Technische Universität Dresden

On Fault Resilient Network-on-Chip for Many Core Systems

Sadia Moriam

von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr. sc. techn. habil.
	Dipl. Betriebswissenschaften Frank Ellinger
Gutachter:	Prof. Dr.-Ing. Dr. h.c. Gerhard Fettweis
	Prof. Dr. sc.techn. Andreas Herkersdorf

Tag der Einreichung: 05.04.2018
Tag der Verteidigung: 03.07.2018

Abstract

Rapid scaling of transistor gate sizes has increased the density of on-chip integration and paved the way for heterogeneous many-core systems-on-chip, significantly improving the speed of on-chip processing. The design of the interconnection network of these complex systems is a challenging one and the network-on-chip (NoC) is now the accepted scalable and bandwidth efficient interconnect for multi-processor systems on-chip (MPSoCs). However, the performance enhancements of technology scaling come at the cost of reliability as on-chip components particularly the network-on-chip become increasingly prone to faults. In this thesis, we focus on approaches to deal with the errors caused by such faults. The results of these approaches are obtained not only via time-consuming cycle-accurate simulations but also by analytical approaches, allowing for faster and accurate evaluations, especially for larger networks.

Redundancy is the general approach to deal with faults, the mode of which varies according to the type of fault. For the NoC, there exists a classification of faults into transient, intermittent and permanent faults. Transient faults appear randomly for a few cycles and may be caused by the radiation of particles. Intermittent faults are similar to transient faults, however, differing in the fact that they occur repeatedly at the same location, eventually leading to a permanent fault. Permanent faults by definition are caused by wires and transistors being permanently short or open. Generally, spatial redundancy or the use of redundant components is used for dealing with permanent faults. Temporal redundancy deals with failures by re-execution or by retransmission of data while information redundancy adds redundant information to the data packets allowing for error detection and correction. Temporal and information redundancy methods are useful when dealing with transient and intermittent faults.

In this dissertation, we begin with permanent faults in NoC in the form of faulty links and routers. Our approach for spatial redundancy adds redundant links in the diagonal direction to the standard rectangular mesh topology resulting in the hexagonal and octagonal NoCs. In addition to redundant links, adaptive routing must be used to bypass faulty components. We develop novel fault-tolerant deadlock-free adaptive routing algorithms for these topologies based on the turn model without the use of virtual channels. Our results show that the hexagonal and octagonal NoCs can tolerate all 2-router and 3-router faults, respectively, while the mesh has been shown to tolerate all 1-router faults. To simplify the restricted-turn selection process for achieving deadlock freedom, we devised an approach based on the channel dependency matrix instead of the state-of-the-art Duato's method of observing the channel dependency graph for cycles. The approach is

general and can be used for the turn selection process for any regular topology.

We further use algebraic manipulations of the channel dependency matrix to analytically assess the fault resilience of the adaptive routing algorithms when affected by permanent faults. We present and validate this method for the 2D mesh and hexagonal NoC topologies achieving very high accuracy with a maximum error of 1%. The approach is very general and allows for faster evaluations as compared to the generally used cycle-accurate simulations. In comparison, existing works usually assume a limited number of faults to be able to analytically assess the network reliability. We apply the approach to evaluate the fault resilience of larger NoCs demonstrating the usefulness of the approach especially compared to cycle-accurate simulations.

Finally, we concentrate on temporal and information redundancy techniques to deal with transient and intermittent faults in the router resulting in the dropping and hence loss of packets. Temporal redundancy is applied in the form of ARQ and retransmission of lost packets. Information redundancy is applied by the generation and transmission of redundant linear combinations of packets known as random linear network coding. We develop an analytic model for flexible evaluation of these approaches to determine the network performance parameters such as residual error rates and increased network load. The analytic model allows to evaluate larger NoCs and different topologies and to investigate the advantage of network coding compared to uncoded transmissions. We further extend the work with a small insight to the problem of secure communication over the NoC. Assuming large heterogeneous MPSoCs with components from third parties, the communication is subject to active attacks in the form of packet modification and drops in the NoC routers. Devising approaches to resolve these issues, we again formulate analytic models for their flexible and accurate evaluations, with a maximum estimation error of 7%.

Acknowledgments

This doctoral thesis was written as a result of my research work at the Vodafone Chair Mobile Communications Systems at Technische Universität Dresden during 2013 to 2017.

I would first like to express my sincere gratitude to my supervisor Professor Dr. Gerhard Fettweis for providing the opportunity to join his group and for providing all the support and guidance for this work. He especially provided me with the interesting topic of hexagonal Network on Chip which forms the core of the research. I would also like to thank my group leader Dr. Emil Matus for his all his invaluable advice during my work. Similarly, I am grateful to my group members for providing many interesting and useful feedback to my research during our many meetings and seminars. In particular I would like to thank Dr. Erik Fischer for his countless advises and also Sebastian Haas, who patiently helped during our work on the chip implementations. I am very grateful to Dr. Elke Franz of the Chair for Privacy and Data Security at Technische Universität Dresden for our work together on Network Coding and security for NoC. I would also like to thank Prof. Dr. Akash Kumar, Paul Walther and Prof. Dr. Thorsten Strufe for work on security for NoC.

I would also like to thank my colleagues at the chair for their support over the years. I would like to thank Kathrin and Sylvia, for providing me all the assistance at the chair as well as the IT guys Raffael and Rüdiger whose support was invaluable to running the extensive simulations on the servers.

I am most grateful to my parents without whose unwavering support I would not be here today. I thank hardheartedly my husband and daughter for their patient tolerance and support during these years which allowed me complete this doctoral thesis.

Contents

Abstract **III**

List of Abbreviations **X**

List of Figures **XI**

List of Tables **XV**

1 Introduction **1**

2 Fundamentals of NoC **5**

2.1 Network-on-chip Architecture 5

2.1.1 Topology 5

2.1.2 Routing 7

2.1.2.1 Deadlock avoidance 8

2.1.2.2 Adaptive routing 9

2.2 NoC Reliability Challenge 11

2.2.1 Fault-tolerance techniques 12

2.3 Introduction to Network Coding 14

2.3.1 Random Linear Network Coding 15

2.3.2 Introduction to NoC Security 16

2.4 Conclusion 17

3 Spatial Redundancy and Fault-tolerant Routing **19**

3.1 Introduction 19

3.2 Reconfigurable routing 20

3.2.1 Introduction 20

3.2.2 Related Works on reconfigurable router architecture 21
3.2.3 System Model 22
3.2.4 Reconfigurable Router Architecture 23
3.2.4.1 Synthesis results 26
3.2.5 Application Scenario- Traffic rerouting for fault resilience and link reservation 29
3.3 Fault-tolerant routing in NoC 31
3.4 Application of the turn model 32
3.4.1 Turn model applied to Hex NoC 32
3.4.1.1 Channel Dependency Matrix 33
3.4.1.2 Selecting restricted turns for deadlock freedom 34
3.4.2 Turn model applied to Oct NoC 38
3.5 Fault Tolerant Routing Algorithm 39
3.5.1 Destinations to the NE 40
3.5.1.1 Destinations to the N or E 41
3.5.1.2 Destinations in other directions 43
3.5.1.3 Destinations along the South or West Edge 44
3.6 Performance Evaluation 44
3.6.1 Latency Evaluation 45
3.6.2 Fault Tolerance Evaluation 46
3.7 Summary 49

4 Analytic Assessment of Fault-tolerant Routing 51
4.1 Introduction 51
4.2 Related Works 51
4.3 Adaptive routing and turn model 52
4.4 System Model and Matrix Algebra 53
4.4.1 Connectivity matrix 53
4.4.2 CDM Algebra with adaptive routing 54
4.4.3 Proof of fault tolerance via CDM Algebra 56
4.5 Analytic Model for NoC Routing Connectivity 59
4.5.1 Mesh Negative first fault-tolerant routing algorithm [GN93] 61
4.5.2 Hex Negative first fault-tolerant routing algorithm 64
4.5.3 Performance Evaluation 65
4.6 Analysis of border effect 66
4.7 Summary 69

5 Temporal and Information Redundancy 71

5.1 Introduction . . . 71

5.2 Network Coding in NoC . . . 72

5.2.1 System Model . . . 73

5.2.2 Performance evaluation . . . 74

5.2.2.1 Analytic Model for Evaluating RLNC Performance . . . 75

5.2.2.2 Uncoded Network (UC) . . . 77

5.2.2.3 RLNC coded Network (NC) . . . 79

5.2.3 Results and discussion . . . 80

5.2.4 Applications of the analytic model . . . 84

5.2.4.1 Evaluation of 2D hexagonal and octagonal NoC . . . 84

5.2.4.2 Evaluation of large NoC . . . 86

5.2.5 Summary . . . 87

5.3 NoC Security . . . 87

5.3.1 Security in MPSoCs . . . 88

5.3.2 System and Attacker Model . . . 89

5.3.3 Communication Model . . . 90

5.3.4 Concept for Authentication . . . 91

5.3.4.1 S1: Send the MAC in a Separate Flit . . . 92

5.3.4.2 S2: Include Data and MAC in One Flit . . . 92

5.3.5 Evaluation . . . 93

5.3.5.1 Parameters and Performance Metrics . . . 93

5.3.6 Simulation . . . 94

5.3.6.1 Analytical Model . . . 96

5.3.6.2 Area Overhead . . . 102

5.3.6.3 Summary . . . 103

6 Conclusion and Future Work 105

6.1 Conclusion . . . 105

6.2 Future Research Directions . . . 106

Bibliography 107

List of Abbreviations

2D	Two-dimensional
ARQ	Automatic Repeat Request
avg	Average
CDG	Channel Dependency Graph
CDM	Channel Dependency Matrix
CW	Clockwise
CCW	Counter-clockwise
DSM	Deep sub-micron
FIFO	First-in-first-out
FEC	Forward Error Correction
Flit	Flow Control Digit
GEV	Global encoding vector
hex	hexagonal
MAC	Message Authentication Code
MPSoC	Multi-Processor System-on-Chip
NoC	Network-on-chip
NoCIF	Network-on-chip interface
NC	Network coded
PE	Processing Element
RLNC	Random Linear Network Coding
oct	octagonal
SOTA	State of the art
TC	Transitive closure
UC	Uncoded

List of Figures

2.1 A 3×3 mesh connected NoC illustrating its basic components. 6

2.2 Common NoC topologies. 6

2.3 Dimension ordered routing in the 2D mesh and hexagonal topologies. . . . 7

2.4 Valiant routing: an example of oblivious routing where an intermediate node is randomly selected and then the flit is routed from source to intermediate node and from there to the destination using deterministic routing. 8

2.5 Wormhole switching of flits. 9

2.6 A deadlock involving packets 1, 2, 3 and 4. 9

2.7 Minimal adaptive routing. 10

2.8 The turns allowed in X-Y routing and in the turn model. 11

2.9 Issuing ARQ and retransmission to deal with the loss of a flit due to a transiently failing NoC link. 13

2.10 Butterfly network displaying the concept of network coding. 14

2.11 Network Coded unicast communications in a directed network. 15

2.12 Symmetric key encryption and decryption. 16

2.13 Message authentication by MAC check. 17

3.1 NoC topologies created by addition of diagonal links to the mesh topology. 19

3.2 Different link redundancies in NoC. 20

3.3 Overview of the underlying programming model 23

3.4 Destination regions for 4x4 2D hexagonal NoC and corresponding router LUT for router 10, showing which output port is used for flits destined for the different regions. 24

3.5 Schematic diagram of the programmable router 25

3.6 Examples of flits for reading and writing to the router LUT 26

3.7 Schematic diagram and die photo of Tomahawk4 MPSoC showing hexagonal NoC with programmable routers. 27

3.8 3D mesh off-chip network of Kachel chips with internal NoC consisting of programmable routers. 28
3.9 Area distributions of the centre router of a Hex NoC 29
3.10 Modified LUT when the east output link of router 5 fails. 30
3.11 Residual error rate 30
3.12 Examples of some deadlock cycles forming in the 60 degrees or hexagonal NoC between the three directions. 33
3.13 The channel dependency matrix, A and the Transitive Closure, TC for 2×2 Mesh NoC with the indicated turns restrictions. 34
3.14 Finding CW restricted turns. 35
3.15 Finding CCW restricted turns. 36
3.16 Finding combination of CW, CCW and 180° restricted turns. 36
3.17 A unique set CW restricted turns. 37
3.18 Examples of some deadlock cycles forming in the octagonal NoC between the four directions. 38
3.19 Paths to the destination at NE of the source. 42
3.20 Destinations to the East of the source. 43
3.21 Destinations to the North of the source. 43
3.22 Alternate paths to the destination at SW of the source. 44
3.23 Routing along the west or south Edge. 44
3.24 Latency versus flit acceptance rate for 8×8 NoC with uniform traffic. . . . 46
3.25 Comparison of fault resilience of 8×8 mesh and Hex and Oct NoCs for different numbers of faulty routers. 47
3.26 Comparison of fault resilience of 8×8 mesh and Hex and Oct NoCs with higher number of faults. 47
3.27 Fault resilience comparison for different NoC sizes. 48
3.28 Comparison of fault resilience of different sized Hex NoCs at different ratio of faulty routers. 48

4.1 The channel dependency matrix (CDM), A for 2×2 Mesh NoC with Negative-First routing. 54
4.2 CDM with for 2×2 Mesh NoC with Negative-First routing and a permanently faulty link. 54
4.3 Powers of the CDM, A and the channel connectivity matrix, A_{conn} with Negative First routing. 55

4.4 Effect of faulty router on the channel connectivity Matrix, A_{conn}. 56
4.5 Concept of disjoint paths. 57
4.6 Disjoint paths between s=9 and t=8. 58
4.7 Disjoint paths between s=14 and t=8. 59
4.8 Routing to the NE. 61
4.9 Routing to the S or NW. 63
4.10 Routing to the NE in Hex NoC. 64
4.11 Results of fault resilience vs. percentage faulty routers for mesh NoC in comparison to cycle-accurate simulation. 65
4.12 Results of fault resilience vs. percentage faulty routers for Hex NoC in comparison to cycle-accurate simulation. 66
4.13 Comparison of fault resilience of 16×16 mesh and hex NoC. 67
4.14 Fault resilience excluding faults along the border. 67
4.15 Relation between path length and fault resilience in mesh NoC. 68
4.16 Relation between path length and fault resilience in hex NoC. 69

5.1 Information redundancy in NoC via network coding. 72
5.2 Analytic model results for 8x8 2D mesh with randomly located eight error-prone routers over 1000 iterations of different locations. The plots depict the effect of the flit loss in the routers resulting in ARQs and retransmissions on the average network load, residual error rate, latency and the ratio useful information transmitted. 82
5.3 Variation of ratio of residual error probability of UC and G2C3 and of UC and G3C4 with increasing flit loss probability. 84
5.4 Relative improvements of performance for 8×8 2D hex and 2D oct over the 2D mesh, obtained from analytic model results for the hex and oct NoCs. 85
5.5 Analytic model results for 32x32 2D mesh with 128 error-prone routers. . . 86
5.6 Flit structure for coded and uncoded case with different fields in the header and payload (GEV field is absent in UC case). 89
5.7 Simulation results for 8x8 2D mesh with 8 attacking routers. The plots depict the effect of the flit drop or modification in the attacking routers (resulting in ARQs and retransmissions) on the average network load, residual error rate and the ratio of useful information transmitted. 95

List of Tables

3.1 Area and frequency results of the programmable router 28

3.2 Prevented turns for adaptive routing algorithms in the hex NoC. 37

3.3 Prevented turns for adaptive routing algorithms in the Oct NoC. 39

3.4 Hex Fault Tolerant Negative-First Routing algorithm 42

4.1 Fault tolerant Negative-First Routing algorithm [GN93] 53

4.2 Overview of symbols used in analytic model 60

5.1 Retransmission buffer depth at sender with uncoded transmission for receivers at different path lengths. 75

5.2 Overview of model parameter symbols 76

5.3 Overview of simulation parameters. 81

5.4 Maximum Estimation Error of Analytic Model with iterations over 1000 random fault locations 81

5.5 Example for the authentication of one bit. 93

5.6 Overview of simulation parameters. 94

5.7 Analytic model parameter symbols 96

5.8 Relative error between analytical model and simulation. 101

5.9 Overview of area overhead. 102

Chapter 1

Introduction

The rising power consumption and reduced performance of single core processors has led to the evolution towards multi-processor systems-on-chip (MPSoCs) [Bor07]. Network-on-chip (NoC) has emerged as the dominant solution to the interconnect problem of complex heterogeneous MPSoCs [DMB06] replacing previously used bus based systems. NoCs comprise a high bandwidth packet switched communication network composed of links and routers, connected in any arbitrary topology and with arbitrary routing protocol. Flow control digit or flit is the smallest transmission unit in the NoC. As with any communication network, the challenges are to provide high throughput and low latency as well as providing secure and fault-tolerant communication. The aggressive scaling of transistor gate sizes into the deep sub-micron (DSM) has increased the susceptibility of NoC components to transient and permanent faults, making fault resilience a critical design parameter [RFZJ13]. Transient faults occur randomly lasting for a few cycles only and may be caused by radiation of neutrons and α-particles. Intermittent faults are similar to transient faults occurring in bursts but usually at the same location and eventually leading to a permanent fault. Permanent faults refer to physical shorts and opens in the transistors and wires and may be caused by electromigration and time dependent dielectric breakdowns, processes which are accelerated by process variations.

Redundancy is the general approach to fault tolerance and there are different modes to deal with different fault classes [RFZJ13]. Using redundant links and routers to tolerate failing links and routers is called spatial redundancy. In addition, adaptive routing must be used to be able to bypass faulty links and routers using redundant paths. Using packet retransmission to deal with lost flits or in general the re-execution of a task can deal with the loss of flits due to transient and intermittent errors and is termed temporal redundancy. Finally, another redundancy involves sending redundant information to compensate for flit loss or errors due to bit flips.

State-of-the-art (SOTA) works concentrate on the standard 2D mesh NoC topology, which already provides redundant pathways for flit traversal and devise fault-tolerant routing algorithms with/without the use of virtual channels [IY11, MDP13, VMPL10, FFH09,

Wu03]. In this dissertation, we focus on fault-tolerant routing without the use of virtual channels since they are costly in terms of on-chip area and power consumption. One of the most important features of adaptive routing algorithms is that it should be deadlock free. To prevent deadlock, we use the turn model [GN94], commonly used in many SOTA works [Wu03, GN93, FDC+09]. However, the process of selecting the right combination of prevented turns quickly becomes complicated when we consider topologies other than the mesh such as the hexagonal (hex) and octagonal (oct) topologies which have multiple turns and many possible deadlock cycles in comparison to the relatively simple mesh. To simplify this process, instead of using the general approach of observing the channel dependency graph (CDG)[Dua93, JTWB09], which can again be quite cumbersome, we use powers of the channel dependency matrix (CDM) to obtain the reachability matrix. Using a simple algorithm to loop over the possible prevented turns, we simplify this process to achieve a set of adaptive routing algorithms for the hex and oct NoCs. The approach is very general and can be used to develop adaptive routing algorithms for any regular topology based on the turn model.

Evaluation of fault resilience of adaptive routing algorithms is done via cycle-accurate simulations [MDP13, ZPG07, JTWB09], which can be quite time-consuming especially for larger NoC sizes. To resolve this problem, we further use the algebraic manipulations of the CDM to assess analytically the fault-resilience of the adaptive routing algorithms for the mesh and hex topologies. Analytic assessment of fault-tolerant routing algorithms have been tackled in previous works [VMS08, Val11]. However, in these works a small number of failures is assumed which simplifies the reliability assessment problem considerably. In our analytic assessment approach, we modeled the connectivity of nodes in the 2D mesh and hex NoCs with in the presence of any number of faults and we assume the router has only the knowledge of fault status of its immediate neighbors. The approach is general and be easily adapted to other fault-tolerant routing algorithms.

Finally, we consider the loss of flits in the NoC due to transient and intermittent faults. With rising fault rates in the DSM region, SOTA ARQ (Automatic Repeat Request) and retransmission with/without Forward Error Correction (FEC) techniques used in works such as [PNJ+06, MTV+05] are not sufficient anymore. Therefore we consider the use of other approaches such as network coding.

Scope and Outline of this Work

The dissertation is structured as follows:

- In Chapter 2, we provide the general background on NoC such as interconnection topology, routing, flow control etc. Moreover, a deeper introduction to faults and fault tolerance in NoC is provided. These provide the necessary background for the concepts introduced and investigated in the following chapters.

- In Chapter 3, we investigate spatial redundancy via addition of redundant links to the 2D rectangular mesh topology. We introduce and investigate deadlock-free fault-adaptive routing algorithms for the hexagonal and octagonal topologies. An approach based on matrix algebra of the CDM was used to efficiently generate the different possible routing algorithms based the combinations of prevented turns to avoid deadlock. The investigation of the resulting algorithms show the increase of fault tolerance with spatial redundancy combined with adaptive routing.

- In Chapter 4, the CDM approach is further extended to analytically asses the network connectivity in the presence of permanent faults. The approach is highly general and provides highly accurate results (estimation error of approximate 1%) when validated against cycle-accurate simulations. The approach was used to assess fault resilience of adaptive routing algorithms for the mesh and hexagonal NoCs.

- In Chapter 5, temporal and information redundancy techniques were investigated to compensate for transient flits loss in NoC routers. Fast and highly accurate analytic models were introduced to deduce performance parameters such as residual error rate and network load or gross traffic. The analytic model allowed the investigation of larger NoCs of 1000 cores with relatively fast speed as well as investigation of different topologies. Further, a brief investigation into security in the form of protecting NoC communication against active attacks is also provided in this chapter. The results showed the benefit of network coding to increase throughput and robustness when faced with data modification and loss by malign routers in NoC. An analytic model was further developed for faster and flexible evaluations.

- Chapter 6 provides a summary of the findings of the dissertation and look ahead into future research possibilities.

Chapter 2

Fundamentals of NoC

2.1 Network-on-chip Architecture

The main components of NoCs are the links, routers and network interfaces, as depicted in Fig. 2.1. Links connect the processing elements (PE) or cores to each other via the routers. One or multiple PEs maybe connected to a router via the Network-on-chip interface (NoCIF). The NoCIF receives the data to be sent from the connected core and convert them into the NoC packets flits after the addition of routing information such as the source and target addresses. Flits or flow control digits are the smallest transmission unit in the NoC. The flit is then sent to the router which then determines to which connected router to forward the flit based on the implemented routing algorithm. In the following sections, we describe in greater details the fundamental architecture and concepts of NoC.

2.1.1 Topology

The NoC topology determines how the routers, links and PEs are physically connected to each other in the network and as such has a great effect on the NoC performance. The topology dictates how many routers (or hops) must be crossed by a flit to reach a destination, thereby affecting the average network latency and throughput. Since each hop incurs energy consumption, topology affects the overall network energy consumption. Importantly, topology decides the number of alternate paths between nodes affecting the ability to use different paths to avoid network congestion or faults. Fig. 2.2 shows the common NoC topologies, in addition to the 2D mesh shown in Fig.2.1.

Metrics for comparing topologies A number of factors are used to compare topologies, which are described in the following according to [EJP09].

- Degree The degree defines the number of links at each router connecting it to its neighbors. Looking at the common NoC topologies in Fig. 2.2, the ring has a degree

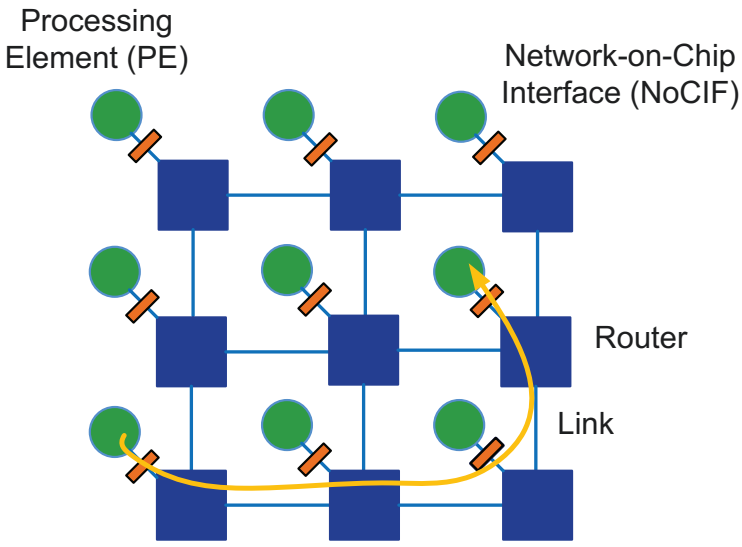


Figure 2.1: A 3×3 mesh connected NoC illustrating its basic components.

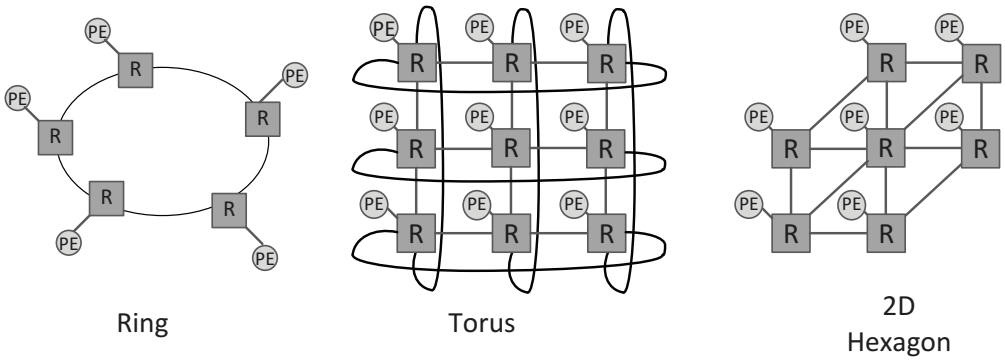


Figure 2.2: Common NoC topologies.

of 2 as it is connected to 2 neighbor routers whereas the torus with its wrap-around links has a degree of 4. The 2D mesh or hexagonal NoC does not have a uniform degree since not all its routers have the same degree. The routers at the corners of the 2D mesh e.g. has a degree of 2 while the edge and center routers have a degree of 3 and 4, respectively.

- Bisection bandwidth The bisection bandwidth is the bandwidth across a cut through the middle of the network and provides an estimation of the cost of the network.

- Hop count Hop count refers to the total number of links that are crossed by a flit from the source to the destination. It provides a useful estimation of the latency

when there is no congestion in the NoC.

- Maximum channel load This metric provides the maximum bandwidth supported by the NoC or the number of flits that can be injected by a node into the NoC before the network saturates. We refer to this metric as the NoC acceptance rate in later chapters such as in Chapter 3.

- Path diversity Path diversity is the number of alternate paths provided by the network between a source and destination node pair. Although, diversity usually refers to the minimum paths, we do not restrict our calculations to minimal paths only since we focus on NoCs with faulty components so that even non-minimal paths count toward the path diversity. A NoC with a higher path diversity is more robust to faulty links and routers and is better able to balance traffic in the network.

2.1.2 Routing

Routing determines the path taken by the flit from the source to the destination and aims to evenly spread out traffic between the available paths in order to reduce latency and increase throughput. There are three main types of routing algorithms: deterministic, oblivious and adaptive [EJP09]. With deterministic routing, the flit will always follow the same path from the source to the destination. A popular example of deterministic routing is dimension-ordered routing (DOR). With DOR, the flit travels to the destination dimension by dimension traveling first along one dimension until reaching the ordinate matching its destination before switching to the next dimension e.g. X-Y routing or the diagonal-X-Y routing, depicted in Fig. 2.3.

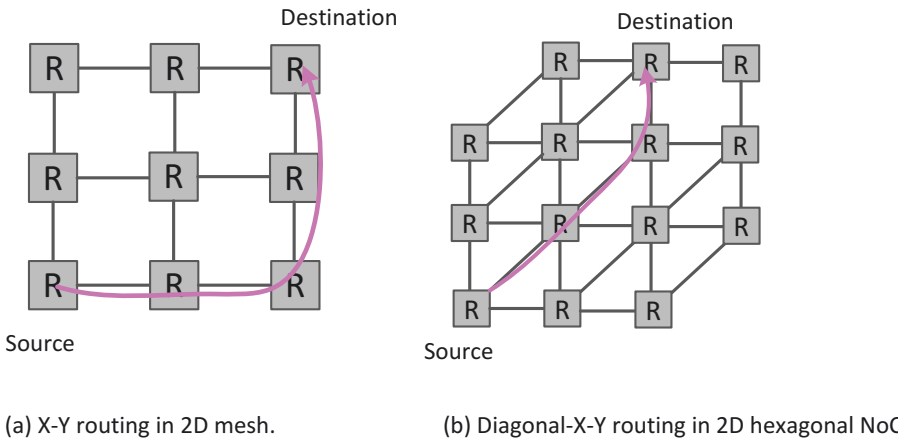


Figure 2.3: Dimension ordered routing in the 2D mesh and hexagonal topologies.

With oblivious routing, different routes are selected by the algorithm in traversing from the source to the destination but network conditions such as faults or congestion is not

taken into consideration. Valiant's routing algorithm is an example of oblivious routing in which the flit is first routed to a random intermediate node (via e.g. X-Y routing) and then routed from there to destination, as illustrated in Fig.2.4. In contrast, adaptive routing takes these into account when choosing between alternate routes to a destination. Adaptive routing is discussed in greater detail in Sec. 2.1.2.2.

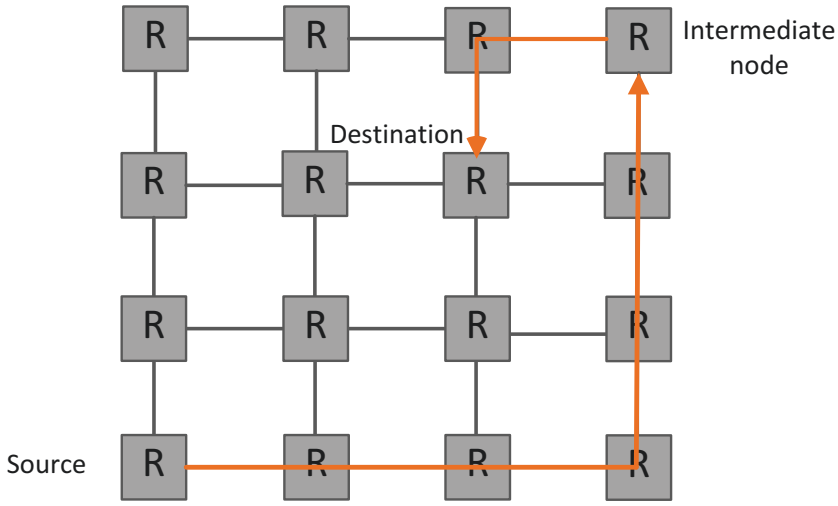


Figure 2.4: Valiant routing: an example of oblivious routing where an intermediate node is randomly selected and then the flit is routed from source to intermediate node and from there to the destination using deterministic routing.

The computation of the route to the destination can be decided entirely at the source, called source routing or decided at each hop, known as distributive routing. In this thesis, we always use distributed routing and limited forms of non-minimal routing in the case of fault-tolerant routing.

2.1.2.1 Deadlock avoidance

NoCs use wormhole switching, in which a bigger packet is broken down into smaller flits before being transported in a pipeline fashion along the route [NM93]. The first or header flit determines the route, reserving the channels as it travels to the destination and the remaining flits follow. The last or tail flit releases a channel as it passes each each channel, so that it may be used by other packets, as depicted in Fig. 2.5. Wormhole switching has the advantage that it requires only small FIFO (first-in-first-out) buffers to store the flits at the intermediate routers. Moreover, once a route has been acquired by a packet, it does not face any contention from other packets.

The disadvantage of wormhole switching is that it makes the routing algorithm particularly prone to deadlock, a situation in which a set of packets may be blocked forever in

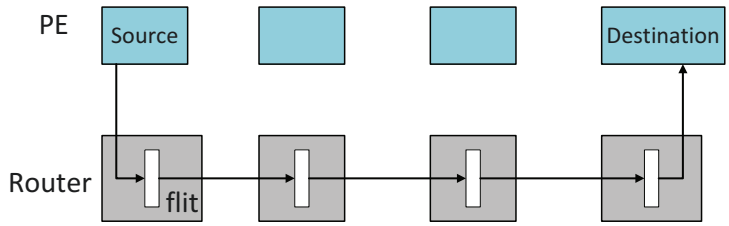


Figure 2.5: Wormhole switching of flits.

the network [NM93]. It occurs when packets holding channels are waiting to get access to some other channels and the waiting-for and requesting happens in a closed cycle. Fig. 2.6 shows an example of a deadlock involving four packets 1, 2, 3 and 4 wanting to eventually reach routers B, A, C, and D respectively, but unable to do so since the channels are in turn held and blocked by packets 2, 3, 4 and 1 respectively. The usual approach to avoiding deadlock is via designing the routing algorithm so that this closed cycle of dependencies are never allowed to occur [DYN03].

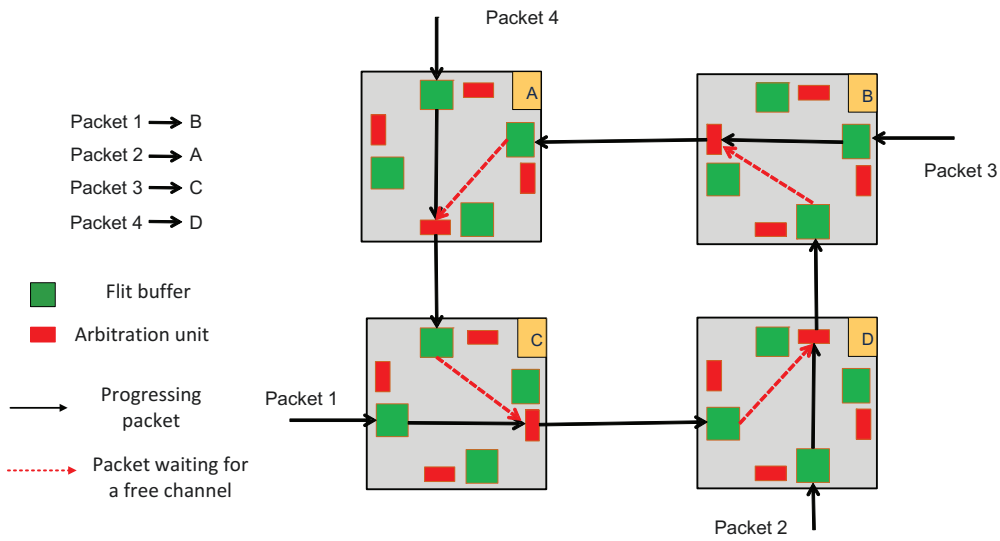


Figure 2.6: A deadlock involving packets 1, 2, 3 and 4.

2.1.2.2 Adaptive routing

As mentioned, adaptive routing takes into account network conditions such as congestion and/or faults in making routing decisions from a source to a destination. Local or global information of the network state can be used such as the buffer occupancy or fault state of neighboring links/routers. Adaptive routing can be minimal or non-minimal. In minimal routing, paths with the minimum hop count are adaptively selected to the destination as shown in Fig. 2.7 whereas in non-minimal routing longer routes may be taken and

is a more robust approach. However, with non-minimal routing there is a possibility of livelock, in which the flit traverses the network without ever reaching the destination. This is managed by the routing algorithm by ensuring a fixed number of mis-routing. The most critical challenge of adaptive routing is avoiding deadlock and one of the common approaches to avoid deadlock in adaptive routing is by using the turn model.

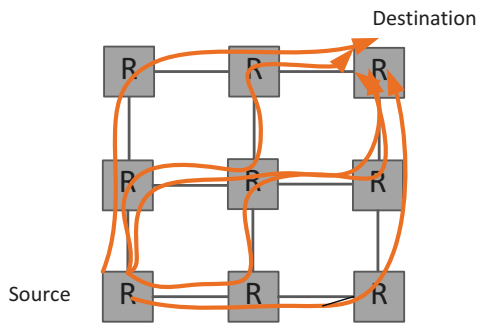


Figure 2.7: Minimal adaptive routing.

Turn model Deadlock is created when due to routing a set of channels are requested in a closed cycle. According to the turn model, a turn occurs when a flit traveling in a certain direction is forwarded to a different direction and a closed cycle of turns can lead to deadlock. In the 2D mesh, there are four turns possible in the clockwise (CW) and counter-clockwise directions (CCW). X-Y routing allows only allows 2 of these turns i.e. the turns from the X-direction to the Y-direction only 2.8(a). The turn model allows 3 of these turns, preventing only one turn in each direction.

As a result, three different adaptive routing algorithms are possible in the 2D mesh, as illustrated in Fig. 2.8(b). Considering that the north-to-west or NW turn is prevented in the CCW direction, preventing the SW, NE or ES turns in the CW direction results in adaptive routing algorithms called the West-First, North-Last or Negative-First adaptive routing algorithms respectively. In West-first, all turns to the west are prohibited. Thus, a packet must start out in that direction in order to travel west. In North-last, all turns from the North are prohibited, so that a flit should only travel north when that is the last direction it needs to travel. In Negative-first, all turns from a positive direction (+X or +Y) to a negative direction are prohibited, so that a packet must start out in a negative direction in order to travel in a negative direction. These are all partially adaptive routing algorithms in the sense that the algorithm does not provide adaptive minimum routes for all source-destination pairs. It is to be noted that preventing the NW and the the WN turns will result in complex cycles and is this not a valid combination of turns.

