A

Ph.D. Thesis

on

Design of Highly Adaptive and Fault-tolerant Routing for Networks-on-Chip

submitted as a partial fulfillment of

Ph.D. program

in

Engineering

Place: Jaipur **Date: January 2, 2017**

Department of Computer Science and Engineering

Malaviya National Institute of Technology , Jaipur - 302017

Certificate

We hereby certify that the Thesis titled, "**Design of Highly Adaptive and Fault-tolerant Routing for Networks-on-Chip**" submitted by **Manoj Kumar (2011RCP7126)** is the research work done under our supervision, thus is accepted and finalized for submission in partial fulfillment of Ph.D. Program.

Place: Jaipur

Supervisor(s):

Manoj Singh Gaur, (Professor)

Date: January 2, 2017

Vijay Laxmi, (Associate Professor)

Declaration

I, Manoj Kumar, declare that I own the research work introduced in this Thesis titled, "Design of Highly Adaptive and Fault-tolerant Routing for Networks-on-Chip" and the research contents used in this thesis. I with this assure that:

- The research work produced in this thesis is for the partial fulfillment of the degree of "*Doctor of Philosophy*" at MNIT, Jaipur.
- I have stated all the major resources used for the help.
- Where I have used proper citation for the work proposed by other researchers and quoted the source. This entire thesis belongs to me with the some exception of such citations.
- Where I have taken references of previously published work of my co-authors and other researchers and this is always clearly attributed.
- I have clearly stated any part of this Thesis that has been previously submitted for a degree or any other qualification at MNIT or any other institution.

Signed: ________________________________

Date: ________________________________

Acknowledgement

It was a matter great pleasure for me to enroll for Ph.D. program in the Department of Computer Science and Engineering, Malaviya National Institute of Technology, Jaipur, India. I would like to express sincere thanks and appreciation to my Ph.D. supervisors Dr. Manoj Singh Gaur and Dr. Vijay Laxmi. I am also thankful to Dr. Seok-bum Ko (Professor, University of Saskatchewan, Canada) and Dr. Masoud Daneshtalab (Associate Professor, University of Mlardalen, Sweden) for their continuous technical/non-technical support. They have devoted valuable time to guide me. Their suggestions and cooperation together with the motivation have made it possible to complete my research work. They have provided me a good atmosphere for doing work peacefully as well as resourcefully.

I am also grateful to Dr. Parvez Faruki (Lecturer, Polytechnic College, Bhavnagar, Gujrat) for his continuous motivation during Ph.D. duration.

I convey special thanks to my DREC committee members Dr. Dinesh Gopalani (MNIT, Jaipur), Dr. Girdhari Singh (MNIT, Jaipur) and Dr. Lava Bhargava (MNIT, Jaipur) for their inspiration and encouragement that directed me toward the completion of this research.

My special thanks to my wonderful friends and fellow researchers, Smita, Rimpy, Anil, Gaurav, Sonal, Ashish for revitalizing each day. From the depth of my heart, I acknowledge all those persons who have motivated me to complete the thesis. I also express my apology that I couldn't mention them in person.

Finally, at the end, I would like to thank God, the Creator of the Earth, the Almighty. The Supreme in power and knowledge and the most merciful, Whom I acquired guidance and knowledge to do something beneficial for the humanity.

Dedications

This Thesis is Dedicated to my wife ***Mahima****, who was always with me to motivate for the work and my Kids* ***Cheena and Cherry****, who were always ready to play with me to make me fresh.*

Abstract

Network-on-Chip (NoC) has emerged as a promising alternative to traditional bus-based architectures for inter-core communication. It has also been accepted commercially as the communication paradigm for Systems-on-Chip (SoC), instead of dedicated wires or shared buses. The overall NoC performance depends on many parameters such as topology, flow control, routing schemes, task mapping, quality-of-service and switching methods. In all cases, the *degree of adaptiveness* has a significant effect on the overall performance of any adaptive routing algorithm. In addition, *fault tolerance* is the another aspect related to the reliability of NoCs. The objective of this thesis is to design and develop turn models for routing algorithms that provide high degree of adaptiveness and/or fault tolerance to improve the overall performance of the network.

Most of the routing algorithms proposed in the recent literature achieve deadlock-freedom by forcibly restricting certain routing turns so that the channel dependency graph remains acyclic. This requirement for the deadlock-freedom makes these algorithms more restrictive, thus reduces the degree of adaptiveness. The proposed algorithm enhances the functionality (routing turns) in virtual channels of existing algorithms to achieve high degree of adaptiveness for 2D mesh with wormhole switching. We have extended the aforementioned 2D turn model for three dimensions (3D) to achieve high adaptivity. The 3D turn model uses one, two and two virtual channels in X, Y and Z dimensions, respectively. We have also developed a highly adaptive and fault-tolerant routing algorithm for 2D mesh topology. The proposed algorithm can handle single link faults. It can also handle multiple link faults if fault's boundary do not overlap with each other. We have designed another fault-tolerant routing algorithm for 2D mesh topology. The proposed algorithm is able to handle single link faults, and it uses LBDR as its implementation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The current Chip Multiprocessors (CMPs) and Multi-Processor Systems-on-Chip (MPSoCs) architectures have replaced the single core systems due to their scalability issues. Intel Lab has developed terascale processor [98], a single chip cloud computer consists of 48 cores on same silicon CPU chip. This development is intended of achieving the scalable communication, power consumption and on-chip performance for the near future. Other industrial examples include Tilera's *TILE*-Gx72 and *TILE*64^{TM} [4] processors. These examples show the increasing growth of integrating multiple components on a single silicon chip, and this will continue till near future. As the number of cores is increasing, communication among them is becoming challenging for these multi-core architectures. This growth leads to the paradigm shift from computation-centric designs to communication-centric designs.

In the beginning, the integrated chips had deployed a small number of processing elements and interconnected them using the conventional interconnects such as ring [26], cross-bar, shared-buses and point-to-point (P2P). In P2P techniques, a pair of processing elements is interconnected via dedicated link (wire). Thus, because of this dedicated link, the P2P interconnects based systems are able to offer 100% communication bandwidth. However, as the number of cores are increasing, it results in increased wire density, thus, for multi-core architectures, these are not suitable candidates. The shared-bus architectures are simple as far as the number of processing elements is less. With an increase in the number of such elements, the arbitration delay is also increasing and results into the bandwidth bottleneck. Crossbar designs interconnect a set of cores to another set of cores in a matrix fashion. However, the power and area constraints prevent their use in multicore systems. The ring-based communication systems interconnect the processing elements in a close loop (ring topology). In ring architecture, the communication becomes slow because the messages must cross other components of the network between source and the destination. Moreover, if one of the elements gets down, it affects the whole network. In short, the increasing complexity, bandwidth bottleneck and communication delay are the main limiting factors of traditional communication architecture based System-on-Chip (SoC). To address these communication issues for multi-core architectures, Network-on-Chip (NoC) has been introduced as a viable and scalable substitute to traditional bus-based architectures for intercore communication.

The overall performance of NoC depends on several network parameters such as topology, switching technique, flow control, and routing strategies. This thesis is focused on routing algorithms. A routing algorithm affects several nonfunctional requirements of an NoC-based system. Performance, reliability, energy consumption, power dissipation, thermal aspects, and fault tolerance are among the major parameters that are affected by the routing algorithm. Significant research has been published on the improvement of routing algorithms for the parallel and distributed computing domains. The main issues addressed include the development of high-performance, fault-tolerant and low cost (power and area) router micro-architectures, the development of bandwidth-aware and contention-aware selection policies, and the design of deadlock-free highly adaptive routing functions. In all cases, routing function (one phase of routing algorithm) has a significant effect on the overall performance of any routing algorithm [46].

1.1 Routing Overview

A routing algorithm computes the path from source node to the destination node. The routing algorithm can be either deterministic or adaptive. The deterministic routing scheme always provides the same path between source and the destination nodes irrespective of network congestion status. However, an adaptive scheme uses current network status in making the routing decision and may compute different output channel for different packets. The deadlock is a major issue in designing any adaptive routing algorithm. A deadlock occurs when a set of packets cannot advance toward their destination because the buffers requested by them are full [26]. The blocked packets form a deadlock configuration. Packets, involved in deadlock configuration, are waiting for each other in a cyclic manner to release the resources (buffer), thus blocked forever.

The deadlocks are severe to any routing algorithm as they can disrupt the communication by paralysing network operations. To avoid deadlocks, NoC designers use either deadlock avoidance or deadlock recovery. In deadlock avoidance methods, it is ensured that a deadlock never occur. On the contrary, deadlock recovery method detect deadlock in the network, then recover from it. The deadlock avoidance methods are generally preferred over deadlock recovery methods because recovery methods impose an overhead of deadlock detection and then buffer releasing that may degrade the network performance considerably.

To avoid deadlocks, researchers have used two main theories; Dally's theory [17] and Duato's theory [24]. The first theory does not allow cyclic dependencies among channels. Whereas, the other one allows cyclic dependencies provided that there exists deadlock free routing subfunction. The details of routing algorithms and deadlocks are discussed in Sections 2.5 and 2.6.

1.2 Motivations

With deterministic routing, packets can be routed over single output channel at each node. Thus, it is mandatory to remove all cyclic dependencies between network channels in order to achieve deadlock freedom. In adaptive routing, packets often have several options for routing at each node. Thus, it is not mandatory to eliminate all cyclic dependencies between channels, provided that each packet can be forwarded on a route whose channels are not involved in the cyclic dependencies. The channels involved in these acyclic routes are considered as *escape channels* from deadlocks (cycles). Based on the extensive literature survey on challenges in routing algorithm, following are the motivations of this thesis:

1. The first motivation for the proposed routing algorithms is derived from the fact that a less restrictive routing algorithm offers a high degree of adaptiveness [46]. Most of routing algorithms proposed in the recent literature [70, 36, 68, 71, 45, 34, 32, 33] achieve deadlock-freedom by forcibly restricting certain routing turns so that the CDG remains acyclic. This

acyclic CDG requirement for the deadlock-freedom makes these algorithms more restrictive, thus reduces the degree of adaptiveness. If we relax this requirement for these algorithms, we can be blessed with a less restrictive routing algorithm. The main focus of this proposed work is to relax this requirement by allowing cycles in the CDG provided that ECDG is acyclic (using Duato's theorem [24]). We have explained our point by comparing with two recent algorithms LEAR [34] and HARA [32]. It should be noted that LEAR and HARA routing algorithms are based on Mad-y [45] turn model.

We present an example which shows that a routing scheme would be deadlock-free, even if there exist cyclic dependencies between channels. Figure 1.1 shows a ring consisting of four routers (1, 2, 3 and 4). Each router is connected to the adjacent router using two channels a_i and b_i, $i = \{1, 2, 3\}$ (physical or virtual) except the routers 4 and 1. Router 4 is connected to router 1 using a single channel a_4.

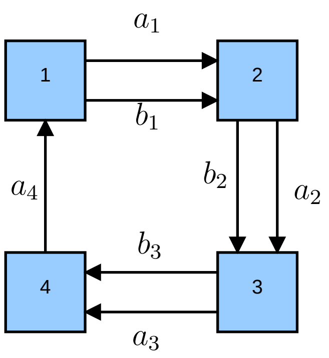


Figure 1.1: Ring architecture for four routers

The routing function R for the ring is defined as below

$$R = \begin{cases} a_i & , \forall j \neq i \\ b_i & , \forall j > i \end{cases}$$

Figure 1.2 shows the channel dependency graph for the routing function R. The dependencies are because of followings:

(a) If a packet is at the router 1 and the destination is router 3, it can occupy channel a_1 and request channels a_2 and b_2.

(b) If a packet is at the router 2 and the destination is router 4, it can occupy channel a_2 and request channels a_3 and b_3.

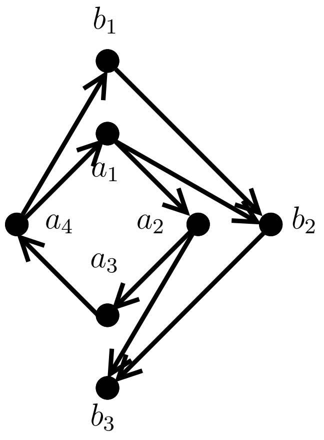


Figure 1.2: Channel dependency graph for the routing function R

(c) If a packet is at the router 3 and the destination is router 1, it can occupy channel a_3 and request channel a_4.

(d) If a packet is at the router 4 and the destination is router 2, it can occupy channel a_4 and request channels a_1 and b_1.

We can observe that there exist cyclic dependencies among a_i channels. However, it is clear from Figure 1.2 that the b_i channels are not involved in any cyclic dependencies. Thus, they are always free (empty). We prove the deadlock-freedom of R by contradiction. We assume that there is a deadlock in routing function R. There must exist a deadlock configuration, and any deadlock configuration can involve only a_i channels as the b_i channels are always empty. Although, Figure 1.2 depicts that the a_i channels involved in the cyclic dependencies, but this deadlock configuration will not be legal. At router 1, channel a_1 can be occupied by a packet that is destined for either router 2 or 3 or 4. At router 2, this packet can use b_2 and b_3 (always empty channels) channels for these destinations thus breaking the cyclic dependency. We have proved that the routing function R with cyclic dependencies does not have any deadlock-configuration. Thus, it would be deadlock-free.

2. In on-chip communications, reliability is a critical factor in multi-core systems. It is affected by transient and/or permanent faults. Faults affect the functionality and can degrade the performance of on-chip networks. Thus, it has become essential to design on-chip networks that can tolerate faults. The second motivation for this thesis to achieve fault-tolerance to make communication more reliable.

1.3 Objectives

The objectives of this thesis are as stated below:

1. To design a turn model to provide high degree of adaptiveness for the 2D mesh network. The ultimate aim is to add more functionality (routing turns) to virtual channels of existing algorithms minimally or non-minimally around hotspot and congested areas of the network to achieve the high degree of adaptiveness.
2. To extend the 2D turn model for three-dimensional mesh to improve the network performance.
3. To implement reconfigurable and fault tolerant routing algorithms by analyzing deadlock-freedom to bypass faulty links of the network.

1.4 Contributions of the Thesis

The major contributions of this thesis are as follows:

1. We propose a novel turn model that provides high degree of adaptiveness for a 2D mesh. The end result is that the proposed turn model reduces the number of restrictions on routing turns and hence is able to provide path diversity through additional minimal and non-minimal routes between the source and destination. Based on turn model, we have presented a highly adaptive and congestion-aware routing method (CHARM). The proposed method is implemented using double-y network.
2. We have extended aforementioned 2D turn model for three dimensions (3D) to high achieve adaptivity. It uses one, two and two virtual channels in X, Y and Z dimensions, respectively.
3. We have developed a highly adaptive and fault-tolerant routing algorithm for 2D mesh topology. The proposed algorithm is able to handle single link faults within a 2D mesh. It can also handle multiple link faults if fault's boundary do not overlap with each other.

4. In an MPSoC or CMP architecture, the reliability of NoC is affected by transient and/or permanent faults. Faults affect the functionality and can degrade the performance of on-chip networks. We have designed a fault-tolerant and reconfigurable routing algorithm for the 2D mesh topology. The presented approach can handle single link faults within the 2D mesh, and it uses Logic Based Distributed Routing (LBDR) as its implementation.

In this thesis, we mainly focus on the development of highly adaptive and fault-tolerant routing algorithms for mesh networks with wormhole flow control. Our proposal tries to identify non-essential restrictions and remove these to improve adaptivity of underlying architecture in terms of routing function. The effectiveness of proposed algorithms is shown by evaluating them for both real and synthetic traffic profiles. In addition, power and area overheads are also analyzed.

1.5 Thesis Outline

The remainder of the thesis is partitioned into three categories i.e. the introductory chapter, contributory chapters and the concluding chapter. The introductory chapter (Chapter 2) demonstrates the background required for understanding the concepts of on-chip networks. The contributory chapters include four chapters, each presenting the proposed work. Chapter 3 presents the highly adaptive and congestion-aware routing method for the 2D mesh. A three dimensional extension to proposed 2D turn model is presented in Chapter 4. It also presents the analysis how this two-dimensional method can be applied to three dimensions. Chapter 5 provides the fault tolerance analysis of the 2D turn model. It also introduces fault tolerant routing technique for a 2D mesh. Chapter 6 provided another fault tolerant and reconfigurable routing method using LBDR. It can handle all single link failures. Finally, Chapter 7 concludes the research work proposed in this thesis and provides future directions.

Chapter 2

Networks-on-Chip (NoCs)

The device scaling has resulted in an exponential increase in the circuit performance that sustained the recent microelectronic evolution. The 28nm silicon technology is already in use for the production. A single chip with 32nm technology consists of billions of transistors with a descent chip density of 1.5 $Mgate/mm^2$. Decreasing returns and the increased design complexity (chip density, etc.) of a single processor system have resulted in the emergence of many-core architectures in the form of multi-core Systems-on-Chip (SoCs).

An SoC is a complex computing system which consists of processing elements, data converters, hardware accelerators, peripheral interfaces, I/Os, on-chip memory, and other components. For a specific application domain, it represents a complex device that provides all the needed hardware and electronic circuits to form a working system. An SoC is generally customized for a specific application which is very similar to the traditional application-specific integrated circuit (ASIC). However, the main emphasis is not on specialized hardware design for the SoCs, like conventional ASICs. SoCs utilize reusable existing components as much as possible so that these can reduce the production of newly designed elements of the chip. These reusable components known as intellectual-property (IP) cores and include both the soft and hard core components.

In the current Silicon-technology era, the main development perspective is shifted towards increasing the number of cores in multi-core systems from increasing the operating frequency of a single core processor. As the number of cores is increasing, the communication among the cores is becoming more challenging for these multi-core architectures. These trends have resulted in a paradigm shift towards communication-centric designs from computation-centric designs.

2.1 Why NoC: Evolution of On-chip Communication Architectures

A communication infrastructure is required to interconnect various application specific integrated circuits (ASICs), memories, processors and other elements on a single-chip. The basic requirements for a good communication infrastructure include high performance, low power, and low latency. Thus, the efficient implementation of SoCs requires scalable and bandwidth-aware communication infrastructure. The two major technological challenges in the design of an efficient communication architecture for SoCs are (i) reducing the huge computing power (ii) handling the large amount of traffic generated by number of concurrently running applications.

Traditionally, SoCs employ communication schemes [26], namely point-to-point, rings, crossbars, and buses. By connecting a pair of elements with a dedicated link, the point-to-point architectures (Figure 2.1) provide the advantage of 100% bandwidth availability. However, these schemes are not well scalable in terms of cost, design efforts, complexity, and flexibility. As the number of cores increases, these also result in the increased wire density, thus not suitable for the many-core SoCs.

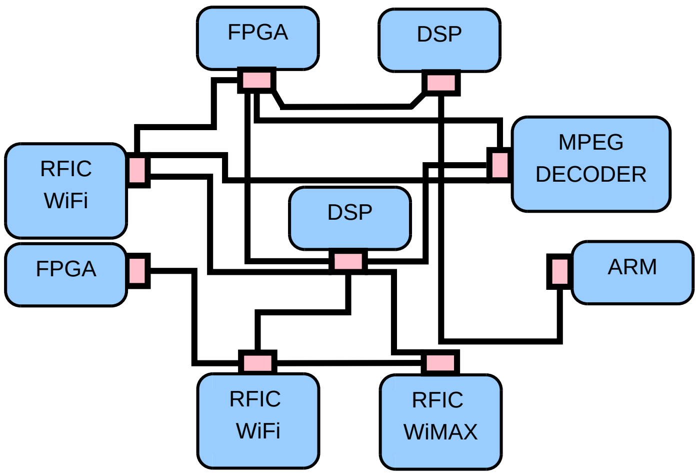


Figure 2.1: Point-to-point communication architecture

In order to eliminate dedicated link requirement of the point-to-point architectures, the bus-based architectures (Figure 2.2) can be used. These architectures interconnect a small number (few tens) of cores in a cost-effective manner and result into reduced overall design complexity. The main advantage of bus-based

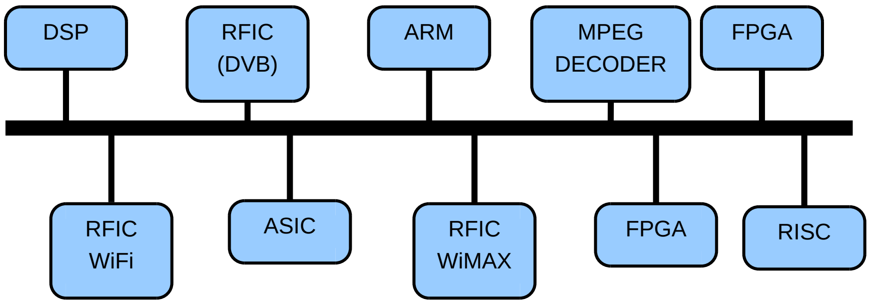


Figure 2.2: Shared-bus communication architecture

architectures is their simplified interconnection design. However, due to sharing property, these architectures limit the maximum achievable bandwidth. In addition, electrical noise, variability, and crosstalk are growly serious issues with recent technology. The use of long global wires makes the shared bus systems more vulnerable to these problems. Moreover, the addition of new components can deteriorate performance of buses drastically. Because, with miniaturization, the propagation delay of long global wires is monotonously increasing and thus, with new chip generations, the clock frequency of the bus system is reducing.

The segmented (hierarchical) bus structure provides a solution to the bandwidth constraints of the shared bus interconnect. As shown in Figure 2.3, it interconnects a bus structure with another bus structure using a bridge component. The computation of optimum settings for the segmented bus is very time-consuming and complex [14]. Because, the bus arbitration of segmented bus is distributed in nature which combines the operations of all arbitration units.

Crossbar designs interconnect a set of cores to another set of cores in a matrix fashion. However, the power and area constraints prevent their use in multicore systems. The ring-based communication systems interconnect the processing elements in a close loop (ring topology). In ring architecture, the communication becomes slow because the messages must cross other components of the network between the source and the destination. Moreover, if one of the elements gets down, it affects the whole network. In short, the increasing complexity, bandwidth bottleneck and communication delay are the main limiting factors of traditional

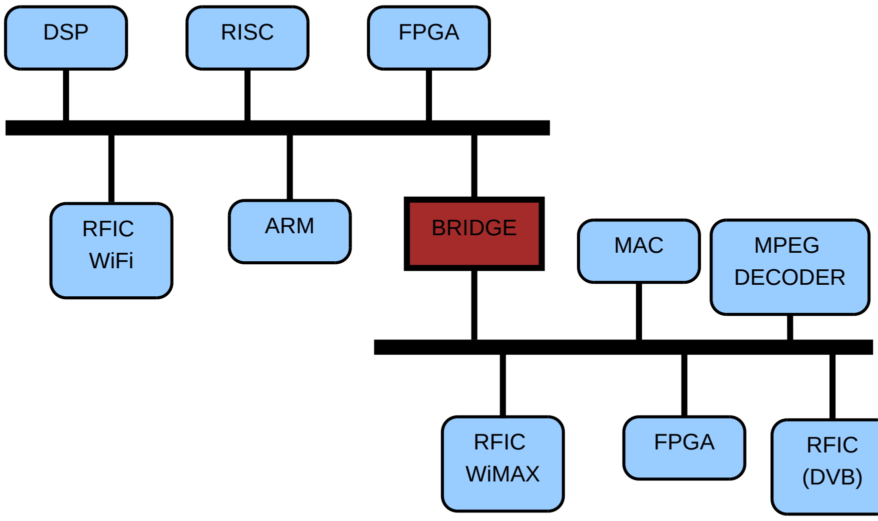


Figure 2.3: Segmented-bus communication architecture

communication architecture based System-on-Chip (SoC). To address these communication issues for multi-core architectures, Network-on-Chip (NoC) paradigm is introduced as a viable and scalable substitute to traditional bus-based architectures for intercore communication.

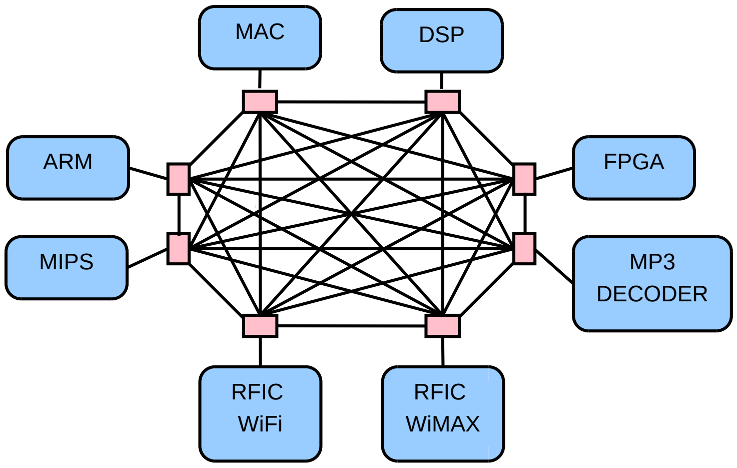


Figure 2.4: Fully-crossbar communication architecture

Figure 2.5 shows 3×3 mesh on-chip network. The main idea behind the development of NoC as a communication paradigm is to route the packets instead of the wires [19]. To route a message/packet generated by one core to another, the NoC uses several routers or switches.

The main purpose of designing SoC is to develop applications which are generally specific to embedded system domain. We can classify the SoCs architectures on the basis of the use of NoC into two major categories; Chip Multiprocessors (CMPs) and MultiProcessors System-on-Chip (MPSoCs). If the multi-core SoC consists of heterogeneous core elements, the SoC is known as MPSoC. If the cores of SoCs are of homogeneous in nature, the SoC is known as CMP. The cores communicate with each other by sending and receiving packet to/from other cores. This interaction is generally of parallel nature in the embedded applications. The major designing factors of these multi-processor architectures include low communication latency and low power. In portable electronic appliances and electronic-handhelds, the power is completely dependent on the battery life. Thus, the main issue to design the NoC-based CMPs/MPSoCs architectures is the power constraint that depends on the chip area.

In CMP architectures, a tile includes three components; IP-core, a router and a network interface. The IP-core consists of an IO interface, a memory controller (MCtrl), a global (shared) memory, a local memory block, some CPU blocks and some other elements. Each router is connected to the IP-core using a network interface. The purpose of the network interface is to decompose a data packet into a number of flits when one core sends the data to another. The network interface is also responsible for the assembling the flits to reconstruct the data packet at the receiving tile. Similarly, an MPSoC architecture can have different types of IP-cores such as FPGA-based configurable block, an ASIC component, a bus-based microprocessor system (such as RISC, MIPS, ARM or processor system), a digital signal processor (DSP), a shared memory, or any other type of IP-core.

The NoC architecture has already been used as the communication backbone for current embedded SoC applications. The Xbox-360 [2] and Cell Broadband Engine Processor [64] gaming consoles are examples of potential commercial products of multiprocessor applications which rely on NoC as communication architecture. The Xbox-360 gaming console is an example of CMP architecture comprised of graphics processing unit (GPU), memory, I/O units and CPU units. The units are connected using node crossbar/queuing as the number of elements are small. The node crossbar/queuing is similar to the single crossbar-switch of a NoC router. Cell Broadband Engine Processor (also known as Cell Processor) is a result of joint efforts of Toshiba, Sony, and IBM. It incorporates a 64-bit power processor element (PPE), eight specialized processors called synergistic processor elements (SPEs), a high-bandwidth bus interface, and a high-speed memory controller. All units are interconnected through four slotted rings and integrated on chip. The Cell Processor is dedicatedly designed for Playstation 3 Game Console. The on-chip network

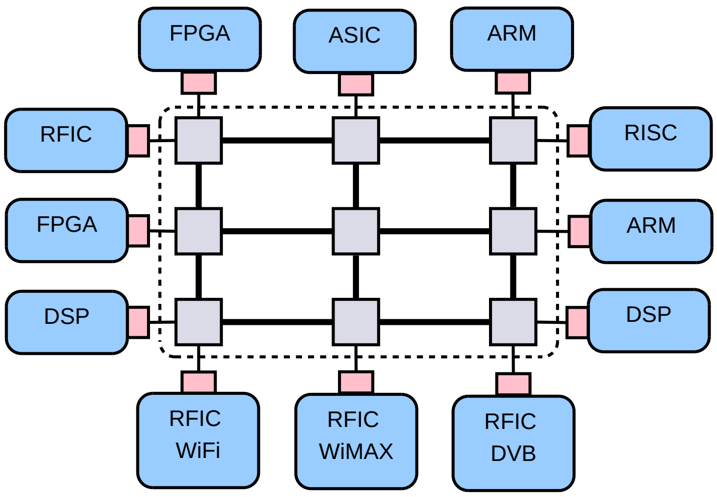


Figure 2.5: Network-on-Chip (NoC) communication architecture

paradigm is also accepted by the industry as the communication infrastructure for complex Systems-on-Chip (SoC) to replace shared buses and dedicated wires. Intel's terascale processor [98] and Tilera's $TILE\text{-}Gx72$, $TILE64^{TM}$ [4] processors are the few commercial examples who have adopted mesh based NoC.

Implementation of an NoC includes several designing characteristics such as network topology, flow-control methods, routing methods, quality-of-service and switching methods. This chapter describes the basics and general theory [26, 85, 20] about these characteristics of interconnection networks.

2.2 NoC Topologies

The network topology defines how the different IP-cores of an MPSoC/CMP are interconnected to each other through routers and network interfaces. As the selection of specific flow-control method and routing algorithm depends largely on the network topology, the very first step in designing on-chip network is the selection of the topology. The network topology not only defines the static arrangement of network components (routers, channels, and NIs), it also specifies other details such as the bit-rate and bandwidth of each channel, the number of stages, bisection bandwidth, path diversity, and the radix of the router.

The network designers select a specific topology on the basis of its performance and the implementation complexity-cost. The network performance is defined by the communication bandwidth, power utilization and network throughput. The implementation complexity-cost has three main components; (a) length and density of wires (b) router degree (number of channels at each router) (c) number of metal layers needed to realize the network.

A network node may act as either router node or terminal node or both. A router node routes packets from input ports to output ports. A terminal node acts as a source and sink for the packets. On the basis of the role of the node in the network, a topology can be classified into two categories direct and indirect. In a direct topology, every network node acts as both a router and a terminal. On the other hand, in an indirect topology, a network node acts as either a router or a terminal. In a direct network, packets are routed directly from one terminal node to other. With an indirect network, packets are routed through a sequence of intermediate router nodes between the source and the destination. Till now, the majority of the NoC designs have utilized direct networks. The main advantage of a direct network is that a router node is co-located with the terminal node so that each router can use various resources of a terminal node. Moreover, it is more suitable to place routers with the terminal nodes in area-constrained systems like NoCs.

2.2.1 Direct Topologies

A variety of network topologies has been modeled and designed using their graph-theoretical characteristics for on-chip networks. The majority of the implemented topologies are orthogonal topologies. In an orthogonal topology, all of its vertices (nodes) are organized in an n-dimensional orthogonal space, and every edge (channel) is organized in such a way that it produces a displacement in a single dimension. Thus, we can number the network nodes using their coordinates in the n-dimensional space. The advantage of using orthogonal topologies is that the hardware implementation of a routing algorithm becomes efficient.

Figures 2.6, 2.7 and 2.8 show the most commonly used orthogonal direct topologies; 2D-dimensional mesh (k-ary n-mesh), 2D-torus (k-ary n-cube) and 3D-hypercube. Mesh topology (Figures 2.6) is among the most popular and important NoC topology for large-scale MPSoCs and CMPs architectures. The main advantages of mesh topologies are their good scalability, regularity, and simplicity. Most of the industrial and academic MPSoCs and CMPs use mesh-based NoC topologies as their communication backbone.

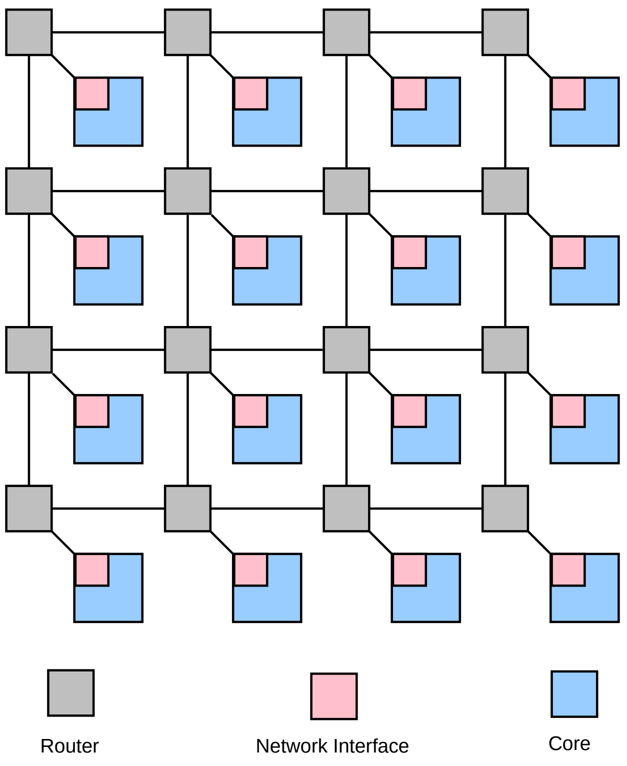


Figure 2.6: 2D-Mesh topology

Formally, an n-dimensional mesh has $k_0 \times k_1 \times ... \times k_{n-2} \times k_{n-1}$, k_i nodes along each dimension i, where $k_i \geq 2$ and $0 \leq i \leq n-1$. Every node X is represented by n coordinates, $(x_{n-1}, x_{n-2}, ... , x_1, x_0)$, where $0 \leq x_i \leq k_i - 1$ for $0 \leq i \leq n-1$.

In torus topology, as shown in Figure 2.7, each node has the same degree. The mesh topology has the disadvantage of increased long hop count between opposite borders. The torus topology eliminates this disadvantage. In this topology, the routers at the border are directly connected to the routers at the opposite border. Thus, it reduces the hop count on a path. Because of edge symmetric property, the torus is able to distribute traffic more evenly across the network. The hypercube (Figure 2.8) is another commonly used topology for 3D NoCs. It is considered as a special case of mesh topologies.

2.2.2 Indirect Topologies

Figure 2.9 shows an example of 2-ary 3-fly butterfly topology. The butterfly topology is a clear example of the difference between router node and the terminal node. The router nodes are in the middle (shown by rectangles) and terminal nodes are

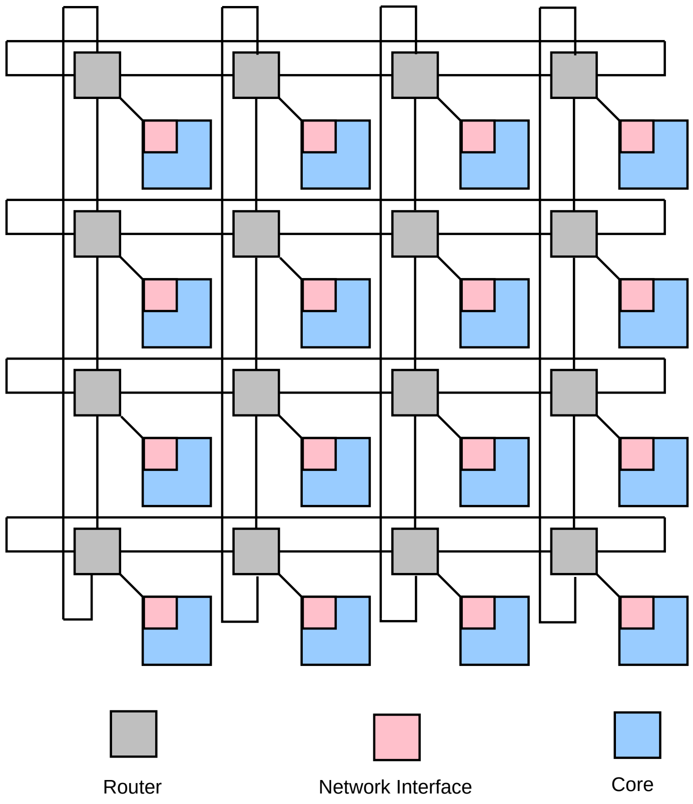


Figure 2.7: 2D-Torus topology

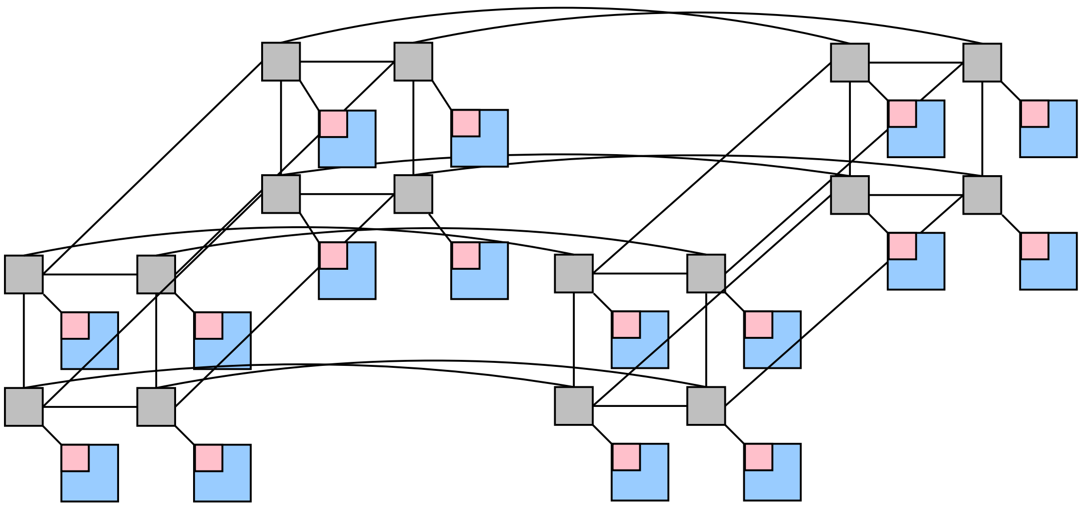


Figure 2.8: 3D-Hypercube topology

on the borders. The router node accepts packets at its input port and computes their destinations using underlying routing algorithm, then routes packets to the output channel.

