4.39: Tolerating any single faulty switch using only the shortest paths

packet is sent through the less congested direction. Using MiCoF, packets can be routed adaptively in the network as long as the remaining distance along both directions is equal or greater than two hops.

To sum up, it is guaranteed that any single faulty switch is covered by MiCoF taking only the shortest paths between each pair of source and destination switches. In addition, packets can be adaptively routed within the network when they are not close to the destination switch. However, for better reliability, the adaptiveness of MiCoF can be limited. The limitation is applied to a situation when the distances along both directions are two or greater than two hops. In this case, the packet is sent to a greater-distance dimension. If the neighboring switch in the greater-distance dimension is faulty, the smaller-distance direction is selected (Figure 4.39(f)). In this way, packets maintain the adaptivity, and thus they have an alternative choice when facing a fault in one direction. Finally, the packet can be sent through either direction when the distances along both directions are equal. Obviously, this adaptivity limitation does not affect the behavior of the algorithm in addressing any single faulty switch without rerouting packets.

4.3.2.3 Tolerating Multiple Faulty Switches by MiCoF

MiCoF is able to tolerate multiple faulty switches with high reliability. In this section, we investigate MiCoF specifically when it is used to tolerate two and three faulty switches in the network.

A. Tolerating Two Faulty Switches

The rules of the MiCoF routing algorithm remain the same when tolerating more number of faults. Now, we investigate how tolerating two faulty switches can be addressed in the network using the same rules.

As shown in Figure 4.40(a), when the current and destination switches are located in the same row or column, the packet can pass through the faulty switches, regardless of the number of faults in the path. Figure 4.40(c) indicates all the positions of two faulty switches when the distances are two and one hops along the X and Y dimensions. By default, the packet is sent to the Y direction. If the neighboring switch in the Y direction is faulty, the packet is sent through the X direction. In patterns 1, 2, and 3 of Figure 4.40(c), the packet is sent to the neighboring switch in the Y direction as it is non-faulty. From the next hop, the packet faces the similar conditions as one faulty switch in the network which is already discussed. Patterns 4, 5, and 6 cover the cases when the neighboring switch in the Y direction is faulty, and thus the packet is sent to the X direction. In the remaining path, another fault can be easily tolerated. A similar approach is applied when the distance is one and two hops along the X and Y dimensions (Figure 4.40(d)).

Figure 4.40(e) shows the cases in which the distance is two hops along both directions. If the neighboring switch in one of the directions is faulty, the packet is sent through the other direction (pattern 1 and 2). From the next hop to the destination switch, there might be at most one faulty switch in the path that can be addressed according to MiCoF. If both neighboring switches are non-faulty (pattern 3), in the remaining path from the next hop to the destination switch, the packet might face to at most two faults. As indicated in

Figure 4.40(c) or Figure 4.40(d), a network is reliable against two faults by MiCoF. If both neighboring switches are faulty, the packet can be sent through either direction. This packet will not face any fault in the remaining path as both faults are already bypassed.

So far, all two faulty switches are tolerated by MiCoF using only the shortest paths. There is only one position in which two faulty switches cannot be addressed using the shortest paths. This is the case when the distance from the source to the destination switch is one hop along both dimensions while the neighboring switches in both directions are faulty (Figure 4.40(b)). These positions of faults are called diagonal positions. The source switch still can send and receive packets to/from every other switch in the network except the destination switch. If the source switch is farther away from the destination, the packet never stands in this unsupported position as the packet already chooses other routes prior to reaching this position (e.g. similar to the pattern 2 of Figure 4.40(c) and Figure 4.40(d)). Therefore, packets from this specific source position cannot reach to this specific destination position (or vice versa) if both neighboring switches are faulty. All the other packets can be normally routed in the network.

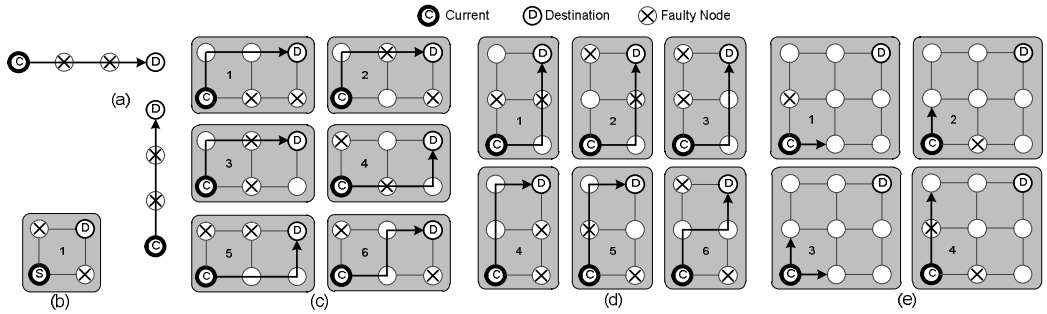

Figure 4.40: Tolerating two faulty switches by the MiCoF approach

In our measurements, we use two reliability metrics as reliability1 and reliability2. Reliability1 shows the probability that the network can successfully deliver any packet under the existence of faults. Reliability2 is the probability that a packet can be successfully delivered under faults.

- **Reliability by the first definition (Reliability1):**

According to MiCoF, if two faults are located in diagonal positions, the network may fail. At first, we calculate the number of total combinations of two faulty switches in the network. Then, we measure the number of combinations in which two faults occur in diagonal positions. By dividing these two numbers, the reliability value is obtained. The number of different combinations of two faulty switches in an $n \times n$ mesh network can be measured by:

$$N_{all\_combinations} = \binom{n^2}{2} = \frac{n^2(n^2-1)}{2}$$

Figure 4.41 shows all combinations in which two faulty switches are located in diagonal positions. By extending the idea to an $n \times n$ mesh network, the number of diagonal combinations can be calculated by:

$$N_{diagonal\_combinations} = 2(n-1)^2$$

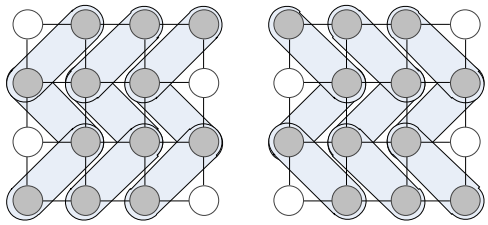

Figure 4.41: A couple indicates a diagonal position
(i.e. nine diagonal positions in each figure)

Finally, Reliability1 can be calculated by:

$$R1 = 1 - \frac{N_{diagonal\_combinations}}{N_{all\_combinations}} = 1 - 4\frac{(n-1)^2}{n^2(n^2-1)}$$

According to this formula, for example in an 8×8 mesh network, with the probability of 95.2%, two faults will not be located in diagonal positions, and thus the network functions normally without dropping any packet.

- **Reliability by the second metric (Reliability2)**

The second metric is mostly used in literature to report the reliability value. Let us assume that the network is examined under all combinations of two faulty switches. Thereby, the number of examinations is equal to the combinations of two faulty switches ($N_{all\text{-}combinations}$). Per examination, each non-faulty switch delivers one packet to every other non-faulty switch in the network (i.e. total of n^2-3 packets, except itself and two faulty switches). As faulty switches do not send or receive any packets (i.e. total of n^2-2 switches are able to deliver packets), the total number of delivered packets per combination is:

$$N_{delivered\_per\_combination} = (n^2-2)(n^2-3)$$

Therefore, the total number of delivered packets in the whole examinations is:

$$Total = N_{delivered\_per\_combination} \times N_{all\_combinations}$$

On the other hand, per diagonal position, two packets must be dropped (those from the source to the destination switch or vice versa), so that the total number of defeated packets is calculated by:

$$Defeated = 2 \times N_{diagonal\_combinations} = 4(n-1)^2$$

Therefore, Reliability2 can be measured by:

$$R2 = 1 - \frac{Defeated}{Total}$$

According to this formula, in an 8×8 mesh network, 99.998% of packets reach their destinations considering all combinations of two faulty switches.

B. Tolerating Three Faulty Switches

Now we take a quick look at three faulty switches in the network. If three faults are distributed over the network, they are easily tolerated according to Figure 4.39 and Figure 4.40. However, when faults are close to each other, the situations depicted in Figure 4.42 are obtained. If the locations of three faults are similar to the patterns 1 or 2 in either Figure 4.42(a) or Figure 4.42(b), then faults are tolerated by the MiCoF routing algorithm. In the patterns 3 and 4, a few packets cannot reach the destination. Even under these positions, the rest of the packets reach their destinations through the shortest paths.

The focus of MiCoF is to tolerate faults by only using the shortest paths without any performance loss. However, non-minimal paths or virtual channels can be used to support the remaining cases.

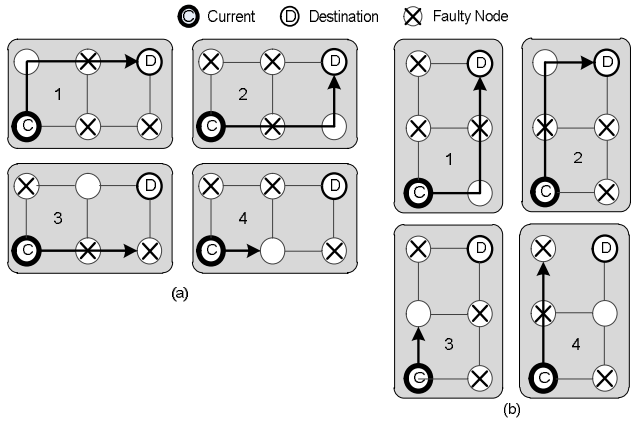

Figure 4.42: Three faulty switches in the network, which are located close to each other

4.3.2.4 MiCoF Routing Algorithm

As it is already mentioned, fault-tolerant routing algorithms are usually very complex. In contrast with them, the MiCoF algorithm is very simple with a negligible area overhead. The general MiCoF routing algorithm is shown in Figure 4.43. This algorithm only requires the fault information of four neighboring switches (i.e. it is normally provided for every fault-tolerant method). Thereby, this algorithm does not impose any area overhead due to collecting the fault information throughout the network. Taking into account that faults might occur on additional resources, the less amount of requirements leads to a more reliable method.

```
ALGORITHM: MiCoF routing algorithm
                    **-------------------------------------------------**
Definitions:    Xd,Yd: X and Y coordinates of the destination switch
                Xc,Yc: X and Y coordinates of the current switch
                N,S,E,W: North, South, East, West
                    **-------------------------------------------------**
x_dir <= E when Xd>Xc else W;
y_dir <= N when Yd>Yc else S;

delta_x <= (Xd-Xc) when Xd>Xc else (Xc-Xd);
delta_y <= (Yd-Yc) when Yd>Yc else (Yc-Yd);

if (delta_x>=1 and delta_y=0) then select <= x_dir;
elsif (delta_x=0 and delta_y>=1) then select <= y_dir;
elsif (delta_x>1 and delta_y=1) then
        if neighbor(y_dir)=healthy then select <= y_dir; else select <= x_dir; end if;
elsif (delta_x=1 and delta_y>=1) then
        if neighbor(x_dir)=healthy then select <= x_dir; else select <= y_dir; end if;
else
        if (neighbor(x_dir)=faulty and neighbour(y_dir)=faulty) then
                if (delta_x>delta_y) then select <= x_dir; else      select <= y_dir; end if;
        elsif (neighbor(x_dir)=healthy and neighbor(y_dir)=healthy) then
                if (delta_x>delta_y) then select <= x_dir;
                elsif (delta_x<delta_y) then select <= y_dir;
                else select <= x_dir or y_dir;
                end if;
        elsif neighbor(y_dir)=healthy then select <= y_dir;
        elsif neighbor(x_dir)=healthy then select <= x_dir;
        else select <= x_dir or y_dir;
        end if;
end if;
```

Figure 4.43: MiCoF routing algorithm

4.3.3 Results and Discussion

To evaluate the efficiency of the proposed approaches, a NoC simulator is developed with VHDL to model all major components of the on-chip network. For all switches, the data width is set to 32 bits. Each input buffer can accommodate 8 flits in each virtual channel. Moreover, the packet length is uniformly distributed between 5 and 10 flits. As a performance metric, we use latency defined as the number of cycles between the initiation of a packet issued by a Processing Element (PE) and the time when the packet is completely delivered to the destination PE. The request rate is defined as the ratio of the successful packet injections into the network over the total number of injection attempts. The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles.

Our baseline method is ReRS [79] which is described in Section 4.1.2. This method does not require any virtual channel and it is able to tolerate any single faulty switches. MiCoF and HiPFaR require one and two virtual channels along the X and Y dimensions. To have a fair comparison, we use two virtual channels for each method. The extra virtual channels are used for the performance purposes.

4.3.3.1 Performance Analysis under Uniform Traffic Profile

The performance analysis under uniform random traffic is shown in Figure 4.44 and Figure 4.45. The average communication latency of HiPFaR and ReRS are measured under fault-free and one-faulty switch cases. The average communication latency of MiCoF is obtained under fault-free and six-faulty switch cases. In the uniform traffic profile and in the fault-free network, the ReRS method performs the best as the underlying routing algorithm is a dimension-order routing. According to the simulation results, shown in both figures, the performance of the ReRS method is degraded significantly when a single fault occurs in the network. HiPFaR has a negligible performance loss under single faulty cases.

We increase the number of faulty switches up to six faults and measure the performance of MiCoF. Surprisingly, performance gradually starts increasing under the same traffic load. This is due to the fact that the routing does not take place in faulty switches and the total number of hops is decreased. For clarity, the performance curves from two- to five-faulty switches are omitted, but they are distributed between the curves of one- and six-faulty switches. This improvement is from the communication point of view while the whole system performance will be obviously decreased by occurring faults in the network.

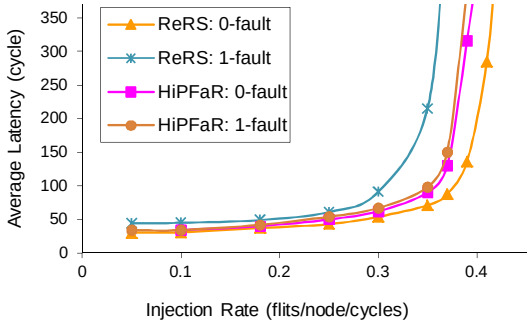

Figure 4.44: Performance analysis of HiPFaR and ReRS in an 8×8 mesh network under uniform traffic profile

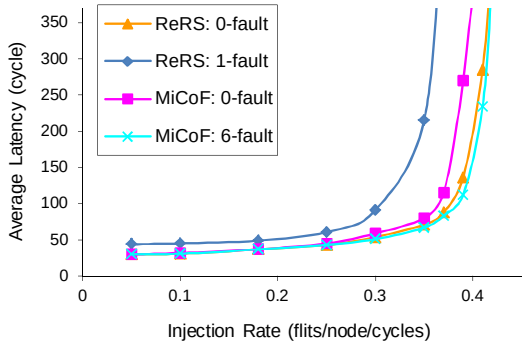

Figure 4.45: Performance analysis of MiCoF and ReRS in an 8×8 mesh network under uniform traffic profile

4.3.3.2 Performance Analysis under Hotspot Traffic Profile

Under the hotspot traffic pattern, one or more switches are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In simulations, given a hotspot percentage of *H*, a newly generated packet is directed to each hotspot switch with an additional *H* percent probability. We simulate the hotspot traffic with a single hotspot switch at (4,4) in an 8×8 mesh network. Figure 4.46 and Figure 4.47 show the performance of HiPFaR, MiCoF, and ReRS under different numbers of faulty switches and *H=10%*. In fault-free cases, both HiPFaR and MiCoF lead to better performance than ReRS as they are fully adaptive routing algorithms. Under the existence of a single fault in the network, the performance of ReRS is dramatically decreased. On the other hand, the performance of the network using MiCoF is improved when there are six faults in the network.

