TURUN YLIOPISTON JULKAISUJA
ANNALES UNIVERSITATIS TURKUENSIS

SARJA - SER. A I OSA - TOM. 429

ASTRONOMICA - CHEMICA - PHYSICA - MATHEMATICA

Exploring Adaptive Implementation of On-Chip Networks

by

Masoud Daneshtalab

TURUN YLIOPISTO
UNIVERSITY OF TURKU
Turku 2011

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

Supervisors

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

Reviewers

Professor José Flich
School of Engineering in Computer Science
Technical University of Valencia
Valencia, Spain

Professor Timo D. Hämäläinen
Department of Computer Systems
Technical University of Tampere
Tampere, Finland

Opponent

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

ISBN 978-951-29-4786-7 (PRINT)
ISBN 978-951-29-4787-4 (PDF)
ISSN 0082-7002
Painosalama Oy – Turku, Finland 2011

Abstract

As technology geometries have shrunk to the deep submicron regime, the communication delay and power consumption of global interconnections in high performance Multi-Processor Systems-on-Chip (MPSoCs) are becoming a major bottleneck. The Network-on-Chip (NoC) architecture paradigm, based on a modular packet-switched mechanism, can address many of the on-chip communication issues such as performance limitations of long interconnects and integration of large number of Processing Elements (PEs) on a chip. The choice of routing protocol and NoC structure can have a significant impact on performance and power consumption in on-chip networks. In addition, building a high performance, area and energy efficient on-chip network for multicore architectures requires a novel on-chip router allowing a larger network to be integrated on a single die with reduced power consumption. On top of that, network interfaces are employed to decouple computation resources from communication resources, to provide the synchronization between them, and to achieve backward compatibility with existing IP cores.

Three adaptive routing algorithms are presented as a part of this thesis. The first presented routing protocol is a congestion-aware adaptive routing algorithm for 2D mesh NoCs which does not support multicast (one-to-many) traffic while the other two protocols are adaptive routing models supporting both unicast (one-to-one) and multicast traffic. A streamlined on-chip router architecture is also presented for avoiding congested areas in 2D mesh NoCs via employing efficient input and output selection. The output selection utilizes an adaptive routing algorithm based on the congestion condition of neighboring routers while the input selection allows packets to be serviced from each input port according to its congestion level. Moreover, in order to increase memory parallelism and bring compatibility with existing IP cores in network-based multiprocessor architectures, adaptive network interface architectures are presented to use multiple SDRAMs which can be accessed simultaneously. In addition, a smart memory controller is integrated in the adaptive network interface to improve the memory utilization and reduce both memory and network latencies.

Three Dimensional Integrated Circuits (3D ICs) have been emerging as a viable candidate to achieve better performance and package density as compared to traditional 2D ICs. In addition, combining the benefits of 3D IC and NoC schemes provides a significant performance gain for 3D architectures. In recent years, inter-layer communication across multiple stacked layers (vertical channel) has attracted a lot of interest. In this thesis, a novel adaptive pipeline bus structure is proposed for inter-layer communication to improve the performance by reducing the delay and complexity of traditional bus arbitration. In addition, two mesh-based topologies for 3D architectures are also introduced to mitigate the inter-layer footprint and power dissipation on each layer with a small performance penalty.

Acknowledgments

"Life is like riding a bicycle. To keep your balance you must keep moving."
"Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world."

— Albert Einstein

The research work presented in this thesis has been carried out in the department of Information Technology, University of Turku from September 2008 to November 2011. This work would not have been possible in three years without the support of many people.

First of all, I would like to express my deepest gratitude to my supervisors, Prof. Hannu Tenhunen, Adj. Prof. Juha Plosila, and Adj. Prof. Pasi Liljeberg, for their excellent guidance, patience, and providing me with an excellent atmosphere for doing research.

I share the credit of this work with my wonderful wife, Masoumeh Ebrahimi, for being a collaborator on almost all my publications and giving the necessary comments and criticism. Moreover, without her love, encouragement, and patience over the years, I would not be able to finish this research work. She is always the first one I would go to whenever I need support, and the first one to share my happiness on my success.

It gives me great pleasure in acknowledging Prof. José Flich from Polytechnic University of Valencia and Prof. Timo D. Hämäläinen from Technical University of Tampere for the detailed reviews and the constructive comments on the manuscript.

I greatly appreciate the financial support for my doctoral studies from the Graduate School in Electronics, Telecommunication and Automation (GETA). This research work was also financially supported by the Nokia Foundation, ST-Micro, and Ulla Tuominen Foundation.

Finally, I would like to thank my parents for their constant love, support, and prayers and dedicate this thesis to them.

Turku, November 2011
Masoud Daneshtalab

List of Publications

The work presented in this thesis is based on the following publications:

Journal publications:

M. Daneshtalab, M. Ebrahimi, P. Liljeberg, Juha Plosila, and H. Tenhunen, "Memory-Efficient On-Chip Network with Adaptive Interfaces," IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems (*IEEE TCAD*) - (To appear).

M. Daneshtalab, M. Ebrahimi, T. C. Xu, P. Liljeberg, and H. Tenhunen, "A generic adaptive path-based routing method for MPSoCs," Journal of Systems Architecture *(JSA-elsevier)*, Vol. 57, No. 1, pp. 109-120, 2011.

M. Daneshtalab, M. Kamali, M. Ebrahimi, S. Mohammadi, A. Afzali-Kusha, and J. Plosila, "Adaptive Input-output Selection Based On-Chip Router Architecture," Journal of Low Power Electronics (*JOLPE*) - (To appear).

M. Daneshtalab, M. Ebrahimi, S. Mohammadi, and A. Afzali-Kusha, "Low distance path-based multicast algorithm in NOCs," Journal of the Institute of Engineering and Technology ***(****IET - Computers and Digital Techniques****)***, Special issue on NoC, Vol. 3, Issue 5, pp. 430-442, Sep 2009.

P. Lotfi-kamran, A. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, "EDXY - A Smart Congestion-Aware and Link Failure Tolerant Routing Algorithm for Network-on-Chips," Journal of Systems Architecture (*JSA-elsevier*), Vol. 56, No. 7, pp. 256-264, Jul 2010.

Conference publications:

M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Cluster-based Topologies for 3D Stacked Architectures," in Proceedings of ACM International Conference on Computing Frontiers (*CF*), pp. 1-3, May 2011, Italy.

M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "High-Performance On-Chip Network Platform for Memory-on-Processor Architectures," in Proceedings of IEEE International Symposium on Reconfigurable Communication-centric Systems-on-Chip (*ReCoSoC*), pp. 1-6, June 2011, France.

M. Daneshtalab, M. Ebrahimi, P. Liljeberg, Juha Plosila, and H. Tenhunen, "CMIT- A Novel Cluster-based Topology for 3D Stacked Architectures," in Proceedings of 2nd IEEE International 3D System Integration Conference (*3DIC*), pp. 1-5, Nov 2010, Germany.

M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Input-Output Selection Based Router for Networks-on-Chip," in Proceedings of 9th IEEE International Symposium on VLSI (*ISVLSI*), pp. 92-97, July 2010, Greece.

M. Daneshtalab, M. Ebrahimi, P. Liljeberg, Juha Plosila, and H. Tenhunen, "A Low-Latency and Memory-Efficient On-hip Network," in Proceedings of 4th IEEE/ACM International Symposium on Network-on-Chip (*NOCS*), pp. 99-106, May 2010, France.

M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "HAMUM – A Novel Routing Protocol for Unicast and Multicast Traffic in MPSoCs," in Proceedings of 18th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (*PDP*), pp. 525-532, February 2010, Italy.

M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "A High-Performance Network Interface Architecture for NoCs Using Reorder Buffer Sharing," in Proceedings of 18th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (*PDP*), pp. 547-550, February 2010, Italy.

M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Pipeline-Based Interlayer Bus Structure for 3D Networks-on-Chip," in Proceedings of 15th International Symposium on Computer Architecture and Digital Systems (*CADS*), IEEE Press, pp. 41-47, Sept 2010, Iran.

M. Ebrahimi, M. Daneshtalab, N. Sreejesh, P. Liljeberg, Juha Plosila, and H. Tenhunen, "Efficient Network Interface Architecture for Network-on-Chips," in Proceedings of 27th IEEE *Norchip Conference*, pp. 1-4, Nov 2009, Norway.

M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "An Efficient Unicast/Multicast Routing Protocol for MPSoCs," in Proceedings of 12th IEEE Euromicro Conference On Digital System Design (*DSD*), pp. 203-206, August 2009, Greece.

M. Ebrahimi, M. Daneshtalab, S. Mohammadi, Juha Plosila, and H. Tenhunen, "An Efficent Dynamic Multicast Routing Protocol for Distributing Traffic in NOCs," in Proceedings of 12th IEEE/ACM Design, Automation, and Test in Europe (*DATE*), pp. 1064-1069, April 2009, France.

P. Lotfi-Kamran, M. Daneshtalab, Z. Navabi, and C. Lucas, "BARP- A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs to Avoid Congestion," in Proceedings of 11th ACM/IEEE Design, Automation, and Test in Europe Conference (*DATE*), pp. 1408-1413, Mar 2008, Germany.

M. Daneshtalab, A. Pedram, M. H. Neishaburi, M. Riazati, A. Afzali-Kusha, and S. Mohammadi, "Distributing Congestions in NoCs through a Dynamic Routing Algorithm based on Input and Output Selections," in Proceedings of 20th IEEE International Conference on VLSI Design (*VLSID*), pp. 546-550, Jan 2007, India.

Workshop publications:

M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "An Efficient Topology for 3D Stacked Architectures," 3D Integration Workshop, The Design, Automation, and Test in Europe conference *(DATE)*, March 2011, France.

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

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

CONTENTS

List of Figures

List of Tables

List of Abbreviations

3D-IC	Three Dimensional Integrated Circuit
ACP	Adaptive Column-Path
AIOS	Adaptive Input-Output Selection
AMBA	Advanced Microcontroller Bus Architecture
AMP	Adaptive Multi-Path
AXI	Advanced eXtensible Interface
B2B	Back-to-Back
BOM	Begin Of Message
CAIS	Contention-Aware Input Selection
CAM	Content Addressable Memory
CARS	Contention Aware Routing Selection
CF	Congestion Flag
CIT	Concentrated Inter-layer Topology
CMIT	Clustered Mesh Inter-layer Topology
CL	Congestion Level
CMOS	Complementary Metal–Oxide–Semiconductor
CMP	Chip-Multiprocessor
CP	Column-Path
CS	Congestion Status
DA	Destination Address
DoA	Degree of Adaptiveness
DOR	Dimension Order Routing
DP	Dual-Path
DRAM	Dynamic Random Access Memory
dTDMA	Dynamic Time Division Multiple Access
DTL	Device Transaction Level
DU	Depacketizer Unit
DyAD	Dynamic Adaptive Deterministic
DyXY	Dynamic XY
ECC	Error Correcting Codes
EDXY	Enhanced Dynamic XY
EOM	End Of Message
F2B	Face-to-Back
F2F	Face-to-Face
FCFS	First Come First Service
FIFO	First In First Out

FLIT	FLow control digIT
FPGA	Field Programmable Gate Array
GALS	Globally Asynchronous Locally Synchronous
GEMS	General Execution-driven Multiprocessor Simulator
HAMUM	Hamiltonian Adaptive Multicast Unicast Method
HoL	Head of Line blocking
IP	Intellectual Property
ITRS	International Technology Roadmap for Semiconductors
LD	Low Distance
MESI	Modified Exclusive Shared Invalid
MH	Multiple Hop
MID	Message IDentifier
MLBS	Multi-Layer Buried Structures
MOESI	Modified Owned Exclusive Shared Invalid
MP	Multi-Path
MPSoC	Multi-Processor System-on-Chip
NF	Negative-First
NI	Network Interface
NMOS	N-Channel Metal–Oxide–Semiconductor
NL	North-Last
NoC	Network-on-Chip
OCP	Open Core Protocol
OE	Odd-Even
OS	Order Sensitive
PARSEC	Princeton Application Repository for Shared-Memory Computers
PB	Path-Based
PCI	Peripheral Component Interconnect
PE	Processing Element
PPE	Programmable Priority Encoder
PR	Priority-based Router
PU	Packetizer Unit
QoS	Quality-of-Service
RAM	Random Access Memory
RF	Row First
ROM	Read Only Memory
RR	Round-Robin
RTL	Register Transfer Level
RU	Reorder Unit
SA	Source Address
SAMBA	Single Arbitration Multiple Bus Accesses
SDRAM	Synchronous Dynamic Random Access Memory
SH	Single Hop

SIA	Semiconductor Industry Association
SNUCA	Static Non-Uniform Cache Architecture
SoC	Systems-on-Chip
SPARC	Scalable Processor ARChitecture
SPLASH	Stanford Parallel Applications for Shared Memory
STL	Standard Template Libraries
SV	Stress Value
TB	Tree-Based
TR	Typical Router
TS	Transfer Stage
TSV	Through Silicon Via
UB	Unicast-Based
VC	Virtual Channels
VCT	Virtual Circuit Table
VCTM	Virtual Circuit Tree Multicasting
VHDL	VHSIC hardware description language
VHSIC	Very-High-Speed Integrated Circuits
VLSI	Very Large Scale Integration
VOPD	Video Object Plane Decoder
WF	West-First
WRR	Weighted Round Robin

Chapter 1

Introduction

As indicated by several researchers and the International Technology Roadmap for Semiconductors (ITRS), nanometer Systems-on-Chip (SoCs) will most likely not have an economic yield if all transistors must be functional [1][2]. Besides, it is expected that Moore's law will continue to hold for another five to fifteen years where billion gates can be integrated in a chip. This capacity will allow integration of several tens to hundred resources like processor cores, DSP cores, and interface circuits (like Blue-tooth or Ethernet adapter), FPGA blocks, analog blocks, and memory blocks (any kind such as RAM, ROM and CAM). Thereby, it is possible to integrate more than one Processing Element (PE) in a SoC, being known as Multi-Processor System-on-Chip (MPSoC). MPSoCs have been widely used in high performance embedded systems, such as web servers, network processors, and parallel media processors. They combine the advantages of data processing parallelism of multi-processors and the high level integration of SoCs. The continuously increasing number of cores for such multi-billion transistor SoCs calls for a new communication architecture as traditional bus-based architectures are inherently non-scalable, making communication a bottleneck [1][2][3].

The Network-on-Chip (NoC) architecture paradigm, based on a modular packet-switched mechanism, can address many of the on-chip communication design issues such as performance limitations of long interconnects, and integration of high number of PE on a chip [1][2][3][4][5]. Notable examples of this architecture include Intel's 80-core Teraflops Research Chip [6] and Tilera's TILE64 [7].

A tiled-based 2D-mesh NoC based system, where one or more cores and other resources are encapsulated into a tile, is shown in Fig. 1-1. It consists of Routers (R), PE, and Network Interfaces (NI). PEs may be intellectual property (IP) blocks or embedded memories. Each PE is connected to the corresponding router port using the network interface. This enables to use packets for transferring information between PEs without requiring dedicated wirings for point to point connection. In brief, NoCs not only offer a scalable performance needed by systems which grow with each new generation [1][2], but also allow to mitigate the energy consumption by avoiding the use of long global wires. Since all links in the NoC can operate simultaneously on different data packets, a high level

of parallelism is making it attractive for replacing previous communication architectures like dedicated point-to-point signal wires, shared buses, or segmented buses with bridges. Furthermore, NoCs are reusable templates and aid to reduce the so called design productivity gap. Finally, none of the current on-chip interconnect approaches (buses and dedicated point-to-point channels) will meet all the requirements of future SoCs, as NoCs could potentially fulfill.

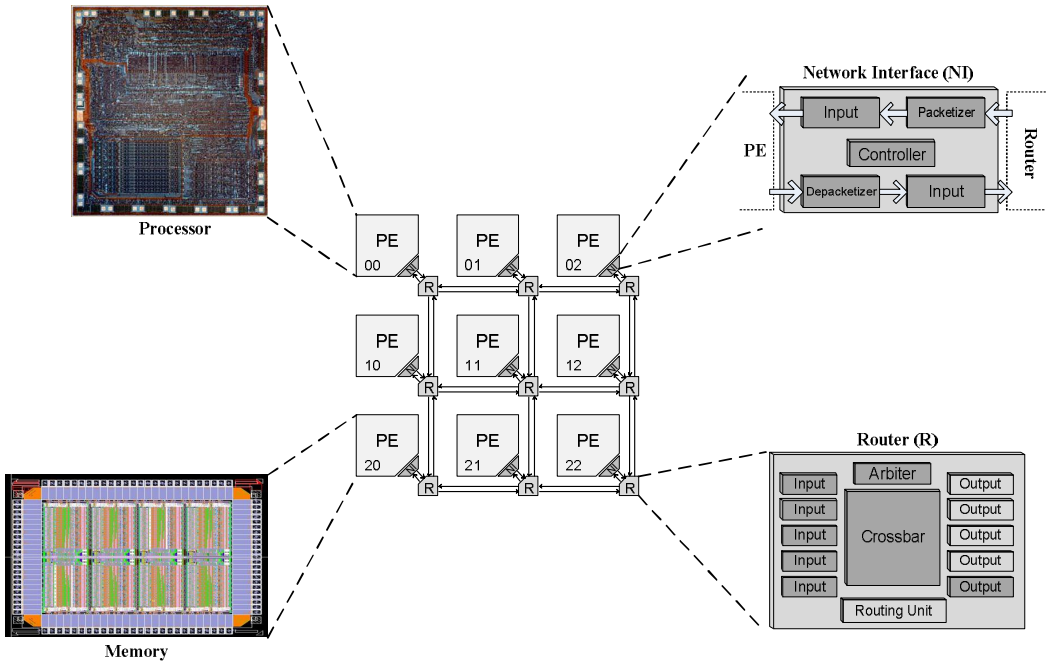


Fig. 1-1. Tile-based 2D-Mesh topology.

1.1 The Advantages of On-Chip Networks

Energy efficiency, reliability, reusability, scalability, and flexibility are the most important benefits of NoC from other on-chip communication approaches.

1.1.1 Energy Efficiency

According to the International Technology Roadmap for Semiconductors (ITRS) [8] and Semiconductor Industry Association (SIA) [9] roadmaps, clock frequency and number of on-chip devices are increased. That is, much tighter power budgets for all system components are required. Based on the roadmaps, as computation and storage components benefit from device scaling, the energy for global communication does not scale down. Hence, communication-energy minimization will be a growing concern in future

technologies. The on-chip networks aim to reduce this problem by scaling wires. This new model allows the decoupling of the PEs from the network. The need for global synchronization can thereby disappear. This new approach employs explicit parallelism, exhibits modularity to minimize the use of global wires, and utilizes locality for power minimization [10][11]. Furthermore, network traffic control and monitoring can help in better managing the power consumed by networked computational resources. For instance, clock speed and voltage of end nodes can be varied according to available network bandwidth. The emphasis on energy minimization creates a sleuth of novel challenges that have not been addressed by traditional high-performance network designers [10][11].

1.1.2 Reliability

As the geometries of the transistors reach the physical limits of operation, it becomes increasingly difficult for the hardware components to achieve reliable operation. The variability in process manufacturing, issues of thermal hotspots and effects of various noise sources, such as power supply fluctuations, pose major challenges for the reliable operation of current and future NoC-based MPSoCs. NoCs are particularly suited for implementation of fault-tolerant techniques, due to their inherent parallelism and potential for re-configurability. Fault-tolerant techniques can be implemented at different levels, from hardware redundancy to software-based error recovery schemes. Adaptive routing algorithms combined with error detection mechanisms show great promise in achieving fault-tolerant on-chip communication. If data is sent on an unreliable channel in packets, error detection and recovery is easier, because the effect of errors is contained by packet boundaries, and error recovery can be carried out on a packet-by-packet basis. Error correction can be achieved by using standard error correcting codes (ECC), whereas robust and fault-tolerant routing algorithms can route around faulty regions [12].

1.1.3 Reusability

PEs are usually obtained from internal sources or third parties, and integrated on a single chip. These reusable PEs may include embedded processors, memory blocks, interface blocks, analog blocks, and components that handle application specific processing functions. Corresponding software components are also provided in a reusable form and may include real-time operating systems and kernels, library functions, and device drivers. That is, PEs are reusable in nature if they conform to a common interface and synchronization mechanisms with the on-chip network. Using a standard interface such as AXI [13], OCP [14], and DTL [15], in on-chip networks facilitates the employment of reusable components. In fact, employing a standard interface does not change the way PEs are developed, since they will still be developed for a certain protocol. What changes is that a public domain protocol is used and accepted by the industry as a standard, like the PCI standard for microcomputer manufacturers. Accordingly, not only the PEs reusability becomes higher but also the design time is reduced [16]. In addition, on-chip routers are generic in nature and the communication can be employed with any conforming PE.

1.1.4 Scalability

NoC platform is composed of on-chip routers and communication links that are basically distributed and independent. Each PE is added into the network along with a dedicated router having a unique address or coordinate in the network. The communication exploits the packet switching scheme while there is no central arbitration mechanism of the communication platform. Therefore, the performance in this communication architecture is not constrained or degraded by the addition of PEs. This is the essential characteristic of a scalable and modular architecture [1][2][3]. Indeed, on-chip interconnection network plays an important role in providing scalability to integrate hundreds or even thousands of processing elements in a single billion-transistor chip and alleviate design productivity gap [17]. In fact, using data packets for communication, a high level of parallelism is achieved as all channels can be operated simultaneously. Thereby, on-chip network improves the scalability in comparison with previous communication structures such as shared buses or segmented buses.

1.1.5 Flexibility

Utilizing common buses between the communicating resources in SoCs will not give any flexibility since the needs of the communication have to be thought of every time a design is made. However, they suffer from low scalability [1]-[5]. NoC solves their shortcomings by implementing a communication network of routers and resources. NoC is a very flexible communication infrastructure allowing the same physical link to be shared by many different connections. As future SoC platforms are expected to contain hundreds of PEs, NoC needs to support an even larger number of connections and many connections span a large number of routers. This leads the same SoC platform to be used in a wide range of different applications and thereby increases the production volume. As the same SoC platform is to be used for many different applications, the NoC must be able to support a wide range of bandwidth and Quality-of-Service (QoS) requirements. The requirements of the applications can be very different, and the NoC must therefore be very flexible.