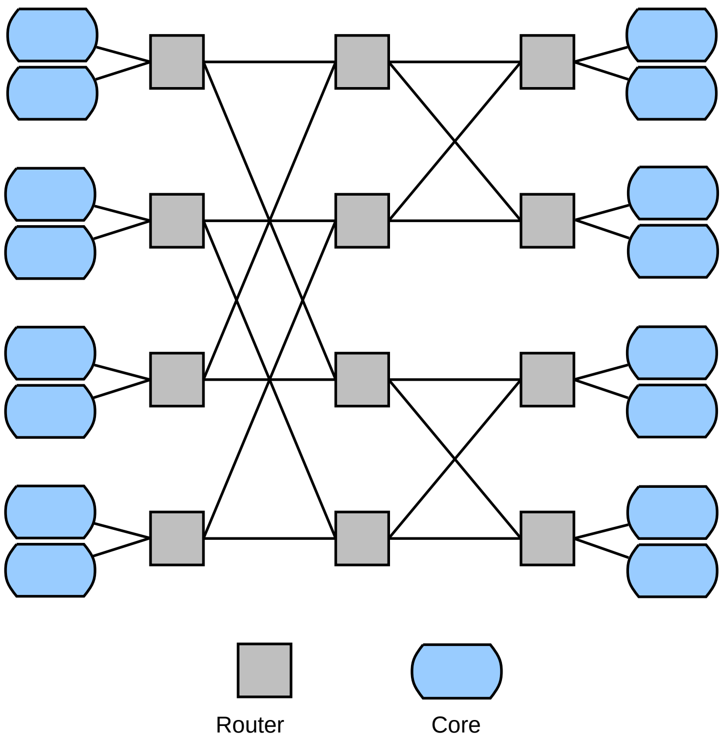


Figure 2.9: Butterfly topology

Figure 2.10 shows an example of fat tree topology that logically represent a binary tree. In a fat tree, packets are forwarded towards the root till they reach a common ancestor and then forwarded down towards the destination. Each router in the fat tree has a logical degree of four. Though, the links in higher-level nodes are much wider than those in the lower levels.

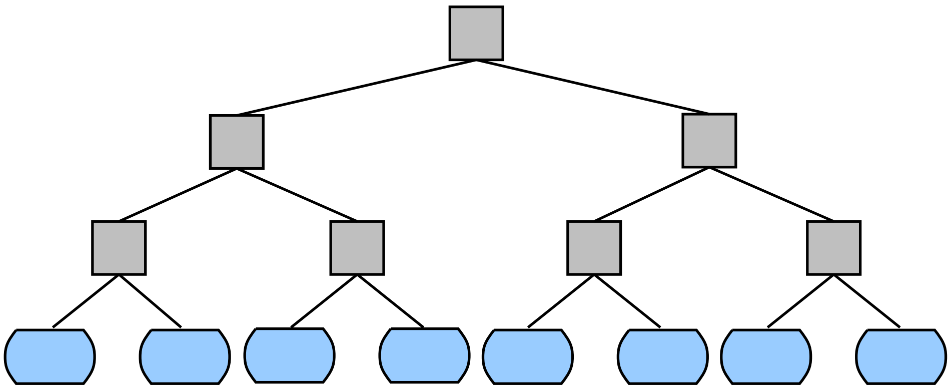


Figure 2.10: Fat-tree topology

2.3 NoC Router: Generic Architecture

A major challenge towards the multi-core architecture is the designing an efficient router microarchitecture. The performance factors for a NoC router includes latency with area and power constraints. A router is capable of receiving packets at the input ports, computes the output port depending upon the destination address using the routing algorithm, and then routes the packet along the appropriate output channel. For a specific on-chip network, the router microarchitecture is generally unique. It depends on various network characteristic such as quality of service, chosen routing method, switching mechanism implementation, flow-control methods, and quality of service.

Figure 2.11 shows a generic router microarchitecture for a 2D mesh topology. It is composed of five input-output (I/O) ports, one for each of the directions (North, South, East, West, and Local). The Local I/O port is connected to the corresponding IP-core through a network interface. Typically, an NoC router has five major components as following:

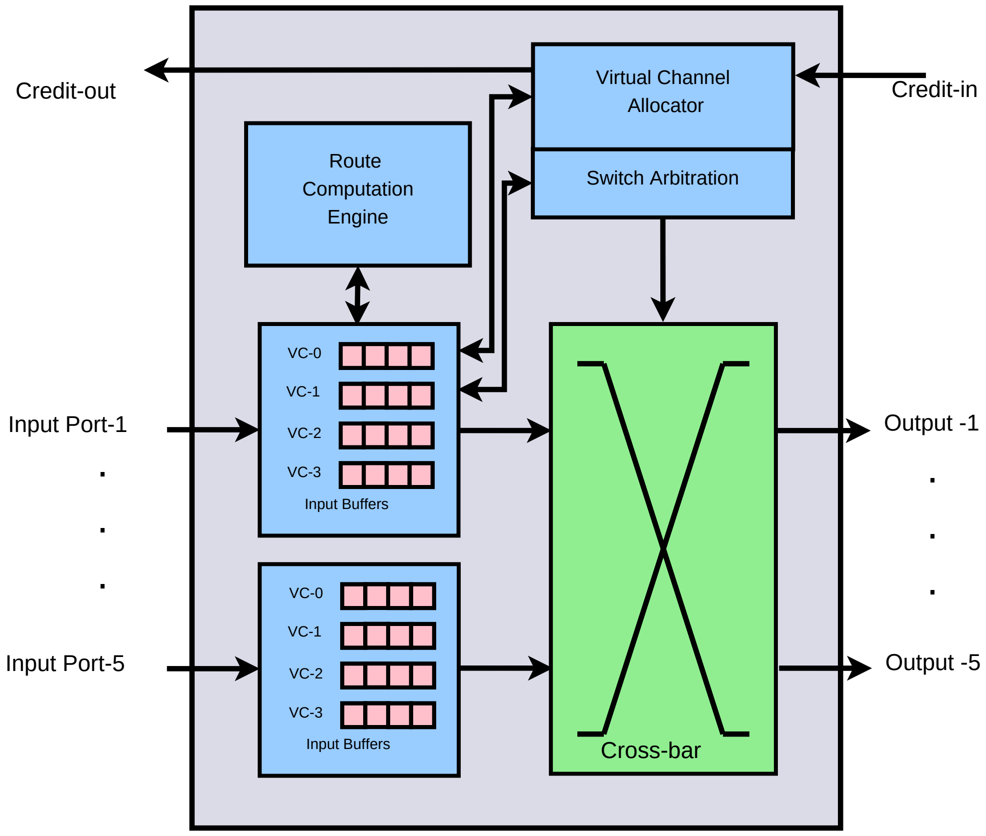


Figure 2.11: Generic NoC router architecture

1. Buffer: The First-In First-Out (FIFO) buffers are used to hold the transiting packet. Each input and/or output physical channel is associated with one buffer. The transit packet can be incoming and outgoing packet in a router. Depending upon the requirement, different NoC routers implement FIFO buffers either at input ports or at output ports or at both places. This implementation reduces the data buffering cost. If a packet (say p1) occupies a buffer for a channel, the other packet (say p2) cannot access the physical channel, even if p1 is blocked. In an alternative appraoch, the buffer can be multiplexed into the number of virtual channels in order to provide quality of service and deadlock-freedom. These logical channels are multiplexed across the physical channel as shown in the Figure 2.11. As the buffer adds significant cost (power and area) to a router, some recent router implementations [100, 42] have eliminated the need of virtual channels.

2. Route Computation Engine: The task of Route Computation Engine is to compute the appropriate output channel for an incoming packet. This unit implements the routing scheme. Typically, NoC routers make use of two different implementations of the route computation engines: table based routing and routing finite state machine. The details of routing algorithms and their implementations are discussed later in Section 2.5.3

3. Arbiter: Multiple packets can request same output channel simultaneously. The purpose of arbitration unit is to resolve these concurrent requests for the same output channel. It provides the arbitration among them. When a packet requests a particular output channel and if that channel is occupied by the other packet, the first packet waits in the input virtual channel (buffer). Once, the occupied channel is released by the other packet, the first packet again participates in the arbitration and is routed along the requested channel, if wins in arbitration.

   Thus, the role of an arbiter is similar to a referee who resolves the contentions among several packets requesting the same output channel in a router. The NoC routers implement various arbitration policies such as flit-by-flit rotating, contention-aware, priority-based, round-robin, first-come first-serve, etc.

4. Crossbar: The Crossbar unit is responsible for interconnecting the input ports to the output ports of the router. The arbitration unit controls the

output data of the crossbar. All possible input data lines are linked with the input ports of the crossbar.

5. Link Controllers: The Link Controllers are responsible for the movement of packets through the physical channel which interconnects the input and output ports of adjacent routers. This data transmission control is required to prevent incorrect data replications and data overflow. The link controllers coordinate the transmission units of flow control on either side of a physical link. Existing control techniques like stop-and-go and credit-based are generally implemented for this unit. This unit is also responsible for implementing data synchronization interfaces to synchronize correct data transfer from one router to other. The NoC routers implement various flow control policies such as pipelined repeater-based, asynchronous queue-based, mesochronous, source-synchronous, etc.

2.4 Switching Methods

A switching method is responsible for the effective management of network resources. The buffers and channels are counted as the main resources of an interconnection network. The switching method determines when and how a packet (or message) header is allocated the network resources during its travel along the route. One can also view the switching method that determines the connection of an input channel to an output channel. A good switching mechanism allocates the network resources efficiently so that the packets are delivered to their destinations with low latency and high throughput by efficient sharing of channels and buffer among them.

Flow-control methods are tightly coupled with buffer allocation and switching methods. The flow-control methods establish the connection between adjacent routes. They control the flow of the information by stopping and allowing it. When a packet is stopped, it needs some buffer area to get stored. If the flow-control method finds that the available buffer space is not enough to store the information, it stops the flow. As soon as, it detects the availability of the buffer space, it again starts the information transmission. In the absence of flow control mechanism, the packets may get dropped.

On-chip router architecture depends on the selection of the switching method. The switching method also defines the services provided by the network. In this section, we brief about commonly used switching methods such as circuit switching, store-and-forward, virtual cut-through and wormhole.

2.4.1 Circuit Switching

We begin our discussion with the circuit-switching method that works at coarsest granularity level and generally uses bufferless flow control. In circuit switching, prior to the transmission of data, the entire route (circuit) is first reserved. A packet header containing the destination address together with some control data is injected into the network. As the packet header advances towards the destination in the network, it reserves the required resources along the route. If the header is unable to acquire the required resource immediately at an intermediate router, it has to wait for the resource till being released. Once the header reaches the destination, an acknowledgment is sent back to source router indicating that the entire route followed by the header is setup. As soon as, acknowledgment packet reaches the source router, the source can then start the transmission of data packets over the reserved path. Once all data packets are sent, a control packet is transmitted in the network to terminate the connection circuit.

The circuit switching technique is generally employed for providing guaranteed-throughput and/or guaranteed-bandwidth in data transfer. In high-performance computing domain, Intel iPSC/2 [80] has used circuit switching technique in parallel machines. These machines use a Direct Connect Communications Technology. Another example that uses circuit switching is Motorola-based BBN GP 1000 [7]. This machine deploys butterfly topology: a multistage connection network. In on-chip networks, the circuit switching techniques is used to provide quality of service. Æthereal [87], MANGO [6], PNoC [52], DSPIN [83] are a few on-chip network proposals which use the circuit switching technique.

The main advantage of circuit switching is that it can transmit the data at full hardware bandwidth. It is useful for long and infrequent messages. However, it may block other messages, because, it reserves the entire route during the transmission of a particular message. Thus, this switching mechanism is inefficient as it wastes expensive channel-bandwidth in order to save relatively inexpensive buffer space.

2.4.2 Store-and-Forward Switching

In circuit switching, once the path is reserved, the entire message is sent along the circuit till the transmission ends. Alternatively, before injecting the message into the network, it is divided into small fixed-length units called packets as shown in Figure 2.12. Each packet can be further divided into a number of flow control unit(s) called flit(s). There are three types of flits: Head flits, Body flits, and Tail flits. The first flit of each packet contains control and routing information, some data (optional) and is called head flit. Body flits contain the data. Tail flit contains some data (optional), and control information that shows that this flit is last flit for a particular packet. Flits can be further segmented into phits that represent the width of the channel. This switching method is called packet switching. The routing algorithm computes the route for each packet and forwards the packet individually from the source to the destination.

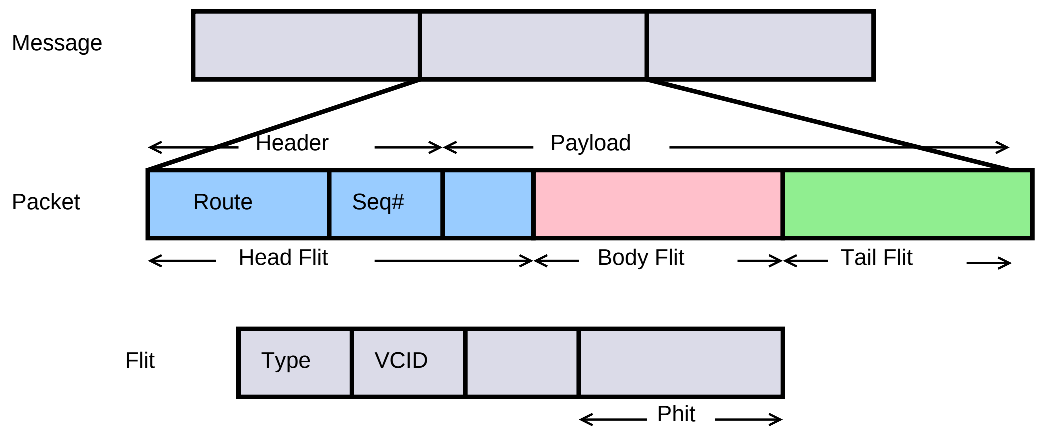


Figure 2.12: Message Composition

The store-and-forward switching is a kind of packet switching technique in which each router is provided with a buffer space equal to the size of the packet so that it can hold a packet completely. Before forwarding the packet to the neighboring router, the packet is stored at each intermediate router. Since every packet is first stored and then forwarded to the next hop, this switching is called store-and-forward switching. Figure 2.13 depicts the time-space diagram for store-and-forward switching. A packet consisting of five flits is routed with no contention assumption. The path length is four hop. Before moving to the next channel, the complete packet is routed along one channel at each step of routing.

The store-and-forward technique is the first switching that has been used in many

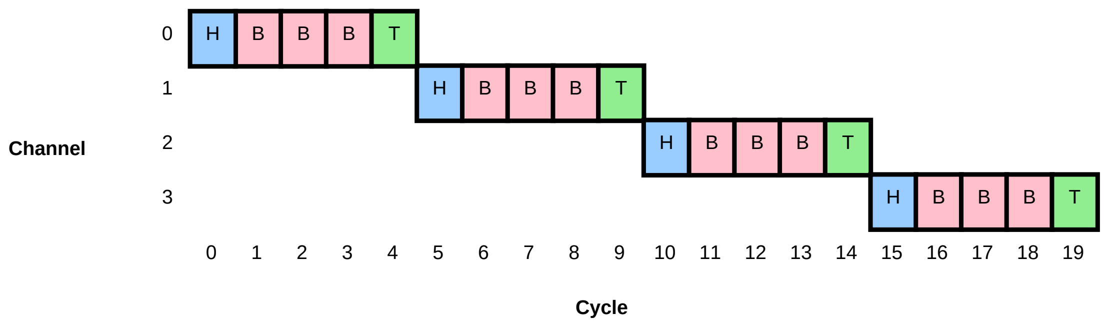


Figure 2.13: Timing diagram for store-and-forward switching

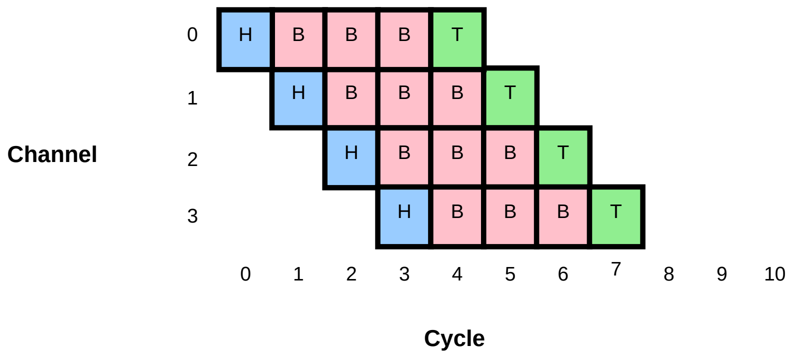
parallel machines. The Manchester Dynamic Dataflow computer [50], the MIT Tagged Token Dataflow system [79] and the Denelcor HEPmachine are the few examples of earlier parallel systems that have deployed store-and-forward switching technique. Similar to the off-chip network field, Arteris [74], CLICHE [67], MicroNet [99], MESCAL [93], Nostrum [76] and Proteo [94, 89] form a short list of early NoC proposals that have used store-and-forward switching technique.

The store-and-forward is preferred when packets are frequent and short. This technique fully utilizes the communication channel when there is data to be sent, unlike circuit switching, where a set of reserved channels may be idle for a substantial period. In this switching, several packets of the same message can travel simultaneously in the network without waiting for the acknowledgment of first packet. However, dividing a message into the number of packets results in delay. Moreover, this switching needs additional time to assemble the packets into the original message at the destination. As the route computation is done at every intermediate router, it also causes overhead.

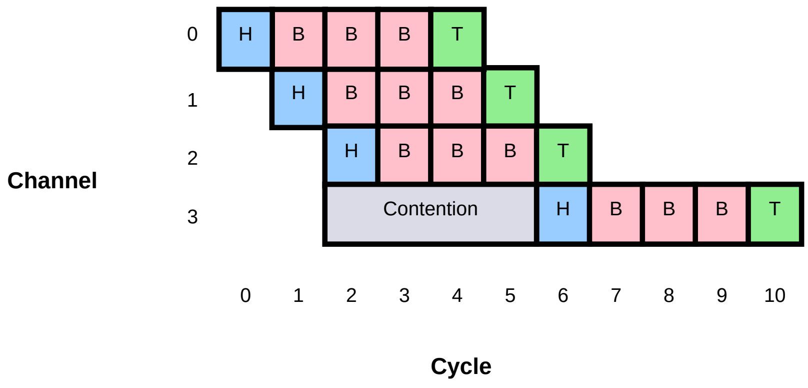
2.4.3 Virtual Cut-Through Switching

In store-and-forward switching technique, a packet must be entirely buffered before it can move to the next hop. Thus, each router experiences serialization latency that results in high average network latency. The virtual cut-through switching is an improvement over the store-and-forward that tries to reduce the serialization delay required to buffer the complete packet at each router. It allows the packet to move to the next hop router soon after the routing decision is made for the packet header, if the next router is having enough buffer space needed to store the packet header at least. Thus, in this switching, it is not required to buffer the complete packet at the current intermediate router and can be cut through to the next router. The packet can be effectively pipelined via consecutive routers. However, when the packet header is blocked at an engaged router, it is buffered at that router. Therefore, virtual cut-through switching works as store-and-forward switching at high traffic loads.

Figure 2.14 depicts the time-space diagram for virtual cut-through switching. A packet consisting of five flits is routed with no contention assumption (Figure 2.14(a)) and with contention assumption(Figure 2.14(b)). Once any of the flits of the packet is arrived at the router, it is transmitted to next hop router if channel and buffer space is available at neighboring router.



(a) For low traffic load



(b) For high traffic load

Figure 2.14: Timing diagram for virtual cut-through switching

The virtual cut-through and store-and-forward techniques are packet-based techniques and thus has one main drawback. The buffer allocation must be in units of packets that makes these switching techniques very inefficient in terms of the buffer area. In the next Section, we have discussed an effective use of buffer space where the allocation of buffer space is performed in units of flits.

The router of Alpha 21364 [77] system and Chaos Router [65] are the examples that have used virtual cut-through switching technique. On-chip network proposals such as IMEC NoC [3] and SPIN [49] have also used the virtual cut-through switching technique.

2.4.4 Wormhole Switching

The major shortcoming associated with packet switching is that the buffer size must be at least of the packet size that makes the design of fast and compact on-chip router difficult. The wormhole switching operates in a similar way like virtual cut-through with one difference. It reduces the buffer requirement of the router significantly as the buffer space is allocated in units of flits. The buffer of wormhole router can be set as small as possible which reduces the hardware cost. A packet like a worm may spread over several channels (holes in the ground) during its flow. Thus, this switching is referred as wormhole switching. Other packets will not be able to reserve the channels that are occupied by a particular packet. The wormhole switching is preferred in the modern NoC-based systems like Teraflops [98], TRIPS [48] and Tile64 [4].

The interconnection networks researchers have presented some hybrid switching mechanisms such as buffered wormhole switching (BWS) [1] and pipelined circuit switching (PCS) [44]. BWS is a variation of the wormhole switching, and it combines the wormhole switching and packet switching aspects. The BWS was firstly presented and used in IBM Power Parallel SP2 [51]. PCS is a combination of the features of circuit and wormhole switching methods. The other switching techniques include Scouting switching [21, 25] and Mad Postman switching [59]. The scouting switching is introduced to enhance the capability of PCS technique to handle faulty channels. It also improves the performance of PCS technique.

2.5 Routing Algorithms

After topology selection, next logical step is the selection of a particular routing algorithm. A routing algorithm is responsible for computing the path that a packet should follow from the source node to the destination node. In this section, we introduce concepts and background of various routing strategies. As mentioned earlier in the Section 2.2, the selection of a specific routing scheme generally depends on the underlying network topology. We have presented routing concepts and background needed for the mesh-based network only as this thesis proposes routing algorithms for mesh networks.

2.5.1 Deadlocks in Routing

The major issue related to every routing strategy is deadlock-freedom. A deadlock is a situation in which some packets cannot proceed further toward their destination as the resources (channels) requested by them are occupied by some other packets. In a deadlocked configuration, all packets are waiting for each other in a cyclic manner to release the resources, thus blocked forever. However, it is possible that a packet is permanently blocked in the network because the destination node does not consume it. This category of deadlock is produced at the application level, and we have not considered this type of deadlock in our thesis and thus, is beyond the scope of this thesis.

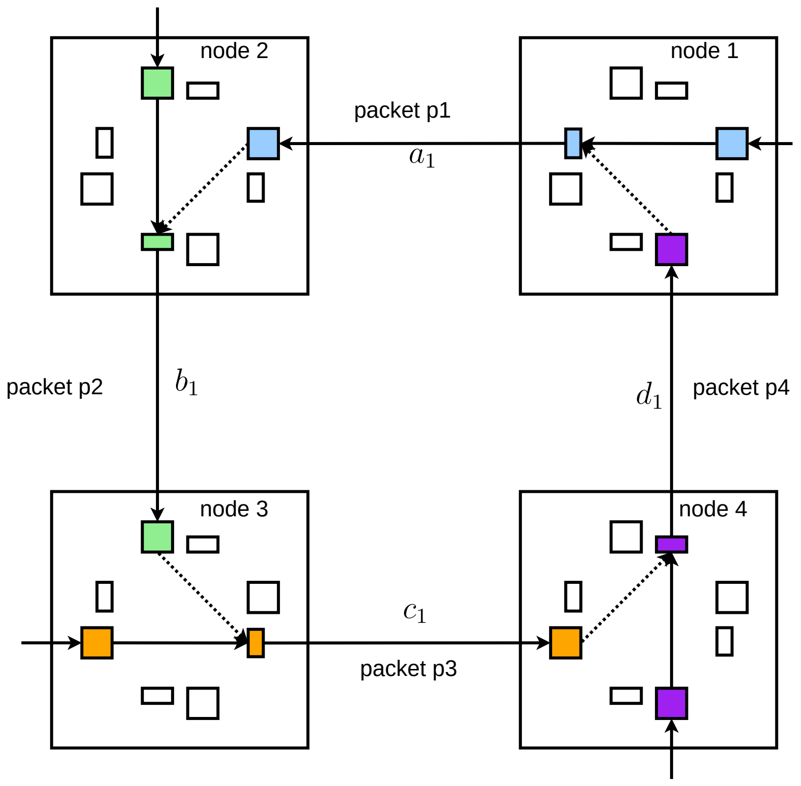


Figure 2.15: Deadlock configuration

The existing routing algorithms utilize either deadlock avoidance or deadlock recovery to achieve freedom from deadlocks. In deadlock avoidance strategies, some constraints are imposed on the routing algorithm such that deadlock can never occur. On the other hand, deadlock recovery strategies do not force any constraint on routing functions, thus allow deadlock to occur. These strategies need a method first to detect and then resolve deadlock configuration. If the deadlock is detected, some channels are deallocated (packets holding those channels are generally aborted) to resolve the deadlock. These schemes are useful only when deadlocks rarely occur [26]. Otherwise, the overhead produced by deadlock detection and buffer releasing would degrade the network performance considerably.

Figure 2.15 illustrates an example of deadlock configuration. This deadlock configuration is formed because four packets (p1, p2, p3 and p4) are involved in cyclic wait forever. At node 1, a packet p1 is coming from East direction and occupies the channel a_1 for the destination node 3. This packet p1 is waiting for the channels b_1 that is occupied by the packet p2. The packet p2 is coming from the North direction and has node 4 as its destination. Similarly, channels c_1 and d_1 are occupied by the packets p3 and p4, respectively. Packets p3 and p4 are coming from the West and South directions, respectively and have node 1 and node 2 as their destinations, respectively. All the packets are involved in cyclic wait and cannot advance further.

The deadlock configuration has formed because there exist cyclic dependencies among channels a_1, b_1, c_1 and d_1. The packets are allowed to take any turn either clockwise or anti-clockwise. This cyclic dependency can be eliminated if packets are not allowed to take one turn (called prohibited turn) in each of clockwise or anti-clockwise cycle. The prohibited turns remove one dependency from each of the cycles, thus avoids the deadlock configuration. We have discussed routing schemes derived from the prohibition turns (simply turn model) in section 2.6. In another type of routing (deflection routing), deadlock-freedom is achieved using the idea that no. of input channels must be equal to the no. of output channels. Therefore a incoming packet will always has free output channel. In deflection routing, no network deadlock is possible [81], as all incoming flits are always routed outside of the router, without having to check in advance that the neighbor router has buffer space.

2.5.2 Livelocks in Routing

Livelock is a situation where a packet never reaches its destination. However, it is possible that packets are not in a deadlock configuration, and they may be

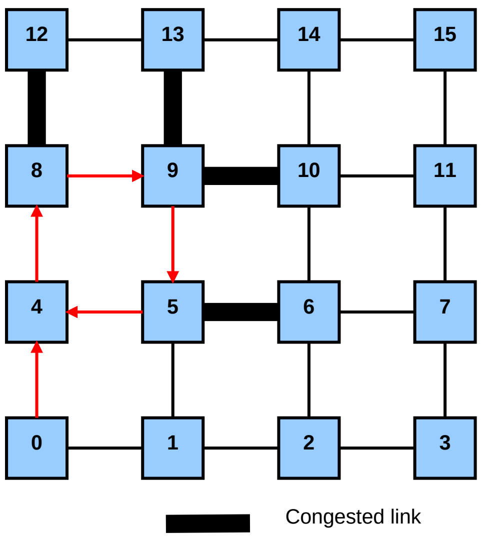


Figure 2.16: An example of livelock

in livelock. A packet may be moving around its destination and never reaching it. The livelock occurs because every time, the channels requested by the packet to reach the destination, are held by some other packets. For minimal routing, livelock never occurs because, at every node, a packet moves one step towards its destination and finally reaches it. However, in the case of non-minimal routing, packets may follow non-minimal paths and livelock can occur, if measure are not taken to guarantee the progress.

Figure 2.16 shows an example of livelock. The source node is 0 and destination node is 12. A packet from 0 to 12 finds congestion at 8 and is misrouted to 9. At node 9, it finds more congestion and is again misrouted to 5. This causes a cycle in which the packet takes two steps forward from 5 to 8, and then two steps back, from 8 to 5.

2.5.3 Classification of Routing Algorithms

Routing algorithms can be classified in several ways depending upon different criteria. These criteria include the place of routing decision, the number of destinations, adaptivity, minimality, network congestion status, topology, etc. In this section, we present several classification of routing algorithms on the basis of these criteria.

2.5.3.1 Source vs. Distributed Routing

On the basis of the location where the routing function computes the route for a packet, routing algorithms can be classified as source based routing and distributed routing. In source routing, source node computes the complete path that the packet should follow from the source node to the destination node. After the computation of the path, it is stored in the packet header. The intermediate nodes extract and utilize information stored in the header of the packet to route it further towards the destination. As the packet advances in the network, the intermediate routers are required to perform following tasks:

- decode the packet header to extract the route,
- check the next computed direction in header, and
- route the packet to the particular output channel

Source-based routing requires less complex routers as the routers only need to perform a small functionality for every incoming packet. For source-based routing, the packet header should be long enough to store the router ids corresponding to the longest distant path, i.e., equal to network diameter. Thus, for a network having the diameter (d), the packet header is required to store the d number of router ids at most. As the network diameter grows with the network size, header overhead increases accordingly. This fact makes source routing sometimes impractical for NoC. On the other hand, in distributed routing, header only contains the destination address. The routing decision is taken at every intermediate router by extracting destination address. Thus, the header size remains constant as the network size increases that make it appealing for current NoC based MPSoCs and CMPs architectures. However, the router architecture of distributed routing is more complex than source based routing because the route computation is performed at every intermediate router.

Hybrid schemes use a combination of both. These schemes are known as multiphase routing. In multiphase routing, the source node computes some destination nodes. Routing between the computed nodes is performed in a distributed fashion.

2.5.3.2 Unicast vs. Broadcast vs. Multicast Routing

On the basis of the number of destinations of a packet, routing algorithms are categorized as either unicast or broadcast or multicast. In unicast routing, there is one to one mapping between the source and the destination. It means that a packet which has raised the routing request, is intended for a single destination node. Whereas, in broadcast routing, the packet is forwarded to all the other nodes of the network. This type of routing is used when the same information is required to be distributed among all the nodes. For example, some type of notifications in case of cache coherency. In multicast routing, a packet is routed to a group of destination nodes.

2.5.3.3 Deterministic vs. Adaptive routing

Routing algorithms are classified as either deterministic or adaptive [26] on the basis of inclusion of current network information in the routing decision. Deterministic routing algorithms always choose the same route between a given pair of source and destination nodes. These do not consider the current network status, even if there exist multiple possible routes. The most popular dimension order routing (DOR) algorithms such as XY and YX are examples of deterministic routing algorithms for 2D meshes. These algorithms first forward the packet along one dimension and then forward to the other dimension. Whereas, adaptive routing algorithms use the current state (traffic and/or link status) of the network in making the routing decision. NoC designers are inspired to develop adaptive routing algorithm because of two main motivations, i.e. to avoid the packet to get into congested (hotspot) regions and to avoid the faulty links and/or nodes of the network. Adaptivity of a routing algorithm can boost the NoC performance (better throughput and latency) than deterministic routing, especially under non-uniform (bursty and hotspot) traffic patterns [46, 5].

The adaptive routing algorithms can be further categorized as fully adaptive and partially adaptive routing algorithms. The fully adaptive routing algorithms can use any of the shortest routes between a pair of source and destination nodes. In partially adaptive routing algorithms, some packets can use all available shortest paths between source and destination nodes depending upon the destination direction, whereas some packets cannot. It means for some destinations; routing algorithm behaves like fully adaptive and for some destinations it behaves like deterministic.

2.5.3.4 Minimal vs. Non-minimal Routing

According to their minimality, adaptive routing algorithms can be classified as minimal (profitable) or non-minimal (mis-routing). Minimal adaptive routing algorithm only provides shortest routes from the source to the destination. At each node, it generates a productive output channel vector that suggests which output channels of the current node will make the current packet closer to its destination. Non-minimal adaptive routing algorithms no longer restrict packets to move along a minimal path towards the destination. Packets may be misrouted over output channels, which move them temporary away from the destination.

However, minimal routing methods guarantee the shortest route from the source to the destination, it is imprudent to neglect the promising performance improvements provided by non-minimal routing schemes. For example, if all output channels corresponding to the minimal routing paths are faulty (or congested); routing the packets along a non-minimal route may be the only (or good) alternative. The non-minimal adaptive routing algorithms are susceptible to livelock [26, Page 83] and [20, Section 10.3], thus must be designed carefully.

2.5.3.5 Congestion-aware vs. Congestion-oblivious Routing

The functionality of a routing algorithm is modeled using two functions: route computation and selection [26]. The route computation function produces an output-channel vector for a packet. This output-channel vector consists of eligible output channels for the packet. The selection function is responsible for selecting one output channel from this vector. The selection may be on the basis of the current status of the output channels.

On the basis of the selection function used, the routing algorithms can be classified as either congestion-oblivious or congestion-aware. In congestion-oblivious algorithms, routing decisions are independent of the current congestion status of the network. This policy may disrupt the load balance since the network status is not considered. On the contrary, congestion-aware routing algorithms consider the current congestion status of the network in their routing decisions. The current congestion level is determined using local or global information. In approaches considering local traffic conditions, the routing decision is made only based on the congestion status of adjacent neighbors. These methods provide a limited thus less precise view of the network condition. Routing algorithms based on global information provide a better distribution of the traffic load. However, the collection of global information, and then the use of this information in the selection of output channels make global congestion-aware routing methods more complex. Congestion-aware algorithms can take advantages of different metrics such as the number of free buffer slots, available virtual channels, crossbar demand, or combinations of these factors.

2.5.3.6 Topology Dependent vs. Topology Independent Routing

Routing algorithms that consider the underlying network topology in making routing decisions are known as topology dependent algorithms. These algorithms are sensitive to topology change and may not work once the topology is modified. Algorithms such as DOR and OE [13], which are based on various turn models designed for 2D meshes, are few examples of topology dependent algorithms. The topology independent routing algorithms [40] also known as topology agnostic routing algorithms. These are agnostic to the particular topology and insensitive to any topology modification. Segment based routing (SR) [75] and up/down [91] are few examples of topology agnostic routing algorithms.