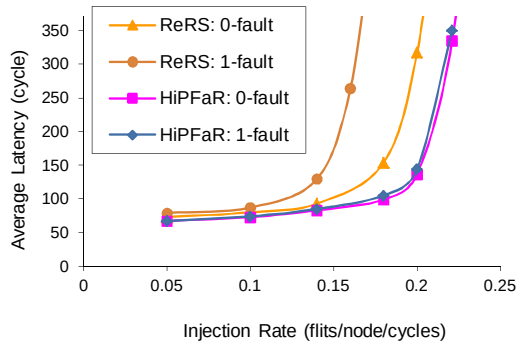

Figure 4.46: Performance analysis of HiPFaR and ReRS in an 8×8 mesh network under hotspot traffic profile

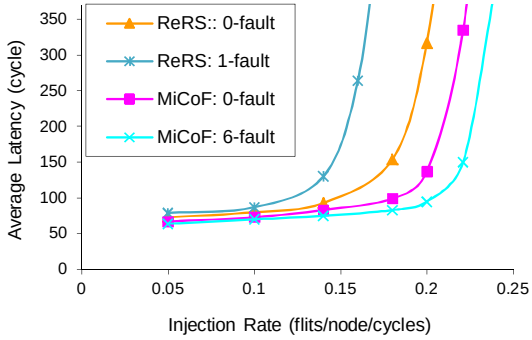

Figure 4.47: Performance analysis of MiCoF and ReRS in an 8×8 mesh network under hotspot traffic profile

4.3.3.3 Reliability Evaluation under Uniform Traffic Profile

In the uniform traffic profile, each processing element (PE) generates data packets and sends them to another PE using a uniform distribution [49]. The mesh size is considered 8×8. To evaluate the reliability of MiCoF, the number of faulty switches increases from one

to six. All faulty switches are selected using a random function. The results are obtained using 10,000 iterations when traffic is uniform random.

Reliability is measured based on two metrics. In the first reliability metric (reliability1), we measure the number of combinations in which no packet is lost in the network. In the second metric (reliability2), the average number of successful packet arrivals at destinations over the total number of delivered packets is calculated. This value is obtained over all combinations of faulty switches.

The reliability values based on the first metric is shown in Figure 4.48. All three approaches are 100% reliable when there is a single fault in the network. As illustrated in this figure, HiPFaR and MiCoF have a higher degree of reliability compared with ReRS under multiple faulty switches. MiCoF always provides better reliability than HiPFaR. For instance, in 50% of all combinations of six faulty switches, the network using MiCoF is functioning normally without any packet loss. As illustrated in this figure, the reliabilities of HiPFaR and ReRS drop as the number of faults increases. Figure 4.49 shows the reliability values based on the second metric. Based on the simulation results, MiCoF is by 99.5% reliable under the cases of six faulty switches in the network. This value is considerably higher than the reliability value provided by ReRS which is about 63%.

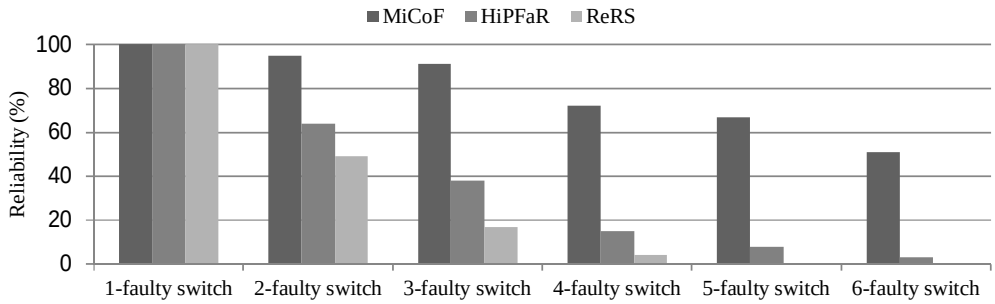

Figure 4.48: Reliability measurement based on the first metric

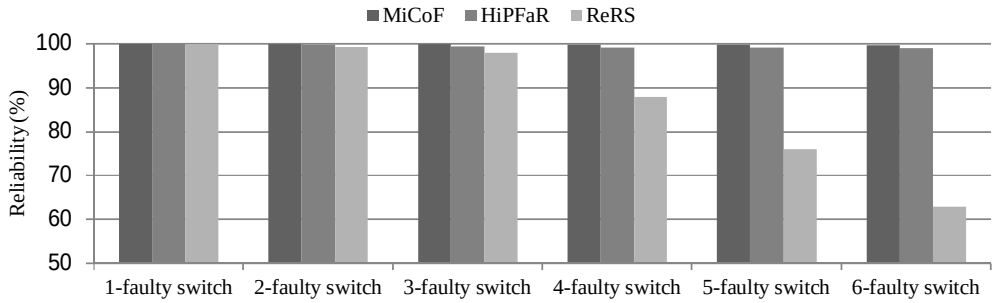

Figure 4.49: Reliability measurement based on the second metric

4.3.3.4 Hardware Analysis

To assess the area overhead and power consumption, the whole platform of each method is synthesized by Synopsys Design Compiler. We measured the area overhead and power consumption of the HiPFaR, MiCoF, and ReRS methods. In this set of analysis, HiPFaR and MiCoF have one and two virtual channels along the X and Y dimensions while ReRS does not use any virtual channel. Power consumption of all methods is measured under a single faulty switch in the network. For each scheme, we include network interfaces, switches, and communication links. All methods are synthesized under the TSMC 65nm technology at the operating frequency of 500MHz and supply voltage of 1V. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation is calculated using Synopsys PrimePower in an 8×8 mesh network.

The layout area and power consumption of each platform are shown in Table 4.3. As indicated in the table, ReRS has the lowest area overhead as it does not use any virtual channel. However, its power consumption is comparable with HiPFaR and MiCoF since by using ReRS hotspot creates around the fault and packets take unnecessary longer paths. MiCoF has a larger area overhead than the HiPFaR method as it uses extra resources to retain connectivity between surviving switches. MiCoF saves more power than HiPFaR by routing packets through only the shortest paths.

Table 4.3: Hardware implementation details

Network platforms	Area (mm^2)	Power (mW)
ReRS	2.126	1.466
HiPFaR	2.566	1.543
MiCoF	2.569	1.502

4.4 Summary of the Proposed Methods

In this chapter, we proposed four fault-tolerant routing algorithms; called MD, MAFA, HiPFaR, and MiCoF. Two of which are designed to tolerate faulty links and two others deal with faulty switches. These methods are mainly compared with two conventional methods, called RAFT and ReRS. All of these methods are able to tolerate a single fault by 100% but have different reliabilities against multiple faults in the network. The main characteristics of these methods are listed in Table 4.4. Note that reliability values in this table show the probabilities that a packet can be successfully delivered under the presence of faults in the network.

Table 4.4: Summarized characteristics of different fault-tolerant approaches

Methods	Type of fault	Use of non-minimal paths	Adaptiveness	Number of virtual channels	Reliability against two faults	Reliability against six faults
RAFT	Link	Even if a minimal path is available	Nearly fully adaptive	2 VCs along both dimensions	100%	86.1%
MD (proposed)	Link	Only when a minimal path does not exist	Nearly fully adaptive	1 and 2 VCs along the X and Y dimensions	99.8%	94.7%
MAFA (proposed)	Link	Only when a minimal path does not exist	Fully adaptive + deterministic	2 VCs along both dimensions	100%	97.8%
ReRS	Switch	Even if a minimal path is available	Static	No virtual channels	99.3%	63.1%
HiPFaR (proposed)	Switch	Only when a minimal path does not exist	Nearly fully adaptive	1 and 2 VCs along the X and Y dimensions	99.8%	99%
MiCoF (proposed)	Switch	Never	Nearly fully adaptive	1 and 2 VCs along the X and Y dimensions	99.9%	99.7%

Chapter 5

Unicast and Multicast Routing Algorithms for a 3D Mesh Network

NoC enables the integration of a large number of Intellectual Property (IP) cores into a chip [23], [24]. However, planar chip fabrication technology is facing new challenges in the deep submicron regime [10], [89]. By integrating diverse components into a two-dimensional (2D) chip, the manufacturing process can become highly complex [90]. In addition, wire delay and power consumption increase significantly by the usage of global interconnects in 2D designs. To overcome these limitations, technology is moving rapidly towards the concept of three-dimensional (3D) ICs where multiple active silicon layers are vertically stacked. 3D technology overcomes the limited floor-planning choices of 2D designs and allows each layer to be instantiated with a different technology [90]. The major advantages of 3D NoCs are the considerable reduction in the average wire length and wire delay, resulting in lower power consumption and higher performance [10], [11], [12], [13].

In this chapter, we investigate efficient communication protocols for a 3D mesh network with a mixture of unicast and multicast packets. We propose several partitioning methods, named TBP, VBP, and RP, for the path-based multicast approach, each with different levels of efficiency. In Two-Block Partitioning method (TBP), destinations are divided into two groups and a multicast packet is delivered for each group. In Vertical-Block Partitioning method (VBP), destinations are divided into several groups depending on their vertical columns and a multicast packet is generated per group. Recursive partitioning method (RP) tries to have a comparable number of switches within each partition while keeping the number of packets low. This result in lower average latency compared with TBP and VBP methods.

On top of all partitioning methods and in order to efficiently distribute the unicast and multicast packets more efficiently, we design a minimal and adaptive routing algorithm, called MAR, which is based on the Hamiltonian path. The algorithm is simple and does not require any virtual channel for neither unicast nor multicast packets. The main properties of the final approach which is the combination of the RP and MAR methods can be summarized as follows: 1) decreasing the latency of packets by addressing the non-optimal solutions of ordinary partitioning methods; 2) alleviating the traffic congestion by

providing the adaptive routing for both unicast and multicast packets; and 3) causing a relatively small area overhead mainly by not using virtual channels for deadlock avoidance and a simple implementation of the routing algorithm.

5.1 Traditional Approaches

The vast majority of traffic in Multi-Processor Systems-on-Chip (MPSoCs) consists of unicast traffic and most studies have assumed that the traffic is only unicast. Based on this assumption, the concept of unicast communication has been studied extensively in the literature [39], [43], [44], [81]. Unicast protocols are designed for better network performance assuming that all injected packets are unicast. However, if only a small percentage of the total traffic is multicast, the efficiency of the overall system is considerably reduced. Indeed, multicast communication has a great impact on the performance of Chip Multi-Processor (CMP) systems [91], [92].

The multicast communication is frequently present in many cache coherency protocols (e.g. directory-based protocols, token-based protocols, and Intel QPI protocol [48], [93]). For example, around 5% of total traffic in a SGI-Origin protocol (which is a directory based protocol) consists of multicast messages. In this protocol, message latency can be reduced by 50%, if multicast is supported in hardware, thus highlighting the importance of hardware-level multicast support [93]. It can also be taken into account that some cache coherence protocols are heavily multicast (e.g. around 80% of the token-based MOESI traffic consists of multicast traffic). In the following, we investigate traditional routing algorithms supporting both unicast and multicast traffic in the network.

5.1.1 Virtual Circuit Tree Multicasting (VCTM)

Virtual Circuit Tree Multicasting (VCTM) approach is proposed in [48] which is a tree-based method targeting a 2D mesh network. In VCTM, a virtual circuit tree should be built over the network having the source switch as a root and destinations as the leaves of the tree. Each switch maintains a table to store the information of different trees. An entry of this table is corresponding to one multicast packet. To form the tree, a single unicast packet is delivered per destination of the multicast packet. This packet updates the corresponding entry of the routing tables in its path. After the setup phase, the multicast packet is sent over the tree. The multicast packet uses the same virtual path which is already established by unicast packets.

Figure 5.1(a) shows an example of VCTM when the multicast packet is sent from the source switch S to the destination switches A, B, C, D, and E. At first, an entry is allocated to the multicast packet, if it is not already assigned. The ID of the multicast message is carried by each of the five setup unicast packets that are going to be sent to different destinations. These packets follow a dimension-order routing and update the routing tables regarding the path of the packet.

VCTM has several shortcomings as follows: If a multicast message includes many destinations, a large number of setup packets must be delivered into the network (before the real multicast packet is delivered) which increases the packet latency significantly. The area

overhead of VCTM is also relatively high due to maintaining a table at each switch to store the information of a virtual circuit tree. If the size of routing tables is smaller than the amount of multicast packets, the entries are overwritten by other packets. In this case, VCTM stops working as packets lose the information of their paths. The common disadvantage of tree-based methods, which exists in the VCTM method as well, is that a packet may hold several channels for extended periods of time to receive all requested output channels, thereby increasing network contention [27]. Finally, VCTM is based on deterministic algorithms and cannot provide adaptiveness for neither unicast nor multicast packets. Although in VCTM, packets are routed through the shortest paths from the source to each destination switch, this is not the least number of hops for a multicast packet. For example, in Figure 5.1(b) the multicast packet reaches to all destinations using the shortest paths while reducing the number of hops from sixteen in VCTM to nine.

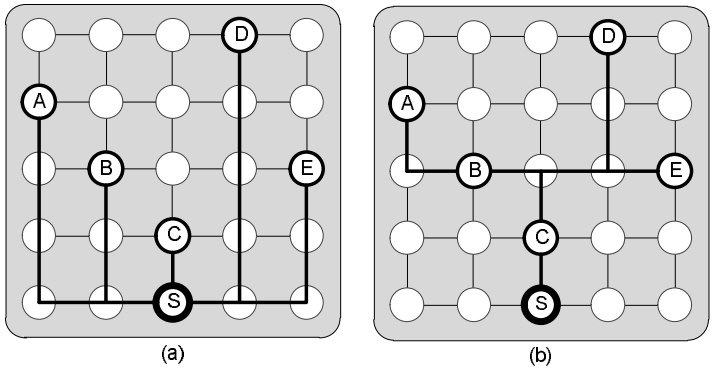

Figure 5.1: (a) VCTM (b) the number of hops can be reduced

5.1.2 Dual-Path Multicast Routing (DP)

The Dual-Path (DP) multicast approach is a path-based method presented in [94] for a 2D mesh network. The idea of this approach relies on using the Hamiltonian path. The Hamiltonian path strategy [94] guarantees that the network will be free of deadlock under unicast and multicast traffic. A Hamiltonian path visits every switch in a network exactly once. As shown in Figure 5.2(a), for each switch in an $a \times b$ mesh network, a label $L(x, y)$ is assigned as follows:

$$L(x,y) = (a \times y) + (x + 1) \qquad where\ y\ is\ even$$
$$L(x,y) = (a \times y) + (a - x) \qquad where\ y\ is\ odd$$

where x and y are the coordinates of the switch.

In the DP method, two directed Hamiltonian paths (or two subnetworks) are constructed by the labeling. The high channel subnetwork (Figure 5.2(b)) starts at the switch 1 and the low channel subnetwork (Figure 5.2(c)) ends at the switch 1. When the label of the destination switch is greater than the label of the source switch, the routing always takes place in the high channel subnetwork; otherwise it takes place in the low channel

subnetwork. The destinations are placed in two groups. One group contains all the destinations that can be reached using the high channel subnetwork (G_H), and the other contains the remaining destinations that can be reached using the low channel subnetwork (G_L). Thereby, every switch in G_H has a higher label than the source switch and every switch in G_L has a lower label than the source switch. To reduce the path length, the vertical channels that are not part of the Hamiltonian path could be used in appropriate directions. Deadlock is prevented as packets are routed in the network strictly in the ascending or descending order using separate resources.