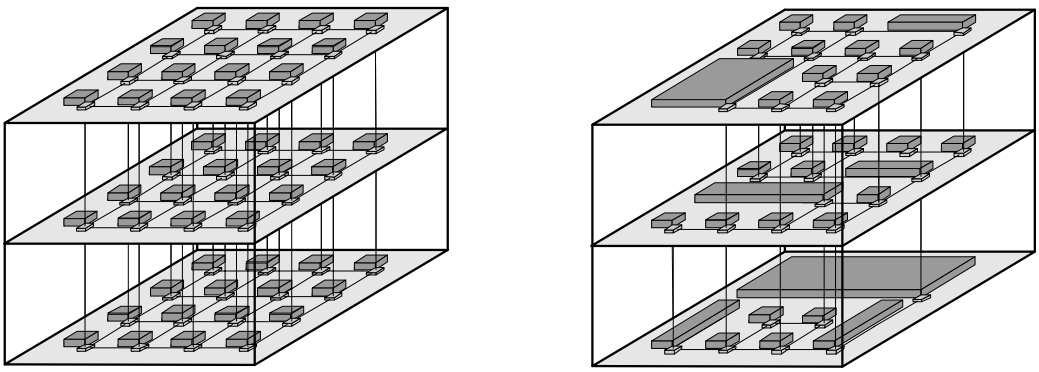
1.2 Three-Dimensional ICs

Two-dimensional (2D) chip fabrication technology is facing lots of challenges in the deep submicron regime even by utilizing NoC architectures [18][19], e.g. designing the clock-tree network for a large chip, limited floor-planning choices, increasing the wire delay and power consumption, integrating various components that are digital, analog, MEMS, RF, etc. The Three Dimensional (3D) integration has emerged as a potent solution to address these problems and the design complexity of MPSoC in 2D Integration Circuits (IC). 3D ICs reduce the interconnect delay problem by stacking vertically active silicon layers as well as offering a number of advantages over the traditional 2D chip [18][19][20][21][22]: (1) shorter global interconnects; (2) higher performance; (4) high memory bandwidth; (3) lower interconnect power consumption due to wire-length reduction; (4) higher packing

density and smaller footprint; and (5) support for the implementation of mixed-technology chips, e.g. NMOS DRAM stacking on top of CMOS processor cores. However, thermal problem is still an important challenge for 3D IC circuit design.

1.2.1 3D IC Technology Overview

There are many technologies for die stacking being pursued by industry and academia. Wafer-Bonding [30][31] and Multi-Layer Buried Structures (MLBS) [32][33] are the most promising ones. The details of these processes are described in [18]. Wafer-to-wafer bonding appears to be the leading contender in industry and many recent academic studies have assumed this type of 3D stacking technology [18]-[23][34]. Wafers can be stacked either Face-to-Face (F2F) or Face-to-Back (F2B) and both have pros and cons. While the former provides the greatest layer-to-layer via density, it is suitable for two-layers; and additional layers would have to employ Back-to-Back (B2B) placement using larger and longer vias. On the other hand, Face-To-Back provides uniform scalability to an arbitrary number of layers, despite a reduced inter-layer via density [27]-[35]. Layers, stacked on top of each other, are connected via vertical interconnects tunneling through them. Wire bonding, micro-bump, contactless, and Through Silicon Via (TSV) are some of the vertical interconnect (Inter-layer communication) technologies that have been used in stacked structures [32]. The distance between wafers can range from 5μm to 50μm [22][24], which is much shorter than the wire length between cores on a tier, and the pitches of a TSV can range from 1μm to 10μm square [22][24]. That is, the wire delay, power consumption and chip form factor are significantly reduced [25][26][28]. Thus, the TSV interconnection has the potential to offer the greatest vertical interconnect density and is the most promising one among these vertical interconnect technologies [27]-[35]. In this thesis, we assumed the F2B method with TSV interconnects to provide more scalability when more than two layers are employed.



(a) (b)

Fig. 1-2. (a) Homogeneous and (b) Heterogeneous 3-D Network-on-Chip structures using Through Silicon Vias (TSVs) technology to connect stacked layers vertically.

1.2.2 3D NoC

Combining the benefits of 3D ICs and NoCs schemes provides a significant performance gain for 3D architectures. 3D NoC topologies not only create scalable networks to provide communication requirements in 3D ICs [19]-[22], but also are a crucial factor of 3D chips in terms of performance, cost, and energy consumption [19]. Various on-chip network topologies have been studied for 3D NoCs [19]-[23][25][27][29]. Mesh-based structures are popularly used in 3D systems (Fig. 1-2), because their grid-based regular structure is intuitively considered to be matched to the 2D VLSI layout for each stack layer [19][20][21][22][25]. Nevertheless, if the number of IP-cores and memories increases in each layer, more TSVs are necessitated to handle the inter-layer communication. Inasmuch as each TSV employs a pad for bonding, the area footprint of TSVs in each layer is augmented significantly [22][29].

1.3 Adaptive On-Chip Network

NoC is flexible to dynamically support the communication among modules in a system with heavily varying workloads. To augment resource utilization and flexibility, the architecture of NoC needs to be integrated with novel adaptive methodologies employing resource multiplexing mechanisms at varying workloads. In the scope of this thesis, several novel adaptive schemes for the on-chip network architecture are presented to exploit all the benefits that can be obtained. Each component of on-chip network platform can be implemented adaptively to increase the utilization and performance. In order to route data packets to the non-congested area and/or give the priority to a packet passing through a congested area of the network to ease the congestion faster, respectively. Network interfaces can handle in-order delivery which is a practical approach when exploiting an adaptive routing algorithm for distributing packets through the network or when exploiting dynamic memory access scheduling in memory controller to reorder memory requests. Adaptive on-chip network interface architectures are utilized to increase the resource utilization and system performance.

In the realm of 3D NoCs, the router-based and bus-based organizations are the two dominant architectures for utilizing TSVs as inter-layer communication channel. The former suffers from poor scalability and deteriorates the performance at high injection rates, and the latter consumes more area and power. Using adaptive inter-layer communication structure not only can reduce the delay and complexity of traditional arbitration but also reduces the area overhead of TSVs, which can impact designing 3D architectures with a large number of TSVs.

1.4 Thesis Contributions

Multicore designers have moved from a bus-based view of design to a network-based view to overcome several problems outlined above. NoC architectures are emerging as a scalable and modular solution to global communication within large MPSoCs. NoCs diminish the emerging wire-delay problem and address the need for substantial interconnect bandwidth by replacing shared buses with packet-switched on-chip router networks. The idea of on-chip network creates many new research opportunities. In particular, this thesis has explored the adaptive implementation of on-chip network and communication design spaces in the following directions:

- Adaptive on-chip communication routing protocols
- Adaptive on-chip router
- Adaptive network interface
- Adaptive inter-layer communication structure for 3D NoCs

This thesis contains several key ideas to support on-chip communication in the realm of 2D and 3D NoCs. The contributions of this dissertation with a brief summary are as follows:

- An adaptive unicast routing algorithm in the realm of 2D-mesh NoCs is presented [36]. The routing algorithm, based on Dynamic XY (DyXY), is called Enhanced Dynamic XY (EDXY). It is an adaptive congestion-aware routing algorithm implemented by adding two congestion wires (one in each direction) between each two cores which indicate the existence of congestion in a row and a column. These signals enable the routing algorithm to avoid these paths when there are other paths between the source and destination pair.
- For both unicast and multicast traffic, two adaptive routing protocols are presented. The proposed routing protocols, named Low Distance (LD) [37][38] and Hamiltonian Adaptive Multicast Unicast Method (HAMUM) [39][40], maximize the degree of adaptiveness of the routing functions while guaranteeing deadlock freedom. The presented routing protocols invoke non-congested paths in routing the messages to prevent creating highly congested areas. This is achieved by considering the congestion condition of the input ports. Furthermore, both unicast and multicast aspects of the presented methods have been widely investigated separately.
- To reduce the power consumption and improve the performance of on-chip networks, a novel on-chip router architecture is proposed. The router architecture, named Adaptive Input-Output Selection (AIOS), is for avoiding congested areas in 2D-mesh NoCs via employing efficient input and output selection [41][42][43]. The output selection utilizes an adaptive routing algorithm based on the congestion condition of neighboring routers while the input selection allows packets to be serviced from each input port according to its congestion level.
- To achieve higher memory bandwidth and increasing memory parallelism in network-based multiprocessor architectures, multiple SDRAMs can be accessed simultaneously. In such architectures, not only resource utilization and latency are the critical issues but also a reordering mechanism is required to deliver the response transactions of

concurrent memory accesses in-order. To cope with these issues in this thesis, an adaptive on-chip network interface architecture is presented [44][45][46][47]. The proposed network interface exploits an efficient reordering mechanism to handle the in-order delivery and utilizes the AXI transaction based protocol to bring compatibility with existing IP cores. On top of that, a smart memory controller is integrated in this network interface to improve the memory utilization and reduce both memory and network latencies.

- To diminish the area overhead of TSVs and power dissipation on each layer with minimal performance penalty, two stacked structures for 3D architectures are proposed [48][49]. The presented schemes benefit of clustering the mesh topology in order to mitigate TSV footprint on each stacked layer. On top of that, to improve the performance of vertical channels, a new bus architecture is introduced [50]-[55]. The proposed bus architecture overcomes the drawbacks of previously presented buses, designed for vertical channels, and improves the performance by reducing the delay and complexity of traditional bus arbitration.

1.5 Thesis Organization

This thesis is organized as follows. Chapter 2 gives a general overview of on-chip networks while Chapter 3 introduces three adaptive routing protocols, whereas the first one is related to the unicast traffic and the other two routing protocols are associated with unicast and multicast traffic. Two low latency and power efficient router architectures are presented in Chapter 4, while the adaptive network interface architecture for on-chip networks is described in Chapter 5. Concerning 3D architectures, two cluster-based topologies along with a novel pipeline bus architecture are explained in Chapter 6. The idea of the balance partitioning as well as multiple partitioning methods, supported by an adaptive routing model, is also presented in this chapter. Finally, the thesis is concluded in Chapter 7.

Chapter 2

On-Chip Networks

On-chip networks are emerged as a highly scalable, reliable, and modular interconnect fabric for MPSoCs [1][2][3][4][5]. As the network fabric takes up a substantial portion of system power budget [10], and power is one of the most important constraint in billion-transistor chips, in addition to network delay and area, the interconnect power consumption should be taken into consideration. Therefore, on-chip interconnection networks should be accommodated into the limited silicon area using efficient topology, routing algorithm, and router implementation.
In this chapter, concepts of on-chip networks including network topologies, switching techniques, flow control mechanisms, virtual channels, output selection, routing algorithms, and a general network-on-chip architecture are presented.

2.1 Network Topology

The network topology is the study of the arrangement and connectivity of the routers. In other words, it defines the various channels and the connection pattern that are available for the data transfer across the network. Performance, cost, and scalability are the important factors in the selection of the appropriate topology. Shared-Bus, Crossbar, Butterfly Fat-Tree, Ring, Torus, and 2D-Mesh are the most popular topologies for on-chip interconnects which have been commercially used [2][60].
Direct networks have at least one PE attached to each router of the network so that routers may regularly spread between PEs. This helps to simplify the physical implementation. The shared-bus, ring, and 2D mesh/torus topologies (Fig. 2-1) are examples of direct networks, and provide tremendous improvement in performance, but at a cost of hardware overhead, typically increasing as the square of the number of PEs. On the other hand, indirect networks have a subset of routers not connected to any PE. All tree-based topologies where PEs are connected only to the leaf routers (e.g. the butterfly topology) as well as crossbar switch (Fig. 2-1) are indirect networks.

The shared-bus topology is the simplest using a shared link common to all PEs where they compete for exclusive access to the bus. For communication intensive applications it is necessary to overcome the bandwidth limitations of the shared-bus topology and move to scalable networks. However, this topology scales very poorly as the number of PEs increases. A small modification to the shared-bus topology to allow more concurrent transactions is to create the ring topology where every PE has exactly two neighbors. In this topology, messages hop along intermediate PEs until they arrive at the final destination. This causes the ring to saturate at a low injection rate for most traffic patterns. The crossbar topology is a fully connected one which allows every PE to directly communicate with any other PE. Hence, each topology has its own advantages and disadvantages.
The fat-tree topologies suffer from the fact that the number of routers exceeds the number of PEs, when the amount of PEs increases. This incurs an important network overhead. For the on-chip interconnects the network overhead is more critical than for the off-chip networks, and the design scalability is more essential. Because of the simple connection and easy routing provided by adjacency, mesh and torus networks are widely used in multiprocessor architectures. Both torus and mesh topologies are fully scalable. Although torus provides a better performance, the regularity, better utilization of links, and lower network overhead are some of the preferences for mesh. That is, the mesh topology is more economic scheme since the routers on the borders are smaller.

2.2 Switching Mechanism

The switching mechanism determines how messages traverse a route in a network. The goal is to effectively share the network resources among messages traversing the network. Basically, circuit switching and packet switching form the two extremes of switching mechanisms.
In circuit switching a connection from a source to a destination is established prior to the transmission of data and exclusively reserved until the message is completely transferred, i.e. as in telephone networks that set up a circuit through possibly many routers for each call. This mechanism has low delay and guaranteed bandwidths, but suffers from channel utilization, low throughput, and long initialization time to setup a connection.
Packet switching is an alternative mechanism where data is not transmitted on a predefined circuit. A message can be divided into packets which share channels with other packets. Each packet consists of a header which contains routing and control information, data payload, and possibly a tail. The data payload follows the channel reserved by header while the tail releases the channel reservation. Packets are individually and independently routed through the network, and at the destination the packets are assembled into the original message. If a message is divided into several packets, the order of packets at arrival PE must be the same as departure. Therefore, in-order delivery is an essential part that should be supported by on-chip networks. The packet switching mechanism improves channel utilization and network throughput.

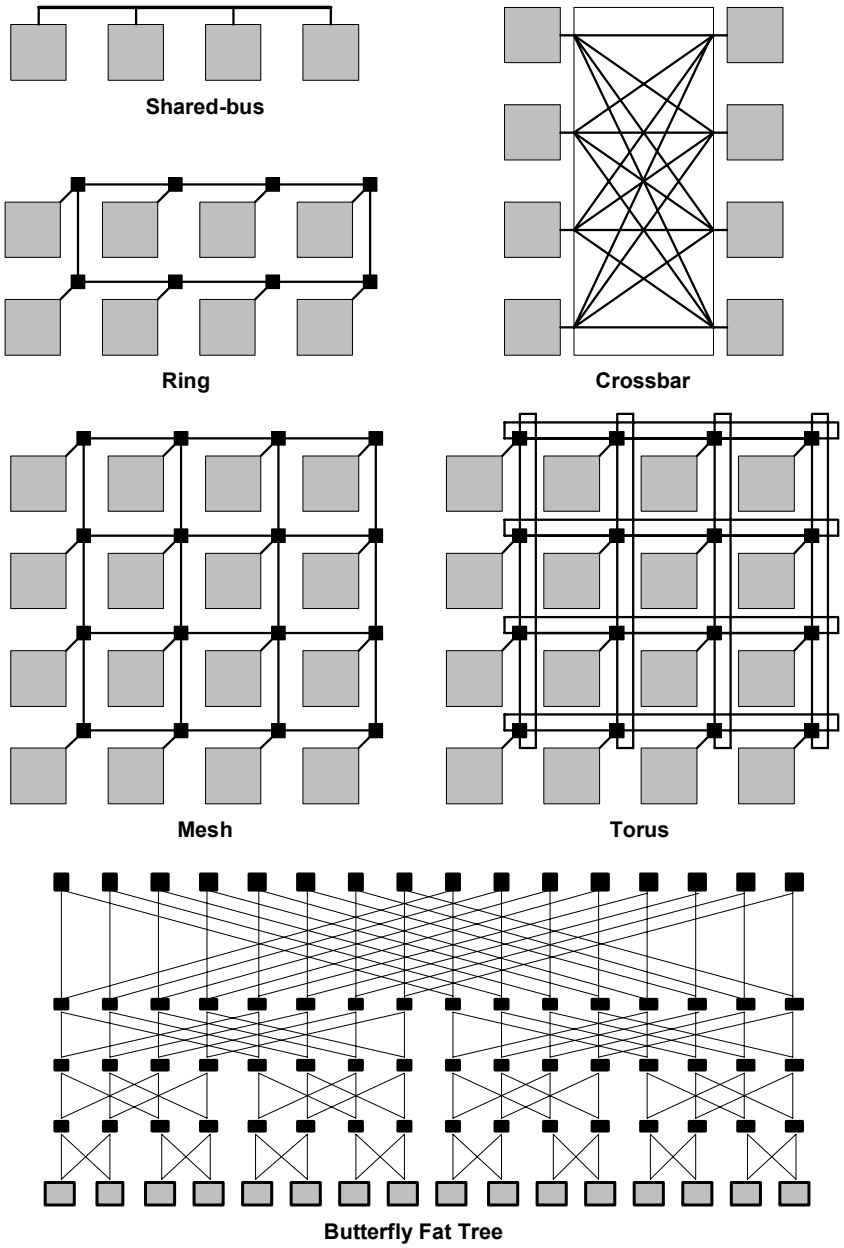


Fig. 2-1. Network topologies of Shared-bus, Ring, Crossbar, Mesh, Torus, and Butterfly.

In the packet switching domain, buffered flow control defines the mechanism that deals with the allocation of channels and buffers for the packets traversing between source and destination. The flow control mechanism is necessary when two or more packets compete to use the same channel, at the same time. Commonly three different buffered flow control strategies are used: store-and-forward, virtual cut through, and wormhole. When these mechanisms are implemented in on-chip networks, they have different performance metrics along with different requirements on hardware resources.

2.2.1 Store-and-Forward

The store-and-forward mechanism is the simplest flow control mechanism. In this approach, each router along the path stores the entire packet in the buffer and then, the packet is forwarded to a selected neighboring router if the chosen neighboring router has enough empty buffering space available to hold the whole packet. This mechanism requires a large amount of buffering space (at least the size of the largest packet) in each router of the network, which can increase the implementation cost dramatically. On top of that, network latency increases significantly because a packet cannot be forwarded to the next router until the whole packet is received and stored in the current router. Consequently, the store-and-forward approach is impractical in large-scale Networks-on-Chip.

2.2.2 Virtual Cut-Through

The virtual cut-through mechanism was proposed to address the large network latency problem in the store-and-forward strategy by reducing the packet delays at each routing stage. In this approach, one packet can be forwarded to the next stage before its entirety is received by the current route which reduces the store-and-forward delays. However, when the next stage router is not available, similar to the store-and-forward, the virtual cut-through approach also requires a large buffering space at each router to store the whole packet.

2.2.3 Wormhole

In this mechanism, a packet is divided into smaller segments called FLITs (FLow control digIT) [59]. Then, the flits are routed through the network one after another, in a pipelined fashion. The first flit in a packet (header) reserves the channel of each router, the body (payload) flits will then follow the reserved channel, and the tail flit will later release the channel reservation. The wormhole mechanism does not require the complete packet to be stored in the router while waiting for the header flit to route to the next stages. One packet may occupy several intermediate routers at the same time. That is, the wormhole approach is similar to the virtual cut-through, but here the channel and buffer allocation is done on a flit-basis rather than packet-basis. Accordingly, the wormhole approach requires much less buffer space, thus, enabling small, compact and fast router designs. Because of these advantages, the wormhole mechanism is an ideal flow control candidate for on-chip networks.

2.3 Virtual Channels

There is a possibility of blocking in the wormhole network when a packet reserves a channel along a path which is prevented to be used by other packets. The use of Virtual Channels (VCs) overcomes the problem of blockages in the wormhole network via allowing blocked packets to be passed by other packets. This is accomplished by assigning several VCs, each with a separate flit queue, to each physical channel. For each VC, when the header flit arrives, a buffer will be assigned to the incoming packet, and is reserved until the trailer flit is transmitted. If a packet holding a VC gets blocked, other packets from other VCs can still traverse the physical channel.

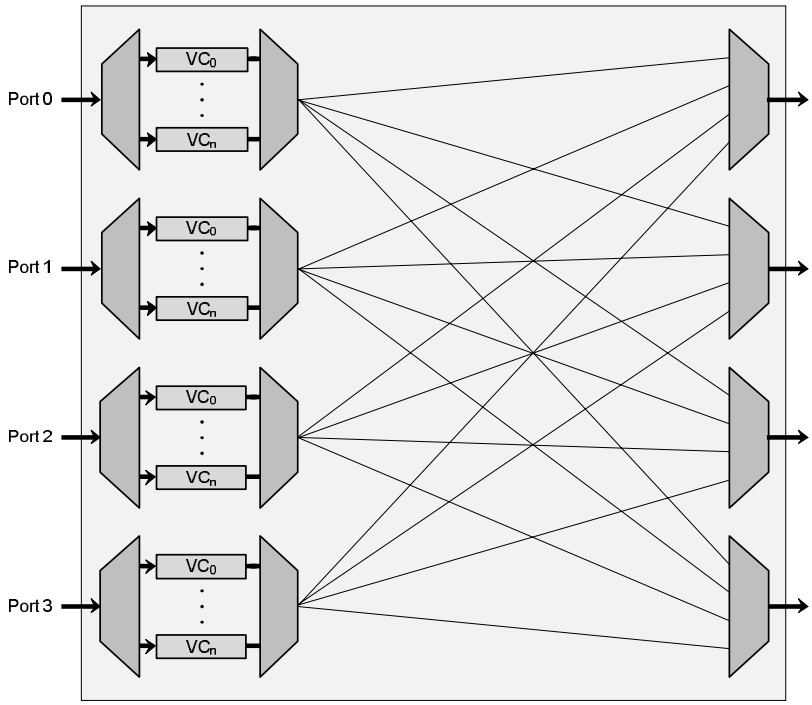


Fig. 2-2. A typical router using VCs.

As depicted in Fig. 2-2, at an input port the incoming flits are stored in distinct channel buffers which are multiplexed together again onto the output ports. If one of the channels is blocked, the other channels can access the outputs. Also, with VCs a network can be divided into multiple disjoint subnetworks which have been explained in Chapter 3. VCs were introduced to solve the deadlock avoidance problem, and to improve network latency and throughput. Fig. 2-3(a) shows how a packet *A* blocked between routers 3 and 4 which also blocks the packet *B* when the network is not equipped with VCs. As illustrated in Fig. 2-3(b), using VCs allows dual utilization of the physical channel between routers 3 and

4 where the packet *B* can pass the router 3. However, although employing VCs improves the performance and reduces head of line blocking (HoL) efforts in the network, it increases design complexity of the link controller and flow control mechanisms.