2.6 Turn Model based Routing Algorithms

Turn model representation is an effective way to describe routing algorithm and its restrictions. A packet moving towards direction A makes a routing turn A-B, if the packet turns towards direction B. For example, a turn East-North taken by a packet means that the packet is moving from East to North channel. In mesh networks, routing turns can be of three types as shown in Figure 2.17:

1. $0°$ **(0-degree)**: If a packet switches from one channel to another channel in the same direction
2. $90°$ **(90-degree)**: If a packet switches from one channel to another channel in the different dimension

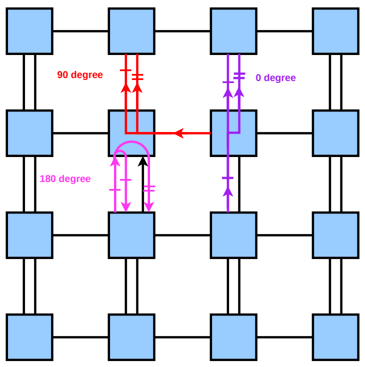


Figure 2.17: Different types of turns for mesh network

3. 180° **(180-degree)**: It is also known as U-turn. If a packet switches from one channel to a channel in opposite direction of same dimension

For minimal routing, 180-degree turns cannot be used.

2.6.1 Partially Adaptive Algorithms

In a mesh topology, each abstract cycle (clock-wise and anti-clockwise) contains four different turns as shown in Figure 2.18.

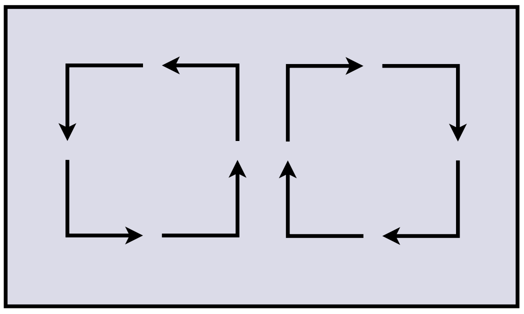


Figure 2.18: Abstract cycles for 2D mesh

Figure 2.19 shows the turn model representation of dimension order routing (XY). The solid lines are used for the permitted turns, and the dashed lines are used for the prohibited turns. In order to prevent deadlocks, the XY turn model prohibits four 90-degree turns of abstract cycles. The prohibited turns are South-West, North-West, South-East and North-East turns. The remaining four turns cannot cause cyclic dependencies. Because of these four prohibited turns, the XY routing becomes static (deterministic). The XY routing is also called X-first routing as it routes packets first in X-dimension. Once, it completes routing in X-dimension, it routes packets in Y-dimension. It is a minimal routing algorithm as it always computes the shortest path for each pair of the source and destination. Algorithm 1 describes the routing function of XY routing algorithm.

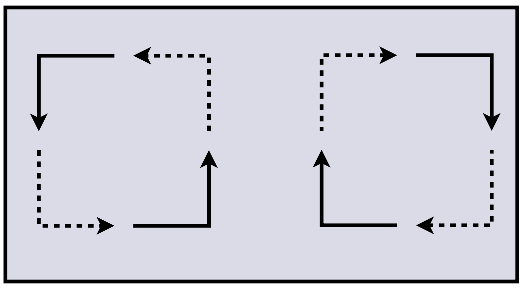


Figure 2.19: XY turn model

Algorithm 1 Deterministic XY (X-first) Routing Algorithm

```
$(X_c, Y_c)$ : Current router coordinates
$(X_d, Y_d)$ : Destination router coordinates
$\triangle x = (X_d - X_c)$
$\triangle y = (Y_d - Y_c)$
if $(\triangle x == 0$ && $\triangle y == 0)$ then
    Output_Dir = Local
    /* destination is local core */
else if $(\triangle x < 0)$ then
    Output_Dir = West
    /* destination is in West direction */
else if $(\triangle x > 0)$ then
    Output_Dir = East
    /* destination is in East direction */
else if $(\triangle x == 0$ && $\triangle y < 0)$ then
    Output_Dir = South
    /* destination is in South direction */
else if $(\triangle x == 0$ && $\triangle y > 0)$ then
    Output_Dir = North
    /* destination is in North direction */
end if
```

The Y-first routing (YX) algorithm is a counterpart to the X-first (XY) routing and another deterministic routing scheme. The YX routing algorithm routes packets first in Y-dimension and then in X-dimension In order to avoid deadlocks, it forbids four 90-degree turns, i.e. West-South, West-North, East-South, and East-North turns.

The degree of adaptiveness (*DoA*) is defined by the number of shortest routes which the packets can take from source node (X_c, Y_c) to destination node (X_d, Y_d).

For the X-first routing algorithm, DoA is 1 because of no adaptivity. For fully adaptive routing method, it is given by

$$DoA_{algo} = \frac{(\triangle x + \triangle y)!}{\triangle x. \triangle y} \tag{2.1}$$

where, $\triangle x = (X_d - X_c)$ and $\triangle y = (Y_d - Y_c)$.

The X-first routing algorithm is deterministic in nature because it forbids more turns than required to achieve deadlock-freedom. Deadlocks can be prevented by prohibiting fewer than four turns. Moreover, it is required to prohibit only two turns. Thus, to increase adaptivity of XY turn model while maintaining deadlock-freedom, Glass and Ni [46] proposed three elegant routing algorithms namely West-first, North-last, and negative-first for the n-dimensional mesh network. They presented turn model for designing wormhole switching based partially adaptive routing algorithms for n-dimensional mesh and hypercube topology networks.

In order to avoid deadlocks, these adaptive routing algorithms forbid two turns, one turn from each of abstract cycle (clockwise and counter-clockwise). It should be noted that arbitrary selection of one turn for prohibition from each of the abstract cycle may lead to the formation of a cycle, thus resulting in deadlock configuration for wormhole switching. In wormhole switching, a packet can spread along the sequence of channels. Thus, this switching is more prone to deadlocks.

Figures 2.20 and 2.21 illustrate an example of incorrect selection of forbidden turns that leads to deadlock configuration. Figure 2.20(a) shows that the forbidden 90-degree turn (North-West) avoids the formation of cycle. However, the other petmitted 0-degree North-North turn (Figure 2.20(b)), 90-degree North-East, East-South and South-West turns (Figure 2.20(c)) and 0-degree West-West turn (Figure 2.20(d)) still allow the formation of cycle in the form of a knot. Thus, the three permitted 90-degree turns (Figure 2.21(a)) in anti-clockwise cycle are equivalent to the forbidden 90-degree turn (Figure 2.21(b)) in clockwise cycle. Similarly the three permitted 90-degree turns (Figure 2.21(b)) in clockwise cycle are equivalent to the forbidden 90-degree turn (Figure 2.21(a)) in anti-clockwise cycle. Thus, both knots still can result in deadlock configuration (Figure 2.21(c)).

Figure 2.22 shows the turn model for the adaptive West-first routing algorithm. In order to avoid deadlocks, it forbids two turns, i.e., South-West and North-West. The West-first routing prohibits all 90-degree turns towards West direction. If a

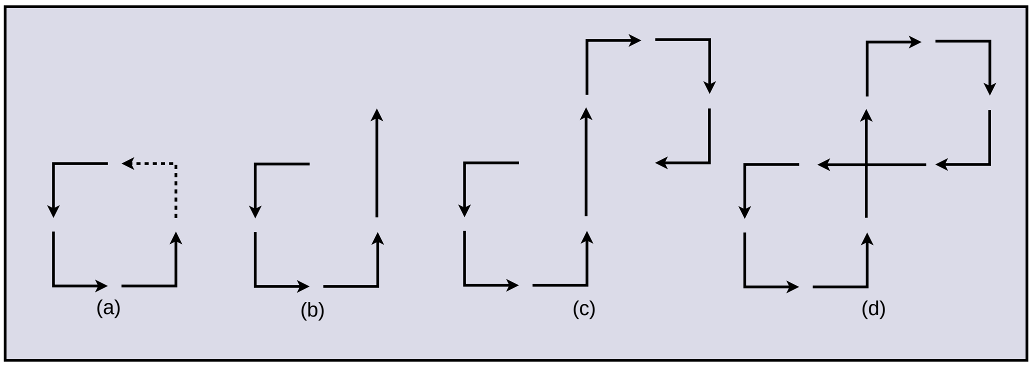


Figure 2.20: Six turns that form the abstract cycle resulting deadlock configuration

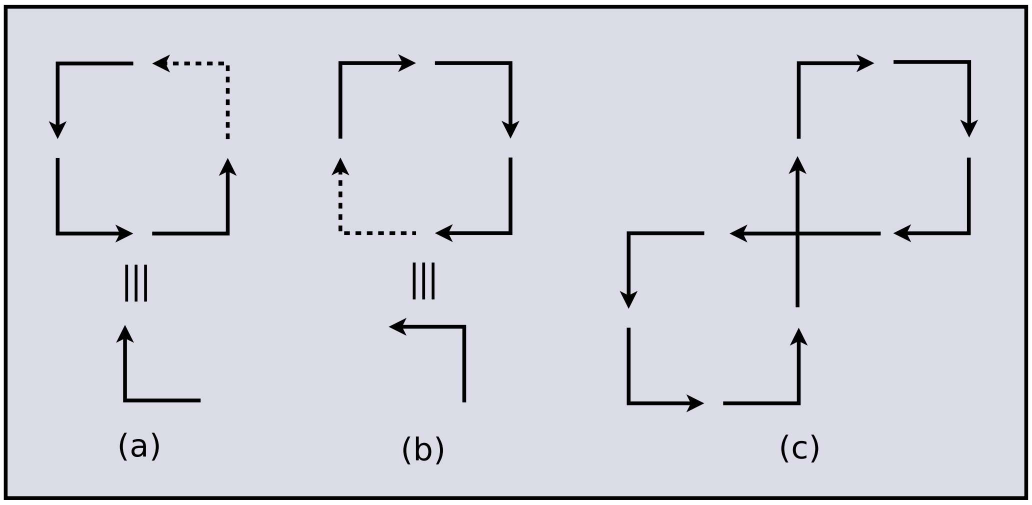


Figure 2.21: Six turns that form the abstract cycle (a) the three petmitted turns in anti-clockwise cycle are equivalent to the prohibited turn in clockwise cycle (b) the three petmitted turns in clockwise cycle are equivalent to the prohibited turn in anti-clockwise cycle (c) cycle formed by combining (a) and (b).

packet is required to travel West, it has to be first routed in the West only. Thus, the West-first routing is defined as: Route the packet first in the West direction, if required, and then it can be routed to other directions South, North and East adaptively. Algorithm 2 describes the routing function of the West-first routing algorithm.

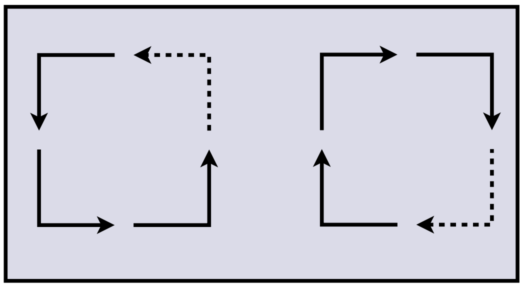


Figure 2.22: West-first turn model

Figure 2.23 shows the turn model for the adaptive negative-first routing algorithm. In order to avoid deadlocks, it forbids two turns, i.e., North-West and East-South. The negative-first routing prohibits all turns from positive directions (East and North) to negative directions (West and South). If a packet is required to travel to negative directions, it is first routed in the negative directions only. Thus, the negative-first routing is defined as: Route the packet first adaptively in negative directions if required, and then it can be routed to positive directions adaptively. Algorithm 3 describes the routing function of negative-first routing algorithm.

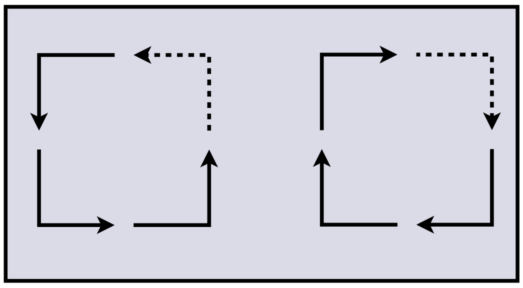


Figure 2.23: negative-first turn model

Figure 2.24 shows the turn model for the adaptive North-last routing algorithm. In order to avoid deadlocks, it forbids two turns, i.e., North-East and North-West. The North-last routing prohibits all turns from North direction. If a packet is required to travel North, it is routed in the North at the end. Thus, the North-last routing is defined as: Route the packet first adaptively in the South, East

Algorithm 2 Adaptive West-first Routing Algorithm

```
$(X_c, Y_c)$ : Current router coordinates
$(X_d, Y_d)$ : Destination router coordinates
$\triangle x = (X_d - X_c)$
$\triangle y = (Y_d - Y_c)$
if ($\triangle x == 0$ && $\triangle y == 0$) then
    Output_Dir = Local
    /* destination is local core */
else if ($\triangle x < 0$) then
    Output_Dir = West
    /* destination is in West direction */
else if ($\triangle x > 0$ && $\triangle y == 0$) then
    Output_Dir = East
    /* destination is in East direction */
else if ($\triangle x > 0$ && $\triangle y < 0$) then
    Output_Dir = Select(South, East)
    /* destination is in South-East quadrant */
else if ($\triangle x > 0$ && $\triangle y > 0$) then
    Output_Dir = Select(North, East)
    /* destination is in North-East quadrant */
else if ($\triangle x == 0$ && $\triangle y < 0$) then
    Output_Dir = South
    /* destination is in South direction */
else if ($\triangle x == 0$ && $\triangle y > 0$) then
    Output_Dir = North
    /* destination is in North direction */
end if
```

Algorithm 3 Adaptive negative-first Routing Algorithm

```
$(X_c, Y_c)$ : Current router coordinates
$(X_d, Y_d)$ : Destination router coordinates
$\triangle x = (X_d - X_c)$
$\triangle y = (Y_d - Y_c)$
if ($\triangle x == 0$ && $\triangle y == 0$) then
    Output_Dir = Local
    /* destination is local core */
else if ($\triangle x > 0$ && $\triangle y > 0$) then
    Output_Dir = Select(North, East)
    /* destination is in North-East quadrant */
else if ($\triangle x < 0$ && $\triangle y < 0$) then
    Output_Dir = Select(South, West)
    /* destination is in South-West quadrant */
else if ($\triangle x >= 0$ && $\triangle y < 0$) then
    Output_Dir = South
    /* destination is in South direction */
else if ($\triangle x < 0$ && $\triangle y >= 0$) then
    Output_Dir = West
    /* destination is in West direction */
else if ($\triangle x == 0$ && $\triangle y > 0$) then
    Output_Dir = North
    /* destination is in North direction */
else if ($\triangle x > 0$ && $\triangle y == 0$) then
    Output_Dir = East
    /* destination is in East direction */
end if
```

and West directions if required, and then it can be routed to North. Algorithm 4 describes the routing function of the North-last routing algorithm.

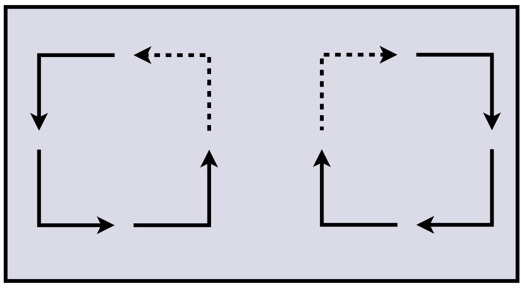


Figure 2.24: North-last turn model

Chiu [13] proposed one new Odd-Even (OE) turn model that has the higher degree of adaptiveness than Glass's turn model. Figure 2.25 shows the turn constraints for odd and even columns of a 2D mesh. In this model, North-West & South-West turns are not allowed at nodes located at odd columns, and East-South & East-North turns are not allowed on the nodes located at even columns.

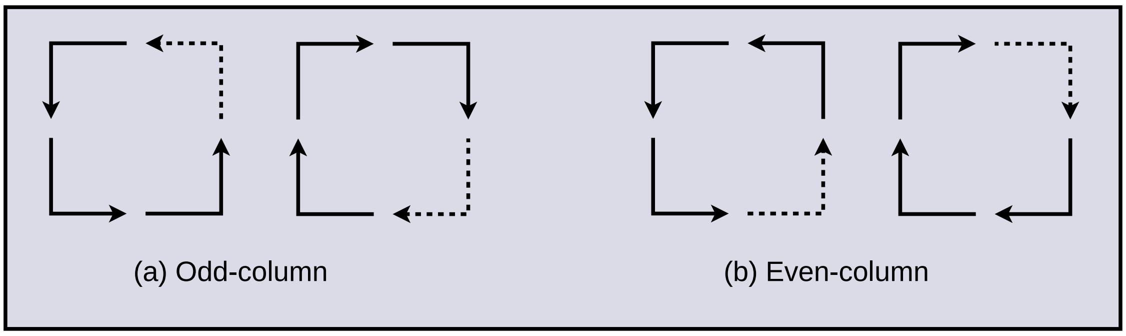


Figure 2.25: Turn model for odd-even routing (a) odd columns (b) even columns

2.6.2 Fully Adaptive Algorithms

In [16, 17], authors have introduced the concept of virtual channels in designing deterministic routing schemes to achieve deadlock-freedom. Generally, a physical channel is associated with one buffer. However, it can be multiplexed into a number of virtual channels. Various research proposal [12, 9, 23, 45, 69, 92, 95, 97, 60] have used virtual channels in developing fully and partially adaptive routing algorithms for different network topologies that include mesh based networks.

However, attaching virtual channels to a physical channel is less expensive than adding one full physical channel, but this solution is still expensive, particularly in the low area and power NoCs. The main area costs associated with virtual channels are the extra control lines to implement virtual channels and additional buffer space required for each virtual channel. In [13, 78], authors have shown that it is necessary to add virtual channels to design deadlock-free fully adaptive routing algorithms for mesh networks. Most of fully adaptive routing schemes depend upon the elimination of cycles in channel dependency graph to achieve deadlock-freedom.

Some routing strategies implement fully adaptive routing algorithms by dividing the network into virtual networks. A virtual network consists of a subset of virtual channels and is used to forward the packets for a particular set of destinations. Routing algorithm uses a particular virtual network depending upon the destination for routing packets. This section presents one fully adaptive routing scheme that uses the concept of virtual networks to inject packets.

Algorithm 4 Adaptive North-last Routing Algorithm

```
$(X_c, Y_c)$ : Current router coordinates
$(X_d, Y_d)$ : Destination router coordinates
$\triangle x = (X_d - X_c)$
$\triangle y = (Y_d - Y_c)$
if ($\triangle x == 0$ && $\triangle y == 0$) then
    Output_Dir = Local
    /* destination is local core */
else if ($\triangle x > 0$ && $\triangle y < 0$) then
    Output_Dir = Select(South, East)
    /* destination is in South-East quadrant */
else if ($\triangle x < 0$ && $\triangle y < 0$) then
    Output_Dir = Select(South, West)
    /* destination is in South-West quadrant */
else if ($\triangle x == 0$ && $\triangle y < 0$) then
    Output_Dir = South
    /* destination is in South direction */
else if ($\triangle x == 0$ && $\triangle y > 0$) then
    Output_Dir = North
    /* destination is in North direction */
else if ($\triangle x < 0$ && $\triangle y >= 0$) then
    Output_Dir = West
    /* destination is in West direction */
else if ($\triangle x > 0$ && $\triangle y >= 0$) then
    Output_Dir = East
    /* destination is in East direction */
end if
```

In [58], Jesshope *et al.* presented a fully adaptive routing methodology in order to

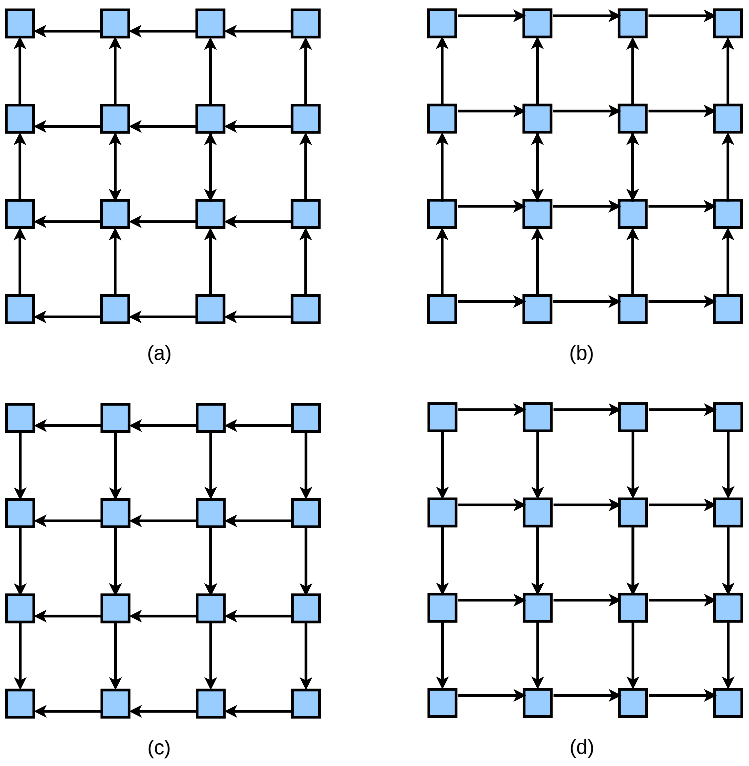


Figure 2.26: Example of virtual networks for 2D mesh (a) X-Y+ virtual network (b) X+Y+ virtual network (c) X-Y- virtual network (d) X+Y- virtual network

avoid deadlocks in 2D meshes. They split the network into a set of four disjoint virtual networks such that in each virtual network, the injected packets can travel only in one direction for each dimension. The four independent virtual networks of 4×4 2D mesh are shown in Figure 2.26. According to the destination positions, all packets are divided into four categories. These four categories represent four quadrants of a 2D mesh.

If a packet is inserted into the X- Y+ virtual network, it can travel only towards the negative direction (West) of X dimension and positive direction (North) of Y dimension. Similarly, if a packet is once routed into the X- Y- virtual network, it can travel along the negative direction (West) of X dimension and negative direction (South) of Y dimension, and so on. According to the destination of a packet, it is introduced to a particular virtual network. Once a packet is being routed in a given virtual network, it can travel along all the channels that corresponding to minimal routes. However, the packet cannot move from one virtual network to another. It can be observed that there does not exist any cyclic dependencies between channels. Thus, no deadlock configuration can be formed. The above methodology can be extended to the 3D mesh network as well.

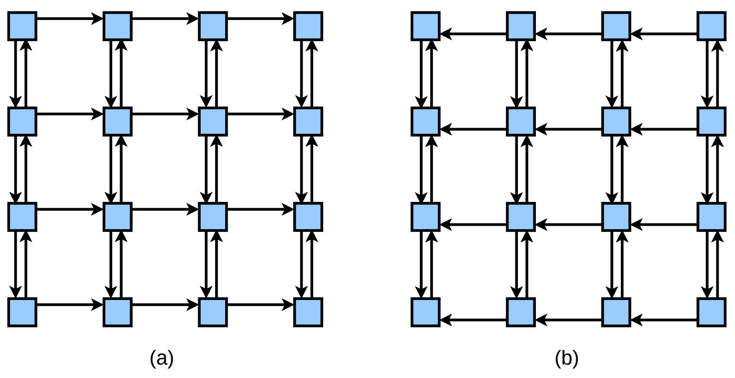


Figure 2.27: Virtual networks used by double-y routing for 2D mesh (a) eastward virtual network (b) westward virtual network

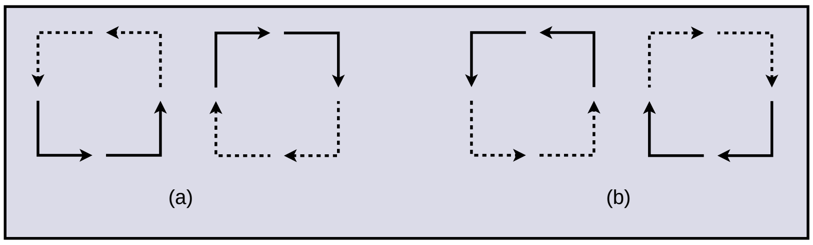


Figure 2.28: Turn model for double-y routing (a) eastward virtual network (b) westward virtual network

Linder and Harden [69] presented a fully adaptive and minimal routing scheme namely double-y which is an improvement over the scheme proposed in [58]. They proved that the number of virtual networks needed to avoid deadlocks could be reduced to half for n-dimensional meshes. For a 2D mesh, each virtual network requires two virtual channels along any (say X-dimension) of the two dimensions as opposed to the scheme [58]. The first virtual network consists of one and two virtual channels along X and Y dimensions, respectively. This virtual network is deployed to route packets towards the East direction. Thus, it is known as the eastward virtual network. Similarly, the second virtual network also consists of one and two virtual channels along X and Y dimensions, respectively. It is purposed for westward packets, thus known as the westward virtual network. When, a packet is destined for the destination in either same row or the same column, it can use any of the virtual networks. Figure 2.27 shows the two virtual networks used by double-y routing. Figure 2.28 shows the turn model representation of double-y routing algorithm that illustrates its turn constraints. All the packets follow minimal routes. Thus, the decrease in the number of virtual networks can not result in cyclic dependencies between channels. As discussed earlier in Section 2.5.1, a deadlock configuration is formed using a number of turns in the cyclic manner.

The 180-degree turns are also prohibited because of minimal routing restriction. Thus, with minimal routing, it is required that at least two dimensions must have channels in both the directions to form a cycle. Thus, each virtual network is cycle free and does not have cyclic dependencies between channels.

2.7 Recent Research Works and Issues

Advanced research problems in the area of future NoC include design of new architectures and frameworks to integrate a high performance communication technology (like NoC) with the multi-processor systems. The performance of on chip networks depends on many factors and the routing method is one of them.

In the recent past, many researchers have worked on the routing algorithms in NoCs domain. In [11], authors have proposed a congestion-aware routing method to relieve congestion in resource constraints systems. This algorithm considers switch and channel congestion. The method uses adaptive routing function along with selection strategy to get more precise congestion information. It adopts the path congestion information to make routing decisions which results in reduced packet latency along the routing path.

Hsien-Kai Hsin *et al.* [54] have presented a framework named network information region (NIR) for on-chip network systems. The NIR can indicate the network information which is used by the routing algorithms. Authors have presented a demonstration on how to apply the proposed framework on the analysis of the other adaptive routing algorithms. In addition, the paper also proposes the ant colony optimization based pheromone diffusion (ACO-PhD) adaptive routing framework based on the NIR. This framework specifies how NIR can help to integrate the temporal or spatial network information.

In [73], authors have proposed a routing algorithm to support workload consolidation. The workload consolidation means that it is likely that many applications will run parallel on a multi-core system with given the difficulty of extracting parallelism. Local adaptive methods do not consider enough status information to mitigate the congestion. At the same time, globally adaptive routing methods use network status beyond neighboring nodes. But, they may suffer from interference, coupling the behavior of otherwise independent applications. The proposed method combines local and global network information to gain effective adaptivity.

For on-chip components, the advanced technology in VLSI integration has increased the risk of failures. The failures can degrade the performance of the system and even can fail the entire system. Fault-tolerant routing is one of the way to provide reliability in on-chip communication for multi-core processor architectures. In [42], focus of the authors is on a particular class of routing schemes which do not use virtual channels to provide fault tolerance. In the design of any fault tolerant routing scheme, the major challenge is to get deadlock-freedom in the presence of faults. The proposed ZoneDefense routing not only includes faults into convex faulty blocks but also spreads the faulty blocks' position information in corresponding columns. Nodes having information about the faulty nodes create a defense zones. Thus, packets know in advance about the faulty node and can route themselves in other direction. Authors in [61] presented a fault tolerant routing method (FTR) based on XY routing algorithm. FTR algorithm is capable of handling single link failure. On the occurrence of fault, FTR reconfigures the paths to divert packets on other available paths. Authors in [101] proposed an adaptive fault-tolerant routing algorithm for mesh network. This method uses two VCs to achieve fault tolerance. Authors also presented a new planar network (PN) fault model which supports fault-tolerant routing in mesh for wormhole switching.

In [53], authors have proposed an ant colony optimization-based fault-aware routing (ACO-FAR) scheme that balances traffic load in the network. The proposed scheme uses three steps on the arrival of packet at the fault: 1) encounter; 2) search; and 3) select. Therefore, it implements three corresponding methods as: 1) notification of fault information; 2) route search method; and 3) route selection method. The router can check the available paths and route the packets along less-congested and fault-free path.

Letian Huang *et al.* in [56], have proposed a reconfigurable router architecture with bypassing channels that provides the cores connectivity in the network when the routers are under test. Authors have proposed adaptive routing algorithm TARRA to make the routers reconfigurable. The algorithm is capable of handling multiple routers which are under test. TARRA also uses one and two channels along the X and Y dimension, respectively. To prove the TARRA routing algorithm deadlock-free, authors have partitioned the network into two disjoint subnetworks. Each subnetwork has disjoint sets of channels.

With the advent of 3D integration technology, a number of research area have been opened [41]. The issues related 3D integration includes from Through-Silicon-Vias

(TSV) technology to high-level system configuration and design. The network designers are not only facing challenges related to emerging 3D technology, they need to re-looked many issues which were already resolved for 2D planes. The traditional 2D integration technology has limited floor-planning options, and consequently, it constraints the performance enhancements arising out of NoC architectures [37]. 3D ICs are capable to result in better functionality, performance and packaging density compared to conventional 2D ICs.

In [27], a distributed routing scheme is presented for vertically partially connected regular 2D topologies of different shapes and sizes (e.g. 2D mesh, torus, ring). These topologies are of practical interest in the 3D integration of heterogeneous dies using TSVs. A 3D NoC struture requires a number of vertical links while 3D Integration has a major limitation on the number of vertical interconnects (TSVs) to be exploited. In this research, focus of authors is to consider nonregular 3D topologies in which the usual planar topologies are partially connected together by only some vertical links. This reduces the number of vertical links. Authors have also implemented regular topology for the fair comparison with other methods.

In [90], authors have designed a routing method for partially connected 3D-NoCs. The routing scheme is adaptive and is able to tolerate the failures on vertical channels as compared to the pre-designed routing schemes. In this thesis, we are motivated to improve the performance of previously proposed routing methods by increasing the adaptivity and providing fault tolerance in 2D and 3D mesh architectures.

Chapter 3

2D-CHARM: Congestion-aware and Highly Adaptive Routing Method for 2D-Mesh

3.1 Overview

The requirement of acyclic channel dependency graph (CDG) for deadlock avoidance in Network-on-Chip (NoC) routing imposes unnecessary restrictions on routing turns, thus reduces the degree of adaptiveness. The degree of adaptiveness has a major impact on the performance of an adaptive routing method. In addition, inappropriate selection of output channel at the router may result into network "hot spots" which may lead to congestion. Network congestion may lead to increased power consumption and communication delay, thus degrades the performance of on-chip networks. However, the performance of NoC can be improved by routing packets along the non-minimal path through less congested areas and distributing the traffic load across the network. Thus, in this chapter, we present a novel highly adaptive and congestion-aware routing method (CHARM) to address issues mentioned above. Since, the proposed algorithm applies to the 2D mesh network, we have named it 2D-CHARM. The proposed method is equally applicable to 3D mesh network as well. In the next chapter, we present 3D-CHARM, an extension to 2D turn model proposed in this chapter.

Congestion control is a critical issue in on-chip networks as it affects overall network performance. Congestion can be alleviated with balanced traffic load across various links. Adaptiveness and non-minimality of a routing algorithm are used to balance traffic load while maintaining deadlock freedom. Turn model produces partially adaptive routing [46, 13, 55, 102, 43] schemes that restrict some routing turns to achieve deadlock freedom thus resulting in low adaptiveness. Deadlock free fully adaptive routing schemes require additional virtual channels (Section 2.3) to achieve the high degree of adaptiveness. Several minimal/non-minimal fully adaptive routing algorithms have been proposed in [69, 12, 45, 68, 47, 86, 72, 33, 34] using additional virtual channels.

Ming Li *et al.* [68] proposed congestion-aware and dynamic routing algorithm DyXY that determines output channel using congestion status of input buffers of next hop routers. CATRA [33], DAR [86], RCA [47] and DBAR [72] are congestion-aware routing schemes that use local and non-local congestion information to route the packets using extra hardware.

Minimal routing algorithms suffer from the low degree of adaptiveness, which is not sufficient in distributing the network traffic, even if they accurately detect the status of congestion. Ebrahimi *et al.* [34] proposed non-minimal routing scheme for 2D mesh. It provides better adaptiveness than [45] with the same number of virtual channels. However, it imposes some unnecessary restrictions on routing turns, which can be removed to increase path diversity.

The proposed algorithm permits cyclic dependencies in channel dependency graph providing a higher degree of adaptiveness. The algorithm uses congestion-aware channel selection policy that results into the balanced distribution of traffic load across the network. A packet, once injected in NoC, follows non-minimal paths only when minimal paths are congested at the neighboring routers.

3.1.1 Motivation and Background

The motivation of the proposed routing algorithm is derived from the fact that a less restrictive routing algorithm offers a high degree of adaptiveness [46]. Most of routing algorithms proposed in the recent literature [70, 36, 68, 71, 45, 34, 32, 33] achieve deadlock-freedom by forcibly restricting certain routing turns so that the CDG remains acyclic. This acyclic CDG requirement for the deadlock-freedom makes these algorithms more restrictive, thus reduces the degree of adaptiveness.

If we relax this requirement for these algorithms, we may acquire a less restrictive routing algorithm. The main focus of this research is to relax this requirement by allowing cycles in the CDG provided that extended channel dependency graph (ECDG) is acyclic (using Duato's theorem [24]). The ultimate aim of the proposed work is to add more functionality (routing turns) to virtual channels of existing algorithms to achieve high degree of adaptiveness. In our proposed method, we have used double-y network. The double-y network uses one physical channel in each dimension. However, in the Y dimension, the physical channel is logically divided into two virtual channels, thus the network is called double-y network.

We have explained our point by comparing with two recent algorithms LEAR [34] and HARA [32]. It should be noted that LEAR and HARA routing algorithms are based on Mad-y [45] turn model.

In short, the main contribution of this work is a novel highly adaptive routing algorithms with less routing restrictions. We have shown that the proposed routing algorithm results in significant performance improvement compared to existing routing schemes under certain traffic patterns designed for NoCs.

3.1.2 Mad-y Turn Model

Turn model representation is an effective way to describe routing algorithm and its restrictions. Figure 3.1 shows a turn model representation of Mad-y routing algorithm. It imposes following constraints on routing turns in order to avoid deadlocks:

1. It prohibits four 90-degree turns (E-$N1$, E-$S1$, $N2$-W, and $S2$-W) as shown in Figures 3.1(i) and 3.1(ii).
2. It prohibits two 0-degree turns ($S2$-$S1$ and $N2$-$N1$) as shown in Figure 3.1(iii).
3. It prohibits all 180-degree turns as it is a minimal routing algorithm.

We can deduce following restrictions from above constraints.

1. A packet is allowed to take 90-degree turns ($N1$-E and $S1$-E) only when it has not already routed to the East. These routing turns are restricted only because a packet cannot use $N1$ or $S1$ after using the East channel.

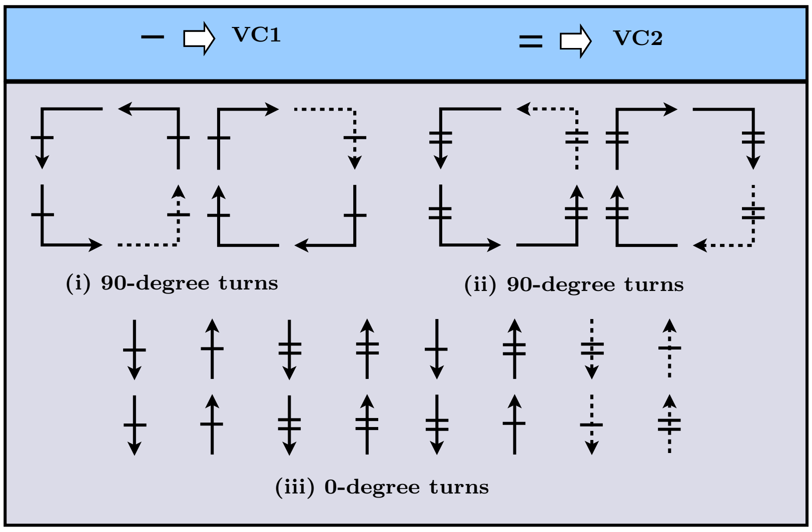


Figure 3.1: Mad-y turn model (permitted (prohibited) turns are represented by solid (dash) lines)

2. A packet is allowed to take 90-degree turns (W-$N2$ and W-$S2$) only when it does not need to take the West turn further. These routing turns are restricted only because a packet cannot use West channel after using $N2$ or $S2$.