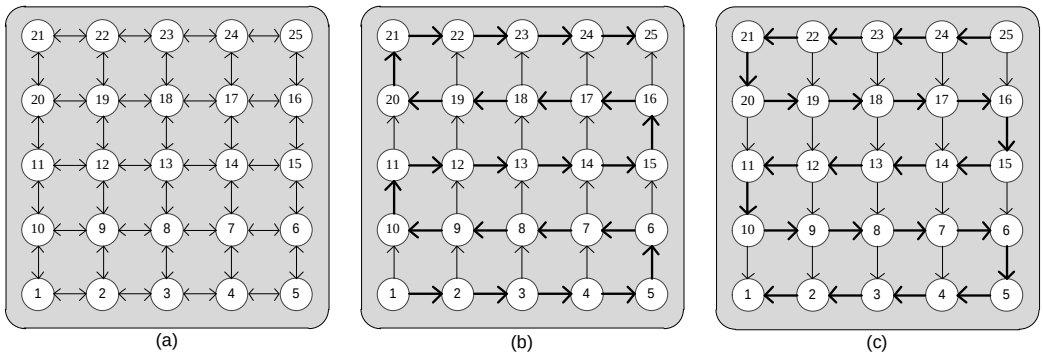

Figure 5.2: (a) A physical channel (b) high channel subnetwork (c) low channel subnetwork

In sum, all packets in the network (either unicast or multicast) should follow the Hamiltonian path, based on the following rules.

- For the high channel subnetwork:
  Rule1: North and East directions are allowed in even rows.
  Rule2: North and West directions are allowed in odd rows.
- For the low channel subnetwork:
  Rule1: South and West directions are allowed in even rows.
  Rule2: South and East directions are allowed in odd rows.

Consider the example illustrated in Figure 5.3 when the switch 13 sends its multicast packet to ten destinations as *m=(13,{2,4,6,8,9,11,17,20,23,24})*. Accordingly, two groups are organized. The first group has all the destinations that could be reached from the source switch using the high channel subnetwork, G_H*={17,20,23,24}*. The multicast packet always traverses in the ascending order as *{13,14,**17**,18,19,**20**,21,22,**23**,**24**}*. The second group, G_L*={11,9,8,6,4,2}*, has the remaining destinations that could be reached using the low channel subnetwork using the path *{13,12,**11**,10,**9**,**8**,7,**6**,5,**4**,3,**2**}*.

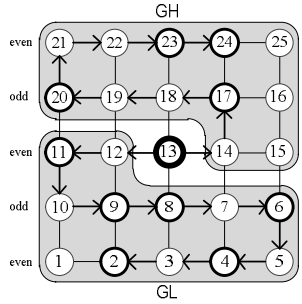

Figure 5.3: Dual-path multicast approach

5.1.3 Multi-Path Multicast Routing (MP)

To reduce the path lengths, in the Multi-Path (MP) multicast approach, G_H and G_L are also partitioned. The set G_H is divided into two subsets (G_H: G_{H1},G_{H2}). If the source switch is located in an odd row, G_{H1} covers the switches whose X coordinates are greater than or equal to that of the source switch and G_{H2} contains the remaining switches in G_H (Figure 5.4(a)). If the source switch is located in an even row, then G_{H1} covers the switches whose X coordinates are greater than the source switch (Figure 5.4(b)). The set G_L is partitioned in a similar way into two subsets (G_L: G_{L1},G_{L2}). Hence, all destinations of a multicast message are grouped into four disjointed subnetworks. An example of MP is illustrated in Figure 5.4(b). The destinations in G_H are divided into two subsets, which are $G_{H1}=\{20,23\}$ and $G_{H2}=\{17,24\}$. In the same way, the destinations in G_L are divided into two subsets, $G_{L1}=\{11,9,2\}$ and $G_{L2}=\{8,6,4\}$. Subsequently, all destinations in G_{H1} and G_{H2} should be sorted in ascending order, and the destinations in G_{L1} and G_{L2} should be sorted in descending order. Finally, one packet is created per group. All packets must follow the Hamiltonian path and reach destinations in the order they are arranged. The Multi-Path is a deterministic and deadlock-free approach that could be used for unicast and multicast routing simultaneously.

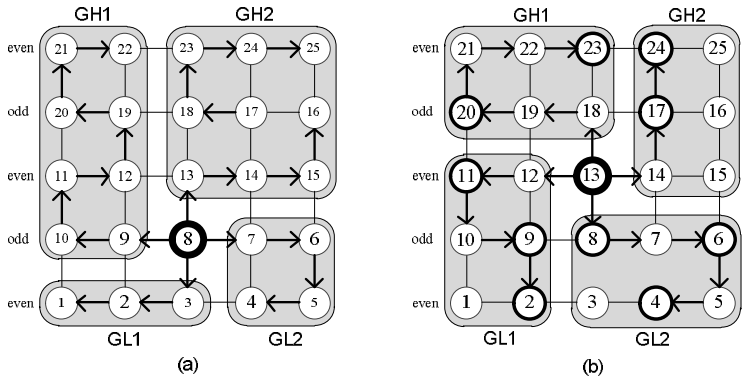

Figure 5.4: Multi-path multicast approach

5.1.4 Hamiltonian Adaptive Multicast and Unicast Model (HAMUM)

Adaptive routing algorithms can improve performance significantly compared with deterministic methods. Toward this direction, there are several proposals addressing the adaptivity for unicast packets [49], [81]. However, when both unicast and multicast packets are routed in the same network, the unicast packets have to obey the rules for multicast packets. For example, in VCTM, unicast packets are also routed according to the dimension-order routing. In DP and MP, unicast packets are delivered from the source to the destination switch through a single path. This path is compatible with those used by the multicast packets. Taking into account that unicast packets form the vast majority in most traffic profiles, the performance can be significantly affected by routing packets deterministically inside the network. In general, the purpose of supporting multicast traffic is to improve performance. However, using these algorithms, the performance is degraded due to the adaptivity restriction imposed on all unicast packets.

HAMUM [95] is one of our traditional methods, which allows both unicast and multicast packets to be adaptively routed inside the network. The simple idea behind this method is to deliver packets through all possible shortest paths which are located in the Hamiltonian path. For example, in Figure 5.5(a), the unicast packets generated at the source switches 2 and 3 can reach destinations switches 21 and 25 through different routes. Figure 5.5(b) shows two examples of multicast packets. As can be seen in this figure, packets can be routed adaptively between each two successive destination. All these paths shown in Figure 5.5(a) and Figure 5.5(b) are compatible with the rules defined by the Hamiltonian path.

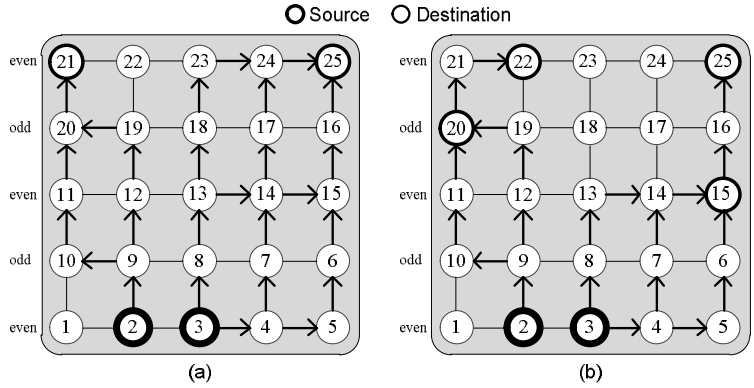

Figure 5.5. An example of HAMUM for (a) unicast packets (b) multicast packets

5.1.5 Summary of Traditional Methods

In low traffic loads, tree-based approaches (e.g VCTM) perform well as packets can easily receive the requested channels. In high traffic load, the performance is significantly reduced. This is due to the fact that a packet may hold (or wait to take) multiple outgoing channels simultaneously to deliver the packet. This increases the probability of packets'

blocking. Moreover, VCTM is a deterministic method, degrading performance. In contrast, path-based methods (e.g. DP and MP) are efficient in high traffic load, as there is no blocking issue. However, the main problem is that in DP and MP, the unicast and multicast packets are routed deterministically in the network which affects performance significantly. HAMUM address this problem by allowing both unicast and multicast packets to be adaptively routed within a 2D mesh network.

5.2 The Proposed Partitioning Methods for a 3D Mesh Network

Partitioning methods try to reduce latency and increase performance by an efficient partitioning of destinations into disjoint subsets [96], [97], [98]. The performance of a multicast operation can be measured in terms of its latency in delivering a packet to all its destinations [97], [98]. Multicast latency consists of two components: the startup latency and the network latency. The startup latency (startup-latency; SL) is the time required to create several packets (each with a different set of destinations), prepare packets, and start injecting them into the network. The network latency for multicast packets is defined as the time elapsed from the first flit injection into the network to the reception of the last flit by all destinations. Based on that, we define the mean multicast latency (mean-mul-latency; MML) and the maximum multicast latency (max-mul-latency; MxML). As previously stated, partitioning methods help in reducing the network latency component. In particular, these methods divide the network into several logical partitions and assign destinations to different sets, one set for each partition and including destinations that belong to that partition. Smart partitioning methods must balance the sets and reduce the path length within each partition.

However, breaking the network into logical partitions may have the following deficiencies: 1- A large number of network partitions leads to additional latency as more startup packets (SP) need to be prepared at the source switch and this latency is usually high. 2- An unbalanced configuration of partitions will create long paths within the network. In both cases the latency of the multicast operation will be increased.

5.2.1 Hamiltonian Path in a 3D Mesh Network

The concept of the Hamiltonian path can be used in NoCs for different purposes, for example to support multicast communication [96] or to tolerate faults [99]. Several Hamiltonian paths can be considered in the mesh topology. In an $a \times b \times c$ mesh network, each switch is labeled by an integer value according to its x, y and z coordinates. The following equations show one possibility of assigning the labels, which we utilize in this thesis [96], [97], [98]:

$$L(x,y,z) = \{(a \times b \times z) + (a \times y) + (x+1)\} \quad \text{where } z: even, y: even$$
$$L(x,y,z) = \{(a \times b \times z) + (a \times y) + (a-x)\} \quad \text{where } z: even, y: odd$$
$$L(x,y,z) = \{(a \times b \times z) + (a \times (b-y-1)) + (a-x)\} \quad \text{where } z: odd, y: even$$
$$L(x,y,z) = \{(a \times b \times z) + (a \times (b-y-1)) + (x+1)\} \quad \text{where } z: odd, y: odd$$

As exhibited in Figure 5.6, two directed Hamiltonian paths (or two subnetworks) are constructed by this labeling. The high channel subnetwork (Figure 5.6(b)) and the low channel subnetwork (Figure 5.6(c)) are defined similar to a 2D mesh network. In case the label of the destination switch is greater than the label of the source switch, the routing always takes place in the high channel subnetwork; otherwise it takes place in the low channel subnetwork.

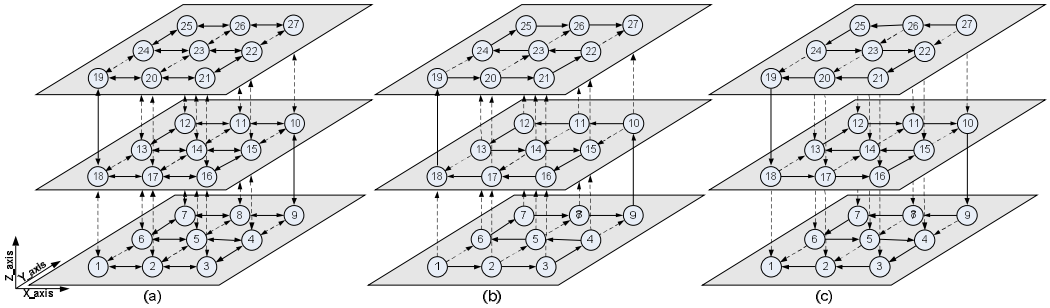

Figure 5.6: (a) A 3×3×3 mesh network with the label assignment (b) high channel (c) low channel subnetworks. The solid lines indicate the Hamiltonian path and dashed lines indicate the links that could be used to reduce the path length

5.2.2 Two-Block Partitioning Method (TBP)

By using the Hamiltonian path, the network is divided into high channel and low channel subnetworks. In the partitioning methods, destinations (at each source switch) are grouped in two sets. One set includes all the destinations that are reached using the high channel subnetwork, and the other set includes the remaining destinations reached using the low channel subnetwork.

In the next section, we explain the TBP method in detail. Notice that the TBP method is a straight forward extension of the dual-path multicast from a 2D NoC to a 3D NoC and it can be seen as a naïve method since no effort is made to balance the two sets. However, this method is used as a reference method for comparison purposes. For each partitioning method, we provide an analysis of the number of startup packets (SP), the latency of multicast operations (MML and MxML), and the average latency of unicast operations (AUL). Analytical models are provided for unicast and multicast packets assuming zero-load latency [13]. Based on the zero-load latency, a packet never contends for network resources with other packets. Under this assumption, the performance of each approach can be measured based on the number of hops required for delivering a packet from a source switch to its destination(s). Contention effects will be accounted both analytically and in experiments with our simulation platform, presented in Section 5.4.

Figure 5.7(a) shows an example of the TBP partitioning policy and the portions of each partition that depend on the source switch position. As illustrated in this figure, if the source switch is located at the middle layer, two partitions cover comparable numbers of switches but still with a large number of switches in both partitions. However in

Figure 5.7(b), one partition contains considerably more switches than the other. Now, suppose that the multicast message *m=(7,{2,3,20,26,45})* is generated at the switch *7*. Destinations are split into two sets and should be visited accordingly to their labels: G_H*={20,26,45}* and G_L*={3,2}*. The packet created for G_H uses the Hamiltonian path as follows: *{7,10,11,12,13,**20**,21,22,23,**26**,39,42,43,44,**45**}* where fourteen hops are needed to reach the last destination. The packet path for the G_L is:*{7,6,**3**,**2**}* where three hops are required for delivering the packet to all its destinations. In the TBP method, the number of startup packets is low and never becomes larger than two. However, it suffers from high network latency due to unbalanced partitions and the high probability of forming long paths in the network. The TBP algorithm is shown in Figure 5.8.

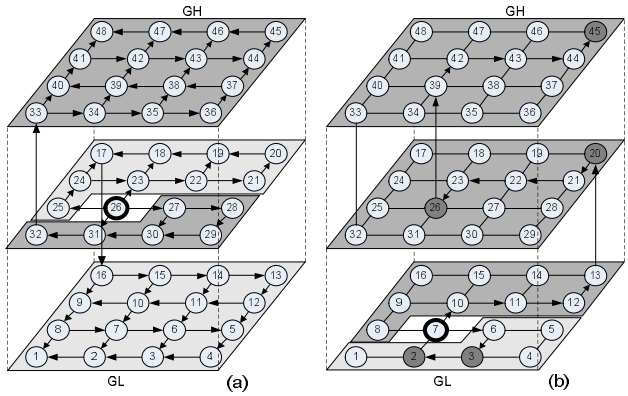

Figure 5.7: The TBP method (a) balanced (b) unbalanced partitions

```
ALGORITHM: Two-Block Partitioning (TBP)
                **-------------------------------------------------**
Definitions:    GH , GL: High and Low channel subnetwork
                **-------------------------------------------------**
for "i: 1 to number of destinations" loop
    if (label(destination(i)) > label(source) then
        GH <= label (destination);
    else
        GL <= label (destination);
    end if;
end loop;
Construct a packet for each non-empty set;
```

Figure 5.8: The pseudo code of the TBP method

5.2.2.1 Avg-Uni-Latency (AUL)

Since packets can utilize shortcut channels without introducing new cycles, the paths taken by unicast packets are reduced to the shortest paths between each pair of source and destination switches. Assuming uniform distribution of destinations and using the shortest paths for unicast packets, the average unicast latency for an $a \times b \times c$ mesh network is given by [13]:

$$AUL_{3D} = \frac{a^2bc+ab^2c+abc^2-ac-bc-ab}{3abc} \qquad (1)$$

Regardless of the partitioning method used, unicast packets are routed within the network in the same manner, so the formula (1) is valid for the VBP and RP methods as well. This equation can be easily applied to one-dimensional (when b=1 and c=1) and two-dimensional (when c=1) mesh networks.