2.4 Output Scheduling

When multiple packets request for an output port, the need of an output scheduling algorithm that determines the priority order of candidate packets to advance emerges. In fact, the scheduler gives a priority order to each packet, and then the output ports select the highest-ordered packets to be forwarded. There is a variety of solutions with different implementation complexity and different performance characteristics, e.g. round-robin, first-come first-served, etc. The starvation prevention is the main concern that must be considered in the scheduler.

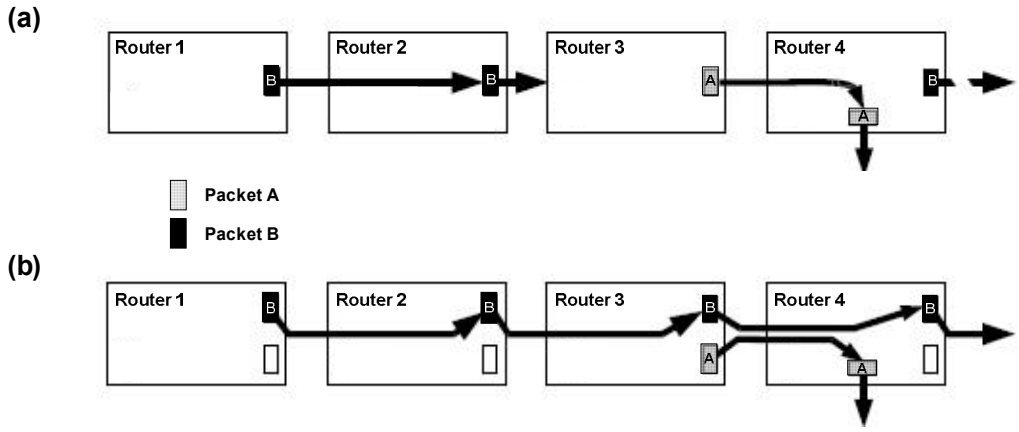


Fig. 2-3. Using VC for avoiding deadlock.

2.5 Routing Algorithm

Routing is the process that are used to forward the packets along appropriate directions in the network between a source and a destination. Routing algorithms not only affect the transmission time but also can impact the power consumption and congestion conditions in the network.

2.5.1 Source versus Distributed Routing

Routing can be utilized either at the source router or with a distributed manner by routers along the path. In the source routing scheme the entire route of a packet is decided by the source router stacking the exact router-to-router itinerary of a packet in the header. As the

packet traverses in the network this information is used by each router on the path to navigate the packet towards the destination. This scheme is a simple solution for on-chip networks while the problem of the routing information overhead is the drawback of this scheme, i.e. for a network with a diameter of *k*, each packet requires at most *k* routing information stacked on the header of the packet. Accordingly, if the network grows the header overhead becomes significant which is impractical for on-chip networks. In contrast, in the distributed routing approach the routing decision is taken by the individual routers depending on different parameters while the header of a packet has to include only the destination address. Each intermediate router examines the destination address and decides along which channel to forward the packet. However, the router complexity of the latter scheme is higher than the former scheme.

2.5.2 Deterministic versus Adaptive Routing

Distributed routing scheme can be classified as deterministic and adaptive. Deterministic routing algorithms route packets in a fixed path between the source and destination routers. Implementations of deterministic routing algorithms are simple but they are not able to balance the load across the links in non-uniform or bursty traffic [61][62]. Adaptive routing algorithms are proposed to address these limitations. By better distributing load across links, adaptive algorithms improve network performance and also provide tolerance if link or router failure occurs. In adaptive routing algorithms, path of a packet from the source to the destination is determined by network conditions. An adaptive routing algorithm decreases the probability of passing a packet from a congested or malfunction link. While deterministic routing algorithms are the best choice for uniform or regular traffic patterns, the adaptive schemes are preferable in presence of irregular traffic (non-uniform or bursty traffic) or in networks with unreliable routers and links. Furthermore, since packets may arrive to the destination from different paths and with different latencies an adaptive routing could not guarantee the order of packets. To achieve in-order delivery property, a hardware reordering module is required. These requirements increase both design complexity, and likely communication latency.

2.5.3 Minimal versus Non-Minimal Routing

Adaptive routing algorithms can either be minimal or non-minimal. Minimal routing algorithms allow only shortest paths to be chosen, while non-minimal routing algorithms also allow longer paths. Besides, a minimal fully adaptive routing algorithm can route packets along any shortest path adaptively; and a minimal partially adaptive routing algorithm cannot route packets along every shortest path.

2.5.4 Unicast and Multicast Routing Protocols

The communication in on-chip networks can be either unicast (one-to-one) or multicast/broadcast (one-to-many) [58]. In the unicast communication, a packet (message) is sent from a source router to a single destination router, while in the multicast

communication a packet is transmitted from a source router to an arbitrary set of destination routers. Thus, the former is a special case of the latter. These protocols are described in the next chapter.

The multicast communication is frequently employed in many application of MPSoC such as replication [70], barrier synchronization [71], cache coherency in distributed shared-memory architectures [72], and clock synchronization [73]. Although the multicast communication can be implemented by multiple unicast communications, it produces significant amount of unnecessary traffic increasing the latency and congestion in the network [74].

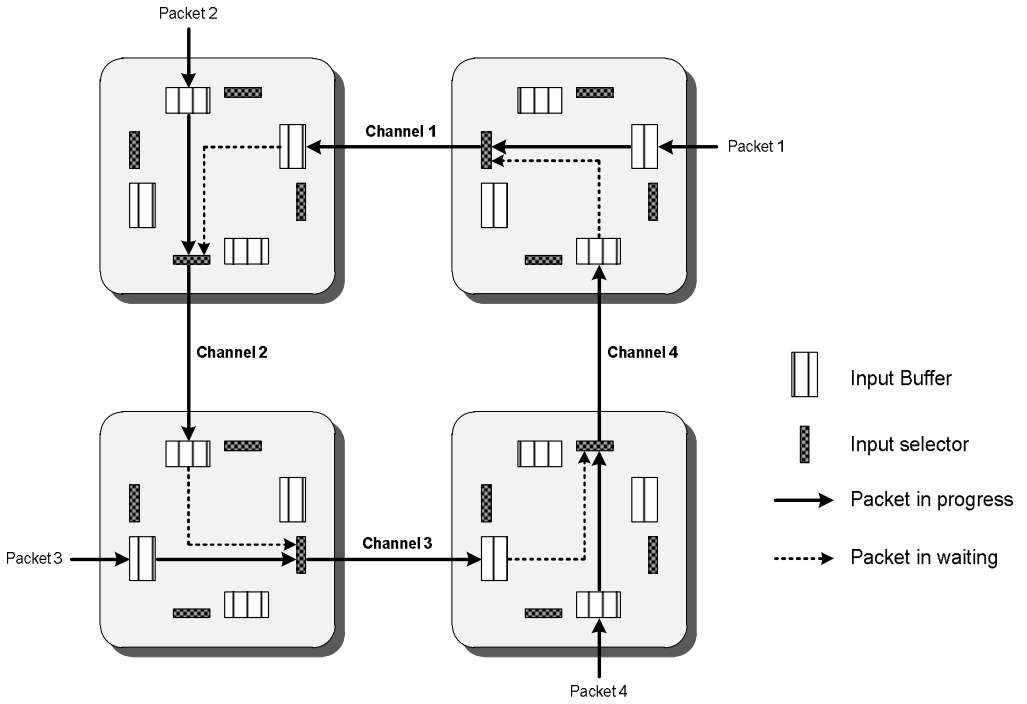


Fig. 2-4. Deadlock scenario with four packets [68].

2.5.5 Deadlock and Livelock

Deadlock is a situation that occurs when a cycle of packets are waiting for one another to release a shared channel in a circular dependency. Fig. 2-4 shows a deadlock scenario with four packets routed in a circular manner. Each packet is holding a flit buffer while requesting the buffer held by another packet. The packet 1 occupying channel 1 is requesting for the channel 2 allocated to the packet 2 which wants to use the channel 3. But that channel is occupied by the packet 3 requesting the channel 4 which is held by the packet 4. The packet 4 completes the circle by waiting for the channel 1 so that no packet

can advance since the required resource is already held by another packet and will never be released.
Livelock is a condition where a packet keeps circulating within the network without ever reaching its destination. It is the result of using a non-minimal adaptive routing algorithm. A livelock free routing algorithm has to guarantee forward progress of each packet, where after each hop the packet is in one step closer to its destination.

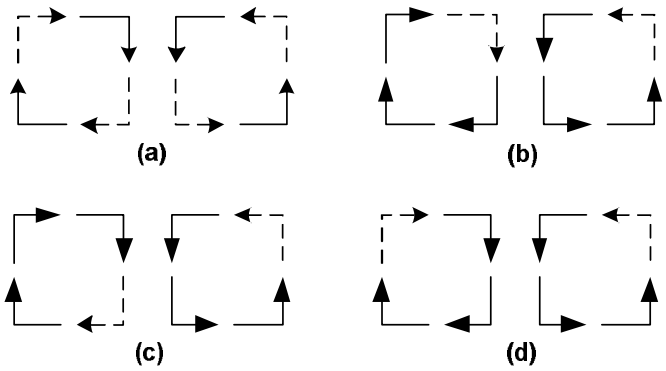


Fig. 2-5. All possible turns in (a) XY routing (b) Negative-First (c) West-First (d) North-Last (The solid lines indicate the allowable turns and the dash lines indicate the unallowable turns).

2.5.6 Turn Model Routing

Turn Model routing scheme based on wormhole switching mechanism provides deadlock and livelock freedom in the two-dimensional mesh topology [64][97]. This model is also chosen as a representative of minimal and partial adaptive routing. In the turn model, deadlock can be avoided by prohibiting just enough turns to break all the cycles. Four well-known turn models are XY, Negative-First(NF), West-First(WF) and North-Last(NL) as shown in Fig. 2-5. Although the XY routing algorithm prohibits four turns to avoid deadlock, the other models avoid only two turns out of eight turns.
The Odd-Even model is one of the most popular partial adaptive wormhole routing algorithms in 2D mesh on-chip interconnection network [64] without virtual channels. Unlike the turn model which prohibits certain turns in all locations of the network, in the Odd-Even model some turns are restricted only in even columns and some other turns are prohibited in odd columns. Therefore, the degree of adaptiveness provided by this model is higher than the other turn models. Odd-Even rules can be described by the following rules:

Rule 1: East-North and East-South turns cannot be taken in even columns (Fig. 2-6(a)).
Rule 2: North-West and South-West turns cannot be taken in odd columns (Fig. 2-6(b)).

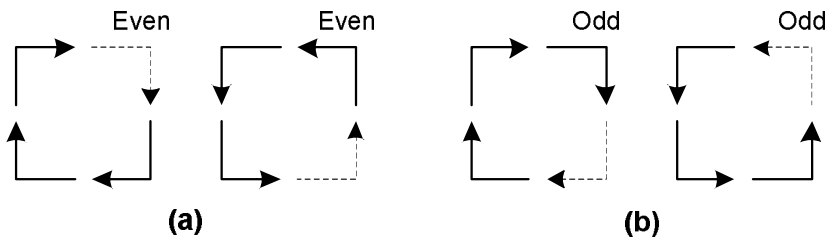


Fig. 2-6. The Odd-Even turn model rules: (a) prohibited turns in even columns (b) prohibited turns in odd columns.

2.6 Network-on-Chip Architecture

As described previously, 2D-mesh has many desirable properties for NoCs, including scalability, high bandwidth, and the fixed degree of routers [83]. A 2D-mesh NoC based system is shown in Fig. 2-7. As mentioned earlier, NoC consists of Routers (R), Processing Elements (PE), and Network Interfaces (NI). PEs may be intellectual property (IP) blocks or embedded memories. Each core is connected to the corresponding router port using the network interface. To be compatible with existing transaction-based IP-cores, the AMBA AXI protocol is used. AMBA AXI is an interfacing protocol, having advanced functions such as a multiple outstanding address function and data interleaving function [13]. AXI, providing such advanced functions, can be implemented on NoCs as an interface protocol between each PE and router to avoid the structural limitations in SoCs due to the bus architecture. The protocol can achieve very high speed of data transmission between PEs [13]. In the AXI transaction-based model [13][108], IP cores can be classified as master (active) and slave (passive) IP cores [109][111]. Master IP cores initiate transactions by issuing read and write requests and one or more slaves (memories) receive and execute each request. Subsequently, a response issued by a slave can be either an acknowledgment (corresponding to the write request) or data (corresponding to the read request) [109]. The AXI protocol provides a "transaction ID" field assigned to each transaction. Transactions from the same master IP core, but with different IDs have no ordering restriction while transactions with the same ID must be completed in-order. Thus, a reordering mechanism in the network interface is needed to afford this ordering requirement [13][14][112]. The network interface lies between a PE and the corresponding attached router. This unit forms the foundation of the generic nature of the architecture as it prevents the PEs from directly interacting with the rest of the network components in the NoC.

A generic network interface architecture is shown in Fig. 2-7. The network interface consists of input buffers (forward and reverse directions), a Packetizer Unit (PU), a Depacketizer Unit (DU), and a Reorder Unit (RU). A data burst coming from a PE is latched into the input buffer of the corresponding network interface. PU is configured to packetize the burst data stored in the input buffer and transfer the packet to the router.

Similarly, data packets coming from the router are latched into the input buffer located in the reverse path. DU is configured to restore original data format, required for the PE, from the packet provided by the router. The RU performs a packet reordering to meet the in-order requirement of each PE.
As master IP-cores may operate at high clock frequencies and slave IP-cores operates at low clock frequencies, an interface between the IP-cores and on-chip network is required for crossing two clock domains.

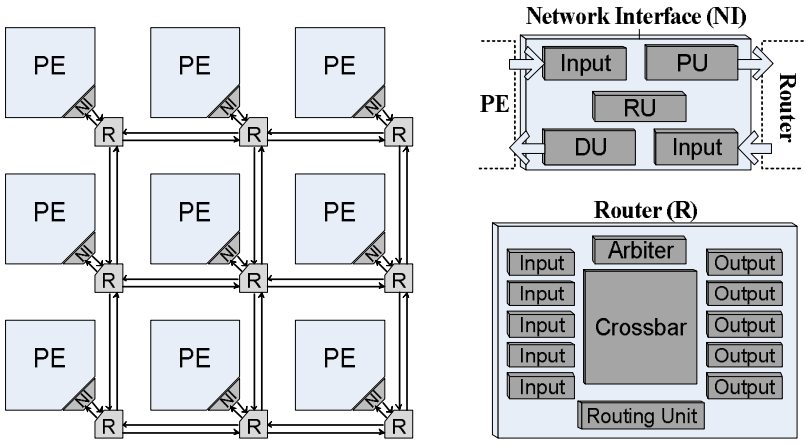


Fig. 2-7. Tile-based 2D-Mesh topology.

2.7 Summary

In this chapter, some of the most important concepts in the domain of NoC design that help to describe the thesis were presented. We have discussed various topologies for direct and indirect networks. Different switching, flow control mechanisms along with using virtual channels, routing schemes, output selection technique, and a general network-on-chip architecture were also described. These concepts presented here are further mentioned in various places in the rest of this thesis.

Chapter 3

Adaptive Routing Protocols in Networks-on-Chip

On-chip networks like computer networks may take advantage of data packetization to ensure the fairness of communication [56][57]. Since on-chip networks should use lighter and faster protocol layers, they do not need to follow all the standard schemes for the communication in computer networks.
In this chapter, we consider NoCs with 2D mesh topologies which offer many desirable properties including better parallelism and scalability, low cross-section bandwidth, and fixed degree of nodes compared to many other topologies for MPSoC interconnection [59]. Besides, meshes are suitable for a variety of applications including matrix computation, image processing and problems whose task graphs can be embedded naturally into the topology [60]. An $m \times n$ 2D mesh consists of N ($= m \times n$) nodes where each node has an associated integer coordinate pair (x, y) such that $0 \leq x < n$ and $0 \leq y < m$. Two nodes with coordinates (x_i, y_i) and (x_j, y_j) are connected by a communication channel if and only if $|x_i - x_j| + |y_i - y_j| = 1$.

3.1 Unicast Routing Protocols

In NoCs, routing algorithms are used to determine a path of a packet from source node to destination node. In this chapter, some of the related routing algorithms are also described.

3.1.1 XY Routing Scheme

XY, a deterministic routing algorithm for 2D meshes, is introduced in [2][61][63]. As shown in Fig. 3-1(a), in this routing algorithm, each packet first travels along the X and then the Y direction to reach the destination. For this scheme, deadlock never occurs but no adaptivity exists in this algorithm.

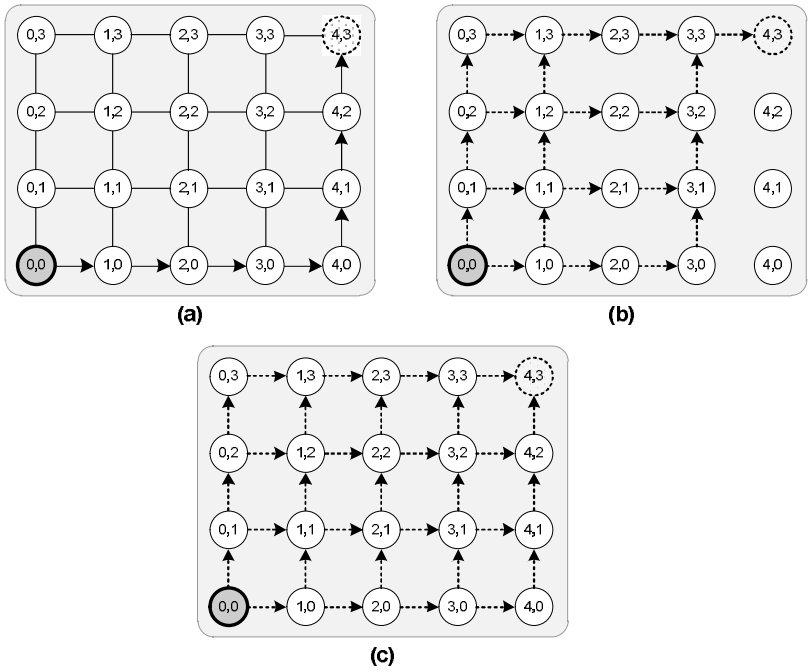


Fig. 3-1. Illustration of different routing schemes.

Except for the boundary routers, XY routers have five input/output ports (four connected to the neighboring routers and one for the local core). Main architectural elements of an XY router include the input FIFO for each port, route computation unit, Virtual Channel (VC) allocation unit (if any), crossbar control logic, and the crossbar. To minimize the delay and the required resources, the wormhole method is used for the switching. A flit enters into the router through one of the ports and is stored in its FIFO. If the flit is a header, indicating the start of a new packet, it proceeds to the routing unit, which determines the output port that the packet should use. The header flit attempts to acquire a channel (maybe virtual) for the next hop. Upon a successful channel allocation, the header flit enters the router arbitration stage, where it competes for the output port with other flits from the other input ports. Once the crossbar passage is granted, the flit traverses the router and enters the channel. Subsequent flits belonging to the same packet can proceed directly to the crossbar and go to the output port. The main architectural element of an XY router is shown in Fig. 3-2.

3.1.2 DyAD Routing Scheme

A partial adaptive routing algorithm, named Odd–Even turn model, is proposed in [64]. The Odd-Even turn model prohibits the east to north and east to south (north to west and south to west) turns at any routers located in an even (odd) column. This makes the technique as a partial adaptive deadlock-free scheme employing minimum paths.

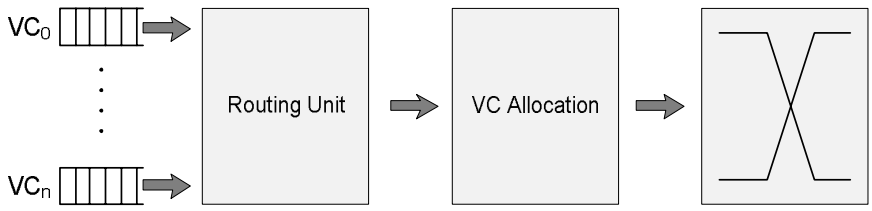


Fig. 3-2. Structure of the XY router.

Since it is deadlock-free, there is no need for implementing virtual channels in the router to prevent the deadlock problem but virtual channels can be employed to gain the performance. Considering the Odd-Even example, illustrated in Fig. 3-1(b), all the possible minimal routing paths (10 paths) for packets from source node (0, 0) to destination node (4, 3) have been exhibited. DyAD, dubbed from Dynamic Adaptive Deterministic switching, is a partial adaptive routing scheme, based on the Odd-Even [65]. It is a combination of deterministic (XY) and adaptive (Odd–Even) routing schemes [65]. Depending on the congestion condition of the network, one of the routing schemes is invoked. More precisely, when the network is not congested, the DyAD router works in a deterministic mode, and when the network becomes congested, the DyAD router uses the adaptive routing mode and thus avoids the congested links by exploiting other routing paths [65]. Hence, the main difference between the DyAD/Odd-Even and the XY routers is that, depending on the network condition, the routing unit may select different paths at different times for the same source and destination pair. For this to happen, a pre-port selection unit is added to the router to select the best candidate for every adjacent ports (i.e., North vs. East, North vs. West, South vs. East, and South vs. West) and provide routing unit this information (Fig. 3-3). One of the factors for choosing an output port is the number of free buffers at the corresponding input port in the next hop [66]. This technique has been used in the several adaptive routing schemes where the free buffer count at a downstream node is used for congestion estimation. To transfer the count information, which can be considered as a stress value, some wires are added between adjacent routers.

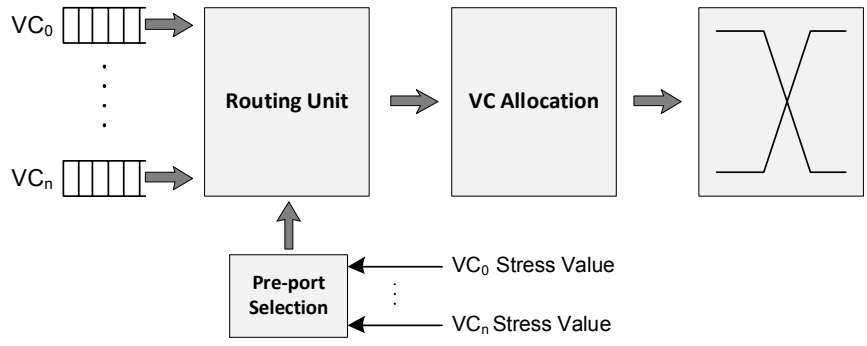


Fig. 3-3. Structure of the DyAD router.