3. A packet is allowed to take 0-degree turns ($N1$-$N2$ and $S1$-$S2$) only when it does not need to take the West turn further. These routing turns are restricted only because a packet cannot take the West turn after using $N2$ or $S2$.

4. A packet is allowed to take 0-degree turns ($N1$-$N2$ and $S1$-$S2$) only when it has not already routed to the East. These routing turns are restricted only because a packet cannot use $N1$ or $S1$ after using the East channel.

5. If a packet needs to route West, it cannot use $N2$ or $S2$ at the source node.

Mad-y routing algorithm is proved deadlock-free on the basis of work of Dally and Seitz [18], who show that a routing algorithm is deadlock free if channels of network can be assigned numbers such that the algorithm routes each packet along channels with strictly increasing (or decreasing) numbers. A two-digit number (a, b) in base-r is assigned to each output channel of a router in $n \times m$ 2D mesh network as shown in Figure 3.2, where r is greater of $2m$ and n. In the Mad-y scheme, every packet is routed along channels with strictly increasing numbers. It can be easily observed from Figure 3.3, for each input channel into an arbitrary node, the scheme routes the packet only along a channel with a higher number.

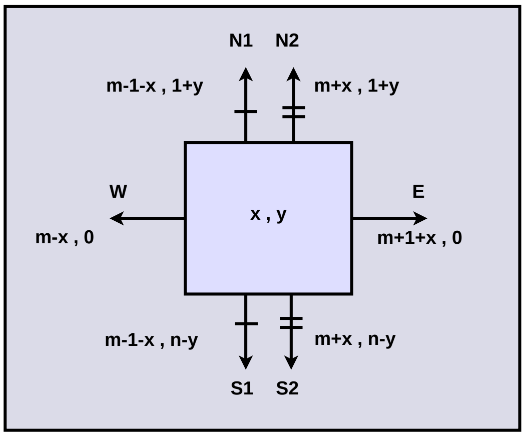


Figure 3.2: Numbering of the output channels leaving each router (x, y) of $n \times m$ mesh for Mad-y algorithm

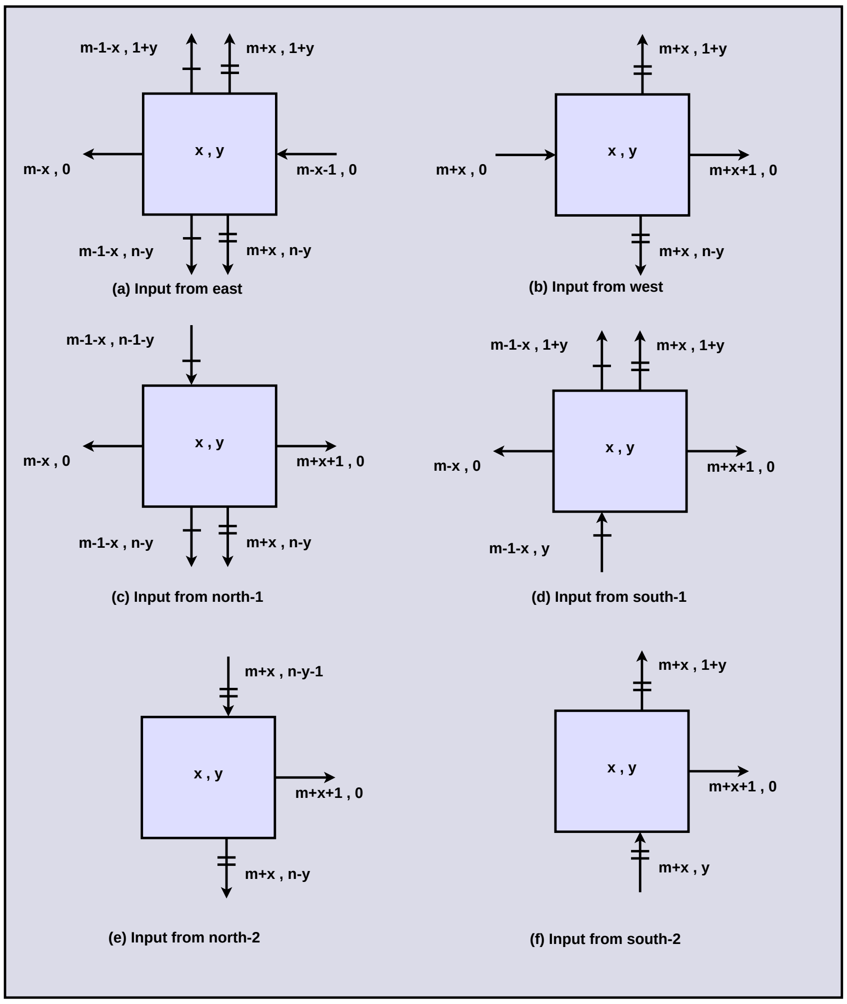


Figure 3.3: For each input channel, the eligible output channels of the Mad-y routing scheme have higher numbers

Figure 3.4 shows a node $(2,1)$ in a 4×4 mesh. If the packet is coming on East input channel (numbered $1,0$), then it can be routed along any output channel including East. Because all output channels have higher number than the input. However, if the packet is coming on west input channel (numbered $6,0$), then it can be routed only along higher numbered output channels that are North-2 $(6,2)$, South-2 $(6,3)$ and East $(7,0)$ output channels.

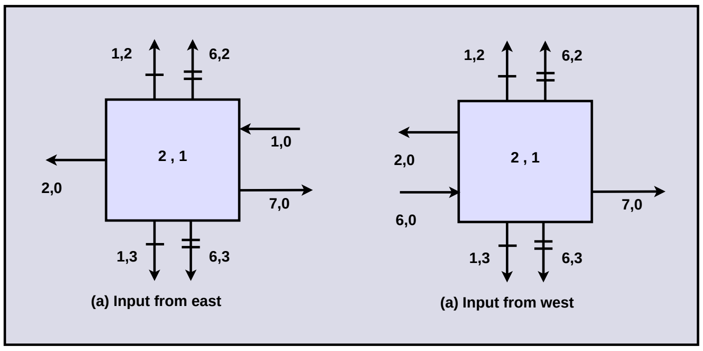


Figure 3.4: Numbering assignment of a node $(2,1)$ in a 4×4 mesh for Mad-y routing scheme

3.1.3 LEAR and HARAQ Turn Models

The LEAR [34] turn model has the same turn constraints as Mad-y for 90-degree and 0-degree turns (Figures 3.5(i), 3.5(ii) and 3.5(iii)). In addition, the LEAR model allows some 180-degree turns as shown in Figure 3.5(iv), whereas the Mad-y prohibits all 180-degree turns. In HARAQ [32, 30], authors have improved LEAR turn model by allowing a few more 180-degree turns (which were prohibited in LEAR) as shown in Figure 3.6(iv). The deadlock-freedom of these algorithms is also proved using Dally's work [18].

Turn model based routing algorithms can work for any system size. These algorithms [45, 32] are proved deadlock free for $m \times n$, where m and n are greater than 2, thus scalable. In principle, turn models define the routing restrictions for all routing turns within a network irrespective of system size. Thus, any algorithm which is based on the turn model remains scalable.

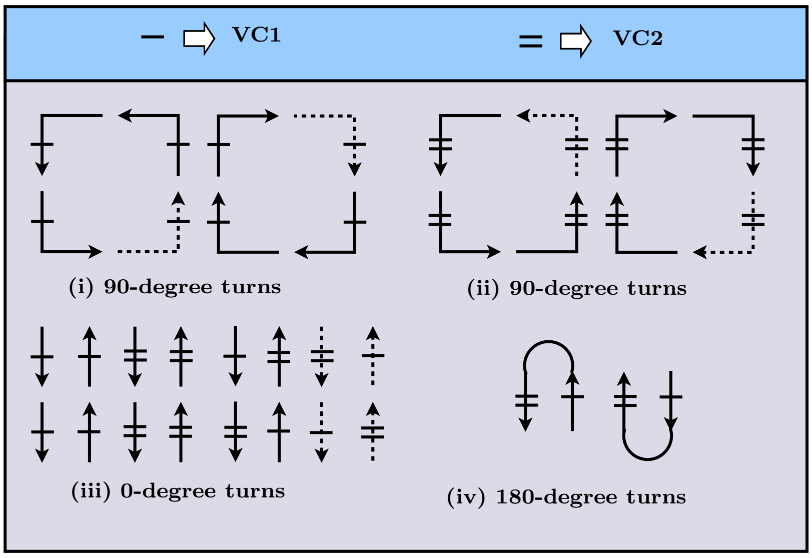


Figure 3.5: LEAR turn model (permitted (prohibited) turns are represented by solid (dash) lines)

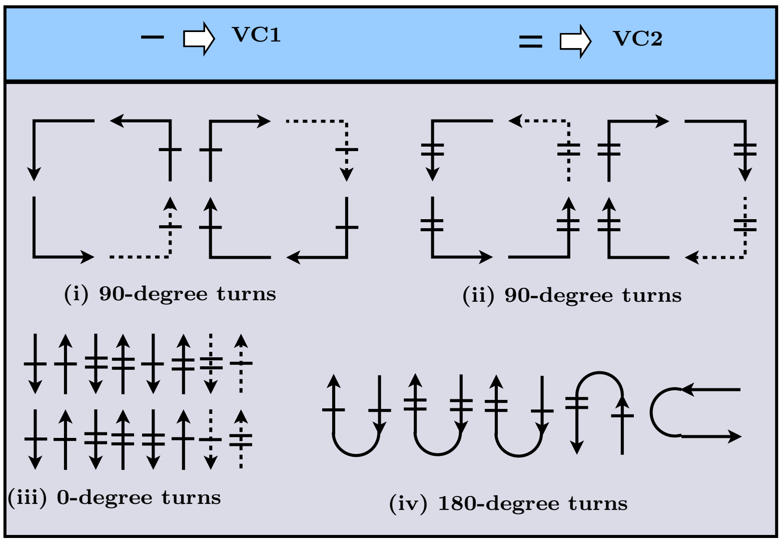


Figure 3.6: HARAQ turn model (permitted (prohibited) turns are represented by solid (dash) lines)

3.2 Proposed Work

In this section, we present our novel turn model (2D-CHARM), which is a *major* improvement in existing 2D turn models used by several routing algorithms [70, 36, 68, 71, 34, 32] including Mad-y [45]. Mad-y algorithm is the base of all mentioned algorithms [70, 36, 68, 71, 34, 32]. We have also presented routing method on the basis of proposed turn model. We have also extended our proposed 2D turn model for the 3D mesh in the next chapter, which can be further extended for n-dimensional mesh as well.

3.2.1 2D-CHARM: Turn Model

An acyclic channel dependency graph requirement to avoid deadlocks places unnecessary, thus avoidable restrictions on the routing turns in a routing algorithm. Other related work (Mad-y, LEAR and HARAQ routing methods) are proved deadlock-free using acyclic channel dependency graph [18]. Thus, their routing functions cannot use all qualified turns to forward packets through less congested areas. The proposed method imposes substantially fewer restrictions on routing turns (especially on 90-degree) using [24], thus it provides additional minimal and non-minimal paths between source and destination than Mad-y, LEAR and HARAQ.

Figure 3.7 shows the turn model representation of 2D-CHARM. For minimal routing, a packet is permitted to use the first VC ($N1$ and $S1$) at any time, as shown in Figure 3.7(i). It can use the second VC ($N2$ or $S2$) only if it has already been routed to negative direction of X dimension (West), as shown in Figure 3.7(ii).

In short, in order to avoid deadlocks, 2D-CHARM imposes the following constraints on routing turns:

1. It prohibits only two 90-degree turns ($S2$-W and $N2$-W).
2. It allows all 0-degree turns as shown in Figure 3.7(iii). However, It allows 0-degree turns ($S1$-$S2$, $N1$-$N2$, $S2$-$S1$ and $N2$-$N1$) only when a packet does not need to be forwarded further West.
3. It permits a few 180-degree turns, as shown in Figure 3.7(iv).

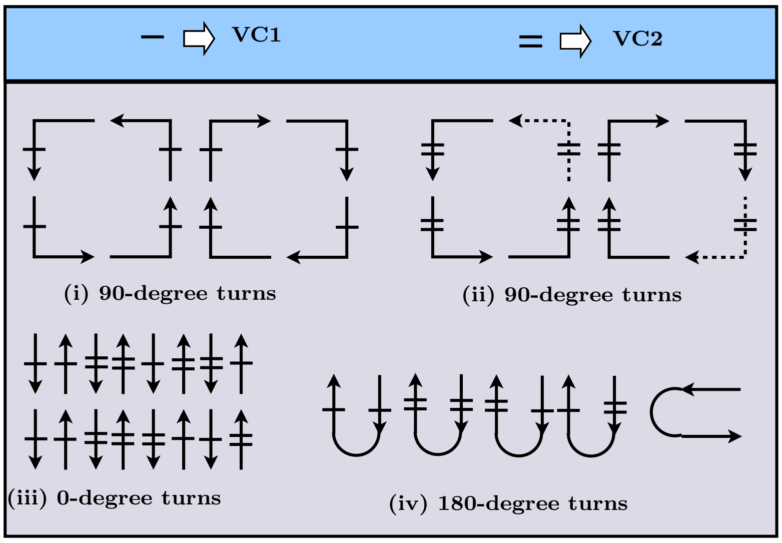


Figure 3.7: 2D-CHARM turn model (permitted (prohibited) turns are represented by solid (dash) lines)

We can deduce following restrictions from above constraints:

1. A packet is allowed to take 90-degree turns (W-$S2$ and W-$N2$) only when it does not need to take the West turn further. This restriction is because of prohibited 90-degree turns ($N2$-W and $S2$-W).
2. A packet cannot use $N2$ or $S2$ at the source node if it needs to be forwarded West.

Table 3.1 shows prohibited routing turns for 2D-CHARM and other related work. In contrast to other related work [45, 34, 32], 2D-CHARM forbids only two 90-degree turns. 2D-CHARM permits all 0-degree turns. However, It should be noted that some 0-degree turns ($N2$-$N1$, $S2$-$S1$, $N1$-$N2$ and $S1$-$S2$) are permitted only if the destination is not in the West. The packet is allowed to take one 180-degree turn in each dimension X and Y (West to East and South to North), as shown in Figure 3.7(iv), but only if it has completed routing in the West and South directions, respectively.

By comparing the prohibited routing turns as shown in Table 3.1, we can conclude that 2D-CHARM results in a larger set of output channels due to high degree of adaptiveness than other related work.

Table 3.1: Prohibited routing turns for different routing algorithms

		Routing Algorithms			
		Mad-y [45]	**LEAR** [34]	**HARAQ** [32]	**Proposed (2D-CHARM)**
Turns	**90-degree**	E-N1, E-S1, S2-W, N2-W	E-N1, E-S1, S2-W, N2-W	E-N1, E-S1, S2-W, N2-W	S2-W, N2-W
	0-degree	N2-N1, S2-S1	N2-N1, S2-S1	N2-N1, S2-S1	-
	180-degree	ALL	ALL except N1-S2, S1-N2	ALL except W-E, N1-S2, S1-N2, S1-N1, S2-N2	ALL except W-E, S1-N1, S1-N2, S2-N1, S2-N2

3.2.2 2D-CHARM: Routing Algorithm

The functionality of 2D-CHARM routing algorithm is divided into two phases: route computation phase and output channel selection phase. On the basis input channel (on which packet has arrived) and relative position of the destination node with respect to the current node, routing function computes a set of output channels using turn model explanation discussed above. The route computation function (*rfun*) of 2D-CHARM is described in the Table 3.2. The highlighted entries are part of 2D-CHARM and discussed later in Theorem 3. *rfun* produces output-channel vector for a packet using the packet's destination position (*des_pos*) and packet's input channel (*in_ch*).

Selection function selects one output channel from the set of output channels provided by the route computation function. Our selection function first checks all qualified output channels corresponding to shortest routes and forwards the packet to the output channel in which the corresponding next hop node has its congestion status flag set to zero (and possibly, corresponding to *non-escape channels*). If the congestion status flags of all next hop nodes on shortest routes are set to one, the congestion status flag of each eligible non-minimal route is inspected. If there exist such non-minimal routes, which are not congested, our method selects one of the output channels to forward the packet (and possibly, corresponding to *non-escape channels*). Our method prefers adaptive output channels over escape channels[1] because it results in increased probability of escape output channels being available when they are required to avoid deadlocks.

Figure 3.8 shows an example of the 2D-CHARM routing algorithm for a 6×6 mesh network. A source node 7 sends a packet to destination node 22. 2D-CHARM routing function provides all six channels (E, $N1$, $N2$, W, $S1$ and $S2$) as output at node 7. All neighboring nodes on minimal node in paths (E, $N1$, $N2$) are in congestion region. If West channel is selected, packet has five choices ($N1$, $N2$, $S1$, $S2$ and W) on next node 6. Since node 6 is on the West border, the West cannot be selected as output direction. Now at node 6, packet has four choices ($N1$, $N2$, $S1$ and $S2$). 2D-CHARM selection function will give preference to non-congested minimal paths, it selects North direction at node 6. The same strategy is followed till the packet is received at the destination. Specially at node 33 (or 28), 2D-CHARM routing function provides both $S1$ and $S2$ as output directions.

[1]output channels which are used to escape from deadlocks

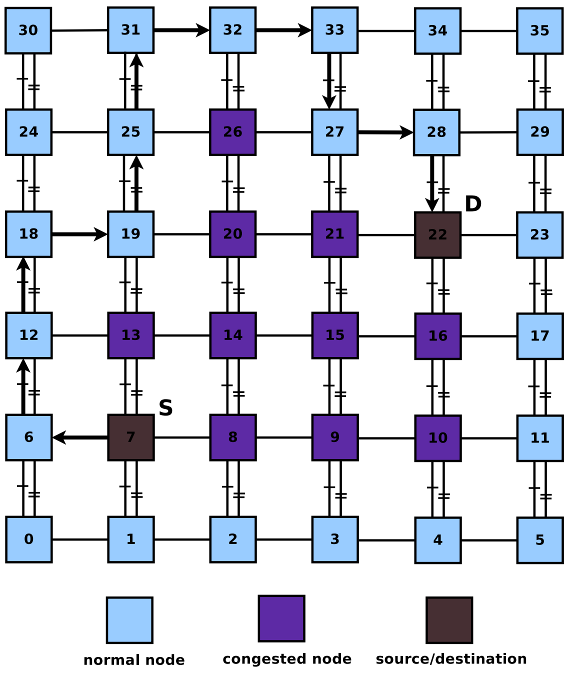


Figure 3.8: Example of 2D-CHARM method

Whereas other methods provide only E-$S2$ (Figures 3.1, 3.5 and 3.6) as E-$S1$ is prohibited. This example shows 2D-CHARM method is capable of routing packets around the congested region and distributing the traffic across the network.

3.2.3 Deadlock Freedom of 2D-CHARM

With deterministic routing, packets can be routed over single output channel at each node. Thus, it is mandatory to remove all cyclic dependencies between network channels in order to achieve deadlock freedom. In adaptive routing, packets often have several options for routing at each node. Thus, it is not mandatory to eliminate all cyclic dependencies between channels, provided that every packet can be forwarded on a route whose channels are not involved in cyclic dependencies. The channels involved in these acyclic routes are considered as *escape channels* from deadlocks (cycles).

The deadlock-freedom of 2D-CHARM is assured by using Duato's theory [24] stated as follows:

Theorem 1. (Duato's Theorem) *For an interconnection network I, a connected and adaptive routing function R is deadlock-free if there exists a routing subfunction $R_1 \subseteq R$, that is connected and has an acyclic ECDG (with no cycles because*

Table 3.2: Route computation function (*rfun*) of 2D-CHARM

Input channel (*in_ch*)	**Destination position (*des_pos*)**							
	S	**N**	**E**	**W**	**SE**	**SW**	**NE**	**NW**
S1	-	N1, N2	E, N1, N2	W	E, N1, N2	-	N1, N2, E	N1, W
N1	S1, S2	N1, N2, S1, S2	E, S1, S2, N1, N2	W	S1, S2, E, N1, N2	S1, W	N1, N2, E, S1, S2	-
S2	-	N1, N2	E, N1, N2	-	E, N1, N2	-	N1, N2, E	-
N2	S1, S2	N1, N2, S1, S2	E, S1, S2, N1, N2	-	S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
E	S1, S2, W	N1, N2, W, S1, S2	E, N1, N2, S1, S2, W	W	S1, S2, E, N1, N2, W	S1 ,W	N1, N2, E, S1, S2, W	N1 ,W
W	S1, S2	N1, N2, S1, S2	E, N1, N2, S1, S2	-	S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
L	S1, S2	N1, N2, S1, S2	E, N1, N2, S1, S2	W	S1, S2, E, N1, N2	S1 ,W	N1, N2, E, S1, S2	N1 ,W

of direct, indirect, direct-cross and indirect-cross dependencies).

Following Duato's terminology, we represent the *rfun* of 2D-CHARM by R. C represents the channel set used by R and contains all VCs ($N1$, $S1$, $N2$, $S2$, E and W). To assure the deadlock-freedom of 2D-CHARM, we first identify the subset of channels $C_1 \subseteq C$ (escape channels), that defines routing subfunction $R_1 \subseteq R$. For 2D-CHARM, this subset C_1 contains channels $N2$, $S2$, E and W. Table 3.3 describes the *rfun* for the routing subfunction R_1 of 2D-CHARM.

Lemma 1. *The routing subfunction R_1 is connected and cycle-free (deadlock-free).*

Proof. The *rfun* for routing subfunction R_1 (Table 3.3) with channel set C_1 is non-minimal version of West-first routing [46]. Since non-minimal West-first routing is connected and cycle-free, so R_1 is connected and deadlock-free. □

Lemma 2. *The ECDG of channel set C_1 does not have any cycle because of direct-cross and indirect-cross dependencies.*

Proof. We can observe from Tables 3.2 and 3.3, routing subfunction R_1 is defined using a channel subset C_1, according to the following expression:

$$R_1(in\_ch, des\_pos) = R(in\_ch, des\_pos) \cap C_1, \quad \forall in\_ch, des\_pos \tag{3.1}$$

A channel belonging to C_1 is used as an escape channel for all the destinations for which it can be supplied by R. It means whenever routing function R (Table 3.2) provides a channel from set C_1 ($N2$, $S2$, E and W) for a particular destination, that channel is also provided by the routing function R_1 (Table 3.3) for that destination. The cross dependencies may exist if we add any routing option between channels of C_1 while developing routing function R from R_1 by adding channels $N1$ and $S1$. We have not added any routing option between channels of C_1. Thus, there does not exist any cross-dependency between channels in C_1 meaning ECDG is cycle-free because of cross dependencies. □

Lemma 3. *The ECDG of channel set C_1 does not have any cycle because of direct and indirect dependencies.*

Proof. From Lemma 1, R_1 is proved cycle-free, thus no direct dependency can cause cycles in ECDG. Since, the R_1 is West-first routing algorithm, a West channel

Table 3.3: Route computation function for routing subfunction R_1

		des_pos							
		S	**N**	**E**	**W**	**SE**	**SW**	**NE**	**NW**
in_ch	**S2**	-	N2	E, N2	-	E, N2	-	N2, E	-
	N2	S2	N2, S2	E, S2, N2	-	S2, E, N2	-	N2, E, S2	-
	E	S2, W	N2, W, S2	E, N2, S2, W	W	S2, E, N2, W	W	N2, E, S2, W	W
	W	S2	N2, S2	E, N2, S2	-	S2, E, N2	-	N2, E, S2	-
	L	S2	N2, S2	E, N2, S2	W	S2, E, N2	W	N2, E, S2	W

is always utilized before any other channel (South or North or East) in C_1. Thus, the dependencies towards West channel from any other channel (South or North or East) are absent. However, the additional channels ($N1$ and $S1$) introduced by R can cause indirect dependencies between West channels as a packet can use West channel, then any addition channel ($N1$ or $S1$) and later can use West channel of different row as shown in Figure 3.9. But, this indirect dependency does not introduce any cycle in ECDG. Because to form a cycle, at least one of the 90-degree turns ($S2$-W or $N2$-W) must be allowed, however, these 90-degree turns are prohibited. Thus, these indirect dependencies introduce new dependencies between only the West virtual channels and do not result in cycles. Since there are no direct and indirect dependencies that produce cycles in ECDG of C_1. Therefore, ECDG of C_1 is acyclic because of direct and indirect dependencies. □

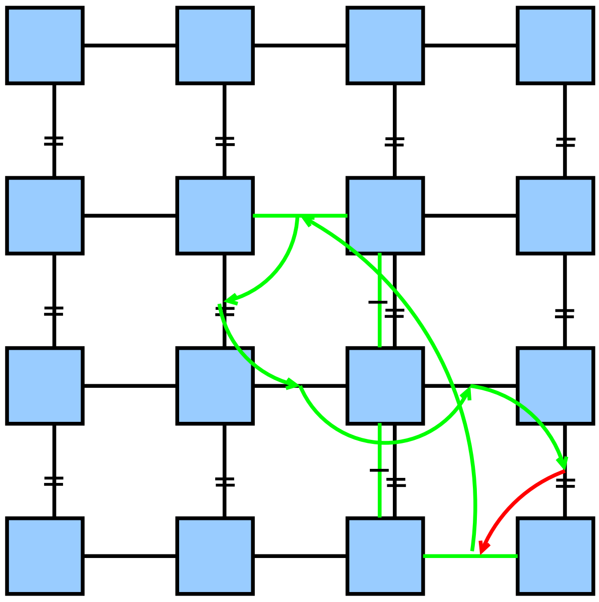


Figure 3.9: Example of indirect dependency in ECDG because of additional channels ($N1$ and/or $S1$)

Theorem 2. *The proposed routing algorithm is deadlock-free.*

Proof. We can conclude from Lemmas (1, 2 and 3) and using Theorem 1 that the proposed routing algorithm is deadlock-free. □

3.2.4 Livelock Freedom of 2D-CHARM

Since, the non-minimal routing algorithms are suspicious to livelock, we have proved the livelock-freedom for the 2D-CHARM using following theorem.

Theorem 3. *The proposed routing algorithm is livelock-free.*

Proof. The key idea behind livelock-freedom of 2D-CHARM is that in each dimension, only one 180-degree turn is allowed. From Lemma 1, It is proved that R_1 routing is non-minimal version of West-first routing [46], thus it is livelock-free.

We design a new routing function (R_2) by splitting the North VC ($N2$) into two VCs ($N1$ and $N2$) and South VC ($S2$) into $S1$ and $S2$. We impose same routing constraints on both newly added VCs $N1$ and $N2$ as of old $N2$. Similarly, newly added VCs $S1$ and $S2$ are also having same routing constraints as of old $S2$. Table 3.4 describes the routing restriction of the new routing function R_2. We can observe that R_2 is also non-minimal West-first routing algorithm. The only difference is that R_2 uses the double-y network, whereas R_1 uses single VC in each dimension. Thus, we can conclude that the R_2 is also livelock-free.

It is well known that minimal routing never causes livelock and if we add minimal paths to the R_2, it will remain livelock-free. We design a new routing function (R_3) as shown in Table 3.5 by adding some new minimal routing options to R_2. We have shown aforementioned newly added minimal routing options as highlighted entries in the Table 3.5. These newly added entries are corresponding to minimal paths and never cause livelock. Thus, we can conclude that the R_3 is also livelock-free.

It can be observed that Table 3.5 is same as Table 3.2. Thus, we can conclude that 2D-CHARM also is livelock-free. □

3.3 Results Analysis

We have evaluated proposed routing method (CHARM) with real and synthetic traffic profiles. To evaluate the effectiveness of CHARM, we have implemented two versions of CHARM: 1) CHARM, and 2) CHARM-XY. CHARM is implemented for double-y network. We have implemented a few other well-known routing methods to compare with CHARM. These methods include the dimension order routing (XY), Mad-y [45], LEAR [34] and HARA [32]. CHARM-XY is implemented for double-xy network and restricts all 180-degree turns which make the algorithm suitable to compare with other similar approaches JJMM [62], PCAR [11] and DBSS [73]. The double-xy network has 2 VCs in each of X and Y dimensions.

Table 3.4: *rfun* for the new routing function R_2 derived from R_1 by splitting each of N2 and S2 VC of R_1 into two VCs (N1, N2, S1, S2)

		des_pos							
		S	**N**	**E**	**W**	**SE**	**SW**	**NE**	**NW**
in_ch	**S1**	-	N1, N2	E, N1, N2	-	E, N1, N2	-	N1, N2, E	-
	N1	S1, S2	N1, N2, S1, S2	E, S1, S2, N1, N2		S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
	S2	-	N1, N2	E, N1, N2	-	E, N1, N2	-	N1, N2, E	-
	N2	S1, S2	N1, N2, S1, S2	E, S1, S2, N1, N2	-	S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
	E	S1, S2, W	N1, N2, W, S1, S2	E, N1, N2, S1, S2, W	W	S1, S2, E, N1, N2, W	W	N1, N2, E, S1, S2, W	W
	W	S1, S2	N1, N2, S1, S2	E, N1, N2, S1, S2	-	S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
	L	S1, S2	N1, N2, S1, S2	E, N1, N2, S1, S2	W	S1, S2, E, N1, N2	W	N1, N2, E, S1, S2	W

Table 3.5: *rfun* for the new routing function R_3 after adding a few entries (highlighted) in R_2 corresponding to minimal paths for the destination toward West (W, SW and NW)

		des_pos							
		S	**N**	**E**	**W**	**SE**	**SW**	**NE**	**NW**
in_ch	**S1**	-	N1, N2	E, N1, N2	W	E, N1, N2	-	N1, N2, E	N1,W
	N1	S1, S2	N1, N2, S1, S2	E, S1, S2, N1, N2	W	S1, S2, E, N1, N2	S1,W	N1, N2, E, S1, S2	-
	S2	-	N1, N2	E, N1, N2	-	E, N1, N2	-	N1, N2, E	-
	N2	S1, S2	N1, N2, S1, S2	E, S1, S2, N1, N2	-	S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
	E	S1, S2, W	N1, N2, W, S1, S2	E, N1, N2, S1, S2, W	W	S1, S2, E, N1, N2, W	S1,W	N1, N2, E, S1, S2, W	N1,W
	W	S1, S2	N1, N2, S1, S2	E, N1, N2, S1, S2	-	S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
	L	S1, S2	N1, N2, S1, S2	E, N1, N2, S1, S2	W	S1, S2, E, N1, N2	S1,W	N1, N2, E, S1, S2	N1,W

We evaluate CHARM and all other routing methods on both synthetic and realistic traffic profiles. As synthetic traffic profiles, we consider both uniform (*random*) and non-uniform (*hotspot* and *transpose*) traffic patterns. For realistic communication profiles, we consider traces of some application suites extracted from benchmark suite E3S [22]. In order to perform all required set of experiments involving various routing methods and traffic profiles, we modified and extended [57], a cycle-accurate and open source SystemC based NoC simulator.

For all experiments, we consider 7×7 mesh. Packet size and input-channel buffer size for each virtual channel is kept constant for all experiments and set to 8 and 6 flits, respectively. The simulator is warmed up for 10000 cycles and afterward; the average performance is measured over another 100000 cycles. Congestion threshold is set to 66% of total buffer size meaning that when four slots of input channel buffers are occupied. Congestion threshold specifies whether the next hop router is congested or not. For the synthetic traffic, traffic generator generates traffic at constant bit rate. Traffic load of 10% implies that packet injection rate is 10% of channel bandwidth. As communication performance parameter, we consider latency (delay) and throughput. The latency is defined as the time difference (in clock cycles) between header flit injection from source router and tail flit reception at the destination router. Throughput is defined as the fraction of the maximum load that the network can physically handle.

3.3.1 Uniform Traffic Pattern

Under uniform (random) traffic profile, a node sends several packets to every other node in the network with same probability using uniform probability distribution. Figures 3.10 and 3.11 show the comparison of CHARM with other approaches for latency and throughput respectively under uniform traffic. All algorithms exhibit similar average latency and throughput at lower traffic loads. But with increased packet injection rate, it is observed that the dimension order routing (XY) performs much better than all other adaptive routing methods as expected. XY incorporates relatively long term and more global information about uniform traffic load characteristic [46]. Since XY routes packets first along the X dimension and then in the Y dimension, it distributes packets as evenly as possible throughout the network in the long-term. We observe that CHARM performs better than other adaptive routing methods at higher traffic loads.

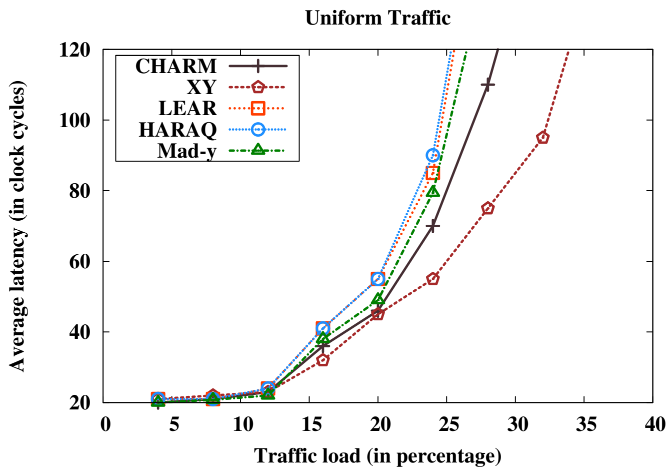


Figure 3.10: Average latency under uniform traffic for double-y network

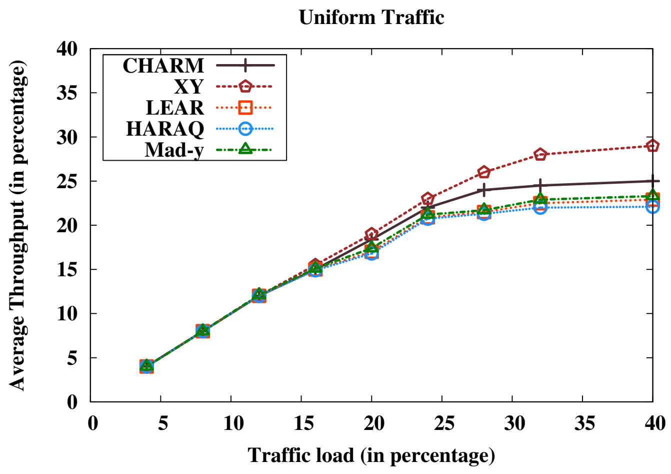


Figure 3.11: Average throughput under uniform traffic for double-y network

Figures 3.12 and 3.13 show the comparison of CHARM-XY with other similar approaches. We can observe that at higher load the DBSS performs better than the other algorithms. This is because DBSS uses XY-routing in VC-1 and adaptive routing in VC-2 which makes it better than other routing schemes under uniform traffic. CHARM-XY performs better than the other routing schemes because of additional path diversity provided.