5.2.2.2 Mean-Mul-Latency (MML)

The multicast latency depends on the number and the location of destinations. This makes computing the analytical multicast latency complex. In order to simplify this complexity, we consider that the latency of a multicast packet is set by the final destination so that the multicast packet always takes the longest path within the network (without using shortcut channels). This is the worst case. For instance, in the example of Figure 5.7, the two packets have their final destinations set as 45 and 2, and their distances from the source switch are fourteen and three hops, respectively. However, in MML, we consider the longest path from the source to the destinations 45 and 2 which are forty-one and six hops, respectively. In the TBP method, the path between two destinations to reach in a sequential order is minimal while the path from the source to each destination is not necessarily a minimal path. As an example in Figure 5.7, the path from the switch 7 to the switch 20 (similarly, from 20 to 26, and from 26 to 45) are minimal; however, the paths from the switch 7 to the switch 26 (similarly, from 7 to 45) are non-minimal. According to this assumption, MML for every switch j in an $a \times b \times c$ network is: (where n is the total number of switches in the network.)

$$MML_j = \frac{1}{n}\left(\sum_{i=1}^{i=j-1} i + \sum_{i=j+1}^{i=n} (i-j)\right) \qquad (2)$$

The average multicast latency for the whole network in the TBP method can be obtained by:

$$MML_{TBP} = \frac{1}{n}\sum_{j=1}^{j=n} MML_j = \frac{1}{n}\sum_{j=1}^{j=n}\left(\frac{1}{n}\left(\sum_{i=1}^{i=j-1} i + \sum_{i=j+1}^{i=n} (i-j)\right)\right) = \frac{n^2-1}{3n} \qquad (3)$$

This equation is proved by using the following set of formulas. The sum of partial factorial formula is given by:

$$\frac{m!}{0!} + \frac{(m+1)!}{1!} + \frac{(m+2)!}{2!} + \ldots + \frac{(m+n-1)!}{(n-1)!} = \frac{(m+n)!}{(m+1)(n-1)!}$$

For all positive integers, we get the formula when $m = 1$ or $m = 2$:

$$1+2+3+\ldots+n = \sum_{i=1}^{i=n} i = \frac{n(n+1)}{2} \qquad (4)$$

$$1\times2+2\times3+3\times4+\ldots+n\times(n+1)=\sum_{i=1}^{i=n} i(i+1)=\frac{n(n+1)(n+2)}{3} \qquad (5)$$

By using formulas (4) and (5), MML_{TBP} can be written as follow and the equation (3) is proved:

$$MML_{TBP}=\frac{1}{2n^2}\left(\sum_{j=1}^{j=n}(j-1)(j)+\sum_{j=1}^{j=n}(n-j)(n-j+1)\right)=\frac{(n-1)(n+1)}{3n}=\frac{n^2-1}{3n}$$

By another perspective, the network can be seen as a 1D network (Figure 5.9) where the only dimension (a') contains $a'=a\times b\times c$ switches. Therefore, by using the formula (1) for a 1D network (when $b=1$ and $c=1$), MML_{TBP} is:

$$MML_{TBP}=\frac{a'^2-1}{3a'}=\frac{n^2-1}{3n}$$

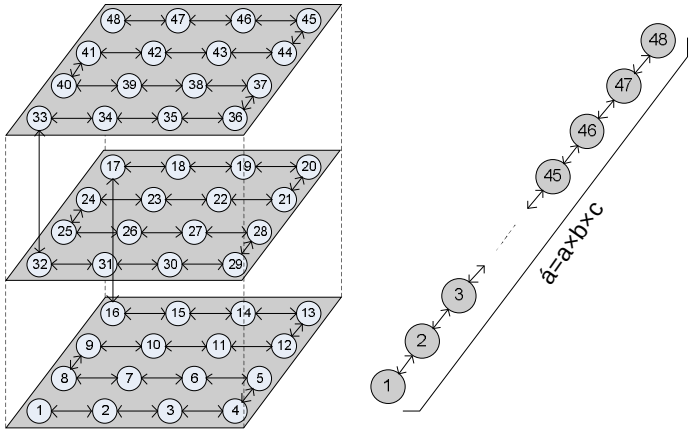

Figure 5.9: Measuring MML for the TBP method

5.2.2.3 Max-Mul-Latency (MxML)

MxML is the time when a multicast operation is completed and the multicast packet has reached all its destinations. To calculate max-mul-latency, we consider the number of hops that can be taken by a multicast packet from a source switch to the most distant switch. For an instance, when the source switch is at switch 7, the longest path is related to the switch 48 which requires the maximum of 41 hops. MxML for a source switch j is given by:

$$MxML_j=\begin{cases} n-j & \text{if } 0\le j\le \left\lfloor\frac{n}{2}\right\rfloor \\ j-1 & \text{if } \left\lceil\frac{n}{2}\right\rceil\le j\le n \end{cases}$$

We calculate MxML for half of the network which is the same as the next half, and thus MxML for the whole network is measured by:

$$MxML_{TBP} = \frac{2}{n}\sum_{j=1}^{j=n/2}(n-j) = \begin{cases} \frac{3n-2}{4} & \text{if n:even} \\ \frac{3n^2-2n-1}{4n} & \text{if n:odd} \end{cases}$$

5.2.2.4 Startup-Packet (SP)

In TBP, destinations are split in two sets. Thus, the number of maximum startup packets (SP) is set to two regardless of the source switch location. There are two exceptions regarding the first and last switches which can deliver only one multicast packet to the network.

5.2.3 Vertical Block Partitioning (VBP)

The TBP method performs well when the network size is small. However, as the network size enlarges, it may take a long path to deliver the multicast packet to all destinations and thus increasing latency. In Vertical Block Partitioning (VBP), similar to the TBP method, the network is partitioned into high and low channel subnetworks. In an additional step, each subnetwork is vertically partitioned such that switches in the same column (with the same a value in an $a \times b \times c$ network) are included in a new set. The algorithm is shown in Figure 5.10.

```
ALGORITHM: Vertical-Block Partitioning (VBP)
                    **-------------------------------------------------**
Definitions:    GH ,GL: High and Low channel subnetwork
                Xd: The X coordinate of destinations
                    **-------------------------------------------------**
for "i: 1 to the number of destinations" loop
        if (label(destination(i)) > label(source) then
                case "Xd(i)" is
                        when 1 => GH1 <= label (destination);
                        when 2 => GH2 <= label (destination);
                        ...
                        when a => GHa <= label (destination);
                end case;
        else
                case "Xd(i)" is
                        when 1 => GL1 <= label (destination);
                        when 2 => GL2 <= label (destination);
                        ...
                        when a => GLa <= label (destination);
                end case;
        end if;
end loop;
Construct a packet for each non-empty set;
```

Figure 5.10: The pseudo code of the VBP method

As illustrated in Figure 5.11, this scheme does not guarantee balanced partitions. For the switch located at 26, partitions are balanced, but they are not balanced when the source is at the switch 7 (i.e. four subnetworks cover more switches than the others). Moreover, the time required to prepare at most eight packets is considered as the number of startup packets. For the multicast message *m=(7,{2,3,20,26,45})*, four sets are formed: G_{H2}*={26}*, G_{H4}*={20,45}*, G_{L2}*={2}*, and G_{L3}*={3}*. One packet is generated for each set and paths are *{7,**26**}*, *{7,10,11,12,13,**20**,**45**}*, *{7,**2**}*, and *{7,6,**3**}* where the maximum hop count is six.

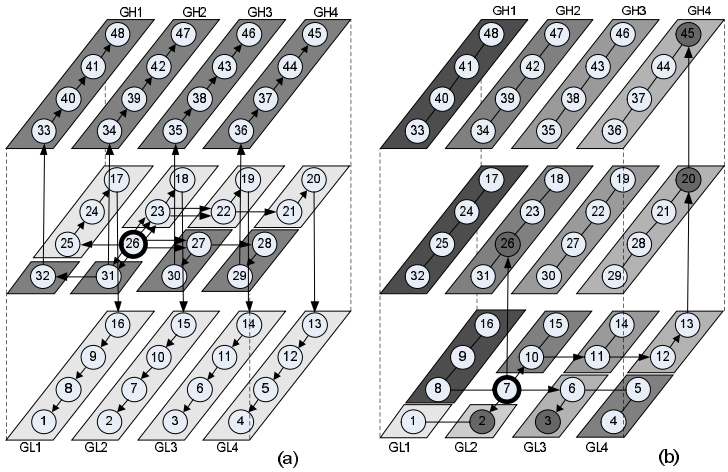

Figure 5.11: The VBP method (a) balanced partitions (b) unbalanced partitions

This scheme has several advantages over the TBP method as it achieves a high level of parallelism; avoids the creation of long paths and reduces the network latency. The VBP method increases, however, the number of startup packets as it requires up to *2a* packets in an a×*b*×*c* network. In addition, this scheme does not guarantee balanced partitions as it is balanced only when the source switch is located in middle layers while some partitions may cover considerably more switches than the others when the source switch is located at the top or bottom layer.

5.2.3.1 Mean-Mul-Latency (MML)

Since the network is symmetric and is partitioned vertically, the MML value can be measured in one vertical partition and then generalized to other partitions. For this purpose, we consider that an *a*×*b*×*c* mesh network is divided into *a* vertical partitions where each partition contains *bc* switches. Using formulas (2) and (3), the MML value for a source switch *j* inside a vertical partition and for all switches in a partition can be computed as follow:

$$MML_j = \frac{1}{bc}\left(\sum_{i=1}^{i=j-1} i + \sum_{i=j+1}^{i=bc} (i-j)\right) \qquad MML_{bc} = \frac{1}{bc}\sum_{j=1}^{j=bc}\frac{1}{bc}\left(\sum_{i=1}^{i=j-1} i + \sum_{i=j+1}^{i=bc} (i-j)\right) = \frac{(bc)^2-1}{3bc}$$

Moreover, packets are required to traverse in the X dimension to reach their relative vertical partitions. For example, if $a=4$ in an $a\times b\times c$ network and the source switch is located at the first vertical partition, it takes 1, 2 and 3 hops to reach the second, third and fourth vertical partitions, respectively. So, this value should be considered when measuring the MML value.

$$MML_a = \frac{1}{a}\sum_{j=1}^{j=a}\frac{1}{a}\left(\sum_{i=1}^{i=j-1} i + \sum_{i=j+1}^{i=a}(i-j)\right) = \frac{a^2-1}{3a} \qquad (6)$$

Finally, the MML value for the whole network is given by:

$$MML_{VBP} = MML_a + MML_{bc} = \frac{a^2-1}{3a} + \frac{(bc)^2-1}{3bc} = \frac{a^2bc+ab^2c^2-bc-a}{3abc}$$

From another point of view, the network can be viewed as a 2D network ($a\times b'$) where b′=b×c (Figure 5.12). The dimension-order routing can be utilized for packets, and thus, by using formula (1) in a 2D network (when $c=1$) the average multicast latency can be measured by:

$$MML_{VBP} = \frac{a^2b'+ab'^2-a-b'}{3ab'} = \frac{a^2bc+ab^2c^2-bc-a}{3abc}$$

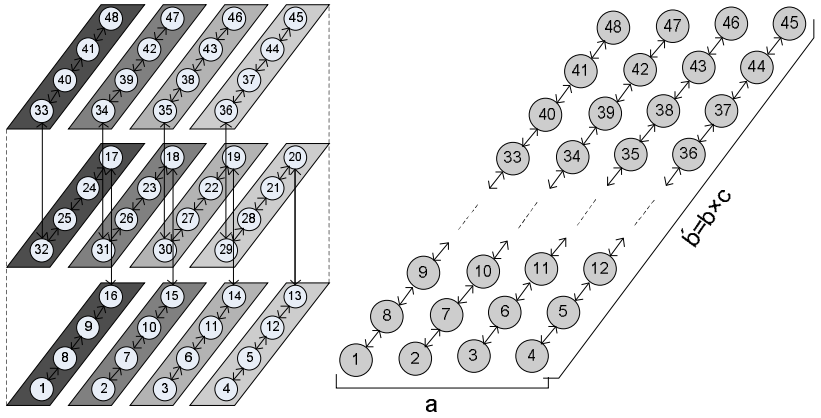

Figure 5.12: Measuring MML for the TBP method

5.2.3.2 Max-Mul-Latency (MxML)

In the VBP method, the network is divided into several vertical partitions according to the value a in an $a\times b\times c$ network. Thereby, the following formula is used for computing the MxML value in the network. In this formula, the first term shows the maximum number of hops within each partition while the second term indicates the required number of hops to reach a partition.

$$MxML_{VBP} = \frac{2}{n}\sum_{i=1}^{i=n/2}\left(\left\lceil\frac{n-j}{a}\right\rceil + \frac{a^2-1}{3a}\right)$$

5.2.3.3 Startup-Packet (SP)

In the VBP method, the number of partitions depends on the location of switches that result in different startup packets. The source switches 1, 2, 3, and 4 located in the first row in Figure 5.11 (or switches 48, 47, 46, and 45 in the last row), for instance, divide the network into 4, 5, 6, and 7 partitions, respectively, while the other switches divide the network into 8 partitions. As a result, the average number of startup packets for the VBP method in an $a \times b \times c$ network is:

$$SP_{VBP} = \frac{(3a^2-a)+((abc-2a)(2a))}{abc} = \frac{2a^2bc-a^2-a}{abc}$$

5.2.4 Recursive Partitioning (RP)

The VBP method suggests a better degree of parallelism and lower latency as a packet is dedicated to a smaller set of destination switches and thus a shorter path is taken by each packet. The main disadvantage of this method is in the possibility of creating unbalanced partitions as a group may contain a large set of switches than others. This results in taking long paths by some packets and short paths by others, keeping the multicast latency still high. The objective of the Recursive Partitioning (RP) method is to optimize the number of switches that can be included in a partition and achieve parallelism. In this method, the network is recursively partitioned until each partition contains *k* switches. In the worst case, the network is partitioned into *2a* vertical partitions like in the VBP method. So, we have considered the value *k* as a reference value indicating the number of switches in each partition of the VBP method, i.e. *(k=bc)* in an $a \times b \times c$ network. The RP algorithm is shown in Figure 5.13. An example of the RP method is illustrated in Figure 5.14(a) where a multicast message is generated at the source switch 26. The required steps of the RP method can be expressed as follows:

- **Step1:** The value *k* is set with 12 switches in a 4×4×3 network.
- **Step2:** The network is divided into two partitions using the TBP method. Figure 5.7(a) shows two formed partitions when the source switch is located at the switch 26.
- **Step3:** If the number of switches in a partition exceeds the reference value *k*, the partition is divided into two new partitions. This step is repeated until all partitions in the network cover at most *k* switches. Following the example of Figure 5.7(a), 22 switches are covered by the high channel subnetwork which is greater than *k=12*. So, the high channel subnetwork needs to be further divided into two new partitions (G_{H1} and G_{H2} as shown in Figure 5.14(a)). The G_{H1} and G_{H2} partitions contain 10 and 12 switches, respectively. Since both numbers are less than or equal to *k=12*, no further partitioning is needed for the high channel subnetwork. The same partitioning technique is applied to the low channel subnetwork.

```
ALGORITHM: Recursive Partitioning (RP)
                    **-----------------------------------------------**
Definitions:    GH, GL: High and Low channel subnetwork
                Num_P: Number of switches in the partition
                (Xp,Yp,Zp): X, Y, and Z coordinates of the given partition
                k value: (k=bc) in a×b×c network
                    **-----------------------------------------------**
function Partitioning (G,Num_P) is
        if (Num_P = n) then
                //Partition the network using the TBP method
                G => GH,GL;
                Partitioning (GH,Num_PH);
                Partitioning (GL,Num_PL);
        elsif (Num_P > k) then
                //Divide the given P into two new partitions (Gi,Gi+1)
                G => Gi→((0:[(Xp)/2]),Yp,Zp),
                Gi+1→(([(Xp)/2]:Xp-1),Yp,Zp);
                Partitioning (Gi,Num_Pi);
                Partitioning (Gi+1,Num_Pi+1);
        else
                Return (G,Num_P);
        end if;
end;
Construct a packet for each non-empty set;
```

Figure 5.13: The pseudo code of the RP method

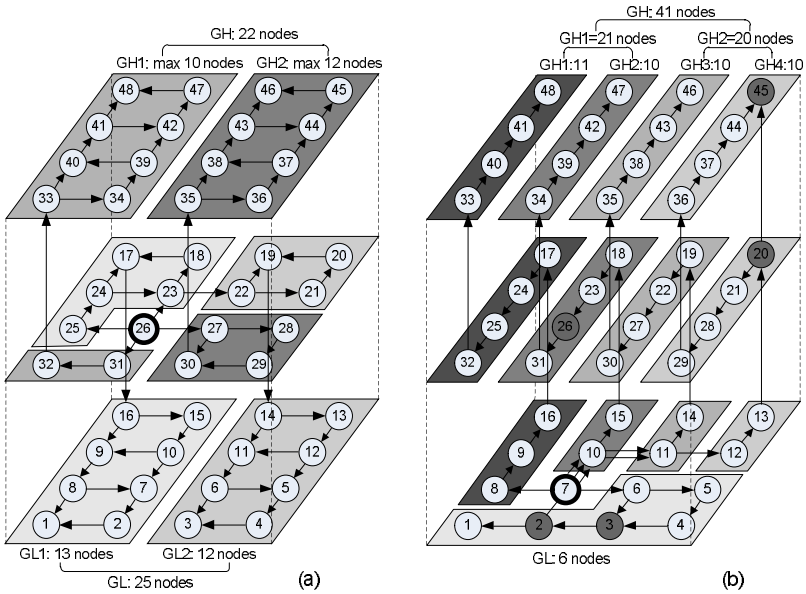

Figure 5.14: RP when the source switch is at (a) the switch 26 (b) the switch 7

Figure 5.14(b) shows another example of the RP method where the multicast message is *m=(7,{2,3,20,26,45})*. In this example three packets are formed and their paths are *{7,10,11,12,13,**20**,**45**},{7,**26**}*, and *{7,6,**3**,**2**}* as the maximum latency is six hops.