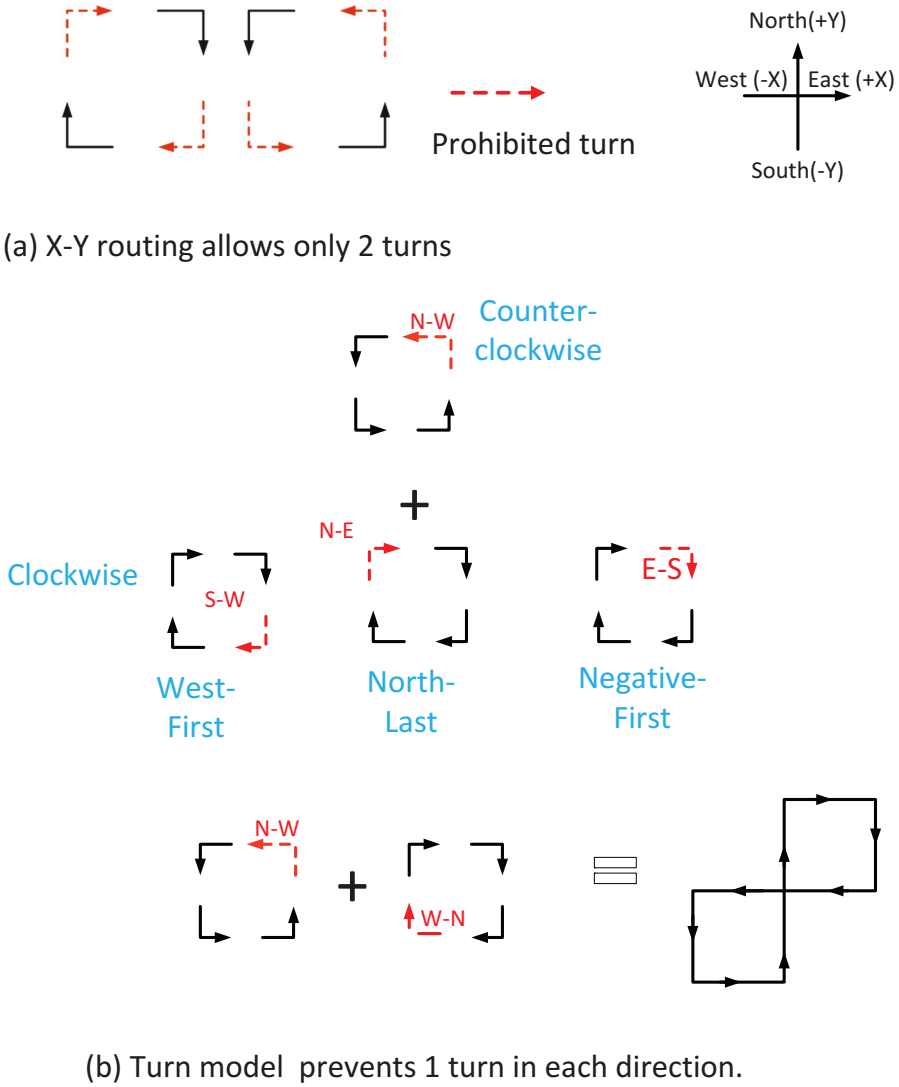


Figure 2.8: The turns allowed in X-Y routing and in the turn model.

2.2 NoC Reliability Challenge

Delving into the deep submicron (DSM) has allowed a higher integration of cores on to a single chip allowing massive parallelization of on-chip processing and thus significantly improving performance. Unfortunately, with smaller transistor gate sizes the chip components including the NoC becomes more susceptible to failures. Heat from high power dissipation accelerates the aging process of NoCs. Moreover, there is a higher susceptibility to radiation and crosstalk effects [RFZJ13]. Thus, technology scaling has made NoC reliability a serious concern for present and future technologies. According to [RFZJ13], a common classification of NoC faults is into transient, intermittent and permanent faults.

- *Transient faults* appear randomly and last for a short period of time causing unexpected changes to data in interconnects or in storage elements (e.g. SRAM and

DRAM). They may be caused by crosstalk coupling between long wires or due to radiation of neutron and alpha particles from impurities in electronic materials. Due to their transient nature, the effect of these faults may be reversed with a re-execution e.g. retransmission of a flit in the NoC.

- *Intermittent faults* are similar to transient faults, however they tend to appear repeatedly at the same location and occurring in bursts. Replacing the faulty elements may remove the intermittent errors. They may be caused by electromagnetic interference and also depend on manufacturing variations. Unless removed, intermittent faults may eventually lead to permanent faults.

- *Permanent faults* can be caused by transistors being permanently shortcut or open or by delay faults resulting from imperfect manufacture and accelerated device wearout from mechanisms such as electromigration, time-dependent dielectric breakdown etc.

The technique to deal with these faults depends on the class of fault which is described in the next section.

2.2.1 Fault-tolerance techniques

Redundancy is the general approach to fault tolerance. The different forms of redundancy [RFZJ13] and their application is described in greater detail in the following paragraphs.

- **Spatial redundancy** is the use of redundant components to deal with the effect of failing ones and is thus appropriate to deal with permanent faults. A popular example is the use of triple modular redundancy (TMR) to have 3 identical copies of a component executing the same function after which the results are compared to find a majority. For the NoC, this might mean e.g. using 3 route computation logic units to determine the route. More commonly, spatial redundancy translates into the use of redundant links and routers and/or routes to deal with the permanent failures of these components. Many topologies such as the 2D mesh inherently provides redundant routes for packet traversal; however, the routing algorithm must be able to use these redundant pathways while taking into account the network fault conditions i.e. it must be adaptive. Although, the network may provide many redundant pathways, the use of many of these may be restricted in order to avoid deadlock. Spatial redundancy, although effective for dealing with permanent faults, entails an overhead in resources and there is often a trade-off in their use and the advantage gained depending on the occurrence of faults.

  Temporal redundancy is time-based redundancy and are best suited to deal with transient and intermittent faults and means re-execution of the process to achieve

a correct result. For the NoC, this could be e.g. used to deal with transiently failing logic in the links and routers causing corruption or dropping of flits. Retransmission of the lost or corrupted flit can resolve the error and has been used in NoC in works such as [MTV+05][PNJ+06]. As shown in Fig. 2.9, when a transiently failing link causes some of the transmitted flits (marked in green) to be lost and the receiver becomes aware of this, an automatic repeat request or ARQ is issued by the receiver to ask for a retransmission. Upon receiving the ARQ, the sender issues a retransmission of the requested flit(s). The process of retransmission will be controlled by mechanisms such as timers in case the ARQ or the retransmission is again lost and the receiver must again issue a further ARQ. ARQ and retransmission can be issued on a hop-by-hop basis between the NoC routers or on a end-to-end basis between the sender and the receiver. As can be expected, the transmission of ARQs and retransmissions increases the network traffic and their use must often be limited in order to reduce the network load.

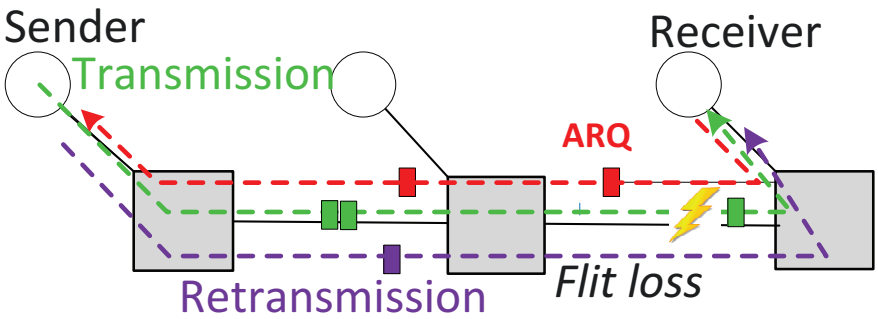


Figure 2.9: Issuing ARQ and retransmission to deal with the loss of a flit due to a transiently failing NoC link.

Information redundancy is the use of redundant bits of information in the flit, such as error detection and correction codes which can be used to recover the original message in case a transient or intermittent error causes the corruption of the flit. Common examples are the use of coding schemes such as single error correction codes or forward error correction codes [DT07]. Information redundancy causes an increase in area and power overhead e.g. due to the increase in flit size as well greater processing and the use of the coding scheme must be traded off against the gain obtained by their use. In Chapter 5 we investigate network coding (explained in greater detail in Sec. 2.3)for NoC as an information redundancy approach to overcome transient and intermittent errors in NoC.

2.3 Introduction to Network Coding

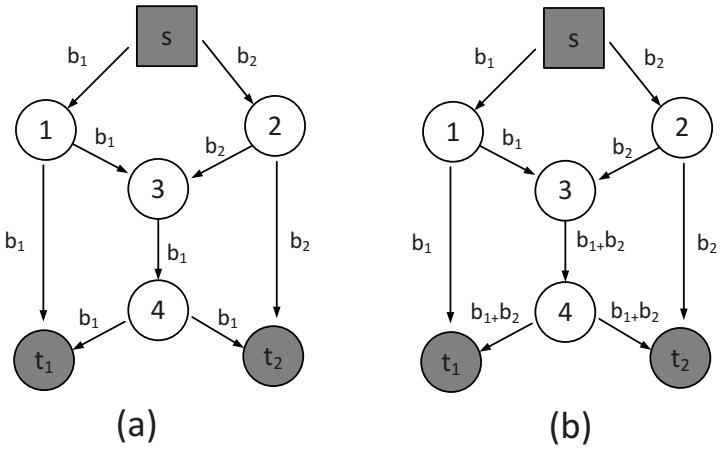


Figure 2.10: Butterfly network displaying the concept of network coding.

In this section we give a brief introduction to network coding as a form of information redundancy, which will be used for overcoming transient and intermitent errors in NoC in Chapter 5. Network coding is the notion of using coding at intermediate nodes in a network [ACLY00][Yeu08]. In the scenario of multicasting information from a source node to a specific group of destination nodes, the advantage of network coding over simple forwarding is explained in the following example. Considering the network in Fig.2.10(a), the source node s generates two messages b_1 and b_2 and wants to send them to two destination nodes, t_1 and t_2. If only routing is allowed, then the central link between node 3 and node 4, would be able to carry only one of the messages but not both. Hence, simple routing would be insufficient since it cannot simultaneously transmit both messages to both destination. Here, if network coding is done, as shown in Fig.2.10(b), both b_1 and b_2 are multicast to t_1 and t_2 by doing a modulo 2 addition of at node 3 $(b_1 + b_2)$ and forwarded to the destinations. At node t_1, b_1 is received and b_2 can be recovered by the modulo2 addition of b_1 and $b_1 + b_2$. Similarly b_2 is received at t_2 and b_1 can be recovered so that in total smaller number of transmissions are needed, thereby increasing system throughput.

The advantage of network coding for unicast can be observed in the simple example of Fig.2.11 [LL04]. Here, there is only one path connecting s_1 to t_1 via $s_1 \rightarrow 1 \rightarrow 2 \rightarrow t_1$ and there is only one oath connecting s_2 to t_2 via $s_2 \rightarrow 1 \rightarrow 2 \rightarrow t_2$. These paths share the link $1 \rightarrow 2$ with a unit link capacity bound and without coding, simple routing would break this bound.

Another advantage gained from network coding is robustness in a lossy environment with packet loss. Here, an FEC can be used at the source to combat the packet. However, with intermediate coding the rate of packets received would be greater than with source coding. For communication security, network coding can be both advantageous. Taking

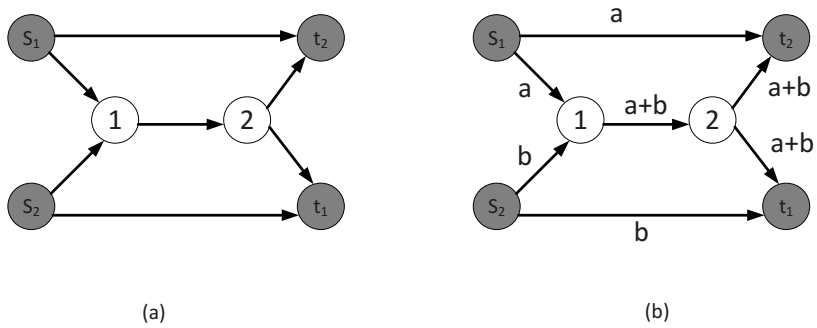


Figure 2.11: Network Coded unicast communications in a directed network.

our previous example of Fig. 2.10, an eavesdropper has greater difficulty getting the information when he receives only $b_1 \oplus b_2$. On the other hand, a malicious node could alter the contents of $b_1 \oplus b_2$ thus sending out false information over the network.

2.3.1 Random Linear Network Coding

In Random Linear Network Coding (RLNC), the sender nodes first organize data packets $\mathbf{x}_i$ into matrices (generations) of size g and sends out a linear combination of the g input packets using randomly chosen coefficients from a finite field[HKM+03] so that the output packets, known as combinations are $y_i = \sum_{j=1}^{g} G_{t,i} \times x_j$:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_g \end{bmatrix} = \begin{bmatrix} G_{t,1} \\ G_{t,2} \\ \vdots \\ G_{t,g} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_g \end{bmatrix} = \mathbf{G_t} \times \begin{bmatrix} x_1 \\ \vdots \\ x_g \end{bmatrix}$$

where $\mathbf{G_t}$ is known as the global encoding vector (GEV) and the receiver can recover the original data packets as long as the matrix $\mathbf{G_t}$ has rank g. Chou et al. introduced a practical implementation of this approach that allows for a decentralized solution [CWJ03], where no knowledge of network topology or the encoding and decoding schemes is necessary. This is accomplished by including the GEV within each packet by prepending each packet x_i with the ith unit vector, so that by multiplication with G_t, GEV is included within each combination:

$$G_t \times \begin{bmatrix} 1 & \cdots & 0 & x_1 \\ \vdots & \ddots & \vdots & \\ 0 & \cdots & 1 & x_g \end{bmatrix} = \begin{bmatrix} G_{t,1} & y_1 \\ G_{t,2} & y_2 \\ \vdots & \\ G_{t,g} & y_g \end{bmatrix}$$

The receiver can decode the received combinations using the GEV, only he must wait to receive g number of packets. For the receiver to distinguish which packets belong to a generation, each combination of the same generation must be tagged with a generation

identifier. In a lossy environment, if a packet is lost, the receiver may not have 'G' number of packets for decoding. To combat this situation, redundant combinations are sent to the receiver so that if he can receive at least G combinations, he can still decode the generation. This is the form of RLNC with redundant combinations that we will focus on in Chap. 5. Thus with RLNC there is information redundancy within each packet and within a generation in the form of redundant combinations.

2.3.2 Introduction to NoC Security

In this section, we give a short introduction to concepts in information security, which is a part of our investigations in Chapter 5. Keeping information secure has become an integral and necessary part of our our daily life. Our activities involve using computers for simple everyday tasks as buying goods online or checking our bank balances online or using our smart phones to check our emails, keeping track of our health activities etc. It is only natural that we would want to prevent unauthorized access or modification to our information. The three fundamental concepts of security are **confidentiality**, **integrity** and **availability** [And11]. Confidentiality means keeping information secure from unauthorized access whereas integrity is the ability of preventing data being modified or even deleted in an undesirable manner. The final requirement of security i.e. availability refers to the ability to access stored information by authorized users and applications. The three goals of security can be threatened by attacks. Snooping or eavesdropping threaten confidentiality while any form of modification, replaying or masquerading threatens the integrity of data. Denial of service attacks by e.g. by flooding the network with redundant messages can affect availability by interrupting and slowing down the system. Security measures must be able to protect data by preventing these attacks.

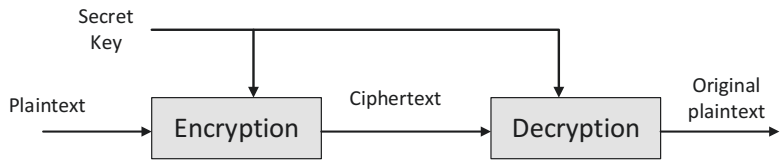


Figure 2.12: Symmetric key encryption and decryption.

Encryption is the process of encoding information called plain text so that it cannot be understood by any interceptor and thus provided confidentiality. Decryption is the reverse process. The encryption and decryption algorithm may involve a shared secret key between the sender and receiver and is called symmetric encryption, shown in Fig.2.12. Examples of symmetric ciphers are the DES(Data Encryption Standard) and AES(Advanced Encryption Standard) algorithms. Message authentication provided the guarantee of message integrity i.e. it has not been modified. For this a message authentication code (MAC) is generated using a symmetric key cryptographic algorithm at the sender and used to tag

the message. The receiver runs the same cryptographic algorithm and then checks its generated MAC against the received MAC to ensure that the message has not been modified by a third party, as depicted in Fig.2.13.

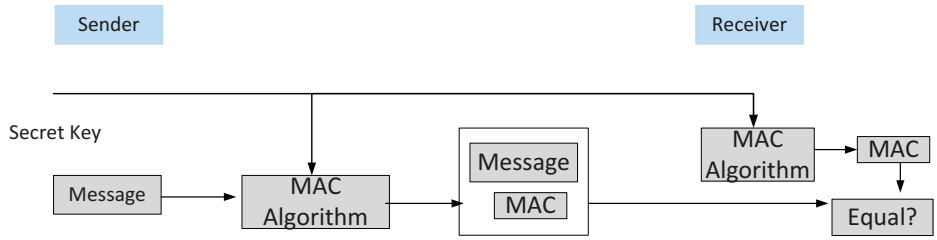


Figure 2.13: Message authentication by MAC check.

2.4 Conclusion

In this chapter, we have gone through the fundamental concepts of NoC including its architecture and basic working concepts in order to lay a foundation for our investigations in the next chapters. In summary we can conclude the following points:

- NoC is a highly scalable and bandwidth efficient network for MPSoCs providing a solution to the interconnect problem. Topology, routing, fault resilience are some of the challenges of NoC design.
- Adaptive routing is an important concept for NoC which allows for dynamically coping with network conditions such as faults and congestion. However, wormhole switching makes adaptive routing prone to deadlock and so design of adaptive routing algorithms must be done carefully to avoid deadlock.
- Technology scaling has significantly increased the density of on-chip integrations and enhanced the performance of MPSoCs but has also increased its susceptibility to faults. For the NoC, there are different classes of faults according to their duration: transient, intermittent and permanent. Different fault tolerance techniques are necessary to deal with each class of fault.
- Network coding is an information redundancy technique to increase robustness and efficiency of NoC communication.

Chapter 3

Spatial Redundancy and Fault-tolerant Routing

3.1 Introduction

In this chapter, we concentrate on permanent failures of NoC components and using spatial redundancy i.e. usage of redundant components such as extra links and routers to tolerate these faults, explained in Sec: 2.2.1. According to [Bar64], redundancy level is a measure of the network connectivity since it leads to a greater path diversity. Baran [Bar64] defines redundancy as the link-to-node ratio in network of infinite size. The most commonly used NoC topology is the rectangular 2D mesh NoC and this already provides redundant path ways for the flit traversal. We add further redundant links along one diagonal direction to create the hexagonal (hex) NoC [MF16] and along both diagonal directions to create the octagonal (oct) NoC, as shown in Fig. 3.1. Our goal is find out increased path diversity for the hex and oct NoCs. Depending on how the redundant links are connected, different topologies can result, as shown in Fig. 3.2. However, we limit our analysis to the hexagonal and octagonal NoCs only.

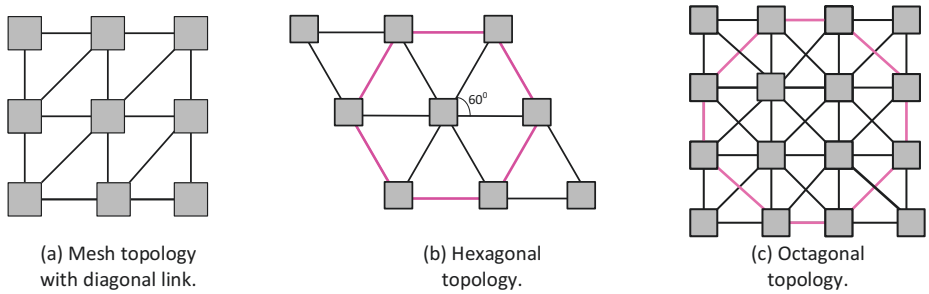


Figure 3.1: NoC topologies created by addition of diagonal links to the mesh topology.

In addition to the redundant links (approximately 50% for the Hex and 100% for the Oct), adaptive routing must be used in order to by-pass faulty components. Routing in NoC involves using wormhole switching (described in 2.1.2.1) in which several flits make up a packet and the first or the header flit determines the path, reserving the channels as it progresses forward. The remaining flits follow the header in a pipeline manner and the tail (last) flit releases the channels. The advantage gained is reduced latency as well as reduced buffer sizes[NM93] but the disadvantage is a greater susceptibility to deadlock. Thus we must first develop deadlock-free fault-tolerant routing algorithms for the hex and oct NoC in order to investigate the reliability of these topologies, which are described in the following sections.However, before proceeding to fault-tolerant routing, we first look to reconfigurable routing as an initial approach to reconfigure routes around faults in the NoC as a way to achieve fault resilience.

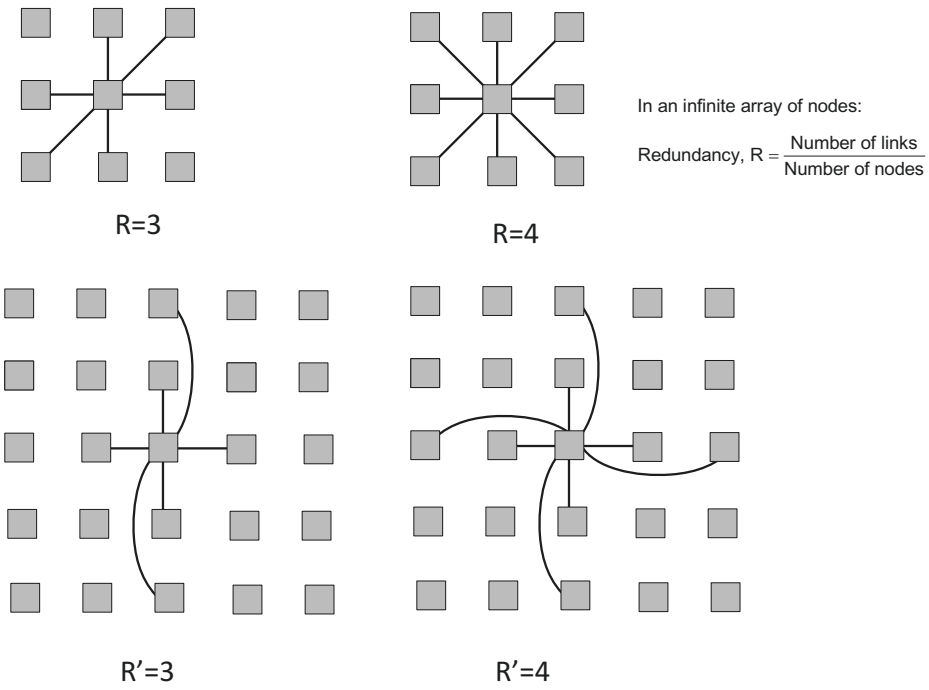


Figure 3.2: Different link redundancies in NoC.

3.2 Reconfigurable routing

3.2.1 Introduction

In this section, we describe a flexible router architecture that allows a central managing core to access any router configuration register or any router state register. It also allows to reconfigure these registers as necessary, as for example to program the routing table of the router, allowing to flexibly change the routing. There could be many reasons for

requiring to change the NoC communication scheme after design and manufacture. One example is due to the mapping of applications on to the NoC, which has an effect on the throughput and latency of the system. The efficient mapping is a challenging task which has been tackled in many works [MCRG06, HM05]. In some of of these works, it has also been argued that the same NoC architecture may be used for multiple-use cases [HM05], which may not be possible to check all at design time. In such cases, it may happen that the mapping and communication via the NoC with the existing routing scheme may not be the optimal one. In such a scenario, it is highly advantageous if the routing of the flits across the NoC can be reconfigured dynamically to better suit the application traffic scenario. Moreover, with continuous technology scaling and with new innovative technologies try to tackle the end of Moore's law [cfa17], the technologies are getting more error-prone, showing also new types of errors and thus require special algorithms and architectures to provide stable and reliable systems [RFZJ13]. For this purpose, we need to include resiliency mechanisms on both sides: the processing and the communication. While adaptive routing is usually followed in such a faulty NoC scenario, this may incur hardware overhead if the fault rate is low. Deterministic routing is simple and less costly and will often provide the most optimal performance. Having routers with reconfigurable routing, it is possible to flexibly change the routing scheme to avoid permanent or intermittent faults in the NoC links and routers.

3.2.2 Related Works on reconfigurable router architecture

The concept for reconfigurable NoC for MPSoCs has gained popularity and been investigated for the advantages of possible reduction of design time and flexible changing of characteristics with changing applications. Authors in [DPCD09] have presented R2NoC, a fat-tree based NoC comprised of dynamically reconfigurable routers, which are defined as partially reconfigurable regions where dynamic communication links are implemented. In [KdlTR10], a reconfigurable communication approach called DRNoC is presented. Their work focuses on solutions oriented to reconfigurable systems based on FPGAs and allows to define a broad set of communication strategies, point to point and point to mulitpoint. In [CMC14], authors present DyAFNoC (dynamic automatic flexible direction order routing or FDOR based NoC). They adopt a logic-based implementation of FDOR to automatically adapt the routing of packets to new context topologies. Therein each router receives a signal to determine which routing mechanism is used (XY or YX) and is based on implementation of the two non overlapping routing algorithms. Bahrebar et. al. [BS16] present a routing algorithm based on the Abacus Turn Model, in which healthy parts of a 2D mesh NoC can be dynamically reconfigured according to the location of faults and congestion in the network, without the use of virtual channels or routing tables. In [FDC+09], authors present a highly resilient routing algorithm for 2D mesh and torus NoCs by reconfiguring the NoC to avoid faulty components. The routers area table-based and consist of entries for all destinations of the network. The algorithms consits of multiple iterations in which

routers communicate each other via flags to update entries in the tables to avoid faulty components. Another work based on updating table entries between routers is presented in [SZBR07], where the authors present a mechanism to exchange information between network switches, which is derived from distance-vector routing. During initialization, all switches build up a routing table of the entire NoC and each switch send information to neighboring routers containing its own address and the number of hops to destination to its neighbors. Authors in [FLJ+13], present a buffer-less fault-tolerant router to address transient and permanent faults in NoC. They also use router tables, which are however, reduced in size in comparison to the work in [SZBR07]. Also in contrast to [SZBR07], which require additional packets to update the routing table, the work [FLJ+13] in the information is transmitted with hard wires. Our approach is different from previous works in that we design the router not only for programmable routing but also getting access to and reconfiguring, if necessary other registers in the NoC routers. This allows us to have an insight into the processes inside a NoC/router implemented on a chip.

3.2.3 System Model

Before we proceed to the details of the reconfigurable router, it is essential to get some background about the underlying multi-processor programming model that is assumed in this work. An overview of the programming model is depicted in Figure 3.3. We assume a central unit (Core Manager), which is responsible for the task scheduling, i.e. distributing the tasks onto the PEs [AMN+14]. When a task is scheduled for a certain PE, a local direct memory access (DMA) controller transfers the program and input data from the global memory into the local PE memory. After the task execution has been finished, the result is either transferred back to the global memory or remains in the local memory as input for the next scheduled task to exploit data locality [N+14]. Furthermore, the result may directly be transferred to a different PE via DMA to realize a pipeline type of processing. Finally, it shall be possible to share the local memory of adjacent PEs (memory stealing) to dynamically scale the memory size as required or realize some kind of shared memory task communication model [H+17]. For this purpose, it is required to reserve the connecting link as a dedicated resource. This is just one application aspect where the configurable router, described in Sec. 3.2 comes into play.

Routers in NoC generally use distributed routing, i.e. the decision of routing is taken in each router along the route from the source to the destination, in contrast to source routing, in which the entire path is specified at the source. The routing algorithm inside the router decides the path of the flit from the source to the destination. The routing algorithm can be deterministic or adaptive and can be implemented as combinational logic or as decision tables (look-up table or LUT) inside the router [EJP09], in which an output port is given for each destination from the current router. The advantage of distributed routing is that it allows for adaptivity in routing according to the network conditions such as to avoid congestion or to bypass faulty links/routers.

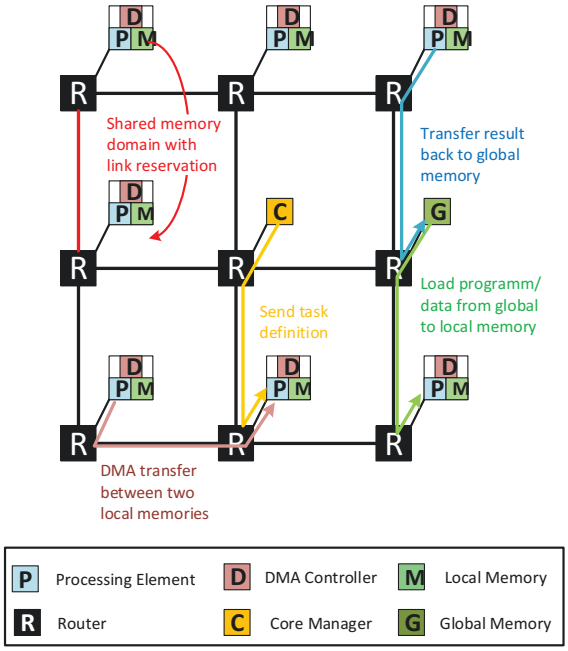
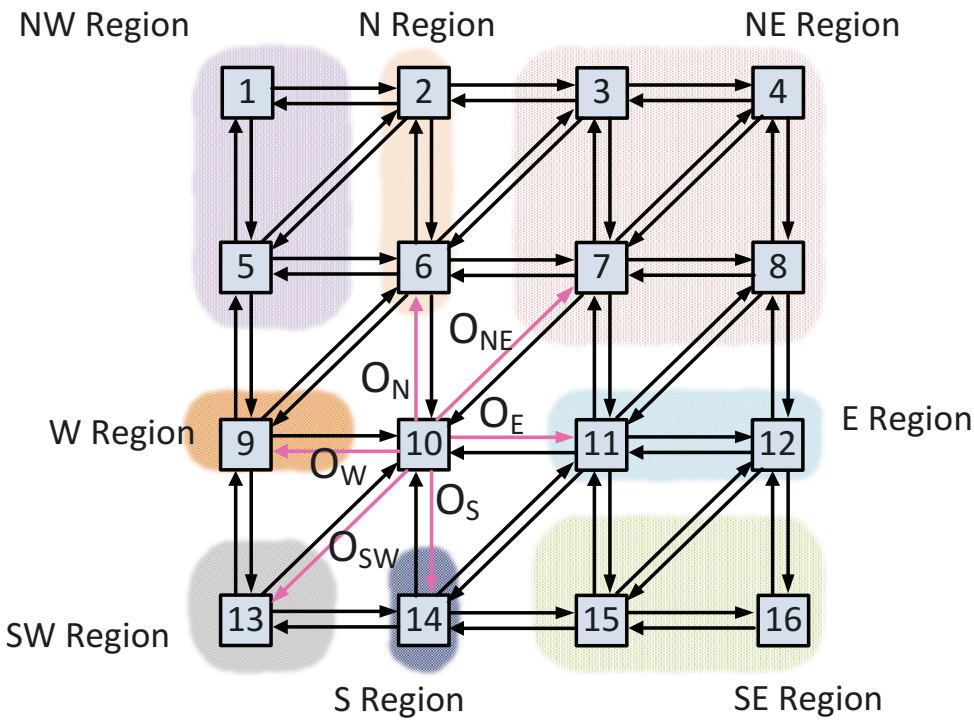


Figure 3.3: Overview of the underlying programming model

The advantage of using router tables is that they can be changed dynamically during runtime after design and production. The downside, however, is the problem of scalability. For larger NoCs, the tables can become very big and therefore consume a lot of area and power at each router. To prevent this problem, we have router tables that are configured as region based similar to the approach proposed in [FMLD07]. In our case, for simplicity the regions are defined for the 8 possible destination directions i.e. to the N, NE, E, SE, S, SW, W and NE directions. However, this is an assumption on our part and the number and size of regions can be varied in many ways. Considering a center router, there are 4 possible output ports for the mesh and 6 possible output ports for the hexagonal NoC, resulting in table sizes of 4×8 or 32 bits and 6×8 or 48 bits respectively. The tables are implemented as register arrays. An example of the table for the case of dimension ordered diagonal-x-y routing 2.1.2 over the hexagonal NoC is shown in Fig. 3.4. As can be seen here, each output port is used to reach destinations in a specific direction. If, for example, for a flit arriving at router 10, the destination is to the South East or East of the current router, the flit will be forwarded to the east output port, O_E(Fig. 3.4(b)) .

3.2.4 Reconfigurable Router Architecture

In our reconfigurable router, in addition to the existing link and routing logic, a small dummy module called the router module has been added. This module allows the flexible access to any registers inside the router such as the router LUT. The router module has an ID like any other PE in the NoC and can receive and send flits in response to requests.



(a) 4×4 NoC with the different routing regions.

Relative direction of target wrt to current router (target regions)	Router output Port					
	O_W	O_{SW}	O_S	O_E	O_{NE}	O_N
W	1	0	0	0	0	0
SW	0	1	0	0	0	0
S	0	0	1	0	0	0
SE	0	0	0	1	0	0
E	0	0	0	1	0	0
NE	0	0	0	0	1	0
N	0	0	0	0	0	1
NW	1	0	0	0	0	0

Use O_E to reach destinations to the East and South East

Use O_N to reach destinations to the North

(b) Routing look-up table (LUT) for center router, 10.

Figure 3.4: Destination regions for 4x4 2D hexagonal NoC and corresponding router LUT for router 10, showing which output port is used for flits destined for the different regions.

It receives the reconfiguration flits and makes the corresponding changes to the specified register of the router. Only the central core manager is allowed to be able to access the router registers. This is controlled by a special register called the mask register inside the router module, where the ID of the controlling core is stored. At the boot up of the chip, the core manager sets this mask_id to its ID. Hence forth, only the core manager can access the router LUTs for reading or rewriting. This feature is necessary for security

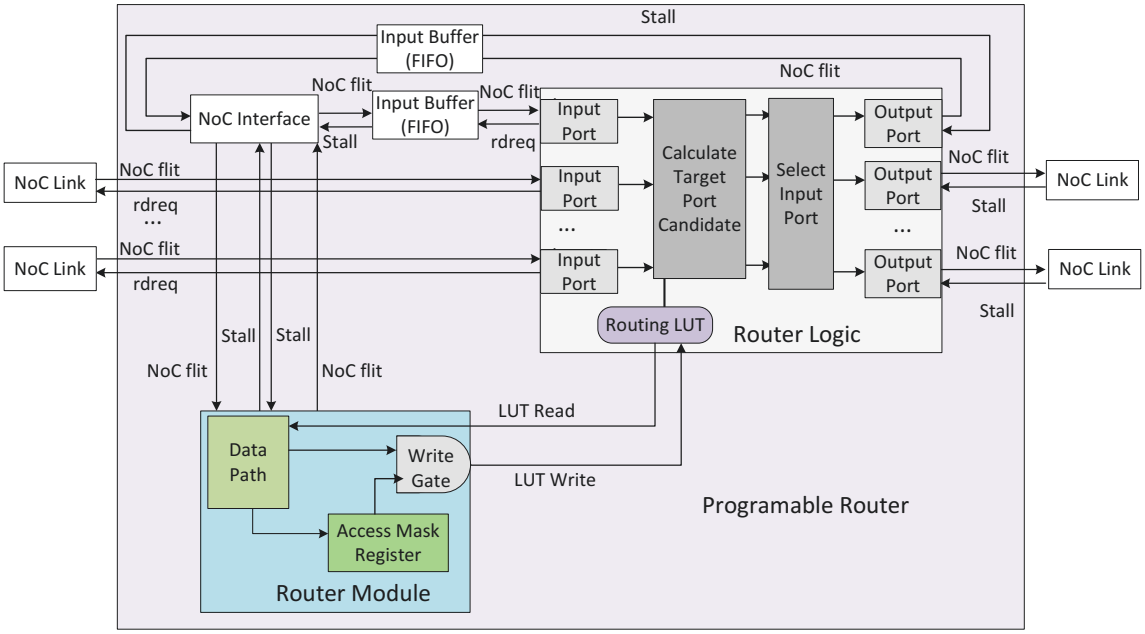


Figure 3.5: Schematic diagram of the programmable router

reasons so that any module cannot change the router LUTs.

Normal flits of the NoC are used to read or write specific registers of the router via the router module. Like any other node or module of the NoC, the router module of each router also has an ID but with the same address as the router itself, so that whenever the core manager sends a configuration flit to the NoC router, the ID of the target router module is specified. The payload of the flit also has three fields: the 'mode', the 'address' and the data. Normally, the address field is used to indicate the address of the memory where a read or write action will be performed. The mode field is used to indicate the type of flit e.g. whether the flit is a read-request from memory or a write to the specified address. For flits sent to the router module, the mode field is used to specify whether a read or write is requested of a certain register in the router. The address field is used to indicate which register should be read or modified. For example, it is possible to read or modify the LUT of all the input ports of the router or that of a certain input port. When a register should be modified such as the router LUT, the value of the LUT in bit serialized form is sent in the data part of the flit. The data field of the payload has 64 bits and so the maximum routing table size of 48 or 32 bits can be easily fitted into this field. However, if the size of the routing table is larger than the length of the data field, it is easily possible to divide it over multiple flits and send them subsequently one after another.