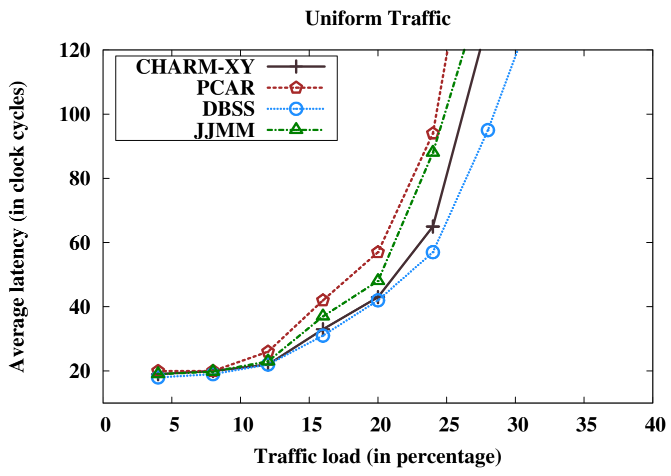


Figure 3.12: Average latency under uniform traffic for double-xy network

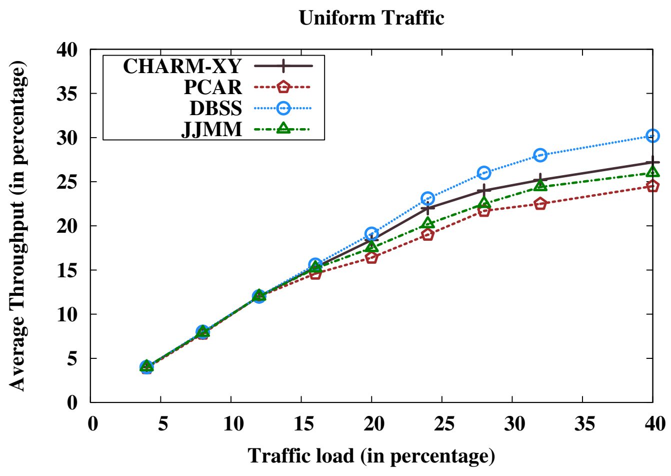


Figure 3.13: Average throughput under uniform traffic for double-xy network

3.3.2 Non-uniform Traffic Patterns

We consider two non-uniform traffic profiles (hotspot and transpose) for the evaluation of proposed method. If a node is having significantly greater demand than the other similar nodes, it is called a hot-spot. For example, in a shared memory multicomputer, a specific destination becomes a hot-spot when several processors are reading simultaneously the same memory location [20]. hotspot is considered a more realistic traffic profile [96] because, in most applications, processor (one node) communicate frequently with a small subset of other nodes (memory nodes, I/O resources). In the mesh network, we can have intermediate nodes having the Memory and I/O resource. These nodes can also receive traffic from any other node and can be positioned at any place within the mesh. This traffic pattern also reveals the load balancing capability of the routing schemes [54]. Under this traffic pattern, some nodes are appointed as hot-spot nodes, which receive some additional hotspot traffic besides their normal uniform traffic. For simulation, we set nodes $(1, 3)$, $(1, 5)$ and $(2, 0)$ as hot-spot nodes with 0.4 probability of getting additional traffic. It means that these nodes will be receiving 40% extra packets than the other nodes. The probability assumption of other nodes is uniform and decided by uniform probability distribution. However, hot-spot nodes will be receiving more packets which is decided by the additional probability.

Transpose traffic pattern is another example of non-uniform traffic like hotspot traffic. Under this traffic profile, a node at position (X, Y) only sends packets to another node at position $(n-1-X, n-1-Y)$, for a $n \times n$ 2D mesh. This traffic profile simulates the concept of transposing a matrix. This traffic profile results into a non-uniform distribution of traffic with heavy traffic flows for the central nodes creating network "hot spots".

Figures 3.14 and 3.15 show the comparison of CHARM with other similar approaches for hotspot traffic. Similarly, Figures 3.16 and 3.17 depict the performance evaluation of CHARM for transpose traffic. We can observe that performance of algorithms is almost similar in both non-uniform traffic patterns. It can be observed from Figures 3.14, 3.15, 3.16 and 3.17 that XY, in contrast with the random traffic profile, has a higher average latency and lower throughput than the three adaptive algorithms. The adaptive methods can cope with congestion better than XY. Because, when multiple traffic flows are oriented towards a small subset of "hot spot" nodes, a non-adaptive XY router will be compelled to forward them towards the same output direction, thus saturating the virtual channel queues. On the other hand, adaptive algorithms can direct packets, destined for the same destination, to different output channels. It can also be observed that due to higher adaptiveness (both minimal and non-minimal), CHARM scheme achieves better performance than other adaptive algorithms by avoiding "hot spots". CHARM method leads to better performance because it can more evenly distribute traffic in a congested network using additional paths both minimal and non-minimal than other routing algorithms.

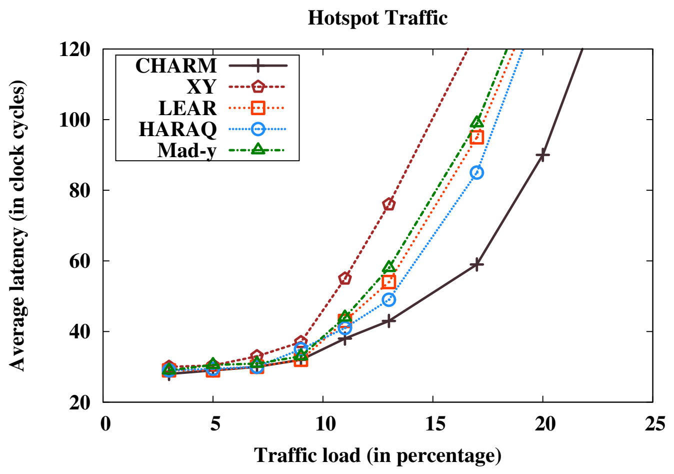


Figure 3.14: Average latency under hotspot traffic for double-y network

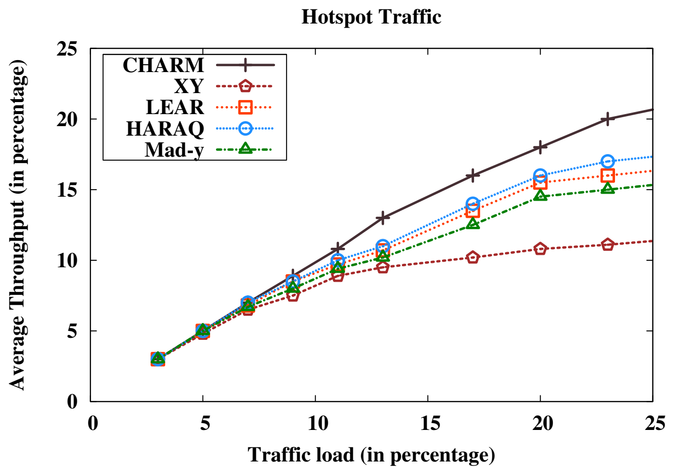


Figure 3.15: Average throughput under hotspot traffic for double-y network

Figures 3.18 and 3.19 show the comparison of CHARM-XY with other similar approaches for hotspot traffic. Similarly, Figures 3.20 and 3.21 depicts the performance evaluation of CHARM-XY for transpose traffic. It can be observed that DBSS outperforms other schemes as it provides two hop visibility for the congestion together with path diversity. However, CHARM-XY performs better than the other two algorithms because of higher adaptivity provided by it. CHARM-XY provides single hop visibility for the congestion. It should be noted that PCAR and JJMM provide better congestion detection mechanism than CHARM-XY but the the degree of adaptiveness of CHARM-XY is higher than these algorithms.

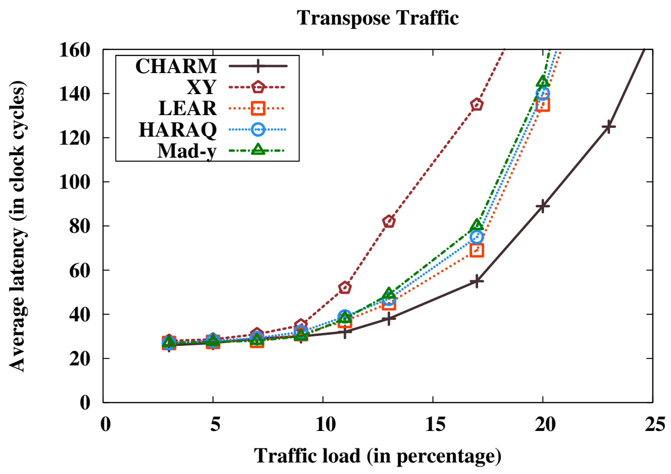


Figure 3.16: Average latency under transpose traffic for double-y network

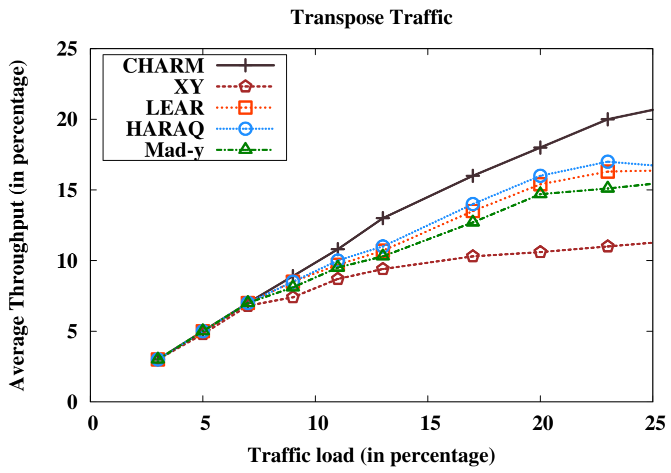


Figure 3.17: Average throughput under transpose traffic for double-y network

Thus, the performance of CHARM-XY is better than PCAR and JJMM schemes.

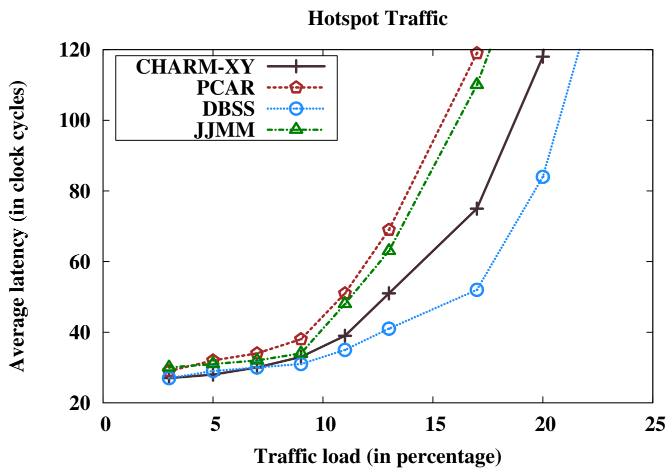


Figure 3.18: Average latency under hotspot traffic for double-xy network

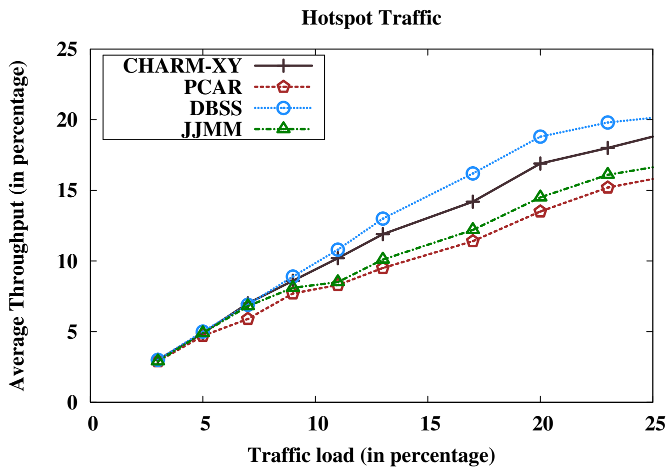


Figure 3.19: Average throughput under hotspot traffic for double-xy network

3.3.3 Application Traffic

To evaluate proposed work in a more realistic scenario, we consider E3S benchmark suite. We select four application suites automotive/industrial, networking, consumer, and office-automation. This selection is intended to represent various applications used in the real-time embedded systems. Each application suite is represented by a .tgff file. We generated communication task graphs for each .tgff file by parsing it. A communication task graph represents communication pattern

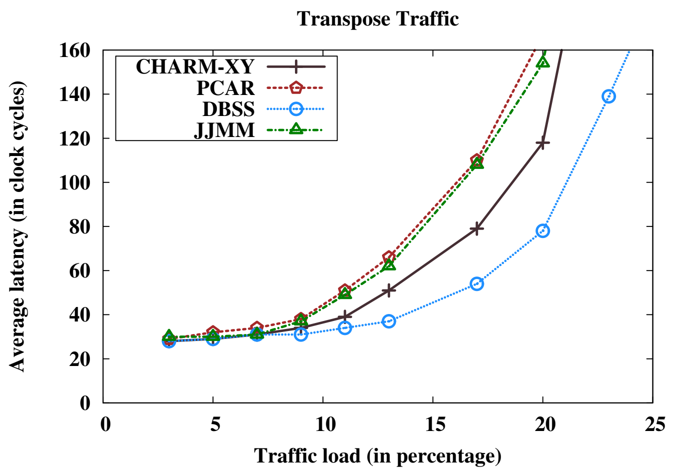


Figure 3.20: Average latency under transpose traffic for double-xy network

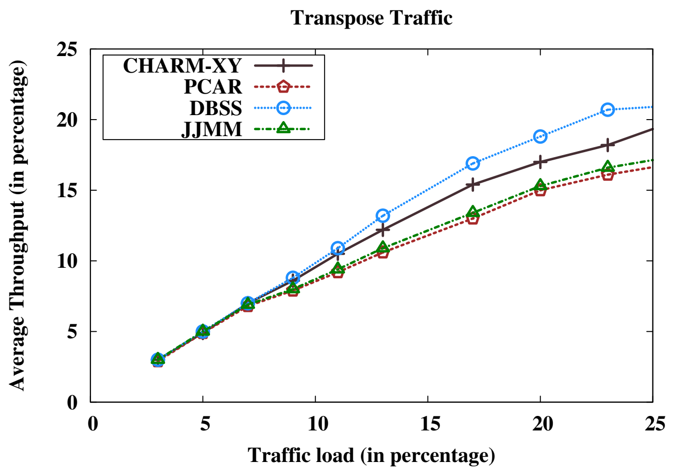


Figure 3.21: Average throughput under transpose traffic for double-xy network

and volume among tasks. We assign each task to the core (processor) using *minimum execution time scheduler* that executes it in the fastest time. Task mapping strongly depends on particular application traffic. A random mapping algorithm is used to compute the locations of cores within NoC to enable honest and unbiased comparisons among routing algorithms. We executed routing algorithms several times using random mapping, and the average of simulation results is used. Figures 3.22 and 3.23 show average packet latency normalized to XY and DBSS routing algorithms, respectively. As shown in Figure 3.22, CHARM exhibits lower latency than other methods across all four application suites in the double-y network. CHARM outperforms all other related work for all benchmark applications. The performance improvement of CHARM is 29% and 14% when compared with XY and other adaptive methods, respectively for a 7×7 mesh. However, in the case of double-xy network as shown in Figure 3.23, DBSS outperforms other algorithms. However, CHARM-XY performs better than JJMM and PCAR methods.

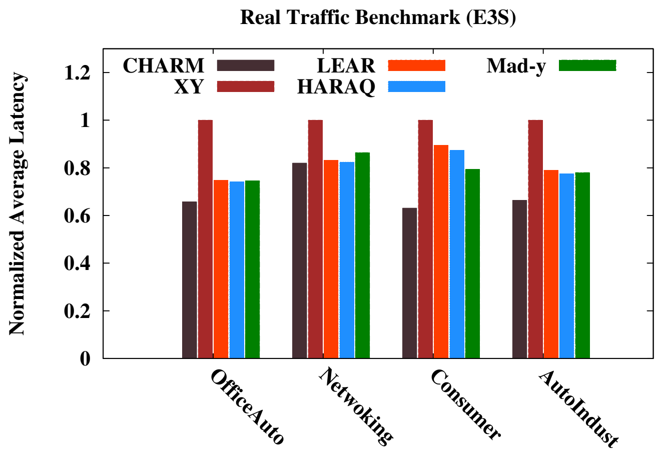


Figure 3.22: Performance for application traces for double-y network

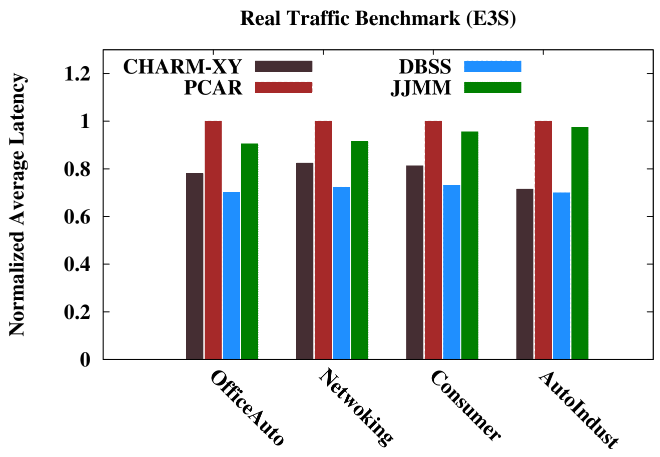


Figure 3.23: Performance for application traces for double-xy network

3.3.4 Power Analysis

We deploy an existing NoC power estimation tool ORION [63], which is integrated with NoC simulator [57]. It estimates total power consumption of a router into various sub-components: input buffers, router control logic (arbiter and crossbar) traversal and channels. Figures 3.24 and 3.25 illustrate average power consumption for hotspot traffic with different traffic loads. It can be observed from Figure 3.24 that XY consumes less power for all traffic load. Because, it always routes packets through minimal paths. At lower traffic loads, CHARM performs better than other adaptive routing methods. It uses minimal paths due to small "hot spots" creation at lower traffic load. However, other adaptive methods consume less power than CHARM at higher traffic loads. Proposed method uses non-minimal paths to alleviate congestion, which causes increase in hop count.

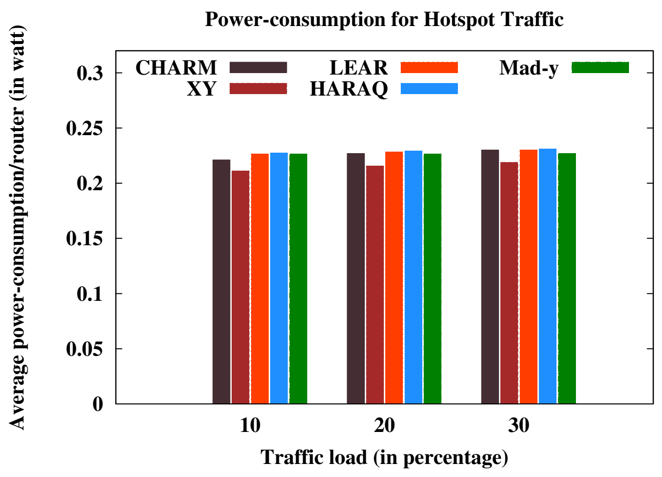


Figure 3.24: Power consumption results under hotspot traffic for double-y network

It can be observed from Figure 3.25 that PCAR consumes less power for all traffic load. Because of comparatively simple route computation function, PCAR and JJMM perform better than the CHARM-XY and DBSS schemes. DBSS consumes more average power than the CHARM-XY because of complex congestion detection mechanism.

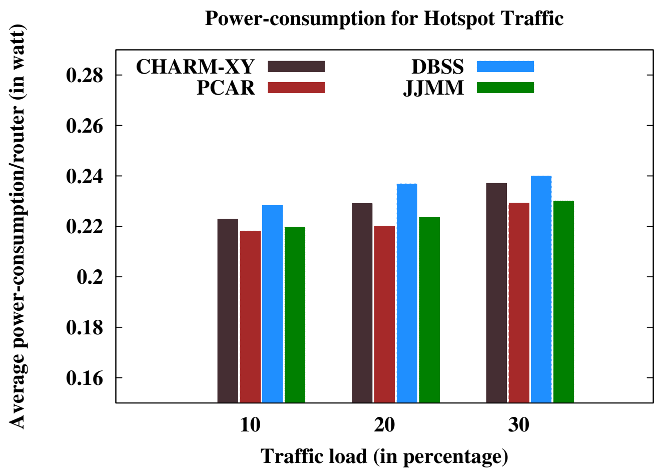


Figure 3.25: Power consumption results under hotspot traffic for double-xy network

3.3.5 Area Analysis

To calculate area overhead, we have implemented whole platform of each routing method. The whole platform includes network interfaces, routers, and communication channels and is synthesized by Synopsys Design Compiler. For synthesis, we consider the UMC 90nm technology with an operating frequency of 1 GHz and supply voltage of 1 V. Tables 3.6 and 3.7 show the layout area for different routing methods. From table 3.6, we observe that area requirement for XY is lesser than the other routing methods because of simple routing function of XY. However, other adaptive methods exhibits almost same area requirement. From table 3.7, it is observed that area requirement for DBSS is higher than other routing methods because of its complex routing logic. The area requirement of CHARM-XY is also higher than PCAR and JJMM.

Table 3.6: Area Requirement

Algorithm	**Area**(mm^2)
CHARM	7.763
XY	6.692
LEAR	7.765
Mad-y	7.532
HARAQ	7.783

Table 3.7: Area Requirement

Algorithm	**Area**(mm^2)
CHARM-XY	7.963
PCAR	7.622
DBSS	8.176
JJMM	7.632

3.4 Inferences

The degree of adaptiveness has a major impact on the performance of an adaptive routing algorithm. In this Chapter, we have proposed a novel turn model that provides high degree of adaptiveness for a 2D mesh. The end result is that the proposed turn model reduces the number of restrictions on routing turns and hence is able to provide path diversity through additional minimal and non-minimal routes between the source and destination. Deadlock freedom of CHARM is ensured using Duato's theory [24].

We have observed that proposed turn model can be extended for three dimensional mesh networks with less resources (virtual channels). Thus, next chapter presents a 3D extension to 2D-CHARM.

Chapter 4

3D-CHARM: Adaptive Routing Method for 3D-Mesh

4.1 Overview

Three-dimensional very large-scale integration (3D VLSI) is emerging as a promising solution to physical limitations of semi-conductors manufacturing processes [84]. In 3D-Integration, multiple 2D dies are vertically stacked on top of each other with Through-Silicon-Vias (TSVs). The main advantage of 3D ICs is the considerable reduction in the count and length of global interconnects, which leads to increase in performance. TSVs are used for the irregular network topology. However, our proposed turn model is applicable to the regular mesh network. Thus, we have considered simple links for connecting 2D layers.

Routing method presented in [29] uses four, four, and two virtual channels along the X, Y and Z dimensions, respectively for deadlock avoidance. Dahir *et al.* [15] presented partially adaptive routing method which is 3D extension of 2D odd-even turn model [13]. Ebrahimi [28] proposed a fully adaptive routing algorithm for 3D NoCs that uses two, two and four virtual channels along the X, Y and Z dimensions, respectively. Deadlock avoidance method adopted in well-known planar adaptive routing method [12] uses one, three and two virtual channels along the X, Y and Z dimensions, respectively. It is fully adaptive within a 2D plane only. Thus, in this chapter, we present congestion-aware, non-minimal and fully adaptive (CHARM) routing algorithm. Since, the proposed method is applicable to the 3D mesh, we have named it 3D-CHARM. It offers high degree of adaptiveness by permitting cycles in channel dependency graph while remaining deadlock free. 3D-CHARM uses only one, two and two virtual channels along the X, Y and Z dimensions, respectively. We have shown that the proposed routing algorithm results in significant performance improvement compared to existing routing schemes under certain traffic patterns designed for NoCs. It uses less number of virtual channels compared to other well known adaptive routing methods [28, 29, 12].

4.2 Proposed Method

In this section, we extend 2D-CHARM for three dimensions. We present the turn model for the 3D-CHARM with deadlock-freedom and the livelock-freedom proofs.

4.2.1 3D-CHARM: Turn Model

3D-CHARM deploys double-yz network that uses one virtual channel along X dimension ($+X$: $East$, $-X$: $West$), two virtual channels along Y dimension ($+Y$: $North$, $-Y$: *south*) and Z dimension ($+Z$: *up*, $-Z$: *down*) to achieve high degree of adaptiveness. Figures 4.1, 4.2 and 4.3 show turn model representations for different planes (YZ, XY and XZ) of 3D mesh topology. A packet is permitted to use the first virtual channel at any time as shown in Figures 4.1(i) and 4.1(ii). It can use the second virtual channel only if it has already routed to negative directions (West and south) of all lower dimensions. It is allowed to take 180-degree turn from West to East, south to North and down to up only if it has completed routing in the West, south and down directions, respectively.

For XY*-plane*, 3D-CHARM imposes same constraints as in 2D-CHARM (Section 3.2.1).

For YZ*-plane*, 3D-CHARM imposes following constraints on routing turns:

1. It prohibits four 90-degree turns ($U2$-$S1$, $U2$-$S2$, $D2$-$S1$ and $D2$-$S2$) as shown in Figures 4.1(iii) and 4.1(iv).

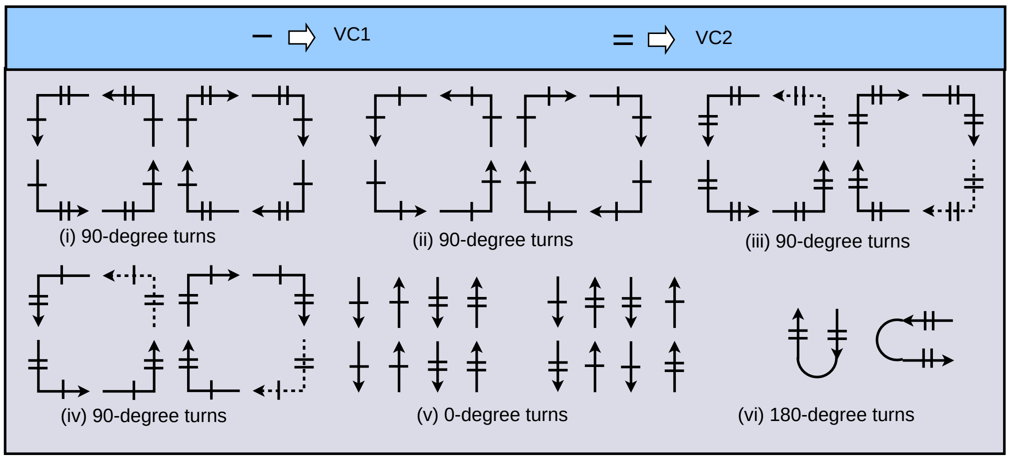


Figure 4.1: Turn model for *YZ-plane* of 3D-CHARM

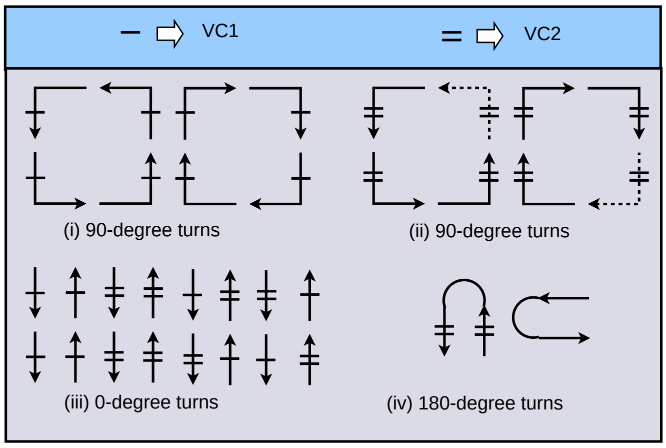


Figure 4.2: Turn model for *XY-plane* of 3D-CHARM

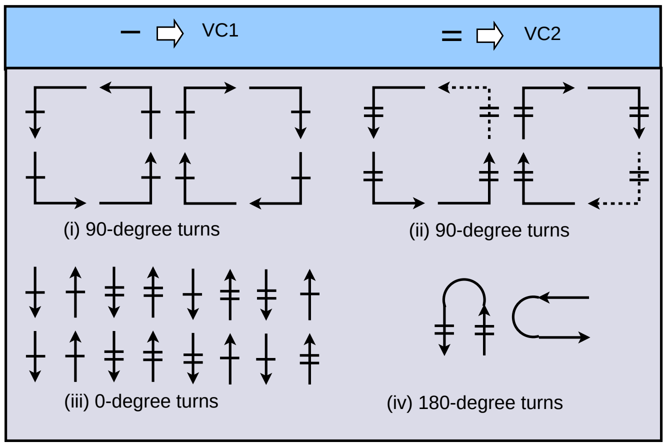


Figure 4.3: Turn model for *XZ-plane* of 3D-CHARM

2. It allows 0-degree turns ($U1$-$U1$, $D1$-$D1$, $U2$-$U2$ and $D2$-$D2$) as shown in Figure 4.1(v). 3D-CHARM allows 0-degree turns ($U1$-$U2$, $U2$-$U1$, $D1$-$D2$ and $D2$-$D1$) as shown in Figure 4.1(v), with some restrictions. It allows these restricted turns only when packet does not need to be forwarded further West or south. Similarly, we can find allowed and restricted 0-degree turns for North and south as well.

3. It permits some 180-degree turns as shown in Figure 4.1(vi).

We can deduce following restrictions from above constraints:

1. A packet is allowed to take 90-degree turns ($S2$-$U2$, $S1$-$U2$, $S2$-$D2$ and $S1$-$D2$) only when it does not need to be forwarded to West or south further. This restriction is due to prohibited 90-degree turns ($U2$-$S1$, $U2$-$S2$, $D2$-$S1$ and $D2$-$S2$).

2. A packet cannot use $U2$ or $S2$ at the source node if it needs to be forwarded West or south.

Constraints on routing turns for XZ*-plane* can be deduced in a similar fashion as in XY*-plane*.

4.2.2 Deadlock Freedom of 3D-CHARM

The deadlock-freedom proof of 3D-CHARM is divided into two main steps.

1. We first prove the deadlock-freedom of each plane individually, then

2. We prove that the deadlocks do not exist in inter-planes communications (between planes).

4.2.2.1 Deadlock Freedom: Individual Plane

To assured the deadlock-freedom of each plane, we have used Duato's theory [24] as stated in the Section 3.2.3.

Lemma 1. *Any of XY-plane does not contain any cyclic dependency if it follows proposed 3D-CHARM turn model.*

Proof. Table 4.1 shows the route computation function used for the XY-*plane*. It can be observed that the routing function of XY-*plane* is exactly same as the routing function of 2D-CHARM (Figure 3.2). We have proved in chapter 3 that the 2D-CHARM routing function is deadlock-free. Thus, we can conclude that the routing function of XY-*plane* does not have any cyclic dependency. Thus, it is deadlock-free. □

Lemma 2. *Any of XZ-plane does not contain any cyclic dependency if it follows proposed 3D-CHARM turn model.*

Proof. Table 4.2 shows the route computation function used for the XZ-*plane*. It can be observed that if we replace the $N1$ and $S1$ channels of XY-*plane* (Table 4.1) with $U1$ and $D1$, respectively we obtain the routing function of XZ-*plane* (Table 4.2). As the XY-*plane* is deadlock-free, thus the XZ-*plane* is also deadlock-free. □

To prove the deadlock-freedom of YZ-*plane* of 3D-CHARM, we again use Duato's theorem. The route computation function of YZ-*plane* of 3D-CHARM is denoted by R as shown in Table 4.3. The set of channels used by R is denoted by C ($N1$, $S1$, $U1$, $D1$, $N2$, $S2$, $U2$ and $D2$). To assure deadlock freedom of YZ-*plane* of 3D-CHARM, we first identify the subset of channels $C_1 \subseteq C$, that defines routing subfunction $R_1 \subseteq R$ that is connected and has an ECDG with no cycles arising from direct, indirect, direct-cross and indirect-cross dependencies. We have identified C_1 that has all VCs except $N1$, $S1$, $U1$ and $D1$. The *rfun* for the R_1 is shown in Table 4.4.

Lemma 3. *The routing subfunction R_1 is connected and deadlock-free.*

Proof. The *rfun* for routing subfunction R_1 (Table 4.4) with channel set C_1. To prove the deadlock-freedom of R_1, we have assigned the numbers to the channel of R_1 in such a way that the packets will always follow channels in deceasing order [18].

Table 4.1: Route computation function for *XY-plane* of 3D-CHARM

		Destination position (*des_pos*)							
		S	**N**	**E**	**W**	**SE**	**SW**	**NE**	**NW**
Input channel (*in_ch*)	**S1**	-	N1, N2	E, N1, N2	W	E, N1, N2	-	N1, N2, E	N1, W
	N1	S1, S2	N1, N2, S1, S2	E, S1, S2, N1, N2	W	S1, S2, E, N1, N2	S1, W	N1, N2, E, S1, S2	-
	S2	-	N1, N2	E, N1, N2	-	E, N1, N2	-	N1, N2, E	-
	N2	S1, S2	N1, N2, S1, S2	E, S1, S2, N1, N2	-	S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
	E	S1, S2, W	N1, N2, W, S1, S2	E, N1, N2, S1, S2, W	W	S1, S2, E, N1, N2, W	S1, W	N1, N2, E, S1, S2, W	N1,W
	W	S1, S2	N1, N2, S1, S2	E, N1, N2, S1, S2	-	S1, S2, E, N1, N2	-	N1, N2, E, S1, S2	-
	L	S1, S2	N1, N2, S1, S2	E, N1, N2, S1, S2	W	S1, S2, E, N1, N2	S1 ,W	N1, N2, E, S1, S2	N1 ,W

Table 4.2: Route computation function for *XZ-plane* of 3D-CHARM

		Destination position (*des_pos*)							
		D	**U**	**E**	**W**	**DE**	**DW**	**UE**	**UW**
Input channel (*in_ch*)	**D1**	-	U1, U2	E, U1, U2	W	E, U1, U2	-	U1, U2, E	U1, W
	U1	D1, D2	U1, U2, D1, D2	E, D1, D2, U1, U2	W	D1, D2, E, U1, U2	D1, W	U1, U2, E, D1, D2	-
	D2	-	U1, U2	E, U1, U2	-	E, U1, U2	-	U1, U2, E	-
	U2	D1, D2	U1, U2, D1, D2	E, D1, D2, U1, U2	-	D1, D2, E, U1, U2	-	U1, U2, E, D1, D2	-
	E	D1, D2, W	U1, U2, W, D1, D2	E, U1, U2, D1, D2, W	W	D1, D2, E, U1, U2, W	D1, W	U1, U2, E, D1, D2, W	U1,W
	W	D1, D2	U1, U2, D1, D2	E, U1, U2, D1, D2	-	D1, D2, E, U1, U2	-	U1, U2, E, D1, D2	-
	L	D1, D2	U1, U2, D1, D2	E, U1, U2, D1, D2	W	D1, D2, E, U1, U2	D1 ,W	U1, U2, E, D1, D2	U1 ,W

Table 4.3: Route computation function for YZ-*plane* of 3D-CHARM

		Destination position (*des_pos*)							
		D	**U**	**N**	**S**	**DN**	**DS**	**UN**	**US**
Input channel (*in_ch*)	**D1**	-	U1, U2	N1, N2, U1, U2	S1, S2	N1, N2, U1, U2	-	U1, U2, N1, N2	U1, S1, S2
	U1	D1, D2	U1, U2, D1, D2	N1, N2, D1, D2, U1, U2	S1, S2	D1, D2, N1, N2, U1, U2	D1, S1, S2	U1, U2, N1, N2, D1, D2	-
	D2	-	U1, U2	N1, N2, U1, U2	-	N1, N2, U1, U2	-	U1, U2, N1, N2	-
	U2	D1, D2	U1, U2, D1, D2	N1, N2, D1, D2, U1, U2	-	D1, D2, N1, N2, U1, U2	-	U1, U2, N1, N2, D1, D2	-
	N1	D1, D2, S1, S2	U1, U2, S1, S2, D1, D2	N1, N2, U1, U2, D1, D2, S1, S2	S1, S2	D1, D2, N1, N2, U1, U2, S1, S2	D1, S1, S2	U1, U2, N1, N2, D1, D2, S1, S2	U1,S1, S2
	N2	D1, D2, S1, S2	U1, U2, S1, S2, D1, D2	N1, N2, U1, U2, D1, D2, S1, S2	S1, S2	D1, D2, N1, N2, U1, U2, S1, S2	D1, S1, S2	U1, U2, N1, N2, D1, D2, S1, S2	U1,S1, S2
	S1	D1, D2	U1, U2, D1, D2	N1, N2, U1, U2, D1, D2	-	D1, D2, N1, N2, U1, U2	-	U1, U2, N1, N2, D1, D2	-
	S2	D1, D2	U1, U2, D1, D2	N1, N2, U1, U2, D1, D2	-	D1, D2, N1, N2, U1, U2	-	U1, U2, N1, N2, D1, D2	-
	L	D1, D2	U1, U2, D1, D2	N1, N2, U1, U2, D1, D2	S1, S2	D1, D2, N1, N2, U1, U2	D1 ,S1, S2	U1, U2, N1, N2, D1, D2	U1 ,S1, S2

Table 4.4: Route computation function for the R_1 of YZ-*plane* of 3D-CHARM

		Destination position (*des_pos*)							
		D	**U**	**N**	**S**	**DN**	**DS**	**UN**	**US**
Input channel (*in_ch*)	**D2**	-	U1, U2	N1, N2, U1, U2	-	N1, N2, U1, U2	-	U1, U2, N1, N2	-
	U2	D1, D2	U1, U2, D1, D2	N1, N2, D1, D2, U1, U2	-	D1, D2, N1, N2, U1, U2	-	U1, U2, N1, N2, D1, D2	-
	N2	D1, D2, S1, S2	U1, U2, S1, S2, D1, D2	N1, N2, U1, U2, D1, D2, S1, S2	S1, S2	D1, D2, N1, N2, U1, U2, S1, S2	D1, S1, S2	U1, U2, N1, N2, D1, D2, S1, S2	U1,S1, S2
	S2	D1, D2	U1, U2, D1, D2	N1, N2, U1, U2, D1, D2	-	D1, D2, N1, N2, U1, U2	-	U1, U2, N1, N2, D1, D2	-
	L	D1, D2	U1, U2, D1, D2	N1, N2, U1, U2, D1, D2	S1, S2	D1, D2, N1, N2, U1, U2	D1 ,S1, S2	U1, U2, N1, N2, D1, D2	U1 ,S1, S2

For a $m \times n$ *YZ-plane*, we assign a two-digit number (a,b) to each channel. The number is in radix r that satisfies following equalities:

$$\begin{cases} r > 3m - 2 \\ r > n - 1 \end{cases}$$

Figures 4.4 and 4.5 show the numbering system and corresponding number assignment, respectively for the router of R_1. It can be easily observed from Figure 4.5, for each input channel into an arbitrary node, the R_1 routes the packet only along a channel with a lower number.