In brief, this scheme has a similar performance in avoiding long paths as the VBP method while it provides better parallelism as the number of switches is comparable among partitions. By considering the RP method, the creation of balanced partitions is less dependent of the source switch position, and thus it avoids long paths in the network and increases parallelism while keeping the number of startup packets relatively low.

5.2.4.1 Mean-Mul-Latency (MML)

In the RP method, a network is divided into high and low channel subnetworks and each subnetwork is recursively partitioned until all partitions cover around *k* switches, where *k=bc*. The next set of formulas is concerned only the high channel subnetwork while the low channel subnetwork has similar formulas. According to this assumption, if the high channel subnetwork covers *x* switches where *x>k*, it is divided into two new partitions. Each of the formed partitions might still cover more than *k* switches (*x>k)*. Thereby, the partition is further divided into two new partitions. In other words, the MML formula is recursively called until all partitions cover at most *k* switches. Finally, the average multicast latency is computed when the number of switches (*x)* in a partition become less than or equal to the value of *k*:

$$MML_x = \begin{cases} \dfrac{MML_{\lceil \frac{x}{2} \rceil} + MML_{\lfloor \frac{x}{2} \rfloor}}{2} & \text{where } x>k \\ \dfrac{1}{x}\sum_{i=1}^{i=x} i = \dfrac{x+1}{2} & \text{where } 0<x\leq k \\ 0 & \text{where } x=0 \end{cases}$$

Similar to formula (6), in order to deliver packets from the source switch to different partitions, average multicast latency in the X dimension should be taken into account:

$$MML_A = \frac{a^2-1}{3a}$$

Finally, the MML for the RP method is given by:

$$MML_{RP} = \frac{1}{2n}\sum_{j=1}^{j=n}\left(MML_{(j-1)}^{low} + MML_{(n-(j+1))}^{high}\right) + MML_A = \frac{1}{n}\sum_{i=1}^{i=n} MML_{(i-1)} + MML_A$$

5.2.4.2 Max-Mul-Latency (MxML)

For measuring MxML, the number of switches in the biggest partition should be identified. To do this, we first find the MxML value for the high and low channel subnetworks and then determine the number of switches in the biggest partition of the network.

$$MxML = Max(MxML^{High}, MxML^{Low}), \quad MxML_x^{High} \text{ or } MxML_x^{Low} = \begin{cases} Max(MxML_{\lceil \frac{x}{2} \rceil}, MxML_{\lfloor \frac{x}{2} \rfloor}) & \text{where } x > k \\ \frac{1}{x}\sum_{i=1}^{i=x} i = \frac{x+1}{2} & \text{where } 0 < x \leq k \\ 0 & \text{where } x = 0 \end{cases}$$

To compute the MxML value, the following formula is utilized:

$$MxML_{RP} = \frac{2}{n}\sum_{i=1}^{i=n/2} (MxML + MML_A)$$

5.2.4.3 Startup-Packet (SP)

Clearly, in the case that $x \leq k$, the number of startup packets is equal to 1. However, when $x > k$, the partition needs to be divided into two new partitions and the SP equation is called for every newly formed partition.

$$SP_x = \begin{cases} SP_{\lceil \frac{x}{2} \rceil} + SP_{\lfloor \frac{x}{2} \rfloor} & \text{where } x > k \\ 1 & \text{where } x \leq k \end{cases}$$

5.3 The Proposed Adaptive Routing Algorithm

In the previous section, we provide different partitioning methods. All of them require a routing algorithm capable of forwarding all the packets to their sets of destinations.

5.3.1 Minimal Adaptive Routing (MAR)

We present a Minimal Adaptive Routing (MAR) algorithm based on the Hamiltonian path. Using MAR, unicast and multicast packets can be adaptively routed inside the network. Each switch in the graph has a label *(L)* determined by the Hamiltonian path labeling mechanism. For a given switch *c* and destination *d*, the MAR algorithm finds possible neighbors *{p}* of the current switch that can be selected to deliver a packet:

$$\text{if } L(c) < L(d) \text{ then } MAR(c,d) = \{\{p\} \mid p \in P \text{ and } L(c) < L(p) \leq L(d) \text{ and } ((x_c,y_c,z_c) \leq (x_p,y_p,z_p) \leq (x_d,y_d,z_d) \text{ or } (x_c,y_c,z_c) \geq (x_p,y_p,z_p) \geq (x_d,y_d,z_d))\}$$

$$\text{if } L(c) > L(d) \text{ then } MAR(c,d) = \{\{p\} \mid p \in P \text{ and } L(d) \leq L(p) < L(c) \text{ and } ((x_c,y_c,z_c) \leq (x_p,y_p,z_p) \leq (x_d,y_d,z_d) \text{ or } (x_c,y_c,z_c) \geq (x_p,y_p,z_p) \geq (x_d,y_d,z_d))\}$$

The MAR algorithm is implemented at switches and can be described in three steps as follows:

- **Step1:** Determines the neighbors of switch *c* that can be used to move a packet closer to its destination *d*. The pseudo code for Step1 is shown in Figure 5.15.

- **Step2:** due to the fact that in the Hamiltonian path all switches are visited in ascending order (in the high channel subnetwork) or descending order (in the low channel subnetwork), all of the selected neighbors in Step1 do not necessarily satisfy the ordering constraint. Therefore, if the labels of the selected neighbors (in Step1) are between the label of the switch *c* and the destination *d*, it/they can be selected as the next hop. The pseudo code for Step2 is shown in Figure 5.15.
- **Step3:** since the MAR algorithm provides several choices at each switch, the goal of Step3 is to route a packet through the less congested neighboring switch. So, in the case when the packet can be forwarded through multiple neighboring switches, the congestion values of the input buffers in the selected neighbors are checked and then the packet is sent to the neighbor with the smallest stress value.

```
ALGORITHM: Minimal adaptive routing for a 3D mesh network
                    **-------------------------------------------------**
Definitions:    (Xc,Yc,Zc): X,Y, and Z coordinates of the current switch
                (Xd,Yd,Zd): X,Y, and Z coordinates of the destination switch
                    **-------------------------------------------------**
------ STEP 1 ------
X_dir = East when (Xc<Xd) else West;
Y_dir = North when (Yc<Yd) else South;
Z_dir = Up when (Zc<Zd) else Down;
------ STEP 2 ------
if ((label(currentSwitch) = label(destSwitch)) then
        select the local port;
elsif ((label(currentSwitch) < label(destSwitch)) then
        ----------High Channel Subnetwork--------
        if (label(currentSwitch) < label(neighbor(X_dir))) and
           (label(neighbor(X_dir)) < label(destSwitch)) then
                first choice <= neighbor(X_dir); end if;
        if (label(currentSwitch) < label(neighbor(Y_dir))) and
           (label(neighbor(Y_dir)) < label(destSwitch)) then
                second choice <= neighbor(Y_dir); end if;
        if (label(currentSwitch) < label(neighbor(Z_dir))) and
           (label(neighbor(Z_dir)) < Label(destSwitch)) then
                third choice <= neighbor(Z_dir); end if;
elsif ((label(currentSwitch) >Label(destSwitch)) then
        ----------Low Channel Subnetwork---------
        if (label(currentSwitch) >label(neighbor(X_dir))) and
           (label(neighbor(X_dir)) > label(destSwitch)) then
                first choice <= neighbor(X_dir); end if;
        if (label(currentSwitch) > label(neighbor(Y_dir))) and
           (label(neighbor(Y_dir)) > label(destSwitch)) then
                second choice <= neighbor(Y_dir); end if;
        if (label(currentSwitch) >label(neighbor(Z_dir))) and
           (label(neighbor(Z_dir)) > label(destSwitch)) then
                third choice <= neighbor(Z_dir); end if;
end if;
```

Figure 5.15: The pseudo code of the MAR algorithm

An example of the MAR algorithm is illustrated in Figure 5.16(a) where the source and destination are at the switches 6 and 48, respectively. According to the algorithm, in the first step the neighbors are chosen in a manner that brings the packet closer to its destination, i.e. *p={7,11,27}*. At the second step, the selected neighbors in Step1 are checked to determine whether they are in the Hamiltonian path or not. Since the labels of the three selected neighbors are between the labels of the current switch *(c=6)* and destination switch *(d=48)*, the packet can be routed through all of them. Suppose that the neighbor *p=11* has a smaller congestion value than the other neighbors, so the algorithm chooses this neighbor to forward the packet. If we continue with the switch *c=11*, this switch has three neighbors belonging to the minimal paths, i.e. *p={10,14,22}*. However, only two of them *(p={14,22})* have the labels greater than the label of the current switch *(c=11)* and lower than the label of the destination switch *(d=48)*. Finally, according to the stress values of the input buffers in the corresponding direction, one of them is selected as the next hop. The algorithm is repeated for the rest of the switches until the packet reaches the final destination. Figure 5.16(b) shows all possible shortest paths from the source switch *(c=6)* to the destination switch *(d=48)*. It is worth noting that the stress value is updated whenever a new flit enters or leaves the buffer (flit events: flit_tx or flit_rx).

The MAR algorithm can be adapted for multicast packets such that alternative paths are used to route a packet between the source switch and the first destination and also between successive destinations. An example is shown in Figure 5.16(c) where the source switch (*c=6*) forwards a multicast packet towards three destinations (*D={15,32,46}*). The MAR algorithm provides a set of alternative paths to send a packet from the source switch to the first destination (*d1=15*). Similarly, the packet can be adaptively routed between each two destinations. For example, at the switch 15, the packet can make progress towards the destination 32 either by selecting the switch 18 in the next layer or the switch 16 in the current layer. The MAR algorithm is compatible with all methods supporting the Hamiltonian path. Therefore, all the TBP, VBP and RP methods can utilize the MAR algorithm for both unicast and multicast packets.

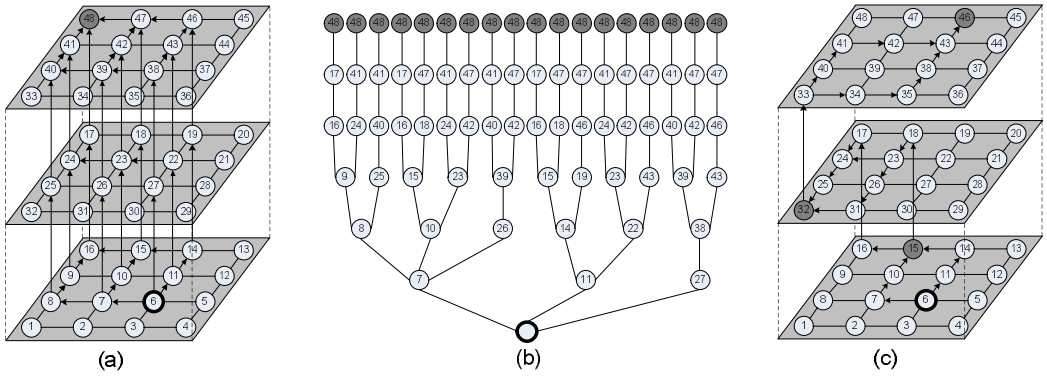

Figure 5.16: (a) An example of the MAR algorithm for a unicast packet (b) showing all possible paths between the source 6 and destination 48 (c) An example of the MAR algorithm for a multicast packet

5.3.2 Deadlock Avoidance

Deadlock is a situation where packets cyclically wait for network resources to be released. To show that the proposed algorithm is deadlock-free, we need to prove that the channel dependency graph (CDG) is acyclic [94]. First we demonstrate the MAR algorithm is deadlock-free for unicast packets. Then, we extend the concept to multicast packets, taking into account the partitioning methods.

The MAR algorithm follows the Hamiltonian paths and divides the network into two disjoint sets of channels. Those packets going to higher-labeled destinations ($L(d)>L(s)$) are routed through one set (the high channel subnetwork) and those packets going to lower-labeled destinations ($L(d)>L(s)$) are routed through the other set of channels (the low channel subnetwork). Dependencies between channels of one set do not close cycles in the CDG as switches are ordered in an ascending or descending order. To close a cycle in the high channel subnetwork, a packet may require requesting a channel that forwards the packet to a lower-labeled switch, which is not allowed by the MAR algorithm. The same applies for the low channel subnetwork. Also, as no packet changes from one subnetwork to the other, no cycles are formed between both subnetworks. Therefore, the MAR algorithm is deadlock-free for unicast traffic.

For multicast traffic two subtle cases may arise. The first one influenced by the partitioning method, and the second one as a structural problem of path-based multicast. In the first one, if the ordering of destinations in a multicast packet is not carefully considered, a deadlock may have occurred. It happens when destinations are not reached with the Hamiltonian labels. That is, for example, a multicast packet generated at the switch 9 must be delivered to the destination 5 before destination 2. Otherwise, once the packet arrives at the switch 2, it needs to take the high channel subnetwork to reach the switch 5, thus changing from one subnetwork to the other and introducing a dependency between subnetworks. It is easy to imagine that two packets doing the same (in opposite directions) will create a cycle in the CDG. This is easily solved if the destinations of a multicast packet follow strict Hamiltonian order, thus once an intermediate destination is reached, the next one is reached through the same subnetwork.

The other case for deadlock in path-based multicast is when replicating the packet at a switch. This happens when a packet is both delivered at the local switch and forwarded through an output channel (to move towards its next destination). At that point two branches of a tree have been formed. This may cause deadlock if at a given switch two multicast packets reach the switch and request both channels, and each gets access to only one channel. There is a branch dependency that creates a deadlock situation. This can be solved basically using extra resources to avoid such conflict. In our case, we implement at each switch two ejection channels.

As it is discussed, the implementations of partitioning methods along with the MAR algorithm require two ejection channels but they do not need any virtual channels. However, virtual channels can be used to improve performance (e.g. on top a fully adaptive method [100]).