Examples of flit structures to read and and write a specified LUT of a router are shown in Fig. 3.6. In ths case, it is the centre router and reconfigurable flit either contains the new LUT or in read-mode request to read the LUT at certain input port. As response to the read request, the router module sends the response flit with the requested LUT in

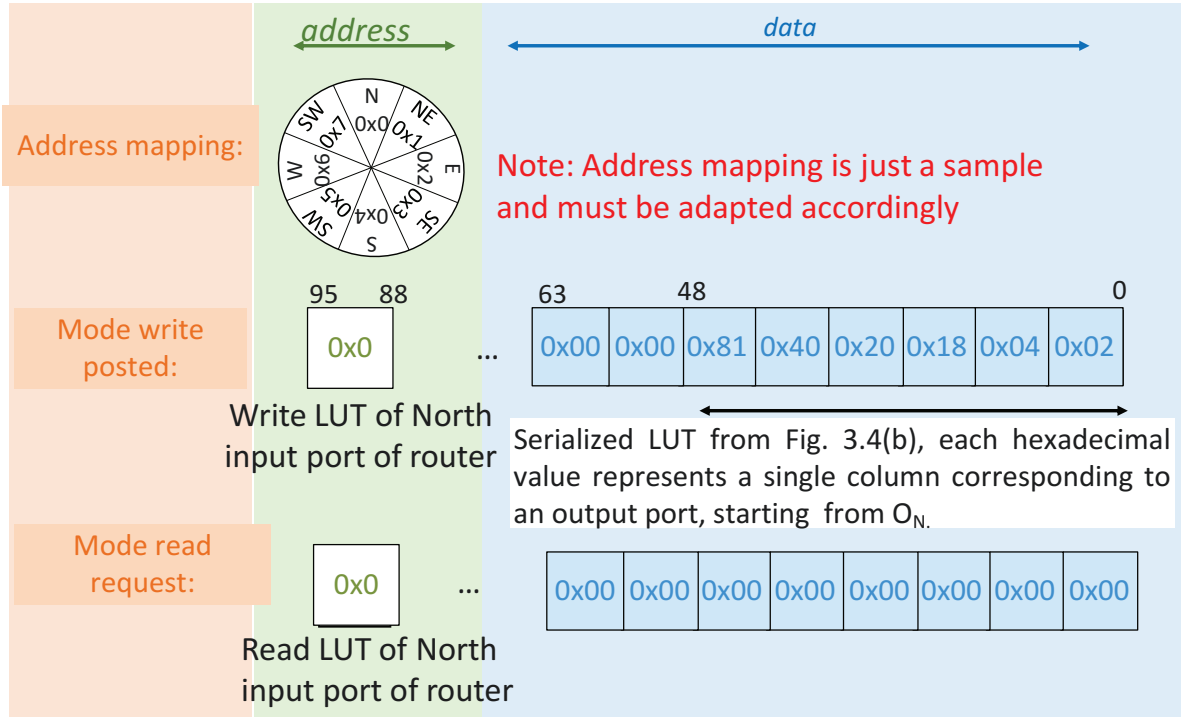


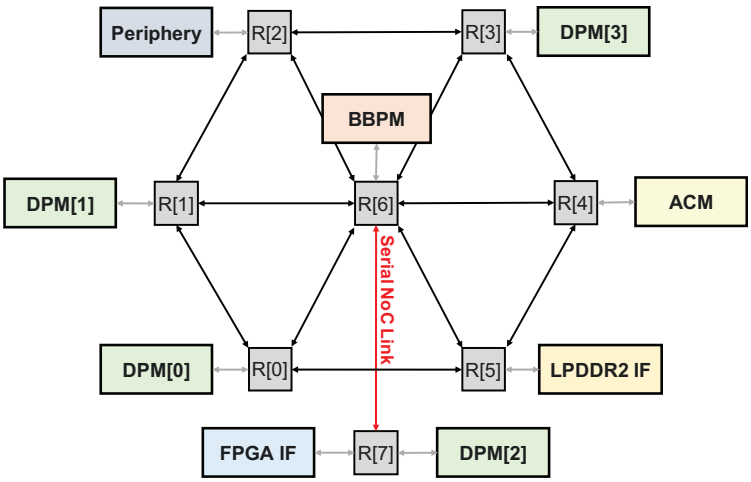
Figure 3.6: Examples of flits for reading and writing to the router LUT

the data field. In this way, the core manager is able to read or write any or all LUTs of a router. In addition to the router LUT, the core manager can similarly access registers such as counters which record the number of flits that have arrived through each input port of the router. Similarly, the counter registers can also count up the number of flit CRC fails for a certain input port. Thus, via reading these register values, the core manager can determine the link usages of links of the NoC or the number of packet losses due transient errors in the NoC.

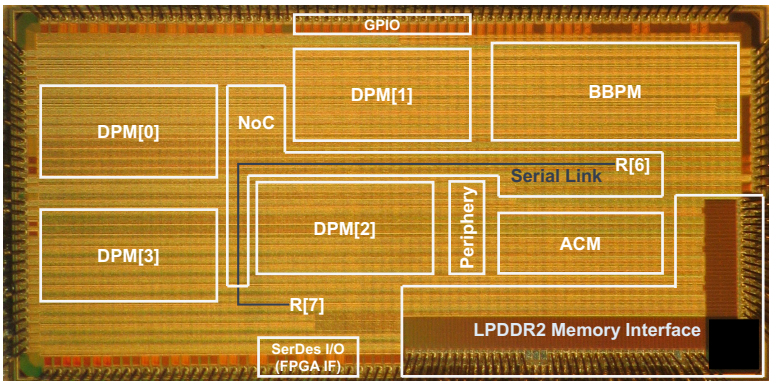
3.2.4.1 Synthesis results

Our routers were tested in Cadence simulator where the core manage successfully reconfigures the router LUTs. As also mentioned, the router was integrated into a 7-router hexagonal NoC, fabricated in Globalfoundries 28 nm SLP CMOS technology shown in schematic diagram (Fig.3.7(a)) with die photo (Fig.3.7(b)) and presented in [H+17].

Tests on the router have confirmed the router programmability function. The reconfigurability function has also been successfully implemented into the routers of the MPSoC Kachel, to be taped out in May 2018 and consisting of 4 processing modules as wells as chip-to-chip links. A 3D off-chip NoC topology connects chips in a tiled chip architecture. The LUT inside these routers is greater than 64 bits due to the 3D routing function so that the entire LUT cannot be fitted into the data field of a flit. Thus, it requires a burst



(a) Hexagonal NoC topology Tomahawk4 MPSoC composed of reconfigurable routers.

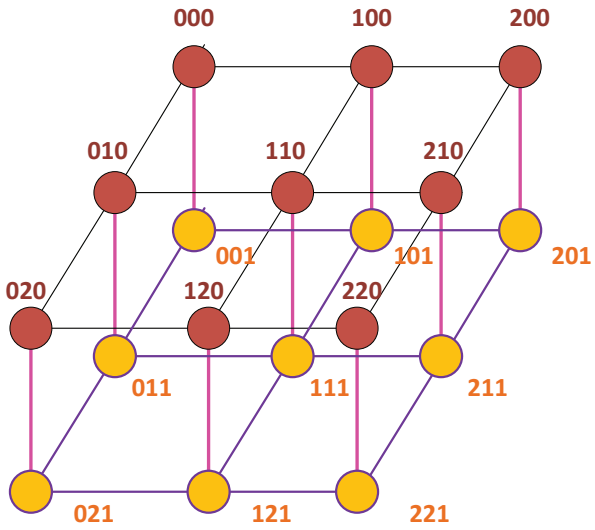


(b) Tomahawk4 die photograph.

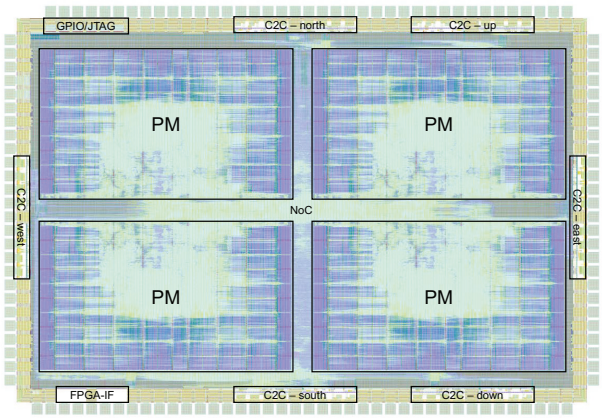
Figure 3.7: Schematic diagram and die photo of Tomahawk4 MPSoC showing hexagonal NoC with programmable routers.

transfer of 2 flits to reconfigure the full LUT. A schematic diagram of the off-chip network of Kachel and the die photo of Kachel MPSoC is shown in Fig. 3.8

The area and achievable frequency results from synthesis of the router, at different locations of the NoC, is given in Tab. 3.1. Moreover, the component-wise area distribution of the center router (having 6 links to adjacent routers and 2 further links to the local PE and the router module respectively) is shown in Fig. 3.9. As can be seen, the additional are of the reconfigurable logic (including the nocif) in comparison to the basic router logic is very small (5% of total area when routing logic is 26% of total area). The area of any router is usually dominated by the input/output fifo buffers. The router link input/output buffers are 8 flits deep with a flits depth of 160 bits. In this case, the NoC link buffers and the buffer between the router module and the nocif consumes 69% of the total area. The routers are able to achieve more than 500 MHz frequency (when synthesized either with 28nm Global Foundries or with 65nm CMOS TSMC technologies). Thus, showing



(a) 3D off-chip interconnect of size 3x3x2 between multiple Kachel MPSoCs.



(b) Kachel die photograph.

Figure 3.8: 3D mesh off-chip network of Kachel chips with internal NoC consisting of programmable routers.

that even with the addition of the programmibility logic, the NoC is able to achieve high frequencies.

Table 3.1: Area and frequency results of the programmable router

Router	Area (μm^2)		Frequency (MHz)
	Combinational	Register	
Center (6+2)	21152.8	30845.8	518
Corner(3+2)	10674.8	18958.7	518

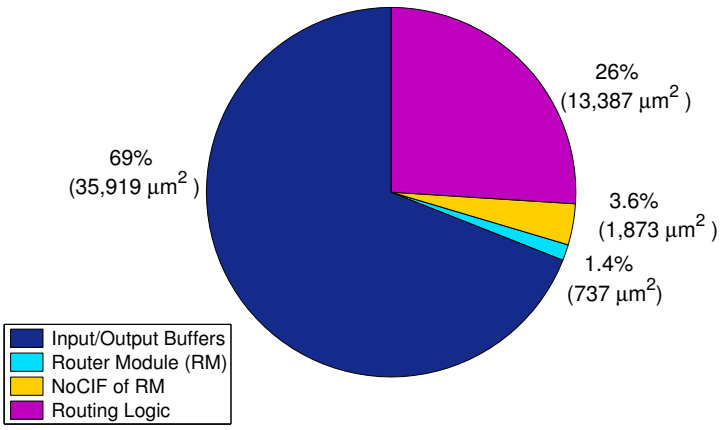


Figure 3.9: Area distributions of the centre router of a Hex NoC

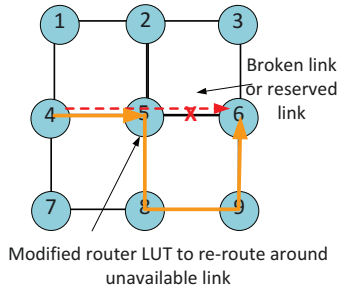
3.2.5 Application Scenario- Traffic rerouting for fault resilience and link reservation

The notion of reserving links for communication between PEs is useful in our MPSoC scenario to access the memory of adjacent PEs, dubbed as 'memory stealing'. As on-chip memory is always the bottleneck for many applications, memory stealing helps to alleviate the performance of such applications. Moreover, after design and production, a link may fail in the MPSoC. Thus, links may become unavailable due to (temporary) reservation or due to permanent failure, as depicted in Fig. 3.10. In theses, cases to keep the NoC function, the unavailable links should be by-passed by the traffic. In this example, initially X-Y routing (Fig. 3.10(a)). When the east output port of router 5 becomes unavailable, the south output port is used to reach destinations to the East, NorthEast, South and SouthEast (Fig. 3.10(b)).

In order to demonstrate the benefit of the routing scheme in a generic application scenario, we chose a more abstract simulation model using Matlab. Therein, we considered a 8×8 mesh NoC with faulty links. We assume that the router has some method implemented to detect failures in its links and then inform the core manager (located at middle of the network), which will then change the router LUTs accordingly. All links of the router connected to the core manager were considered fault-free. By injecting different number of faults over all possible random locations, we obtained the average successful connectivity between nodes, denoted as reliability. The opposite of reliability i.e. 1-reliability is the residual error rate which denotes the percentage of node pairs which were unconnected due to the link faults.

The results show that at any location of a single unavailable link, with XY routing the NoC error rate becomes 3%, while with reconfigured routing, it is 0%. Similarly, with 3 (1%)

Relative direction of target wrt to current router	O_W	O_S	O_E	O_N
W	1	0	0	0
SW	1	0	0	0
S	0	1	0	0
SE	0	0	1	0
E	0	0	1	0
NE	0	0	1	0
N	0	0	0	1
NW	1	0	0	0



(a) Mesh NoC with original LUT at router 5 before east link becomes unavailable.

O_S now used to reach targets in, E, NE, S and SE directions

Relative direction of target wrt to current router	O_W	O_S	O_E	O_N
W	1	0	0	0
SW	1	0	0	0
S	0	1	0	0
SE	0	1	0	0
E	0	1	0	0
NE	0	1	0	0
N	0	0	0	1
NW	1	0	0	0

O_E Faulty- this outport is not selected

(b) Modified LUT of router 5 after east link of the router fails.

Figure 3.10: Modified LUT when the east output link of router 5 fails.

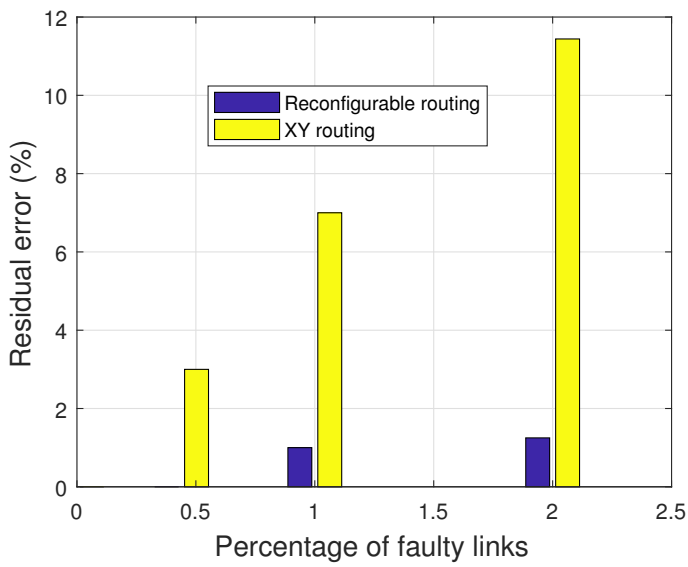


Figure 3.11: Residual error rate

unavailable links, the residual error rate of XY routing is 7% whereas with reconfigured routing, it is 1%, as shown in Fig. 3.11. This clearly shows the advantage of reconfigurable routing as opposed to XY routing. However, with high rate of faults, it may happen that the managing core is longer able to reach certain router in order to reconfigure their LUTs so that in this scenario reliability will fall.

3.3 Fault-tolerant routing in NoC

Fault tolerance of networks to permanently faulty components, both off-chip and on-chip has been a topic of intense research over the last few years[MDPT12, JTWB09, FDC+09, PLB+04]. A great portion of these works are based on adaptive routing techniques to route around faults and many of these are based on the turn model [GN94], which give a methodology on how to design adaptive routing algorithms for wormhole switched networks. In [GN93], fault-tolerant version of the Negative-First Routing algorithm for the 2D mesh is presented and it is shown that the algorithm can tolerant any one faulty router. Authors in [IY11] extend this work to tolerate any one faulty link or router. Another work which considers the turn model approach for 2D meshes is [Wu03], in which a fault-tolerant and deadlock free routing protocol based on the odd-even turn model is presented. The faults are contained in a disjointed rectangular sets called faulty blocks comprising both faulty as well as disabled healthy nodes. Authors in [FFH09] propose a fault-tolerant routing algorithm which is deadlock-free and able to achieve a higher throughput with less number of deactivated nodes. All of the above approaches do not require the use of virtual channels. The approach of sacrificing healthy nodes to route around fault regions containing both healthy and faulty nodes is considered in many works, such as in [BC95][F+09].

Another approach to designing fault tolerant routing algorithms considers the use of virtual channels as in [MDPT12][MDP13], where two virtual channels are needed to tolerate all one and two fault links and all one faulty routers, respectively while keeping the network performance optimal by providing alternate minimal routes for the packet traversal. Instead of using adaptive routing, authors in [FDC+09] reconfigure the routing algorithm in the case of link failures. Authors in [FLJ+10] propose an algorithm that reconfigures the routing table through reinforcement learning and their approach is applicable to any topology and not dependent on the shape of the fault region. In [LHLM15], authors present a low-cost fault tolerant routing algorithm for efficient routing path selection using traffic status of the NoC. Although all of the above mentioned works consider mesh topology, which is the commonly used NoC topology, other topologies such as the torus and the hexagonal mesh have also been considered in some works. The hexagonal network is investigated in works such as [SB13][GZLT06] as the hexagonal network having lower average hop count is considered to have better performance than the mesh. In [SB13], authors present a deadlock free adaptive routing algorithm for the hexagonal torus in-

terconnection network using three virtual channels per physical channel. In [GZLT06], authors propose a different addressing scheme for the hexagonal network which allows to adapt the turn model based adaptive routing algorithms for the mesh to the hexagonal network. A diagonally connected mesh network is presented in [HLB08] with adaptive quasi-minimal routing algorithm.

In most works, the deadlock freedom verification of the routing algorithm is based on examining the channel dependency graph (CDG). For deadlock freedom the CDG must be acyclic as was proved in [Dua93].In [CHKP06], a method is presented for designing application specific deadlock-free routing algorithms for any topology by using an heuristic to cut edges in the CDG and thereby preventing cycles. In the following sections, we propose fault-tolerant deadlock free routing algorithms for the hexagonal and octagonal NoCs using the turn model, which was not done before. We do not use costly virtual channels but the addition of virtual channels may bring further advantages in term of fault tolerance. Unlike the widely analyzed mesh NoC, we expect the application of the turn model to be have greater complexity in our case since due to the additional directions which create many possibilities of deadlock. Thus we develop a simple approach to ensure deadlock freedom of the adaptive routing algorithm, using graph theory.

3.4 Application of the turn model

As explained in chapter 2, section 2.1.2.2, adaptive routing algorithms are prone to deadlock due to the usage of wormhole switching and must be designed with care to prevent deadlock. We use the turn model [GN94] for developing deadlock-free adaptive routing algorithms as this approach does not require the usage virtual channels. For the formation of adaptive routing algorithms according to this model, the channels in the network must be first partitioned into the directions in which they route packets. Deadlock cycles are created between turns leading to closed cycles and so sufficient turns between these directions should be avoided to prevent cycle formations. Moreover, these set of turns should also prevent all possible simple and complex deadlock cycles formed as a result of combination of turns in clockwise (CW) and counter-clockwise (CCW) directions.

3.4.1 Turn model applied to Hex NoC

In the case of the hex NoC, as shown in Fig. 3.12, there are 3 directions (the x, y and diag directions in the 2D rectangular system), each pair of which are at least 60 degrees apart. On closer inspection, it can be seen that the simplest cycles in hexagonal network are the two basic triangular cycles, as can be seen in Fig. 3.12. Furthermore, many different bigger deadlock cycles can form by combinations of these 2 basic cycles (Fig. 3.12(b)). According to the turn model, a turn in the simplest cycle should be prevented so that deadlock is also prevented in more complex cycles.

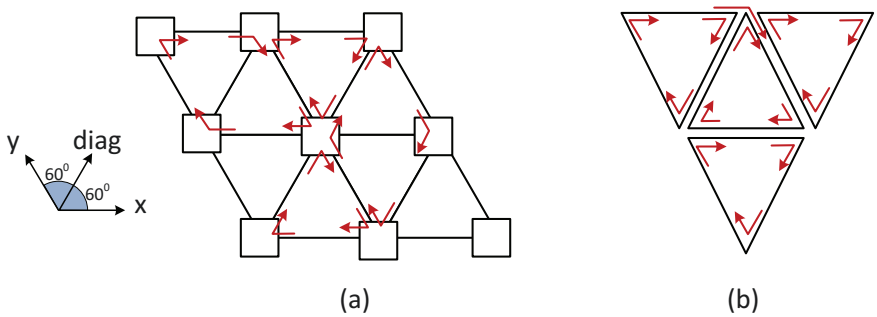


Figure 3.12: Examples of some deadlock cycles forming in the 60 degrees or hexagonal NoC between the three directions.

To prevent deadlock in e.g. the CW direction, two 60° turns (i.e. two turns between two different pairs of directions) must by prevented as well as the turn formed by the combination of these two turns, i.e. a 120° turn. This is because the two triangles can be combined in 3 ways (due to the 3 sides of a triangle). Thus by preventing a turn in each triangle (between different pairs of directions) and another turn from the combination of these two triangles, deadlock is prevented. Some 180° turns can be allowed to increase adaptivity when deadlock is not created by them.

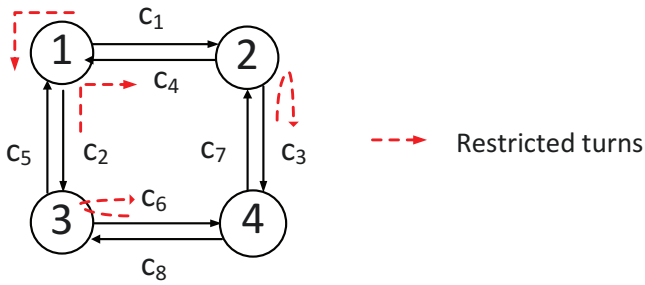
3.4.1.1 Channel Dependency Matrix

As there are many possibilities of simple and complex cycles forming due to combinations of the cycles in the CW and CCW directions, the process of obtaining the correct selection of prevented turns can be quite cumbersome for topologies with many different flit directions. Generally, after the selection of the turns, the channel dependency graph (CDG) must be examined to determine whether any cycles exist [Dua93]. To simplify the selection of the right combination of turns, we have devised an approach using matrix algebra.

After selecting the required number of turns in the CW and CCW directions, the channel adjacency matrix representation of the CDG, called channel dependency matrix (CDM) is generated. The entries of the CDM, A_{ij} represents the dependency of a channel i to channel j, i.e. whether a turn is allowed from channel i to channel j. A_{ij} equals 1 iff i and j are adjacent and if channel j can be requested immediately after channel i:

$$A_{ij} = \begin{cases} 1, \text{ if i and j are adjacent and turn from i to j is allowed} \\ 0, \text{ otherwise} \end{cases} \tag{3.1}$$

Then the CDM is checked for cycles by finding its transitive closure (TC), which indicates the reachability of the nodes. If the TC has 1's along the diagonal, it indicates that via



$$A = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{array} \begin{array}{c} \begin{array}{cccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad TC = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{array} \begin{array}{c} \begin{array}{cccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 3.13: The channel dependency matrix, A and the Transitive Closure, TC for 2×2 Mesh NoC with the indicated turns restrictions.

one or more intermediate channels, it is possible to start from a channel and return to that channel. Thus a cycle of channel dependencies is present indicating a deadlock possibility. As an example, the CDM and the TC for a 2×2 mesh NoC with a selected set of prevented turns is shown in Fig. 3.13. With the selection of prevented turns: North-to-East and West-to-South and the 180^0 turns West-to-East and North-to-South, the diagonal of the TC contains all zeros. Thus, for these prevented turns, the routing algorithm is deadlock free. However, the size of the NoC must also be taken into consideration. For the hexagonal NoC, to include all cycles formed from the combination of the two triangles, a minimum NoC size of 3×3 should be used to generate the CDM. NoC sizes beyond this can also be used, however, it must be taken into account that the time for the TC calculation increases with the network size.

3.4.1.2 Selecting restricted turns for deadlock freedom

When presenting the routing algorithms, we consider the hexagonal NoC as presented in Fig. 3.1(a) and use the direction names of the x-,y- and diagonal directions, as in a standard rectangular coordinate system for easier comparison to the mesh topology. We can summarize the procedure for finding the possible turn restrictions as follows:

- **Step1:** Initially we want to find the prevented CW turns. Thus, first assume all turns in CCW direction and all 180° turns are disallowed but all CW turns are allowed, depicted in Fig. 3.14(a).

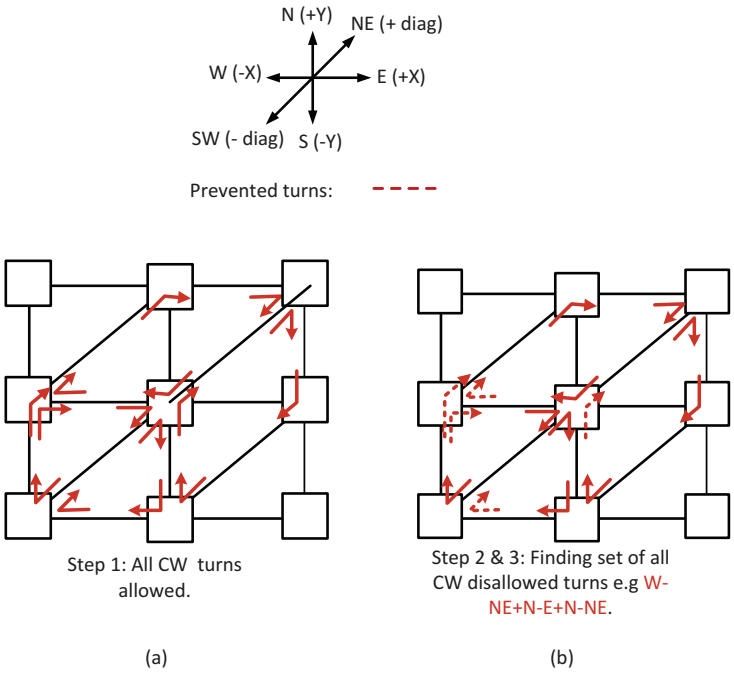


Figure 3.14: Finding CW restricted turns.

- **Step2:** Choose a disallowed CW turn between a pair of directions e.g. between x and diagonal direction i.e. W-NE turn in one of the basic cycles. Choose another disallowed CW turn between another pair of directions e.g. between x and y directions i.e. N-E turn in the other basic cycle. The third prevented CW turn is that between between the remaining two directions i.e y and diagonal direction e.g. in this case the N-NE turn. Generate the CDM and the TC and if trace(TC)>0, then this combination of prevented turns will lead to a deadlock.

- **Step3:** Iterate through other possible CW turns in the first cycle and repeat. The result is the set of all possible turn combinations for no cycle in the CW direction.

- **Step4:** Repeat steps 1, 2 and 3 for the CCW direction turns (Fig. 3.15(a)).

- **Step5:** For all sets of the prevented CW and CCW turns, generate the CDM and then the TC and select the combinations with no cycles. If the TC has a non-zero diagonal i.e trace(TC)>0, then this combination of prevented turns will lead to a deadlock cycle and is thus not valid. This step will reveal any complex cycles existing between the CW and CCW cycles (Fig. 3.15(b).

- **Step6:** For all the sets of the valid prevented turns, iterate through all 180° turns and allow as many as possible so that the generated the CDM has no cycles as shown in Fig. 3.16.

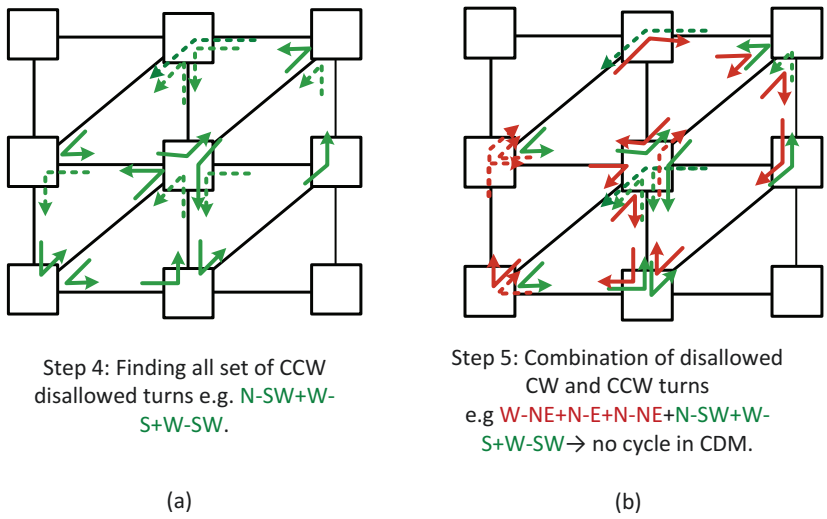


Figure 3.15: Finding CCW restricted turns.

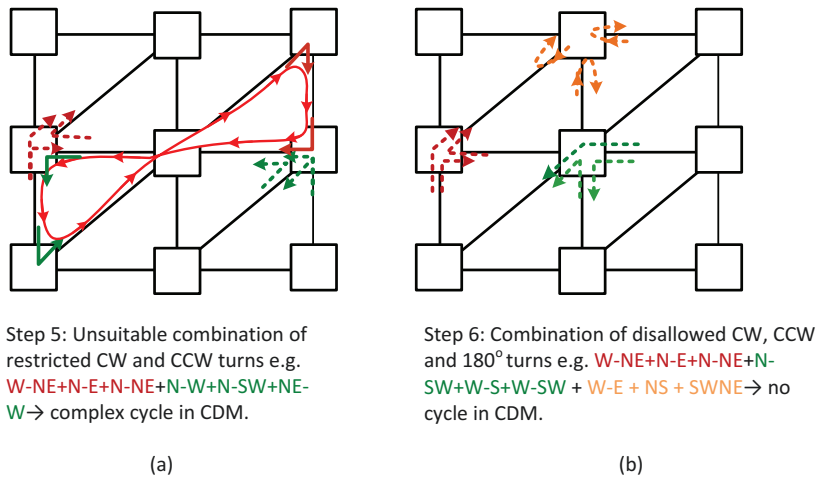


Figure 3.16: Finding combination of CW, CCW and 180° restricted turns.

Our investigations resulted in 18 possible adaptive routing algorithms, 6 of which are unique due to symmetry. This can also be calculated intuitively by observing all the turns as follows: firstly for the CW directions, for a restricted turn in one of the basic cycles (e.g. between x and diagonal), there are 2 other possibilities for restricted turn in the other basic triangular cycle (between x and y or between y and diagonal) so that there are 2 sets possible of the CW turn restrictions (Fig. 3.17). If we change the first restricted CW turn, then another 2×2 sets of CW turn restrictions are possible. Thus considering symmetry, there are 2 unique combinations of the CW restricted turns. Similarly, for the CCW turns, we can find 6 possible sets of turn restrictions. When combining the CW and CCW turns, complex cycles consisting of both CW and CCW turns can form. As an example, if we observe Fig. 3.16(a), the complex cycle resulted due to prevented turns

between the exact same directions but in opposite manner regarding the starting direction (W-NE and NE-W turns in Fig. 3.16(a)). This condition allowed a complex turn to result when both these turns could be bypassed due to combining CW and CCW cycles.

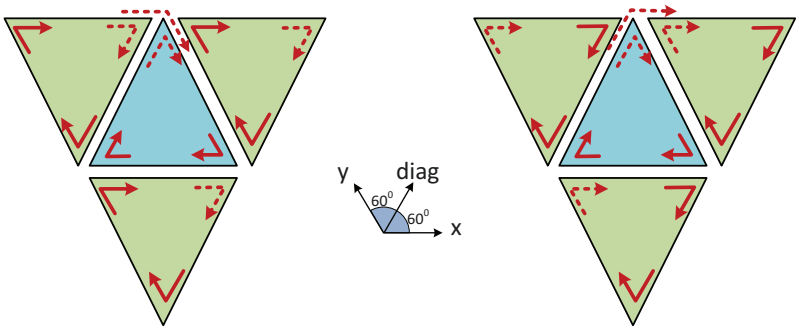


Figure 3.17: A unique set CW restricted turns.

A set of these algorithms with the corresponding prevented turns set is shown in the Table 3.2. Here E,W,S,N,NE and SW represents the East, West, South, North, Northeast and Southwest directions, respectively. The name of the algorithm is based on the set of directions to which the flit is prevented to turn to e.g. in the E-S First, we see that all turns to the E and S directions are disallowed. Thus, any flit wanting to travel to these directions (e.g. E, NE or SE or S or SW etc.) must first travel first to the E or S and then turn to the next directions. Otherwise, if the flit starts in the N, NE, S, or SW direction , it can cannot afterward turn towards the E and S as all of these turns are forbidden.

Table 3.2: Prevented turns for adaptive routing algorithms in the hex NoC.

	Prevented turns					
Name of Algorithm	CW			CCW		
	x/diag	y/diag	x/y	x/diag	y/diag	x/y
E,S First	NE-to-E	NE-to-S	N-to-E	SW-to-E	SW-to-S	W-to-S
E,S, SW First	NE-to-E	NE-to-S	N-to-E	W-to-SW	N-to-SW	W-to-S
N,NE Last	NE-to-E	NE-to-S	N-to-E	NE-to-W	N-to-SW	N-to-W
S,SW First	E-to-SW	NE-to-S	E-to-S	W-to-SW	N-to-SW	N-to-W
W,S,SW First	E-to-SW	NE-to-S	E-to-S	NE-to-W	N-to-SW	N-to-W
E,NE Last	E-to-SW	NE-to-S	E-to-S	NE-to-W	NE-to-N	E-to-N

Two of the set of algorithms from the 6 adaptive routing algorithms provide at least three alternate paths for flit transmission, via three output-port directions and reaching via three input-port directions. Therefore these algorithms should be able to tolerate 2 faulty routers, if 3 disjoint paths exist so that if up to 2 faults block 2 paths, then the remaining can be used to reach the destination. Of these algorithms, we will concentrate

on the "W-S-SW First" algorithm (highlighted in blue) for fault-tolerant design and for performance evaluation. In later sections, we compare the performance of this algorithm to existing similar fault-tolerant routing algorithm for the rectangular mesh NoC.

3.4.2 Turn model applied to Oct NoC

The 2D oct NoC, shown in Fig.3.1, has twice the number of links as the 2D mesh, which is a considerable overhead considering that the links have input buffers which contribute to the main area and power overhead of the router. Moreover, the size of the crossbar inside the router increases quadratically with the number of links. However, we still investigate oct since its performacne may be worth the overhead in situations of high error rates. Using the procedure described in Sec. 3.4.1.2, we are able to find the prevented turns for the octagonal NoC. There are four directions of flit travel in the oct NoC- the x-, the y- and the two diagonal directions. As a result, between the pairs of directions there are 6 possible cycles (x/y, x/diagonal1, x/diagonal2, y/diagonal1, y/diagonal2, diagonal1/diagonal2), so that at least 6 turns should be prevented in both the CW and CCW directions. Some of the possible turns and the resulting deadlock cycles forming are shown in Fig. 3.18. Using again the method of the transitive closure of the CDM, we can find the necessary combination of turn preventions to achieve deadlock freedom. The results of this analysis yields 24 different possible adaptive routing algorithms, 6 of which are unique due to symmetry. The names of a unique set of these algorithms with the corresponding restricted turns are given in Table 3.3. As can be observed, two of these algorithms i.e. the NW-W-SW-S First (go adaptively first in the NW, W, SW and S directions and then the remaining 4 directions- N, NE,E and SE to reach the destination) and the W-SW-S-SE Last algorithms will provide 4 alternate paths to the destination.We choose the NW-W-SW-S First and design the fault-tolerant version of this algorithm and compare the performance of this algorithm to the Negative-First algorithms for the mesh and hex NoCs.

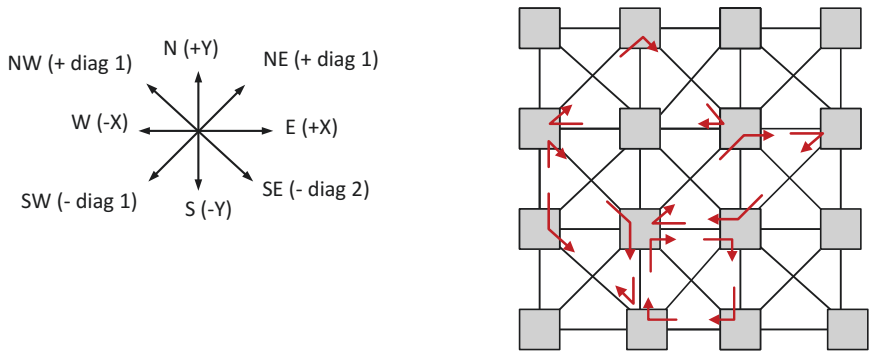


Figure 3.18: Examples of some deadlock cycles forming in the octagonal NoC between the four directions.

Table 3.3: Prevented turns for adaptive routing algorithms in the Oct NoC.