3.1.3 DyXY Routing Scheme

As addressed earlier, routing packets along any shortest path are called fully adaptive. The Dynamic XY (DyXY) routing scheme is a well-known minimal fully adaptive routing algorithm [2][3][67]. Due to the fact that in each node, packets can be routed in both X and Y directions without restriction, this routing algorithm needs a mechanism to guarantee deadlock avoidance. In networks having virtual channels (general case), usually the following method is used to guarantee deadlock avoidance. Virtual channels in Y dimensions are divided into two parts, thereby, as illustrated in Fig. 3-4, The network is partitioned into two subnetworks called +X subnetwork and −X subnetwork each having half of the channels in the Y dimension.

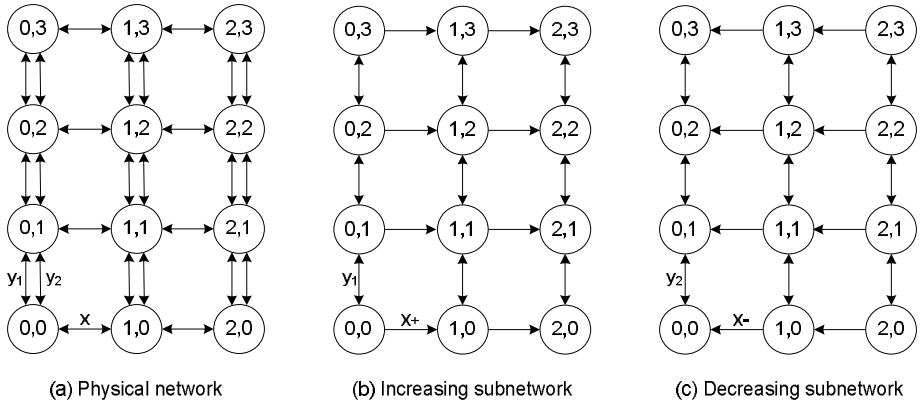


Fig. 3-4. (a) A 3×4 mesh physical network and the corresponding (b) increasing and (c) decreasing subnetworks.

If the destination node is to the right of the source, the packet will be routed through the +X subnetwork. If the destination node is to the left of the source, the packet will be routed through the −X subnetwork. Otherwise that packet can be routed using either subnetwork [68], thus, DyXY is deadlock-free. Inasmuch as the subnetworks are acyclic, packets can be adaptively routed along shortest paths meaning at an intermediate node, a packet may be routed along either dimension. Fig. 3-1(c) also shows the possible minimal paths of DyXY, i.e. $\frac{(3+4)!}{3!\times 4!} = 35$ possible paths [64]. The router structure of DyXY is identical to DyAD router shown in Fig. 3-3.

Now, we discuss a weakness of the DyXY algorithm in routing packets from routers whose X position (Y position) is one unit apart from that of the destination. Let us consider the simple example shown in Fig. 3-5 where (1, 2) and (4, 1) are the source and destination routers. In the DyXY routing, router (1, 2) compares the current length of the west queue of router (2, 2) and the north queue of router (1, 1) and the packet is sent to the direction containing more empty buffer space. Note that if router (1, 1) is selected, the entire path has been determined and hence routers (2, 1) and (3, 1) are the next hops without any choice. If

they are congested, the DyXY algorithm has selected a wrong path-based on local information. However, if router (2, 2) is selected, there are other choices for the next hops, and hence, the congestion may be avoided. Note that this undesired effect may occur when routing from a router whose X (or Y) position is just one unit apart from that of the destination and the congestion is taking place further away from the current hop.

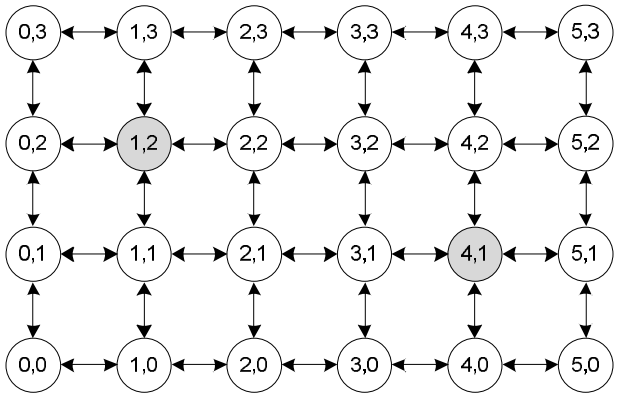


Fig. 3-5. A simple NoC with mesh structure.

3.1.4 EDXY Routing Scheme

The objective of the proposed routing scheme, named Enhanced Dynamic XY (EDXY), is to solve the problem of enhance the DyXY algorithm. This is achieved by using a flag which indicates congestion along the path of a row (or column). This flag propagates in a row (or column) and indicates to the adjacent rows (or columns) that this row (or column) is near saturation and should be avoided. As congestion flag should propagate along a row (or column), each router transparently propagates its prior router congestion flag. Besides, each router monitors its input buffers; if the number of occupied slots of a buffer is larger than a threshold value, the router will activate its congestion flag. To track the congestion condition efficiently, two flags are added to each row (or column): one informs left hand side routers of congestion in a right hand side router and the other one informs right hand side routers of congestion in a left hand side router in the row. For this purpose, between every two adjacent routers, two congestion wires, one in each direction, are added. The congestion wires are grouped with wires which are employed to transmit the free buffer count (stress value) to the adjacent routers. The congestion flag transmitted by a router is obtained by ORing its flag and that of the previous router (to transparently propagate prior routers congestion flag). Therefore, if the flag is active, it shows that either the current or at least one of the previous routers is congested.

In EDXY routing, if a router is one hop apart from the destination in a row (or column), the congestion flag in the destination column (or row) is used in routing decision. If the congestion flag is one, the algorithm uses the other path to route the packet. For the example shown in Fig. 3-5, the congestion wire in the destination row is passed to router (1, 2) by router (1, 1). Therefore, the packet will be routed to router (2, 2).

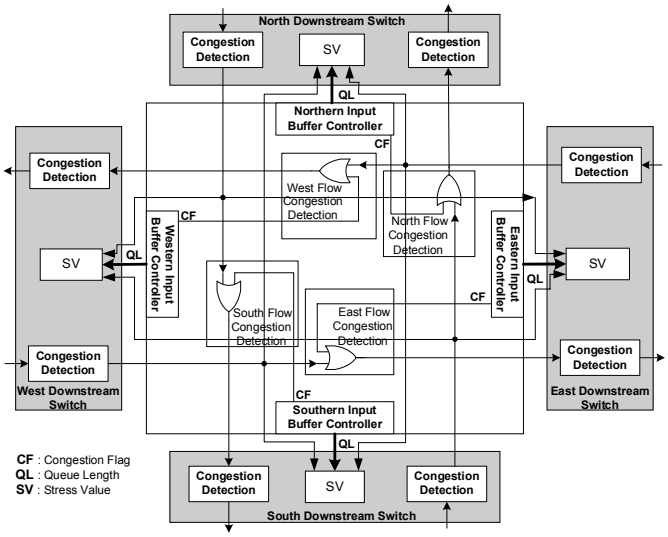


Fig. 3-6. An EDXY router implementation.

In EDXY, every router first looks at the destination address of the received packet. If the router address is not one hop apart from that of the destination in either the X or Y direction, the EDXY algorithm ignores the congestion wire and routes the packet the same way as the DyXY algorithm does. However, if the destination address is just one hop apart from the router in either the X or Y direction, one of the congestion wires (based on the position of the destination) is also used for routing. That is, the congestion value is employed as the most significant bit of the stress value. In these cases, we refer to the port which is one hop apart from the destination in either X or Y directions as critical port while the other port is called non-critical port. For the non-critical port, we ignore congestion flag value and a zero is used for the congestion value to favor this port against the other port.

It should be noted that congestion wires may report false information because they do not provide information regarding the location of the congestion node in a row (or column). We compare shared congestion wires with the general case in which the position of the congestion node is also propagated in a row or column. The results show only slight differences between these two mechanisms. We proposed shared congestion wires because they are simple and have the desired characteristics for reducing latency. In cases where the congestion wire should be used in the decision making process, a function of the queue length and the congestion wire value can be used to return the stress value. In order to

simplify the hardware implementation, we used the congestion wire as the MSB of the queue length to form the stress value.

$X_{diff} = X_c - X_d$
$Y_{diff} = Y_c - Y_d$
If (one destination port) **Then**
    Use that port to route the packet;
Else
    If ($ABS(X_{diff}) = 1$) **Then**
        If ($Y_{diff} > 0$) **Then**
            $S\_Value_X = \{I\_queue\_length_X, C\_Wire_{UpToDown}\}$;
        Else
            $S\_Value_X = \{I\_queue\_length_X, C\_Wire_{DowntoUp}\}$;
        End If;
    Else
        $S\_Value_X = I\_queue\_length_X$;
    End if;
    If ($ABS(Y_{diff}) = 1$) **Then**
        If ($X_{diff} > 0$) **Then**
            $S\_Value_Y = \{I\_queue\_length_Y, C\_Wire_{RightToLeft}\}$;
        Else
            $S\_Value_Y = \{I\_queue\_length_Y, C\_Wire_{LeftToRight}\}$;
        End If;
    Else
        $S\_Value_Y = I\_queue\_length_Y$;
    End If;
    Use S_Value_X and S_Value_Y to choose the destination port.
End If;

Fig. 3-7. EDXY routing algorithm.

For the implementation of the EDXY routing algorithm, an extra hardware should be added to the DyXY router. This extra hardware is divided into two parts. An extra unit is needed in each router to drive the congestion signals in four directions. In addition, each router should have logic to use the congestion signals in the routing decisions. In each direction, the output congestion wire is set either if the input congestion signal due to the congestion in the previous routers in that direction is set or the occupied part of the input buffer for routing in that direction is larger than the threshold value. For implementing this logic, a comparator and an OR gate are used. The comparator is used to compare the queue length with the predefined threshold value. In the case of exceeding the threshold value, the output of the comparator, integrated inside the input controller, becomes one. As shown in Fig. 3-6, the output congestion signal is the result of ORing the output of the comparator and the input congestion signal. The routing algorithm should also be modified to use the new information (i.e., congestion wires) in routing decisions. The EDXY routing algorithm is shown in Fig. 3-7. Compared to the DyXY routing architecture, extra hardware (SV) is

needed to produce the stress value for two competing ports from their queue length and the values of the congestion signals.

3.1.5 Experimental Results

For assessing the efficiency of EDXY, three other routing algorithms were also implemented. These algorithms included the XY, Odd–Even turn-model (DyAD), and DyXY. A NoC simulator was developed in VHDL to model all major components of the on-chip network and simulations were carried out to determine the latency characteristic of each network. For all the routers, the data width was set to 32 bits. Each input virtual channel had a buffer (FIFO) with the size of 6 flits. The congestion threshold value (for EDXY routing) was set to 4 meaning that the congestion condition was considered when 4 out of 6 buffer slots were occupied.
As a performance metric, we used latency defined as the number of cycles between the initiation of a message transmission issued by a PE and the time when the message is completely delivered to the destination PE. The request rate is defined as the ratio of the successful message injections into the network interface over the total number of injection attempts. The simulator was warmed up for 3,000 cycles and then the average performance was measured over another 100,000 cycles.
The router used the minimal fully adaptive reserved VC deadlock avoidance technique discussed in [68]. Four synthetic traffic profiles of transpose, uniform random, hotspot 5%, and hotspot 10% and SPLASH-2 benchmark traces were used. Table 3-1 shows the baseline network configuration, and the variations used in the sensitivity studies.

Table 3-1. Baseline network configuration and variation.

Characteristics	Baseline	Variations
Topology	7 × 7 2D Mesh	15 × 15 2D Mesh
Routing	XY, DyXY, and EDXY	Odd-Even
Virtual channels/port	2	0
Flit buffers/VC	6	-
Packet length (flits)	9	15
Traffic workload	Transpose, uniform, hotspot	SPLASH-2 traces
Simulated packets/node	3000	-

A. First set of experiments

In the first set, 7 × 7 2D meshes and packets with a length of 9 flits were used. The average packet latency for different traffic profiles are shown in Fig. 3-8.

Transpose traffic profile

In the transpose traffic profile, for a $n \times n$ mesh network, a core at position $(i,j)(i,j\epsilon[0,n))$ only sends a data packet to another core at position $(n-1-i,\ n-1-j)$. This traffic pattern is similar to the concept of transposing a matrix [64][65]. In these simulations, each core generates packets and injects them into the network using the time intervals determined using the exponential distribution. This traffic profile leads to a non-uniform traffic distribution with heavy traffic for the central nodes of the mesh. Therefore, close to the center of the network hotspots may be created. As shown in Fig. 3-8(a), if the data packet injection rate is very low, hotspots are not created and the routing scheme behave similarly. As the injection rate increases and congestion is created in the mesh, the EDXY algorithm leads to smaller average delays.

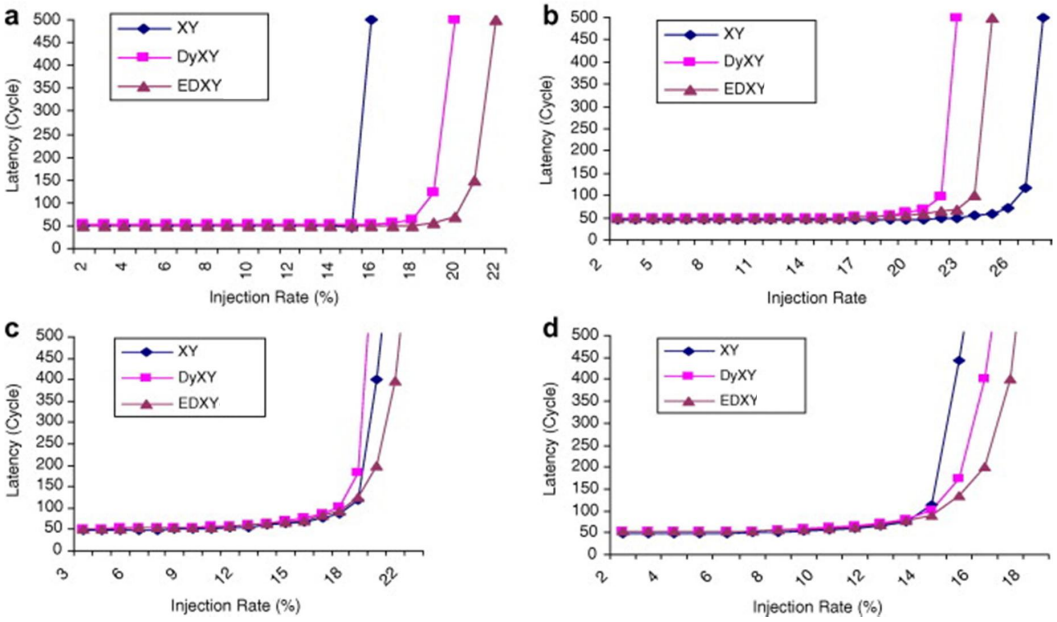


Fig. 3-8. Latency vs. packet injection rate for EDXY, DyXY, and XY for a 7 × 7 2D mesh for 9-flit packets with virtual channel. (a) transpose traffic, (b) uniform random traffic, (c) hotspot 5%, and (d) hotspot 10%.

Uniform traffic profile

In this traffic profile, each node sends several messages to other nodes in the network where a uniform distribution is used to construct the destination set of each message [2][64][65]. The average communication delay as a function of the average message injection rate has been plotted in Fig. 3-8(b). For this case, the XY scheme leads to lower latencies because uniform traffic is balanced under XY routing [69].

Hotspot traffic profiles

The same as the uniform traffic profile, each node sends several messages to other nodes in the network while nodes (2, 2), (2, 4), (4, 2), and (4, 4) receives 5% and 10% more packets in hotspot 5% and 10% traffic profiles, respectively. The average communication delays as a function of average packet injection rate for hotspot 5% and 10% traffic profiles have been plotted in Fig. 3-8(c) and (d), respectively. As observed from these figures, the proposed scheme has lower delay compared to other schemes for both hotspot percentages. This reveals that the proposed scheme can distribute the traffic among minimal paths more efficiently.

Table 3-2. System configuration parameters.

Processor Configuration	
Instruction set architecture	SPARC
Number of processors	16
Issue width	1
Cache configuration	
L1 cache	Private, split instruction and data cache, each cache is 16KB. 4-way associative, 64-bit line, 3-cycle access time
L2 cache	Shared, unified 48MB (48 banks, each 1MB). 64-bit line, 6-cycle access time
Cache coherence protocol	MESI
Cache hierarchy	SNUCA
Memory configuration	
Size	4GB DRAM
Access latency	260 cycles
Requests per processor	16 outstanding
Network configuration	
Router scheme	Wormhole
Flit size	32 bits

SPLASH-2 benchmark traffic

In order to know the real impact of EDXY, traces are generated from SPLASH-2 [87] using the GEMS simulator [90]. We configured a 64-node on-chip network which models a single-chip CMP for our experiments. A full system simulation environment with 16 processors and 48 L2 cache nodes has been implemented. The simulations were run on the Solaris 9 operating system based on SPARC instruction set in-order issue structure. Each processor is attached to a wormhole router and has a private write-back L1 cache. The L2 cache shared by all processors is split into banks. The size of each cache bank node is 1MB. Hence, the total size of shared L2 cache is 48MB. The simulated memory/cache

architecture mimics static non-uniform cache architecture (SNUCA) [91][92] while the cache coherence protocol for generating the traces is MESI [91]. The detailed configurations of processor, cache and memory configurations can be found in Table 3-2.
Fig. 3-9 shows the average packet latency across five SPLASH-2 benchmark traces, normalized to XY. Contention is the cause of significant packet latency in *lu*, *fft*, and *raytrace*; thus adaptive routing has an opportunity to improve performance. Although EDXY provides equal or lower latency than other schemes, EDXY shows the greatest benefit on *raytrace* with 36% reduction in latency. On average EDXY provides a latency reduction of 20% across all benchmarks vs. XY and 12% vs. DyXY.

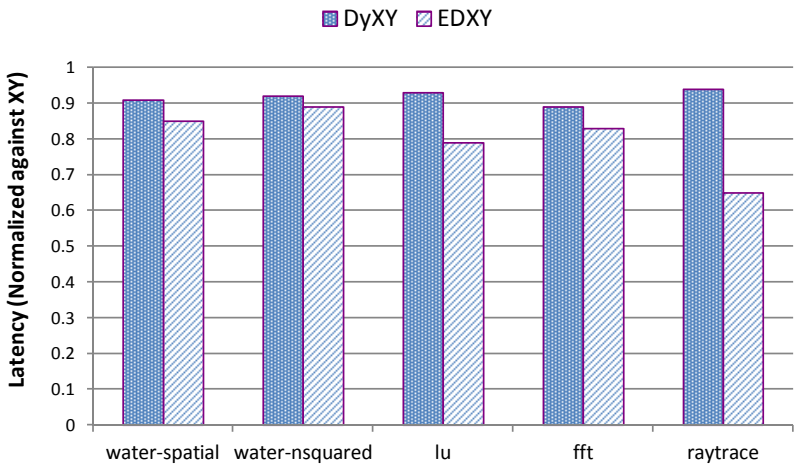


Fig. 3-9. Average latency across SPLASH-2 benchmarks normalized to latency of XY.

B. Second set of experiments

In this section, we change some of the NoC parameters considered in the first set of experiments.

NoC size

Fig. 3-10 shows the latency vs. the packet injection rate for a 15 × 15 mesh NoC under the transpose traffic profile with 9-flit packets. For this large network, adaptive approaches do not perform as well as XY. That is, when the network size increases, the effect of the congestion information reduces.

Packet length

Fig. 3-11 shows the latency for long packets (15 flits) in a 7 × 7 2D mesh under different traffic profiles. As can be seen, the average packet latencies of all routing schemes using long packets are higher than other schemes with short packets. The increased average latency is a known characteristic of wormhole routing with long packets. The reason is that

for long packets, an imbalance in the resource utilization occurs because packets hold resources over multiple routers. Similar to the case of 9-flit packets, the EDXY scheme performs better than DyXY and XY over all traffic patterns except for the uniform traffic profile.

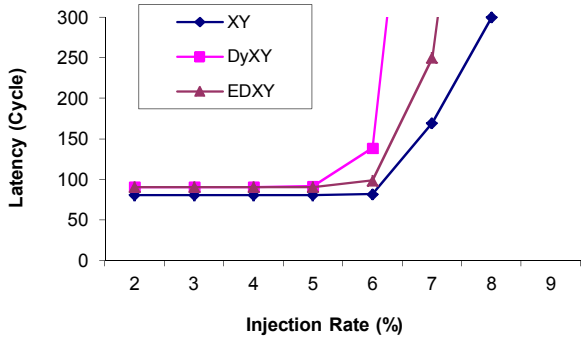


Fig. 3-10. Latency vs. packet injection rate for 15 × 15 mesh with virtual channel under transpose traffic profile using 9-flit packets.

Network without virtual channel

Fig. 3-12 shows latency vs. packet injection rate for Odd–Even, DyXY, and EDXY routers without virtual channels. In this case, we consider a 5 × 5 2D mesh with 5-flit packets. The buffer size of each channel is 6 flits. Odd–Even turn-model [64] is a technique for avoiding deadlock in NoCs without virtual channel. In this section, for the deadlock avoidance in DyXY and EDXY, Odd–Even turn-model is used. Actually, when the Odd–Even provides more than one output ports, in the Odd–Even router, the port in the Y direction is chosen while in the DyXY and EDXY routers, the stress value is examined to choose one of the ports. For these simulations, core (2, 2) receives 5% and 10% more packets in hotspot 5% and 10% traffic profiles, respectively. As the results show, EDXY continues to perform better than other routing algorithms in a network without virtual channel with transpose, uniform, hotspot 5%, and hotspot 10% traffic profiles.

C. Hardware overhead

To evaluate the area overhead of EDXY, the VHDL reference model was synthesized with Synopsys Design Compiler using a standard cell CMOS library. For all routers, the data width was set to 32 bits (flit size), and each channel had two virtual channels with a buffer size of 6 flits. In order to achieve better performance/power characteristics, the FIFOs were implemented using registers.