5.4 Results and Discussion

5.4.1 Analytical Results

We analyzed and compared the unicast latency, the startup latency, and the network latency of the TBP, VBP, and RP partitioning methods using analytical models. For this purpose, the previously presented factors (SP, MML, MxML) are utilized. For each method, we explore the values for two different network sizes along with two different destination numbers, injection rates, and packet lengths. Finally, we estimated the total latency experienced by the packets under different configurations and methods.

5.4.1.1 Startup Latency

We developed formulas to extract the number of startup packets (SP) of the TBP, VBP and RP methods. However, the startup latency not only depends on the SP value but also it is affected by the packet length, injection rate, and the number of destinations per multicast packet. Table 5.1 compares the startup latency of these three methods for different parameters.

A. Effect of the Number of Destinations on the Startup Latency

We computed the upper-bound of the SP value for the TBP, VBP, and RP methods by assuming that there is one packet per partition. The 3^{rd} column of Table 5.1 shows the number of startup packets in the TBP, VBP, and RP methods. However, in reality, the number of packets may be lower than the number of partitions (e.g. when the number of destinations is lower than the sets or destinations are not evenly distributed among sets). We have assumed uniform distribution to find out the probability that a partition has received a packet when there are eight or sixteen destinations per packet. Based on this evaluation, the 4^{th} and 7^{th} columns in Table 5.1 are filled. For example, when there are eight partitions and eight destinations per packet, on average, five partitions include at least a destination and three partitions are empty, thus the average number of startup packets is five. As the number of destinations per packet increases (e.g. from 8 to 16 destinations), with a high probability there are at least one destination per partition. In this case, the startup packets almost reach the upper-bound values.

According to the values in Table 5.1, the RP method offers a lower number of startup packets than the VBP method since partitions are merged together. The average unicast latency for different network sizes is also listed in the 2^{nd} column. As already mentioned, the unicast latencies of different methods are similar. This is due to the fact that unicast packets are routed similarly in the network using the TBP, VBP, and RP methods. Obviously, the unicast latency is increased as the network scales up.

Table 5.1: UL: Unicast Latency; SP: Startup packets; SL: Startup Latency; D/M: Destination per Packet; F/M: Flit per Packet; R: Rate.

Method	1st Size	2nd UL (hop)	3rd SP	4th SP 8 D/M	5th SL 8 D/M 5 F/M 1% R 1st M	6th SL 8 D/M 5 F/M 10% R 100th M	7th SP 16 D/M	8th SL 16 D/M 10 F/M 1% R 1st M	9th SL 16 D/M 10 F/M, 10% R 100th M
TBP	4×4×4	3,75	2	2	5	5	2	10	10
TBP	8×8×8	7,88	2	2	5	5	2	10	10
VBP	4×4×4	3,75	8	5	20	20	7	60	60
VBP	8×8×8	7,88	16	6	25	25	11	100	100+990
RP	4×4×4	3,75	5	4	15	15	5	40	40
RP	8×8×8	7,88	10	5	20	20	8	70	70

B. Effect of Packet Length on the Startup Latency

To show the effect of the packet length on the startup latency, let us assume that a multicast packet (*mul-msg-1*) includes all destinations, and thus only one packet is sent to the network. If there is no contention in the network, the first flit enters the network at cycle 0 (Figure 5.17(a)). However, by partitioning the network, the destinations are distributed among several sets. In this case, multiple copies of *mul-msg-1* (with different sets of destinations) are injected into the network at cycles *0, N, 2N, …* where *N* is the number of flits per packet (Figure 5.17(b)). We compute the startup latency by considering the average packet length as follow:

$$SL_A = (\text{startup packets-1}) * (\text{flits per packet})$$

In Table 5.1, the *4th* and *5th* columns indicate the differences between the startup latencies when the packet size increases from 1 to 5 flits. Similarly, the *7th* and *8th* columns show the startup latencies by changing the packet size from 1 to 10 flits. The values show an increased in the startup latencies when the packet size increases. In all configurations, the TBP method has the lowest startup latency while RP offers lower startup latency than the VBP method.

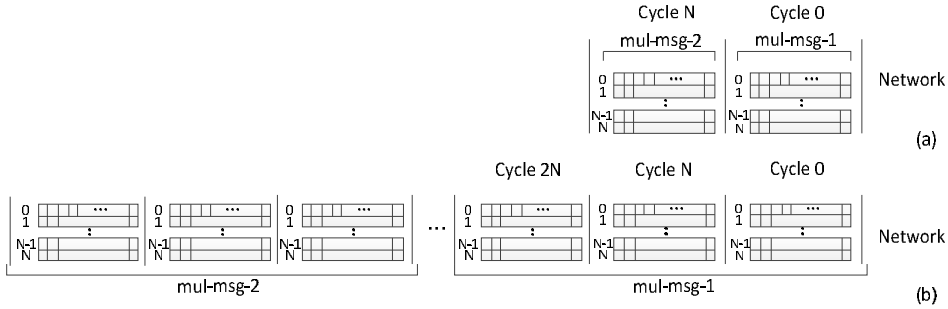

Figure 5.17: Packet length can affect the startup latency

C. Effect of the Injection Rate on the Startup Latency

In a low injection rate, the second multicast packet (*mul-msg-2*) is probably generated by the core when all packets of *mul-msg-1* have already sent to the network. However, in a case of high injection rate, the *mul-msg-2* is ready to be sent to the network while the packets of *mul-msg-1* are still in the queue and have not completely delivered to the network. Therefore, if the number of cycles required for delivering all the packets of a multicast message is larger than (*100 - rate%*), the following formula is obtained: (Latency is cumulative, with each additional generated packet)

$$SL_B = SL_A + (\text{total number of generated packets}-1) * (SL_A - (100 - \text{rate}\%))$$

Table 5.1 also includes the results when the injection rate is taken into consideration. The values are obtained based on two injection rates, 1% and 10%. As can be seen in the 5^{th} and 6^{th} columns (or 8^{th} and 9^{th} columns), in most cases, the startup latencies do not change as *mul-msg-1* has delivered all its packets before *mul-msg-2* is generated. However, in a high injection rate (i.e. 10%), the time required to send startup packets may exceed 90 cycles (100-10). As shown in the 9^{th} column, in one case, it takes more than 90 cycles to deliver startup packets completely to the network. Indeed, the newly generated packets experience considerably larger delays to send their first flit into the network. The values in the 6^{th} and 9^{th} columns are computed for the 100^{th} packet, while in 5^{th} and 8^{th} columns are measured for the first packet.

5.4.1.2 Network Latency

Using analytical formulas, we have estimated the MxML and MML values for TBP, VBP, and RP methods in 4×4×4 and 8×8×8 networks. Since MxML and MML reveal the number of hops, to estimate the network latency the switch delay should be taken into consideration. By assuming 3-stage pipeline architecture, the network latency is computed by multiplying the number of hops with a factor of three. On the other hand, as the injection rate and contention increases, per-hop delay is increased. We assume that in a 10% injection rate, on average, latency increases by three cycles per hop (six cycles in total). According to this assumption, we estimate the total latency using the following formula:

$$Total\ Latency = \begin{cases} MML * 3 + SM & with\ 1\%\ injection\ rate \\ MML * 6 + SM & with\ 10\%\ injection\ rate \end{cases}$$

Table 5.2 shows the total latency (i.e. both network latency and startup latency) in the TBP, VBP, and RP methods considering different parameters. The values in 2^{nd} and 3^{rd} column of Table 5.2 indicate that MxML and MML of the TBP method are considerably larger than those of values in the VBP and RP methods. The VBP method can reduce the MML value significantly at a cost of more startup packets. The 4^{th}, 5^{th}, 6^{th}, and 7^{th} columns show the total latency values when the startup latency is taken into consideration. Since, the high number of startup packets in the VBP method may result in a time-overlapping of different packets, as can be seen in the last column, in some cases the VBP method even behave worse than the TBP method.

Table 5.2: MML, MXML, and total latency in TBP, VBP, and RP methods

Method	1st Size	2nd MxML	3rd MML	4th MML*3+SL 5flits,8dests, 1%rate, 1th packet	5th MML*6+SL 5flits,8dest, 10% rate, 100th packet	6th MML*3+SL 10flits,16dests, 1%rate, 1th packet	7th MML*6+SL 10flits,16dest, 10% rate, 100th packet
TBP	4×4×4	48	21	68	131	73	136
TBP	8×8×8	384	171	518	1031	523	1036
VBP	4×4×4	14	6	38	56	78	96
VBP	8×8×8	51	24	97	169	172	1234
RP	4×4×4	15	7	36	57	61	82
RP	8×8×8	59	26	98	176	148	226

5.4.2 Simulation Results

To assess the efficiency of the proposed partitioning methods in experiment, we have developed a cycle-accurate NoC simulator based on wormhole switching in a 3D mesh configuration. The simulator calculates average latency and power consumption for the packet transmission. The simulator inputs include the array size, the routing algorithm, the link width, the buffer size, and the traffic type. To estimate power consumption of switches, we have used Orion library functions [101]. The power and delay of both horizontal and vertical links are modeled based on the equation in [13]. Finally, we have compared the proposed partitioning methods with each other.

The on-chip network considered for experiment is formed by a typical wormhole-switching structure including input buffers, a routing unit, a switch allocator, and a crossbar. Each switch has 7 input/output ports, a natural extension from a 5-port 2D switch by adding two ports to make connections to the upper and lower layers [10]. There are some other types of 3D switches such as the hybrid switch [10], [14] and MIRA [26], however, since switch efficiency is out of the goals of this thesis, we have chosen a simple 7-port switch in our simulation. The data width and the frequency were set to 64 bits and 1GHz, respectively, and each input channel has a buffer size of five flits with the congestion threshold at 80% of the total buffer capacity.

This congestion threshold is utilized by the presented MAR algorithm to choose the less congested path if there is any alternative path(s). The experiments were performed on a 48-switch (4×4×3) 3D stacked architecture with a constant packet size of five flits. We also assume that the 3D mesh topology is regular and the delays in wires will not exceed the clock period. Two synthetic traffic profiles including uniform and hotspot, and five application benchmarks selected from SPLASH-2 [64] were used to evaluate the partitioning schemes as well as MAR. For the performance metric, we used the multicast latency defined as the number of cycles between the initiation of a multicast message operation, including preparation and startup latency, and the time when the multicast message receives by all destinations. For each load value, the result of packet latency is averaged over 80,000 packets after a warm-up session of 20,000 arrived packets.

5.4.2.1 Performance Evaluation under Multicast Traffic Profile

The first set of simulations is performed for a random traffic profile. In the multicast traffic profile, each core sends a packet to a set of destinations. A uniform distribution is used to construct the destination set of each multicast packet. The number of destinations has been set to eight and sixteen. The average communication delay as a function of the average packet injection rate has been shown in Figure 5.18 and Figure 5.19. As observed from the results, the RP method meets lower delay than the TBP and VBP methods. The foremost reason for this performance gain is due to the efficiency of the RP method which not only reduces the number of hops for multicast packets but also the number of startup packets as much as possible. In fact, TBP suffers from long paths while the performance of VBP degrades due to a large number of startup packets. Adaptive routing algorithms obtain better performance in congested networks due to using alternative routing paths. In Figure 5.20 and Figure 5.21, ARP (Adaptive RP), utilizing MAR in RP, and AVBP (Adaptive VBP), utilizing MAR in VBP, are the adaptive models of the RP and VBP, respectively. As illustrated in this figure, adaptive routings become more advantageous when the injection rate increases.

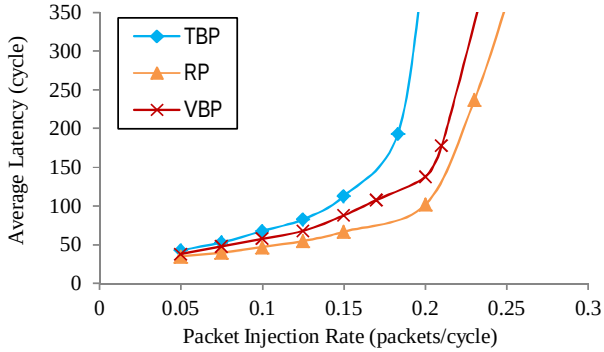

Figure 5.18: Performance analysis in a 4×4×3 mesh network using deterministic routing with 8 destinations

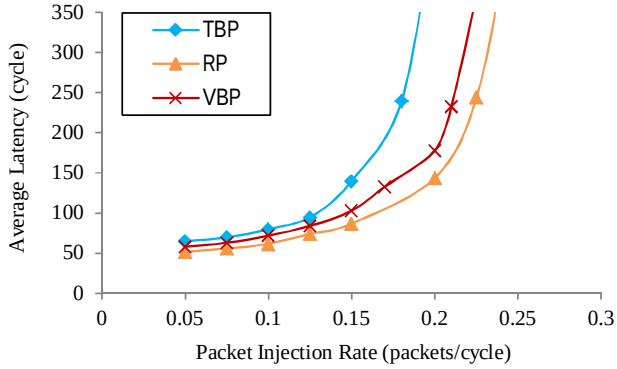

Figure 5.19: Performance analysis in a 4×4×3 mesh network using deterministic routing with 16 destinations

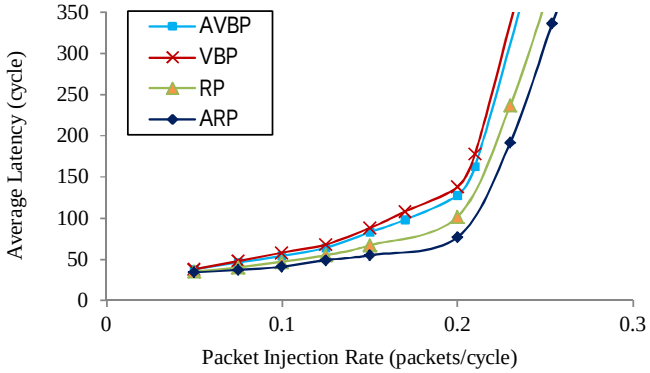

Figure 5.20: Performance analysis in a 4×4×3 mesh network using adaptive routing with 8 destinations

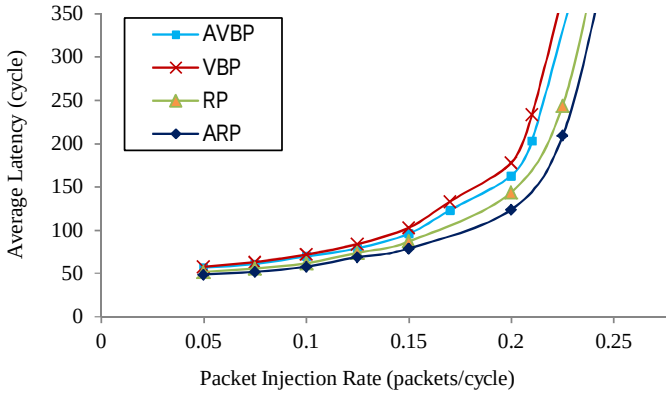

Figure 5.21: Performance analysis in a 4×4×3 mesh network using adaptive routing with 16 destinations

5.4.2.2 Performance Evaluation under Hotspot Traffic Profile

In this set of simulations, we used a mixture of unicast and multicast traffic, where 70% of injected packets are unicast packets and the remaining 30% are multicast packets. The hotspot traffic profile has been taken into account for unicast traffic generation. Under the hotspot traffic pattern, one or more switches are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In the hotspot traffic model, given a hotspot percentage of *H*, a newly generated packet is directed to each hotspot switch with an additional *H* percent probability. We simulate hotspot traffic with a single hotspot switch. The hotspot switch is chosen to be the switch (2,2,2) in a 4×4×3 mesh network. Figure 5.22 and Figure 5.23 shows the performance with *H=10%*. Figure 5.24 and Figure 5.25 indicates that the adaptive routing reduces average latency in comparison with the deterministic routing.