Algorithm	Prevented turns					
	x/diag1	y/diag1	x/y	x/diag2	y/diag2	diag1/diag2
NE,E,N Last	E-to-SW NE-to-W	NE-to-S N-to-SW	E-to-S N-to-W	E-to-SE E-to-NW	N-to-SE N-to-NW	NE-to-SE NE-to-NW
W,SW,S,SE First	E-to-SW NE-to-W	NE-to-S N-to-SW	E-to-S N-to-W	E-to-SE NW-to-W	N-to-SE NW-to-S	NE-to-SE NW-to-SW
S,SW,SE First	E-to-SW W-to-SW	NE-to-S N-to-SW	E-to-S W-to-S	E-to-SE W-to-SE	N-to-SE NW-to-S	NE-to-SE NW-to-SW
NE,E,SE Last	E-to-SW NE-to-W	NE-to-S NE-to-N	E-to-S E-to-N	SE-to-W E-to-NW	SE-to-S SE-to-N	SE-to-SW NE-to-NW
S,SW,W,NW First	E-to-SW NE-to-W	NE-to-S N-to-SW	E-to-S N-to-W	SE-to-W E-to-NW	SE-to-S N-to-NW	SE-to-SW NE-to-NW
W,SW,S First	E-to-SW NE-to-W	NE-to-S N-to-SW	E-to-S N-to-W	SE-to-W N-to-WS	SE-to-S NW-to-W	SE-to-SW NW-to-NW

3.5 Fault Tolerant Routing Algorithm

Designing adaptive routing algorithms is challenging since care must be taken to avoid deadlock and livelock, while trying to make efficient use of the path diversity. In these algorithms, the routing decision is taken based on the network conditions such as network congestion or faults. The difference between traffic-aware and fault-tolerant adaptive algorithms is that in case of the latter, non-minimal paths may be taken to increase path diversity and thus fault-tolerance. In the fault-tolerant W-SW-S First algorithm for the hex NoC, the flits should be routed adaptively first in the South, SouthWest or West directions (if necessary) and then adaptively in the North, NorthEast and East directions to reach the destination. This algorithm is also called Negative First Fault-tolerant (HexNegFirstFT) routing algorithm, as W, SW and S are the negative x, diagonal and y directions respectively for the hexagonal NoC and all turns from the positive directions to the negative directions are prevented. The Negative-First Fault-tolerant routing algorithm for the mesh was presented in [GN93] and was shown to be able to tolerate all cases of 1 faulty router, as it can provide 2 disjoint routes from a source to a destination node. The turns prevented in the HexNegFirstFT algorithm are the following: (CW) E-to-SW (60^0), NE-to-S (60^0), E-to-S (120^0) and (CCW) N-to-SW (60^0), NE-to-W (60^0), N-to-W (120^0). In addition the following 180^0 turns are disallowed: E-to-W , NE-to-SW and N-to-S. Initially only local knowledge of faults is assumed, i.e. a router knows only which of its immediate neighbors are faulty. However, as we will see later, in the case when the destination is to the NE of the current router, then local knowledge is not sufficient to have 2 router-fault tolerance. Although in this section, we describe the HexNegFirstFT algorithm, the E-S-SW algorithm also provides 3 alternate paths to the destination and is therefore also 2 router fault tolerant.

The algorithm is designed in such a way that whenever more than one output port is available towards a destination, the output port which will be selected is the one which leads to the highest number of choices for the next hop, in order to increase fault-tolerance. Accordingly, when the destination is to the W, S or SW of the current node, the output ports in the W, S or SW directions are chosen adaptively (according to having non-faulty router in the corresponding direction), to reach the destination. When the destination is to the N or E or NE of the current node, the flit should be first forwarded in the negative direction, i.e. in the W, SW or S directions, adaptively, to reach a position from which 3 disjoint paths toward the destination are available. Then the positive directions, i.e. the N, NE or E directions are selected adaptively towards the destination. Since, it is possible for all source-destination pairs to begin in 3 possible directions and also to reach the destination via 3 ports, the routing algorithm can tolerate any 2 router faults.

3.5.1 Destinations to the NE

The pseudo-code for destinations to the NorthEast of the source is given in Algorithm 1. Essentially, when the destination is to the NE of the current node, it is possible to go in three directions, i.e. N or E or NE or even in the negative directions, but only if the previous direction of travel was not already in a positive direction. For a source-destination pair, it is first important to determine the number of hops to the destination from the current router (since the decision is taken again at each router) in the x- and y-directions, denoted by x_{offset} and by y_{offset}. If the $x_{offset} > 1$ and $y_{offset} > 1$, the three positive directions, N, NE or E directions are taken adaptively. The output port direction which leads to the higher path diversity and which does not lead to a faulty router is selected, as can be seen in Algorithm1. If $x_{offset} = y_{offset}$ as in Fig. 3.19(a), NE output port is preferred since this is the shortest path. If the NE router is faulty, then either the E or N port can be selected, as both lead to the destination in equal number of hops (Fig. 3.19(b)).

However, at this point, with only local knowledge of faults, it is not possible to ensure reaching the destination, as seen in Fig. 3.19(b), where if E output port had been selected, destination would not have been reachable. Because of this, the router requires the fault knowledge of the neighbor which is 2 hops away in the E-NE or alternatively in the N-NE direction. With this knowledge, the router is able to find a path even in the presence of two faulty routers. This problem occurs only when the destination to the NE of the current node and $x_{offset} = y_{offset} = 2$. The flit has exactly 3 paths toward the destination and if the NE is faulty, the other two paths are completely independent of each other. The same situation for the SW is not such a critical point since there the movements are in the negative direction so that, even if a path leads to a dead end, the flit can still reach the destination by going around the fault, as shown in Fig 3.22(a).

Algorithm 1 HexNegFirstFT: Destination to NE

```
X_offset = X_target − X_current, Y_offset = Y_target − Y_current
if X_offset > 1 & Y_offset > 1 & X_offset ≠ Y_offset then
    if X_offset > Y_offset then
        if neighbour(E direction) is notfaulty then
            Select East
        else if neighbour(NE direction) is notfaulty then
            Select NorthEast
        else if neighbour(N direction) is notfaulty then
            Select North
        else
            Drop Flit
        end if
    else if X_offset < Y_offset then
        if neighbour(N direction) is notfaulty then
            Select North
        else if neighbour(NE direction) is notfaulty then
            Select NorthEast
        else if neighbour(E direction) is notfaulty then
            Select East
        else
            Drop Flit
        end if
    end if
else if X_offset > 1  &  Y_offset == 1 then
    if neighbour(S direction) is notfaulty then
        Select South
    else if neighbour(E direction) is notfaulty then
        Select East
    else if neighbour(NE direction) is notfaulty then
        Select NorthEast
    else
        Drop Flit
    end if
else if X_offset == 1  &  Y_offset > 1 then
    if neighbour(W direction) is notfaulty then
        Select West
    else if neighbour(N direction) is notfaulty then
        Select North
    else if neighbour(NE direction) is notfaulty then
        Select NorthEast
    else
        Drop Flit
    end if
else if X_offset = Y_offset then
    if neighbour(NE direction) is notfaulty then
        Select NorthEast
    else if neighbour(E direction) is notfaulty then
        Select East
    else if neighbour(N direction) is notfaulty then
        Select North
    else
        Drop Flit
    end if
end if
```

3.5.1.1 Destinations to the N or E

The complete routing algorithm is given in Table 3.4. When the destination is to the East or North, the output ports to the E or N, respectively should not be taken since the flit

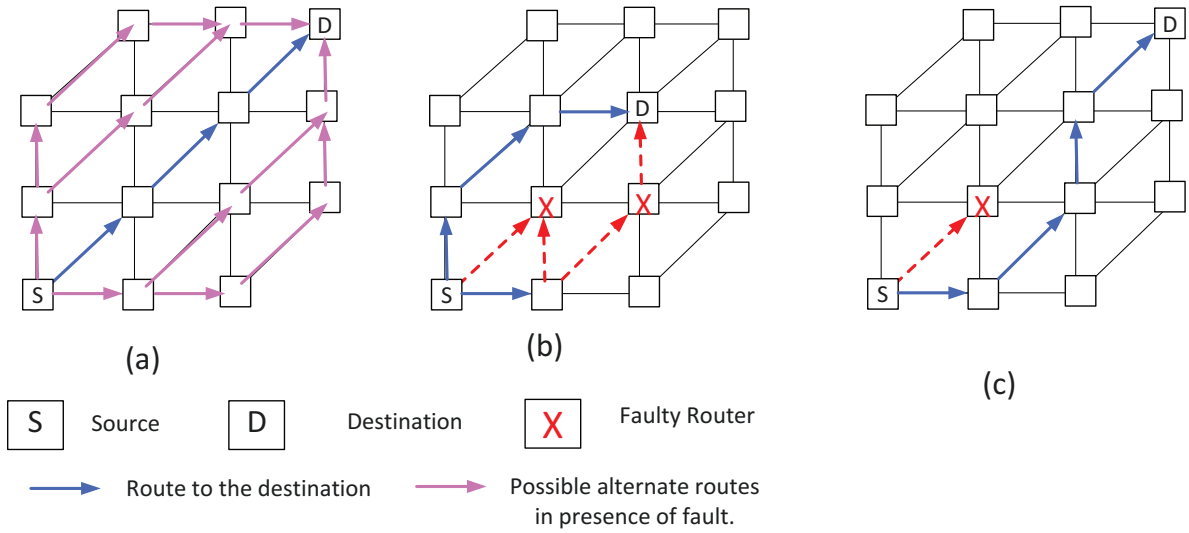


Figure 3.19: Paths to the destination at NE of the source.

Table 3.4: Hex Fault Tolerant Negative-First Routing algorithm

	Incoming flit direction				
Dest	N	E	S	W	local
NE	NE,N,E*	NE,N,E*	NE,N,E*	NE,N,E*	W/S**, NE,N,E*
E	E	E	S,SW,E	-	S,SW,E
N	N	N	-	W,SW,N	W,SW,N
NW	-	-	W,SW,S	W,SW,S	W,SW,S
SE	-	-	S,SW,W	S,SW,W	S,SW,W
S	-	-	S,SW,W	S,SW,W	S,SW,W
W	-	-	W,SW,S	W,SW,S	W,SW,S
SW	-	-	SW,W,S	SW,W,S	SW,W,S

** $X_{offset} = 1/Y_{offset} = 1$

*Select the output port that leads to highest path diversity.

starts out in a positive direction, it is not allowed to move into a negative direction when a faulty router is encountered. Instead, here first 2 hops in the negative direction i.e. in the W or S or SW are taken, so as to bring the destination to the North East of the current router. At this position, with $X_{offset} = 2$ or $Y_{offset} = 2$ respectively and there are 3 disjoint paths to the destination. Then the rules in Algorithm 1 are followed to the destination. Some examples with different locations of faulty routers are shown in Fig. 3.20 and Fig. 3.21. As in these cases, the path taken to the destination is a non-minimal one, there is some loss of performance. However, due to the diagonal link, the average path

length is still lower in comparison to mesh with the Negative First fault tolerant routing algorithm.

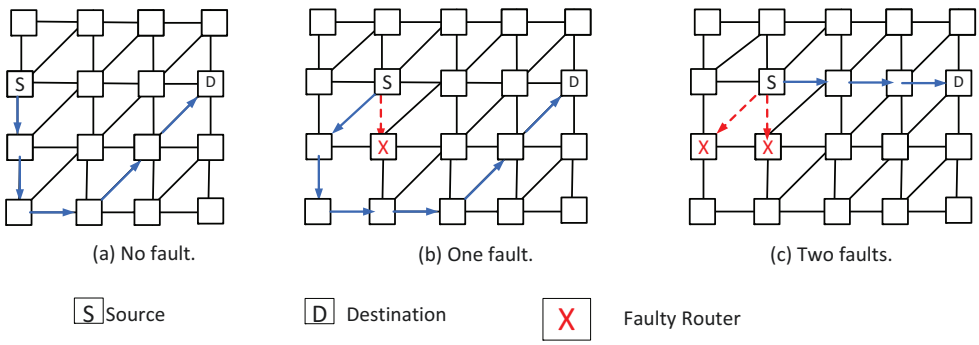


Figure 3.20: Destinations to the East of the source.

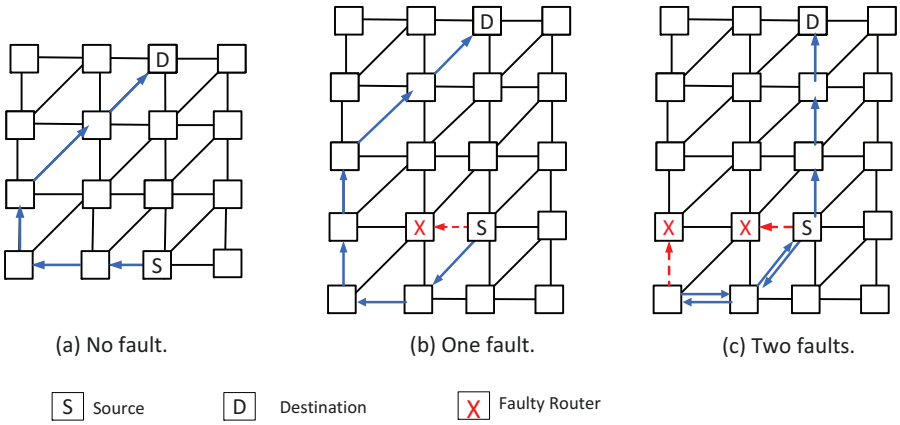
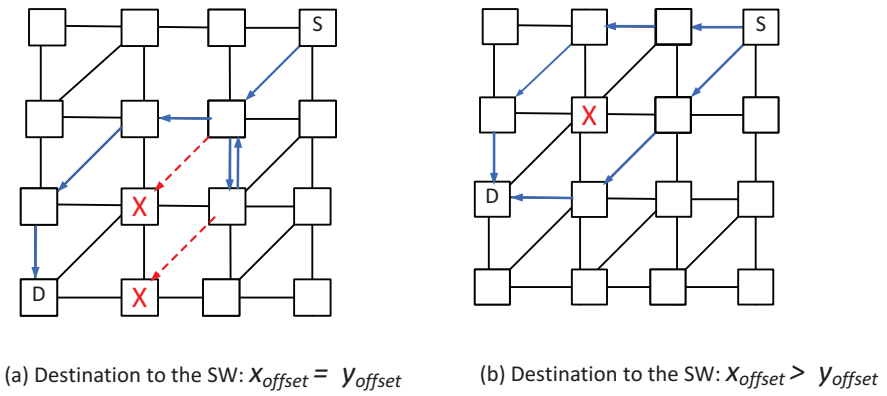


Figure 3.21: Destinations to the North of the source.

3.5.1.2 Destinations in other directions

When the destination is to the SW of the source, if the SW router is faulty, either the W or S can be chosen as here there are more than 1 path towards the destination in each direction, as shown in Fig. 3.22. Since the flit already starts out in a negative direction (W,S or SW), the routing path is more flexible here. However, similar to the NE case, again the output direction is chosen so that the current and destination router is not in one line. In this case, knowledge of neighbor routers' fault condition is sufficient. However, greater knowledge of non-neighbor faults would reduce the path length as can be seen in Fig. 3.22(a).

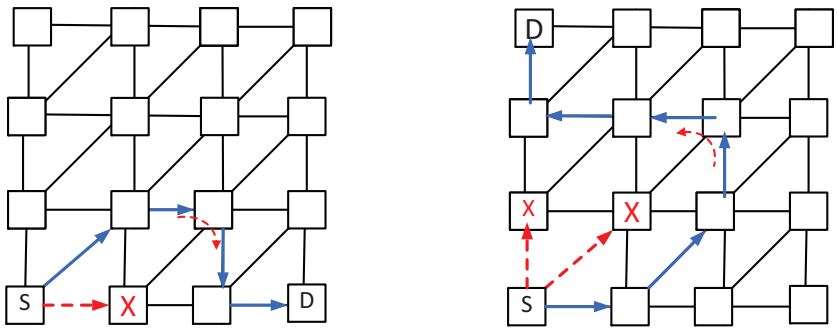


(a) Destination to the SW: $x_{offset} = y_{offset}$ (b) Destination to the SW: $x_{offset} > y_{offset}$

Figure 3.22: Alternate paths to the destination at SW of the source.

3.5.1.3 Destinations along the South or West Edge

When the path is along the West or South edge of the NoC, and a faulty router blocks the path to the destination, then some hops are taken around the fault to bypass it, as can be seen in Fig. 3.23. Although, in such cases, some forbidden turns are taken (such as E-to-S turn in Fig. 3.23(a) and such as N-to-W turn in Fig. 3.23(b)), deadlock does not happen since the cycle of dependencies cannot form through a faulty node at the edge.



(a) Disallowed turn: East-to-South, allowed here (b) Disallowed turn: North-to-West, allowed here

Figure 3.23: Routing along the west or south Edge.

3.6 Performance Evaluation

To evaluate the performance of the fault-tolerant routing algorithm, the hexagonal and octagonal NoC topology with the Negative First fault tolerant routing scheme was implemented in a cycle accurate simulator [WF08], based on a C++/SystemC simulation model. We also implemented the Negative First fault tolerant routing algorithm for the

mesh NoC, in order to compare its performance to the Hex and Oct. We evaluated the performance in terms of average network latency (in cycles) and the flit acceptance rate (flits/cycle/node) including the saturation point with varying flit injection rates. We further evaluated the fault resilience of the network to different numbers of router faults, measured by the average ratio of successfully delivered flits (to the destination) to the total number of injected flits into the network:

$$Fault\ resilience = \frac{Total\ number\ of\ flits\ received\ successfully}{Total\ number\ of\ flits\ injected\ into\ the\ network} \tag{3.2}$$

Another measure is the error rate , which is related to the total number of flits that were dropped due to all paths being blocked:

$$Error\ rate = \frac{Total\ number\ of\ flits\ dropped}{Total\ number\ of\ flits\ injected\ into\ the\ network} \tag{3.3}$$

The faulty routers are modeled as permanently defective units. Each router is aware of the fault status of its connected neighbour routers. As discussed earlier in Sec. 3.5, in the hexagonal NoC, each router is also aware of the fault of the neighbor two hops away in the E-NE direction (shown in Fig. 3.19(b)). We assumed uniform random traffic pattern i.e. all nodes generate packets with equal probability and with uniform random distribution. Moreover, we assumed nodes connected to faulty routers do not generate any packets and no messages are destined for these nodes. We evaluated the throughput performance with routers having input buffers with buffer depth of 8 flits, while the packet size is kept at 5 flits.

3.6.1 Latency Evaluation

To understand the saturation rates in the two topologies, let us consider that when minimal paths exist to the destination e.g. for destinations to the W or S of the source, each fault encountered increases the path length by 2 hops for the mesh whereas it is increased by 1 hop for the hex NoC. Thus, in comparison to the 2D mesh the performance of the hexagonal mesh is always higher in terms of average latency and fault-tolerance. For the latency evaluations, the simulations were run for an average of $200,000$ cycles and the number of faulty routers were varied from 0 to 2. The location of faults were chosen randomly and over all possible combinations. The results shown in Figure 3.24 show the average latency (averaged over all source-destination pairs) versus the flit acceptance rate. From the figures, it can be seen that as the hexagonal topology provides shorter paths with the adaptive routing algorithm (due to the diagonal link), the average path length is shorter and therefore the average path latency is lower than that for the mesh at the same flit acceptance rate. As a result, the hexagonal NoC has a higher flit acceptance rate before the network saturates. With 0 faults, the 8×8 mesh NoC has an average path length of 6.92 hops, while the same sized hex has an average path length of 5.48 hops, which is approximately 21% less.

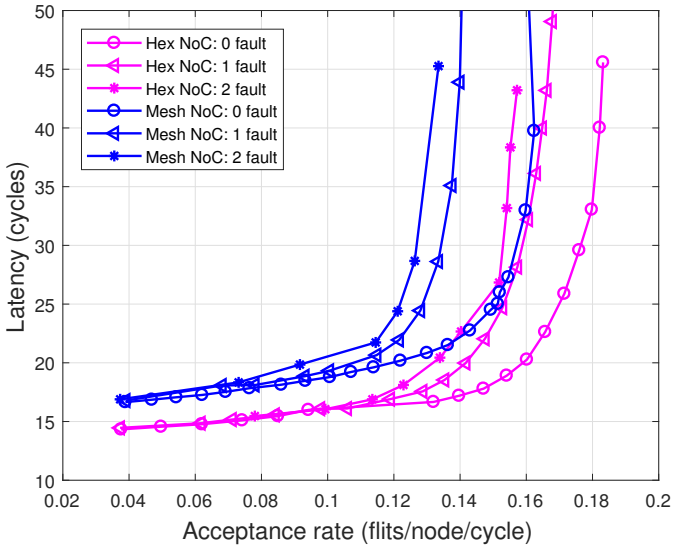


Figure 3.24: Latency versus flit acceptance rate for 8×8 NoC with uniform traffic.

3.6.2 Fault Tolerance Evaluation

The results for 8×8 NoC for the three topologies are depicted in Fig. 3.25 and Fig. 3.26. The location of the faulty routers were randomly selected over 1000 iterations. The results show that while the mesh has 100% fault reliability at any position of 1 faulty router, the hex and oct NoC has 100% reliability to any position of 2 and 3 faulty routers respectively. Note, however, for the hex NoC the two corner routers (upper left and bottom right) have only two input or output ports and therefore cannot be two-router fault tolerant. We do not consider these two nodes in our calculation, as they are not part of the true hexagonal NoC. Similarly the 4 corner routers are not considered for the Oct NoC. With increasing number of faults, the error rate of the mesh increases significanlty while it does so more gradually for the hex and oct. In particular, with 6 faulty routers, oct NoC has an error rate of 0.11% while that of the hex and mesh NoCs are 11× and 63.7× higher respectively.

We also investigated the fault reliability for higher number of faults for the hex, oct and mesh NoCs, compared in Fig. 3.26, where we can see that the octNoC has 70% higher fault resilience than the mesh and 21% higher fault resilience than the hex NoC. We also compared the reliability of different sizes of NoC to determine the effect of network size on fault tolerance. The results for 6×6, 8×8 and 10×10 sized NoCs are depicted in Fig. 3.27. For the investigated NoCs, by comparing the fault tolerance at 30% ratio of faulty routers, we found that the advantage of hex NoC compared to mesh increases with network size and for the 10×10 NoC, it is 53.6% more fault resilient than the mesh when there are 30% faults in the network.

The variation of the fault resilience for increasing network sizes, for the same ratio of faults is illustrated in Fig. 3.28. As can be seen, for the same percentage of faults, the

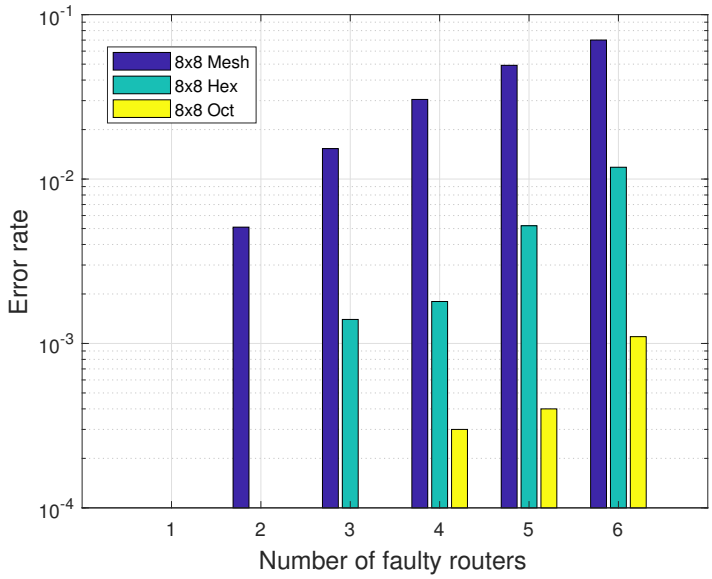


Figure 3.25: Comparison of fault resilience of 8×8 mesh and Hex and Oct NoCs for different numbers of faulty routers.

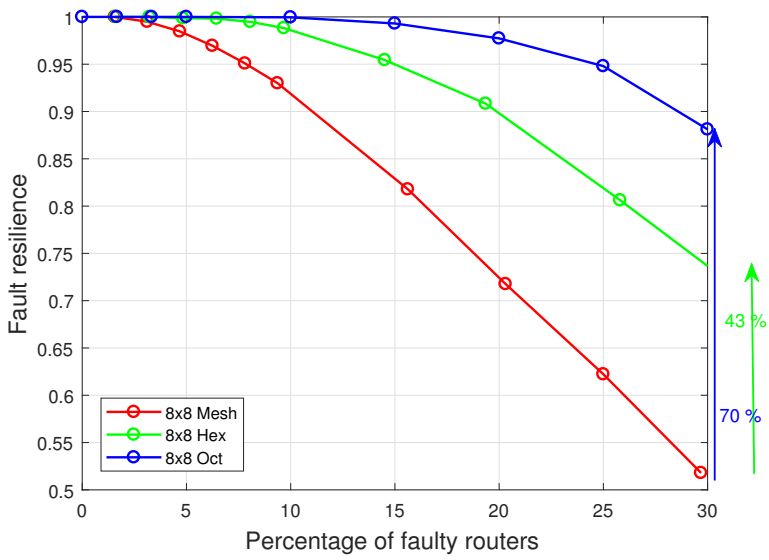
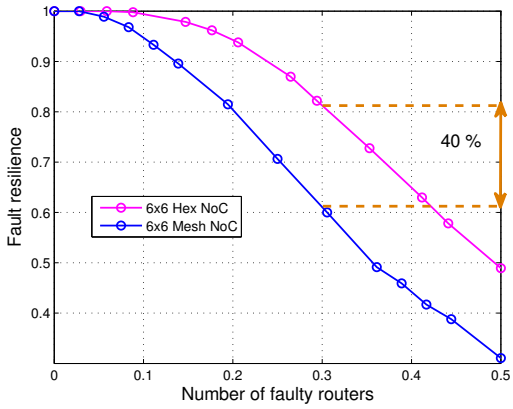
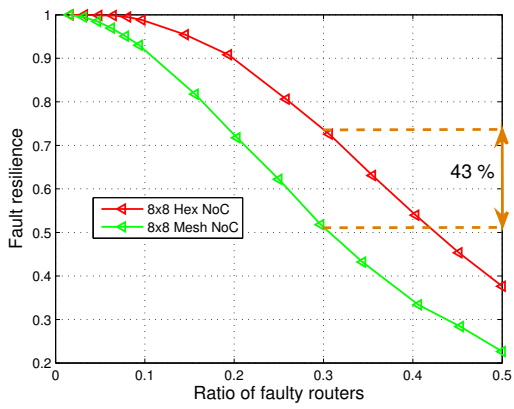


Figure 3.26: Comparison of fault resilience of 8×8 mesh and Hex and Oct NoCs with higher number of faults.

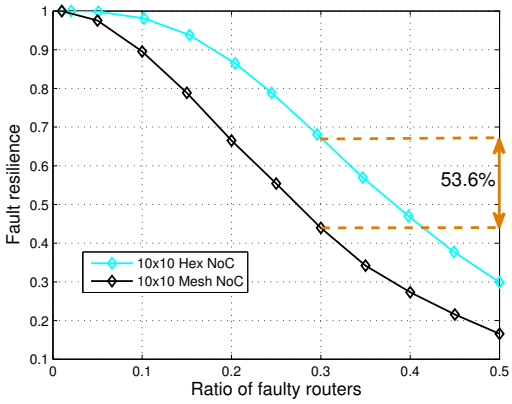
smaller size NoC has a higher fault tolerance than the larger sized NoC e.g. with 25% faulty routers, the 6×6 network has a resilience of 88.6% while 8×8 and 10×10 NoCs have a resilience of approximately 83% and 77% respectively. This is due to the fact that for the same percentage of faulty routers, the bigger network has a greater number of faulty routers and also has a longer average path length. As a result, while traversing through the network, flits have a higher probability of encountering a faulty router in the larger network. This is analysed further in chapter 4 (sec.4.6).



(a) 6×6 NoC size.



(b) 8×8 NoC size.



(c) 10×10 NoC size.

Figure 3.27: Fault resilience comparison for different NoC sizes.

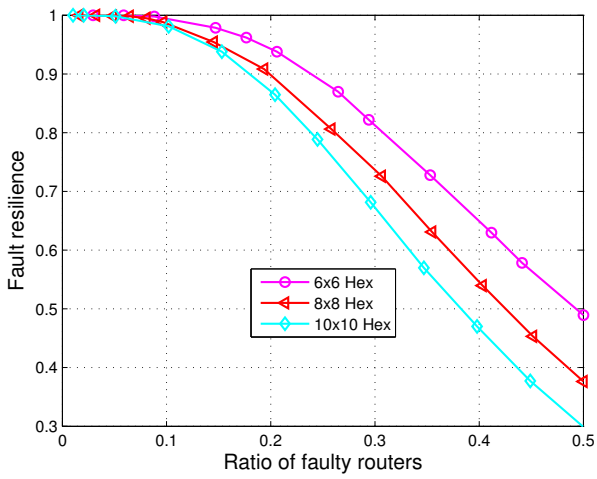


Figure 3.28: Comparison of fault resilience of different sized Hex NoCs at different ratio of faulty routers.

3.7 Summary

In this chapter we investigated spatial redundancy for NoC resulting in the hex and oct topologies. We can summarize the following points:

- In addition to extra links, we must use efficient deadlock-free adaptive routing algorithms to cope with permanent failure in NoC.
- To avoid deadlock, we apply the turn model to the hex and oct NoCs. To simplify the process of appropriate turn selections, we propose and use matrix algebra of the CDM to determine the combination of turns in the CW and CCW directions.
- The hex NoC and oct NoC are able to tolerate any position of two and three faulty routers, respectively whereas the mesh can tolerate only one faulty router.
- As network size increases, there is a higher relative improvement of fault tolerance of the hex and oct in comparison to the mesh.

Chapter 4

Analytic Assessment of Fault-tolerant Routing

4.1 Introduction

The general approach for analyzing the performance of adaptive routing algorithms for NoC is by lengthy cycle-accurate simulations, as done by us in the previous chapter (Chapter 3). Usually done for hundreds of thousands of cycles (depending on the network size), this can be very time consuming especially for large networks. To overcome this, in this chapter we propose an approach to evaluate analytically the network resilience with fault-tolerant Negative-First adaptive routing algorithm for the mesh and hexagonal NoCs. The analytic approach determines whether a path connects a source-destination pair as provided by the algorithm, in the presence of any number of random router faults. An analytic approach is very useful as we have more flexibility for the design space exploration concerning NoC reliability, for different topologies or routing algorithms or with large sized NoCs. Even when we have an already existing simulator, the implementation of different routing algorithms or topologies can also take some effort. Moreover, using the analytic approach we can also explore specific situations such as e.g. whether faults in a certain of the NoC (e.g. along the border) have greater effect 4.6 on the fault tolerance.

4.2 Related Works

Most investigations of fault-tolerant routing for NoC, such as [IY11][FFH09] are based on cycle-accurate simulations. Due to the challenges of accurately modeling adaptive routing algorithms, there are fewer works in this category. An analytic model was given in [VMS08] to evaluate the reliability of XY and XY-YX mesh NoC based on the average path length calculation.The authors also propose extensions of their analysis to adaptive routing algorithms such as the west-first routing algorithm. A fault-tolerance analysis of

different NoC architectures was presented in [LLP07] based on reliabilities of the different components such as the routers and NoC interfaces (NI). All combinations of different number of inoperable routers and NIs were investigated and the overall system reliability was obtained by summing up the reliabilities of all the combinations. With simple source-based routing, it was found that network structures built from simple 3-port routers provided better fault tolerance than those based on more complex multi-port routers. Refan et.al.[RAS+08] determine NoC reliability for application specific traffic with XY routing, when considering router failures. The path reliability for a source-destination pair is obtained by considering the product of the reliabilities of all routers in the path. The dependence of Through-silicon-via (TSV) failures on 3-D NoC reliability has been explored by analytic method in [KYEB15]. There the authors modeled the TSV characteristics as a time-invariant failure probability and quantified the relationship between NoC reliability and TSV failure. Thus, very few SOTA works have attempted to assess analytically the reliability of fault-tolerant routing algorithms and even the few works were based on limited assumptions of faults. Our approach (presented in [MF17]) is quite flexible in that it can be applied to any algorithm based on the turn model, regardless of the topology and is not limited to a fixed number of faults .

4.3 Adaptive routing and turn model

The adaptive routing algorithms whose reliability are assessed analytically in this chapter are based on the turn model, which was explained in details in Sec. 2.1.2.2. The algorithms have been introduced in Chapter 3.

For the mesh, a total of four turns in the CW direction or in the CCW direction contribute to cycles, so that one turn must be prevented in CW and CCW directions to prevent deadlock. Accordingly, three different adaptive routing algorithms are possible for the mesh 2.1.2.2. Of these we concentrate on the Negative-First routing algorithm [GN94] in which any turn from the positive to the negative directions are prevented i.e any turn from north or east directions to the west or south directions. A fault tolerant version of this routing algorithm was presented in [GN93] and is described briefly in Table 4.1, which shows the possible directions for a flit depending on its initial direction of travel as well as the direction of its intended destination. If we take the first row as an example, the entries show the output directions for a destination to the NE (relative to the current router). For such a case, the resulting direction is always to the North or East, however, between the two possibilities, the chosen direction should not bring the current and destination along one line (in the vertical or horizontal direction) for as long as possible. The algorithm considers router faults only and is modeled with our analytic approach in the following section 4.5. It should be noted that for both the mesh and hex, we consider the algorithm with a certain level of non-minimality although further non-minimal versions are possible.

Application of the turn model to the the hexagonal NoC, created by addition of diagonal

Table 4.1: Fault tolerant Negative-First Routing algorithm [GN93]

	Incoming flit direction				
Destination	N	E	S	W	local
NE	N,E*	N,E*	N,E*	N,E*	N,E*
E	E	E	S,E	-	S,E
N	N	N	-	W,N	W,N
NW	-	-	W,S	W,S	W,S
SE	-	-	S,W	S,W	S,W
S	-	-	S,W	S,W	S,W
W	-	-	W,S	W,S	W,S
SW	-	-	W,S	W,S	W,S

*Select so that source and destination are not in one line

links in the mesh NoC produces similar fault-tolerant algorithms, as described in the chapter3 and given in Table 3.4.

4.4 System Model and Matrix Algebra

4.4.1 Connectivity matrix

Graph theory is a very useful tool for description and property evaluation of networks. In general, a network consisting of N vertices or nodes can be represented by a graph $G = (V, E)$, where V is the set of vertices (v_1, v_2, ..., v_N) and E is the set of edges connecting the vertices. If the connections between the nodes are directed, the graph is called a directed one. The adjacency matrix is an $N \times N$ matrix $A = (d_{ij})$ in which $d_{ij} = 1$ if there is an edge connecting vertices v_i and v_j, otherwise this value is 0. Powers of the adjacency matrix gives the number of available paths between pairs of nodes. Thus, as given in [B+93], the ij-th entry of A^2 gives the number of paths from v_i to v_j using 2 hops:

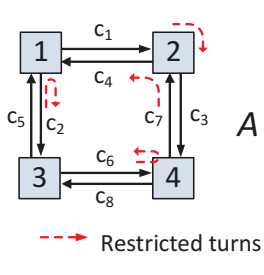
$$(A^2)_{ij} = \sum_{k=1}^{N} A_{ik} \cdot A_{kj}. \tag{4.1}$$

Continuing with higher powers of A, $(A + A^2 + A^3)_{ij}$ would give the total number of paths from v_i to v_j using one, two or three edges or hops. Similarly, the following infinite series can be used for the presence of any path connectivity between any node pair, using any number of hops:

$$A_{conn} = A + A^2 + ... + A^n + A^{n+1} + ... \tag{4.2}$$

The matrix A_{conn} is referred to as the *node connectivity matrix*. A_{conn}, as given above, cannot be used to give the connectivity of a wormhole switched network such as the NoC,

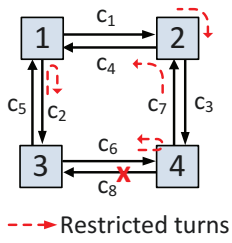
since it does not take into account the turns restricted for deadlock prevention. Instead we consider the channel adjacency matrix also called the channel dependency matrix(CDM), which was introduced in Sec. 3.4.1.1. The CDM for 2×2 mesh NoC with Negative-First routing was depicted in Fig. 3.13, and is also presented here for reference. Please note that in the following discussions, A represents the CDM (and not the node adjacency matrix) and A_{conn} represents the channel connectivity matrix, consisting of the summation of the powers of A.



$$A = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{array} \begin{array}{c} \begin{array}{cccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 4.1: The channel dependency matrix (CDM), A for 2×2 Mesh NoC with Negative-First routing.

In Fig.4.2, the same CDM is shown when a fault occurs on one of the link e.g. c_8. In such a scenario, all dependencies to and from this link are made equal to 0 in the CDM.



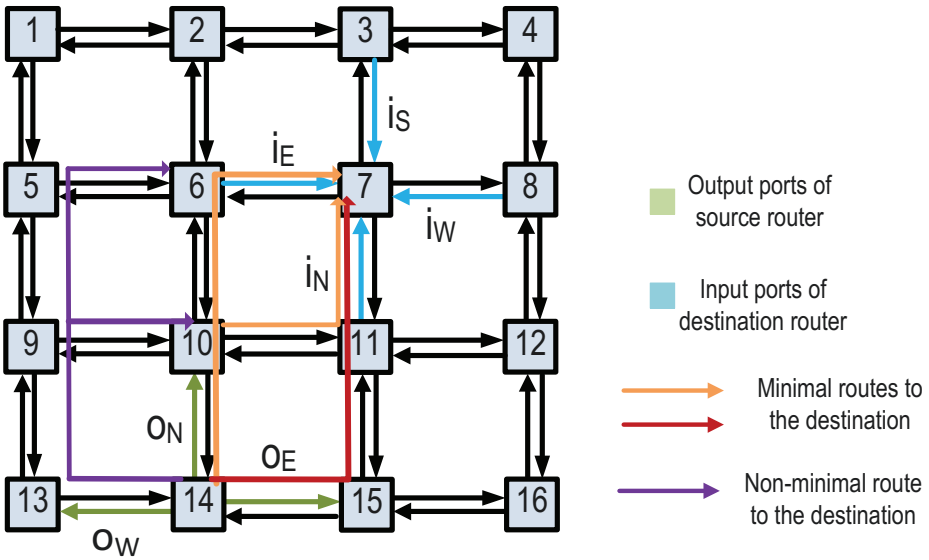
$$A = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{array} \begin{array}{c} \begin{array}{cccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 4.2: CDM with for 2×2 Mesh NoC with Negative-First routing and a permanently faulty link.