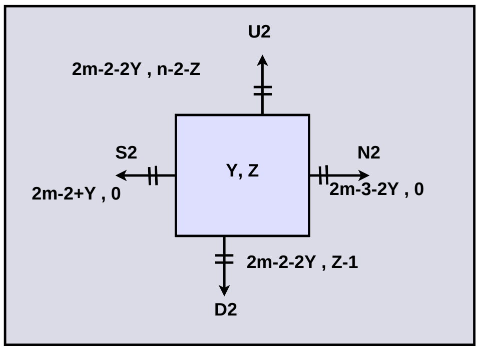


Figure 4.4: Numbering of the output channels leaving each router (Y, Z) of *YZ-plane* for R_1 routing algorithm

To further explain the numbering system, we have taken an example. The channel numbering for a 4×4 *YZ-plane* is shown in Figure 4.6. For the node $(2, 1)$, if the packet is coming on North input channel (numbered $9, 0$), then it can be routed along any output channel including North. Because all output channels have lower number than the input. However, if the packet is coming on Up input channel (numbered $2, 1$), then it can be routed only along lower numbered output channels that are Down $(2, 0)$ and North $(1, 0)$ output channels. It can be easily verified that the packets always follow decreasing order of channels if routed using routing function R_1 (Table 4.4). In addition, each router is connected to every other router in the network. Thus, we can conclude that R_1 is connected and deadlock-free.

□

Lemma 4. *The ECDG of channel set C_1 does not have any cycle because of any dependency (direct-cross, indirect-cross , direct and indirect) with additional channel ($N1$, $S1$, $U1$ and $D1$) introduced by R.*

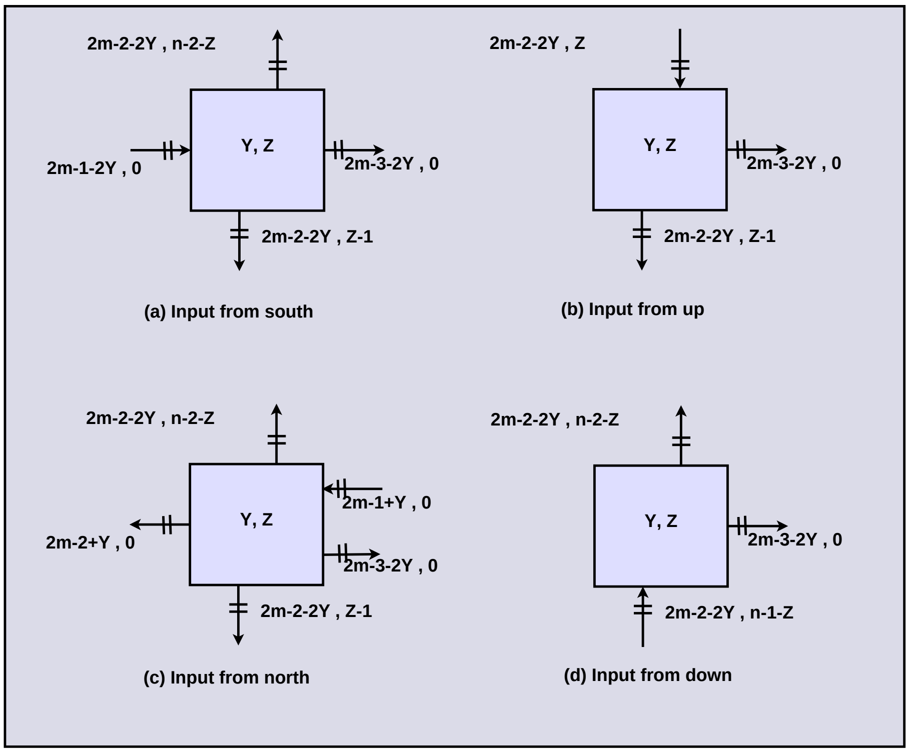


Figure 4.5: For each input channel, the eligible output channels for each router (Y, Z) of YZ-*plane* have lower number for R_1

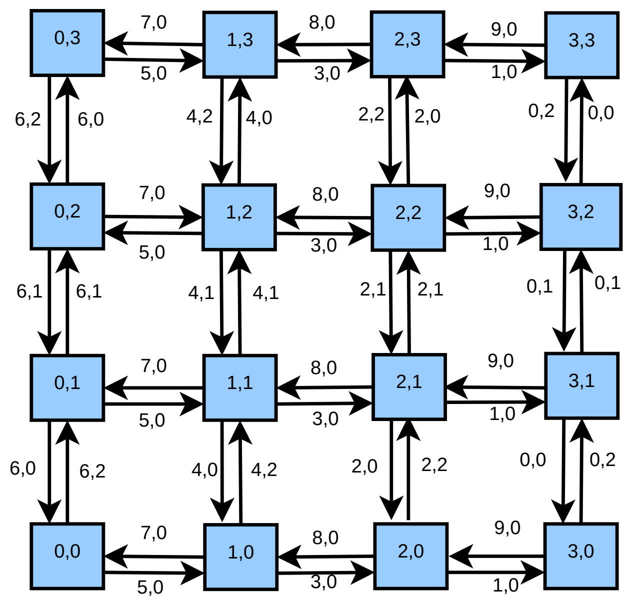


Figure 4.6: Each router (Y, Z) in 4×4 YZ-*plane* for R_1 routing function

Proof. It can be observed from Tables 4.3 and 4.4 that the routing subfunction R_1 that is defined on the channel subset C_1, is according to the following expression:

$$R_1(in\_ch, des\_pos) = R(in\_ch, des\_pos) \cap C_1, \forall in\_ch, des\_pos \tag{4.1}$$

Thus, the channels computed by the routing function R (Table 4.3) is from the set C_1 ($N2$, $S2$, $U2$ and $D2$), and is also provided by the routing subfunction R_1 for the same destination. As we have discussed in 3 that the cross dependencies only exist if we add any routing option between channels of C_1 while developing routing function R from R_1 by adding channels $N1$ and $S1$. We have not added any such routing option between channels of C_1. Thus, there does not exist any cross-dependency between channels in C_1 meaning ECDG is cycle-free because of cross dependencies.

From Lemma 3, R_1 is proved deadlock-free thus, cycle-free, thus no direct dependency can cause cycles in ECDG. However, in the R_1, a south channel is always utilized before any other channel (North or up or down) in C_1. Thus, the dependencies towards south channel from any other channel (North or up or down) are absent. However, the additional channels ($N1$, $S1$, $U1$ and $D1$) introduced by R can cause indirect dependencies between south channels as a packet can use south channel, then any addition channel ($U1$ or $D1$) and later can use south channel of different row. But this indirect dependency does not introduce any cycle in ECDG. Because to form a cycle, at least one of the 90-degree turns ($D2$-$S2$ or $U2$-$S2$) must be allowed; however, these 90-degree turns are prohibited. Thus, these indirect dependencies introduce new dependencies between only the south virtual channels and do not result in cycles. Since there are no direct and indirect dependencies which produce cycle in ECDG of C_1. Therefore, ECDG of C_1 is acyclic because of direct and indirect dependencies. □

Theorem 1. *The routing function proposed for the YZ-plane is deadlock-free.*

Proof. It can be concluded from Lemma 3 & Lemma 4 and using Theorem 1 that proposed routing algorithm is deadlock-free. □

4.2.2.2 Deadlock Freedom: Inter-plane Communication

Now, we prove that there does not exist any deadlock configuration for the inter plane communication. Using Duato's terminology, the route computation function of 3D-CHARM is denoted by R' and the set of channels used by R' is denoted by C'. To assure deadlock-freedom of 3D-CHARM, we first identify the subset of channels $C'_1 \subseteq C'$, that defines routing subfunction $R'_1 \subseteq R'$ that is connected and has an ECDG with no cycles arising from direct, indirect, direct-cross and indirect-cross dependencies. For 3D-CHARM, C'_1 has all VCs except $N1$, $S1$, $U1$ and $D1$.

Lemma 5. *The routing subfunction R'_1 is connected.*

Proof. R'_1 routing subfunction with channel set C'_1 is all-but-one-negative-first [46] routing algorithm. Since non-minimal all-but-one-negative-first routing is connected, so R'_1 is connected. □

Lemma 6. *ECDG of channel set C'_1 with additional channel ($N1$, $S1$, $U1$ and $D1$) introduced by R', does not result in any cyclic dependencies.*

Proof. There is no direct-cross dependency in ECDG of C'_1 as routing function R' does not add any new routing option between channels of C'_1 directly. However, additional channels introduced by routing function R' add new routing options between channels of C'_1 indirectly, but it does not produce any indirect-cross dependency. Additional channels introduced by R' can cause only indirect dependencies between West (south) channels as a packet can use West (south) channel and later can use West (south) channel of different row and column. But this indirect dependency does not introduce any cycle in ECDG of C'_1. The ECDG for C'_1 has no dependencies from a channel in North, East, south, up, or down directions to a channel in the West direction, so the West channels are always used before all other channels in C'_1. Hence, these indirect dependencies introduce new dependencies between only the West VCs and create no cycles using only the West VCs. Similarly ECDG for C'_1 has no dependencies from a channel in North, up, or down directions to a channel in the south direction, so the south channels are always used before other channels (North, up and down) in C'_1. These indirect dependencies introduce dependencies from West to the south or between south VCs. Hence, these indirect dependencies create no cycles using only the south VCs. Since there are no indirect and direct dependencies, which produce cycle in ECDG. Therefore ECDG of C'_1 is acyclic. □

Theorem 2. *The proposed routing algorithm is deadlock-free.*

Proof. It can be concluded from Lemma 5 & Lemma 6 and using Theorem 1 that proposed routing algorithm is deadlock-free. □

4.2.3 Livelock Freedom of 3D-CHARM

Non-minimal routing algorithms are susceptible to livelock. The proposed routing algorithm is proved to be livelock free using following theorem.

Theorem 3. *The proposed routing algorithm is livelock-free.*

Proof. From the discussion in Section 3.2, we can notice that whenever a packet is routed in the East direction, it is not allowed to route it back in the West direction. So, in the worst scenario, the packet may reach the West most column then it starts to move towards destination column. Similarly, whenever a packet is routed in the North direction, it is not allowed to route back in the south. Similarly, whenever a packet is routed in the up direction, it is not allowed to route back in down. So, in the worst case, the packet may reach the down most XY*-plane* then it starts to move towards the destination node. In each dimension, only one 180-degree turn is allowed. Therefore, after a limited number of hops, the packet reaches its destination node. Thus, proposed routing algorithm is livelock free. □

4.3 Results Analysis

We evaluate 3D-CHARM and all other routing methods on both synthetic and realistic traffic profiles. As synthetic traffic profiles, we consider both uniform (*random*) and non-uniform (*hotspot*) traffic patterns. For realistic communication profiles, we consider traces of some application suites extracted from benchmark suite E3S [22]. In order to perform all required set of experiments involving various routing methods and traffic profiles, we modified and extended [57], a cycle-accurate and open source SystemC based NoC simulator.

Packet size and input-channel buffer size for each virtual channel are kept constant for all experiments and set to 8 and 6 flits, respectively. The simulator is warmed up for $5,000$ cycles and afterward; the average performance is measured over another $30,000$ cycles, out of which traffic is generated over $20,000$ cycles.

Congestion threshold is set to 66% of total buffer size. As communication performance parameter, we consider latency (delay). It is defined as the time difference (in clock cycles) between header flit injection from source router and tail flit reception at the destination router.

To evaluate the effectiveness of the proposed routing method (3D-CHARM), we have implemented its two versions: 1) 3D-CHARM, and 2) 3D-CHARM-XY. 3D-CHARM uses one and two VCs in X and Y dimensions respectively. For the comparison purpose, we have also implemented three other routing methods. These methods include the dimension order routing (XYZ), 3D Odd-Even (3D-OE) [15], and all-but-one-negative-first (ALL-but-1-NF) [46]. For all experiments of 3D-CHARM, we consider $4 \times 4 \times 4$ mesh.

Whereas, the 3D-CHARM-XY uses two VCs in each of X and Y dimensions and restricts all 180-degree turns. The second version of proposed method (3D-CHARM-XY) is implemented to provide fair comparison with the recent state-of-art. To compare 3D-CHARM-XY, we have also implemented two version of Elevator-First (EF) routing [27] namely EF-0 and EF-10. EF-n means that n percentage of vertical channels are removed. We have also implemented two versions of East-Then-First (ETW) [90] routing namely ETW-0 and ETW-10. For experiments of 3D-CHARM-XY, we implemented $4 \times 4 \times 3$ mesh.

4.3.1 Uniform Traffic

Under uniform (random) traffic profile, a node sends several packets to every other node in the network with same probability using uniform probability distribution. The load latency graph for 3D-CHARM for uniform traffic model is presented in Figure 4.7. As shown, all algorithms exhibit similar average latency at lower traffic loads. But with increased packet injection rate, it is observed that the dimension order routing (XYZ) performs much better than all other adaptive routing methods as expected. XYZ incorporates relatively long term and more global information about uniform traffic load characteristic [46]. Since XYZ routes packets first along the X dimension and then in the other dimensions, it distributes packets as evenly as possible throughout the network in the long-term. On the other hand, adaptive schemes select output channels using local short-term information about the network. This selection decision for packets is advantageous only in immediate future and may tend to create “hot spots” region for other packets. In fact, these

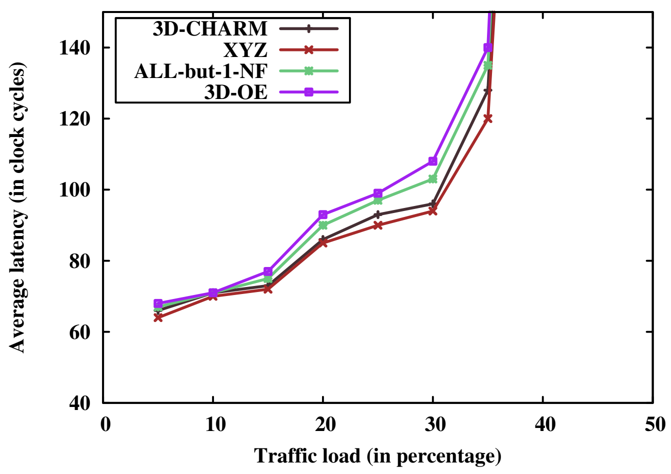


Figure 4.7: Average latency under uniform traffic for $4 \times 4 \times 4$

channel selections hinder the global long-term evenness of random traffic profile by creating zigzag routes. This results into increase contention and reduction in performance at higher packet injection rates. We observe that 3D-CHARM performs better than other adaptive routing methods at higher traffic loads.

Figure 4.8 presents the load latency graph for 3D-CHARM-XY for uniform traffic model. As shown, all algorithms exhibit similar average latency at lower traffic loads. However, at higher load, we can observe that EF-0 perform better than other algorithms. This is because of dimension order routing (DOR) which is used by EF-0 in the 2D plane. It should be noticed that EF-10 performs lower than the EF-0. However, EF-10 also uses DOR as the baseline routing, but it does not have some of the vertical links which increases the average path length. The performance of 3D-CHARM-XY is better than ETW algorithms. Both 3D-CHARM-XY and ETW use adaptive routing function as the baseline routing in 2D plane. However, the baseline routing function of ETW algorithms is nothing but the Mad-y routing which is less adaptive than the baseline routing function of 3D-CHARM-XY (2D-CHARM).

4.3.2 Hotspot Traffic

Under this traffic pattern, we set nodes $(2, 2, 0)$, $(3, 1, 1)$ and $(2, 1, 2)$ as hot-spot nodes with 0.4 probability of getting additional traffic.

The performance evaluation of 3D-CHARM is shown in the Figures 4.9 for $4 \times 4 \times 4$ mesh. It can be observed that XYZ, in contrast with the uniform traffic profile,

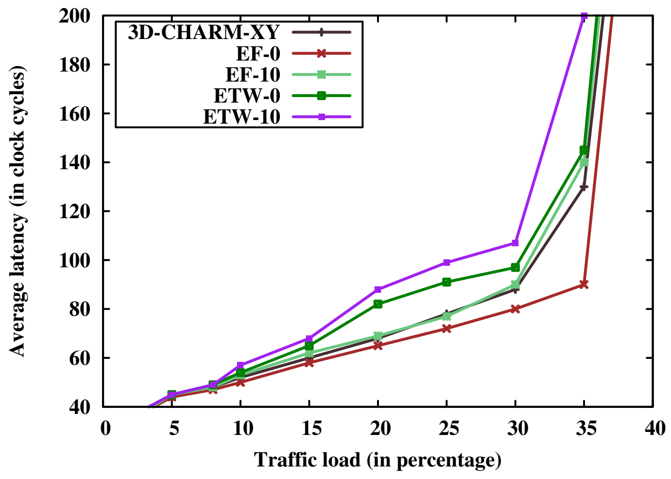


Figure 4.8: Average latency under uniform traffic for $4 \times 4 \times 3$

has a higher average latency than the three adaptive algorithms that can cope with congestion better. Because, when multiple traffic flows are oriented towards a small subset of "hot spot" nodes, a non-adaptive XYZ router will be compelled to forward them towards the same output direction, thus saturating the virtual channel queues. On the other hand, adaptive algorithms can direct packets, destined for the same destination, to different output channels. It can also be observed that due to higher adaptiveness (both minimal and non-minimal), 3D-CHARM scheme achieves better average latency than other adaptive algorithms by avoiding "hot spots". 3D-CHARM method leads to smaller average latencies because it can more evenly distribute traffic in a congested network using additional paths both minimal and non-minimal than other routing algorithms.

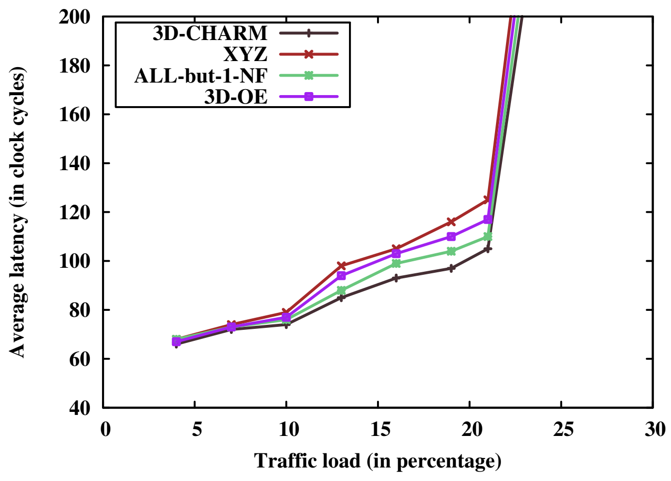


Figure 4.9: Average latency under hotspot traffic for $4 \times 4 \times 4$

Figure 4.10 presents the load latency graph for 3D-CHARM-XY for hotspot traffic model for $4 \times 4 \times 3$ mesh. All schemes show similar average latency at lower traffic loads. However, at higher load, the performance of 3D-CHARM-XY is better than all other routing as it is both more adaptive and fully connected. The baseline routing function DOR of EF algorithms becomes the bottleneck for them. EF algorithms use deterministic routing DOR. The performance of 3D-CHARM-XY is better than ETW algorithms because the baseline routing function of 3D-CHARM-XY provides more number of paths than the baseline routing of ETW algorithms.

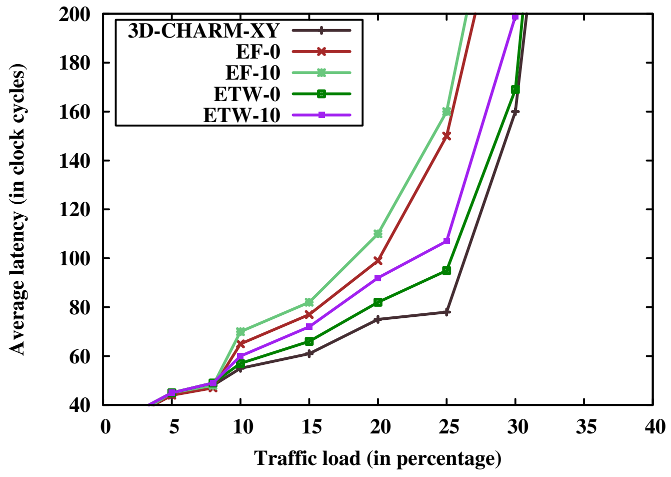


Figure 4.10: Average latency under hotspot traffic for $4 \times 4 \times 3$

4.3.3 Application Traffic

To evaluate proposed work in a more realistic scenario, we consider E3S benchmark suite. We select four application suites automotive/industrial, networking, consumer, and office-automation. This selection is intended to represent various applications used in the real-time embedded systems. We executed routing algorithms several times using random mapping, and the average of simulation results is used.

Figure 4.11 shows average packet latency normalized to XYZ routing. 3D-CHARM provides lower latency than other methods across all four application suites. 3D-CHARM shows the greatest performance gain on consumer application traces. The average performance gain of 3D-CHARM is up to 31% across all selected benchmarks vs. XYZ and 16% vs. other adaptive algorithms for the $4 \times 4 \times 4$ mesh.

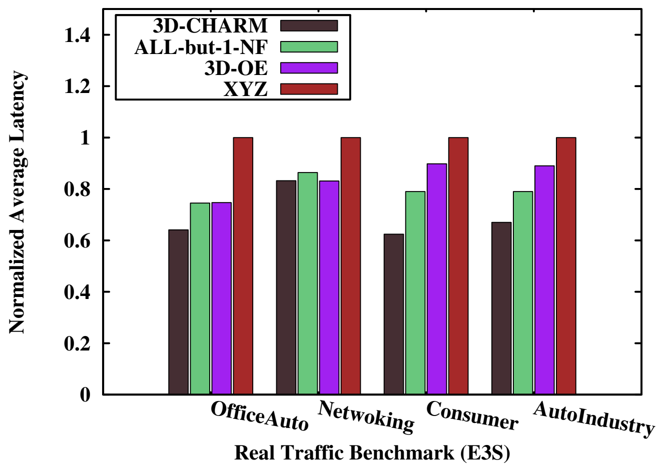


Figure 4.11: Performance for application traces for $4 \times 4 \times 4$

Figure 4.12 presents average packet latency normalized to EF-10 routing. 3D-CHARM-XY provides lower latency than other methods across all four application suites. The average performance improvement of 3D-CHARM is up to 30% vs. EF algorithms and 23% vs. ETW algorithms for the $4 \times 4 \times 3$ mesh. Both 3D-CHARM-XY and ETW algorithms are adaptive. However, it should be noticed that 3D-CHARM-XY always uses minimal path whereas ETW algorithms use both minimal and non-minimal paths. Moreover the path diversity provided by baseline routing of 3D-CHARM-XY is more than the baseline routing of ETW.

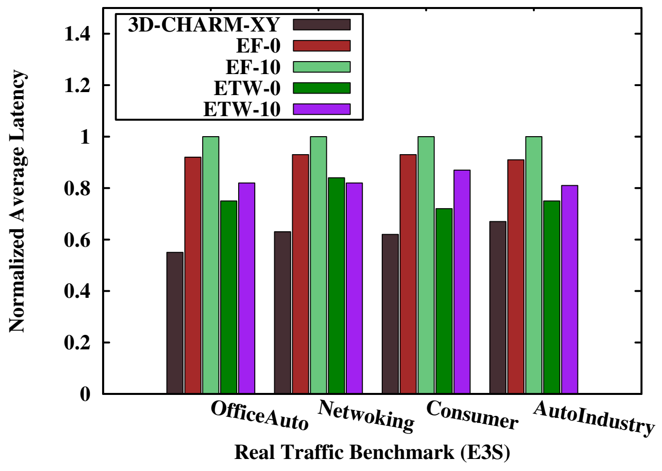


Figure 4.12: Performance for application traces for $4 \times 4 \times 3$

4.3.4 Power Analysis

Figure 4.13 illustrates average power consumption for hotspot traffic for the 4×4×4 mesh with different traffic loads. It can be observed that XYZ consumes less power for all traffic load. Because, it always routes packets through minimal paths and less complex routing function. At lower traffic loads, 3D-CHARM perform better than other adaptive routing methods. It uses minimal paths due to small "hot spots" creation at lower traffic load. However, other adaptive methods perform better than 3D-CHARM at higher traffic loads. Because of use of non-minimal paths to alleviate congestion, the power consumption of the proposed method increases. The increased hop count results in more power consumption.

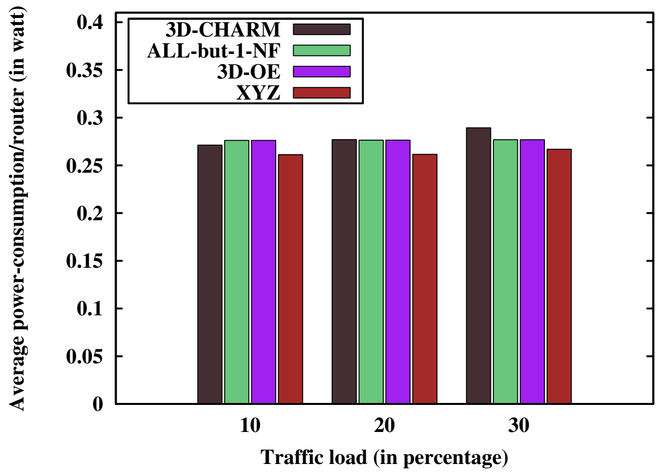


Figure 4.13: Power consumption results under hotspot traffic for $4 \times 4 \times 4$

Figure 4.14 depicts average power consumption for hotspot traffic for the $4 \times 4 \times 3$ mesh. It can be observed that EF-0 consumes less power for all traffic load. Because, it always routes packets through minimal paths. However, 3D-CHARM-XY also uses minimal paths but it has comparatively more complex routing function than EF-0, thus consume more power than EF-0. The ETW-10 algorithm consumes higher power than other routing schemes as it's routing function is more complex than other routing algorithms and it uses non-minimal paths as well.

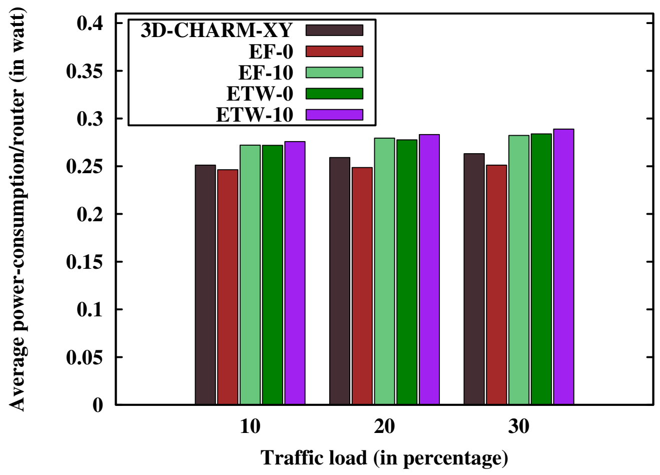


Figure 4.14: Power consumption results under hotspot traffic for $4 \times 4 \times 3$

4.3.5 Area Analysis

To calculate area overhead, we have implemented router of each routing method. We deploy an existing NoC area estimation tool ORION [63], that is integrated with NoC simulator [57]. Table 4.5 shows the router area for different routing methods. We observe that area requirement for XYZ is lesser than the other routing methods because of simple routing function of XYZ. However, 3D-CHARM results in higher area overhead than all other routing algorithms because of complex routing logic. The routing unit is a light-weight unit and the area consumption of EF and ETW are nearly the same. In a router, buffers are the most area-hungry components [90]. As EF uses one more VC than ETW, occupying a relatively larger area. As 3D-CHARM-XY uses 2 VCs and its routing logic is complex, it occupies relatively larger area than EF and ETW methods.

Table 4.5: Area Requirement

Algorithm	**Area(mm^2)**
3D-CHARM	.189876
ALL-but-1-NF	.172541
3D-OE	.178721
XYZ	.166378

4.4 Inferences

The requirement of acyclic channel dependency graph for deadlock avoidance in on-chip network routing places unnecessary thus avoidable restrictions on routing turns. Hence, it reduces degree of adaptiveness. In this chapter, we propose a novel highly adaptive and congestion-aware routing algorithm to address aforesaid issues for 3D meshes. The proposed algorithm permits cyclic dependencies in channel dependency graph providing higher degree of adaptiveness. The algorithm uses congestion-aware channel selection policy that results into the balanced distribution of traffic load across the network. A packet, once injected in NoC, follows non-minimal paths only when minimal paths are congested at the neighboring routers.

Fault tolerance is another aspect of a routing algorithm. The 2D-CHARM (Chapter 3) and 3D-CHARM (Chapter 4) methods provide a high degree of adaptive. These algorithms are not fault-tolerant. However, if we allow some more turns in 2D CHARM turn model, we can get new routing algorithm that would be fault-tolerant. Thus, in the next chapter, we present a fault tolerant and highly adaptive routing for the 2D mesh.

Chapter 5

FTCAR: Fault-tolerant and Congestion-Aware Routing

5.1 Overview

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

In NoCs, any of the components (cores, links or routers) may fail which results in faults. When a core is faulty at a node, the core can be deactivated while the connected router and links at the node can continue functioning. The connected router and links can be used to receive packets from the other nodes and route them further. Therefore, a faulty core does not affect the routing functionality of the node. However, once a link or router is failed, the faulty component cannot be simply discarded as it results in the blocking of other packets inside the network. To tolerate faults, we can use fault tolerant routing algorithms. These algorithms can be classified into two groups: addressing faulty links, and addressing faulty routers. Usually, the algorithms within each group can be modified to tolerate faults from the other group.

Generally, non-minimal routing is used to tolerate faults and alleviate congestion. In [10], authors proposed non-minimal fault tolerant routing algorithm to handle multiple nodes and links failures without using routing tables. Authors in [66], proposed distributed and reconfigurable fault tolerant (DRFT) routing scheme to tolerate single link faults. In the absence of faults, proposed scheme works as dimension order routing. Fick *et al.* [38] introduced a table-based implementation of routing scheme to tolerate faulty components. In [31], authors proposed a fault tolerant routing scheme. It can handle one faulty link or node. But, it provides lower degree of adaptiveness. In [34, 32], Ebrahimi *et al.* proposed non-minimal routing schemes for 2D mesh. These provide better adaptiveness than Mad-y [45] with the same number of virtual channels. However, these impose some unnecessary restrictions on routing turns, which can be removed to achieve fault tolerance and increased path diversity. The proposed method can handle faults occurred at manufacturing time. However, the position of fault can be anywhere within the mesh.

5.2 Proposed Method

In a routing algorithm, an acyclic CDG requirement to avoid deadlocks imposes unnecessary restrictions on the routing turns. The deadlock-freedom of both Mad-y and LEAR routing algorithms is proved using acyclic CDG [18]. Thus, they cannot use all qualified turns to route packets through less congested/faulty areas. The proposed model FTCAR imposes substantially fewer restrictions on routing turns (specially on 90-degree) using [24], thus it provides additional minimal and non-minimal paths between source and destination nodes than Mad-y and LEAR.

Figure 5.1 shows turn model representation of FTCAR. A packet is permitted to use first virtual channel ($N1$ or $S1$) at any time as shown in Fig. 5.1(i). It can use second virtual channel ($N2$ or $S2$) only if it has already routed to the negative direction of X dimension (West). Because, the packet cannot take West turn after using ($N2$ or $S2$) as shown in Fig. 5.1(ii.b.). However to achieve fault tolerance, we have to allow packets to take West turn after using ($N2$ or $S2$) but only at West border as shown in Fig. 5.1(ii.a.). The detailed explanation is provided in the Section 5.2.2.

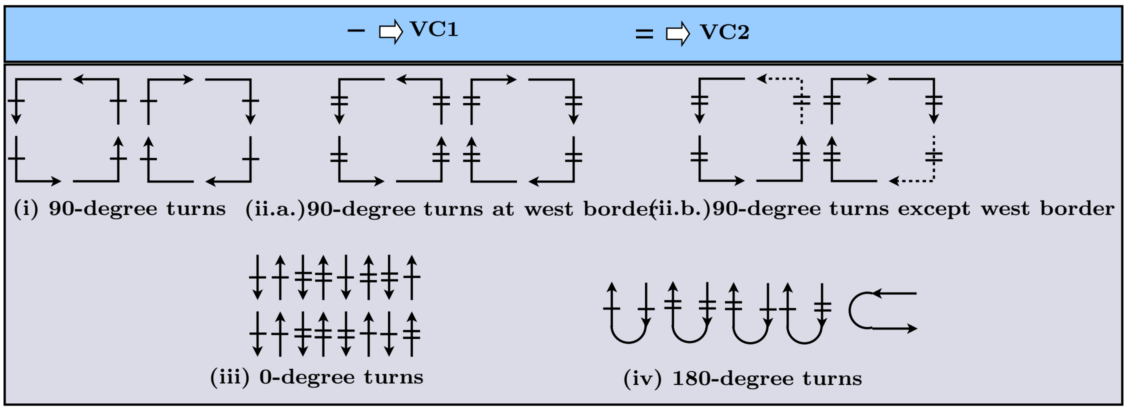

– ⇨ VC1
= ⇨ VC2
(i) 90-degree turns
(ii.a.)90-degree turns at west border
(ii.b.)90-degree turns except west border
(iii) 0-degree turns
(iv) 180-degree turns

Figure 5.1: FTCAR turn model (permitted (prohibited) turns are represented by solid (dash) lines)

It is allowed to take only two 180-degree turns from West to East (W-E) and South to North ($S2$-$N2$) as shown in Fig. 5.1(iv) only if it has completed routing in West and South directions, respectively. In short, in order to avoid deadlocks, FTCAR imposes following constraints on routing turns:

1. It prohibits two 90-degree turns ($S2$-W and $N2$-W) except west border.
2. It allows 0-degree turns ($S1$-$S1$, $N1$-$N1$, $S2$-$S2$ and $N2$-$N2$) as shown in Fig. 5.1(iii). It allows 0-degree turns ($S1$-$S2$, $N1$-$N2$, $S2$-$S1$ and $N2$-$N1$), as shown in Fig. 5.1(iii), with some restrictions. It allows these restricted turns only when packet does not need to be forwarded further West.
3. It permits some 180-degree turns (W-E and $S2$-$N2$).

The proposed method FTCAR provides more minimal and non-minimal paths between source and destination by allowing some routing turns which were prohibited in Mad-y and LEAR. For example, E-$N1$ (i.e. a packet moving from East to North-1 virtual channel) and E-$S1$ (i.e. a packet moving from East to South-1 virtual channel) are permitted by FTCAR turn model. However, these permitted routing turns create cycles in CDG, but we have shown that proposed routing method is deadlock free using Duato's well known theorem [24]. Duato has proved that cycles can be allowed in CDG provided that ECDG is acyclic.

5.2.1 Deadlock and Livelock Freedom of FTCAR

With non-adaptive routing, packets are routed along single output channel at each node. Thus, in order to achieve deadlock freedom, it is essential to eliminate all cyclic dependencies between network channels. In adaptive routing, packets often have many choices for routing at each node. Thus, it is not mandatory to remove all cyclic dependencies between channels, provided that every packet can be routed on a path whose channels are free from cyclic dependencies. The channels involved in these acyclic routes are considered as *escape channels* from deadlocks (cycles).