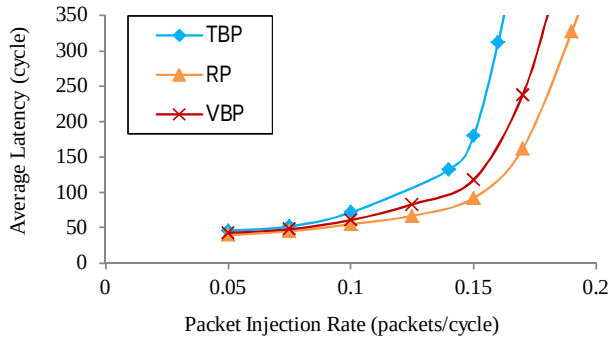

Figure 5.22: Performance analysis in a 4×4×3 mesh network using deterministic routing with 8 destinations under mixed traffic (30% multicast and 70% unicast); Unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and *H=10%*

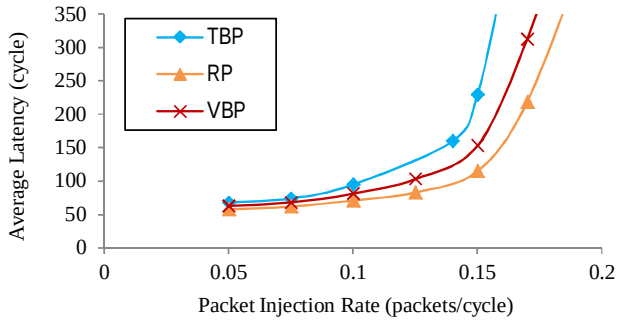

Figure 5.23: Performance analysis in a 4×4×3 mesh network using deterministic routing with 16 destinations under mixed traffic (30% multicast and 70% unicast); Unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and *H=10%*

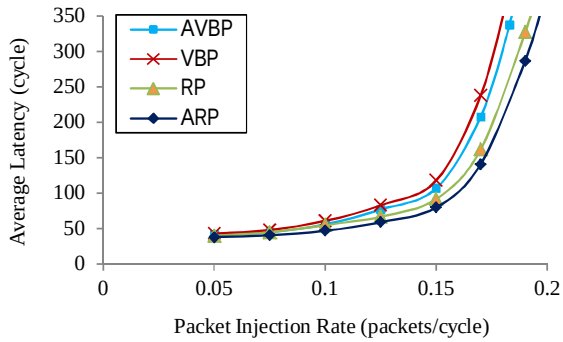

Figure 5.24: Performance analysis in a 4×4×3 mesh network using adaptive routing with 8 destinations under mixed traffic (30% multicast and 70% unicast); Unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and *H=10%*

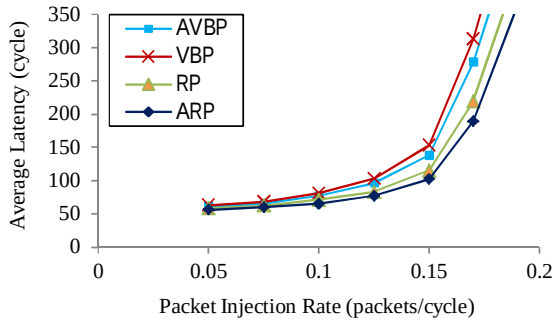

Figure 5.25: Performance analysis in a 4×4×3 mesh network using adaptive routing with 16 destinations under mixed traffic (30% multicast and 70% unicast); Unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and *H=10%*

5.4.2.3 Performance Evaluation under Application Traffic Profile

In order to know the impact of the presented methods on the application traffic profiles, we use traces from some benchmark suites selected from SPLASH-2 [64]. Traces are generated using the GEMS simulator. We used the *Barnes*, *Cholesky*, *fft*, *Ocean*, and *Radix* applications from SPALSH-2 for our simulation. Table 5.3 summarizes our full system configuration where the cache coherence protocol is token-based MOESI and access latency to the L2 cache is derived from the CACTI [102]. We form a 64-node on-chip network (4×4×4) that four layers are stacked on top of each other, i.e. out of the 64 nodes, 16 nodes are processors and other 48 nodes are L2 caches. L2 caches are distributed in the bottom three layers, while all the processors are placed in the top layer close to a heat sink so that the best heat dissipation capability is achieved [26], [103]. For the processors, we assume a core similar to Sun Niagara and use SPARC ISA [59]. Each L2 cache core is 1MB, and thus, the total shared L2 cache is 48MB. The memory hierarchy implemented is governed by a two-level directory cache coherence protocol. Each processor has a private write-back L1 cache (split L1 I and D cache, 64KB, 2-way, 3-cycle access). The L2 cache is shared among all processors and split into banks (48 banks, 1MB each for a total of 48MB, 6-cycle bank access), connected via on-chip switches. The L1/L2 block size is 64B. The simulated memory hierarchy mimics SNUCA [104] while the off-chip memory is a 4GB DRAM with a 260-cycle access time.

Figure 5.26 shows the average network latency normalized to TBP. However, using the adaptive routing scheme, MAR, diminishes the average delay of each partitioning method significantly under all benchmarks. That is, adaptive routing has an opportunity to improve performance. For instance, under the *fft* application, the performance gain of using MAR in TBP, RP, and VBP is about (ATBP/TBP) 7%, (AVBP/VBP) 11.5%, and (ARP/RP) 6%. We can see that ARP, using MAR in the recursive partitioning method, consistently reduces the average network latency across all tested benchmarks. Table 5.4 lists the performance gains of ARP over TBP, ATBP, RP, VBP, and AVBP where the overall performance gain is about 20%.

Table 5.3: System configuration parameters

Processor Configuration	
Instruction set	SPARC, 16 processors
L1 cache	16KB. 4-way associative, 64-bit line, 3-cycle access time
L2 cache	Shared, distributed in 3 layers, unified, 48MB (48 banks, each 1MB)
Cache coherence protocol	Token-based MOESI
Cache hierarchy	SNUCA
Size	4GB DRAM
Access latency	260 cycles
Requests per processor	16 outstanding
Benchmarks	SPLASH-2
Network configuration	
switch scheme	3D mesh with wormhole
Flit size	64 bits
Workloads	
SPLASH-2	Barnes, Cholesky, FFT, Ocean, Radix

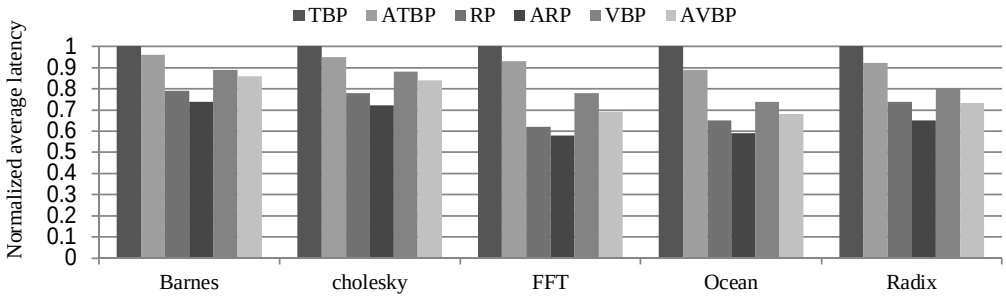

Figure 5.26: Performance analysis under different application benchmarks normalized to TBP

Table 5.4: Performance gain of ARP over the other presented schemes

	TBP	**ATBP**	**RP**	**VBP**	**AVBP**	
Barnes	26%	23%	6%	16%	14%	
Cholesky	28%	24%	7%	18%	14%	
FFT	42%	37%	6%	25%	15%	
Ocean	41%	33%	9%	20%	13%	
Radix	35%	29%	12%	18%	10%	**Overall**
Avg.	34%	29%	8%	19%	13%	20%

5.4.2.4 Hardware Overhead

The presented partitioning methods have been implemented in network interfaces, thereby, to estimate the hardware cost of the proposed methods, the network area of each partitioning scheme, including switches and network interfaces, with the aforementioned configuration were synthesized by Synopsys Design Compiler using the UMC 90nm technology with an operating point of 1GHz and supply voltage of 1V. We performed place-and-route, using Cadence Encounter, to have precise power and area estimations. Depending on the technology and manufacturing process, the pitches of TSVs can range from 1µm to 10µm square [9]. In this work, the pad size for TSVs is assumed to be $5\mu m^2$ with pitch of around $8\mu m^2$. Considering the link width of 64 bits, the area of two unidirectional vertical channels, 2D switch, and 3D switch are $0.01mm^2$, $0.18mm^2$, and $0.23mm^2$, respectively. Therefore, the overhead of adding TSVs in a 3D switch is less than 4%. Different numbers of registers were employed for TBP (the base method), VBP, and RP methods to implement their partitioning mechanisms leading to different area overhead. Comparing the area cost of the TBP with VBP and RP schemes indicate 5% and 6% additional overhead, respectively. However, using the proposed adaptive routing unit (MAR) imposes less than 0.5% overhead for a switch in each method.

5.4.2.5 Power Dissipation

The power dissipation of the TBP, VBP, and RP methods were calculated and compared under the multicast traffic model with sixteen destinations using the simulator based on the Orion and the equation in [13]. The power values of the network interfaces, computed after the place-and-route in the previous subsection, have been also integrated in the Orion functions. The typical clock of 1GHz is applied in the aforementioned network (4×4×3 mesh network). The results for the average power under multicast traffic are shown in Figure 5.27. The average power values are computed near the saturation point under multicast traffic. As the results show, the average power consumption of the RP scheme is 16% and 8% less than that of the TBP and VBP schemes, respectively, when using deterministic routing. In fact, this is achieved by smoothly balancing the traffic over the network using efficient balancing scheme which reduces the number of the hotspots and, hence, lowering the average power.

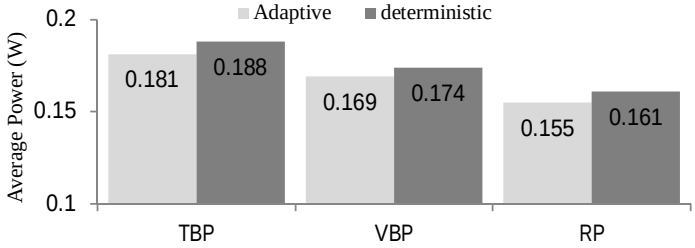

Figure 5.27: Average power dissipation results in a 4×4×3 mesh network under multicast traffic profile

5.5 Summary of the Proposed Methods

The main goal of this chapter is to show how a multicast communication can be efficiently supported without affecting the performance of unicast packets. We presented different partitioning methods (TBP, VBP, and RP) to reduce the overall path length of multicast packets. Each approach offers a different level of efficiency which depends on the number of injected packets into the network per multicast message and the capability to reduce the path lengths. Among these methods, TBP is the simplest approach with the highest latency. VBP may improve latency in low traffic loads; however the large number of startup packets may result in early saturations of the network. RP has the best performance by suggesting the best tradeoff between the number of injected packets and the path length. We equipped these partitioning methods with an efficient adaptive routing algorithm (MAR) for both unicast and multicast packets. Each of the partitioning methods can make benefits of MAR for improving performance. In addition to simulation results, novel analytical models are developed for the latency estimation.

Chapter 6

Conclusion

Technology trends toward an increased number of processing elements with higher levels of integration and performance will require scalable and efficient communication infrastructures. The Network-on-Chip architecture paradigm, based on a modular packet-switched mechanism, can address many of the on-chip communication design issues such as wiring complexity and the integration of a large number of intellectual property cores into a single chip. In the NoC infrastructure, communication protocols perform a critical role in increasing the overall performance of the whole system. This thesis described different routing algorithms for NoCs. The proposed methods include congestion-aware techniques, fault-tolerant routing algorithms, and collective communication support.

Congestion can be avoided by balancing traffic load over a network. This is possible by knowing about the traffic conditions in different regions of the network and routing packets through the less-congested parts. However, distributing the congestion information, storing it at each switch, and keeping it up-to-date is expensive and requires extra wires and registers, which directly affects the area overhead and power consumption. This thesis showed that collecting global congestion information does not necessarily result in a better routing decision. The proposed agent-based approaches confirmed this fact by showing that routing decisions based on the congestion information of nearby switches can result in significant performance gains while the impact of additional congestion information is negligible.

The performance of the network can be improved by propagating packets through less congested non-minimal paths rather than congested minimal routes. However, special care should be taken when choosing a non-minimal path as not all of them lead to better performance. The reason for this is that a packet might take a longer path while passing through a highly congested region, thus exacerbating the traffic condition. This issue has been addressed in this thesis by employing machine learning methods to find an optimal route according to the underlying traffic condition among all available minimal and non-minimal options.

The effectiveness of a routing decision depends not only on how precisely the congestion information represents the traffic condition but also how efficiently this information is utilized in the routing decision. This thesis proposed an efficient way of

utilizing the collected congestion information to make better routing decisions. This efficiency is achieved by employing fuzzy-logic techniques in the routing decision unit.

A high performance system might not be a practical choice if it is highly susceptible to faults. On the other hand, fault-tolerant techniques may not be optimal choices if the performance is considerably degraded by utilizing these methods. The proposed approaches in this thesis fill this gap by presenting fault-tolerant methods which maintain the performance of NoCs in the absence or presence of faults. The main idea of these methods is to diminish the impact of faults on traffic congestion by avoiding unnecessary rerouting of packets around faults. The faults were investigated at both link and switch levels.

An efficient multicast communication can also greatly improve the performance of NoCs. However, multicast support may result in a significant performance loss if the adaptivity of unicast packets is decreased due to this support. In this thesis, we showed how multicast communication can be efficiently implemented without affecting the performance of unicast communication. Unicast/Multicast communication was investigated in both 2D and 3D networks, assisted by analytical models.

References

[1] M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and M. Ould-Khaoua, "An Analytical Performance Model for the Spidergon NoC," in *Proceedings of 21st Annual Conference on Advanced Networking and Applications*, 2007, pp. 1014–1021.

[2] M. Ebrahimi, M. Daneshtalab, N. P. Sreejesh, P. Liljeberg, and H. Tenhunen, "Efficient network interface architecture for network-on-chips," in *Proceedings of NORCHIP*, 2009, pp. 1–4.

[3] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Memory-Efficient On-Chip Network With Adaptive Interfaces," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 1, pp. 146–159, 2012.

[4] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Memory-efficient logic layer communication platform for 3D-stacked memory-on-processor architectures," in *Proceedings of IEEE 3D Systems Integration Conference (3DIC)*, 2012, pp. 1–8.

[5] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "A Low-Latency and Memory-Efficient On-chip Network," in *Proceedings of Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, 2010, pp. 99–106.

[6] Y. Xie, G. H. Loh, B. Black, and K. Bernstein, "Design space exploration for 3D architectures," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 2, no. 2, pp. 65–103, 2006.

[7] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "High-Performance TSV Architecture for 3-D ICs," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2010, pp. 467–468.

[8] A. Y. Weldezion, M. Grange, D. Pamunuwa, Z. Lu, A. Jantsch, R. Weerasekera, and H. Tenhunen, "Scalability of network-on-chip communication architecture for 3-D meshes," in *Proceedings of 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, 2009, pp. 114 –123.

[9] J. Hu, L. Wang, L. Jin, and H. Z. JiangNan, "Electrical modeling and characterization of through silicon vias (TSV)," in *Proceedings of International Conference on Microwave and Millimeter Wave Technology (ICMMT)*, 2012, vol. 2, pp. 1 –4.

[10] B. S. Feero and P. P. Pande, "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32 –45, 2009.

[11] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, "SunFloor 3D: A tool for Networks On Chip topology synthesis for 3D systems on chips," in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2009, pp. 9 –14.

[12] H. Matsutani, M. Koibuchi, and H. Amano, "Tightly-Coupled Multi-Layer Topologies for 3-D NoCs," in *Proceedings of 41st International Conference on Parallel Processing*, 2007.

[13] V. F. Pavlidis and E. G. Friedman, "3-D Topologies for Networks-on-Chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1081 –1090, 2007.