4.4.2 CDM Algebra with adaptive routing

Considering the CDM, A^n_{conn} gives the number of paths from a channel c_i to channel c_j via $n+1$ channels taking into consideration the restrictions for avoiding deadlock. It should be noted that A^2 would mean the number of paths using 3 channels or hops and similarly A^n refers to $n+1$ hops. To get the number of paths from a node or router to another node, the entries of the matrix from all the output channels of the source node to all the

input channels of the destination router have to be considered in the calculation. A small example for a 4×4 mesh NoC is given in Fig. 4.3.



$$A^2 = \begin{array}{c|cccc} & i_S & i_E & i_W & i_N \\ \hline o_N & 0 & 1 & 0 & 1 \\ o_E & 0 & 0 & 0 & 1 \\ o_W & 0 & 0 & 0 & 0 \end{array} \qquad A^4 = \begin{array}{c|cccc} & i_S & i_E & i_W & i_N \\ \hline o_N & 0 & 0 & 0 & 0 \\ o_E & 0 & 0 & 0 & 0 \\ o_W & 0 & 3 & 0 & 3 \end{array}$$

$$A+A^2+A^3+\ldots = A^2+A^4 = \begin{array}{c|cccc} & i_S & i_E & i_W & i_N \\ \hline o_N & 0 & 1 & 0 & 1 \\ o_E & 0 & 0 & 0 & 1 \\ o_W & 0 & 3 & 0 & 3 \end{array}$$

Figure 4.3: Powers of the CDM, A and the channel connectivity matrix, A_{conn} with Negative First routing.

Since the entire CDM is too large (being of size 48×48), only a portion of A^2 having the corresponding output and input ports of the source and destination routers respectively are shown. Here the source is node 14 and destination is node 7. The output and input channels with their directions are shown. Here, we do not show A since for this source-destination pair, with a minimal distance of 3 hops, the entries of A between the output of the source and input channels of destination would be 0. As the destination is to the North-East of the source, both the east (o_E) and north (o_N) output ports can be used adaptively to reach destination using minimal paths. Alternatively the west port (o_W) could also be used, but with a higher number of hops. If o_E is taken, only 1 path exists

whereas using o_N there is greater path diversity as two alternate paths exist. Similarly using the o_W, greater path diversity is present but with higher number of hops.

If a router is permanently faulty, then all the dependencies from and to the output and input channels respectively of the faulty router are made equal to zero. As an example, if the router 11 fails completely, then A($\overline{o}$,:)=0 and A(:,$\overline{i}$)=0, where $\overline{o}$ and $\overline{i}$ are the group of output and input ports of the faulty router respectively. The effect on the path connectivity of node 14 to node 7 can be seen in Fig. 4.4, where node 7 is reachable only via i_E.

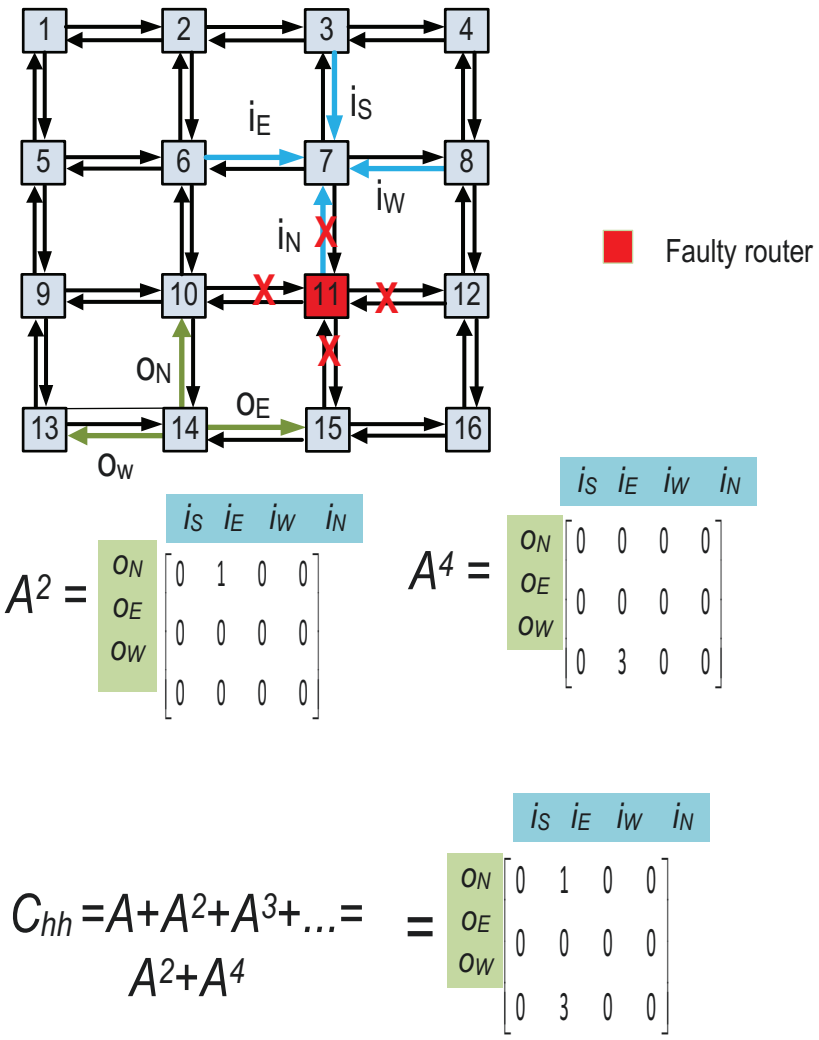


Figure 4.4: Effect of faulty router on the channel connectivity Matrix, A_{conn}.

4.4.3 Proof of fault tolerance via CDM Algebra

In graph theory, network connectivity is measured by its invulnerability to deletions (edges or nodes/routers) i.e. how many nodes or edges can fail without cutting off communication.

In 1927, Menger showed that the connectivity of a graph is related to the number of disjoint (or non-overlapping) paths joining distinct vertices in the graph: *A graph G is k-connected if and only if every pair of vertices are joined by k pairwise internally disjoint paths.*

If we look at Fig.4.5, between s and t, we have

- G1: 1 vertex disjoint path
- G2: 2 vertex disjoint paths
- G3: 3 vertex disjoint paths
- G4: 4 vertex disjoint paths

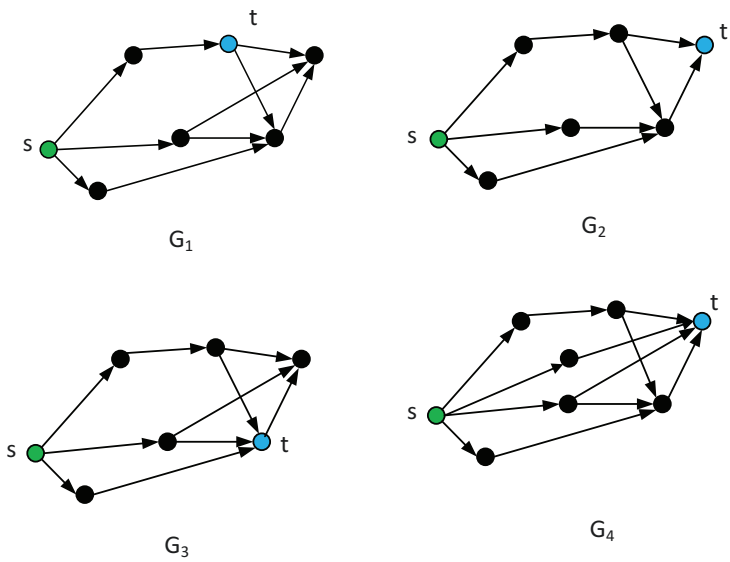


Figure 4.5: Concept of disjoint paths.

By taking powers of the channel dependency matrix i.e. CDM, we can obtain how many paths exist between the 's' and 't'. Consider a 4x4 hex NoC as shown in Fig.4.6. Between nodes, s and t if we consider paths of length 3 hops, 4 hops and 5 hops, corresponding to A^2, A^3 and A^4, respectively, we obtain:

$$A^2 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad A^3 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \end{array} \quad A^4 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 6 & 3 & 3 \end{pmatrix} \end{array}$$

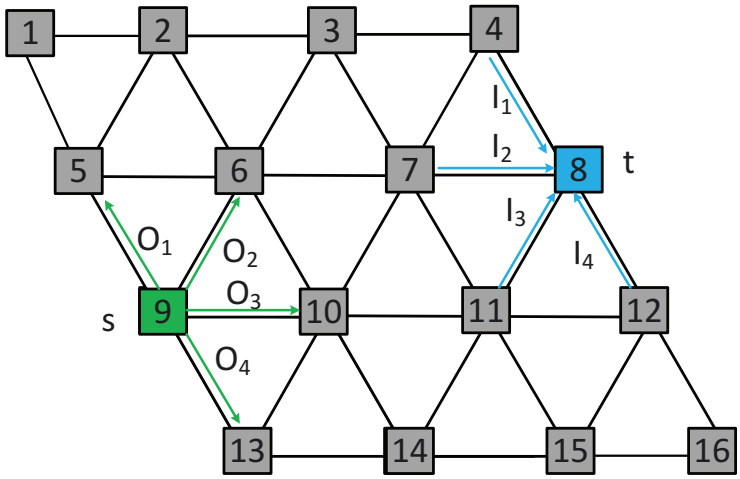


Figure 4.6: Disjoint paths between s=9 and t=8.

We can see that with 4 and 5 hops, there are 3 possible output ports via which the flit can leave the source and 3 possible input ports (I_2, I_3, I_4) via which it can reach the destination. Starting with O_3 or O_4, there are paths leading to the destination via input ports I_2, I_3 and I_4. To determine if the paths shown in A^3 and A^4 are node-disjoint, we can remove nodes surrounding t (the most critical nodes). The aim is to determine whether s and t still remains connected. If we delete node 11 i.e., remove all the dependencies to and from the links of node 11 from A and then calculate A^2, A^3 and A^4, we obtain:

$$A^2 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad A^3 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \quad A^4 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 1 \end{pmatrix} \end{array}$$

We now observe that, there is now only 1 surviving path via O_3 to I_1, which tells us that some of the paths from O_3 to the destination had common nodes and therefore were not node disjoint. Similarly, starting from O_4, there are still at least 2 possible ways to reach the destination via ports I_2 and I_4. Note that this also tells us that, O_4 provides the most path diversity (although at the cost of increased hop count) to the destination. When designing the fault-tolerant routing algorithm, the goal is always to choose the path with the highest path diversity so that packets have the greatest chance of reaching the destination.Thus, for this case, O_4 would be the first choice, then O_3 if O_4 is unavailable due to fault and the final choice is O_2

If we were to observe another set of node pairs, e.g s=14 and d=8 in Fig.4.7, and look at the powers of the CDM:

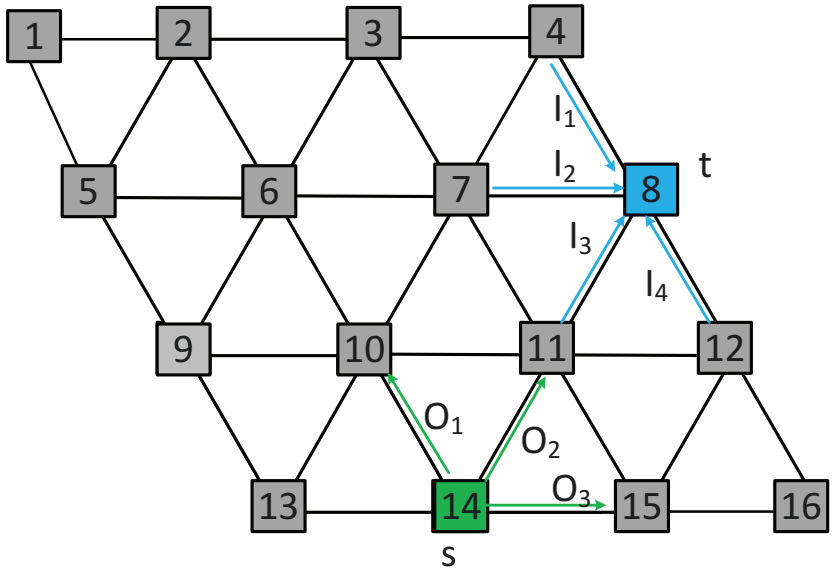


Figure 4.7: Disjoint paths between s=14 and t=8.

$$A = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad A^3 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Since, taking O_2 involves a single hop (as seen in A), this should be the first choice. If, however, router 11 were to be permanently faulty, then we obtain: $A =$

$$\begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad A^2 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \quad A^3 = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

A^2 and A^3 shows us that there are exactly 2 different paths of 2 hops leading from O_1 to I_2 and from O_3 to I_4. Since both paths involve an exclusive path, either one must be chosen. Thus, in order to be able to tolerate another fault in its path, the source must have further knowledge of neighbor fault status beyond its immediate neighbors. This was assumed to be the router in E-NE direction (i.e. router 12 in Fig.4.7) in the algorithm presented in Chapter 3, Algorithm 1.

4.5 Analytic Model for NoC Routing Connectivity

Analytically modeling node connectivity when using a particular routing algorithm in the presence of unlimited faults in the NoC is a challenging task due to the complexity

of adaptive routing. Even when a physical path may exist, as allowed by the deadlock freedom constraint, whether a flit will actually reach the destination will depend upon whether the routing algorithm is able to find this path. NoC routers do not have a global view of the faults in the network. To keep the overhead to a minimum, a router usually has knowledge of faults of only the adjacent connected neighbor routers. As a result, in the presence of multiple faults and with distributed routing, it is not always possible to calculate the connectivity directly using the channel adjacency matrix.

Table 4.2: Overview of symbols used in analytic model

s	Source node
d	Destination node
q	Intermediate node
o_H	Output port of source node in H direction
i_t	Input ports of router t
A_{conn}	Channel connectivity matrix=$\sum_i A^i$
Δx	x-distance between source and destination
Δy	y-distance between source and destination
Δq	distance between source and intermediate node 'q'
f_H	Fault condition of output port in H direction
$P_{u,v}$	Path connectivity from node u to node v

For this reason, we break down the reliability assessment in two parts, first by determining whether the source node is connected to an intermediate node (called 'q' in the following). If this intermediate node is connected to the destination, then the source under consideration is also connected to the destination. The location of the intermediate node depends upon the routing algorithm. For certain destination directions e.g those to the E, N and for some destinations to the NE, we can directly determine the connectivity, as shown in Fig. 4.8. For destinations to NW, SE, S, W and SW, we have to consider the intermediate node approach. This is because, e.g. to go to the NW, a flit should go first in the W or S direction until the destination is to the N of current router and then follows the rules for N-routing. Closer s-d pairs are calculated first and then farther s-d pairs. Consequently, the calculation time taken by the analytic approach will increase with the network size. However, it is still significantly less than that with simulation. Moreover, although we

consider the negative-first routing algorithm only, other turn model algorithms can be similarly modeled by this approach. A short overview of the parameter symbols used in the analytic model is given in Table 4.2.

4.5.1 Mesh Negative first fault-tolerant routing algorithm [GN93]

Destination to the NE of source

In this case, routing is done along the N and E adaptively towards the destination. If $\Delta x = 1$ or $\Delta y = 1$ (as shown in Fig. 4.8 (a),(b)), we can directly calculate the node connectivity from the connectivity matrix, A_{conn} by summing over the matrix entries

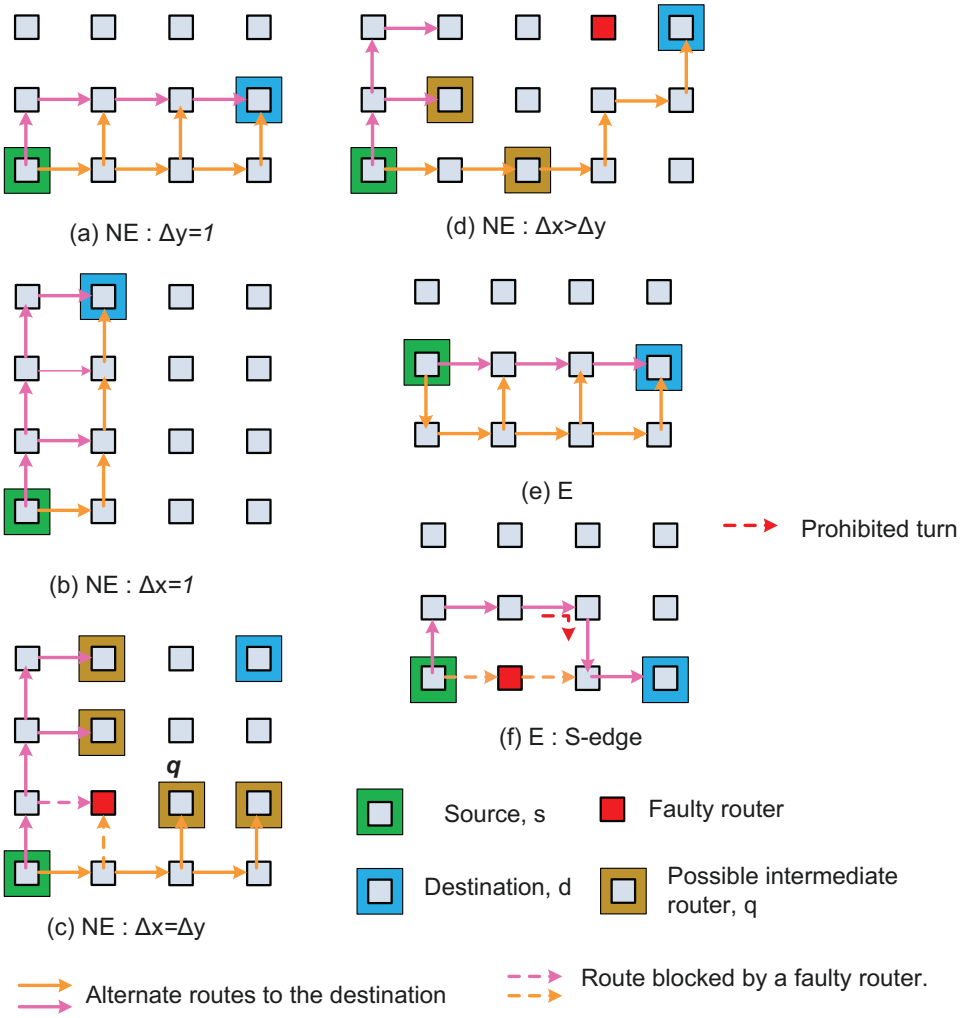


Figure 4.8: Routing to the NE.

from o_N and o_E to the input ports, i_d:

$$P_{s,d} = \sum_{i_d} \sum_{t=o_N,o_E} A_{conn} t, i_d. \tag{4.3}$$

If $\Delta x = \Delta y$, the E port is the preferred output port to the destination. If the E port is faulty (f_E), then the N port is used. If $\Delta x = \Delta y \leq 3$:

$$P_{s,d} = \sum_{i_d} A_{conn} o_E, i_d + f_E \cdot \sum_{i_d} A_{conn} o_N, i_d. \tag{4.4}$$

When $\Delta x = \Delta y > 3$ (Fig.4.8(c)), if the E port is non-faulty then the first node to the East-North that is reachable from s is the intermediate node q. For this condition, i.e. to reach q, which is Δ q hops away from s, we should have $A^{\Delta q-1}(o_E, i_q) > 0$, since $A^{\Delta q-1}$ refers to $A^{\Delta q}$ hops (explained in Sec.4.4.1). If E-port is faulty, then the N-port is selected:

$$P_{s,d} = \sum_{i_d} A_{conn} o_E, i_q \cdot P_{q,d} + f_E \cdot \sum_{i_q} A_{conn} o_N, i_q \cdot P_{q,d}. \tag{4.5}$$

This equation is also applicable to when $\Delta x > \Delta y \neq 1$ (Fig. 4.8(d)). Here, the flit moves first to the E until $\Delta x = \Delta y$, then the rules for this situation is followed. Thus q is the farthest node that can be reached ($A^{\Delta q-1}(o_E, i_q) > 0$) in the E direction without encountering a fault until $\Delta x = \Delta y$. If E port is faulty, then the first node in the North East direction to be reached is q. If $\Delta x < \Delta y$ instead, then N direction is the preferred direction.

Destination to the E or N of source

For destinations to E, if s and d are not along the south edge, then a routing to the S is done and then rules of NE-routing are followed (Fig. 4.8(e)), involving a total of $\Delta x + 2$ hops. If S-port is faulty, then E-port is used to reach the destination in Δx hops. Thus, the reachability of d can be determined by:

$$P_{s,d} = \sum_{i_d} A^{\Delta x+1}_{o_S,i_d} + f_S \cdot \sum_{i_d} A^{\Delta x-1}_{o_E,i_d}. \tag{4.6}$$

If s and d are along the south edge and a faulty router blocks the path, then the flit is routed one hop perpendicular to the edge (Fig. 4.8(f)) to reach the destination using $\Delta x + 2$ hops. Although, a restricted turn (E-to-S) is taken, no deadlock occurs as the cycle through a faulty edge router cannot be completed. The reachability of d is given by:

$$P_{s,d_S-edge} = \sum_{i_d} A^{\Delta x-1} o_E, i_d + f_E \cdot \sum_{i_d} A^{\Delta x+1} o_N, i_d. \tag{4.7}$$

Similarly for destinations to the N, the following expressions are applicable:

$$P_{s,d} = \sum_{i_d} A^{\Delta y+1}_{o_W,i_d} + f_W \cdot \sum_{i_d} A^{\Delta y-1}_{o_N,i_d}. \tag{4.8}$$

$$P_{s,d_W-edge} = \sum_{i_d} A^{\Delta y-1} o_N, i_d + f_N \cdot \sum_{i_d} A^{\Delta y+1} o_E, i_d. \tag{4.9}$$

Destination to the S or W or NW or SE or SW of source

For destinations in these directions, the first moves are always in the negative direction (to the W or S), so that there is more flexibility in selecting an output port. For example if we consider destination to the South, first the S port is selected and if this is faulty, then the W port is selected, as shown in Fig. 4.9(a). Thus, if S port is non-faulty, 'q' is the south-most router that is possible for the flit to reach $(A^{\Delta q-1}(o_S, i_q) > 0)$ along the line from the source to the destination. From there the calculation of a path to the destination is then done:

$$P_{s,d} = \sum_{i_d} A_{conn} o_S, i_q \cdot P_{q,d} + f_S \cdot \sum_{i_q} A_{conn} o_W, i_q \cdot P_{q,d}. \tag{4.10}$$

A similar equation is used for destinations directly to the W. If the destination is to the NW Fig.4.9(b), then the flit is routed first to the west or south until the destination is directly to the N of the current router. From here the routing rules for the N are followed. Here, the intermediate router is the closest router in the west or south-west that can be reached:

$$P_{s,d} = \sum_{i_d} A_{conn} o_W, i_q \cdot P_{q,d} + f_W \cdot \sum_{i_q} A_{conn} o_S, i_q \cdot P_{q,d}. \tag{4.11}$$

For the destinations to the SE or SW, similar equations are used.

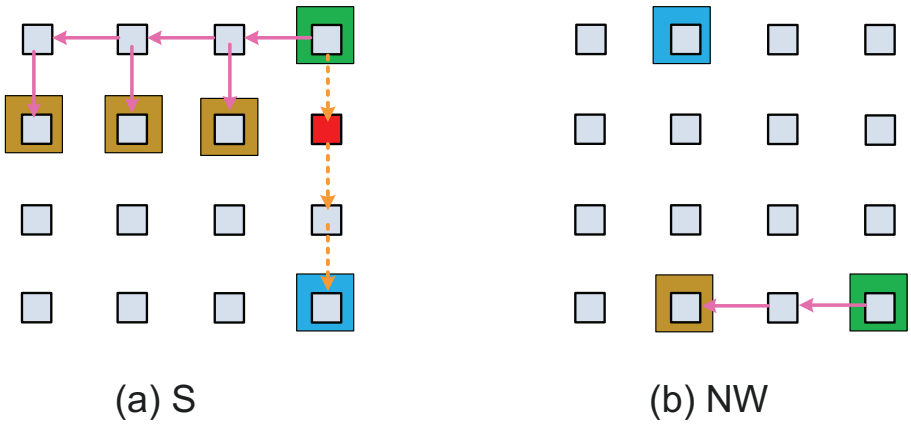


Figure 4.9: Routing to the S or NW.

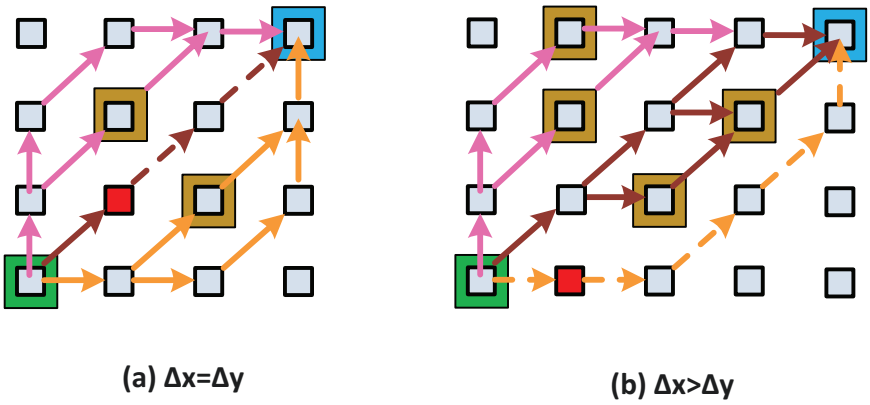


Figure 4.10: Routing to the NE in Hex NoC.

4.5.2 Hex Negative first fault-tolerant routing algorithm

For the hexagonal NoC has 3 directions in the positive direction (E,NE and N) and in the negative directions (W,SW and S). As a result, there are 3 possible directions for reaching a destination. The routing algorithm is given in Table 3.4.

Destination to the NE

For destinations to NE, $\Delta x = \Delta y$ (shown in Fig.4.10(a)), the first choice for routing the flit is to the NE and then to the E or N.

If $\Delta x = \Delta y \leq 3$, then

$$P_{s,d} = \sum_{i_d} \sum_{t=o_{NE},o_E,o_N} A_{conn} t, i_d. \tag{4.12}$$

When $\Delta x = \Delta y > 3$, if $A^{\Delta x-1}_{o_{NE},i_d} < 0$, then the last node in NE to be reached is q. If the NE router is faulty, E port is taken and the first node to the E-NE that is reachable from s (i.e.$A^{\Delta q-1}_{o_E,i_q} > 0$) is the intermediate node q. If E-port is faulty, the N-port is selected:

$$P_{s,d} = \sum_{i_q} A_{conn} o_{NE}, i_q \cdot P_{q,d} + f_{NE} \cdot \sum_{i_q} A_{conn} o_E, i_q \cdot P_{q,d} + f_{NE} \cdot f_E \cdot \sum_{i_q} A_{conn} o_N, i_q \cdot P_{q,d}. \tag{4.13}$$

When, $\Delta x > \Delta y > 1$, then the first choice is to the E until $\Delta x = \Delta y$, after which the rules for $\Delta x = \Delta y$ is followed. Thus, the intermediate node q is the furthest node that can be reached in the E direction until $\Delta x = \Delta y$. If the E router is faulty, then the next choice is NE and the q is the furthest router in the NE-E direction that can be reached, as shown in Fig.4.10(b). If NE router is also faulty, the intermediate node will be the furthest router in the N-NE direction.

$$P_{s,d} = \sum_{i_q} A_{conn} o_E, i_q \cdot P_{q,d} + f_E \cdot \sum_{i_q} A_{conn} o_N E, i_q \cdot P_{q,d} + f_{NE} \cdot f_E \cdot \sum_{i_q} A_{conn} o_N, i_q \cdot P_{q,d}. \tag{4.14}$$

For $\Delta x < \Delta y$, a similar equation is used, except that the first choice is to the N, then to NE and then to the E.

4.5.3 Performance Evaluation

For comparing the result of the analytic approach, we carried out simulations for over $100,000$ to $200,000$ cycles in a cycle accurate simulator [WF08] (C/C++ based). Fault-tolerant negative-first routing algorithm was implemented for both the mesh and hexagonal NoC. The traffic considered was uniform random. Permanent router faults were injected into the NoC at random locations over $10,000$ iterations. Fault-resilience was determined as the average ratio of the number of successfully received flits to the total number of injected flits. Similarly, using the analytic model the network fault-resilience i.e. the average node connectivity was calculated over $10,000$ iterations of random fault locations and compared to the simulation results. Fig.4.11 shows the fault-resilience vs. the percentage of faults for different NoC sizes of 6×6, 8×8, and 10×10 were simulated using both the cycle-accurate simulator and the model. As can be seen, the model results match closely that of the simulation. The model is able to calculate the fault-resilience with error of about 1%.

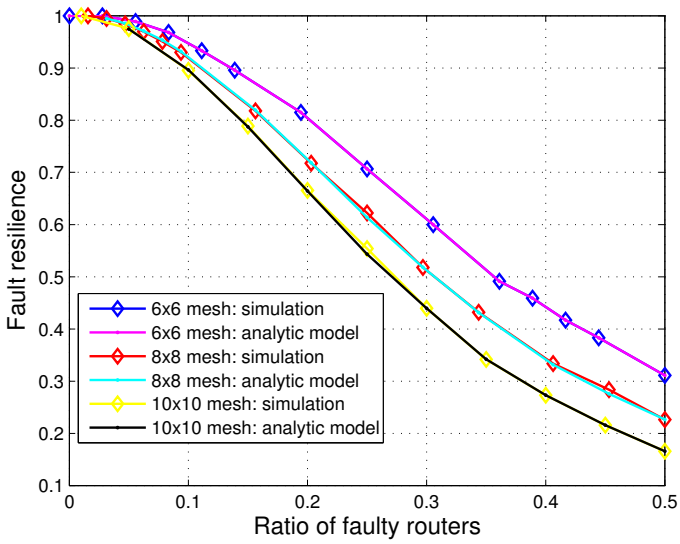


Figure 4.11: Results of fault resilience vs. percentage faulty routers for mesh NoC in comparison to cycle-accurate simulation.

Similar curves were plotted for the hexagonal NoC with negative-first fault tolerant routing. The results, depicted in Fig.4.12 illustrate again the accuracy of the analytic approach.

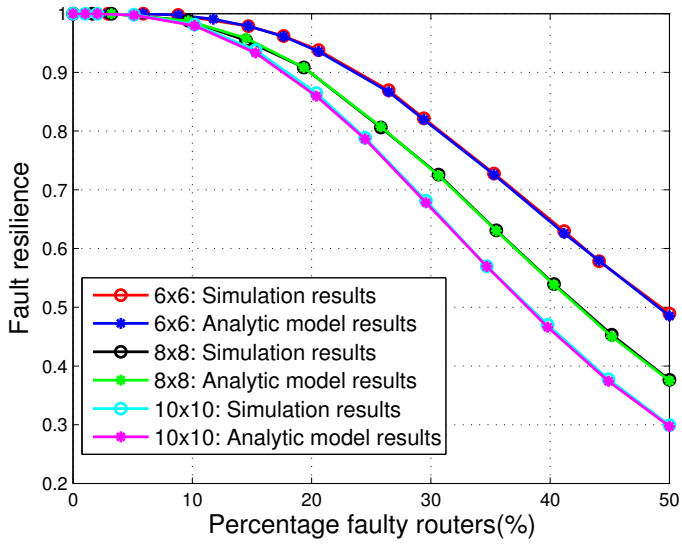


Figure 4.12: Results of fault resilience vs. percentage faulty routers for Hex NoC in comparison to cycle-accurate simulation.

The duration of the cycle-accurate simulations became excessively long as the network size increased. For the 10×10 network, the analytic approach was 70× faster than the simulator. As a result, it was significantly faster to use the analytic approach for the reliability assessment. However, the speed of the analytic model is still dependent on the size of the NoC i.e. $O(N)$, where N is the number of nodes. Nevertheless, it is still significantly faster than the simulations so that we can still determine the fault tolerance with reasonable speed.

To obtain an idea of the reliability of NoCs with large number of cores, we determined the fault-resilience using the analytic model for a network of 256 cores, for both mesh and hex topology. The results shown in Fig.4.13, show that for 15% faulty routers, the hex NoC with a fault resilience of 0.877 is 29% more resilient than the mesh NoC.

The results show that as the NoC size increases, the fault resilience quickly becomes a critical even at lower fault rates. Thus, instead of having very large 2D NoCs it may be more efficient to have clustered NoCs, as investigated in [WPG10] or even 3D NoCs.

4.6 Analysis of border effect

It was observed in Fig. 4.11 and in Fig.4.12 that the as the size of the network increases, the network resilience decreases for the same percentage of faulty routers. This is the

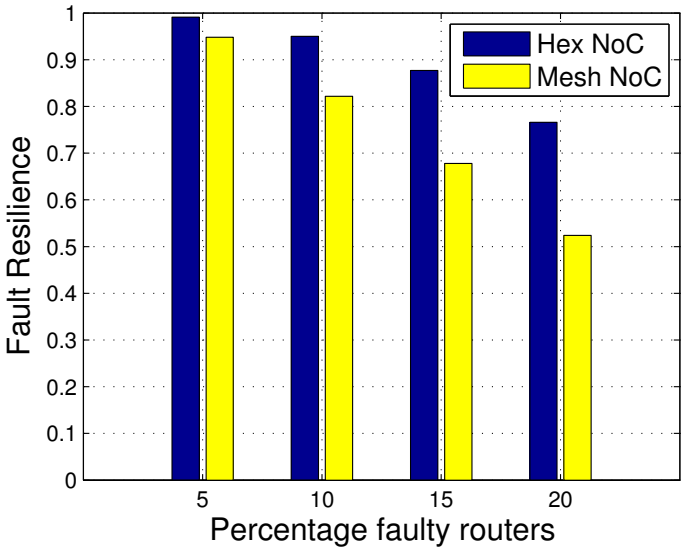


Figure 4.13: Comparison of fault resilience of 16×16 mesh and hex NoC.

consequence of the higher average path length in the bigger network, which increases the probability of encountering a fault. Another reason could be due to the number of routers along the edges of the mesh or hex NoC, so that as the size of the NoC increases the ratio of center to border routers increases (0.47 for 6×6 hex to 0.65 for 10×10 hex). Since these edge routers have lower number of links than the center routers, if these become faulty, lower number of links are made faulty. As a result, fewer paths are affected. To analyze whether the effect of the border nodes was dominant in the reduction of NoC resilience for bigger NoC sizes, we used the analytic model for calculating the fault resilience of 8×8 and 10×10 hex NoCs, excluding any faults along the edges of the NoC. Results for 10,000 random fault locations were averaged and are depicted in Fig.4.14.

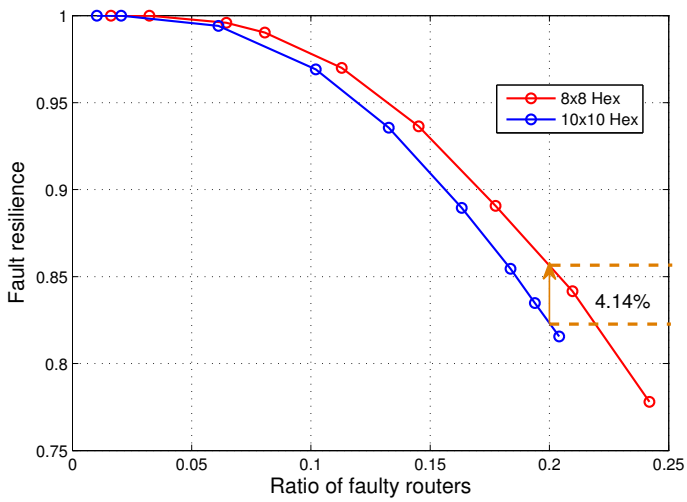


Figure 4.14: Fault resilience excluding faults along the border.

In this case, at 20% router faults, the 8×8 NoC (which has a lower ratio of center routers) has a 4.17% higher fault tolerance than the 10×10 NoC. If we look at Fig.4.12, we see that at 20% faults, the 8×8 NoC has 4.14% higher reliability than the 10×10 NoC. Thus, we see that the effect of inner network node faults on the difference of resilience between the NoC sizes is actually negligent. It is rather the effect of increasing average path length: 6.636 for 10×10 NoC and 5.480 for 8×8 NoC. The bigger NoC has a greater average path length, which makes it more prone for a flit to encounter faults along its path. We see this in more details in Fig.4.15. Here, we consider a 8×8 mesh NoC and look at a scenario of two faulty routers affecting two different source-destination pairs (both in the NE direction), namely the pairs (S-D1) and (S-D2) having path lengths of 6 hops and 8 hops respectively. Of all possible locations of two faulty routers, we highlight those that will result in the flit failing to reach the destination. The location of these faulty router positions can be easily determined from path connectivity using CDM approach. Since the source and destination are considered to be fault-free, then there are $\binom{62}{2}$ different possibilities of 2 faulty routers. Of these, for S-D1 pair there are 6 pairs of faulty routers or a probability of $6/\binom{62}{2}$ that the flit will fail to reach from S. Similarly, for S-D2 pair, there are such 8 pairs of faulty routers or a probability of $8/\binom{62}{2}$ of failure. The results support the observation that with increasing path length, there is a higher probability of encountering faults and thus decreasing fault resilience.