The FTCAR can be proved deadlock-free by using Duato's well known theory [24] stated as follows:

Theorem 1. (Duato's Theorem) *For a given network I, a connected and adaptive routing function R is deadlock-free if there exists a routing sub-function $R_1 \subseteq R$, which is connected and has extended channel dependency graph acyclic.*

Following Duato's terminology, the routing function of FTCAR is denoted by R and the channel set used by routing function R is denoted by C. To prove the FTCAR deadlock-free, we first identify the subset of channels $C_1 \subseteq C$, that defines routing subfunction $R_1 \subseteq R$ that is connected and has an acyclic ECDG. The ECDG must not contain any cycle arising from direct, direct-cross, indirect and indirect-cross dependencies. For FTCAR, the channel set C_1 contains all channels of the network except North-1 ($N1$) and South-1 ($S1$).

Lemma 1. *The routing subfunction R_1 is connected.*

Proof. The routing subfunction R_1 with the channel set C_1 is same as West-first [46] routing (non-minimal) function. Since non-minimal West-first routing is connected, so we conclude that R_1 is connected. $\square$

Lemma 2. *The extended channel dependency graph of channel set C_1 with additional dependencies introduced by channel ($N1$ and $S1$) of R is acyclic.*

Proof. There is no direct-cross dependency between channels of C_1 as a channel belonging to C_1 is always used as escape channel for all the destination for which it can be supplied by R. The additional channels ($N1$ and $S1$) of R can introduce only indirect dependencies between West channels. Because a packet using West channel can further use the West channel of different row and column. But because of this indirect dependency, there is no cycle in ECDG of C_1. The ECDG for C_1 has no dependencies to a West channel from a channel in North, East, or South, so the West channels are always used before all other channels in C_1, if destination is in West. Hence, these indirect dependencies introduce new dependencies between only the West virtual channels and create no cycles using only the West virtual channels. Since there are no indirect and direct dependencies, which produce cycle in ECDG. Therefore, ECDG of C_1 is acyclic. □

Theorem 2. *The proposed routing algorithm is deadlock-free.*

Proof. Using Lemma 1, Lemma 2 and Theorem 1 , we conclude that FTCAR routing algorithm is deadlock-free. □

Non-minimal routing algorithms are susceptible to livelock. The FTCAR can be proved to be livelock free using following theorem.

Theorem 3. *The proposed routing algorithm is livelock-free.*

Proof. We can observe from FTCAR turn model (Fig. 5.1) that whenever a packet is forwarded along East output channels, it is not allowed to route the packet back along West output channel. So, in the worst case, the packet may reach the West border, and then it starts to move towards destination column. Similarly, whenever a packet is forwarded along North output channel, it is not allowed to route back in the South. It can be observed from turn model (Fig. 5.1) that only one 180-degree turn is allowed in each dimension. Therefore, after a certain number of hops, the packet finally arrives at the destination node. Thus, proposed routing algorithm (FTCAR) is livelock free. □

5.2.2 FTCAR: Fault tolerance Analysis

Since FTCAR is a fully adaptive routing method, it can use all minimal paths for each source-destination pair, if the fault is absent. A destination node (**D**) may be located in any of eight directions/quadrants (North, South, East, West, northeast, northwest, southeast, and southwest) with respect to current/source (**C**) node. When the destination node is located in any of quadrant/direction, available minimal paths (S_{min}) between current node (0,0) and destination node (X_1, Y_1) is given by:

$$S_{min} = \frac{(|X_1| + |Y_1|)!}{|X_1|!|Y_1|!}$$

It can be notice that S_{min} is always greater or equal to 2 (≥ 2) for any quadrant (northeast, northwest, southeast, and southwest). Thereby, when the destination is in any of quadrants, and there is a single faulty link, packets can always find a minimal path to bypass the fault.

If the destination node is located in any of direction (East, West, North, and South), S_{min} is always 1. Thus, packets must follow non-minimal paths if any link on the minimal path is faulty. Figure 5.2 illustrates how FTCAR tolerates single link failures in any of direction. When destination node is located in the East (eastward packet), at first the East link is checked. If it is available, the packet is routed through this link. However, if the link is faulty as shown in Fig 5.2a, the packet can be forwarded through $N2$ or $S2$ link. By inspecting the required turns, it can be seen that eastward packets use either the $N2$-E and E-$S2$ turns or the $S2$-E and E-$N2$ turns to bypass the faulty link. It can be easily verified that all these turns are permitted turns in FTCAR turn model (Fig. 5.1). Similarly, we can observe that westward packets can use either the $N1$-W and W-$S1$ turns or the $S1$-W and W-$N1$ turns to bypass the faulty West link as shown in Fig 5.2b. Similarly, northward and southward packets use the turn W-$N2$, $N2$-E, W-$S2$ and $S2$-E to bypass the North and South faulty links as shown in Fig. 5.2c and 5.2d, respectively. The turns used by westward, northward and southward packets are allowed turns in FTCAR turn model (Fig. 5.1).

Figures 5.2e and 5.2f illustrate two special cases of northward and southward packets on the West border of the mesh. When both current and destination nodes are located on the West border, the required turns include $N2$-W and $S2$-W to bypass North and South faulty links, respectively. These turns are prohibited in the FTCAR turn model (Fig. 5.1). However, if we allow these prohibited turn on the West border, they cannot result into complete cycle. Thus, these prohibited turns can be allowed on the West border.

In FTCAR, each node keeps information about the link status for each direction (East, West, North, and South). FTCAR uses the statuses of four links to make its routing decision, in the case of faults. The functionality of FTCAR routing algorithm is divided into two phases: route computation and output channel selection. On the basis input channel (on which packet has arrived) and relative position of destination node with respect to current node, routing function computes a set of output channels using turn model explanation discussed in Section 5.2.

The selection function selects one output channel from the set of output channels provided by the route computation function. Our selection function prefers adaptive output channels over escape channels because it results in increased probability of escape output channels being available when they are required to avoid deadlocks. The selection function first checks all qualified output channels corresponding to shortest routes and forwards the packet to the output channel in which the corresponding next hop node has its congestion status flag set to zero. If the congestion status flags of all next hop nodes on shortest routes are set to one, the congestion status flag of each eligible non-minimal route is inspected. If there exist such non-minimal routes, which are not congested, our method selects one of the output channels to forward the packet (and possibly, corresponding to *non-escape channels*).

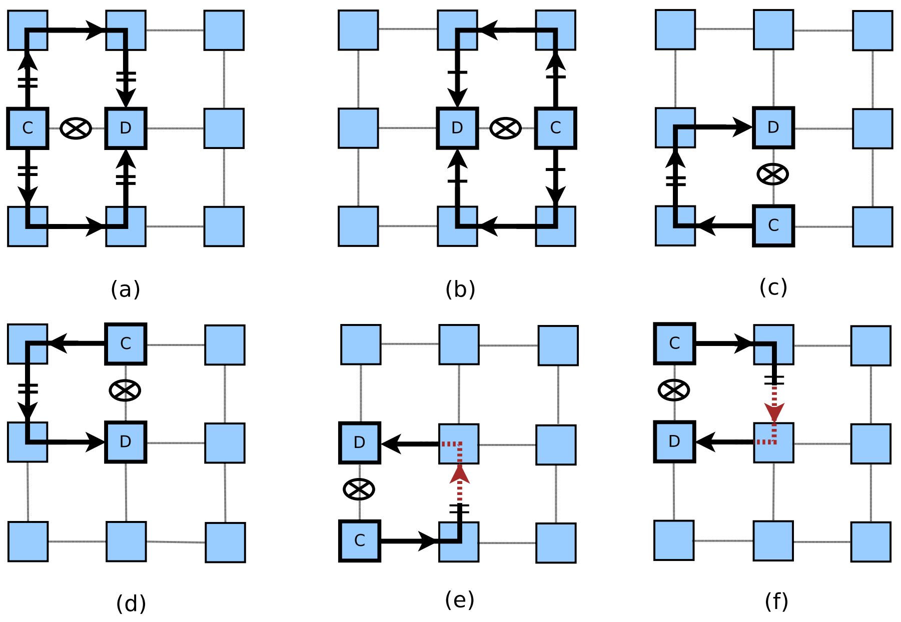


Figure 5.2: Tolerating single link failures for packet directions (a) *eastward* (b) *westward* (c) *northward* (d) *southward* (e) *northward on West border* (f) *southward on West border* (red dash lines indicate prohibited turns)

5.3 Results Analysis

In order to perform simulation involving various routing schemes, traffic scenarios, and power analysis, we modified and extended [57], a cycle-accurate SystemC based open source NoC simulator. We have evaluated proposed routing method (FTCAR) with real and synthetic traffic profiles. To evaluate the effectiveness of FTCAR, we have also implemented other routing methods FTR [61], FLEAR [31], TARRA [56] and iPAR [101]. As synthetic traffic, we use uniform and hotspot traffic patterns. For realistic applications traffic, we consider traces of selected application suites extracted from benchmark suite E3S [22]. For all experiments of simulation, we consider 7×7 mesh topology. The traffic sources are configured to generate packets of size 8 flits. The input-channel buffer size for each virtual channel is set to 6 flits. The warm-up period for the simulator is set to $15,000$ clock cycles and afterward; the average performance is measured over another $85,000$ cycles. The traffic is generated over first $75,000$ cycles. Congestion threshold is set to 66% of total buffer size of the neighboring node. As communication performance parameter, we consider latency which is defined as the time difference (in clock cycles) between header flit injection from source router and tail flit reception at the destination router.

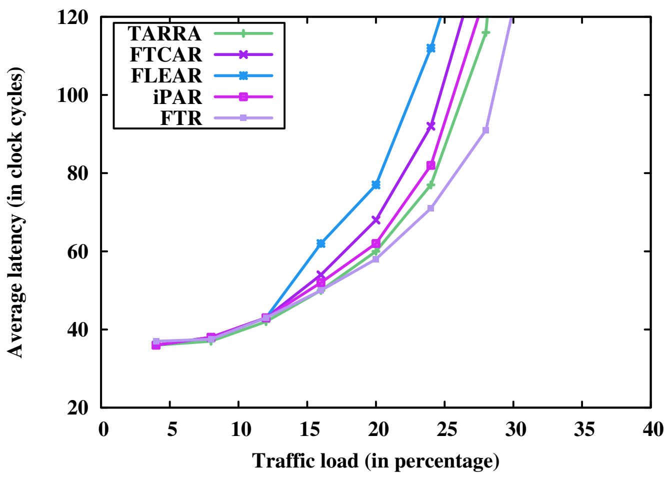


Figure 5.3: Average latency under uniform traffic

5.3.1 Uniform Traffic

Under this traffic pattern, each node sends several packets to every other node in the network with the same probability. The load-latency graph for uniform traffic model is presented in Fig. 5.3. At low traffic load, “hotspot” are not formed in the network, thus all algorithms show similar pattern in average latency. The performance degrades as the traffic load is increased. It is observed that FTR performs better than all other routing methods. Since FTR routes packets first along the X dimension and then in Y dimension, it distributes packets as evenly as possible throughout the network in the long-term for uniform traffic. However, at the fault boundary, it takes non-minimal paths unlike XY routing. In adaptive routing schemes, TARRA and iPAR perform better than FTCAR. The performance of FTCAR is slightly lower than the TARRA and iPAR. However, the FTCAR is more adaptive than TARRA and iPAR, but in the case of faults, FTCAR routes the packet using non-minimal paths that degrades its performance. Whereas, TARRA bypasses router using bypassing channels instead of taking non-minimal routes, thus performs better. In the case of faulty link, iPAR also uses non-minimal paths like FTCAR, however it performs better than FTCAR because it uses 2 VCs in X dimension. FLEAR is also adaptive routing method, but path diversity of this algorithm is less than the other adaptive schemes, thus its performance degrades.

5.3.2 Hotspot Traffic

Under this traffic pattern, we have appointed nodes (2,2), (3,4) and (4,3) as hotspot nodes with 0.3 probability of getting additional traffic. Figure 5.4 shows average latency under this model for 7×7 mesh. It can be observed that due to the fully adaptive nature and use of bypassing channels, TARRA scheme achieves better average latency than other algorithms. iPAR also performs better than FTCAR becuase of multiple resources (VCs) in X dimension unlike FTCAR. The performance of FTCAR is better than FLEAR and FTR as it can more evenly distributed traffic using additional paths.

5.3.3 Application Traffic

To evaluate the proposed method in a more realistic scenario, we select four application suites office-automation, networking, automotive/industrial and consumer from E3S benchmark suite [22]. This selection is intended to represent various applications used in the real-time embedded systems. Each task is allocated a core

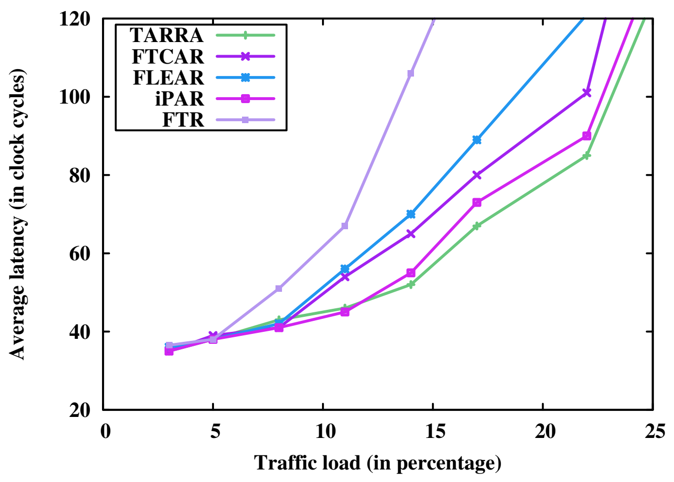


Figure 5.4: Average latency under hotspot traffic

(processor) using *minimum execution time scheduler* that executes it in the fastest time. Task mapping strongly depends on particular application traffic. We use random mapping algorithm to compute the locations of cores within NoC to enable unbiased (fair) comparisons among routing algorithms. We have executed routing algorithms repeatedly several times using random mapping and the average of simulation results are used. Figure 5.5 shows average packet latency normalized to FTR routing. TARRA provides lower latency than other methods across all four application suites. We can observe that TARRA and iPAR schemes outperforms all others. TARRA and iPAR provide path diversity along with bypassing channels and extra resources, respectively which make them better methods than others. The performance of FTCAR is better than FLEAR and FTR schemes as it provides higher path diversity than these schemes which is helpful in distributing traffic evenly.

5.3.4 Power Analysis

We integrated an existing NoC power estimation tool ORION [63] with NoC simulator [57]. It estimates total power consumption of a router into various subcomponents: input buffers, router control logic (arbiter and crossbar) traversal and channels. Figure 5.6 shows average power consumption under hotspot traffic. It can be observed that FTR method consumes less power for all traffic load. Because, it generally routes packets through minimal paths. At lower traffic loads, FTCAR performs better than FLEAR as it uses minimal paths due to small “hot spots” creation at lower traffic load. However, performance of FTCAR is sim-

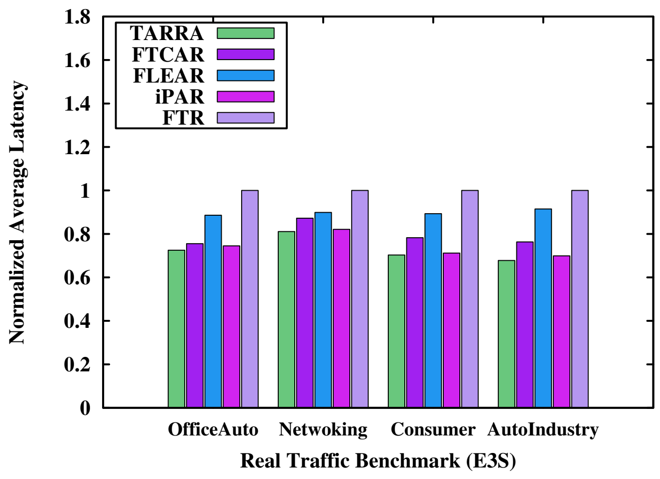


Figure 5.5: Performance for application traffic

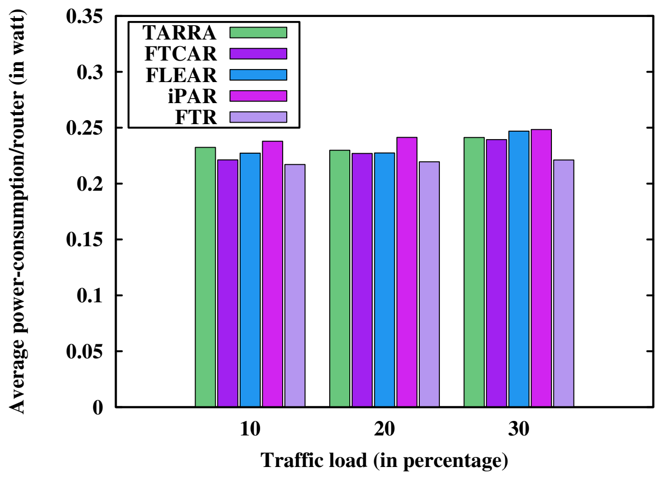


Figure 5.6: Power consumption under hotspot traffic

ilar to LEAR at higher traffic loads. TARRA and iPAR consume more power than FTCAR because of complex routing function and reconfiguration hardware (bypassing channels).

5.3.5 Area Analysis

We have implemented whole platform of each routing method to estimate the area overhead. Synthesis is performed using Synopsys Design Compiler. We use UMC 90nm technology with an operating frequency of 1 GHz and supply voltage of 1 V. Table 5.1 shows the layout area for different routing methods. We observe that TARRA and iPAR have larger area overheads than FTCAR due to bypassing logic of TARRA and extra virtual channel used by iPAR. Because of comparatively easier implementation, area overhead of FTR is lesser than any other method.

Table 5.1: Area Requirement

Algorithm	**Area**(mm^2)
TARRA	8.163
FTCAR	7.757
FLEAR	7.913
iPAR	8.132
FTR	6.963

5.4 Inferences

In this chapter, we have proposed a fault-tolerant and congestion-aware routing algorithm (FTCAR) to mitigate congestion. It can tolerate all single-link failures. It can also handle multiple-links failures depending upon the distance of faults. FTCAR provides higher degree of adaptiveness by allowing cycles in channel dependency graph. It can use minimal or non-minimal routes between source and destination nodes depending on congestion and fault status. Deadlock freedom of FTCAR is ensured using Duato's theory. In the next chapter, we present a reconfigurable and fault tolerant routing using LBDR.

Chapter 6

Reconfigurable Distributed Fault-Tolerant Routing Algorithm

6.1 Overview

Faults can be tolerated by using many methods and majority of them are based on fault tolerant routing algorithms. Routing algorithms can be implemented as either table-based or algorithmic routing [20, 26]. In algorithmic routing mechanism, an algorithm is executed using software or special hardware circuits (generally combinational logic) employing FSM to compute appropriate router port. It is generally suitable for one topology and one routing method for that topology. The algorithmic mechanism is often efficient in the terms of speed and area than table-based routing [20]. Table-based mechanism is used to deal with regular as well as irregular topologies. In this mechanism, each router keeps a table to store one output port channel (deterministic routing) for each destination from the current node in the network. The benefit of this mechanism is that it supports any topology and any routing method, even fault-tolerant and congestion aware routing algorithms. The table based methods cannot scale well since the table size increases with the size of the network and may become impractical for large NoCs.

In the application-specific platforms where communication transactions (concurrent/non-concurrent) among IP cores are known in advance, it is quite possible to use compression techniques [82] to reduce size of tables instead of straight forward table-based implementation. However, it is not a generic scenario in multi-core systems.

In [102], a fault-tolerant routing technique is introduced that can tolerate only single-node failures in mesh-based NoCs. It cannot handle link failures. Boppana and Chalasani [8] proposed a deterministic fault-tolerant routing algorithm that can tolerate faulty nodes and links. It uses additional VCs to avoid deadlocks, thus, results in area overhead. Masoumeh Ebrahimi et al. [35] presented a minimal and defect-resilient routing method to route packets in the presence of one faulty link using two VCs. David Fick et al. [38] proposed a table based implementation of the routing algorithm to handle faulty components. In this chapter, we present a reconfigurable, deadlock free and cost-efficient fault-tolerant routing algorithm without using VCs.

6.1.1 LBDR Overview

The new routing methodology LBDR [39] has the potential of handling practical irregular topologies derived from the initial 2D mesh without using routing tables. It can realize almost all deadlock free routing techniques efficiently. Deadlock freedom of any routing technique can be ensured by enforcing some routing restrictions in LBDR implementation. A routing restriction at a router prohibits packet to take the turn at next hop router during its travel from source to destination. For example, Fig.6.1 shows the routing restrictions corresponding to xy routing algorithm for 3×3 mesh.

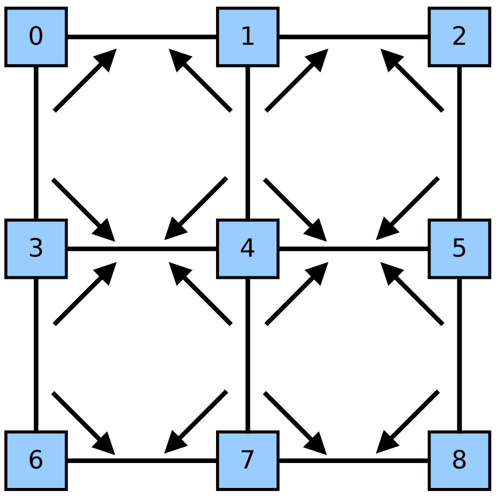


Figure 6.1: Routing restrictions for xy Routing for 3×3 Mesh

In LBDR, each router is designed to keep two different sets of bits.

1. Routing Bits: Routing bits stand for the routing restrictions which are forced by underlying routing technique. These are used to afford one hop visibility from the current router and restrict a packet to take few turns. For example, a routing bit (R_{xy}) states that a packet departing from the x direction of the current router can take a turn towards y direction on next router or not. If R_{xy} bit is set, it means that a packet can take the specified turn on next router.

2. Connectivity Bits: Connectivity bits stand for the connectivity of the current router within the current network topology. Connectivity bits are designated with C_x, to define the connectivity at the x output port of the router. If C_x is set, it means that the current router has connectivity with the neighbor in the x direction.

For 2D mesh topology, it uses total two routing bits and one connectivity bit for each router output port as shown in Fig. 6.2.

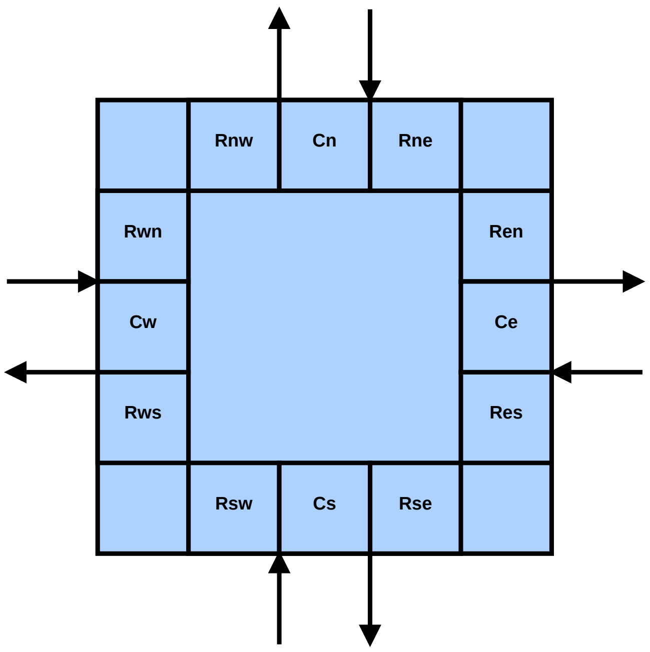


Figure 6.2: LBDR Router

These bits are first computed off-line for a particular combination of topology and routing technique. Table 6.1 shows the routing and connectivity bits corresponding to *xy* routing restrictions as shown in Fig. 6.1.

Table 6.1: Routing and connectivity bits for LBDR implementation of xy routing for 3×3 mesh

Router ID	**Rne**	**Rnw**	**Ren**	**Res**	**Rse**	**Rsw**	**Rwn**	**Rws**	**Cn**	**Ce**	**Cw**	**Cs**
0	1	1	1	1	0	1	1	1	0	1	0	1
1	1	1	1	1	0	0	1	1	0	1	1	1
2	1	1	1	1	1	0	1	1	0	0	1	1
3	0	1	1	1	0	1	1	1	1	1	0	1
4	0	0	1	1	0	0	1	1	1	1	1	1
5	1	0	1	1	1	0	1	1	1	0	1	1
6	0	1	1	1	1	1	1	1	1	1	0	0
7	0	0	1	1	1	1	1	1	1	1	1	0
8	1	0	1	1	1	1	1	1	1	0	1	0

6.1.2 Next Hop Computation Logic

This implementation selects North as next hop ($nextHop$), if current router has connectivity in North (C_n=1) and if

1. destination is in the North direction, or
2. destination is North East quadrant and R_{ne}=1, or
3. destination is North West quadrant and R_{nw}=1

6.2 Proposed Method

With the advances in the deep sub-micron technology of VLSI, the probability of failing of nodes or links has also increased. Therefore topologies which are derived from 2D mesh having some manufacturing defects or faults come into existence. When a link fails in 2D mesh, the resulting irregular topology may not have minimal paths between some pair of nodes. In this section, we present a deadlock free, reconfigurable, fault-tolerant and deterministic routing algorithm which is designed to handle one or multiple single link faults within 2D NoC mesh topology. We modify and extend LBDR to implement our proposed method.

6.2.1 Link Failure Classification and Deadlock Freedom

We adopt *xy* routing (dimension order routing (DOR)) as the basis of our work. It is deterministic and shortest-path routing, according to [18]. In this routing technique, a packet is first forwarded into the x dimension. When it arrives at the proper coordinate in the x dimension, it is forwarded to the y dimension. One of the important attribute of *xy* routing is that it always picks a unique path between any pair of source and destination.

NoC topologies which adopt DOR get freedom from deadlock by ordering channels in such a way that packets always travel along paths of strictly increasing (or decreasing) ordered channel numbers [18]. In this method, all of the channels in second (y) dimension are labeled with greater numbers than all of the channels in the first (x) dimension. This strict ordering eliminates all cycles in the channel dependency graph (CDG) and thus avoids deadlock. This ordering, however, consequences in a singular path from source to destination and thus does not allow any adaptiveness.

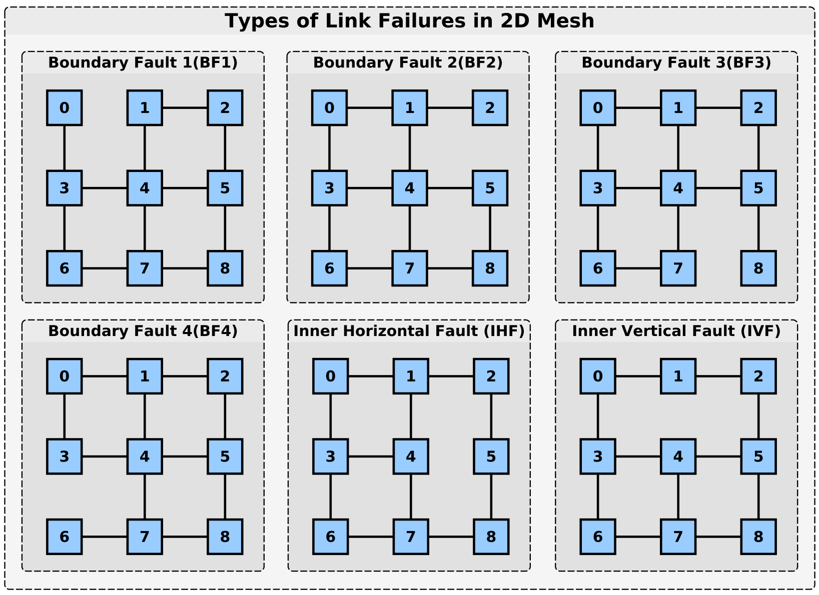


Figure 6.3: Types of link failure in the 2D mesh

We can classify link failures into six different categories depending upon their locations in the mesh as shown in Fig. 6.3. According to Dally [18], Channel Dependency Graph (CDG) can be used to ensure deadlock freedom in the case of boundary faults (BF1, BF2, BF3 and BF4). CDG is a graph in which nodes and edges are network channels and dependencies between network channels, respectively. By constructing CDG for the boundary of BF1 as shown in Fig. 6.4a, we can see that it does not have any cycle. Similarly, we can prove that boundaries of faults BF2, BF3 and BF4 will not have any cycle in their CDGs. Thus, these all boundaries are deadlock free.

However, when we construct CDG for boundary involved in IHF, it contains cycles as shown in Fig. 6.4b. Hence, the boundary of IHF is not deadlock free. To break cycles in IHF's CDG, we have used *new* routing restrictions as shown in Fig. 6.4c. These restrictions prohibit a packet from taking EN and SW turns in IHF boundary. Similarly, we can prove that the boundary of IVF with *new* routing restrictions is also cycle free as shown in Fig. 6.4d, thus, deadlock free. We have used *xy* and *new* routing restrictions in non-faulty and faulty regions, respectively, to prevent deadlock in our proposed method.

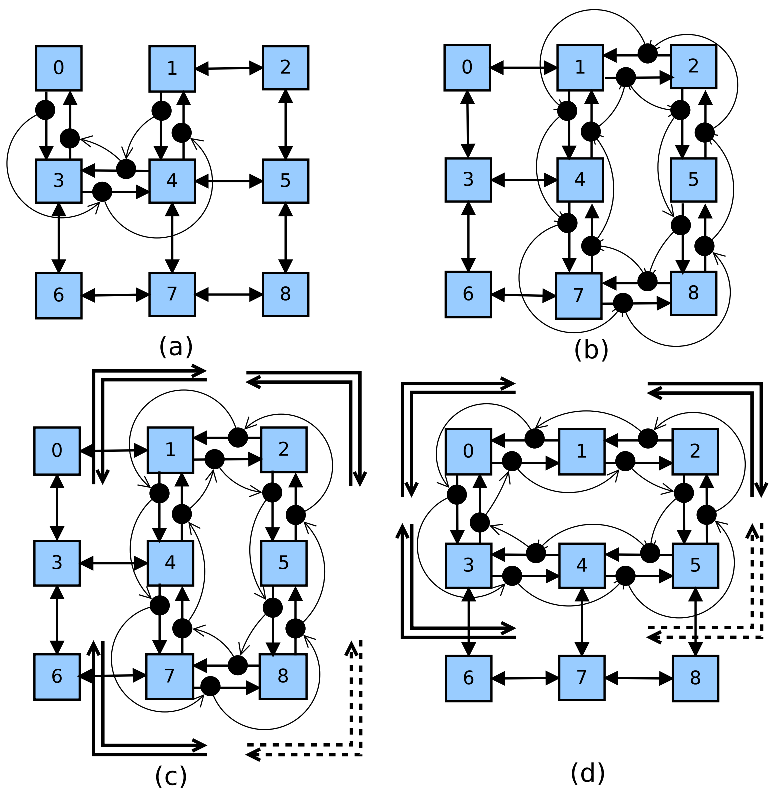


Figure 6.4: Channel dependency graphs for (a) boundary of BF1 (b) boundary of IHF (c) deadlock avoidance in IHF (d) deadlock avoidance in IVF (dashed line shows restricted turn)

6.2.2 Reconfiguration of Routing Paths

In the absence of link failures, proposed method works as deterministic *xy* routing. However, if a fault occurs, *xy* can not find paths between certain pairs of nodes.

In the case of single link failure, when boundary faults (BF1, BF2, BF3 and BF4) occur, they do not affect the total available shortest paths within the current irregular topology. However, in the case of inner faults (IHF and IVF), they may affect a few available shortest paths as discussed below.

- If a packet arrives at a node having an IHF in the East direction (say node 4) and intends for a faulty channel, it always follows the available minimal path.
- If a packet arrives at a node having an IHF in West direction (say node 5) and intends for faulty channel, and
  - If the destination is in the same row, it follows available minimal path.
  - If the destination is in NW quadrant, it follows the available minimal path.
  - If the destination is in SW quadrant, it follows the non-minimal path.
- If a packet arrives at a node (say node 7) vertically below the node having an IHFs in the East direction (say node 4) and
  - If the destination is in same row or column, it follows the minimal path.
  - If the destination is two hop away on the fault boundary and is in NW quadrant (say node 5), it follows the non-minimal path.
  - If the destination is three hop away on the fault boundary and is in NW quadrant (say node 2), it follows available minimal path.

In all case other cases of IHF boundary, packet always follows minimal and same path as in *xy* routing.

Table 6.2 and Table 6.3 show reconfigured paths in IHF and IVF boundaries, respectively, according to new routing restrictions. Similarly, we can reconfigure paths for remaining boundaries of faults (BF1, BF2, BF3 and BF4). We define a reconfiguration method for all routers that are part of any boundary of fault. This reconfiguration method is activated as soon as a built-in self-test (BIST) unit detects a fault. We assume that faults are detected by BIST unit. After fault detection, we change the routing and connectivity bits for concerned routers according to above discussion.

6.2.3 Fault Tolerant Routing Implementation

We propose a deterministic fault-tolerant routing algorithm based on *xy* routing for the 2D mesh. LBDR [39] implementation does not support irregular topologies that contain non-minimal paths (previously minimal in regular 2D mesh). Rodrigo *et al* [88] proposed a method LBDRdr that uses Deroute option in LBDR to support such cases. Since Deroute option results in extra bits (8) per router, we eliminate it and provide a new Misroute option without using additional bits.

Our routing implementation is divided into the following three steps.

1. Compute Relative Position of Destination Router: Using current router (X_c, Y_c) and destination router (X_d, Y_d) coordinates, we compute relative position of destination by invoking Algorithm 5. We use two variables (d_1 and d_2) to encode it. If d_1 is set to $NULL$ and d_2 is set to $WEST$, it means that destination is in West direction. If d_1 and d_2 both are $NULL$, it means destination is Local IP Core.

2. Invoke Reconfiguration Method on Occurrence of Fault: If BIST detects any fault, we invoke reconfiguration method (described in Section 6.2.2) to reset routing and connectivity bits.

3. Next Hop Computation: We compute next hop (*nextHop*) using LBDR. If it is unable to compute next hop (*nextHop* is set to $NULL$), we use Misroute option. Algorithm 6 describes Misroute option for IHF boundary. It computes next hop direction only for routers (4, 5 and 7) depending on their locations on IHF boundary and input direction (*ip_dir*) of the incoming packet. We explain $nextHop$ computation with following two examples.