[14] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. K, "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," in *Proceedings of ISCA-33*, 2006, pp. 130–141.

[15] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "High-performance on-chip network platform for memory-on-processor architectures," in *Proceedings of 6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2011, pp. 1–6.

[16] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das, "A novel dimensionally-decomposed router for on-chip communication in 3D architectures," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 138–149, 2007.

[17] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "A systematic reordering mechanism for on-chip networks using efficient congestion-aware method," *Journal of Systems Architecture (JSA-elsevier)*, 2012.

[18] M. Daneshtalab, M. Ebrahimi, J. Plosila, and H. Tenhunen, "CARS: Congestion-Aware Request Scheduler for Network Interfaces in NoC-based Manycore Systems," in *Proceedings of 16th ACM/IEEE Design, Automation, and Test in Europe (DATE)*, 2013, pp. 1048–1052.

[19] M. Daneshtalab, M. Ebrahimi, and J. Plosila, "GLB-Efficient Global Load Balancing method for moderating congestion in on-chip networks," in *Proceedings of 7th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2012, pp. 1–5.

[20] D. Wu, B. M. Al-Hashimi, and M. T. Schmitz, "Improving routing efficiency for network-on-chip through contention-aware input selection," in *Proceedings of Asia and South Pacific Conference on Design Automation*, 2006, pp. 36–41.

[21] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C.-C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook, "TILE64 - Processor: A 64-Core SoC with Mesh Interconnect," in *Proceedings of Solid-State Circuits Conference (ISSCC)*, 2008, pp. 88–598.

[22] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS," in *Proceedings of Solid-State Circuits Conference (ISSCC)*, 2007, pp. 98–589.

[23] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of Design Automation Conference*, 2001, pp. 684 – 689.

[24] A. Jantsch and H. Tenhunen, *Networks on Chip*. Springer, 2003.

[25] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Pipeline-based interlayer bus structure for 3D networks-on-chip," in *2010 15th CSI International Symposium on Computer Architecture and Digital Systems (CADS)*, 2010, pp. 35–41.

[26] D. Park, S. Eachempati, R. Das, A. K. Mishra, Y. Xie, N. Vijaykrishnan, and C. R. Das, "MIRA: A Multi-layered On-Chip Interconnect Router Architecture," in *Proceedings of International Symposium on Computer Architecture*, 2008, pp. 251–261.

[27] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: an engineering approach*. IEEE Computer Society Press, 1997.

[28] M. Mirza-Aghatabar, S. Koohi, S. Hessabi, and M. Pedram, "An Empirical Investigation of Mesh and Torus NoC Topologies Under Different Routing Algorithms and Traffic Models," in *Proceedings of 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD)*, 2007, pp. 19 –26.

[29] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Cluster-based topologies for 3D stacked architectures," in *Proceedings of the 8th ACM International Conference on Computing Frontiers*, 2011, pp. 14:1–14:3.

[30] M. E. Masoud Daneshtalab, "A Novel Interlayer Bus Architecture for Three Dimensional Network-on-Chips."

[31] M. Daneshtalab, M. Ebrahimi, and J. Plosila, "HIBS-Novel inter-layer bus structure for stacked architectures," in *3D Systems Integration Conference (3DIC), 2011 IEEE International*, 2012, pp. 1–7.

[32] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, 1992.

[33] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks*. Morgan Kaufmann, 2003.

[34] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62 –76, 1993.

[35] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2003.

[36] X. Chang, M. Ebrahimi, M. Daneshtalab, T. Westerlund, and J. Plosila, "PARS- An efficient congestion-Aware Routing method for Networks-on-Chip," in *Proceedings of 16th CSI International Symposium on Computer Architecture and Digital Systems (CADS)*, 2012, pp. 166–171.

[37] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Efficient congestion-aware selection method for on-chip networks," in *Proceedings of 6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2011, pp. 1–4.

[38] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proceedings of 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 849 –852.

[39] P. Lotfi-Kamran, A. M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, "EDXY – A low cost congestion-aware routing algorithm for network-on-chips," *Journal of Systems Architecture*, vol. 56, no. 7, pp. 256–264, 2010.

[40] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip," *IEEE Transactions on Computers*, vol. 57, no. 6, pp. 809 –820, 2008.

[41] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, and Z. Navabi, "BARP-A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs," in *Proceedings of Design, Automation and Test in Europe (DATE)*, 2008, pp. 1408 –1413.

[42] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Adaptive Reinforcement Learning Method for Networks-on-Chip," in *Proceedings of SAMOSXIII*, 2012, pp. 236–243.

[43] S. Ma, N. Enright Jerger, and Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *Proceedings of the 38th annual international symposium on Computer architecture*, 2011, pp. 413–424.

[44] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proceedings of IEEE 14th International Symposium on High Performance Computer Architecture, HPCA*, 2008, pp. 203 –214.

[45] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "Performance evaluation of unicast and multicast communication in three-dimensional mesh architectures," in *Proceedings of 15th CSI International Symposium on Computer Architecture and Digital Systems (CADS)*, 2010, pp. 161–162.

[46] M. Daneshtalab, M. Ebrahimi, T. C. Xu, P. Liljeberg, and H. Tenhunen, "A generic adaptive path-based routing method for MPSoCs," *Journal of Systems Architecture*, vol. 57, no. 1, pp. 109–120, Jan. 2011.

[47] M. Daneshtalab, M. Ebrahimi, S. Mohammadi, and A. Afzali-Kusha, "Low-distance path-based multicast routing algorithm for network-on-chips," *IET Computers Digital Techniques*, vol. 3, no. 5, pp. 430–442, 2009.

[48] N. E. Jerger, L.-S. Peh, and M. Lipasti, "Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support," in *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA)*, 2008, vol. 36, pp. 229–240.

[49] C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," in *Proceedings of the 19th Annual International Symposium on Computer Architecture*, 1992, pp. 278 –287.

[50] J. A. Boyan and M. L. Littman, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach," in *Proceedings of Advances in Neural Information Processing Systems 6*, 1994, pp. 671–678.

[51] C. Watkins and P. Dayan, "Technical Note: Q-Learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[52] C. Feng, Z. Lu, A. Jantsch, J. Li, and M. Zhang, "A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip," in

Proceedings of the Third International Workshop on Network on Chip Architectures, 2010, pp. 11–16.

[53] M. K. Puthal, V. Singh, M. S. Gaur, and V. Laxmi, "C-Routing: An adaptive hierarchical NoC routing methodology," in *Proceedings of IEEE/IFIP 19th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, 2011, pp. 392 – 397.

[54] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Agent-based on-chip network using efficient selection method," in *Proceedings of IEEE/IFIP 19th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, 2011, pp. 284–289.

[55] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "CATRA-congestion aware trapezoid-based routing algorithm for on-chip networks," in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, 2012, pp. 320–325.

[56] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Input-Output Selection Based Router for Networks-on-Chip," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2010, pp. 92 –97.

[57] C. J. Glass, C. J. Glass, L. M. Ni, and L. M. Ni, "Maximally Fully Adaptive Routing in 2D Meshes," in *Proceedings of International Conference on Parallel Processing*, 1992, pp. 101–104.

[58] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 92–99, 2005.

[59] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: a 32-way multithreaded Sparc processor," *IEEE Micro*, vol. 25, no. 2, pp. 21 – 29, 2005.

[60] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "LEAR – A Low-Weight and Highly Adaptive Routing Method for Distributing Congestions in On-chip Networks," in *Proceedings of 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2012, pp. 520–524.

[61] X. Dai, C.-K. Li, and A. B. Rad, "An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 285 – 293, 2005.

[62] M. Ebrahimi, M. Daneshtalab, F. Farahnakian, J. Plosila, P. Liljeberg, M. Palesi, and H. Tenhunen, "HARAQ: Congestion-Aware Learning Model for Highly Adaptive Routing Algorithm in On-Chip Networks," in *Proceedings of International Symposium on Networks-on-Chip*, 2012, pp. 19–26.

[63] W. Feng and K. G. Shin, "Impact of selection functions on routing algorithm performance in multicomputer networks," in *Proceedings of the 11th international conference on Supercomputing*, 1997, pp. 132–139.

[64] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *Proceedings of 22nd Annual International Symposium on Computer Architecture*, 1995, pp. 24 –36.

[65] A. Chandramohan, M. V. C. Rao, and M. S. Arumugam, "Two New and Useful Defuzzification Methods Based on Root Mean Square Value," *Journal of Soft Computing*, vol. 10, no. 11, pp. 1047–1059, 2006.

[66] M. Dehyadegari, M. Daneshtalab, M. Ebrahimi, J. Plosila, and S. Mohammadi, "An adaptive fuzzy logic-based routing algorithm for networks-on-chip," in *Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2011, pp. 208 –214.

[67] M. Ebrahimi, H. Tenhunen, and M. Dehyadegari, "Fuzzy-based Adaptive Routing Algorithm for Networks-on-Chip," *Journal of Systems Architecture*, 2013.

[68] S. Rea and D. Pesch, "Multi-metric routing decisions for ad hoc networks using fuzzy logic," in *Proceedings of 1st International Symposium on Wireless Communication Systems*, 2004, pp. 403 – 407.

[69] C. G. B Sun, "Fuzzy Controller Based QoS Routing Algorithm with a Multiclass Scheme for MANET," *International Journal of Computers, Communications & Control*, vol. 4, pp. 427–438, 2009.

[70] M. M. Thaw, "Fuzzy-based multi-constrained quality of service distance vector routing protocol in mobile ad-hoc networks," in *Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE)*, 2010, vol. 3, pp. 429 –433.

[71] A. Pasupuleti, A. V. Mathew, N. Shenoy, and S. A. Dianat, "Fuzzy system for adaptive network routing," *Digital Wireless Communications*, pp. 189–196, 2002.

[72] E. Aboelela and C. Douligeris, "Fuzzy inference system for QoS routing in B-ISDN," in *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering*, 1998, vol. 1, pp. 141 –144.

[73] J. S. R. Jang and N. Gully, *MATLAB Fuzzy Logic Toolbox: User's Guide*. MathWorks, 1997.

[74] W. Pedrycz, "Why triangular membership functions?," *Fuzzy Sets and Systems*, vol. 64, no. 1, pp. 21–30, 1994.

[75] E. B. V. D. Tol and E. G. T. Jaspers, "Mapping of MPEG-4 decoding on a flexible architecture platform," in *Proceedings of Media Processors*, 2002, pp. 1–13.

[76] W.-C. Tsai, D.-Y. Zheng, S.-J. Chen, and Y.-H. Hu, "A Fault-Tolerant NoC Scheme using bidirectional channel," in *Proceedings of 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2011, pp. 918 –923.

[77] M. Cuviello, S. Dey, X. Bai, and Y. Zhao, "Fault modeling and simulation for crosstalk in system-on-chip interconnects," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 1999, pp. 297 –303.

[78] M. Koibuchi, H. Matsutani, H. Amano, and T. Mark Pinkston, "A Lightweight Fault-Tolerant Mechanism for Network-on-Chip," in *Second ACM/IEEE International Symposium on Networks-on-Chip (NoCS)*, 2008, pp. 13 –22.

[79] Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2D-Mesh Network-on-Chip," in *Proceedings of 45th ACM/IEEE Design Automation Conference (DAC)*, 2008, pp. 441 –446.

[80] J. Wu, "A Fault-Tolerant and Deadlock-Free Routing Protocol in 2D Meshes Based on Odd-Even Turn Model," *IEEE Transaction on Computers*, vol. 52, no. 9, pp. 1154–1169, 2003.

[81] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729 –738, 2000.

[82] M. Valinataj, S. Mohammadi, J. Plosila, P. Liljeberg, and H. Tenhunen, "A reconfigurable and adaptive routing method for fault-tolerant mesh-based networks-on-chip," *AEU - International Journal of Electronics and Communications*, vol. 65, no. 7, pp. 630–640, 2011.

[83] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant NoCs," in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2009, pp. 21 –26.

[84] M. Ebrahimi, M. Daneshtalab, J. Plosila, and Mehdipour, Farhad, "MD: Minimal path-based Fault-Tolerant Routing in On-Chip Networks," in *Proceedings of IEEE/ACM 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2013, pp. 35–40.

[85] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen, "MAFA: Adaptive Fault-Tolerant Routing Algorithm for Networks-on-Chip," in *Proceedings of 15th Euromicro Conference on Digital System Design (DSD)*, 2012, pp. 201 –207.

[86] V. Soteriou and L.-S. Peh, "Dynamic power management for power optimization of interconnection networks using on/off links," in *Proceedings of 11th Symposium on High Performance Interconnects*, 2003, pp. 15 – 20.

[87] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "High Performance Fault-Tolerant Routing Algorithm for NoC-based Many-Core Systems," in *Proceedings of 21th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, 2013, pp. 463–469.

[88] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen, "Minimal-Path Fault-Tolerant Approach Using Connection-Retaining Structure in Networks-on-Chip," in *Proceedings of 7th International Symposium on Networks-on-Chip (NOCS)*, 2013.

[89] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICs: a novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 602 –633, 2001.

[90] S. Murali, C. Seiculescu, L. Benini, and G. De Micheli, "Synthesis of networks on chips for 3D systems on chips," in *Proceedings of Design Automation Conference (ASP-DAC)*, 2009, pp. 242 –247.

[91] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "An Adaptive Unicast/Multicast Routing Algorithm for MPSoCs," in *Proceedings of 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools (DSD)*, 2009, pp. 203–206.

[92] M. Ebrahimi, M. Daneshtalab, M. H. Neishaburi, S. Mohammadi, A. Afzali-Kusha, J. Plosila, and H. Tenhunen, “An efficent dynamic multicast routing protocol for distributing traffic in NOCs,” in *Proceedings of DATE*, 2009, pp. 1064–1069.

[93] P. Abad, V. Puente, and J. Gregorio, “MRR: Enabling fully adaptive multicast routing for CMP interconnection networks,” in *Proceedings of IEEE 15th International Symposium on High Performance Computer Architecture (HPCA)*, 2009, pp. 355–366.

[94] X. Lin, P. K. McKinley, and L. M. Ni, “Deadlock-free multicast wormhole routing in 2D mesh multicomputers,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 8, pp. 793–804, 1994.

[95] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, “HAMUM - A Novel Routing Protocol for Unicast and Multicast Traffic in MPSoCs,” in *Proceedings of PDP*, 2010, pp. 525 –532.

[96] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, J. Flich, and H. Tenhunen, “Path-based Partitioning Methods for 3D Networks-on-Chip with Minimal Adaptive Routing,” *IEEE Transactions on Computers*, 2012.

[97] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, “Partitioning methods for unicast/multicast traffic in 3D NoC architecture,” in *Proceedings of IEEE 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2010, pp. 127–132.

[98] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, “Performance Analysis of 3D NoCs Partitioning Methods,” in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2010, pp. 479–480.

[99] M. Ebrahimi, M. Daneshtalab, and J. Plosila, “Fault-Tolerant Routing Algorithm for 3D NoC Using Hamiltonian Path Strategy,” in *Proceedings of 16th ACM/IEEE Design, Automation, and Test in Europe (DATE)*, 2013, pp. 1601–1605.

[100] M. Ebrahimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, and H. Tenhunen, “DyXYZ: Fully Adaptive Routing Algorithm for 3D NoCs,” in *Proceedings of 21th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, 2013, pp. 499–503.

[101] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, “Orion: a power-performance simulator for interconnection networks,” in *Proceedings of 35th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2002, pp. 294 – 305.

[102] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, “Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0,” in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, 2007, pp. 3–14.

[103] I. Loi and L. Benini, “An efficient distributed memory interface for many-core platform with 3D stacked DRAM,” in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2010, pp. 99 –104.

[104] B. M. Beckmann and D. A. Wood, “Managing Wire Delay in Large Chip-Multiprocessor Caches,” in *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, 2004, pp. 319–330.