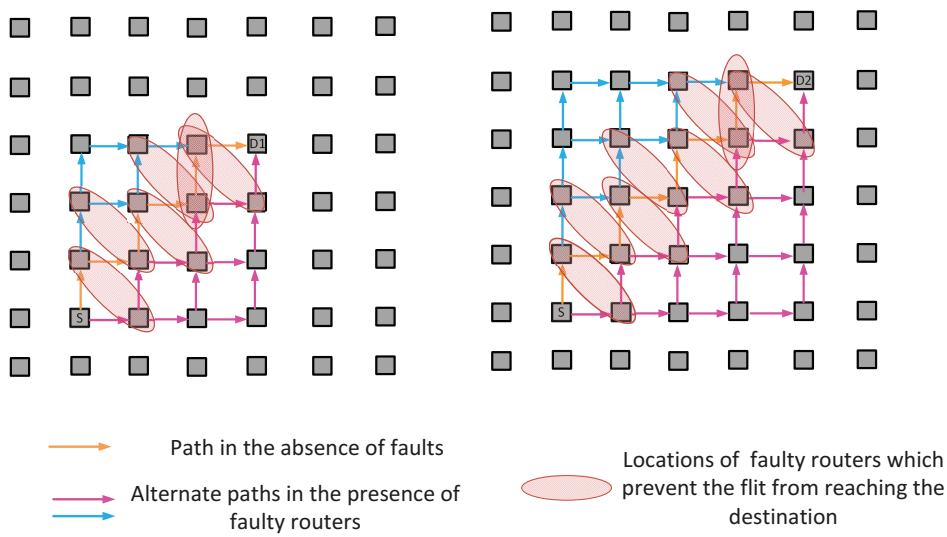


Figure 4.15: Relation between path length and fault resilience in mesh NoC.

Similarly, for the hex NoC we can compare the fault resilience of two source-destination pairs, S-D1 and S-D2 as shown in Fig. 4.16. As the hex NoC can tolerate all two-router faults, we look at a scenario of three faults affecting a 8×8 hex NoC. For S-D1 pair, with a minimal path length of 3 hops, there are 5 such fault combinations whereas for S-D2 pair with a minimal path length of 4 hops, there are 6 such fault combinations. Thus, the probability of the flit failing to reach the destination is $5/\binom{60}{3}$ and $6/\binom{60}{3}$ for S-D1

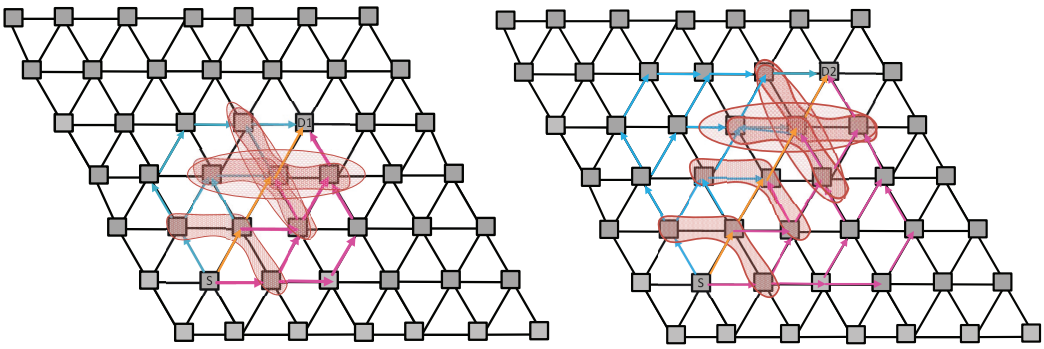


Figure 4.16: Relation between path length and fault resilience in hex NoC.

and for S-D2 respectively. Note that for the hex NoC, we do not include the top-left and down-right corner routers in the calculation, as they are not part of the true hexagonal NoC.

Now that we have observed the effect of path length on the fault resilience for a specific sized NoC having a certain number of faults, we further analyze the effect of path length for different NoC sizes with the same ratio of faults. Let us consider mesh NoCs of size 4×4 and 5×5 having 12.5% of faulty routers, i.e. 2 and 3 faulty routers for the smaller and larger NoC respectively. For the smaller NoC, for S-D pair of path length 4 hops, the probability of the flit not reaching the destination is given by $4/\binom{14}{2}$ or 4.44% and for a S-D pair with path length of 6 hops, the failure probability is $6/\binom{14}{2}$ or 6.59%. For the 5×5 NoC with 3 faulty routers, for S-D pairs of length 4 and 5 hops, the failure probability is $82/\binom{23}{3}$ or 4.63% and $126/\binom{23}{3}$ or 7.11%. Thus, for the same ratio of faults, the bigger NoC will have a lower fault resilience due to greater average path length as well as a greater probability to encounter faults.

4.7 Summary

In this chapter, we proposed an approach based on CDM algebra for analytically determining the fault resilience of a fault-tolerant routing algorithm. We presented and validated the model for the fault-tolerant negative first routing algorithms for both the mesh and hexagonal NoCs. Although, we did not present the model for the oct NoC, the model can easily be extended to this case since the approach is very flexible. Moreover, any other adaptive routing algorithm (based on the turn model) can also be easily modeled. We presented cycle-accurate simulations to compare the accuracy of the modelwhich, is able to estimate the network fault resilience with an estimation error of approximately 1%. Our approach is significantly faster than cycle-accurate simulations, making it very useful for analyzing the reliability of larger network sizes. The investigations for larger NoCs (4.13), showed that fault resilience can become a serious problem for larger NoCs. The speed of the analytic approach is dependent on the network size since we need to first calculate the

path reliability for closer node pairs and then use it in the calculation of path reliability for node pairs which are further apart. However, by parallezing the calculations for the closer node pairs, we can speed up the calculations. Moreover, with the analytic approach we can flexibly investigate specific case scenarios as in Sec 4.6, where the effect of faults the NoC border was analyzed.

Chapter 5

Temporal and Information Redundancy

5.1 Introduction

In this chapter we concentrate on achieving soft error fault tolerance by means of temporal redundancy techniques i.e. retransmissions of corrupted or lost packets [RFZJ13] and by transmission of redundant information. Retransmission, also known as Automatic Repeat Request (ARQ) has been used widely in telecommunication networks for error control and has also been investigated widely for NoC architectures [MTV+05][PNJ+06]. This can be implemented on a hop-by-hop basis (H2H) i.e. the detection and retransmission of flits happening at every hop or at each router. In contrast it can also be implemented on a end-to-end (E2E) basis between the source and destination nodes, as shown in Fig.2.9. In this work, we concentrate on and evaluate only E2E retransmissions to avoid excessive overheads of storing flits at each router, required for retransmission.

Assuming a large NoC with high flit loss probability, ARQ mechanisms are not sufficient anymore. Long path latencies along with a large number of retransmissions lead to a big drop in NoC communication performance. A promising solution for this dilemma might be provided by network coding [ACLY00], which has been introduced in Sec. 2.3. Network coding (NC) allows to increase throughput, energy efficiency, and robustness of data transmission. These benefits are achieved by the key concept of network coding to compute and send linear combinations of data packets. In view of the challenging interconnection problem for future many-core systems, the use of network coding for communication within NoCs seems to be promising. In recent years, some authors have already studied the applicability of network coding for NoCs. Indrusiak investigated the feasibility of network coding by means of the well-known butterfly-network for a multicast scenario with two receivers [Ind11]. He suggested algorithms for finding butterfly arrangements on a 2D mesh and a heuristic for the evaluation of the established butterfly. Further algorithms for the mapping of a butterfly network to a 2D mesh were suggested in [SRG12].

Duong-Ba et al. introduced a possible implementation of a network-coded NoC [DBNC11]. They suggested a router architecture and an appropriate flit structure. Vonbun et al. investigated theoretical bounds in terms of hop count improvements of network coded NoCs in comparison to classical dimension-routing NoCs [VWF+13].

Overall, these studies indicate that network coding can improve the efficiency of communication within NoCs. However, the existing results consider 2D mesh networks of rather small dimensions (up to 12×12 mesh). Further, they focus mainly on multicast communication based on the butterfly structure. Although various coding schemes have been applied to NoC [BD05][YA12] many of these incur overheads in terms of area and power[MTV+05][FLJ+13]. We developed an analytical model for flexible evaluation of the network coded transmission which are verified by means of simulations[MYF+15]. The simulations are based on the implementations of the Student project work [Yan15]. Further, we investigated the threat of active attacks from Hardware Trojans in NoC, which is introduced and explained in details in in Sec. 5.3.

5.2 Network Coding in NoC

We focus on network coding for unicast transmissions. As network coding scheme, we assume RLNC implemented according to [CWJ03], introduced in Sec. 5.2.2.3. Figure 5.1 illustrates the benefit of a network coded unicast transmission in a NoC with flit loss. Instead of sending an initial transmission, waiting for an ARQ and sending a retransmission, additional redundant flits computed as linear combinations from the current generation are directly sent to overcome flit loss (up to a certain level). Opposed to the uncoded case, network coding has the advantage that in case of flit loss, if sufficient flits are received for decoding, no ARQ needs to be sent.

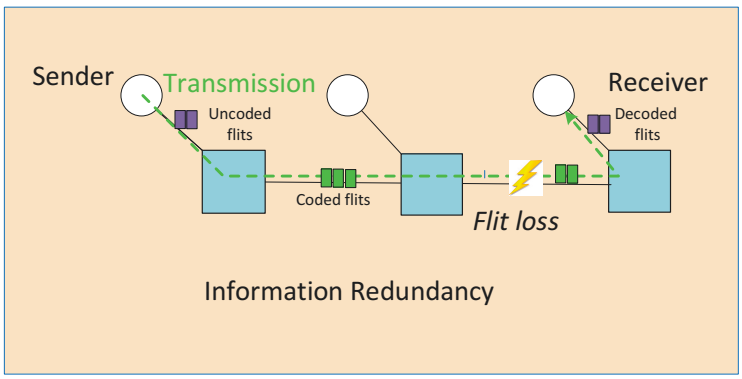


Figure 5.1: Information redundancy in NoC via network coding.

5.2.1 System Model

Traffic and Routing: A spatial uniform traffic distribution with a constant injection probability per module (i.e. Poisson arrival) is assumed. This generic traffic pattern is commonly used for NoC performance evaluation and enables a good comparison with other works. A dimension-ordered, deterministic XY routing is applied.

Coding: Uncoded and RLNC coded transmissions are considered and compared in the following. In case of RLNC, a certain number G (generation size) of subsequent flits is composed as a frame (generation). Out of these G flits, C linear combinations are generated at the sender. With a sufficient symbol size, it can be assumed that all combinations are linear independent. Thus, it is sufficient that any G out of the C generated combinations arrive at the receiver to be able to decode the original flits. The RLNC code rate R is given as

$$R := \frac{G}{C}. \tag{5.1}$$

Thus, a lower code rate indicates a higher level of redundancy.

Failure model: A certain number of N error-prone routers is randomly selected in advance. Each of these routers has a constant probability f (flit loss probability) to drop a flit instead of forwarding it to the subsequent router .

ARQ: If a flit loss is recognized at the receiver, an ARQ is sent back to the source to request a retransmission. For the uncoded case, this flit loss is recognized due to missing flitIDs, since flit numbering between a certain source destination pair are continuous. For the network coded case, all flits of generation are sent consecutively and should be received at the destination immediately one cycle after another. If there is a flit missing and the generation is still incomplete, then an ARQ is sent. For the simulations, investigations revealed that a wait of 8 cycles between consequent flits was optimum. If after this time no new combination is received, an ARQ is sent for a flit of this generation. In the uncoded case, multiple flits might be requested at once. No positive acknowledges are generated for saving bandwidth, which is very important for NoC. The maximum number of retransmission trials is limited and in the following investigations, we assume only 1 retransmission is allowed. However, this is only an assumption on our part for our investigations and the number of ARQs can also be increased to greater than 1.

Retransmission Buffer: Before encoding, copies of the flits are saved in a circular buffer of the sender, so that retransmissions can be generated after receiving ARQs. Since old flits can be overwritten by new flits, the buffer size determines the number of retransmissions that can be generated and thus affects latency and residual error rate. In the simulations a relatively large buffer size was chosen to avoid to effect of flit loss due to buffer overwriting. For practical realizations, the retransmission buffer must be dimensioned carefully. On the one hand, a large buffer allows multiple retransmission

trials over long distance paths. On the other hand, the buffer size can affect the chip area of the NoC significantly. For the network coded case, the flits of a generation are sent immediately one after another in consecutive cycles and recognition of loss happens immediately if the generation cannot be decoded . If we assume the buffer depth to be N_b, then with our assumed flit injection rate of 0.2 flits/node/cycle and with uniform random communication, each flit will stay in the buffer for $5 * N_b$ cycles on average. After this, the new incoming flit will overwrite this flit. If we assume a source-destination pair with a difference of $\Delta\ h$ hops and each hop requires 2 cycles, then the ARQ will reach the original source after a total time of $4 \cdot \Delta h + 8$. For our assumed NoC size of 8×8, the furthest nodes are 15 hops apart. For this node pair, if then the original flit is still to be found in the retransmission buffer:

$$5 \cdot N_b \geq 60 + 8 \tag{5.2}$$

Thus, a retransmission buffer of depth 14 flits is sufficient.

For the uncoded case, if 't' consecutive flits are lost, then an ARQ is sent by the receiver when the next flit arrives and the receiver realizes that there is a discontinuity of flit IDs between the last two received flits. Since there are potentially 63 potential destinations in a 8×8 NoC, the time between 2 consecutive flits to the same receiver is $\frac{63}{0.2}$ or 315 cycles, so that the ARQ is sent $315 \cdot (t+1) + 2 \cdot \Delta h$ after the first flit (which is lost) is sent. This ARQ will reach the sender (if it is not lost) after $315 \cdot (t+1) + 4 \cdot \Delta h$ cycles after the first flit was sent. Assuming, the sender has separate buffers for each target destination, each flit stays in the buffer for $315 \cdot N_b$ cycles. If successful retransmission is to occur, the original flit should be in the buffer for these many cycles, i.e.:

$$5 \cdot N_b \geq 315 \cdot (t+1) + 4 \cdot \Delta h \tag{5.3}$$

For the sender-receiver pair separated by 15 hops (Δh), if 2 consecutive flits are to be lost and still remain in the retransmission buffer when the ARQ reaches the sender, $N_b \geq 201$ i.e. 201 flits deep. For the case of 3, 4 or 5 consecutive flits lost, the retransmission should have a minimum size of 264, 327, or 390 flits, which is quite large. It should be noted this size of buffer has been calculated considering the farthest apart s-d pair. Table 5.1 gives the values of the different sizes of buffers needed with different numbers of consecutive flits lost (t) and with different hops of distance ($\Delta\ h$).

5.2.2 Performance evaluation

For the evaluations, the following four metrics were chosen as they are well suited for comparison and offer a good overview of different performance aspects in error-prone NoC:

Table 5.1: Retransmission buffer depth at sender with uncoded transmission for receivers at different path lengths.

	N_b			
Δh	t=2	t=3	t=4	t=5
5	209	272	335	398
6	213	276	339	402
7	217	280	343	406
8	221	284	347	410
9	225	288	351	414
10	229	292	355	418

Gross traffic or Network load, A: The gross traffic rate is the average number of flits per router and per cycle accepted by the network and represents the network load. Let N_{flits} be the total number of flits that are transmitted during the simulation run time, T_S the send time of the first flit, T_R the receive time of the last flit and M the total number of active modules. Then the network load can be given as follows

$$A := \frac{N_{\text{flits}}}{(T_R - T_S) \cdot M}. \tag{5.4}$$

Without ARQ and retransmission, and under the assumption that the network is not saturated, the network load is equal to injection rate.

Information rate I: The information rate represents the proportion of information flits and the total number of transmitted flits including redundant flits, ARQ flits (N_{arq}) and retransmissions (N_{retr}):

$$I := \frac{R \cdot (N_{\text{flits}} - N_{\text{arq}} - N_{\text{retr}})}{N_{\text{flits}}}. \tag{5.5}$$

Latency ℓ: The end-to-end path latency is the average time (in clock cycles) that a flit needs to travel from a sender to a receiver. For coded transmissions, it is equal to the number of cycles between sending the first flit of a generation and receiving the Gth flit of that generation.

Residual error probability ϵ: The residual error probability is the proportion of flits which could not be received after the maximum allowed number of retransmission trials.

5.2.2.1 Analytic Model for Evaluating RLNC Performance

As the size of the network increases to accommodate thousands of cores on a chip, cycle accurate simulators require a prohibitive length of time to provide results. In this respect,

analytic models become useful as they are faster and more flexible, provided that they can accurately model the system. Moreover, analytic models also provide an insight into the system which helps to explain the system behavior. In the current NoC setup without and with network coding, the four network performance parameters are evaluated by means of an analytic model that is introduced in the following. The common parameter symbols used in the model in addition to those introduced before are given in Tab. 5.7.

Table 5.2: Overview of model parameter symbols

s	Source module
d	Destination module
$\lambda_{x,y}$	Flit injection rate from x to y
$\lambda'_{x,y}$	Total flit injection rate from x to y including ARQs and retransmission
$N_{x,y}$	Number of defect routers in the XY route from x to y
$f_{x,y}$	Total flit loss probability from x to y
$\overline{\epsilon}$	Average residual error probability
$\ell_{x,y}$	Path latency from x to y
$\ell'_{x,y}$	Total Path latency from x to y after retransmission
$\overline{\ell}$	Average path latency over all source-destination pairs

In the presence of the faulty routers with a certain flit loss probability f, the total flit loss probability $f_{s,d}$ between a source and destination pair depends on how many faulty routers ($N_{s,d}$) are present in this path from s to d:

$$f_{s,d} = 1 - (1 - f)^{N_{s,d}}. \tag{5.6}$$

The total flit loss probability is used in the calculation of the different network metrics for the uncoded and RLNC coded cases, as explained in the following sections.

5.2.2.2 Uncoded Network (UC)

Gross traffic or Network load: The injection and thus network load of the network increases with ARQ and retransmission. As explained in sections 5.2.1, an error is recognized and an ARQ is triggered whenever there is a discontinuity in flit IDs of the received flits. The total injection rate $\lambda'_{s,d}$ is composed of the rate of the issued ARQs, $\lambda_{arq\_s,d}$ and the rate of the retransmitted flits, $\lambda_{retr\_s,d}$ in addition to the regular injection:

$$\lambda'_{s,d} = \lambda_{s,d} + \lambda_{arq\_s,d} + \lambda_{retr\_s,d}. \tag{5.7}$$

When computing $\lambda_{arq\_s,d}$ and $\lambda_{retr\_s,d}$, it has to be noted that an error is detected at the receiver d when 1 or more (e.g. t) consecutive flits from s to d are lost (with probability $f^t_{s,d}$) with the successive flit being received successfully (with probability $f'_{s,d} = 1 - f_{s,d}$). Since only a single ARQ is issued for all t consecutively lost flits, the rate of injection due to ARQs is proportional to $\frac{1}{t}$. Thus, $\lambda'_{arq\_s,d}$ is given by

$$\lambda_{arq\_s,d} = \lambda_{d,s} \cdot (1 - f_{d,s}) \cdot \sum_{t=1}^{\infty} \frac{f^t_{d,s}}{t} \tag{5.8}$$

Using the Maclauren series

$$ln(1-x) = -(x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \frac{x^5}{5} + \cdots) \tag{5.9}$$

in Eq.5.8 we obtain

$$\begin{aligned} \lambda_{arq\_s,d} &= \lambda_{d,s} \cdot (1 - f_{d,s}) \cdot \ln \frac{1}{1 - f_{d,s}} \\ &= \lambda_{d,s} \cdot f'_{d,s} \cdot \ln \frac{1}{f'_{d,s}}. \end{aligned} \tag{5.10}$$

The total probability of a retransmission is given by the probability 1 or more consecutive flits are lost but then the successive flit is received($f'_{s,d} \cdot \sum_{t=1}^{\infty} f^t_{s,d}$) $\times$ the probability the ARQ successfully reaches s ($f'_{d,s}$). When the ARQ is successful, then all missing flits are retransmitted, so that the rate of retransmission $\lambda_{retr\_s,d}$ is given by

$$\begin{aligned} \lambda_{retr\_s,d} &= \lambda_{s,d} \cdot f'_{s,d} \cdot f'_{d,s} \cdot \sum_{t=1}^{\infty} f^t_{s,d} \\ &= \lambda_{s,d} \cdot f'_{s,d} \cdot f'_{d,s} \cdot (\frac{1}{1 - f_{s,d}} - 1) \\ &= \lambda_{s,d} \cdot f'_{d,s} \cdot f_{s,d}. \end{aligned} \tag{5.11}$$

Putting Eqs. 5.8 and 5.11 in Eq. 5.7 and taking the average over all modules, the network load is computed as follows:

$$A = \frac{1}{M} \cdot \sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} \lambda'_{s,d}. \tag{5.12}$$

Information rate: Using the above mentioned injection rates and putting them in Eq. 5.5 with code rate $R = 1$, the information rate I is obtained as follows:

$$I = \frac{M(M-1)}{\sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} (1 + f'_{d,s} \cdot \ln \frac{1}{f'_{d,s}} + f'_{d,s} \cdot f_{s,d})}. \tag{5.13}$$

Latency: In the absence of faulty routers, the path latency $\ell_{s,d}$ is composed of the NoC interface injection and ejection delays and the total router transport delays. With retransmissions, the path latency now increases by the round trip delay, $\ell_{Rs,d}$ $(= 2 \cdot \ell_{s,d} + 2)$ needed by ARQ and the retransmission to travel to and back from s. As the ARQ and retransmission must be first stored in a buffer and transmitted 1 cycle later, there is an additional delay of 2 cycles. Also included is the delay Δ $(= \frac{1}{\lambda_{s,d}})$ between consecutively injected flits for the same source-destination pair, as the ARQ is triggered only with the receipt of the next flit. Thus, the total path latency is obtained as shown in Eq. 5.14 by considering that t consecutive flits are lost before the next flit is received successfully with probability $f^t_{s,d} \cdot f'_{s,d}$. Moreover, the retransmission was received successfully with probability $f'_{d,s} \cdot f'_{s,d}$:

$$\begin{aligned} \ell'_{s,d} &= \ell_{s,d} \cdot f'_{s,d} + \sum_{t=1}^{\infty} (\Delta \cdot t + \ell_{Rs,d} + \ell_{s,d}) \cdot f^t_{s,d} \cdot {f'_{s,d}}^2 \cdot f'_{d,s} \\ &= \ell_{s,d} \cdot f'_{s,d} + [\Delta + f'_{s,d} \cdot (\ell_{Rs,d} + \ell_{s,d})] \cdot f_{s,d} \cdot f'_{d,s}. \end{aligned} \tag{5.14}$$

Using the following series summations:

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}. \tag{5.15}$$

$$\sum_{k=0}^{\infty} k x^k = \frac{x}{(1-x)^2}. \tag{5.16}$$

in Eqn.5.14, we get:

$$\ell'_{s,d} = \ell_{s,d} \cdot f'_{s,d} + [\Delta + f'_{s,d} \cdot (\ell_{Rs,d} + \ell_{s,d})] \cdot f_{s,d} \cdot f'_{d,s}. \tag{5.17}$$

The average path latency is obtained by

$$\overline{\ell} = \frac{1}{M(M-1)} \sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} \ell'_{s,d}. \tag{5.18}$$

Residual Error Probability: The residual error probability is obtained by considering that even with retransmission, the flit fails to reach the destination because either the ARQ or the retransmission was lost. The average residual error probability of the system is given by:

$$\bar{\epsilon} = \frac{1}{M(M-1)} \sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} f_{s,d} \cdot (1 - f'_{d,s} \cdot f'_{s,d}). \tag{5.19}$$

Using the above set of equations, we are able to analytically deduce the NoC performance parameters for the UC case. Next, we describe and deduce the equations for calculating these parameters for the RLNC case.

5.2.2.3 RLNC coded Network (NC)

For the RLNC case, an error occurs if less than G flits of a generation are received. In this case, decoding of this generation is not possible. The error is recognized if at least one flit of the generation was received. The error probability is obtained using the binomial distribution, as in[PF14]:

$$f_{NCs,d} = \sum_{i=1}^{G-1} \binom{C}{i} {f'_{s,d}}^{i} f_{s,d}^{C-i}. \tag{5.20}$$

Network load: The total injection rate per module is again composed of ARQs and retransmissions in addition to the regular injection rate $\lambda_{s,d}$. Since only one ARQ and retransmission is sent per C combined flits of a generation, a factor $\frac{1}{C}$ has to be included in Eq. 5.41:

$$\lambda'_{NCs,d} = \lambda_{s,d} + \frac{\lambda_{d,s}}{C} \cdot f_{NCd,s} + \frac{\lambda_{s,d}}{C} \cdot f_{NCs,d} \cdot f'_{d,s}, \tag{5.21}$$

$$A_{NC} = \frac{1}{M} \sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} \lambda'_{NCs,d}. \tag{5.22}$$

Information Rate: Similar to the uncoded case, the information rate is obtained by

$$I_{NC} = \frac{M(M-1) \cdot R}{\sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} (1 + \frac{f_{NCd,s}}{C} + \frac{f_{NCs,d}}{C} \cdot f'_{d,s})}. \tag{5.23}$$

Latency: In the absence of faulty routers, the sender first collects G flits of a generation, and then after computing C combinations, transmits them one after another. At the receiver, the combinations are collected and after the arrival of G flits, the flits are decoded and forwarded to the receiver module. Including theses delays in addition to buffering delays at the sender and the receiver, the path latency $\ell_{NCs,d}$ is obtained. With ARQ and

retransmission, the total latency now includes the round-trip delay, $\ell_{Rs,d}$. The total and average latencies are given by Eqs. 5.24 and 5.25, respectively:

$$\ell'_{NCs,d} = \ell_{NCs,d} \cdot \sum_{i=G}^{C} \binom{C}{i} {f'_{s,d}}^{i} f_{s,d}^{C-i} + (\ell_{NCs,d} + \ell_{Rs,d}) \cdot \binom{C}{G-1} (f'_{s,d})^{G-1} {f_{s,d}}^{C-(G-1)} \cdot f'_{d,s} \cdot f'_{s,d}, \quad (5.24)$$

$$\overline{\ell}_{NC} = \frac{1}{M(M-1)} \sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} \ell'_{NCs,d}. \quad (5.25)$$

Residual Error Probability: As for the uncoded case, the residual error probability with retransmission increases with loss of the ARQ or the retransmitted flit. If $G-2$ or less flits are received, decoding is not possible even with retransmission and so an error is also generated:

$$\overline{\epsilon}_{NC} = \frac{1}{M(M-1)} \sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} \left[\sum_{i=0}^{G-2} \binom{C}{i} (1 - f_{s,d})^{i} {f_{s,d}}^{C-i} + \binom{C}{G-1} (f'_{s,d})^{G-1} {f_{s,d}}^{C-(G-1)} \cdot (1 - f'_{s,d} \cdot f'_{d,s}) \right]. \quad (5.26)$$

5.2.3 Results and discussion

Evaluation parameters: In the simulations as well as in the model, an 8x8 2D mesh topology is studied. On average, 0.2 flits per router and per cycle are injected to the network (injection rate λ). However, the amount of effectively transmitted information λ_{eff} is influenced by the code rate R, due to the injection of redundant flits:

$$\lambda_{\text{eff}} := \lambda \cdot R. \quad (5.27)$$

The following code rates are simulated: 2/2, 2/3, 2/4, and 3/4. We denote these cases as G2C2, G2C3, G2C4, and G3C4. The uncoded unicast (UC) serves as reference. Eight error-prone routers are randomly selected. For one simulation run, the flit loss probability f of all error-prone routers was set to a constant value. For every identified flit loss, at most one ARQ and one retransmission can be sent. For the NC case, at most one ARQ and retransmission is allowed per generation. Table 5.3 gives an overview of the simulation parameters.

The evaluation was repeated over 1000 iterations for different fault locations both by the analytic model and the simulation and the results averaged to remove the effect of any particular fault location. The averaged results obtained from the analytic model are shown in Fig.5.2. The maximum estimation error of the model relative to the simulation results are given in Tab. 5.4.

Table 5.3: Overview of simulation parameters.

Topology	2D mesh of size 8×8
Arbitration	Round-robin
Input buffers	Size: 4
Injection Rate	$\lambda = 0.2$
Code Rates	$R = G/C$ with [2/2, 2/3, 2/4, 3/4, UC]
Error-Prone Routers	8 (randomly selected)
Flit loss probability	$f = 0.01 \cdot i, i = 0, 1, \ldots, 20$
Max. # ARQ	1
Simulation run time	50k cycles

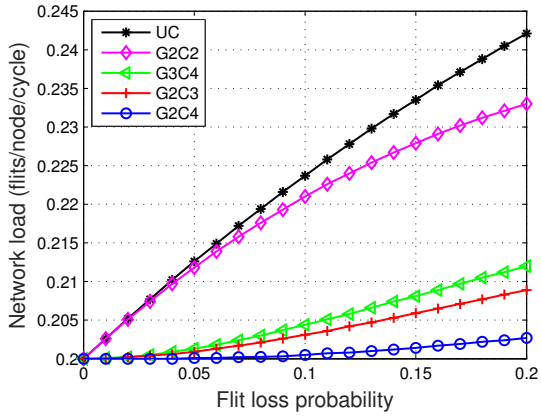
Table 5.4: Maximum Estimation Error of Analytic Model with iterations over 1000 random fault locations

	UC (%)	G2C4(%)	G2C3(%)	G3C4(%)	G2C2(%)
A	1.25	0.34	0.05	0.47	0.49
I	1.22	0.34	0.22	0.75	0.19
$\overline{\ell}$	4.3	0.65	2.93	5.26	6.5
$\overline{\epsilon}$	0.6	1.4	0.6	0.75	0.57

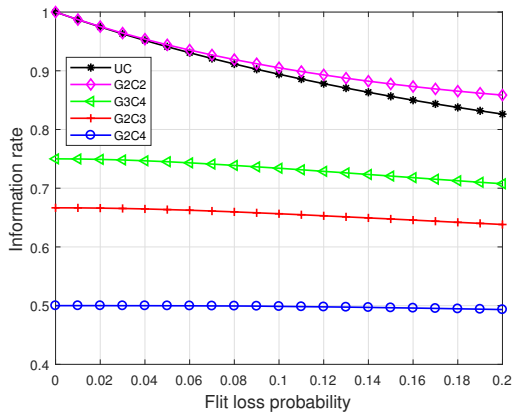
The latency estimation by the model had a maximum error of 6.5% for the G2C2 case. For the other cases, the estimation error was equal to or less than 5%. All results have shown the same trend compared to the simulation results, thus validating the accuracy of the model. The prominent advantage is the fraction of time taken for the estimation by the analytic model: the model is approximately 450 times faster than the cycle-accurate simulations, allowing flexible design space exploration of different NoC sizes and/or topologies.

Latency: As it can be seen in Fig. 5.2(c), the main advantage of RLNC comes from its latency reduction in error-prone NoC. In the considered simulation scenario with eight error-prone routers, RLNC becomes already worthwhile for flit loss probabilities of $> 3\%$. It could be shown that a latency reduction of up to 59% is possible (for 20% flit loss, comparing the UC and G2C3/G2C4 curves).

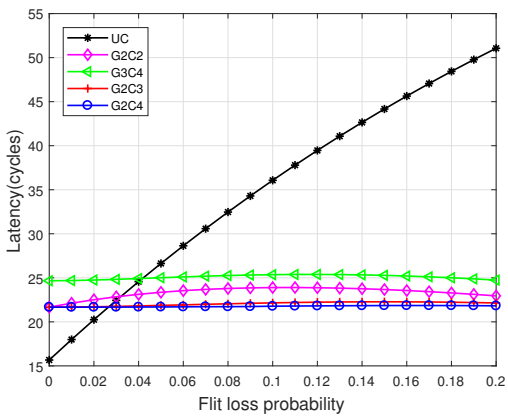
Network load: As well as for the latency, RLNC also shows clear benefits concerning the added load on the network (Fig. 5.2(a)). The lower the code rate, the more advantageous RLNC becomes compared to the UC transmissions. This is due to the included redundant



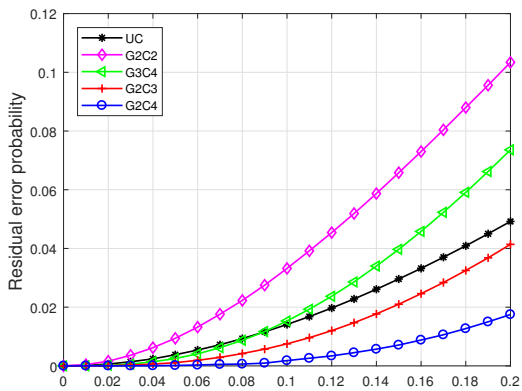
(a) Gross traffic or Network load showing the total traffic including ARQs and retransmissions.



(b) Information rate showing the proportion of actual information in the transmitted flits.



(c) Average path latency



(d) Residual error probability

Figure 5.2: Analytic model results for 8x8 2D mesh with randomly located eight error-prone routers over 1000 iterations of different locations. The plots depict the effect of the flit loss in the routers resulting in ARQs and retransmissions on the average network load, residual error rate, latency and the ratio useful information transmitted.

flits so that in the case of flit loss during transmission, fewer ARQs and retransmissions are needed than the UC case. With respect to the network load, we could also observe a moderate reduction of the network load of up to 15% compared to the UC traffic case (for 20% flit loss probability).

Information rate: The information overhead of network coding i.e. the information redundancy is observed in the plots of the information rate in Fig. 5.2(b). Due to the additional redundant flits (i.e. the generated random linear combinations) the information part is reduced significantly. However, analyzing the curves more closely, it can be seen that the margin between UC traffic and RLNC traffic becomes smaller for higher error probability. Under the zero flit loss condition, a factor of two is observed comparing the

UC and G2C4 curves. The gap is reduced to a factor of 1.7 at 20% flit loss.

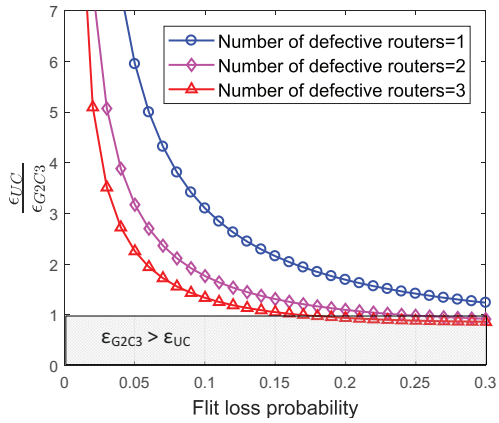
Residual error probability: From the results of the average residual error probability, $\bar{\epsilon}$ in Fig.5.2(d), it is observed that RLNC shows clear advantages over UC. The average residual error was reduced by up to 65% at a 20% flit loss probability. However, $\bar{\epsilon}$ increases with fewer redundant combinations. Comparing the UC and NC cases, G2C3 and G2C4 provides lower error rates in the range of the flit loss probability observed, while G3C4 fares worse than UC as the flit loss probability increases. G2C2, which has no redundancy always has a higher error rate than UC since both of the 2 combinations sent by the sender must be received at the receiver in order to decode the generation. If only one is received and the ARQ or retransmission also fails, then the generation cannot be decoded. This shows us the drawback of using network coding without redundancy.

Let us take a closer look to the comparison of the residual error probability of G2C3 and UC. We assume that on average the number of defective routers ($N_{s,d}$) in the path between s and d are equal i.e. $f_{s,d} = f_{d,s} = f_1$, so that the ratio of the residual error rates of the NC and UC schemes is given by:

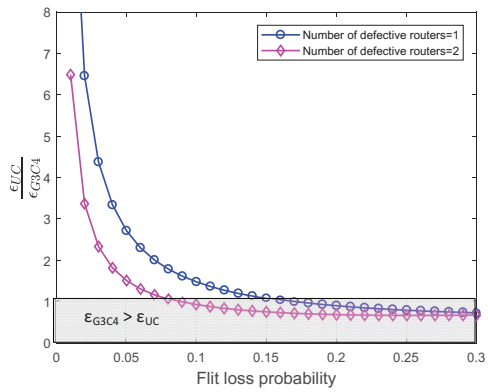
$$\begin{aligned}\frac{\epsilon_{UC_{s,d}}}{\epsilon_{G2C3_{s,d}}} &= \frac{f_{s,d}(f_{s,d} + f_{d,s} - f_{s,d} \cdot f_{d,s})}{f_{s,d}^2 + 3{f_{s,d}}^2(1 - f_{s,d})(f_{s,d} + f_{d,s} - f_{s,d} \cdot f_{d,s})} \\ &= \frac{f_1(2f_1 - {f_1}^2)}{{f_1}^2 + 3f_1(1 - f_1)(2f_1 - {f_1}^2)} \\ &= \frac{2 - f_1}{f_1 + 3(1 - f_1)(2f_1 - {f_1}^2)}\end{aligned} \tag{5.28}$$

This ratio is plotted in Fig.5.3(a), for f increasing from 0 to 0.3, with different numbers of faulty routers ($N_{s,d}$=$N_{d,s}$=1,2,3). The plot shows us that with increasing number of faults, $\epsilon_{G2C3_{s,d}}$ can exceed $\epsilon_{UC_{s,d}}$ e.g. for $N_{s,d} = 3$ at $f = 0.16$ (at which point the ratio becomes less than 1). This happens because at higher error rates, sufficient combinations are not received for decoding the generation so that the whole generation is actually considered lost, even if some combinations are received.

For our evaluations in Gig. 5.2, the 8×8 mesh NoC has an average path length of 6.33 hops so that the average number of faulty routers along this path is less than 1. Thus in the flit loss probability range of $f = 0 : 0.2$, we do not see G2C3 having worse error rates than the UC case. If we were to plot the same comparison of residual error probability of UC to G3C4 (given in Fig.5.3(b)), we can see that G3C4 already has a higher residual error rate for $N_{s,d} = 2$ at $f = 0.09$. We can conclude that if the ratio of defective routers is increased or the average path length is increased (e.g. for a larger NoC investigated in Sec. 5.2.4.2), we will see that even G2C3 will not have sufficient redundancy to outperform UC scheme.