   (a) If router 4 receives a packet coming from West direction (*ip_dir* is WEST) and targets for router 5, LBDR does not produce any next

Table 6.2: Reconfigured paths for IHF boundary

Path No	**Old Path According to *xy* Routing in 2D Regular Mesh**	**New Path According to Proposed Method in Irregular 2D Mesh**	**New Path in Irregular 2D Mesh**	**New Path in Irregular 2D Mesh compared to Regular 2D Mesh**
1	$4 \rightarrow 5$	$4 \rightarrow 1 \rightarrow 2 \rightarrow 5$	Minimal	Non-minimal
2	$5 \rightarrow 4$	$5 \rightarrow 2 \rightarrow 1 \rightarrow 4$	Minimal	Non-minimal
3	$4 \rightarrow 5 \rightarrow 2$	$4 \rightarrow 1 \rightarrow 2$	Minimal	Minimal
4	$5 \rightarrow 4 \rightarrow 1$	$5 \rightarrow 2 \rightarrow 1$	Minimal	Minimal
5	$4 \rightarrow 5 \rightarrow 8$	$4 \rightarrow 7 \rightarrow 8$	Minimal	Minimal
6	$5 \rightarrow 4 \rightarrow 7$	$5 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 7$	Non-minimal	Non-minimal
7	$7 \rightarrow 8 \rightarrow 5$	$7 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 5$	Non-minimal	Non-minimal
8	$7 \rightarrow 8 \rightarrow 5 \rightarrow 2$	$7 \rightarrow 4 \rightarrow 1 \rightarrow 2$	Minimal	Minimal

Table 6.3: Reconfigured paths for IVF boundary

Path No	**Old Path According to *xy* Routing in 2D Regular Mesh**	**New Path According to Proposed Method in Irregular 2D Mesh**	**New Path in Irregular 2D Mesh**	**New Path in Irregular 2D Mesh compared to Regular 2D Mesh**
1	$4 \rightarrow 1$	$4 \rightarrow 3 \rightarrow 0 \rightarrow 1$	Minimal	Non-minimal
2	$1 \rightarrow 4$	$1 \rightarrow 0 \rightarrow 3 \rightarrow 4$	Minimal	Non-minimal
3	$2 \rightarrow 1 \rightarrow 4$	$2 \rightarrow 1 \rightarrow 0 \rightarrow 3 \rightarrow 4$	Non-minimal	Non-minimal
4	$5 \rightarrow 4 \rightarrow 1$	$5 \rightarrow 2 \rightarrow 1$	Minimal	Minimal
5	$0 \rightarrow 1 \rightarrow 4$	$0 \rightarrow 3 \rightarrow 4$	Minimal	Minimal
6	$3 \rightarrow 4 \rightarrow 1$	$3 \rightarrow 0 \rightarrow 1$	Minimal	Minimal
7	$3 \rightarrow 4 \rightarrow 5 \rightarrow 2$	$3 \rightarrow 0 \rightarrow 1 \rightarrow 2$	Minimal	Minimal
8	$4 \rightarrow 5 \rightarrow 2$	$4 \rightarrow 3 \rightarrow 0 \rightarrow 1 \rightarrow 2$	Non-minimal	Non-minimal

Algorithm 5 Relative Position of Destination Router

```
$(X_c, Y_c)$ : Current router coordinates
$(X_d, Y_d)$ : Destination router coordinates
$d1$ and $d2$ : Output directions denoting relative position
$\triangle x = (X_d - X_c)$
$\triangle y = (Y_d - Y_c)$
if $(\triangle x > 0)$ then
  if $(\triangle y > 0)$ then
    $d_1$ = SOUTH $d_2$ = EAST
    /* destination is in SE quadrant */
  else if $(\triangle y < 0)$ then
    $d_1$ = NORTH $d_2$ = EAST
  else
    $d_1$ = NULL $d_2$ = EAST
    /* destination is in EAST direction */
  end if
else if $(\triangle x < 0)$ then
  if $(\triangle y > 0)$ then
    $d_1$ = SOUTH $d_2$ = WEST
  else if $(\triangle y < 0)$ then
    $d_1$ = NORTH $d_2$ = WEST
  else
    $d_1$ = NULL $d_2$ = WEST
  end if
else if $(\triangle x == 0)$ then
  if $(\triangle y > 0)$ then
    $d_1$ = SOUTH $d_2$ = NULL
  else if $(\triangle y < 0)$ then
    $d_1$ = NORTH $d_2$ = NULL
  else
    $d_1$ = NULL $d_2$ = NULL
    /* destination is Local IP core */
  end if
end if
```

hop (*nextHop* is set to *NULL*). Because there is no connectivity between router 4 and router 5 (C_e is 0). Thus, we compute North as next hop using Misroute option

(b) If router 7 receives a packet coming from West direction (*ip_dir* is WEST) and targets for router 5, LBDR does not produce any next hop (*nextHop* is set to *NULL*). Although router 7 is having connectivity toward C_e and C_n, but routing restrictions R_{en} (North turn on next router in East direction) and R_{ne} (East turn on next router in North direction) are set to zero by reconfiguration method. Thus, we compute North as next hop using Misroute option because destination router is NE quadrant.

Since Misroute option for IHF always redirects a packet in North direction only, it does not require additional bits. Similarly, we can compute West as next hop direction for routers (1, 2 and 4) of IVF boundary using Misroute option.

6.3 Results Analysis

In order to perform simulation work, we extended [57], a cycle-accurate SystemC based NoC simulator. We have evaluated proposed routing method (DRFT) for both real (benchmark suite E3S [22]) and synthetic (uniform and hotspot) traffic profiles. We have also implemented other routing methods FTR [61], ZoneDefense (ZDF) [42] and FTCAR [66] to evaluate DRFT. For all experiments of simulation, we consider 7×7 mesh topology. The traffic sources are configured to generate packets of size 8 flits. The input-channel buffer size for each virtual channel is set to 6 flits. The warm-up period for the simulator is set to $15,000$ clock cycles and afterward; the average performance is measured over another $85,000$ cycles. The traffic is generated over first $75,000$ cycles. Congestion threshold is set to 66% of total buffer size of the neighboring node. As communication performance parameter, we consider latency which is defined as the time difference (in clock cycles) between header flit injection from source router and tail flit reception at the destination router.

Algorithm 6 : Next Hop Computation on IHF Boundary using Misroute Option

```
Input: $ip\_dir$, $d_1$, $d_2$
Output: $nextHop$
Procedure:
if ($nextHop$ == NULL) then
  /* Compute next hop for Node 4 of IHF Boundary */
  if ($ip\_dir$ == WEST || $ip\_dir$ == SOUTH) then
    if ($d_1$ == NULL && $d_2$ == WEST) then
      $nextHop$ = NORTH
    end if
  end if
  /* Compute next hop for Node 5 of IHF Boundary */
  if ($ip\_dir$ == EAST) then
    if ($d_2$ == WEST) then
      if ($d_1$ == NULL || $d_1$ == SOUTH) then
        $nextHop$ = NORTH
      end if
    end if
  end if
  /* Compute next hop for Node 7 of IHF Boundary */
  if ($ip\_dir$ == WEST) then
    if ($d_1$ == NORTH && $d_2$ == EAST) then
      $nextHop$ = NORTH
    end if
  end if
end if
```

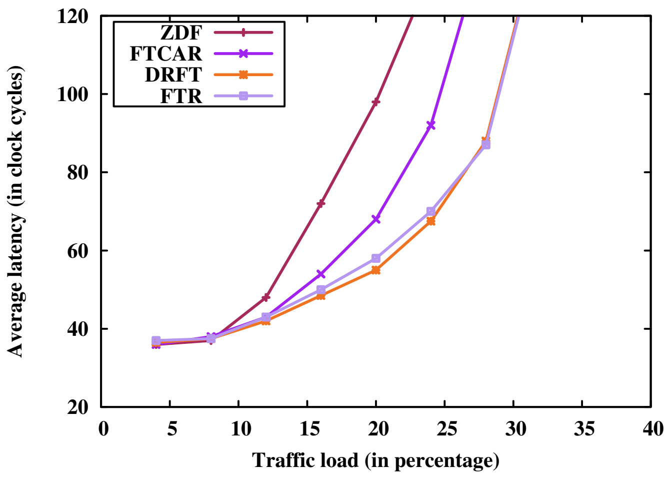


Figure 6.5: Average latency under uniform traffic

6.3.1 Uniform Traffic

Under this traffic pattern, each node sends several packets to every other node in the network with the same probability. The load-latency graph for uniform

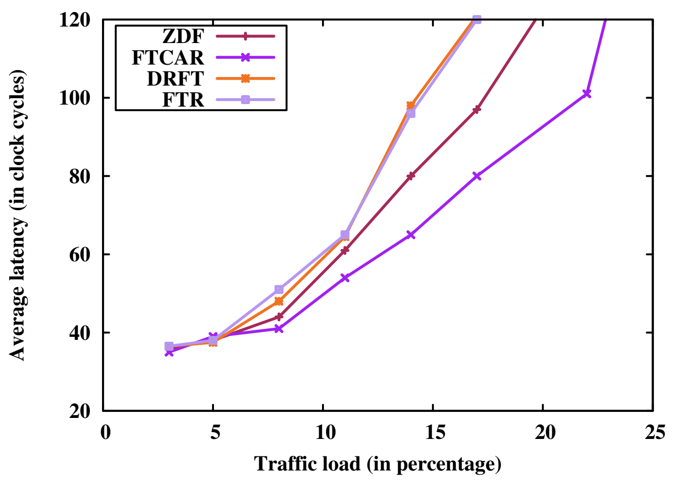


Figure 6.6: Average latency under hotspot traffic

traffic model is presented in Fig. 6.5. At low traffic load, all algorithms exhibit similar pattern in average latency. The performance degrades as the traffic load is increased. DRFT outperforms all other routing methods. For uniform traffic, DRFT evenly distributes packets throughout the network in the long-term because it routes packets in dimension order (first in X dimension and then in Y dimension). However, at the fault boundary, it uses non-minimal routes unlike xy routing. When there is no fault in the network, DRFT is equivalent to xy routing. The performance of FTR is also similar to the DRFT as both schemes are proposed on xy routing and uses reconfiguration of paths in the case of faults.

6.3.2 Hotspot Traffic

Under this traffic pattern, we have appointed nodes (2,2), (3,4) and (4,3) as hotspot nodes with 0.3 probability of getting additional traffic. Figure 6.6 shows average latency under this model for 7×7 mesh. Adaptive algorithms (FTCAR and ZDF) can direct packets to different output channels which are destined for the same destination. It can also be observed that due to the fully adaptive nature, FTCAR scheme outperforms other algorithms. The performance of FTCAR is better than FTR and ZDF schemes as it can more evenly distributed traffic using additional paths than these algorithms. We observe that DRFT and FTR has a higher average latency than other algorithms. Because, when multiple traffic flows are oriented towards a small subset of “hot spot” nodes, these non-adaptive routers will be compelled to forward them towards the same output direction, thus saturating the virtual channel queues.

6.3.3 Application Traffic

To evaluate the proposed method in a more realistic scenario, we select four application suites office-automation, networking, automotive/industrial and consumer from E3S benchmark suite [22]. This selection is intended to represent various applications used in the real-time embedded systems. Each task is allocated a core (processor) using *minimum execution time scheduler* that executes it in the fastest time. Task mapping strongly depends on particular application traffic. Figure 6.7 shows average packet latency normalized to FTR routing. FTCAR provides lower latency than other methods across all four application suites. The performance of FTCAR is better than FTR, ZDF and DRFT schemes as it provides higher path diversity than these schemes which is helpful in distributing traffic evenly. However, performance of DRFT and FTR is almost same as both are based on *xy* routing and use path reconfiguration logic when faults occur.

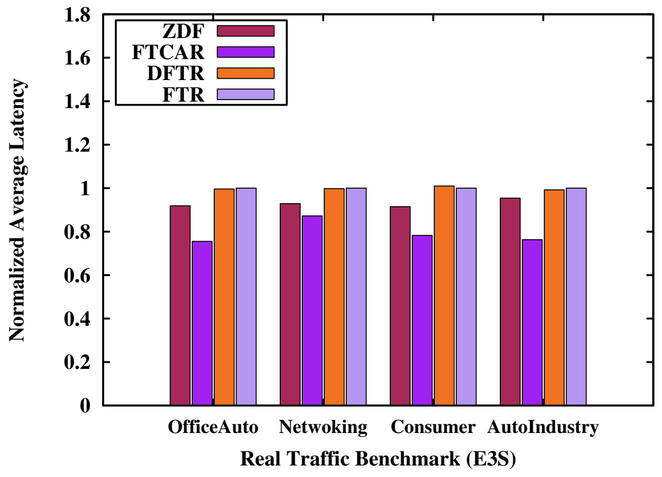


Figure 6.7: Performance for application traffic

6.3.4 Power Analysis

We integrated an existing NoC power estimation tool ORION [63] with NoC simulator [57]. It estimates total power consumption of a router into various subcomponents: input buffers, router control logic (arbiter and crossbar) traversal and channels. Figure 6.8 shows average power consumption under hotspot traffic. It can be observed that DRFT and FTR methods consume less power for all traffic load. Because, it generally routes packets through minimal paths and their

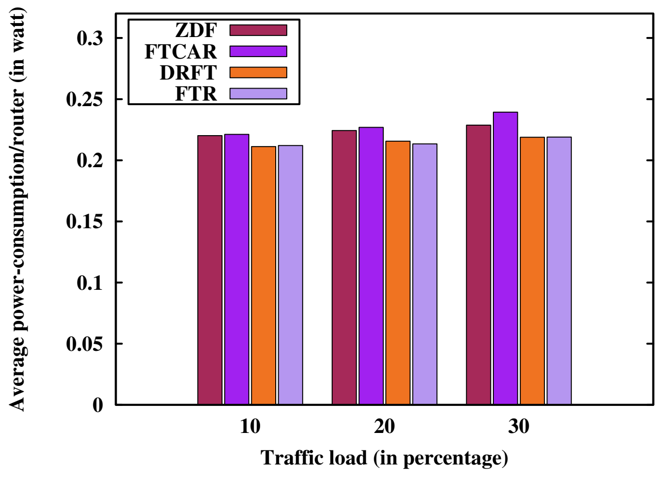


Figure 6.8: Power consumption under hotspot traffic

routing function is also deterministic. FTCAR consumes highest power because of fully adaptive nature and using non-minimal paths.

6.3.5 Area Analysis

We have implemented whole platform of each routing method to estimate the area overhead. All methods are synthesized using Synopsys Design Compiler (UMC 90nm technology) with an operating frequency of 1 GHz and supply voltage of 1 V. Table 6.4 shows the layout area for different routing methods. Because of comparatively easier implementation, area overhead of DFTR is lesser than any other method. We observe that FTCAR and ZDF have larger area overheads than DRFT due to complex routing logic.

Table 6.4: Area Requirement

Algorithm	**Area(mm^2)**
ZDF	7.234
FTCAR	7.757
DRFT	6.893
FTR	6.963

6.4 Inferences

Table based implementation of distributed routing can be costlier in terms of silicon area and power consumption as compared to combinational logic-based implementation for large irregular NoCs. In this chapter, we have proposed an implementation of cost-efficient, reconfigurable and fault tolerant *xy* routing algorithm for large irregular 2D mesh without routing tables. With the help of routing and connectivity bits, we can omit table-based implementation. Our algorithm is designed to tolerate one or multiple single link faults within the 2D mesh. Algorithm uses a few non-minimal paths (reconfigured paths) which improve reliability and fault tolerance of NoC communication. We have used *xy* and *new* routing restrictions in non-faulty and faulty regions, respectively, to achieve deadlock freedom without overhead. In future, we would like to extend the current LBDR implementation of DRFT so that it can handle multiple faults as well. However, current implementation of DRFT can handle single link fault.

Chapter 7

Conclusions and Future Directions

The motivation behind the use of adaptive routing methods in NoCs has two main purposes: (i) to divert the packets from the congested ("hot-spot") regions, and (ii) to bypass the faulty links and/or nodes of the network. The degree of adaptiveness has a major impact on the performance of an adaptive routing method. At the same time, minimality of a routing algorithm also affects the overall NoC performance. Minimal routing scheme provides the shortest paths between the source and destination. However, it would be unwise to neglect the promising performance achievable by non-minimal routing. The non-minimal route can be a good (or the only) choice if the minimal routes are congested (or faulty). The degree of adaptiveness provided by the minimal routing algorithms is also low, even if they accurately detect the state of congestion.

This research work presents a novel turn model based routing methods that provide high degree of adaptiveness for 2D mesh for both minimal and non-minimal routing. We have also shown that with careful design and relaxation of some routing turns make the proposed algorithm fault tolerant as well. The result is that the proposed method reduces restrictions on the routing turns significantly and hence can provide path diversity using additional routes (both minimal and non-minimal). This increase path diversity can be used avoid hotspot regions of the network.

The main conclusions of our research work are summarized as follows:

- We have presented a novel turn model for highly adaptive routing for 2D mesh topology. The proposed method is implemented using the double-y network. It uses one and two virtual channels along X and Y dimensions, respectively. The deadlock freedom is achieved using Duato's well-known theorem.

- We have extended the proposed 2D turn model for 3D as well. For 3D, it utilizes one, two and two virtual channels in X, Y, and Z dimensions, respectively.

- Fault tolerance is another aspect of any routing algorithm. We have presented a fault tolerant and highly adaptive routing method for the 2D mesh. The proposed turn model reduces the number of restrictions on routing turns, hence able to provide path diversity through additional minimal and non-minimal routes between source and destination. The proposed algorithm can handle all single link faults within the 2D mesh. It can also handle multiple link faults if fault's boundary do not overlap with each other.

- We have proposed a cost-effective fault-tolerant routing algorithm for irregular 2D mesh without the use of routing tables. We use one hop visibility of Logic Based Distributed Routing (LBDR) to eliminate routing tables. This algorithm handles one or multiple single link faults within the 2D mesh and uses reconfigured paths (minimal and/or non-minimal) if links fail. We use turn model based approach to avoid deadlocks. Since our method does not require virtual channels to achieve deadlock freedom, it remains area and power efficient.

In continuation of the above contributions, some of our future research plans are as follows:

- The performance of any adaptive routing scheme depends on two functions: routes computation function and selection function. Working of routes computation function is tightly coupled with selection function of the algorithm. Thus, both affect the overall performance of the network. This thesis has proposed efficient route computation function. However, the proposed algorithms also utilize congestion-aware channel selection policy using local congestion scenario. But, for a large mesh network, local congestion-awareness sometimes cannot detect congestion status of the network precisely. The global congestion-aware selection policy can improve the performance of the routing scheme significantly, specially for large mesh networks. Thus, the future work includes a development of global congestion-aware selection policies for improving overall performance.

- Our proposed fault tolerant routing is capable of handling all single link faults within the 2D mesh. Handling of double link faults is another objective for our future work. The proposed fault-tolerant routing algorithm can handle link faults, thus we plan to extend it for node failures as well. Another aspect that can be explored is the 3D extension of fault-tolerant routing.

List of Contributions

A. Journal Publications

[J-1] **Manoj Kumar**, Vijay Laxmi, Manoj Singh Gaur, Masoud Daneshtalab, Mark Zwolinski and Seok-Bum Ko, “Improved Adaptive Routing for Networks-on-Chip”, in Electronics Letters, vol. 51, no. 25, pp. 2092-2094, 12-10-2015.

B. Conference Publications

[C-1] **Manoj Kumar**, Vijay Laxmi, Manoj Singh Gaur, Masoud Daneshtalab and Mark Zwolinski, “Fault tolerant and Highly Adaptive Routing for 2D NoCs”, in Proceedings of 27^{th} IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Amsterdam, Netherlands, Oct-2014.

[C-2] **Manoj Kumar**, Vijay Laxmi, Manoj Singh Gaur, Masoud Daneshtalab, Mosoumeh Ebrahimi and Mark Zwolinski, “A Non-minimal Turn Model for Highly Adaptive Routing in 2D NoCs”, in Proceedings of 22^{nd} IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Playa Del Carmen, Mexico, Oct-2014.

[C-3] **Manoj Kumar**, Vijay Laxmi, Manoj Singh Gaur, Masoud Daneshtalab, Seok-Bum Ko and Mark Zwolinski, “A Novel Non-minimal/minimal Turn Model for Highly Adaptive Routing in 2D NoCs”, in Proceedings of 8^{th} IEEE/ACM International Symposium on NoCs (NOCS), Ferrara, Italy, Sep-2014.

[C-4] **Manoj Kumar**, Vijay Laxmi, Manoj Singh Gaur, Masoud Daneshtalab, Seok-Bum Ko and Mark Zwolinski, “Highly Adaptive and Congestion-aware Routing for 3D NoCs”,in Proceedings of 24^{th} ACM International Grate Lakes Symposium on VLSI (GLSVLSI), Houston, USA, May-2014.

[C-5] **Manoj Kumar**, Vijay Laxmi, Manoj Singh Gaur, Seok-Bum Ko and Mark Zwolinski, "CARM:Congestion Adaptive Routing Method for On Chip Networks",in Proceedings of 27th IEEE International Conference on VLSI Design (VLSI-D), Mumbai, India, Jan-2014.

[C-6] **Manoj Kumar**, Pankaj, Vijay Laxmi, Manoj Singh Gaur and Seok-Bum Ko, "Reconfigurable Distributed Fault Tolerant Routing Algorithm for On-Chip Networks", in Proceedings of 26th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), New York, USA, Sep-2013.

[C-7] Niyati Gupta, **Manoj Kumar**, Vijay Laxmi, Manoj Singh Gaur and Mark Zwolinski, "σLBDR: Congestion-aware Logic Based Distributed Routing for 2D NoCs ", in Proceedings of 19th IEEE International Symposium on VLSI Design and Test (VDAT), Ahmadabad, India, June 2015.

[C-8] Niyati Gupta, **Manoj Kumar**, Vijay Laxmi, Manoj Singh Gaur, Masoud Daneshtalab and Masoumeh Ebrahimi, "Improved Route Selection Approaches using Q-learning framework for 2D NoCs", in Proceedings of 3rd ACM International Workshop on Many Core Embedded Systems (MES), June 2015.

Bibliography

[1] J. D. Allen, P. T. Gaughan, D. E. Schimmel, and S. Yalamanchili. Ariadne&mdash;an adaptive router for fault-tolerant multicomputers. *ACM SIGARCH Computer Architecture News*, 22(2):278–288, Apr. 1994.

[2] J. Andrews and N. Baker. Xbox 360 system architecture. *IEEE Micro*, 26(2):25–37, March 2006.

[3] T. Bartic, J.-Y. Mignolet, V. Nollet, T. Marescaux, D. Verkest, S. Vernalde, and R. Lauwereins. Topology adaptive network-on-chip design and implementation. *IEE Proceedings-Computers and Digital Techniques*, 152(4):467–472, 2005.

[4] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, et al. Tile64-processor: A 64-core soc with mesh interconnect. In *2008 IEEE International Solid-State Circuits Conference-Digest of Technical Papers*, pages 88–598. IEEE, 2008.

[5] L. Benini and G. De Micheli. Networks on chip: a new paradigm for systems on chip design. In *Design, Automation and Test in Europe Conference and Exhibition, Proceedings*, pages 418–419. IEEE, 2002.

[6] T. Bjerregaard and J. Sparso. Implementation of guaranteed services in the mango clockless network-on-chip. *IEE Proceedings-Computers and Digital Techniques*, 153(4):217–229, 2006.

[7] F. Bodin, D. Windheiser, W. Jalby, D. Atapattu, M. Lee, and D. Gannon. Performance evaluation and prediction for parallel algorithms on the bbn gp1000. In *Proceedings of the 4th International Conference on Supercomputing*, ICS '90, pages 401–413. ACM, 1990.

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

[9] Y. M. Boura and C. R. Das. Efficient fully adaptive wormhole routing in n-dimensional meshes. In *Distributed computing systems, 1994., proceedings of the 14th international conference on*, pages 589–596. IEEE, 1994.

[10] F. Chaix, D. Avresky, N.-E. Zergainoh, and M. Nicolaidis. A fault-tolerant deadlock-free adaptive routing for on chip interconnects. In *Proceedings of 14th Design, Automation Test in Europe Conference Exhibition*, pages 1–4, 2011.

[11] E. J. Chang, H. K. Hsin, S. Y. Lin, and A. Y. Wu. Path-congestion-aware adaptive routing with a contention prediction scheme for network-on-chip systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(1):113–126, Jan 2014.

[12] A. A. Chien and J. H. Kim. Planar-adaptive routing: low-cost adaptive networks for multiprocessors. *Journal of the ACM*, 42(1):91–123, 1995.

[13] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):729–738, 2000.

[14] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi. *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*. CRC Press, Inc., 1st edition, 2008.

[15] N. Dahir, T. Mak, A. Yakovlev, et al. Highly adaptive and deadlock-free routing for three-dimensional networks-on-chip. *IET Computers & Digital Techniques*, 7(6):255–263, 2013.

[16] W. J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, 1992.

[17] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 100(5):547–553, 1987.

[18] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 100(5):547–553, 1987.

[19] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684–689. IEEE, 2001.

[20] W. J. Dally and B. P. Towles. *Principles and Practices of Interconnection Networks.* Morgan Kaufmann, 2004.

[21] B. V. Dao, J. Duato, and S. Yalamanchili. Configurable flow control mechanisms for fault-tolerant routing. *SIGARCH Comput. Archit. News*, 23(2):220–229, 1995.

[22] R. Dick. E3S: Embedded system synthesis benchmarks suite. http://ziyang.eecs.umich.edu/∼dickrp/e3s/.

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

[24] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1055–1067, 1995.

[25] J. Duato, B. Dao, P. T. Gaughan, and S. Yalamanchili. Scouting: Fully adaptive, deadlock-free routing in faulty pipelined networks. In *Parallel and Distributed Systems, 1994. International Conference on*, pages 608–613. IEEE, 1994.

[26] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks - An Engineering Approach.* Morgan Kaufmann, 2003.

[27] F. Dubois, A. Sheibanyrad, F. Ptrot, and M. Bahmani. Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3d-nocs. *IEEE Transactions on Computers*, 62(3):609–615, March 2013.

[28] M. Ebrahimi. Fully adaptive routing algorithms and region-based approaches for two-dimensional and three-dimensional networks-on-chip. *IET Computers & Digital Techniques*, 7(6):264–273, 2013.

[29] M. Ebrahimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, and H. Tenhunen. Dyxyz: Fully adaptive routing algorithm for 3d nocs. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 499–503. IEEE, 2013.

[30] M. Ebrahimi and M. Daneshtalab. Learning-based routing algorithms for on-chip networks. In *Routing Algorithms in Networks-on-Chip*, pages 105–125. Springer, 2014.

[31] M. Ebrahimi and M. Daneshtalab. A light-weight fault-tolerant routing algorithm tolerating faulty links and routers. *Computing*, 97(6):631–648, 2015.

[32] M. Ebrahimi, M. Daneshtalab, F. Farahnakian, J. Plosila, P. Liljeberg, M. Palesi, and H. Tenhunen. Haraq: congestion-aware learning model for highly adaptive routing algorithm in on-chip networks. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 19–26. IEEE, 2012.

[33] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen. Catra-congestion aware trapezoid-based routing algorithm for on-chip networks. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 320–325. IEEE, 2012.

[34] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen. Lear–a low-weight and highly adaptive routing method for distributing congestions in on-chip networks. In *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 520–524. IEEE, 2012.

[35] M. Ebrahimi, M. Daneshtalab, J. Plosila, and F. Mehdipour. MD: Minimal path-based fault-tolerant routing in on-chip networks. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 35–40. IEEE, 2013.

[36] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen. Mafa: adaptive fault-tolerant routing algorithm for networks-on-chip. In *Digital System Design (DSD), 2012 15th Euromicro Conference on*, pages 201–207. IEEE, 2012.

[37] B. S. Feero and P. P. Pande. Networks-on-chip in a three-dimensional environment: A performance evaluation. *IEEE Transactions on Computers*, 58(1):32–45, Jan 2009.

[38] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw. A highly resilient routing algorithm for fault-tolerant nocs. In *Proceedings*

of the Conference on Design, Automation and Test in Europe, pages 21–26, 2009.

[39] J. Flich and J. Duato. Logic-based distributed routing for nocs. *IEEE Computer Architecture Letters*, 7(1):13–16, 2008.

[40] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho. A survey and evaluation of topology-agnostic deterministic routing algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):405–425, 2012.

[41] S. Foroutan, A. Sheibanyrad, and F. Ptrot. Assignment of vertical-links to routers in vertically-partially-connected 3-d-nocs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(8):1208–1218, Aug 2014.

[42] B. Fu, Y. Han, H. Li, and X. Li. Zonedefense: A fault-tolerant routing for 2-d meshes without virtual channels. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(1):113–126, 2014.

[43] B. Fu, Y. Han, J. Ma, H. Li, and X. Li. An abacus turn model for time/space-efficient reconfigurable routing. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 259–270. IEEE, 2011.

[44] P. T. Gaughan and S. Yalamanchili. A family of fault-tolerant routing protocols for direct multiprocessor networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(5):482–497, 1995.

[45] C. J. Glass and L. M. Ni. Maximally fully adaptive routing in 2d meshes. In *International Conference on Parallel Processing, volume I*, pages 101–104, 1992.

[46] C. J. Glass and L. M. Ni. The turn model for adaptive routing. *ACM SIGARCH Computer Architecture News*, 20(2):278–287, 1992.

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

[48] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shivakumar, S. W. Keckler, and D. Burger. On-chip interconnection networks of the trips chip. *IEEE Micro*, 27(5):41–50, 2007.

[49] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '00, pages 250–256. ACM, 2000.

[50] J. R. Gurd, C. C. Kirkham, and I. Watson. The manchester prototype dataflow computer. *Communications of the ACM*, 28(1):34–52, 1985.

[51] J. R. Herring, C. B. Stunkel, and R. Sivaram. Multicasting using a wormhole routing switching element, Apr. 1 2003. US Patent 6,542,502.

[52] C. Hilton and B. Nelson. Pnoc: a flexible circuit-switched noc for fpga-based systems. *IEE Proceedings-Computers and Digital Techniques*, 153(3):181–188, 2006.

[53] H. K. Hsin, E. J. Chang, C. A. Lin, and A. Y. . Wu. Ant colony optimization-based fault-aware routing in mesh-based network-on-chip systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(11):1693–1705, Nov 2014.

[54] H. K. Hsin, E. J. Chang, K. Y. Su, and A. Y. . Wu. Ant colony optimization-based adaptive network-on-chip routing framework using network information region. *IEEE Transactions on Computers*, 64(8):2119–2131, Aug 2015.

[55] J. Hu and R. Marculescu. Dyad: smart routing for networks-on-chip. In *Proceedings of the 41st annual Design Automation Conference*, pages 260–263. ACM, 2004.

[56] L. Huang, J. Wang, M. Ebrahimi, M. Daneshtalab, X. Zhang, G. Li, and A. Jantsch. Non-blocking testing for network-on-chip. *IEEE Transactions on Computers*, 65(3):679–692, March 2016.

[57] L. Jain, B. Al-Hashimi, M. S. Gaur, V. Laxmi, and A. Narayanan. Nirgam: A systemc based cycle accurate noc simulator, 2010. http://www.nirgam.ecs.soton.ac.uk/.

[58] C. Jesshope and P. Miller. High performance communications in processor networks. In *Computer Architecture, 1989. The 16th Annual International Symposium on*, pages 150–157, May 1989.

[59] C. R. Jesshope, P. Miller, and J. T. Yantchev. High performance communications in processor networks. *ACM SIGARCH Computer Architecture News*, 17(3):150–157, 1989.

[60] C. R. Jesshope, P. Miller, and J. T. Yantchev. High performance communications in processor networks. *ACM SIGARCH Computer Architecture News*, 17(3):150–157, Apr. 1989.

[61] S. Y. Jiang, G. Luo, Y. Liu, S. S. Jiang, and X. T. Li. Fault-tolerant routing algorithm simulation and hardware verification of noc. *IEEE Transactions on Applied Superconductivity*, 24(5):1–5, Oct 2014.

[62] J. Jose and M. Mutyam. Implementation and analysis of history-based output channel selection strategies for adaptive routers in mesh nocs. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 19(4):35, 2014.

[63] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. Orion 2.0: a fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the conference on Design, Automation and Test in Europe*, pages 423–428. European Design and Automation Association, 2009.

[64] M. KISTLER, M. PERRONE, and F. PETRINI. Cell multiprocessor communication network: Built for speed. *IEEE Micro*, 26(3), 2006.

[65] S. Konstantinidou and L. Snyder. Chaos router: Architecture and performance. In *Proceedings of the 18th Annual International Symposium on Computer Architecture*, ISCA '91, pages 212–221. ACM, 1991.

[66] M. Kumar, Pankaj, V. Laxmi, M. Gaur, and S.-B. Ko. Reconfigurable distributed fault tolerant routing algorithm for on-chip networks. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013 IEEE International Symposium on*, pages 290–295, Oct 2013.

[67] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, pages 105–112, 2002.

[68] M. Li, Q.-A. Zeng, and W.-B. Jone. Dyxy: a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *Proceedings of the 43rd annual Design Automation Conference*, pages 849–852. ACM, 2006.

[69] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on Computers*, 40(1):2–12, 1991.

[70] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, and Z. Navabi. Barp-a dynamic routing protocol for balanced distribution of traffic in nocs. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1408–1413. ACM, 2008.

[71] P. Lotfi-Kamran, A.-M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi. Edxy–a low cost congestion-aware routing algorithm for network-on-chips. *Journal of Systems Architecture*, 56(7):256–264, 2010.

[72] S. Ma, N. Jerger, and Z. Wang. Dbar: An efficient routing algorithm to support multiple concurrent applications in network-on-chip. In *Proceedings of 38th International Symposium on Computer Architecture*, pages 413–424, 2011.

[73] S. Ma, N. E. Jerger, Z. Wang, M. Lai, and L. Huang. Holistic routing algorithm design to support workload consolidation in nocs. *IEEE Transactions on Computers*, 63(3):529–542, March 2014.

[74] P. Martin. Design of a virtual component neutral network-on-chip transaction layer. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '05, pages 336–337. IEEE, 2005.

[75] A. Mejia, J. Flich, J. Duato, S.-A. Reinemo, and T. Skeie. Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, pages 10–pp. IEEE, 2006.

[76] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 890–895. IEEE, 2004.

[77] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb. The alpha 21364 network architecture. *IEEE Micro*, 22(1):26–35, Jan 2002.

[78] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[79] R. Nikhil et al. Executing a program on the mit tagged-token dataflow architecture. *IEEE Transactions on Computers*, 39(3):300–318, 1990.

[80] S. F. Nugent. The ipsc/2 direct-connect communications technology. In *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications: Architecture, Software, Computer Systems, and General Issues - Volume 1*, C3P, pages 51–60. ACM, 1988.

[81] G. Oxman and S. Weiss. An noc simulator that supports deflection routing, gpu/cpu integration, and co-simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(10):1667–1680, Oct 2016.

[82] M. Palesi, R. Holsmark, S. Kumar, and V. Catania. Application specific routing algorithms for networks on chip. *IEEE Transactions on Parallel and Distributed Systems*, 20(3):316–330, 2009.

[83] I. M. Panades, A. Greiner, A. Sheibanyrad, and G. STMicroelcctronics. A low cost network-on-chip with guaranteed service well suited to the gals approach. In *Nano-Net*, pages 1–5, 2006.

[84] V. Pavlidis and E. Friedman. 3-d topologies for networks-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(10):1081–1090, Oct 2007.

[85] L.-S. Peh and N. E. Jerger. *On-Chip Networks.* Morgan and Claypool Publishers, 1st edition, 2009.

[86] R. Ramanujam and B. Lin. Destination-based adaptive routing on 2d mesh networks. In *Proceedings of 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 1–12, 2010.

[87] E. Rijpkema, K. Goossens, A. Rădulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip. *IEE Proceedings-Computers and Digital Techniques*, 150(5):294–302, 2003.

[88] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato. Cost-efficient on-chip routing implementations for cmp and mpsoc systems. *IEEE transactions on computer-aided design of integrated circuits and systems*, 30(4):534–547, 2011.

[89] I. Saastamoinen, D. Sigüenza-Tortosa, and J. Nurmi. Interconnect ip node for future system-on-chip designs. In *Electronic Design, Test and Applications, 2002. Proceedings. The First IEEE International Workshop on*, pages 116–120. IEEE, 2002.

[90] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh. A resilient routing algorithm with formal reliability analysis for partially connected 3d-nocs. *IEEE Transactions on Computers*, PP(99):1–1, 2016.

[91] J. C. Sancho, A. Robles, and J. Duato. An effective methodology to improve the performance of the up*/down* routing algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 15(8):740–754, 2004.

[92] L. Schwiebert and D. Jayasimha. Optimal fully adaptive minimal wormhole routing for meshes. *Journal of Parallel and Distributed Computing*, 27(1):56–70, 1995.

[93] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vencentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 667–672. ACM, 2001.

[94] D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi. Issues in the development of a practical noc: the proteo concept. *Integration, the VLSI Journal*, 38(1):95–105, 2004.

[95] C. Su and K. Shin. Adaptive deadlock-free routing in multicomputers using one extra channel. In *Parallel Processing, 1993. ICPP 1993. International Conference on*, volume 1, pages 175–182, 1993.

[96] M. Tang, X. Lin, and M. Palesi. Routing pressure: A channel-related and traffic-aware metric of routing algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 26(3):891–901, March 2015.

[97] J. Upadhyay, V. Varavithya, and P. Mohapatra. A traffic-balanced adaptive wormhole routing scheme for two-dimensional meshes. *IEEE Transactions on Computers*, 46(2):190–197, 1997.

[98] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, et al. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 43(1):29–41, 2008.

[99] D. Wingard. Micronetwork-based integration for socs: 673. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 677–. ACM, 2001.

[100] D. Xiang, Z. Yu, and J. Wu. Deadlock-free fully adaptive routing in irregular networks without virtual channels. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 983–990. IEEE, 2013.

[101] D. Xiang, Y. Zhang, and Y. Pan. Practical deadlock-free fault-tolerant routing in meshes based on the planar network fault model. *IEEE Transactions on Computers*, 58(5):620–633, 2009.

[102] Z. Zhang, A. Greiner, and S. Taktak. A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip. In *Proceedings of the 45th annual Design Automation Conference*, pages 441–446. ACM, 2008.