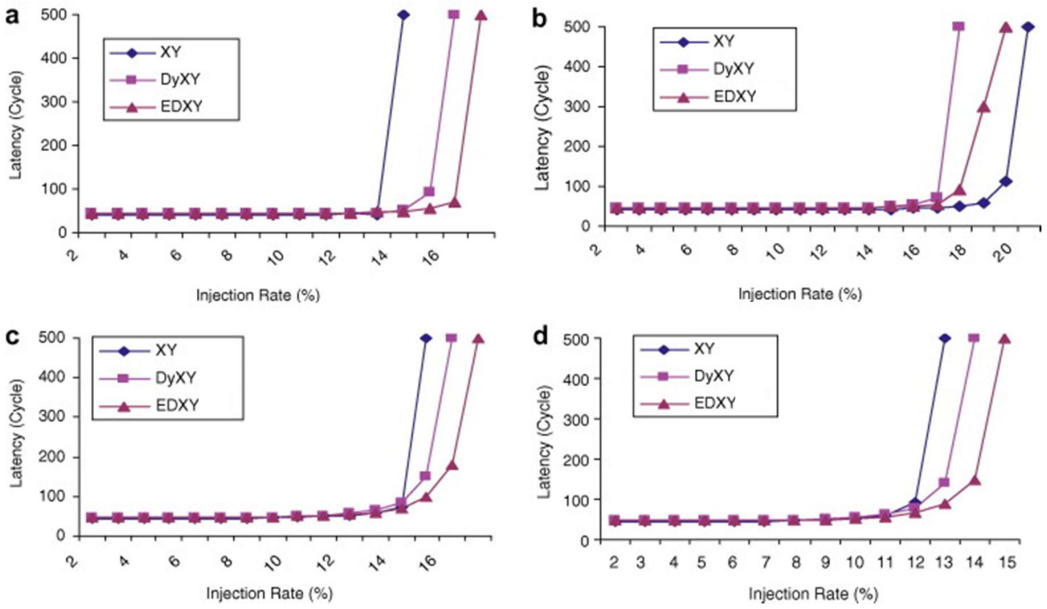


Fig. 3-11. Average latency vs. packet injection rate on a 7 × 7 2D mesh for 15-flit packets with virtual channel. (a) transpose traffic, (b) uniform random traffic, (c) hotspot 5%, and (d) hotspot 10%.

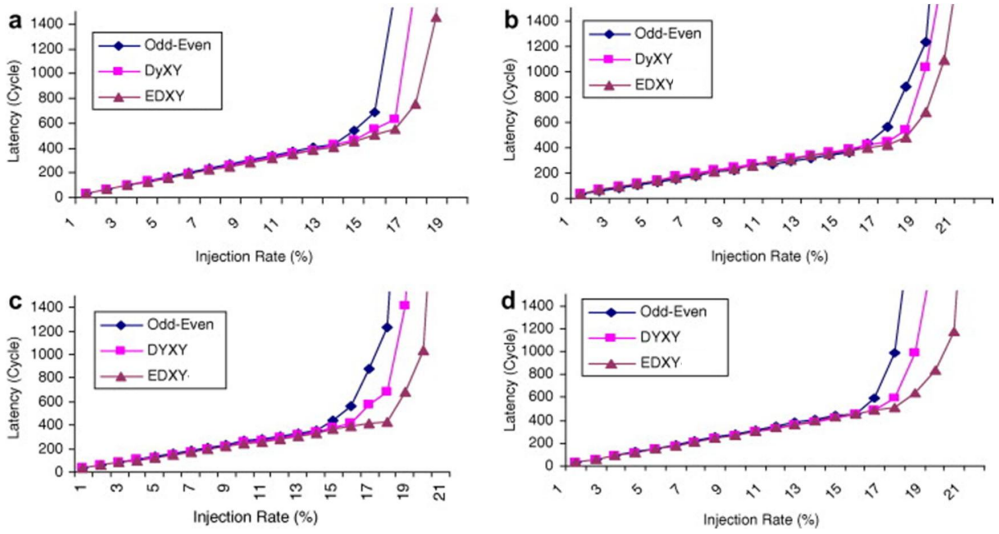


Fig. 3-12. Latency vs. packet injection rate on a 5 × 5 2D mesh without virtual channel. (a) transpose traffic, (b) uniform random traffic, (c) hotspot 5%, and (d) hotspot 10%.

Table 3-3 compares the areas of the routers. As the table reveals, the area overhead of the EDXY compared to that of the XY (DyXY) is 3.6% (1.5%).

Table 3-3. Area comparison of XY, DyXY, and EDXY.

	XY	DyXY	EDXY
Area (μm^2)	86,107	87,881	89,281

D. Power dissipation

The power dissipation of EDXY and DyXY routing algorithms were calculated and compared under the uniform traffic profile using Synopsys PrimePower. Each core in the NoC generated packets based on the uniform random traffic profile with the same average flit injection rate. The typical clock of 1 GHz is applied to the system. Since the post synthesis simulation is very slow, router (3, 3) which is close to the center of a 7 × 7 2D mesh was synthesized while for the other routers, the RTL models were used. The results for the power consumption are given in Table 3-4. As observed from this table, the power consumptions of both schemes are about the same.

Table 3-4. Power consumption of DyXY and EDXY routing under the uniform traffic profile (mW).

	DyXY	EDXY
Power consumption (mW)	27.3	27.7

3.2 Multicast Routing Protocols

The multicast communication has been exploited in multicomputers (see, e.g., [70][71][72][73][74]). Multicast routing algorithms can be classified as unicast-based [75][76], tree-based [76], and path-based [74].

3.2.1 Unicast-based Multicast Routing

Unicast-Based (UB) is a simple multicast routing algorithm where multiple copies of the same message, as a unicast message, are routed independently toward every destination or to a subset of destinations [75]. The drawback of this scheme is that multiple copies of the same message are injected into the network, increasing the network traffic. Furthermore, each copy of the message suffers from considerable startup latency at the source.

3.2.2 Tree-based Multicast Routing

In tree-based multicast routing approach, the destination set is partitioned at the source and separate copies of the message are sent through one or more outgoing channels. Here, a spanning tree is constructed where the source is considered as the root and the messages are

sent down the tree [76]. This way, a message might be replicated at some of the intermediate nodes and forwarded along the multiple outgoing channels toward disjoint subsets of destinations. Since there is no message buffering at routers, if one branch of the tree is blocked, all are blocked [77]. If the message is not proceeded forward, many channels may be in lockstep for extended periods, resulting in increased network contention [77]. Although the tree-based multicasting scheme can be used efficiently in networks employing store-and-forward and virtual cut-through switching, it incurs high congestion in wormhole networks [74]. A tree-based routing algorithm which supports multicasting in NoCs is called virtual circuit tree multicasting (VCTM) [59]. By using virtual circuit table (VCT) and content addressable memory (CAM), and sending separate unicast setup messages (look ahead signals) for each destination, it builds several virtual circuit trees toward the destinations before the multicast messages are injected into the network. The method, however, has some shortcomings. First, its complexity, and hence, hardware overhead strongly depends on the network size. Second, the VCTM is an efficient algorithm mostly for low injection rate network conditions while for high injection rate conditions (or workloads near saturation), the path-based algorithms are more efficient [59]. Third, for updating the virtual circuit table, discrete unicast setup messages per destination should be sent by the source node. If the number of destinations grows, the number of unicast setup messages will be increased, thereby reducing the performance. Therefore, the VCTM scheme is more efficient for applications using a small percentage of multicasts [59]. An example includes token coherence protocol which uses one-to-all communication and has a very few distinct multicast combinations [59]. Finally, the tree-based multicasting may cause a message to hold many channels for extended periods, thereby increasing network contention, and hence, degrading the performance [60]. However, in this approach, cyclic dependencies are avoided by using the dimension-order routing algorithm for each pair of source and destination nodes [59][60].

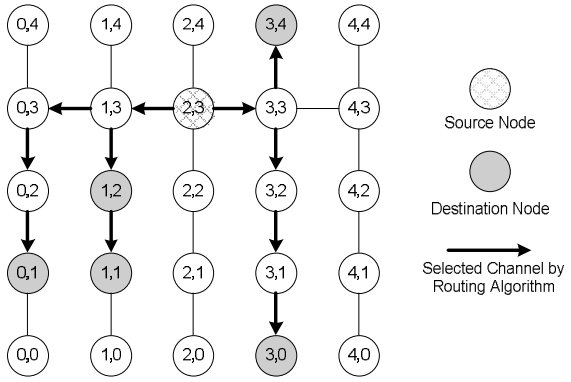


Fig. 3-13. Example of tree-based multicast routing in 5×5 2D-mesh.

An example of a tree-based multicast routing in 5×5 2D-mesh is shown in Fig. 3-13 where the source node (2, 3) is selected as the root and a spanning tree is formed with respect to it.

When the flits enter the routers at the branch point (nodes (1, 3) and (3, 3)), they are duplicated and forwarded to multiple output channels. Since there is no message buffering in the routers, if one branch of the tree is blocked, all are blocked. Therefore, this scheme might lead to increased network contention.

3.2.3 Hamiltonian Path-based Multicast Routing Algorithm

To overcome the disadvantages of the tree-based approaches, one may use path-based multicast wormhole routing algorithms. In this method, a source node prepares a message for delivery to a set of destinations by first sorting the addresses of the destination in the order in which they are to be delivered, and then placing this sorted list in the header of the message. When the header enters a router with the address A, the router checks to see if A is the next address in the header. If so, the address A is removed from the message header and a copy of data flits will be delivered to the local core and the flits are forwarded to the next node on the path. Otherwise, the message is forwarded only to the next node on the path. In this way, the message is eventually delivered to every destination in the header. A number of studies have shown that a path-based exhibit superior performance characteristic over their unicast-based and tree-based counterparts [77][78][79].

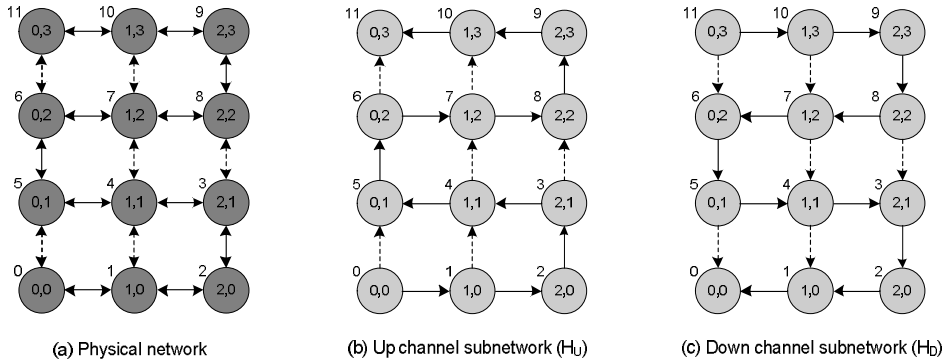


Fig. 3-14. A 3×4 mesh physical network with the label assignment and the corresponding [78] (b) up channel and (c) down channel networks. The solid lines indicate the Hamiltonian path and dashed lines indicate the links that could be used to reduce the path length in routing.

The path-based routing algorithms are based on Hamiltonian path where a undirected Hamiltonian path of the network is constructed [74]. A Hamiltonian path visits every node in a graph exactly once [80]. For each node in an $m \times n$ mesh, a label $L(x, y)$ is assigned as

$$L(x,y)=\begin{cases} y \times n + x & \text{if } y \text{ is even} \\ y \times n + n - x - 1 & \text{if } y \text{ is odd} \end{cases}$$

where x and y are the coordinates of the node.

As exhibited in Fig. 3-14, two directed Hamiltonian paths (or two subnetworks) are constructed by labeling the nodes [74]. The up channel subnetwork (H_U) starts at (0, 0) while the down channel subnetwork (H_D) ends at (0, 0). If the label of the destination node is greater than the label of the source node, the routing always takes place in the H_U subnetwork; otherwise, it takes place in the H_D subnetwork. The destinations are placed into two groups. One group contains all the destinations that could be reached using the H_U subnetwork and the other contains the remaining destinations that could be reached using the H_D subnetwork. To reduce the path length the vertical channels that are not part of the Hamiltonian path (the dashed lines in the Fig. 3-14) could be used in appropriate directions. In fact, if in a routing algorithm all packets in the up channel (down channel) subnetwork follow paths in strictly ascending (descending) order (either in Hamiltonian path or not), no cyclic dependency can be formed among channels; thus the routing algorithm is deadlock-free.
Next, dual-path (DP) [74], multi-path (MP) [74], and column-path (CP) [78] multicast routing algorithms along with the proposed multicast routing schemes, LD and HAMUM, are described.

A. Dual-Path (DP) and Multi-Path (MP) Multicast Routing Algorithms

In Dual-Path (DP) routing algorithm, the destination node set is partitioned into two subsets of D_U and D_D [74]. Every node in D_U has a higher label than that of the source node and every node in D_D has a lower label than that of the source node. D_U and D_D are then sorted in ascending order and descending order, respectively, as the label of each node is used as the key for the sorting. Thus, multicast messages from the source node will be sent to the destination nodes in D_U using the H_U subnetwork and to the destination nodes in D_D using the H_D subnetwork. Consider the example shown in Fig. 3-15(a) for a 6×6 mesh network where node (2, 3) will send its multicast messages to destinations (2, 0), (4, 0), (0, 1), (2, 1), (4, 1), (0, 4), (5, 4), (3, 5), and (5, 5). Two subsets are organized. The first subset (D_U), which contains all the destinations that could be reached from the source node using H_U subnetwork, includes (0, 4), (5, 4), (5, 5) and (3, 5) in sequence. The second subset (D_D), which has the remaining destinations that all could be reached using the H_D subnetwork, includes (2, 0), (4, 0), (4, 1), (2, 1) and (0, 1). Some of the vertical links that are not part of the Hamiltonian paths are used properly, for minimizing the paths.
To reduce the path lengths, the multi-path (MP) multicast routing algorithm has been proposed in [74]. In this scheme, as most nodes have four output channels in the 2D mesh, up to four independent paths can be used to deliver a message. Thus, the dual-path destination sets of D_U and D_D are also partitioned. The set D_U is divided into two subsets. One consists of the nodes whose x coordinates are greater than or equal to that of the source and the other subset contains the remaining nodes in D_U. The set D_D is partitioned in a similar way. Hence, all the destinations of the multicast message are grouped into four disjoint subsets such that all the destinations in a subset are in one of the four quadrants when the source is taken as the origin. For the multi-path example shown in Fig. 3-15(b), at

source (2, 3) the destination set is first divided into two sets of D_U = {(0, 4),(5, 4),(3, 5),(5, 5)} and D_D = {(2, 0),(4, 0),(4, 1),(2, 1), (0, 1)}. As exhibited in Fig. 3-15(a), D_U is divided into two subsets of D_{U1} = {(0, 4)} and D_{U2} = {(5, 4), (3, 5), (5, 5)}. In the same way, D_D is divided into two subsets of D_{D1} = {(0, 1), (2, 1), (2, 0)} and D_{D2} = {(4, 0), (4, 1)}. The dual-path and multi-path are both deadlock-free and could be used for unicast and multicast routing simultaneously [74].

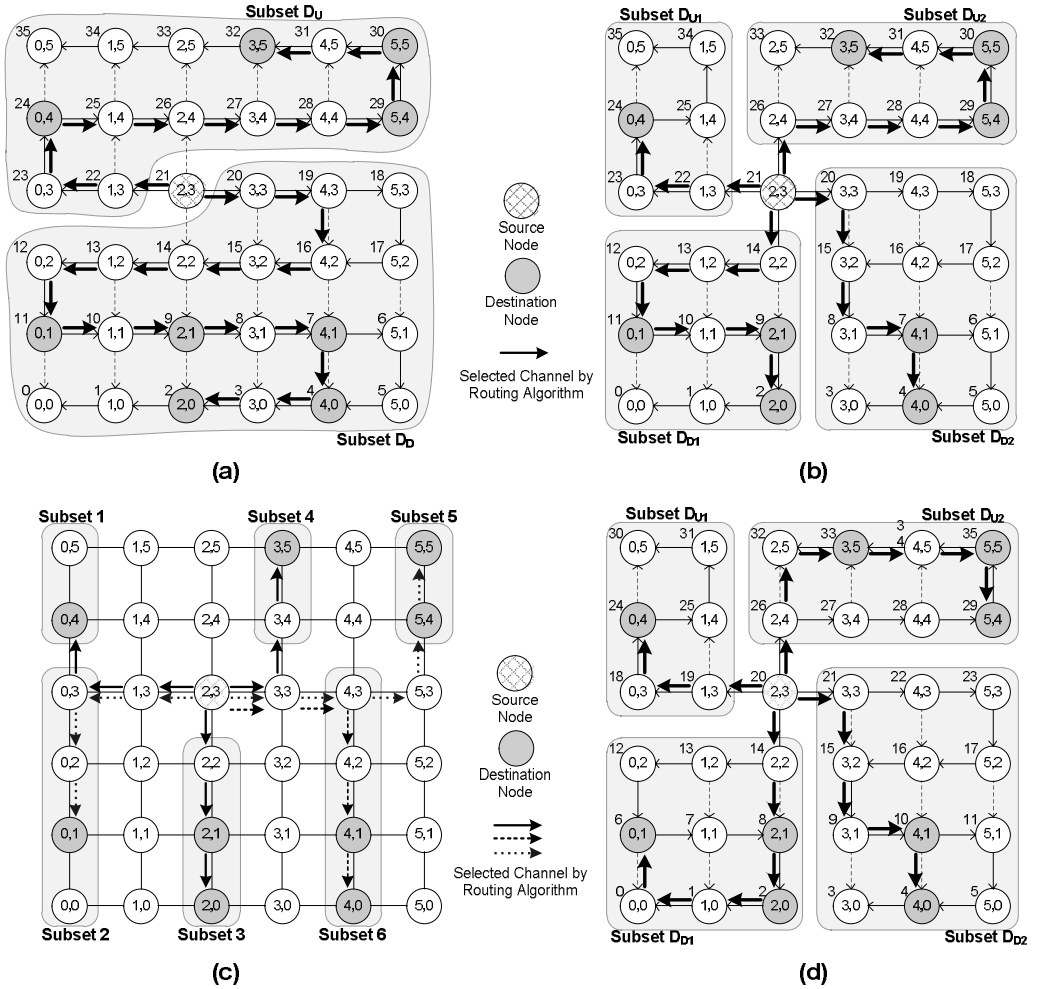


Fig. 3-15. Examples of (a) Dual-path (DP), (b) Multi-path (MP), (c) Column-Path (CP), and (d) Low-Distance (LD) multicast routing from (2, 3). The unused links are not indicated.

B. Column-Path (CP) Multicast Routing

In this method, the destination node set is partitioned into $2k$ subsets where k is the number of columns in the mesh. In this method, at most two messages will be copied to each column. If a column of the mesh has one or more destinations in the rows above that of the source, then one copy of the message is sent to service all those destinations. Similarly, if a column has one or more destinations in the rows below that of the source, then another copy of the message is sent to service all those destinations. One copy of the message is sent to a column if all destinations in that column are either below or above the source node. Fig. 3-15(c) shows an example where a multicast message is sent to destinations (2, 0), (4, 0), (0, 1), (2, 1), (4, 1), (0, 4), (5, 4), (3, 5), and (5, 5) from source node (2, 3) using the column-path (CP) routing algorithm. Six copies of the message are used to achieve the desired multicast operation. The routing algorithm used by this scheme is based on the XY routing algorithm which is deadlock-free and livelock-free. However, since the CP routing, similar to the unicast-based routing method, produces too many messages (i.e. at most $2k$ copies of the message in the CP routing and at most 4 and 2 in the MP, and DP routings, respectively), it suffers from high network latencies for latter copies of the messages due to the excessive number of start-up delays before them. In addition, because many multicast messages would be sent through the columns by each source node, the performance of the network is degraded.

C. Low-Distance (LD) Path-based Multicast Routing

In this part, the proposed adaptive path-based multicast routing, LD, is described. Three features have been incorporated in this scheme.

1) It utilizes a network partitioning similar to multi-path multicast routing technique where up to four destination groups could be formed.
2) The ordering of the destinations in the path should be optimized to shorten the distance of the multicast path. This is achieved at the cost of a small hardware overhead and improves the performance of the algorithm compared to those of previous path-based multicast routing algorithms. For this propose, a sorting algorithm shown in Fig. 3-16 is proposed. In this algorithm, for each node a label obtained from $L(x, y) = y \times n + x$ is assigned. Similar to the multi-path multicast algorithm, the destination node set is partitioned into four subsets of D_{U1}, D_{U2}, D_{D1}, and D_{D2}. The subsets are then sorted in the low-distance order with the distance vector of each node used as its key for the sorting. The distance vector of each node is computed as $k = |y - y_0| + |x - x_0|$. To sort the destinations to the low-distance order, first the node (v) which has the lowest distance vector to the source node (u_0) is placed in the *Temp_set* and is removed from the subset. Then, the selected node will be considered as the source node. While the original subset is not empty, this sequence will be repeated; otherwise, the *Temp_set* which contains the sorted destination subset is placed in the original subset. If there are two nodes with an equal distance vector compared to the source node, the one with the smaller x

dimension relative to that of the source node will be selected first. Subsequently, the original updated subset will be placed in the message header.

3) For routing the messages to the destinations, the algorithm utilizes the Odd-Even turn model [64][65]. The Odd-Even turn model prohibits the east to north and east to south (north to west and south to west) turns at any routers located in an even (odd) column. This makes the technique as an adaptive deadlock-free algorithm which uses the shortest path. Since it is deadlock-free, there is no need for implementing virtual channels in the router to prevent deadlock. Adding virtual channels is costly since the complexity and latency of the controller increase with the number of virtual channels due to increased buffering and arbitration requirements [68].

Algorithm: Ordering and partitioning the destination set

Inputs: Destination set D; source node (x_0, y_0); distance table T;

Outputs: Sorted destination sets D_{U1}, D_{U2}, D_{D1}, D_{D2} for 4 multicast paths.