(a) Comparison of residual error rate of G2C3 and UC



(b) Comparison of residual error rate of G3C4 and UC

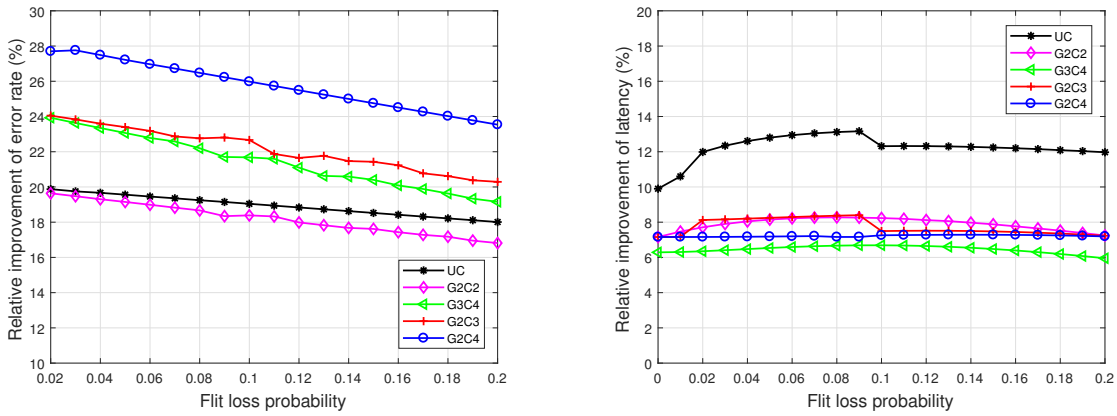
Figure 5.3: Variation of ratio of residual error probability of UC and G2C3 and of UC and G3C4 with increasing flit loss probability.

5.2.4 Applications of the analytic model

The benefit of using an analytic model is seen when evaluating large networks or even different topologies requiring with very small effort.

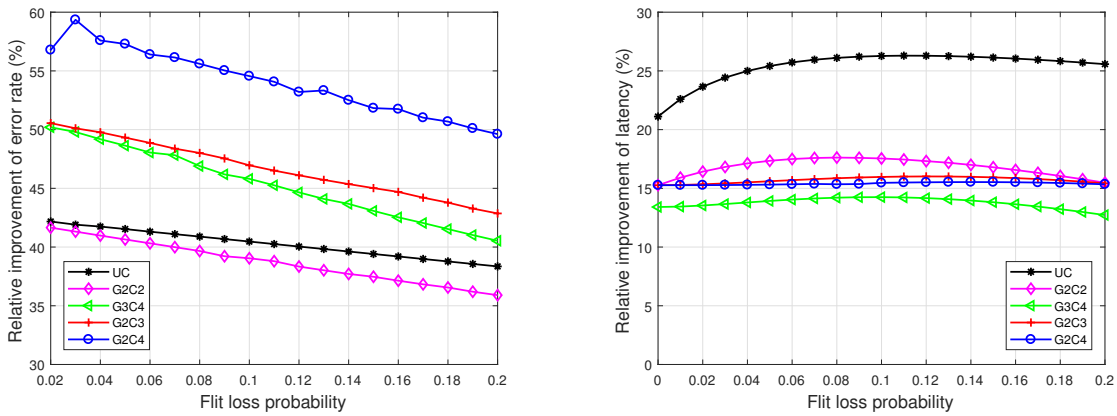
5.2.4.1 Evaluation of 2D hexagonal and octagonal NoC

Firstly, using the analytic model a 8x8 hex and oct NoC was evaluated for 8 defect errors, located at different random locations over 1000 iterations. The relative improvements in the network performance in the residual error probability and the latency of the hex and oct over the mesh are shown in are presented in Fig.5.4. We do not show the relative improvements for the network load or the information rate, as these improvements are relatively small.



(a) Improvement of residual error rate of 2D hex over 2D mesh.

(b) Improvement of latency of 2D hex over 2D mesh.



(c) Improvement of residual error rate of 2D oct over 2D mesh.

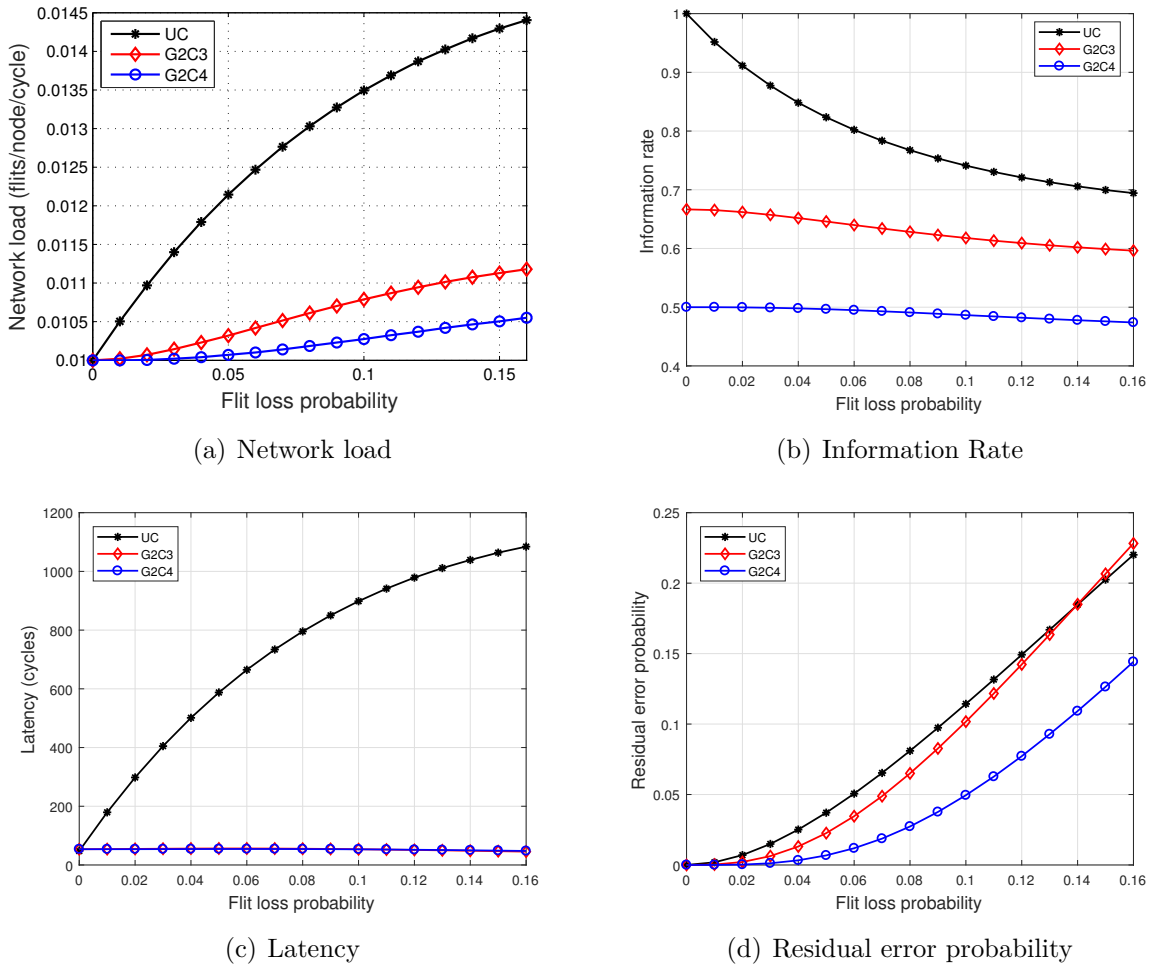
(d) Improvement of latency of 2D oct over 2D mesh.

Figure 5.4: Relative improvements of performance for 8×8 2D hex and 2D oct over the 2D mesh, obtained from analytic model results for the hex and oct NoCs.

Deterministic dimension-ordered routing is assumed in the hexagonal NoC (diagonal-X-Y) and octagonal NoC (diagonal1-diagonal2-X-Y) similar to that in mesh. As can be observed, hex and oct NoCs fare better in performance than the 2D mesh because of their average lower path length. For 8×8 network size, hex and oct NoCs have a average path length of 5.54 hops and 4.69 hops in comparison to that of 6.33 hops for the mesh. As a result, the probability of encountering a defect router is lower in the hex than in the mesh (Eqn. 5.6). For residual error probability, the greatest improvement is for G2C4 case in which 2D oct and 2D hex have an average 54% and 25.7% lower error rate compared to that of the 2D mesh. For latency, the greatest improvement is for UC case in which 2D oct and 2D hex have an average 25% and 12% lower latency compared to that of the 2D mesh.

5.2.4.2 Evaluation of large NoC

When larger NoCs are to be investigated, in this case having 1024 cores, the advantage of the analytic model is significant. Such an evaluation using cycle accurate simulator would take days if not weeks. The large network was evaluated for the RLNC and UC case, having 128 error-prone routers, i.e., the same proportion as for the 8x8 NoC over 5000 different random locations. The results are shown in Fig. 5.5. It can be observed that



(a) Network load

(b) Information Rate

(c) Latency

(d) Residual error probability

Figure 5.5: Analytic model results for 32x32 2D mesh with 128 error-prone routers.

with increasing flit loss the high latencies of UC make it impractical for use in NOC. By using RLNC, compared to the UC case the latency can be reduced by 95% (at 20% flit loss, comparing the UC and RLNC curves). It has to be noted that a larger NoC has a higher average path length (approximately 22.33 hops) and as a result the probability of encountering a faulty router is greater. Thus the average error probability is higher for the large network. For the UC case, with retransmission the residual error rate is 10% higher at 3% flit loss (comparing 32x32 to 8x8 NoC). For RLNC with higher code rates (G2C2, G3C4), one retransmission per generation is not sufficient to complete the generation

which makes decoding impossible and results in a higher residual error rate compared to UC. G2C3 does better in terms of residual error probability but only up to 13% flit loss compared to UC. For G2C4, the residual error probability is 25% less compared to UC (at 20% flit loss). When comparing the network load, RLNC performs better than UC, with a 27% lower network load for G2C4 (at 20% flit loss).

As large networks have higher residual error probabilities, RLNC can provide greater resilience but with a higher redundancy such as G2C4. In terms of network load and latency, which are two very important parameters for NoC, RLNC performs better than UC. As noted before in section 5.2.3, the advantage of RLNC comes at the price of reduced information rate. However, the gap in information rate between RLNC and UC becomes smaller at higher error probability. For the 32x32 NoC, when comparing UC and G2C4, a factor of 2 at zero flit loss is reduced to a factor of 1.46 at 20% flit loss. This indicates that due to the great number of ARQs and retransmissions, UC case will have a high amount of traffic composed of flits other than useful information flits.

5.2.5 Summary

From the above analysis, we can conclude that RLNC is a flexible approach for resilient and low-latency transmissions. In summary

- RLNC comprises a linear combination of data flits of a generation at the sender. As the global encoding vector (GEV) is already included in the generated combinations, the receiver can easily decode the received generation provided it receives enough flits of the generation.
- RLNC is advantageous over UC in terms of lower residual error rate and average network latency. Due to its inherent redundancy, RLNC can even reduce network load at high error rates.
- At very high error rates RLNC may have a higher error rate than UC if sufficient redundant combinations are not generated. By varying the code rate, we can adapt the flit redundancy to appropriately suit the error conditions.
- Using the analytic model, we can flexibly explore the design space with regard to different number of faults, different sizes or topology of the NoC, even with different routing schemes.

5.3 NoC Security

A brief introduction to information security is given in Sec. 2.3.2 and the work of this section is presented in [MFW+18]. The growing complexity of MPSoCs causes

the design process to increasingly rely on the integration of components from third parties, potentially also from untrusted sources [SWS+15]. Even tools used for design and development may be compromised, and hence pose a tangible threat to MPSoCs. Such threats can be realized by inserting hardware Trojans (HT), as reported, e.g., in [ED05, ACR14, SWS+15, FY15, BK16]. Designing NoCs, it hence becomes vital to ensure not only reliability to errors, but increasingly also security. In this section we describe the concept of NoC communication threatened by active attacks within the NoC. One major attack vector for MPSoCs is the incorporation of HTs (e.g. [AK08, JKM09]). Considering HTs, an adversarial router may directly drop or alter the forwarded traffic, thus threatening the availability or integrity of the communication. With local or remote access to the device, the adversary may also eavesdrop on the communication. As attacks on the confidentiality additionally require a hidden channel to the attacker, we consider achieving availability and integrity of communication the more pressing goals. Particularly, we focus on active attackers who may modify or drop flits transmitted in the NoC.

5.3.1 Security in MPSoCs

Ancajas et al. reported on the problem of HTs in NoCs and presented the initial foundation of mitigation strategies: scramble lower layer data, certify packets, and implement process migration to obfuscate the destination applications [ACR14]. The suggested methods aim at preventing an attacker to trigger an HT and an eavesdropping NoC component is disabled from identifying logical data streams. Setumadhavan et al. [SWS+15] also aim to prevent the activation of HTs by preventing their trigger events. However, the suggested measures require a significant overhead and, hence, influence the performance of the system.

Another strategy to reduce the potential risks implied by HTs is to detect and eliminate them. One direction is to focus on the design process, e.g., by conducting static analysis for backdoor during the design/build process and applying security orientated development procedures [KLM+04, SWS+15]. Frey and Yu [FY15] focus on detecting HTs at runtime by means of finite state machine controllers that supervises the component's execution path. The drawback of the proposed solution is the implied computation and storage overhead. In [FY16], they subsequently designed a system for detecting HTs which change the flits, especially control information like type and destination. The countermeasures are integrated into the router, implying overhead for each router.

However, HTs are an attack vector that cannot be mitigated completely. Therefore, other strategies aim to diminish the exposed risk by securing the communication during attacks. For example, Boraten and Kodi [BK16] propose the use of algebraic manipulation detection codes for the detection of modifications of flits. The authors claim a minimal performance impact of 1% compared to NoCs. But they provide no security measure. Kapoor et al. proposed to protect the communication by means of authenticated encryp-

tion [KRAT13]. However, the chosen cryptographic primitive, AES-128 in GCM mode, is heavy-weight regarding efficiency and area overhead and is therefore an infeasible solution for NoCs [ED05].

The presented security measures all target the objectives of integrity and confidentiality, but neglect availability. In the seldom works considering all three objectives, then in turn efficiency and area considerations are neglected. Hence, our proposed solution aims at increasing integrity and availability under the constraints of restricted size and tight latency requirements [MFW+18].

5.3.2 System and Attacker Model

Within this work, we use a spatial uniform traffic distribution with a constant injection rate per module. Further, we assume retransmission buffers in the NoCIF (to store transmitted flits) are of sufficient size to prevent flit loss (Sect. 5.3.5).

The flit structure assumed in this work is shown in Fig. 5.6, which is very similar to the flit structure of chip Tomahawk 4 (T4) [H+17], die photo of which was given in Fig.3.7(b). The data width of the payload is 64 bits, according to the memory interface size inside the PE. In addition to the T4 flit, to be able to refer to a flit in case of a retransmission, a 24-bit flit identifier for the uncoded case (corresponding to generation identifier of network coded transmission, Sec. 5.3.3) is included. The 64-bit data field, the flit identifier, the mode (4 bit) that specifies the flit type, and an address (32 bit) for memory accesses constitute the payload of a flit. The header of a flit contains x and y coordinates of source and target module as well as a bit for indicating burst mode (reserved for future work). To support a 2D mesh of up to 16×16 modules, we assume 4 bits for each coordinate. Thus, the length of a flit is 141 bits. For the RLNC case according to [CWJ03], the unit vector must be prepended to the data part in order to have the GEV included in the combination produced after linear combination of the flits of the generations, as described in Sec. 5.2.2.3. Thus the GEV further increases the size of the flit, described further in Sec. 5.3.3.

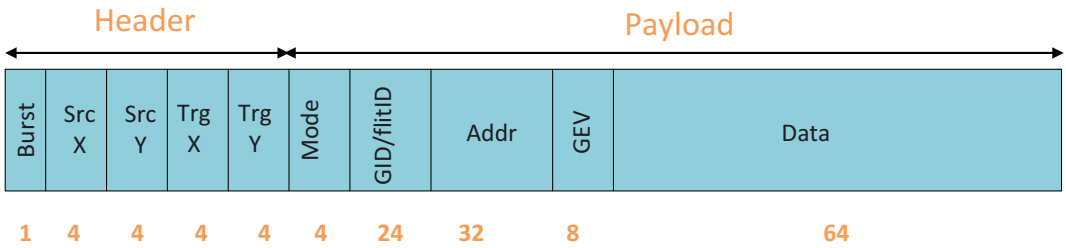


Figure 5.6: Flit structure for coded and uncoded case with different fields in the header and payload (GEV field is absent in UC case).

We assume that routers may be corrupted while PEs and NoCIFs are assumed to be trustworthy. A corrupted router may modify a flit with a certain modification probability p_m or drop it with a drop probability p_d. Security measures are placed in the NoCIF that is assumed to be trustworthy.

5.3.3 Communication Model

We consider two approaches: uncoded (UC) and network coded (NC) transmission according to [CWJ03]. In our case, network coding is applied to flits. We use a generation size of $G = 2$ in order to achieve average latencies comparable to an uncoded transmission [MYF+15]. For each generation, the sender computes C linear combinations(Sec. 5.2.2.3).

Data to be sent is split into packets $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}) \in \mathbb{F}_q^n$ with $q = 2^m$. Each packet is prepended with a unit vector $(\beta_{i,1}, \beta_{i,2}, ..., \beta_{i,G}) \in \mathbb{F}_q^G$ with $\beta_{i,j=i} = 1$ and $\beta_{i,j\neq i} = 0$.

Altogether, we considered three communication scenarios:

UC: uncoded transmission,

G2C3: network coded transmission with $G = 2, C = 3$,

G2C4: network coded transmission with $G = 2, C = 4$.

Hence, a generation always consists of two flits $\mathbf{f}_1, \mathbf{f}_2$ composed of the information bits x_i as well as the prepended unit vector:

$$\begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix} = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ \beta_{2,1} & \beta_{2,2} & x_{2,1} & x_{2,2} & \cdots & x_{2,n} \end{pmatrix} \tag{5.29}$$

The sender computes linear combinations by selecting at random encoding coefficients $\alpha_{i,j} \in \mathbb{F}_q, i = 1, 2, \ldots, C; j = 1, 2$ for the computation of linear combinations $\mathbf{c}_k, k = 1, 2, \ldots, C$. For example, the computation of the combinations for G2C3 is given by:

$$\begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} \\ \alpha_{2,1} & \alpha_{2,2} \\ \alpha_{3,1} & \alpha_{3,2} \end{pmatrix} * \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix} \tag{5.30}$$

The receiver needs $G = 2$ linear independent combinations $\mathbf{c}_i, \mathbf{c}_j$ to be able to decode by multiplying the matrix of these two combined flits with the inverse of the 2×2 matrix A of their corresponding encoding coefficients:

$$\begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix} = \begin{pmatrix} \alpha_{i,1} & \alpha_{i,2} \\ \alpha_{j,1} & \alpha_{j,2} \end{pmatrix}^{-1} * \begin{pmatrix} \mathbf{c}_i \\ \mathbf{c}_j \end{pmatrix} \tag{5.31}$$

Invertibility of matrix of encoding coefficients, A ensures invertibility of the matrix of combined flits and depends both on the size of the A (given by G) and on the size of the finite field q [PF17]. For a 2×2 matrix A, the field GF(2^4) already provides an inverting probability of 0.93384 which is sufficient, especially under the consideration that only the sender randomly selects encoding coefficients. Hence, we assume a symbol size of 4 bits for the randomly selected encoding coefficients and the data symbols so that the total size of the GEV included in the flit payload is 8 bits, resulting in a flit size of 149 bits for the RLNC (in comparison to 141 bits for the UC case).

The *gid* should be unique "within the system" to prevent an injection of a formerly intercepted flit into a transmission or a replay attack. In such a case, the receiver would not be able to decode the current generation since the injected flit does not belong to this generation. With an assumed *gid* size of 24 bits, a total of 2^{24} different combinations for the same meta data field (all fields of the flit apart from the data), which we consider sufficient for our investigations.

5.3.4 Concept for Authentication

As explained in Sec. 2.3.2, Message authentication codes or MACs are suitable when we are considering only authentication of flits. Authentication aims at preventing undetected modifications of transmitted information. Since we wish to protect the entire flit and not only the data, the MAC is generated for the entire flit. AES is a popular algorithm for the generation of MACs, however, it is not suited to NoC because it requires 1,032 cycles per block encryption [E$^+$07]. As a promising algorithm for authentication in NoCs, we selected mCrypton [LK06] that needs only 13 cycles per block encryption and has an area of 2,681 gate equivalents (GEs) only [E$^+$07].

The MAC is generated by the sender and inserted into the flit to be transmitted. Upon receiving the flit, the receiver validates the authenticity of the flit by again generating the MAC and comparing it to the received MAC. If a modification is detected (and there are not sufficient verified flits of the generation in NC case), an ARQ is sent requesting a retransmission. In case of NC, we can have 2 possibilities for authentication: (1) authentication of original flits, or (2) authentication of linear combinations. The first approach allows to compute only a single MAC for a whole generation. There is a low communication overhead if there are no modifications. However, the receiver cannot check the validity of flits before decoding and in case of a modification, retransmission of the whole generation is necessary since the receiver cannot find out which of the flits was modified. Hence, we focus on the second approach where the receiver does not start decoding before a successful verification of the received combinations.The size of the MAC depends on the block size of the underlying cipher; mCrypton has a block size of 64 bits and thus the resulting MAC is also 64 bits, which is exactly the size of the data field in our flit 5.6. The crypto modules are stored in the NoCIF and the sender and receiver side of each NoCIF

share the crypto modules.

We can consider 2 ways in which the MAC should be transmitted with the flit:

Since

- Solution 1 (S1): send the MAC in a separate flit, or
- Solution 2 (S2): include both data and MAC in a flit.

5.3.4.1 S1: Send the MAC in a Separate Flit

In this scenario, for each transmitted data of 64 bits there will be 2 flits which will be transmitted: one containing the original data and the other containing the MAC in the data field. These are called the data and MAC flits respectively. Thus, for the UC case there are always 2 flits (Data and MAC) belonging together and transmitted consecutively. For NC, the sender first computes C linear combinations from G flits. Afterward, he computes a MAC for each combination so that there are $2 \cdot C$ flits which are sent consecutively. If the receiver gets a data flit, he can immediately start with the computation of the MAC. After finishing, he can compare the generated MAC to the MAC inside the corresponding MAC flit. After a successful verification of at least G combined flits, decoding can start.

If the receiver recognizes a loss, a ARQ is issued. In case of UC, the receiver starts a timer when he receives a data flit. If there is a time-out, he issues an ARQ. If the receiver gets first a MAC flit, he can directly issue an ARQ since the order of flits is not changed during transmission. In case of NC, the receiver starts a timer when a flit of a generation arrives. If verification fails, the receiver issues an ARQ for both data and MAC flit since he cannot decide which of these two flits was modified. The sender buffers data and MAC flits so that it is not necessary to compute the MAC again. For NC, it is not necessary to issue an ARQ immediately. For example, if 2 of 4 received combinations failed verification, decoding is possible if the remaining 2 combinations are unmodified. After successful verification and, in case of NC, decoding, the flits are delivered to the PE.

Input for the computation of the MAC is the whole data flit of 141 bits or 149 bits. Given the block length of 64 bits, there are three input blocks for mCrypton. Hence, the computation of the MAC implies a delay of $3 \cdot 13 = 39$ cycles for both sender and receiver. Thus there must be sufficient number of crypto modules in each NoCIF in the NoC so that with the assumed flit injection rate, there is no loss due to queue overflow of flits awaiting MAC generation (Sect. 5.3.6.2).

5.3.4.2 S2: Include Data and MAC in One Flit

In S2, the MAC is included within the data field of the payload. However, we do not want to increase the size of the flit and hence, we consider as an alternative the use of a simple

Authentication Code [Sim91] that requires two key bits for the authentication of a single message bit (Tab. 5.5). If an attacker observes a flit with message bits and MAC bits, two keys remain possible for each message bit so that he can guess only with probability 0.5 the correct key bits and undetectably modify the data bit. For authentication of x message bits, we can simply use a stream of $2x$ key bits. Hence, if we use 32 bits of the data field for the actual data and 32 bits for the MAC bits, we need 64 key bits.

Table 5.5: Example for the authentication of one bit.

key bits		00	01	10	11
message bit	0	0	0	1	1
	1	0	1	0	1

The receiver must be able to generate the same key bits for each flit as the sender, even if previous flits are lost. Hence, the key bits are generated pseudo randomly by encrypting the meta data of a flit. Firstly, the data received in each NoCIF is split into 2 parts of 32 bits. In case of UC, the size of the input for the block cipher is 77 bits (header+mode+flitID+address+data), i.e., there are two input blocks for the mCrypton. For NC, there is an input of 85 bits including the GEV, i.e., there are also two input blocks. Encryption of two blocks with mCrypton requires $2 \cdot 13 = 26$ cycles. The resulting block of 64 bits provide the required pseudo random key bits. Using these key bits and the original data field, the MAC is computed by a simple look up in the small table with negligible effort.

In case of NC, the sender must first compute C linear combinations from the 2 split data parts. After that, he computes the pseudo random key bits for each combination and then the MACC, and, finally, sends the resulting flits to the receiver. The receiver can immediately start with the verification when a flit arrives. Since the sender splits a data block into two flits, the receiver needs both flits in case of UC to successfully deliver the 64-bit data block to the module. In case of NC, the sender needs G correct flits to be able to decode them to 2 flits that deliver the 64-bit data block.

For the recognition of loss, timers are used as well. If the receiver recognizes a modification, he also issues an ARQ but in contrast to S1, only a retransmission of one flit is necessary.

5.3.5 Evaluation

5.3.5.1 Parameters and Performance Metrics

Parameters and their respective value ranges are summarized in Table 5.6. The flit generation rate has to be adapted wrt. the used communication scheme to achieve the mentioned injection rate in order to be able to compare the different schemes. With our assumed effective flit injection rate of 0.2 flits/node/cycle, for UC, the desired network injection rate

needs to be halved i.e it is 0.2 flits/node/cycle(since in both S1 and S2 cases one original data flit results in two transmitted flits by data splitting or due to data+MAC). For NC, this rate needs to be further divided by the number of combinations. The resulting flit creation rates are 0.067 for G2C3 and 0.05 for G2C4.

Table 5.6: Overview of simulation parameters.

Topology	2D mesh of size 8×8
Routing	Deterministic, dimension-ordered XY
Arbitration	Round-robin
Injection rate	$\lambda = 0.2$
Communication model	S1/{UC,G2C3,G2C4}, S2/{UC,G2C3,G2C4}
Rogue routers	8 (randomly selected)
Modification probability	$p_m = 0.005 \cdot i, i = 0, 1, \ldots, 20$
Drop probability	$p_d = 0.005 \cdot i, i = 0, 1, \ldots, 20$
Simulation run time	50000 cycles

The maximum number of retransmissions (and therefore ARQs) is limited to 1 with respect to the logical transmission unit. For UC, one unit is a pair of flits (either a pair of data+MAC flit or a pair of combined data/MAC flits); for NC, one unit is a single generation. As in Sec.5.2, we assumed a single retransmission to limit the network load and to simplify the error control mechanism.We assume correct transmission between routers. Finally, it was assumed that ARQs are prone to dropping attacks, but not to modification. At present, ARQs are not authenticated. This originates from the fact, that the current authentication scheme only targets the transmission of data flit.

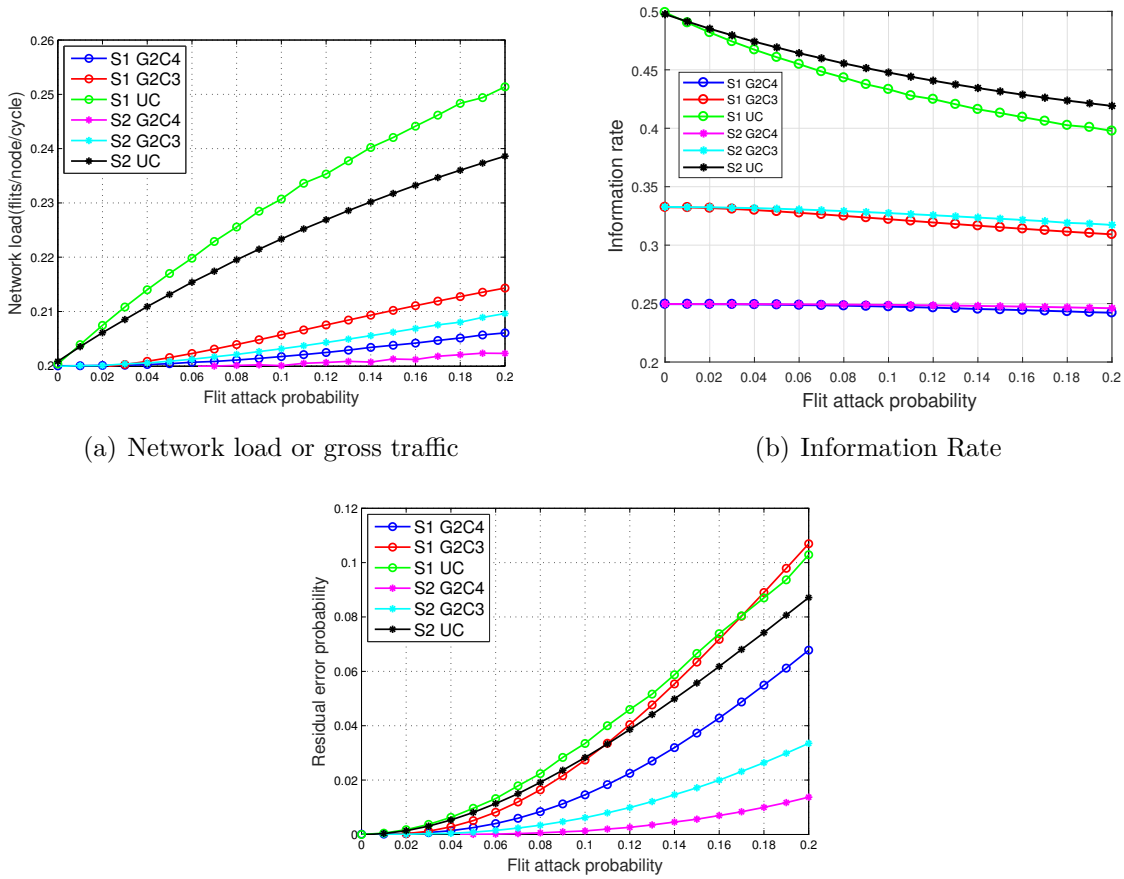
The performance metrics, Network load or gross traffic (A), Information rate (I) and Residual error probability (ϵ), as explained in Sec.5.2.2 were considered in the evaluations.

5.3.6 Simulation

To evaluate the performance of the proposed authentication schemes, the solutions were simulated by a cycle-accurate C++ framework for NoCs [WF08]. The existing framework was extended with the functionality of network coding and cryptographic primitives.

The values of p_m and p_d were kept equal for single runs. To simplify matters, we will refer to their sum as attack probability. To eliminate the influence of attacker positions,

results for each point were calculated by averaging outcomes of 1000 simulation runs with randomly selected positions of the 8 rogue routers.



(a) Network load or gross traffic

(b) Information Rate

(c) Residual error probability

Figure 5.7: Simulation results for 8x8 2D mesh with 8 attacking routers. The plots depict the effect of the flit drop or modification in the attacking routers (resulting in ARQs and retransmissions) on the average network load, residual error rate and the ratio of useful information transmitted.

Fig. 5.7 presents results of the simulation runs. For an attack rate of 0.0, the **network load** equals the network injection rate of 0.2 (Fig. 5.7(a)). With increasing attack probability, losses and modifications occur. The resulting ARQs and retransmissions imply an increase of the network load. UC schemes induce significantly more transmissions. Since there is no redundancy, each modification or loss of a flit implies an ARQ and a retransmission. The robustness of the NC schemes is clearly visible – the higher the redundancy, the lower the network load. Fig. 5.7(a) also shows that S1 performs worse than S2: S1 requires a retransmission of both flits, since the receiver cannot decide whether data or MAC flit was corrupted.

Fig. 5.7(b) depicts the information rate with respect to the attack probability. With an attack probability of 0.0, the **information rate** is determined only by the given communi-

cation schemes. With increasing attack rates and thus greater ARQs and retransmission, I decreases. The relative differences between S1 and S2 once more depict the different robustness of the schemes.

Fig. 5.7(c) presents the residual error probability. Here we again see NC schemes benefit from the redundancy. S2 achieves best results. For the highest attack rate, S2/G2C4 provides the lowest error probability of $\approx 1.36\%$. S1 requires two successful retransmission in case of an ARQ. Although S1/G2C3 starts with a lower residual error than the UC schemes, its error probability increases faster and becomes greater than the UC schemes at 11% and 17% attack probability. The results for S1/G2C4 show that an increased redundancy can effectively counteract this increased error growth. Nevertheless,

Table 5.7: Analytic model parameter symbols

M	Total number of modules or nodes
$\lambda_{x,y}$	Flit injection rate from x to y
$\lambda'_{x,y}$	Total flit injection rate from x to y including ARQs and retransmission
$N_{x,y}$	Total number of rogue routers in the XY route from x to y
$d_{x,y}$	Total flit drop probability from x to y
$d'_{x,y}$	Probability of no drop from x to y
$m_{x,y}$	Total flit modification probability from x to y
$m'_{x,y}$	Probability of no modification from x to y

5.3.6.1 Analytical Model

In this section we describe an analytic model for calculating the performance parameters without resorting to lengthy simulations. A short overview of further parameters used in the analytic model is given in Tab. 5.7. In our assumed schemes, the total flit loss or modification probability between a source-destination pair is determined by the number of rogue routers encountered in the XY route between them, as given in the following.

$$d_{x,y} = 1 - (1 - p_d)^{N_{x,y}}. \tag{5.32}$$

$$m_{x,y} = 1 - (1 - p_m)^{N_{x,y}}. \tag{5.33}$$

The probability of no drop $(1 - d_{x,y})$ or no modification $(1 - m_{x,y})$ are denoted by $d'_{x,y}$ and $m'_{x,y}$, respectively. When an ARQ is sent for a missing flit or modified flit (in S2 scheme), for the retransmitted to reach the receiver successfully, the ARQ should not be

lost or the retransmission should not be lost or modified. The expression below denotes the probability of such a situation:

$$R_{x,y} = 1 - d'_{y,x} d'_{x,y} m'_{x,y}. \tag{5.34}$$

For S1 scheme (MAC and data sent in separate flits), when the MAC verification fails, it is not possible to know whether the data or MAC flit was modified. Therefore, the ARQ sent asks for the retransmission of both the data and the MAC flit. In such a situation, for the retransmission to successfully reach the receiver, the ARQ must be successful and both the retransmitted data and MAC flits must reach the receiver without modification, as denoted in the following expression:

$$Q_{x,y} = 1 - d'_{y,x} {d'_{x,y}}^{2} {m'_{x,y}}^{2}. \tag{5.35}$$

We first start with S1 scheme, i.e. transmission of MAC in a separate flit than the data flit.

S1: Network Coded Transmission

In NC scheme, 'G' pairs of data and corresponding MAC flits need to be received correctly in order to successfully decode the generation. Otherwise, there is an error which would require a retransmission. Based on this assumption, the performance parameters are calculated in the following.