Begin

1. *For every node assign a label as:* $L(x, y) = y \times n + x$
2. $D \rightarrow \{D_U, D_D\};\ D_U = \{(x,y) \mid L(x,y) > L(x_0,y_0)\};\ D_D = \{(x,y) \mid L(x,y) < L(x_0,y_0)\};$
3. $D_U \rightarrow \{D_{U1}, D_{U2}\};\ D_{U1} = \{(x,y) \mid x < x_0, y \geq y_0\};\ D_{U2} = \{(x,y) \mid x \geq x_0, y > y_0\};$
   $D_D \rightarrow \{D_{D1}, D_{D2}\};\ D_{D1} = \{(x,y) \mid x \leq x_0, y < y_0\};\ D_{D2} = \{(x,y) \mid x > x_0, y \leq y_0\};$
4. *For sorting* D_{U1} *in Low-distance order:*
   While D_{U1} *is not empty do the following:*
   Begin
   (a) $u = u_0;\ Temp\_set = \{\phi\};$
   (b) *Find the node v with smallest* $k(x,y)$ *value in* D_{U1};
   $(k(x,y) = |y - y_0| + |x - x_0|$ *is distance vector from* (x,y) *to* $(x_0, y_0))$
   if $k(x_1,y_1) = k(x_2,y_2)$ *then*
   if $|x_0 - x_1| < |x_0 - x_2|$ *then* (x_1,y_1) *is selected first;*
   else (x_2,y_2) *is selected first;*
   (c) *Add node v to Temp_set; Remove node v from* D_{U1};
   (d) $u = v$;
   End.
   D_{U1} = *Temp_set;*
   Do the same algorithm for sorting D_{U2}, D_{D1} *and* D_{D2};
5. *Construct four messages which each one containing one of the four subsets (*D_{U1}, D_{U2}, D_{D1} *and* D_{D2}*) as part of the header.*

End.

Fig. 3-16. Message header construction for Low Distance (LD) multicast routing.

In some cases, explained in the below, for routing a multicast message from one destination to the next destination via a minimal path, a forbidden turn is required to be taken. To prevent a possible deadlock in these cases, the message is first absorbed by the first

destination and then a copy of the message will be retransmitted to the next destination address through the consumption channels discussed in Subsection 3.2.5. In this method, for absorb and retransmission mechanism, we take advantage of a similar approach as the proposed TM_FAR method [81][82]. Fig. 3-15(d) shows an example of the paths used for the message when the proposed multicast routing algorithm is used.

3.2.4 Hamiltonian Adaptive Multicast Unicast Method (HAMUM)

Several path-based multicast routing algorithms based on the Hamiltonian path were proposed to guarantee deadlock freedom [74][78]. But the traditional path-based algorithms are deterministic for both unicast and multicast traffic which degrades the performance significantly. This was the motivation to propose a path-based method to bring adaptivity for both unicast and multicast traffic without using virtual channels. To improve the path-based method, an adaptive, deadlock-free method is presented to bring adaptivity for all of Hamiltonian based models. In fact, unlike other adaptive models in communication networks which are applicable only for unicast traffic, the proposed method handles both unicast and multicast traffic adaptively.
In traditional path-based routing models such as MP and CP, for both unicast and multicast messages according to the current and the next destination nodes positions, only a single shortest path is used by the routing algorithm. Therefore, the network performance is degraded by employing these routing algorithms. HAMUM can solve this problem in the path-based routing algorithms by routing both of the unicast and multicast messages adaptively through destination(s).
In this method, the locations where certain directions can be taken are restricted, so deadlock will be avoided. The rules regulating the proposed scheme are categorized in the up channel subnetwork and down channel subnetwork as follows:
All the nodes in even rows have lower labels than their neighboring nodes in north and east directions; while the nodes in odd rows have lower labels than their neighbors in north and west directions. Therefore, for the up channel subnetwork:

Rule1: North and East directions are allowed in even rows.
Rule2: North and West directions are allowed in odd rows.

Similarly, all the nodes in even rows have higher labels than their neighboring nodes in south and west directions; while the nodes in odd rows have higher labels than their neighbors in south and east directions. So, for the down channel subnetwork:

Rule1: South and West directions are allowed in even rows.
Rule2: South and East directions are allowed in odd rows.

Notice that a message will be forwarded to the destination as in the deterministic Hamiltonian strategy, when the current node is located one row to the south (north) of the

destination row in the up channel subnetwork (down channel subnetwork). Fig. 3-17 shows the pseudo code of the HAMUM model which is executed in each router when a new packet arrives. Inasmuch as the rules keep the messages traveling in strictly ascending order (up channel subnetwork) and descending order (down channel subnetwork), it prevents the occurrence of deadlock.

```
Algorithm HAMUM  is
-- (Cx,Cy) : Current  node , (Dx,Dy) : Destination node
Begin
    If (Dy = Cy) then                                  --Current & Dest. are in the same row
        If (Dx = Cx) then                              --Current& Dest. are in the same column
            direction <= Local;                        --Packet sends to the Local direction
        Elsif (Dx > Cx) then                           --Dest. is to the East of the Current
            direction <= East;                         --Dest. is to the West of the Current
        Else   direction <= West;
        End if;
    Elsif (Dy > Cy) then                               -- up channel Subnetwork
        If ( Cy mod 2 = 0 ) then                       --rule1 in the even rows
            If ( Dx > Cx ) and ( Dy - Cy > 1 ) then    --Dest. is in the East & more than 1 row to Current
                direction <= North or East             --North or East direction can be chosen
            Elsif ( Dx > Cx ) and ( Dy - Cy = 1 ) then --Dest. is in the East & 1 row to the Current
                direction <= East;                     --Packet sends to the East direction
            Else   direction <= North;                 --IF Dest. is in the West of the Current, select North
            End if;
        Elsif ( Cy mod 2 /= 0 ) then                   --rule2 in odd rows
            If ( Dx < Cx ) and ( Dy - Cy > 1 ) then    --Dest. is in the West & more than 1 row to Current
                direction <= North or West             --North or West direction can be chosen
            Elsif ( Dx < Cx ) and ( Dy - Cy = 1 ) then --Dest. is in the West & 1 row to the Current
                direction <= West;                     --Packet sends to the West direction
            Else   direction <= North;                 --IF Dest. is in the West of the Current, select North
            End if;
        End if;
    Elsif ( Dy < Cy ) then                             -- down channel Subnetwork
        If ( Cy mod 2 = 0 ) then                       --rule1 in even rows
            If ( Dx < Cx ) and ( Cy - Dy > 1 ) then    --Dest. is in the West & more than 1 row to Current
                direction <= South or West             --South or West direction can be chosen
            Elsif ( Dx < Cx ) and ( Cy - Dy = 1 ) then --Dest. is in the West & 1 row to the Current
                direction <= West;                     --Packet sends to the West direction
            Else   direction <= South;                 --IF Dest. is in the West of the Current, select South
            End If;
        Elsif ( Cy mod 2 /= 0 ) then                   --rule2 in odd rows
            If ( Dx > Cx ) and ( Cy - Dy > 1 ) then    --Dest. is in the East & more than 1 row to Current
                direction <= South or East             --South or East direction can be chosen
            Elsif ( Dx > Cx ) and ( Cy - Dy = 1 )then  --Dest. is in the East & 1 row to the Current
                direction <= East;                     --Packet sends to the East direction
            Else  direction <= South;                  --IF Dest. is in the West of the Current, select South
            End if;
        End if;
    End if;
End HAMUM;
```

Fig. 3-17. The pseudo code of HAMUM.

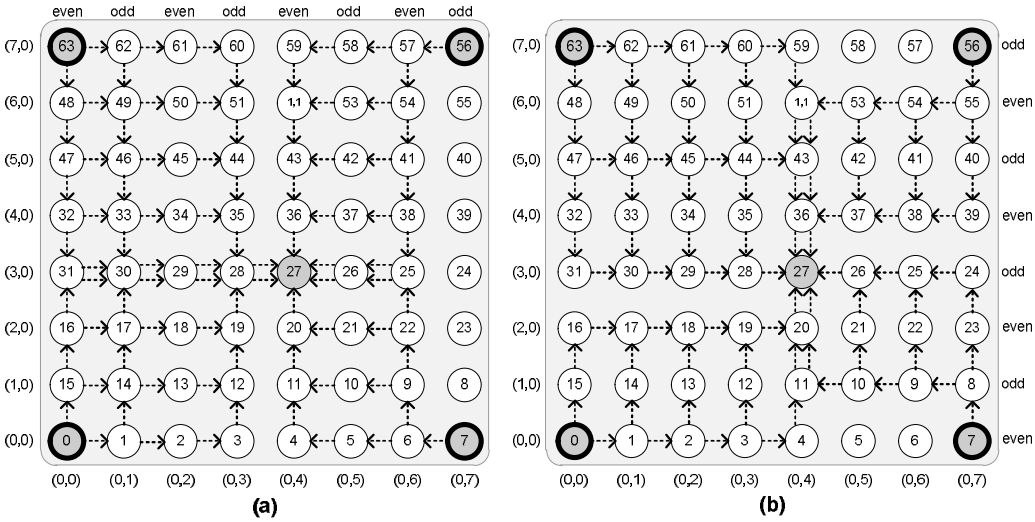


Fig. 3-18. All of the possible minimal paths from the source nodes 63, 56, 7, and 0 to the destination node 27 in (a) the Odd-Even model, and (b) the unicast aspect of HAMUM.

Table 3-5. Eight different location states of the source and destination nodes.

State	Source position (odd/even row)	Destination position (odd/even row)	Destination direction (left/right)
1	even	even	right (east)
2	even	odd	right (east)
3	even	even	left (west)
4	even	odd	left (west)
5	odd	even	right (east)
6	odd	odd	right (east)
7	odd	even	left (west)
8	odd	odd	left (west)

Unicast Aspect of HAMUM:

Based on the proposed method, any intermediate node must first determine set of directions toward which a packet may be forwarded for the next hop based on Rule 1 and Rule 2. As mentioned previously, according to the source and destination labels, the routing may take place in up or down channel subnetwork. Consider a case where the destination of a message is to the west of its source in the up channel subnetwork (e.g. source node 7 and destination 27 in Fig. 3-18(b)). If the current node is in an odd row, the router can route the message to the west or north direction because of the Hamiltonian up channel subnetwork

network strategy. If the current node is in an even row, at first the message should be routed to the north direction (to reach the odd row), and then, it could be routed via the west or north direction. Note that in the up channel subnetwork, using the Hamiltonian path, the packet can choose west or north direction in odd rows and east or north direction in even rows. Additionally, if the current node is located one row to the south of the destination row in the up channel subnetwork, the message will be routed to the west or north direction if the current node is in the odd row, and if the current node is in the even row the packet will be routed to the north direction. In Fig. 3-18(b), all the possible minimal routing paths of HAMUM for four messages in 8x8 2D-mesh have been shown. At least one minimal path always can be selected by the proposed method for any source and destination pair. Since the Odd-Even model is one of the most popular wormhole-based adaptive unicast routing algorithms in on-chip interconnection networks, we compare the unicast aspect of our method with the Odd-Even model. All of the possible routing paths for the Odd-Even model are indicated in Fig. 3-18(a). In order to compare the two algorithms with each other, we use the Degree of Adaptiveness (DoA) factor [2][61][64], which is the number of minimal paths can be taken by a message to travel from a source node (S_x,S_y) to a destination node (D_x,D_y). Suppose that Δx, Δy are defined as $\Delta x = D_x - S_x$ and $\Delta y = D_y - S_y$ and $d_x=|\Delta x|$, and $d_y=|\Delta y|$. The degree of adaptiveness for a fully adaptive algorithm is given by:

$$DoA(\textit{fully adaptive routing})_{s,d} = \frac{(d_x + d_y)!}{d_x! d_y!}$$

Based on the Hamiltonian Path, there can be eight different location states according to the source node position (even or odd row), destination node position (even or odd row), and the direction of the destination node (left or right side of the source node). The states have been summarized in Table 3-5.

First, we compute the DoA for unicast messages in the up channel subnetwork, then we use the similar way to compute the DoA for the down channel subnetwork. As can be seen in Fig. 3-19, the DoA of the state 1 and 8 is equal and can be computed as:

$$DoA(1)_{s,d} = \frac{(d_x + D)!}{d_x! D!} \quad \text{Where } D = \left\lfloor \left| \frac{d_y}{2} \right| \right\rfloor$$

For the other states, the DoA function is calculated as:

$$DoA(2)_{s,d} = \frac{(d_x + D')!}{d_x! D'!} \quad \text{Where } D' = \left\lfloor \left| \frac{d_y - 1}{2} \right| \right\rfloor$$

These equations can be summarized as:

DoA of the up channel subnetwork:

$$DoA(up-channel)_{s,d} = \begin{cases} DoA(1)_{s,d} & \textit{for states } 1 \textit{ and } 8 \\ DoA(2)_{s,d} & \textit{otherwise} \end{cases}$$

DoA of the down channel subnetwork:

$$DoA(down-channel)_{s,d} = \begin{cases} DoA(1)_{s,d} & for\ states\ 3\ and\ 6 \\ DoA(2)_{s,d} & otherwise \end{cases}$$

The Odd-Even model [64] restricts the locations where some types of turns can be taken. While HAMUM rules are based on the mesh rows, the rules of the Odd-Even model are based on the columns. Odd-Even rules are described as follow:

Rule 1: East-North and East-South turns cannot be taken in even columns (Fig. 3-20(a)).
Rule 2: North-West and South-West turns cannot be taken in odd columns (Fig. 3-20(b)).

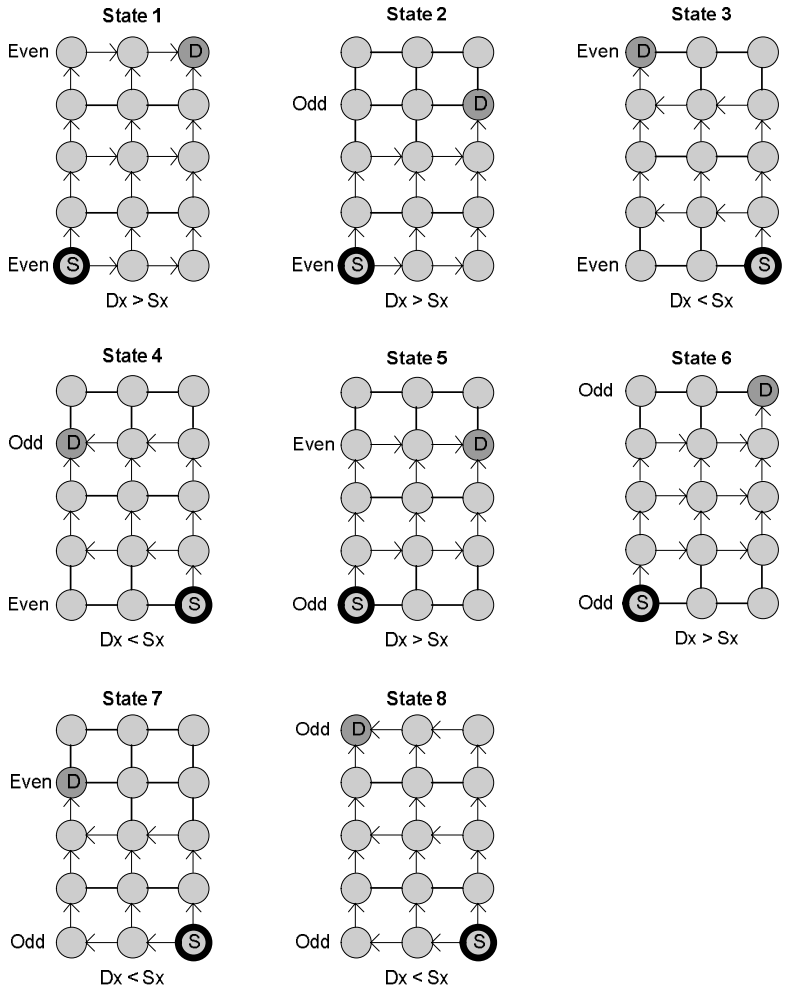


Fig. 3-19. Eight different location states in the up channel subnetwork.

The degree of adaptiveness for the Odd-Even turn model is computed as [64]:
When the destination node is in the right side of the source node (Δx>0):

$$DoA(\Delta x > 0)_{s,d} = \begin{cases} DoA(2)_{s,d} & if\ source\ node's\ column\ is\ an\ allowable\ column, \\ & destination\ is\ in\ odd\ column \\ DoA(1)_{s,d} & otherwise \end{cases}$$

When the destination node is to the left side of the source node (Δx<0):

$$DoA(\Delta x < 0)_{s,d} = \begin{cases} DoA(1)_{s,d} & if\ source\ node's\ column\ is\ an\ allowable\ column, and\ \Delta x = 0 \\ DoA(2)_{s,d} & otherwise \end{cases}$$

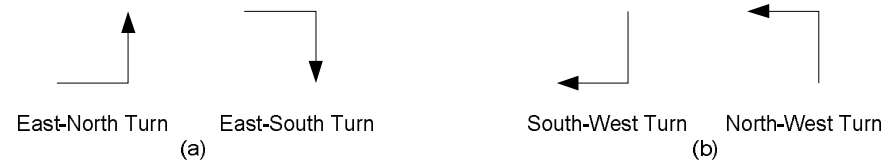


Fig. 3-20. The Odd-Even turn model rules: (a) prohibited turns in even columns (b) prohibited turns in odd columns.

Considering the above analysis, the degree of adaptiveness of HAMUM and the Odd-Even models is about the same. Since the Odd-Even model cannot be utilized for the multicast traffic, described in the motivation, HAMUM not only is compatible with multicast traffic but also provides adaptivity for both unicast and multicast traffic.

Multicast Aspect of HAMUM:

In this section, we describe how the proposed adaptive method affects the path-based multicast routing algorithms. For this purpose, we apply HAMUM on the Multi-Path (MP) and Column-Path (CP) algorithms.

1) Adaptive Multi-Path (AMP) Routing Algorithm

Fig. 3-21(a) shows an example of MP where source node 27 (3, 4) generates a multicast message to be sent towards destinations 31, 9, 59, 8, 50, 57, 26, 19, 62, 37, 0, 63, 1, 7, 32, 55. Accordingly, two subsets are organized. The first subset (D_U) has all the destinations with higher label and the second one (D_D) has the remaining destinations. Afterward, D_U is divided into two subsets, D_{U1}= {31, 32, 50, 62, 63} and D_{U2}= {37, 55, 57, 59}. In the same way D_D is divided into two subsets, D_{D1}= {19, 1, 0} and D_{D2}= {26, 9, 8, 7}. Finally, one packet per subset should be created and sent from the source node to the network. All packets must follow the Hamiltonian path and reach to destinations in the arranged order deterministically.

AMP, Adaptive MP, is the adaptive model of the MP algorithm after the proposed adaptive model is applied in the MP algorithm. Consider the example used for MP in Fig. 3-21(b), the multicast message can be forwarded in three different ways from the node 37 to the node 55 (32 to 50, 19 to 1, and 26 to 8).

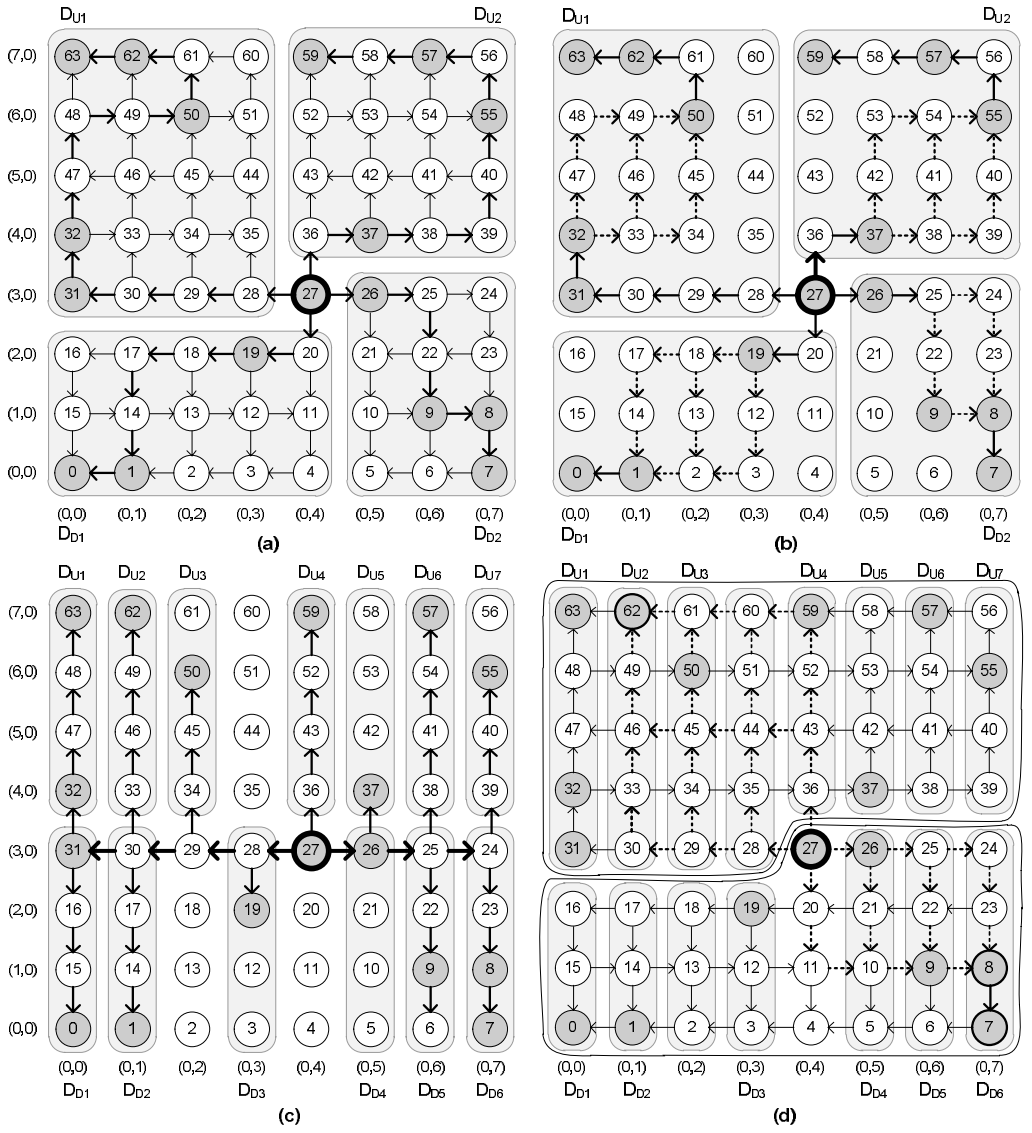


Fig. 3-21. (a) Multi-Path (MP), (b) Adaptive Multi-Path (AMP), (c) Column-Path (CP), and Adaptive Column-Path (ACP) routing algorithms.