Residual Error Probability: When evaluating the residual error rate, we have to consider different cases of flit losses or modifications. Let us go through this step by step. If we consider the case of G2C3, 3 pairs of flits will be transmitted, and at least 2 verified pairs or $2G$ flits must be successfully received at the receiver. Here, a residual error occurs in the following cases:

- in total less than 2G-1 flits received and the rest are lost. Here, even a retransmission will not help since only 1 retransmission is allowed and this is not sufficient for decoding (even if the retransmission were successful and the flits were not modified): $\epsilon_{0x,y} = \sum_{i=0}^{2G-2} \binom{2C}{i} {d'_{x,y}}^{i} {d_{x,y}}^{2C-i}$
- 2G-1 flits received (${d'_{x,y}}^{2g-1} {d_{x,y}}^{2c-2g+1}$), including one pair ($\binom{C}{1}\binom{2C-2}{1}$) but the ARQ/retransmission fails (with probability $R_{x,y}$) or one or more of the received flits were modified. If the received flits are of different pairs ($\binom{2C}{2G-1} - \binom{C}{1}\binom{2C-2}{1}$), then the retransmission does not help:

$$\epsilon_{1x,y} = {d'_{x,y}}^{2G-1} {d_{x,y}}^{2C-2G+1} \cdot \left[\binom{C}{1}\binom{2C-2}{1} \cdot \left({m'_{x,y}}^{2G-1} \cdot R_{x,y} + 1 - {m'_{x,y}}^{2G-1} \right) + \binom{2C}{2G-1} - \binom{C}{1}\binom{2C-2}{1} \right] \tag{5.36}$$

- 2G flits were received but not G pairs ($\binom{2C}{2G} - \binom{C}{G}$) and the ARQ /retransmission fails. When 'G' ($\binom{C}{G}$) pairs are received, if the modified flits are restricted to one pair ($\binom{G}{1}$), an ARQ will be sent for the pair of data and MAC flit. A residual error happens due to failure of ARQ/retransmissions. However, if flits of different pairs are modified, then ARQ does not help, since only one ARQ is allowed.

$$\epsilon_{2x,y} = \underbrace{{d'_{x,y}}^{2G} {d_{x,y}}^{2C-2G}}_{e_{2xy}} \cdot \Bigg[\binom{C}{G} \Bigg\{ \binom{2G}{1} m'_{x,y} {m'_{x,y}}^{2G-1} \cdot Q_{x,y} + {m'_{x,y}}^{2} {m_{x,y}}^{2G-2} \cdot \Bigg(\binom{G}{1} \cdot Q_{x,y} + \binom{2G}{2} - \binom{G}{1} \Bigg) + \sum_{k=3}^{2G} {m'_{x,y}}^{k} {m_{x,y}}^{2G-k} \Bigg\} + \binom{2C}{2G} - \binom{C}{G} \Bigg] \quad (5.37)$$

- more than 2G flits received i.e 2 or 3 pairs of data/MAC flits, but due to modification and failure of ARQ/retransmission, the generation cannot be decoded.

$$\epsilon_{3x,y} = \underbrace{{d'_{x,y}}^{2G+1} {d_{x,y}}^{2C-2G-1}}_{e_{3xy}} \cdot \Bigg[\binom{2G}{1} m'_{x,y} {m'_{x,y}}^{2G-1} \cdot Q_{x,y} + {m'_{x,y}}^{2} {m_{x,y}}^{2G-2} \cdot \Bigg(\binom{G}{1} \cdot Q_{x,y} + \binom{2G}{2} - \binom{G}{1} \Bigg) + \sum_{k=3}^{2G} {m_{x,y}}^{k} {m'_{x,y}}^{2G-k} \Bigg] \quad (5.38)$$

$$\epsilon_{4x,y} = \overbrace{{d'_{x,y}}^{c}}^{e_{4xy}} \cdot \Bigg[{m_{x,y}}^{2} {m'_{x,y}}^{2C-2} \cdot \binom{C}{1}\binom{2}{1}\binom{2}{1} \cdot Q_{x,y} + {m_{x,y}}^{3} {m'_{x,y}}^{2C-3} \cdot \Bigg\{ \binom{C}{1}\binom{2C-2}{1} Q_{x,y} + \binom{2C}{3} - \binom{C}{1}\binom{2C-2}{1} \Bigg\} + {m_{x,y}}^{4} {m'_{x,y}}^{2C-4} \cdot \Bigg(\binom{C}{1} Q_{x,y} + \binom{2C}{2} - \binom{C}{1} \Bigg) + \sum_{k=5}^{2C} {m_{x,y}}^{k} {m'_{x,y}}^{2C-k} \Bigg] \quad (5.39)$$

The average residual error probability is given by :

$$\bar{\epsilon} = \frac{1}{M(M-1)} \sum_{s=1}^{M} \sum_{\substack{d=1 \\ d \neq s}}^{M} \cdot (\epsilon_{0x,y} + \epsilon_{1x,y} + \epsilon_{2x,y} + \epsilon_{3x,y} + \epsilon_{4x,y}). \quad (5.40)$$

Network load: The network load, $\lambda'_{x,y}$ is a sum of the normally injected flits ($\lambda_{x,y}$) and the ARQs ($\lambda_{arq\_x,y}$) and retransmissions ($\lambda_{retr\_x,y}$), which are directly related to the error cases from above. A factor ($\frac{1}{2C}$) is included in the injection rates of the ARQs and retransmissions, since only 1 ARQ or retransmission is allowed per C pairs of combined flits. Similarly, the retransmission rate for the modified flit includes a factor of 2× since the pair must be retransmitted for the modified flit.

$$\lambda'_{x,y} = \lambda_{x,y} + \lambda_{arq\_x,y} + \lambda_{retr\_x,y}. \quad (5.41)$$

$$\lambda_{arq\_x,y} = \frac{\lambda_{y,x}}{2C} \cdot \left[\sum_{k=1}^{2G-1} {d'_{y,x}}^{k} {d_{y,x}}^{2C-k} + e_{2yx} \cdot \left\{ \binom{C}{G}(1 - {m'_{y,x}}^{2G}) + \binom{2C}{2G} - \binom{C}{G} \right\} + \right.$$
$$e_{3yx} \cdot (1 - {m'_{y,x}}^{2G}) + e_{4yx} \cdot \left\{ \sum_{k=2G+1}^{2C} {m_{y,x}}^{k} {m'_{y,x}}^{2C-k} + \binom{2C}{2G-1} {m'_{y,x}}^{2G-1} {m'_{y,x}}^{2C-2G+1} + \right.$$
$$\left. \left. \binom{C}{1}\binom{2}{1}\binom{2}{1} {m'_{y,x}}^{2} {m'_{y,x}}^{2C-2} \right\} \right] \quad (5.42)$$

Provided that the ARQ was not lost ($d'_{y,x}$), retransmission of the required flit(s) will take place.

$$\lambda_{retr\_x,y} = d'_{y,x} \cdot \frac{\lambda_{x,y}}{2C} \cdot \left[\sum_{k=1}^{2G-1} {d'_{x,y}}^{k} {d_{x,y}}^{2C-2G+1} + e_{2xy} \cdot 2 \cdot \left\{ \binom{C}{G}(1 - {m'_{x,y}}^{2G}) + \binom{2C}{2G} - \binom{C}{G} \right\} + \right.$$
$$e_{3xy} \cdot 2 \cdot (1 - {m'_{x,y}}^{2G}) + e_{4xy} \cdot 2 \cdot \left\{ \sum_{k=2G+1}^{2C} {m_{x,y}}^{k} {m'_{y,x}}^{2C-k} + \binom{2C}{2G-1} {m'_{x,y}}^{2G-1} {m'_{x,y}}^{2C-2G+1} + \right.$$
$$\left. \left. \binom{c}{1}\binom{2}{1}\binom{2}{1} {m'_{x,y}}^{2} {m'_{x,y}}^{2C-2} \right\} \right] \quad (5.43)$$

Taking the average over all modules, the network load is computed as follows:

$$A = \frac{1}{M} \cdot \sum_{x=1}^{M} \sum_{\substack{y=1 \\ y \neq x}}^{M} \lambda'_{x,y}. \quad (5.44)$$

Information Rate: The information rate is determined by the ratio of the actual data flits transmitted (without redundancy) to the total number of flits transmitted. Since for each data flit, a flit containing the MAC must also be transmitted, a factor of $\frac{1}{2}$ is included in the calculation of the information rate. The same equation also applies to the uncoded case.

$$I_{NC} = \frac{R}{2} \cdot \frac{\sum_{x=1}^{M} \sum_{\substack{y=1 \\ y \neq x}}^{M} \lambda_{x,y}}{\sum_{x=1}^{M} \sum_{\substack{y=1 \\ y \neq x}}^{M} \lambda'_{x,y}}. \quad (5.45)$$

S1: Uncoded Transmission Residual Error Probability: Here, for each data-MAC pair, error results when both flits are lost or when one unmodified flit is received and the ARQ/retransmission for the lost flit fails. If the single received flit is modified, then error will result anyway since no further ARQs are allowed. If both flits are received and MAC verification fails, then error happens when the ARQ/retransmission fails:

$$\bar{\epsilon} = \frac{1}{M(M-1)} \sum_{x=1}^{M} \sum_{\substack{y=1 \\ y \neq x}}^{M} \left[{d_{x,y}}^{2} + \binom{2}{1} d_{x,y} d'_{x,y} \cdot (m'_{x,y} R_{x,y} + d_{x,y}) + {d'_{x,y}}^{2} (1 - {m'_{x,y}}^{2}) Q_{x,y} \right]. \quad (5.46)$$

Network load: Similar to the network coded case, the network load is calculated from the rate of ARQs and retransmissions given by

$$\lambda_{arq\_x,y} = \frac{\lambda_{y,x}}{2} \cdot \left[\binom{2}{1} d_{y,x} d'_{x,y} + {d'_{y,x}}^2 (1 - {m'_{y,x}}^2) \right] \tag{5.47}$$

$$\lambda_{retr\_x,y} = \frac{\lambda_{x,y}}{2} \cdot d'_{y,x} \left[\binom{2}{1} d_{x,y} d'_{x,y} + 2 \cdot {d'_{x,y}}^2 (1 - {m'_{x,y}}^2) \right] \tag{5.48}$$

Using the above, the network load and the information rate can be calculated using Eqns. 5.44 and 5.57 respectively. The total flit loss or modification probability between a source-destination pair is determined by the number of rogue routers encountered in the XY route between them:

$$d_{x,y} = 1 - (1 - p_d)^{N_{x,y}} \tag{5.49}$$

$$m_{x,y} = 1 - (1 - p_m)^{N_{x,y}} \tag{5.50}$$

A retransmission fails if the ARQ is lost or the retransmission is lost or modified. The expression $R_{x,y}$ denotes this probability:

$$R_{x,y} = 1 - d'_{y,x} d'_{x,y} m'_{x,y}. \tag{5.51}$$

S2:Network Coded Transmission Residual Error Probability: In this scheme, the MAC is transmitted with each flit. Since we consider only generation sizes of 2, error happens no flits of a generation are received (${d_{x,y}}^C$) or when with G-1 received flits ($\epsilon_{1x,y} = \binom{C}{G-1} {d'_{x,y}}^{G-1} {d_{x,y}}^{C-G+1}$) received flits and the ARQ is dropped or the retransmission is dropped or modified ($R_{x,y}$). If any one or more of the G-1 received flits are modified, there is error irrespective of the fate of the ARQretransmission of the lost flit, since no more ARQs are allowed. If G or more flit are received ($\epsilon_{2x,y} = \sum_{i=0}^{C-G} \binom{C}{G+i} {d'_{x,y}}^{G+i} {d_{x,y}}^{C-G-i}$), there is a residual error when too many flits are modified for the generation to be successfully decoded even with retransmission:

$$p_1 = \binom{C}{G-1} {d'_{x,y}}^{G-1} {d_{x,y}}^{C-G+1} \tag{5.52}$$

$$p_2 = \sum_{i=0}^{C-G} \binom{C}{G+i} {d'_{x,y}}^{G+i} {d_{x,y}}^{C-G-i} \tag{5.53}$$

$$\begin{aligned} \overline{\epsilon} = \frac{1}{M(M-1)} \sum_{x=1}^{M} \sum_{\substack{y=1 \\ y \neq x}}^{M} \Bigg[& {d_{x,y}}^C + p_1 \cdot ({m'_{x,y}}^{G-1} \cdot R_{x,y} + 1 - {m'_{x,y}}^{G-1}) \\ & + p_2 \cdot \left\{ \binom{G+i}{1} m'_{x,y} {m_{x,y}}^{G+i-1} \cdot R_{x,y} + \sum_{k=2}^{G+i} \binom{G+i}{k} {m'_{x,y}}^k {m_{x,y}}^{G+i-k} \right\} \Bigg] \end{aligned} \tag{5.54}$$

Network load: The gross traffic at each router, $\lambda'_{x,y}$ is a sum of the normally injected flits ($\lambda_{x,y}$) and the ARQs ($\lambda_{arq\_x,y}$) and retransmissions ($\lambda_{retr\_x,y}$), which are directly related to the error cases from above. The network load A is computed by averaging $\lambda'_{x,y}$ over all modules. The rates of ARQ and retransmission are given by:

$$\lambda_{arq\_x,y} = \frac{\lambda_{y,x}}{C} \cdot \left[p_{1yx} + p_{2yx} \cdot \sum_{k=1}^{G+i} \binom{G+i}{k} {m'_{y,x}}^{k} {m_{y,x}}^{G+i-k} \right] \tag{5.55}$$

$$\lambda_{retr\_x,y} = d'_{y,x} \cdot \lambda_{arq\_y,x} \tag{5.56}$$

Information Rate: Due to the splitting over two flits, a factor of $\frac{1}{2}$ needs to be included for the computation of the information rate:

$$I_{NC} = \frac{G/C}{2} \cdot \frac{\sum_{x=1}^{M} \sum_{\substack{y=1 \\ y \neq x}}^{M} \lambda_{x,y}}{\sum_{x=1}^{M} \sum_{\substack{y=1 \\ y \neq x}}^{M} \lambda'_{x,y}}. \tag{5.57}$$

Uncoded Transmission: The expressions for the residual error probability and network load and information rates are very similar to that in the coded case. The main difference is that, when a flit is modified, the ARQ asks for the retransmission of this flit only:

$$\overline{\epsilon} = \frac{1}{M(M-1)} \sum_{x=1}^{M} \sum_{\substack{y=1 \\ y \neq x}}^{M} {d_{x,y}}^{2} + \binom{2}{1} d_{x,y} d'_{x,y} \cdot (m'_{x,y} R_{x,y} + m_{x,y}) + {d'_{x,y}}^{2} (\binom{2}{1} m'_{x,y} m_{x,y} R_{x,y} + {m_{x,y}}^{2}) \tag{5.58}$$

$$\lambda_{arq\_x,y} = \frac{\lambda_{y,x}}{2} \cdot \left[\binom{2}{1} d_{y,x} d'_{x,y} + {d'_{y,x}}^{2} (1 - {m'_{y,x}}^{2}) \right] \tag{5.59}$$

The analytical model was applied to the same scenarios as the simulation. The difference between the model and simulation results for all the estimated parameters were all less than 5% except for of S1 G2C4, where the residual error rate values had a difference of average of 7% (Tab. 5.8).

Table 5.8: Relative error between analytical model and simulation.

	S1			S2		
	UC	G2C3	G2C4	UC	G2C3	G2C4
Error probability	2.8%	5%	7%	1%	1.5%	1%
Information rate	< 1%	< 1%	< 1%	< 1%	< 1%	< 1%
Network load	< 1%	< 1%	< 1%	< 1%	< 1%	< 1%

5.3.6.2 Area Overhead

In this section, we estimate the overhead of the authentication schemes as well as that due to network coding. The area overhead of the solutions is given in Tab. 5.9. The main area overhead results from the mCrypton modules[LK06] used to generate the MACs. MACs must be generated for both flits incoming from and outgoing to the NoC, requiring 39 and 26 cycles for each flit, according to S1 and S2. The crypto modules are shared between sender and receiver side of the NI, with flits queuing up to get access to the modules. Given the flit injection rate of 0.2 flits/node/cycle, the total average rate of flits queuing up for MAC generation is 0.4 flits/node/cycle for S2 and 0.2 flits/node/cycle for S1. According to [Kle75], the flit queuing rate, λ_q should be lower than the processing rate of the crypto modules, μ (1/39 or 1/26) to prevent buffer overflow and long queuing delays. For the case of m parallel servers (in this case crypto modules), the queue utilization rate, $\rho = \frac{\lambda_q}{m \cdot \mu}$. Aiming to have zero queuing delay, we estimated a total number of 18 crypto modules necessary, so that $\rho = 0.5778$ for S2 and 0.4333 for S1. Since the solutions protect the communication of the complete MPSoC, it is justified to compute the area overhead for the total area of an MPSoC. Considering, e.g., the state-of-the-art MPSoC in [H$^+$17] with a total area of 24.43 M GEs and 10 NoCIFs, the area overhead is only $\approx 1.98\%$.

Table 5.9: Overview of area overhead.

Unit	Area per NI
Crypto modules	$18 \times 2681 = 48258$ GEs
Retransmission buffer (depth=10)	$19 \times 10 = 190$ bytes
LUTs (for network coding)	7080 GEs (G2C3) or 16992 GEs (G2C4)

Another factor contributing to the area overhead is the retransmission buffers in the NI. Flits of a generation are transmitted consecutively, so that they should arrive at the receiver one cycle after the other since congestion delay is almost absent due to the low injection rate. An ARQ is sent when there is a gap greater than 8 cycles between consecutive flits of a generation. Assuming a retransmission buffer size of N_b flits, each flit will be in the buffer for $5 \cdot N_b$ cycles before it is overwritten (injection rate of 0.2 flits/node/cycle). Considering the farthest apart node pair (separated by 15 hops, 2 cycles/hop), the ARQ will reach the sender after a round trip delay of $60 + 8$ cycles which should be less than $5 \cdot N_b$. Thus, with $N_b = \frac{68}{5}$ or $\sim$14, the original flit can be found. Modification of a flit is detected after 26 or 39 cycles after receiving the flit and then an ARQ is sent so that $N_b \sim$18 or 20 flits deep. Thus, a very small sized retransmission buffer is needed at each NI. In our reference MPSoC, the highest inter-module distance is 3 hops, so that an even smaller retransmission buffer of depth 10 flits is needed in order to recover the original flit when the ARQ reaches the sender.

Network coding implies further area overhead due to the matrix multiplications (Sec. 5.3.3) in the GF domain. To reduce complexity, we propose to use look-up tables (LUTs) storing all possible product values over GF(2^4). The LUT was implemented in Verilog HDL and synthesized in 65nm CMOS technology and found to have an area of 118 GEs. For generating one combination for S1, 36 table look ups are needed along with the addition (or XOR in GF). S2 requires only 20 look ups. Therefore, to create all the combinations in parallel in one cycle we need a maximum of 144 LUTs (for S1/G2C4) in each NI. Comparing to our reference MPSoC, this is an overhead of 0.7% only. For the decoding of the generation, we need the inverse of a 2×2 matrix containing the coding coefficients, which can be easily achieved using the determinant method. Since fewer multiplications are required for decoding, we do not need more LUTs.

5.3.6.3 Summary

Concerning NoC security, we presented and evaluated authentication schemes which protect the communication in a NoC against active attackers. In summary

- This protection consists of a combination of creating MACs and applying network coding for increased efficiency and robustness. The use of mCrypton as lightweight cryptographic primitive allows for an efficient solution.
- Computation and verification of a MAC requires 26 or 39 cycles for sender and receiver; the overall area overhead is 2.7% in comparison to the total area of a state-of-the-art MPSoC.
- The scheme of including MAC and data in one flit significantly outperforms the scheme of sending the data and MAC in separate flits. The redundancy of network coding reduces significantly the residual error rates but also results in lower information rate.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Motivation

The aggressive scaling of transistor gates into the deep sub-micron has led to the evolution of multi-processor systems on chip massively increasing the on-chip processing power. The design of the communication backbone of such MPSoCs is a challenging task and must take into consideration parameters such as topology, routing, latency and fault resilience. Fault resilience has become an important factor for investigation since the trend to deep sub-micron has introduced permanent and transient faults in the NoC components. Adding redundant components or doing repeating executions is the general approach to fault tolerance and the type of redundancy used depends on the class of fault being dealt with. To investigate fault resilience of NoCs, we not only investigate application of resilience approaches but generate analytic models for these in order to delve deeper into the process. This gives us a deeper insight into the matter and allows to flexibly investigate different scenarios such as different topologies, routing schemes or varying sizes of the network without having to resort to time costly cycle-accurate simulations.

Contributions of the dissertation

- Investigation of adaptive routing for the hexagonal and octagonal NoCs, obtained by addition of redundant links in one and both diagonal directions, respectively to the 2D rectangular mesh NoC. The fault-tolerant routing algorithms were based on the turn model and involves determining the correct combination of prevented turns (in packet traversal) to avoid the formation of deadlock. Correct selection of these turns is a complicated process with the effort increasing with the different possibilities of deadlock cycles. An approach based on the algebraic manipulations of the channel dependency matrix (CDM) allows to determine the possible combinations of these prevented turns and thus allows the formation of adaptive routing algorithms much

simpler. Moreover, the method is very general and can be used for any topology with turn model based adaptive routing.

- Fault-adaptive routing algorithms are by nature complex, since the packet route changes continuously subject to the fault locations and thus, especially difficult to model analytically. We presented an approach to analytically assess the network fault resilience based on further matrix algebra of the CDM. The approach was validated against cycle accurate simulations for the negative first fault tolerant routing algorithms for the mesh and hexagonal NoCs, for networks of varying sizes and was shown to be highly accurate. Since the approach is much faster than cycle-accurate simulations, it allowed us to investigate fault resilience of larger sized NoCs. Further, we were able to investigate analytically scenarios such as the effect of network border faults or the effect of path length on the fault tolerance using this approach.

- When the NoC is affected by transient faults lasting a few cycles at a time, this may cause flits to be lost e.g in the router. Generally, the approach for tolerance in this respect is to either retransmit the lost flit or send redundant flit in advance to compensate for the loss. We modeled analytically this scenario to compare the NoC performance with ARQ and retransmission against random linear network coding, where a linear combination of flits is sent from the sender and then decoded at the receiver. The results showed that network coding is a promising approach to deal with transient/ intermittent faults in the NoC. The analytic model is highly accurate and significantly faster than the simulations, which allowed the flexible investigation of different sizes and topologies of NoC.

- Security is an crucial emerging issue for MPSoCs, since the design and development of the heterogeneous MPSoCs involves many parties and therefore the possibility of malign hardware Trojans insertion at any sage of the development process. We investigate such a scenario where certain rogue routers were assumed capable to drop or modify NoC flits. To authenticate the flits, a message authentication code was included in the flit and the flit was retransmitted if a modification was determined. Moreover, network coding was assumed to protect against the dropping and modification of flits. The investigation in cycle-accurate simulations revealed the advantage of network coding over the uncoded scenario. A fast and accurate analytic model was further developed for flexible investigation.

6.2 Future Research Directions

- In addition to analytic assessment of the fault resilience for the fault-tolerant routing algorithms, it would be useful to be able to determine analytically the effect of adaptive routing on latency and throughput. This may be achieved by extending

models developed in earlier investigations of design space exploration using queuing theoretic models [FÖFF13].

- Investigation of adaptive code rate selection mechanisms for resilient NoC with transient/intermittent failures or for secure NoC.
- Investigation of reliability/security in wireless NoC (i.e. wireless inter-chip or even intra-chip links).
- Investigation of different attack models for NoC security.
- We have investigated within this thesis best-effort traffic only. It would be interesting to analyze the effect of best effort and guaranteed service together.

Bibliography

[ACLY00] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Inf. Theory*, 46(4):1204–1216, 2000.

[ACR14] Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. FortNoCs: Mitigating the Threat of a Compromised NoC. In *Proc. of DAC*, pages 158:1–158:6, 2014.

[AK08] Y. Alkabani and F. Koushanfar. Extended abstract: Designer #x2019;s hardware Trojan horse. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 82–83, June 2008.

[AMN+14] O Arnold, E Matus, B Noethen, M Winter, T Limberg, and G Fettweis. Tomahawk: Parallelism and heterogeneity in communications signal processing MPSoCs. *Embedded Computing Systems (TECS), ACM Transactions on*, 13(107), 2014.

[And11] Jason Andress. The basics of information security: Understanding the fundamentals of infosec in theory and practice, 2011.

[B+93] F. Baccelli et al. *Synchronization and Linearity : An Algebra for Discrete Event Systems*. John Wiley and Sons, 1993.

[Bar64] Paul Baran. *On Distributed Communications: I. Introduction to Distributed Communications Networks*. RAND Corporation, 1964.

[BC95] R.V. Boppana and S. Chalasani. Fault-tolerant wormhole routing algorithms for mesh networks. *IEEE Transaction on Computers*, 44(7):848–864, 1995.

[BD05] Davide Bertozzi and G De Micheli. Error control schemes for on-chip communication links: The energy-reliability tradeoff. *Computer-Aided Design of Integrated Circuits and Systems,IEEE Transactions on*, 24(6):818–831, 2005.

[BK16] Travis Boraten and Avinash Karanth Kodi. Packet security with path sensitization for nocs. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, DATE '16, pages 1136–1139. EDA Consortium, 2016.

[Bor07] Shekhar Borkar. Thousand Core Chips: A Technology Perspective. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 746–749, 2007.

[BS16] Poona Bahrebar and Dirk Stroobandt. Online reconfigurable routing method for handling link failures in noc-based mpsocs. In *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2016 11th International Symposium on*, 2016.

[cfa17] CFAED. `https://cfaed.tu-dresden.de/`, 2017.

[CHKP06] V. Catania, R. Holsmark, S. Kumar, and M. Palesi. A methodology for design of application specific deadlock-free routing algorithms for NoC systems. In *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS). Proceedings*, pages 142–147, 2006.

[CMC14] E. V. Castillo, G. Miorandi, and W. J. Chau. Dyafnoc: Characterization and analysis of a dynamically reconfigurable noc using a dor-based deadlock-free routing algorithm. In *Eighth IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*, 2014.

[CWJ03] Philip A. Chou, Yunnan Wu, and Kamal Jain. Practical Network Coding. In *Proc. Annual Allerton Conf. on Comm., Control, and Computing*, 2003.

[DBNC11] Thuan Duong-Ba, Thinh Nguyen, and Patrick Chiang. Network Coding in Multicore Processors. In *IEEE 30th Int. Performance Computing and Communications Conference (IPCCC)*, pages 1–7, 2011.

[DMB06] Giovanni De Micheli and Luca Benini. *Networks on Chips : Technology and Tools*. Elsevier Science, 2006.

[DPCD09] L. Devaux, S. Pillement, Daniel Chillet, and D. Demignz. R2NoC : dynamically Reconfigurable Routers for flexible Networks on Chip. In *International Conference on Reconfigurable Computing*, 2009.

[DT07] Avijit Dutta and Nur A. Touba. Reliable Network-on-Chip Using a Low Cost Unequal Error Protection Code . In *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2007.

[Dua93] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *Parallel and Distributed Systems, IEEE Transactions on*, 4(12):1320–1331, Dec 1993.

[DYN03] Jose Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann, 2003.

[E+07] Thomas Eisenbarth et al. A Survey of Lightweight-Cryptography Implementations. *IEEE Design & Test of Comp.*, 24:522 – 533, 2007.

[ED05] Samuel Evain and Jean-Philippe Diguet. From NoC Security Analysis to Design Solutions. In *Proc. of IEEE SiPS*, 2005.

[EJP09] Natalie Enright Jerger and Li-shiuan Peh. *On-Chip Networks*. Morgan & Claypool Publishers, 2009.

[F+09] D. Fick et al. Vicis: a reliable network for unreliable silicon. In *Design Automation Conference*, pages 812–816, April 2009.

[FDC+09] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw. A Highly Resilient Routing Algorithm for Fault Tolerant NoCs. In *Design, Automation Test in Europe Conference Exhibition (DATE),2009*, pages 21–26, April 2009.

[FFH09] Y. Fukushima, M. Fukushi, and S. Horiguchi. Fault-Tolerant Routing Algorithm for Network on Chip without Virtual Channels . In *24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2009.

[FLJ+10] 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 *Third International Workshop on Network on Network on Chip Architectures, NoCArc'10*, pages 11–16. ACM, May 2010.

[FLJ+13] C. Feng, Z. Lu, A. Jantsch, J. Li, M. Zhang, and Z. Xing. Addressing Transient and Permanent Faults in NoC With Efficient Fault-Tolerant Deflection Router . *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(6):1053 – 1066, 2013.

[FMLD07] J. Filch, A. Meijia, P. Lopez, and J. Duato. Region-Based Routing: An Efficient Routing Mechanism to Tackle Unreliable Hardware in Network on Chips. In *Networks-on-Chip, 2007. NoCS 2007. First ACM/IEEE International Symposium on*, pages 183–194, May 2007.

[FÖFF13] Erik Fischer, David Öhmann, Albrecht Fehske, and Gerhard P Fettweis. Design space exploration of many-core nocs based on queueing-theoretic models. *International Journal On Advances in Systems and Measurements*, 6(3 and 4):272–286, 2013.

[FY15] J. Frey and Qiaoyan Yu. Exploiting state obfuscation to detect hardware trojans in NoC network interfaces. In *Proc. of IEEE MWSCAS*, pages 1–4, 2015.

[FY16] Jonathan Frey and Qiaoyan Yu. A hardened network-on-chip design using runtime hardware Trojan mitigation methods. *Integration, the VLSI journal*, 56:15 – 31, 2016.

[GN93] C.J. Glass and L.M. Ni. Fault-tolerant Wormhole Routing in Meshes. In *FTCS23. Proceedings*, pages 240–249, 1993.

[GN94] C. J. Glass and L. M. Ni. The Turn Model for Adaptive Routing. *Association for Computer Machinery*, 41(5):874–902, September 1994.

[GZLT06] H. Gu, J. Zhang, Z. Liu, and X. Tu. Routing in Hexagonal Networks under a Corner-Based Addressing Scheme. *IEICE TRANS. INF. and SYST.*, E89-D(5):1755–1758, May 2006.

[H+17] S. Haas et al. A heterogeneous SDR MPSoC in 28nmCMOS for low-latency wireless applications. In *Proc. of DAC*, 2017.

[HKM+03] Tracey Ho, Ralf Koetter, Muriel Médard, David R. Karger, and Michelle Effros. The benefits of coding over routing in a randomized setting. In *Proc. of the IEEE International Symposium on Information Theory*, 2003.

[HLB08] W.H. Hu, S.E. Lee, and N Bagherzadeh. DMesh: a diagonally-linked mesh Network-on-Chip architecture. In *NoCArc, First International Workshop on Network on Chip Architectures*, 2008.

[HM05] Jingcao Hu and R Marculescu. Energy- and performance-aware mapping for regular NoC architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(4):551–562, 2005.

[Ind11] Leandro Soares Indrusiak. Evaluating the feasibility of network coding for NoCs. In *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pages 1–5, 2011.

[IY11] Masashi Imai and Tomohiro Yoneda. Improving Dependability and Performance of Fully Asynchronous on-chip Networks. In *Proceedings of the 17th IEEE International Symposium on Asynchronous Circuits and Systems*, 2011.

[JKM09] Y. Jin, N. Kupp, and Y. Makris. Experiences in Hardware Trojan design and implementation. In *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 50–57, July 2009.

[JTWB09] Slavissa Jovanovic, Camel Tanougast, Serge Weber, and Christophe Bobda. A new deadlock-free fault-tolerant routing algorithm for noc interconnections. In *Proceedings of Field Programmable Logic and Applications. FPL 2009. International Conference on*, 2009.

[KdlTR10] Yana E. Krasteva, Eduardo de la Torre, and Teresa Riesgo. Reconfigurable networks on chip: Drnoc architecture. *Journal of Systems Architecture*, 6(3 and 4):293–302, 2010.

[Kle75] Leonard Kleinrock. *Queueing systems - 1 : Theory.* Wiley, New York, 1975.

[KLM+04] Paul Kocher, Ruby Lee, Gary McGraw, Anand Raghunathan, and Srivaths Moderator-Ravi. Security as a new dimension in embedded system design. In *Proceedings of the 41st Annual Design Automation Conference*, pages 753–760. ACM, 2004.

[KRAT13] Hemangee K. Kapoor, G. Bhoopal Rao, Sharique Arshi, and Gaurav Trivedi. A Security Framework for NoC Using Authenticated Encryption and Session Keys. *Circuits, Systems, and Signal Processing*, 32(6):2605–2622, 2013.

[KYEB15] M. K̈hayambashi, P.M. Yaghini, A. Eghbal, and N Bagerzadeh. Analytic Reliability Analysis of 3D NoC under TSV Failure. *ACM Journal on Emerging Technologies in Computing Systems*, 2015.

[LHLM15] J. Liu, J. Harkin, Y. Li, and L Maguire. Low cost fault-tolerant algorithm for networks-on-chip. *Journal Microprocessors and Microsystems*, 39:358–372, 2015.

[LK06] Chae Hoon Lim and Tymur Korishko. mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In *WISA 2005*, 2006.

[LL04] Zongpeng Li and Baochun Li. Network coding: The case of multiple unicast sessions. In *in Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, 2004.

[LLP07] T. Lehtonen, P. Liljeberg, and J. Plosila. Fault Tolerance Analysis of NoC Architectures. In *Circuits and Systems. ISCAS. IEEE International Symposium on*, 2007.

[MCRG06] S. Murali, M. Coenen, A. Radelescu, and K. Goossens. A methodology for mapping multiple use-cases onto networks on chips. In *Design, Automation and Test in Europe Conference and Exhibition, 2006. Proceedings*, 2006.

[MDP13] E. Masoumeh, M. Daneshtalab, and J. Plosila. High performance fault-tolerant routing algorithm for noc-based many-core systems. In *Parallel, Distributed and, Network-Based Processing, 21st Euromicro Conference on*, 2013.

[MDPT12] E. Masoumeh, M. Daneshtalab, J. Plosila, and H. Tenhunen. MAFA: Adaptive fault-tolerant routing algorithm for networks-on-chip. In *Digital System Design (DSD), 15th Euromicro Conference on*, pages 201 – 207, 2012.

[MF16] Sadia Moriam and Gerhard P. Fettweis. Fault Tolerant Deadlock-free Adaptive Routing Algorithms for Hexagonal Networks-on-chip. In *Digital System Design (DSD), 19th Euromicro Conference on*, pages 131 – 137, 2016.

[MF17] Sadia Moriam and Gerhard P. Fettweis. Reliability assessment of fault tolerant routing algorithms in networks-on-chip: An analytic approach. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '17, pages 61–66, 2017.

[MFW+18] Sadia Moriam, Elke Franz, Paul Walthers, Akash Kumar, Thorsten Strufe, and Gerhard P. Fettweis. Protecting Communication in Many-Core Systems against Active Attackers. In *Proceedings of ACM Great Lakes Symposium on VLSI (GLSVLSI) , Accepted*, 2018.

[MTV+05] S. Murali, T. Theocharides, N. Vijaykrishnan, M. Irwin, L. Benini, and G. Demicheli. Analysis of error recovery schemes for networks on chips. *IEEE Design and Test of Computers*, 22:434–442, 2005.

[MYF+15] Sadia Moriam, Yexin Yan, Erik Fischer, Elke Franz, and Gerhard P. Fettweis. Resilient and Efficient Communication in Many-Core Systems using Network Coding. In *34th IEEE International Performance Computing and Communications Conference (IPCCC)*, 2015.

[N+14] B. Noethen et al. 10.7 A 105GOPS 36mm2 heterogeneous SDR MPSoC with energy-aware dynamic scheduling and iterative detection-decoding for 4G in 65nm CMOS. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE Int.*, pages 188–189, Feb 2014.

[NM93] L. M. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *Computer*, 26(2):70–78, feb 1993.

[PF14] Stefan Pfennig and Elke Franz. Adjustable redundancy for secure network coding in a unicast scenario. In *Proc. International Symposium on Network Coding (NetCod)*, 2014.

[PF17] Stefan Pfennig and Elke Franz. Security Aspects of Confidential Network Coding. In *Proc. of IEEE ICC CISS*, 2017.

[PLB+04] M. Pirretti, G.M. Link, R.R. Brooks, N. Vijaykrishnan, M. Kandemir, and M.J. Irwin. Fault tolerant algorithms for network-on-chip interconnect. In *VLSI, 2004. Proceedings. IEEE Computer society Annual Symposium on*, pages 46–51, Feb 2004.

[PNJ+06] D. Park, C. Nicopoulos, K. Jongman, N. Vijaykrishnan, and Chita R. Das. Exploring fault-tolerant network-on-chip architectures. In *DSN, 2006. Proceedings. International Conference on Dependable Systems and Networks*, June 2006.

[RAS+08] F. Refan, H. Alemzadeh, S. Safari, P. Prinetto, and Z. Navabi. Reliability in Application Specific Mesh-based NoC Architectures. In *IEEE International On-Line Testing Symposium(IOLTS)*, 2008.

[RFZJ13] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch. Methods of Fault Tolerance in Networks-on-Chip. *ACM Computing Surveys*, 46(1), Oct 2013.

[SB13] A. Shamaei and B. Bose. Adaptive routing in hexagonal torus interconnection networks. In *IEEE High Performance Extreme Computing Conference*, pages 10–12, Sep 2013.

[Sim91] Gustavus J. Simmons, editor. *Contemporary Cryptography – The Science of Information Integrity.* IEEE Press, 1991.

[SRG12] Ahmed Shalaby, M. El-Sayed Ragab, and Victor Goulart. Intermediate nodes selection schemes for Network Coding in Network-on-Chips. In *NORCHIP*, pages 1–5, 2012.

[SWS+15] Simha Sethumadhavan, Adam Waksman, Matthew Suozzo, Yipeng Huang, and Julianna Eum. Trustworthy hardware from untrusted components. *Communications of the ACM*, 58(9):60–71, August 2015.

[SZBR07] T. Schönwald, J. Zimmerman, O. Bringmann, and W. Rosenstiel. Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures. In *Digital System Design (DSD), 10th Euromicro Conference on*, pages 527–534, 2007.

[Val11] Mojtaba Valinataj. Evaluation of Fault-Tolerant Routing Methods for NoC Architectures. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, 2011.

[VMPL10] Mojtaba Valinataj, Siamak Mohammadi, Juha Plosila, and Pasi Liljeberg. A fault-tolerant and congestion-aware routing algorithm for networks-on-chip. In *Proceedings of Design and Diagnostics of Electronic Circuits and Systems (DDECS),IEEE 13th International Symposium on*, 2010.

[VMS08] M. Valinataj, S. Mohammadi, and S. Safari. Inherent reliability evaluation of networks-on-chip based on analytic models. In *System-on-Chip (SoC), Internation Symposium on*, 2008.

[VWF+13] Michael Vonbun, Stefan Wallentowitz, Michael Feilen, Walter Stechele, and Andreas Herkersdorf. Evaluation of Hop Count Advantages of Network-Coded 2D-Mesh NoCs. In *23rd Int. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 134–141, 2013.

[WF08] Markus Winter and Gerhard P Fettweis. A Network-on-Chip Channel Allocator for Run-Time Task Scheduling in Multi-Processor System-on-Chips. *Digital Systems Design, Euromicro Symp. on*, pages 133–140, 2008.

[WPG10] M. Winter, S. Prusseit, and P.F. Gerhard. Hierarchical routing architectures in clustered 2d-mesh networks-on-chip. In *SoC Design Conference (ISOCC), 2010 International*, pages 388–391, Nov 2010.

[Wu03] Jie Wu. A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model. *IEEE Transaction on Computers*, 52(9):1154–1169, September 2003.

[YA12] Q Yu and P. Ampadu. Dual-layer adaptive error control for network-on-chip links. *Very Large ScaleIntegration (VLSI) Systems, IEEE Transactions on*, 20(7):1304–1317, 2012.

[Yan15] Yexin Yan. Machbarkeitsstudie zur anwendung von random linear network coding zur effizienten und fehlertoleranten übertragung in network-on-chip, 2015. Studienarbeit.

[Yeu08] Raymond W. Yeung. *Information Theory and Network Coding.* Springer Publishing Company, 2008.

[ZPG07] Haibo Zhu, Partha Pratim Pande, and Cristian Grecu. Performance evaluation of adaptive routing algorithms for achieving fault tolerance in noc fabrics. In *Proceedings of Application-specific Systems, Architectures and Processors. ASAP. IEEE International Conf. on*, 2007.