2) Adaptive Column-Path (ACP) Routing Algorithm

We use the same example in Fig. 3-21(a) for CP. Thirteen copies of the message are used to achieve the desired multicast operation (Fig. 3-21(c)). Though destinations 1 and 62 are in

the same column, two message copies are sent to this column, since one of the destinations are above the source node's row and the other below.
The ACP, stood for the Adaptive CP is the adaptive method of the original CP by taking advantage of the proposed adaptive model. To indicate how the adaptive scheme affects the CP algorithm, as illustrated in Fig. 3-21(d), again thirteen copies of the multicast message must be used to achieve the desired multicast operation. But in this figure for simplicity, we only consider two subsets D_{U2} and D_{D6}. Due to utilizing the proposed adaptive scheme in the CP, each multicast messages can be delivered to its subset through different paths indicated by dashed lines.

3.2.5 Hardware Implementation

A. Topology and Switching Method

As mentioned before, we make use of an $n \times n$ network of interconnected tiles with a mesh topology. Each tile is composed of a PE (Processing Element) and a router connected to its four adjacent routers in addition to the PE of the tile through a set of channels [83]. Two unidirectional point-to-point links form the channel. To minimize the delay and the required resources, the wormhole method is chosen for the switching mechanism [59].

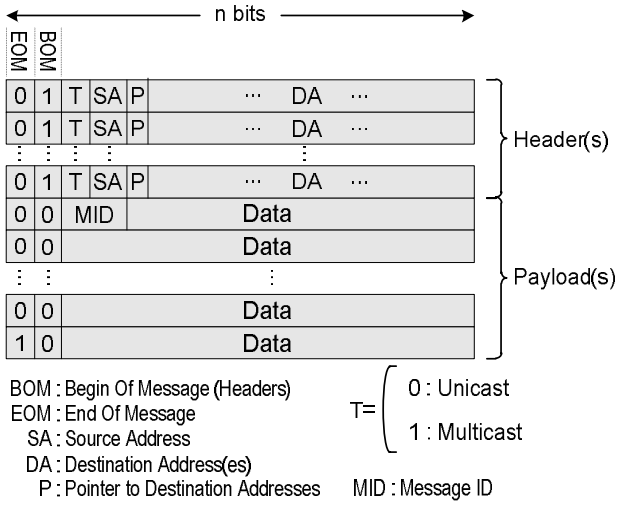


Fig. 3-22. Multicast message format for the proposed technique.

B. Message Format

The multicast message format is shown in the Fig. 3-22. It includes one or several header flits and a parametric number of payload flits. The number of flits depends on the number of destinations and the flit width in the network. Each flit is n bit wide where the n^{th} bit is

the *EOM* (End Of Message) sign and the $(n - 1)^{th}$ bit is the *BOM* (Begin Of Message) sign. In the header, the third field, which is represented by *T*, is used to describe the type of the message. There are two types of message which are unicast ($T = 0$) and multicast ($T = 1$). The address of the source address (*SA*), the pointer counter (*P*), and the destination node address(es) (*DA*) are placed in the last fields of the header, respectively, and the content of the message is located in the rest of the flits (payload). We have used all-destination encoding scheme [60] in which all destination addresses are carried by the header flits.
The pointer in each header flit points to the address of the next destination in the current header flit, and the message identifier (*MID*) is used for the message ordering. The header flits are removed as the multicast messages advances, so that if a multicast message is arrived to all destination nodes included in a header flit, the flit is removed from the message.

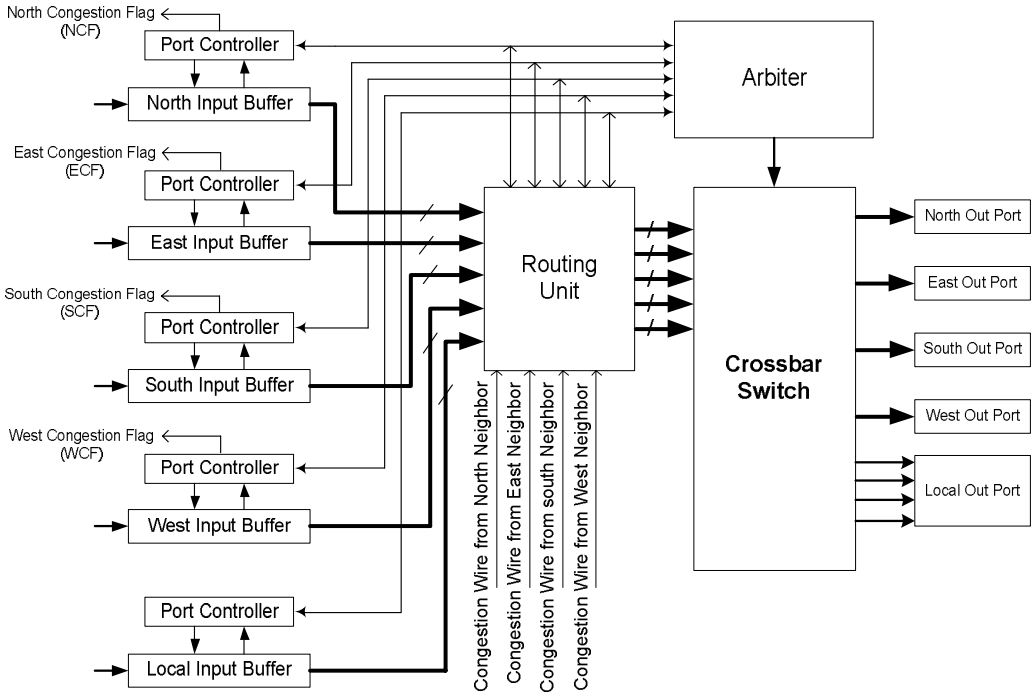


Fig. 3-23. The proposed router structure.

C. Router Architecture

The implementation details of the on-chip router are described more precisely in Section 4.3.2. As shown in Fig. 3-23, each input port has a controller for handshaking and an input buffer for the temporary storage of flits. The wormhole switching method

implemented in the controller unit, is based on on/off flow control mechanism [61]. After receiving the message header, first the routing unit determines which output should be used for routing the message and then the arbiter gives grant to the requests coming from input ports which leads to inject the message from the input port to a proper output port using the crossbar switch. The router has a crossbar which establishes the connection from an input port to an output port. When a new message reaches the input port, it waits until the previously arrived messages leave the port. Then, the new message header is delivered to the routing unit where it is routed to the appropriate output port. The Congestion Flag (CF) [38] of the buffer becomes active when the number of empty cells of the buffer is less than a threshold value. In this case, for warning about the full status, the signal CF is activated indicating that most buffer cells are occupied. Each input port has a CF through which it informs its adjacent routers about its congestion condition. Therefore, the router which uses that input port for forwarding a message to the next router should consider this router as a congested one (hotspot) and should not send messages to this router until the congestion is over.

In the path-based multicast mechanism, when multiple delivery channels (consumption channels) are occupied by one message along the multicast path, cyclic dependencies on the delivery channels may occur [61][78][84][85]. In fact, a message cannot be delivered to two different output channels simultaneously, unless the message should be sent to a consumption channel and an output channel. As illustrated in Fig. 3-24, the multicast message *A* destined to nodes 2 and 3 is generated by node 1. Simultaneously, node 4 generates the message *B* destined to the same set of destinations. As a result, due to the delivery channel contention, this cyclic wait creates a deadlock. To prevent deadlocks in delivery channels, the upper bound of the number of delivery channels required to avoid such deadlocks is equal to $2nv$ where n is the network dimension and v is the number of virtual channels per input port [78][85]. As a result, at least two delivery channels are necessary and sufficient for DP, MP, CP, AMP, and ACP algorithms and four delivery channels are enough to support deadlock-free multicasting mechanism under the LD model in 2D meshes when the base routing is either XY, Odd-Even, or the other turn model routing algorithms [81][85].

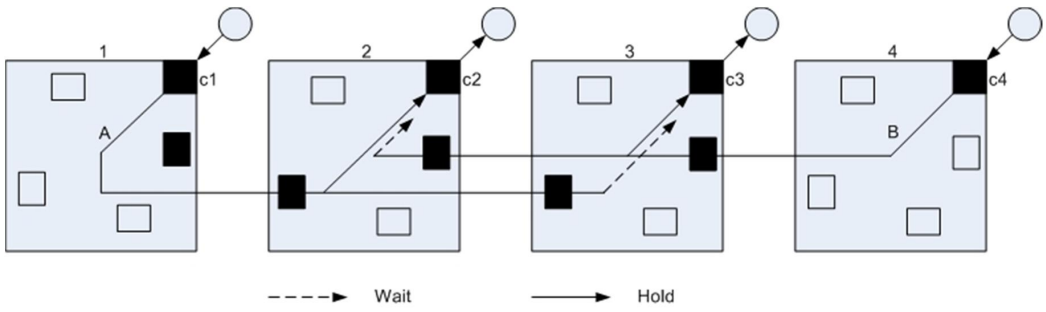


Fig. 3-24. Deadlock due to the delivery channel contention [81].

The router employs a routing unit which decodes the header of the message coming from an input port. If the header belongs to a unicast message ($T = 0$), the minimal path adaptive routing algorithms based on the Odd-Even turn model is used to determine the output port to which the message should be sent. In the Odd-Even adaptive routing algorithm there could be more than one minimal output direction to route the message. In this case, the address decoder will choose the direction in which the corresponding downstream router has not raised its congestion flag. For instance, if a message with a given source and destination could be routed to both output ports of p1 ($CF = 0$) and p2 ($CF = 1$), then it will be routed to p1. If p1 and p2 happen to have both their congestion flags raised, the message will be routed to p1. On the other hand, if the header type is a multicast message ($T = 1$), the routing unit fetches the destination address specified by the pointer in the header. If the destination address is the current node, the routing unit will request the local output port. Otherwise, the routing unit fetches the next destination address from the header and runs the Odd-Even procedure to determine the output port(s) corresponding to the next destination address. Also, after fetching, the routing unit increases the pointer value of the header, and if it is overflowed, it means that the multicast message has been sent to all the destination addresses in this header flit, the routing unit will remove the corresponding header flit from the message.

It should be noted that as a result of exploiting the adaptive Odd-Even routing algorithm, the messages of the same data may traverse different paths reaching at the destination out-of-order. Hence, a technique may be needed to reorder the messages at the destination. In the proposed technique in this chapter, the messages that reach the destination node have the information about the message source node (*SA*) and the message order (*MID*). Using the *SA* and *MID*, the destination core may store each message in its proper location in the core memory such that the original source order can be achieved with negligible overhead. Note that the data in the memory might not be processed by the core unless all parts of the data are received. This is also true for deterministic multicast routing algorithms. Also, the use of the source address enables the destination to concurrently handle data coming from different sources.

3.2.6 Experimental Results

To assess the efficiency of the proposed Low Distance (LD) path-based multicast routing algorithm, Dual-Path (DP), Multi-Path (MP), Column-Path (CP), and UB (Unicast-Based) algorithms are implemented. MP and CP are also used to evaluate the HAMUM routing model. We have developed a cycle accurate wormhole NoC simulator. The simulator calculates the average delay and the power consumption for the message transmission. The simulator inputs include the array size, the operating frequency, the routing algorithm, the link width, and the traffic type. The simulator can generate different traffic profiles. To calculate the power consumption, we have used Orion library functions [86]. For all routers, the data width and the frequency were set to 32 bits and 1GHz, respectively, which led to a bandwidth of 32 Gb/s. Each input channel has a buffer (FIFO) size of 8 flits with the congestion threshold set at 75% of the total buffer capacity. The message size was

assumed to be 16 flits. In addition, we also assumed that the 2D mesh topology was regular and the delays on wires would not exceed the clock period. For the performance metric, we use the multicast latency defined as the number of cycles between the initiation of the multicast message operation and the time when the tail of the multicast message reaches all the destinations. The CP has the most complicated procedure to prepare the multicast messages, while the DP has the easiest procedure [78]. The preparation mechanism consists of partitioning the destination set into appropriate subsets and creating multiple copies of the message. For computing the preparation time, several sets of multicast destinations have been run by the simulator. Under these test sets, the average preparation time to complete multicast messages in the DP, MP, LD, and CP algorithms were 35, 46, 46, and 82 cycles, respectively. Because the DP algorithm generates only 2 multicast messages, it is the best among the other algorithms and the CP is the worst in terms of the startup latency.

A. Multicast Traffic Profile

The first sets of simulations were performed for a random traffic profile. In these simulations, the PEs generate data messages and inject them into the network using the time intervals which are obtained using the exponential distribution. Two mesh sizes of 8×8 and 16×16 have been considered. In the multicast traffic profile, each PE sends a message to a set of destinations. A uniform distribution was used to construct the destination set of each multicast message [74]. The number of destinations was set to 10 and 25.

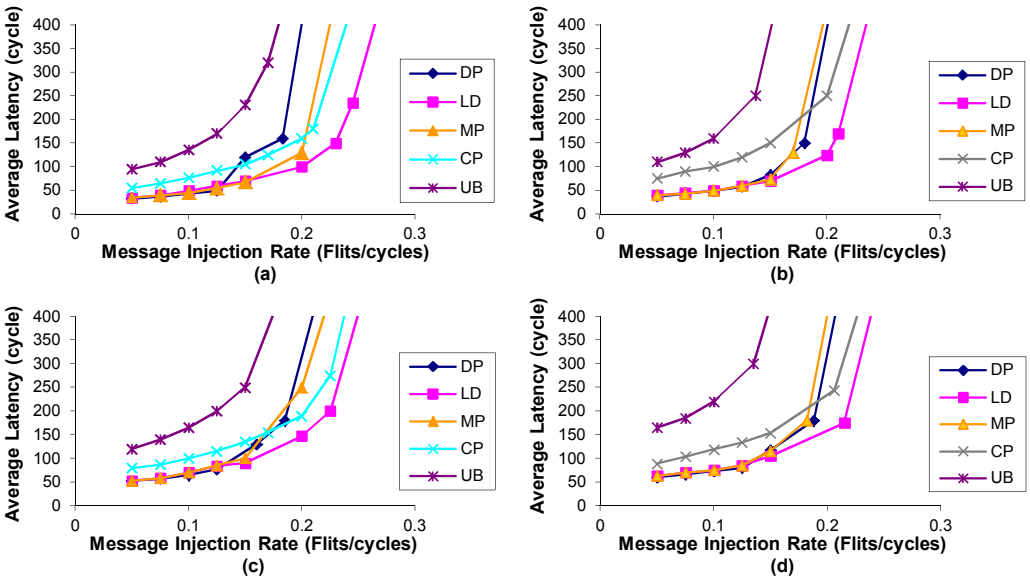


Fig. 3-25. Performance evaluation of LD under different loads in 8×8 2D-mesh with (a) 10 destinations, (b) 25 destinations and in 16×16 2D-mesh with (c) 10 destinations, (d) 25 destinations under multicast traffic model.

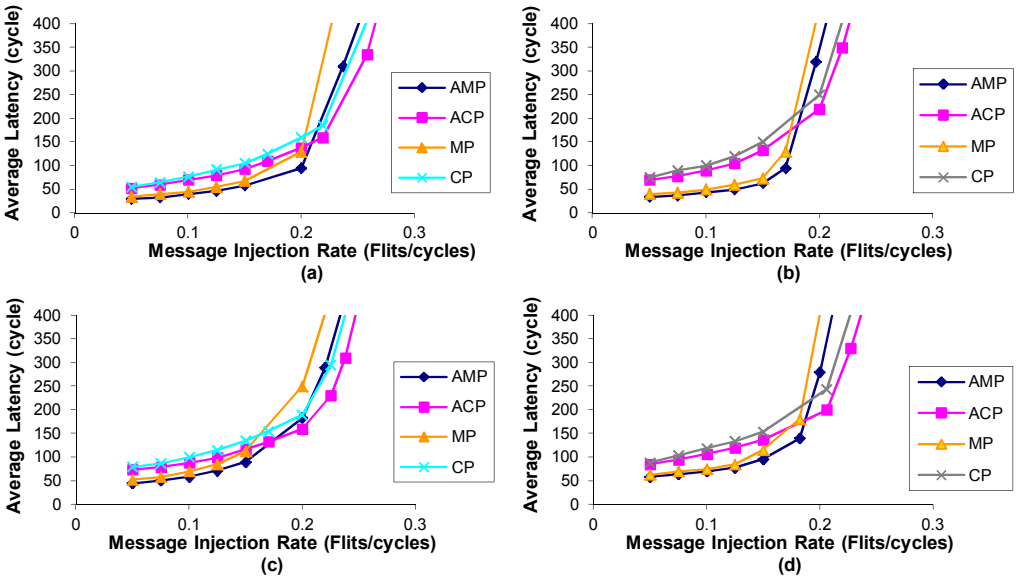


Fig. 3-26. Performance evaluation of HAMUM under different loads in 8×8 2D-mesh with (a) 10 destinations, (b) 25 destinations and in 16×16 2D-mesh with (c) 10 destinations, (d) 25 destinations under multicast traffic model.

In Fig. 3-25 the average communication delay as a function of the average flit injection rate is shown. As the results show, the proposed LD multicast routing algorithm leads to the lowest latency among all the three multicast routing algorithms even at high traffic loads or with a large number of destinations (25 destinations). Fig. 3-26 also shows the performance gain of using HAMUM with different number of destinations and mesh sizes. As observed from the results, the proposed adaptive mechanism which has been applied to the MP and CP schemes even in high traffic loads or with a large number of destinations leads to lower delay.

B. Unicast and Multicast (Mixed) Traffic Profiles

In these simulations, we employed a mixture of unicast and multicast traffic where 80% of the injected messages are unicast messages and the remaining 20% are multicast messages. This pattern may represent the traffic in a distributed shared-memory multiprocessor where updates and invalidation produce multicast messages and cache misses are served by unicast messages [76][78]. For this set, the simulation parameters were similar to the previous simulations in terms of the number of destinations and array sizes. The unicast messages are also routed using the Odd-Even turn model. Uniform and hotspot [64] were the two different traffic profiles considered for the unicast traffic generation. In the uniform traffic profile, each PE sends a message to any other PE with an equal probability. Therefore, the destinations are determined randomly using a uniform distribution. Under the hotspot traffic pattern, one or more nodes are chosen as hotspots receiving an extra

portion of the traffic in addition to the regular uniform traffic. In Fig. 3-27, the average communication latencies versus the message injection rate for different algorithms under the uniform traffic model for unicast traffic profile are shown. As these figures reveal, for this traffic profile, LD outperforms the other algorithms. Regarding HAMUM, as depicted in Fig. 3-28, the adaptive routing algorithms perform better under the presented traffic.

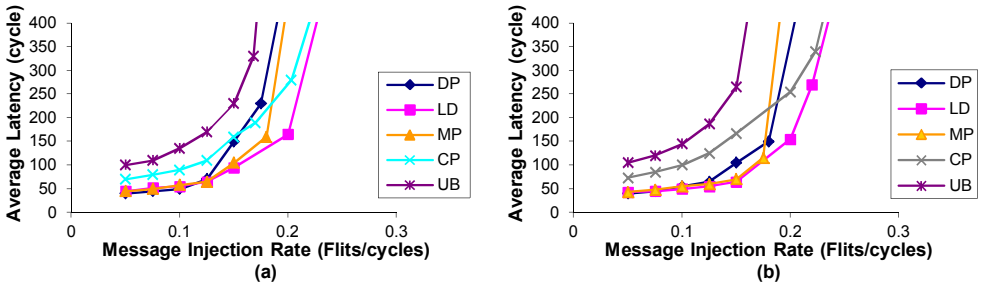


Fig. 3-27. Performance evaluation of LD under different loads in 8×8 2D-mesh with (a) 10 destinations, (b) 25 destinations under mixed traffic (20% multicast and 80% unicast) while unicast traffic is based on the uniform traffic model.

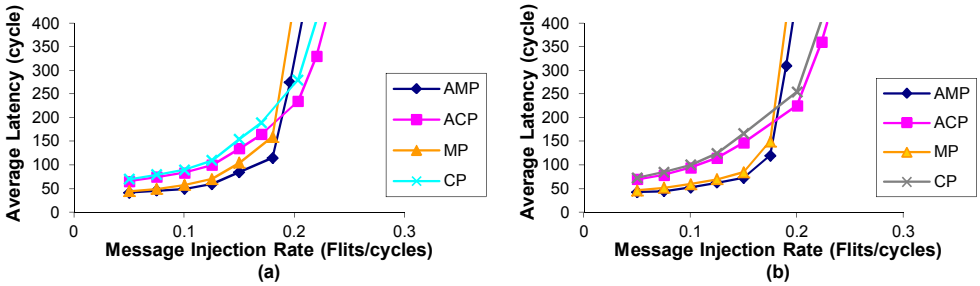


Fig. 3-28. Performance evaluation of HAMUM under different loads in 8×8 2D-mesh with (a) 10 destinations, (b) 25 destinations under mixed traffic (20% multicast and 80% unicast) while unicast traffic is based on the uniform traffic model.

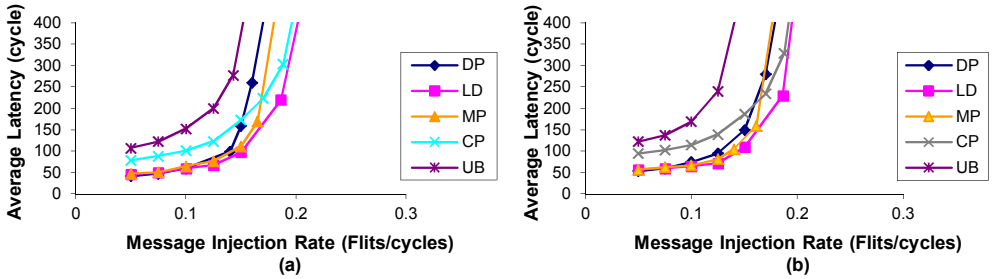


Fig. 3-29. Performance evaluation of LD under different loads in 8×8 2D-mesh with (a) 10 destinations, (b) 25 destinations under mixed traffic (20% multicast and 80% unicast). Unicast traffic is based on the hotspot traffic model with a single hotspot node (4, 4). The hotspot percentage is 10 percent.

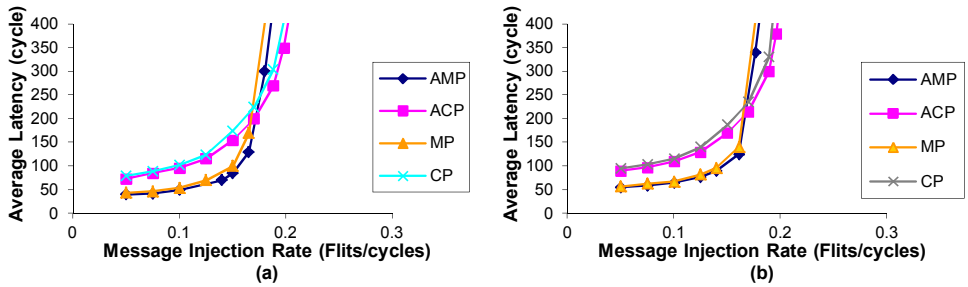


Fig. 3-30. Performance evaluation of HAMUM under different loads in 8×8 2D-mesh with (a) 10 destinations, (b) 25 destinations under mixed traffic (20% multicast and 80% unicast). Unicast traffic is based on the hotspot traffic model with a single hotspot node (4, 4). The hotspot percentage is 10 percent.

Under the hotspot traffic model with the hotspot percentage of h, a newly generated message is directed to each hotspot node with an additional h percent probability. In our simulations, we assumed a single hotspot node (node (4, 4)). Fig. 3-29 shows the average latencies of the algorithms for the 8×8 2D mesh topologies when h = 10%. As the figures show, LD considerably outperforms the other algorithms for different number of destinations under various message injection rates. Inasmuch as HAMUM brings adaptivity to the conventional multicast routing algorithms, Fig. 3-30 reveals the performance gains of the adaptive schemes under the hotspot traffic model.

C. Application Traffic Profile

To show the impact of the proposed model, traces from some application benchmark suites selected from SPLASH-2 [87] and PARSEC [88][89] are used. Traces are generated from SPLASH and PARSEC using the GEMS simulator [90]. We used the x264 application of PARSEC and the Radix, Ocean, and fft applications from SPALSH-2 for our simulation. Table 3-2 summarizes our full system configuration. It is noteworthy that the token-based MOESI protocol [92] is heavily based on multicast. On account of our analysis on average 95% of token-based MOESI traffic is multicast.

As can be seen from Fig. 3-31, the proposed adaptive model diminishes the average delay of MP and CP significantly under all benchmarks. That is, adaptive routing has an opportunity to improve performance. Under the fft application the adaptive model indicates 17% and 21% reduction in latency for MP and CP, respectively.

D. Power Dissipation

The power dissipation of DP, MP, CP, UB, and the proposed LD routing algorithms were calculated and compared under the multicast traffic model. The results for the average and maximum power under this traffic are shown in Fig. 3-32 and Fig. 3-33, respectively.

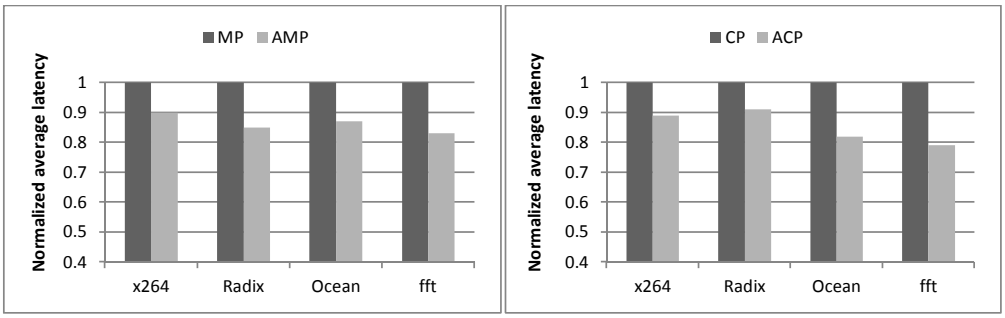


Fig. 3-31. Performance under different application benchmarks for multi-path (left) and column-path (right) routing algorithms.

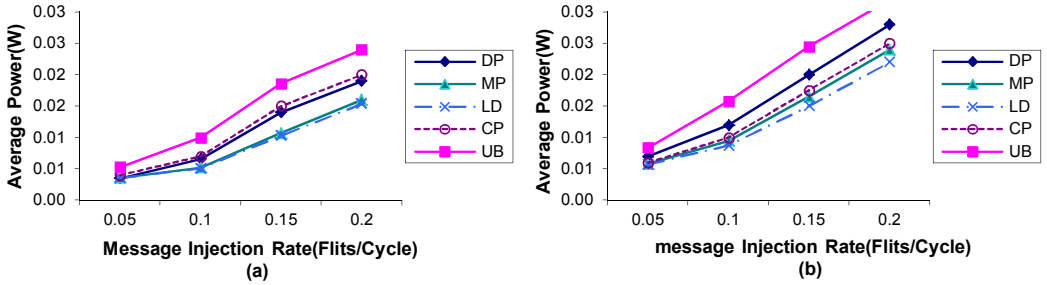


Fig. 3-32. Average power dissipation of the proposed, the DP, the MP and the CP algorithms in 16×16 2D-mesh with (a) 10 destinations and (b) 25 destinations under multicast traffic.

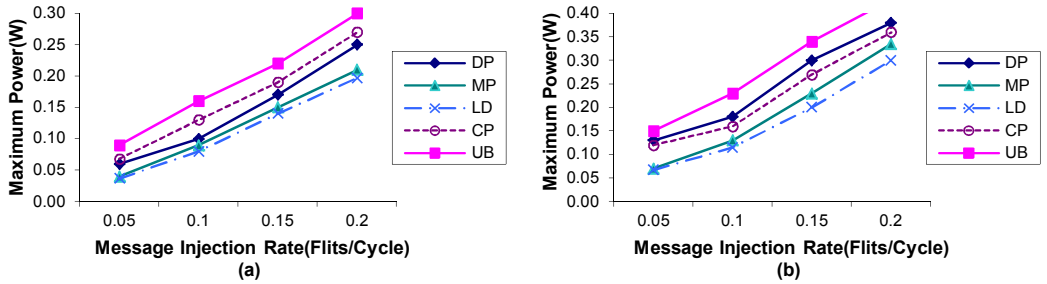


Fig. 3-33. Maximum power dissipation of the proposed, the DP, the MP and the CP algorithms in 16×16 2D-mesh with (a) 10 destinations and (b) 25 destinations under multicast traffic.

As the results presented in Table 3-6 for 10 destinations reveal, the average power dissipation of the network with the LD algorithm is 25%, 3.5%, 33%, and 63% less than those of the DP, MP, CP, and UB algorithms under the multicast traffic model, respectively. Also, the results of Table 3-7 for 10 destinations indicate that the maximum power of the LD algorithm is 27%, 8%, 44%, and 70% less than those of the DP, MP, CP, and UB algorithms, respectively, under the multicast traffic model. Similar power savings

are obtained for 25 destinations. The power reduction for the LD algorithm is achieved by smoothly distributing the power consumption over the network using the adaptive routing scheme which reduces the number of the hotspots, and hence, lowering both the average power and the maximum power. Table 3-8 reveals the average power dissipation of the network with the ACP algorithm is 5% less than that of the CP algorithm and the average power dissipation of the AMP is 3.5% less than that of the MP algorithm. The results of Table 3-8 indicate that the maximum power of the ACP and AMP algorithms is 15% and 11% less than that of the CP and MP algorithms, respectively under the multicast traffic model. We can notice that the average power and the maximum power of the proposed adaptive models are lower.

Table 3-6. Comparative average power dissipation of LD with other algorithms in 16×16 2D-mesh.

Average Power Dissipation	DP	MP	CP	UB
With 10 Destinations	-25%	-3.5%	-33%	-63%
With 25 Destinations	-32%	-8.5%	-13%	-51%

Table 3-7. Comparative maximum power dissipation of LD with other algorithms in 16×16 2D-mesh.

Maximum Power Dissipation	DP	MP	CP	UB
With 10 Destinations	-27%	-8%	-44%	-70%
With 25 Destinations	-43%	-12%	-33%	-64%

Table 3-8. Comparative average power dissipation of the adaptive schemes using HAMUM model with the conventional schemes.

Average Power Dissipation	AMP/MP	ACP/CP
With 25 Destinations	3.5%	5%

Table 3-9. Comparative maximum power dissipation of the adaptive schemes using HAMUM model with the conventional schemes.

Maximum Power Dissipation	AMP/MP	ACP/CP
With 25 Destinations	11%	15%

E. Hardware Overhead

To evaluate the area overhead of the LD algorithm, we designed the routers based on the multicast routing schemes including the additional hardware required for each scheme as described in the previous section, i.e. consumption channels. The routers were described in VHDL for a 16×16 2D mesh NoC environment, and synthesized with the Leonardo-

Spectrum ASIC using a 0.25µm standard cell library. In addition, the destination sorting algorithms were included in the hardware overhead. For all the routers, the data width was set to 32 bits (flit size) and each input channel had a buffer size of 8 flits. As discussed in earlier, for the DP, MP, and CP routers, two delivery channels, and for the LD router four delivery channels are employed, respectively. In order to achieve better performance/power efficiency, the FIFOs were implemented using registers. Fig. 3-34 shows the area of the routers. While the same router structure was used for the CP, MP, and DP multicasting schemes, different number of registers were employed in implementing their sorting mechanisms leading to different areas. Comparing the area of the LD router with the UB, DP, MP, and CP routers indicates an additional overhead of 11%, 6.4%, 5%, and 6%, respectively. In addition, the hardware overhead of implementing HAMUM in both of the MP and CP routers is less than 0.5% and that can be considered negligible.

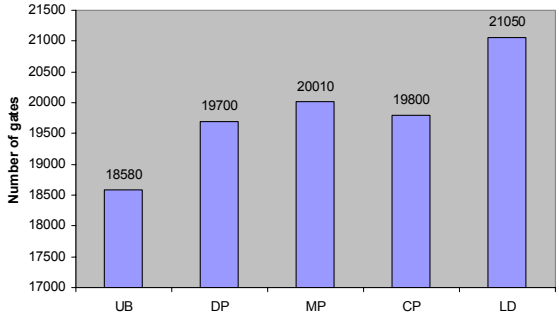


Fig. 3-34. Area cost of routers for implementing different multicast routing algorithms.

3.3 Summary

Routing protocols can have a large impact on performance and power consumption in on-chip networks. Therefore, three adaptive routing algorithms were presented in this chapter. The routing protocols in NoCs can be either unicast (one-to-one) or multicast (one-to-many). The first presented routing protocol was a congestion-aware adaptive routing algorithm for two-dimensional mesh NoCs which does not support multicast traffic while the other two presented protocols are adaptive routing models supporting both unicast and multicast traffic.

The first routing algorithm was an adaptive routing algorithm, called EDXY, which improves the DyXY routing algorithm. In this technique, two congestion wires were added to the router architecture to flag the row or column congestion further away from the current switch. This enabled avoiding the congested path, and thus decreasing the latency of the algorithm. Moreover, two adaptive routing protocols (LD and HAMUM) were presented for mesh-based on-chip networks. LD used network partitioning, optimized destination ordering, and the Odd-Even turn model adaptive algorithm for routing both the multicast and unicast messages through the network. Additionally, the adaptive routing

algorithm used the congestion condition of the input ports to route the messages through non-congested paths. However, exploiting the unicast routing algorithms for multicast traffic may increase the latency. HAMUM was presented as an adaptive routing method for both unicast and multicast traffic which maximizes the degree of adaptiveness of the routing functions which are based on the Hamiltonian path.

Chapter 4

Adaptive On-Chip Router Architecture

The performance and efficiency of NoCs largely depend on the *output-selection* and *input-selection* methods exploited by on-chip routers. The output-selection method, using a routing algorithm, determines which output channel should be chosen for a packet arrived from an input channel. The input-selection method chooses one of input channels to get access to the output channel, which is performed by an arbitration process.
In this chapter, a novel router architecture, named Adaptive Input-Output Selection (AIOS), is presented. AIOS employs adaptive input-selection and output-selection methods in the routing process. The adaptive output-selection method of AIOS uses either minimal or non-minimal path for unicast and multicast messages depending on the congestion condition of the network. The adaptive input-selection method exploits the Weighted Round Robin (WRR) [93] arbitration mechanism which can prevent starvation and improve the performance. To evaluate AIOS, we compare it with the other router schemes under several synthetic traffic profiles along with Video Object Plane Decoder (VOPD), i.e. an example of a real traffic profile.

4.1 Adaptive Input-selection and Output-selection Methods

In this section, we review the previous works on adaptive input-selection (arbitration) and output-selection (routing) methods. Choosing one of input channels to get access to the output channel is performed by an input-selection method using an arbitration mechanism. The arbiter could follow either non-priority or priority scheme [65][94][95]. In the non-priority scheme when there are multiple input port requests for the same available output port, the arbiter does not consider the traffic condition of the input channels to grant access to one input port. First-Come-First-Served (FCFS) [65][96] and Round-Robin (RR) [65][94] are two approaches using non-priority arbitration. Therefore, these methods can avoid starvation on different ports. Unlike the non-priority scheme, in the priority scheme when there are multiple input port requests for the same available output port, the

arbiter would grant access to the input port request which has the highest priority level, e.g. Contention-Aware Input Selection (CAIS) [95]. In CAIS, the busiest input channel obtains the highest priority to access the output channel. The input channel is given priority proportional to the number of requests arrived from the upstream routers. Thus, the traffic can be kept flowing from busy channels to avoid the network congestion. This scheme increases the possibility of the starvation so that in this chapter, we have presented an efficient arbitration mechanism using WRR. The presented input-selection method is able to avoid starvation while serving each input channel according to its traffic condition.

As described in the previous chapter, the routing algorithms, employed by the output-selection method, are classified as deterministic (e.g. XY) or adaptive (Odd-Even, DyAD, DyXY, EDXY, etc.). Some adaptive routings, like Hot-potato, can also select non-minimal paths during the routing process. Hot-potato (or deflection routing) [96][99] is based on the idea of delivering a packet to an output channel at each cycle. If all the minimal path channels are occupied, then the packet is misrouted. When contention occurs and the desired channel is not available, the packet, instead of waiting, will pick any alternative available channels (minimal or non-minimal) to continue moving to the next router; therefore the router does not need buffers. In the hot-potato routing, if the number of input channels is equal to the number of output channels at every router node, packets can always find an exit channel and thus, the routing is deadlock-free. However, livelock is a potential problem in this routing such that message latency increases significantly. Accordingly, performance of hot-potato is not as efficient as other adaptive routing algorithms [60]. All the aforementioned adaptive routing algorithms are utilized without using virtual channels. However, virtual channels can be employed to gain performance.

Regarding the multicast routing, if turn model algorithms are adopted to route multicast packets, some forbidden turns might occur [78][81]. To cope with the forbidden turns the absorb-and-retransmission mechanism is required [78][81]. However this technique degrades the performance significantly. In [37] authors utilized the Odd-Even routing algorithm to route multicast packets. The more frequently forbidden turns occur the more performance is degraded. Accordingly, the Hamiltonian Adaptive Multicast and Unicast Model (HAMUM), presented in the previous chapter, is proposed to support both unicast and multicast traffic adaptively [39]. The adaptivity of HAMUM is identical to the adaptivity of Odd-Even for the unicast traffic while for the multicast traffic the adaptivity of HAMUM is higher than conventional multicast routing algorithms [39]. Hence, HAMUM is exploited as the output-selection method for AIOS because it provides adaptivity for both unicast and multicast traffic efficiently.

4.2 Minimal and Non-Minimal Implementations of HAMUM

As presented in subsection 3.2.4, HAMUM is a minimal path routing algorithm for unicast and multicast traffic [37]. According to the rules given by HAMUM, it provides several directions to deliver a packet from a node.

```
Algorithm modified_HAMUM  is
-- (Cx,Cy) : Current  node , (Dx,Dy) : Destination node
MinPath1=NULL; MinPath2=NULL; NonminPath=NULL;
Begin
    If (Dy = Cy) then                                   --Current & Dest. are in the same row
        If (Dx = Cx) then                               --Current& Dest. are in the same column
            MinPath1 <= Local;                          --Packet sends to the Local direction
        Elsif (Dx > Cx) then
            MinPath1 <= East;                           --Dest. is to the East of the Current node
        Else
            MinPath1 <= West;                           --Dest. is to the West of the Current node
        End if;
    Elsif (Dy > Cy) then                                --up channel Subnetwork
        If ( Cy mod 2 = 0 ) then                        --rule1 in the even rows
            If ( Dx > Cx ) and ( Dy - Cy = 1 ) then     --Dest. is in the East & 1 row to the Current node
                MinPath1 <= East;                       --One minimal path is suggested
            Elsif ( Dx > Cx ) and ( Dy - Cy > 1 ) then  --Dest. is in the East of the Current node
                MinPath1 <= East;                       --Two minimal paths are suggested
                MinPath2 <= North;
            Else                                        --Dest. is in the West of the Current node
                MinPath1    <= North;                   --One minimal and One non-minimal path are suggested
                NonminPath <= East;
            End if;
        Elsif ( Cy mod 2 /= 0 ) then                    --rule2 in odd rows
            If ( Dx < Cx ) and ( Dy - Cy = 1 ) then     --Dest. is in the West & 1 row to the Current node
                MinPath1 <= West;                       --One minimal path is suggested
            Elsif ( Dx < Cx ) and ( Dy - Cy > 1 ) then  --Dest. is in the West of the Current node
                MinPath1 <= West;                       --Two minimal paths are suggested
                MinPath2 <= North;
            Else                                        --Dest. is in the East of the Current node
                MinPath1    <= North;                   --One minimal and One non-minimal path are suggested
                NonminPath <= West;
            End if;
        End if;
    Elsif ( Dy < Cy ) then                              --down channel Subnetwork
        If ( Cy mod 2 = 0 ) then                        --rule1 in even rows
            If ( Dx < Cx ) and ( Cy - Dy = 1 ) then     --Dest. is in the West & 1 row to the Current node
                MinPath1 <= West;                       --One minimal path is suggested
            Elsif ( Dx < Cx ) and ( Cy - Dy > 1 ) then  --Dest. is in the West of the Current node
                MinPath1 <= West;                       --Two minimal paths are suggested
                MinPath2 <= South;
            Else                                        --Dest. is in the East of the Current node
                MinPath1    <= South;                   --One minimal and One non-minimal path are suggested
                NonminPath <= West;
            End If;
        Elsif ( Cy mod 2 /= 0 ) then                    --rule2 in odd rows
            If ( Dx > Cx ) and ( Cy - Dy = 1 )then      --Dest. is in the East & 1 row to the Current node
                MinPath1 <= East;                       --One minimal path is suggested
            Elsif ( Dx > Cx ) and ( Cy - Dy > 1 ) then   --Dest. is in the East of the Current node
                MinPath1 <= East;                       --Two minimal paths are suggested
                MinPath2 <= South;
            Else                                        --Dest. is in the West of the Current node
                MinPath1    <= South;                   --One minimal and One non-minimal path are suggested
                NonminPath <= East;
            End if;
        End if;
    End if;
End modified_HAMUM  ;
```

Fig. 4-1. The pseudo VHDL code of modified HAMUM including the non-minimal routing.

In the up channel subnetwork, the packets arriving from the south direction at the nodes in even (odd) rows are not allowed to turn into the west (east) direction. In contrast, in the down channel subnetwork, the packets coming from the north direction at the nodes in even rows (odd) rows are not allowed to turn into the east (west) direction.
HAMUM can be extended to support the non-minimal path routing in the network. Fig. 4-1 depicts the implementation of the modified HAMUM. Once the presented algorithm is performed, three output variables, MinPath1, MinPath2, and NonMinPath, are evaluated. The variables of MinPath1 and MinPath2 are the minimal directions that can be chosen by a packet while the NonMinPath indicates the allowable non-minimal direction. For example, if the source node is located in the even row and the destination node is in the northeast position of the source node, two minimal directions (i.e. east and north) are suggested by the algorithm; while in the similar case, if the source node is located in the odd row, one minimal direction (i.e. north) and one non-minimal direction (i.e. west) are supplied by the algorithm.
Two examples of the modified HAMUM, using minimal and non-minimal directions, are shown in Fig. 4-2. In the first example, the source node 1 sends a message to destination 23 while the nodes 11 and 18 are faulty or congested. The modified HAMUM allows the packet to route around the congested areas by selecting the non-minimal path at node 8. As another example in Fig. 4-2, a packet can turn around the congested region when traveling from the source node 2 to the destination 16.

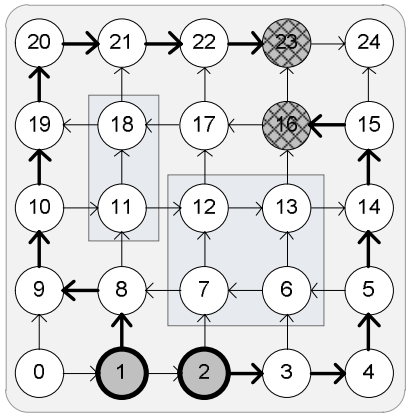


Fig. 4-2. An example of modified HAMUM.

4.2.1 Deadlock Avoidance

Deadlock is a situation where network resources continuously wait for each other to be released. To show that the proposed algorithms are deadlock-free, it is required to prove that there is no cyclic dependency between channels [101].

The modified HAMUM is deadlock-free:

At the source node, the network is divided into two disjoint subnetworks, up channel (G_U) and down channel (G_D). Since each of the up channel and down channel subnetworks uses separate sets of channels, no cyclic dependency will be created among channels. If we could prove that the message routing algorithm in the up channel subnetwork is deadlock-free, that would be sufficient to establish that the down channel subnetwork is also deadlock-free, and since $G_U \cap G_D = \Phi$, the whole network will be free of deadlocks. So, we take the up channel subnetwork into consideration.

A network can be represented by a connected graph G = (V,E), where V denotes a set of vertices (routers or node) and E a set of edges (communication links). A pair (u,v) $\in E$ form an edge of the graph, if u is physically connected to v via a communication link. A path is a sequence of non-repeated nodes such that for a given i, $0 \le i < n-1$ there exists a communication link from v_i to v_{i+1}, i.e. (v_i, v_{i+1}) $\in E$. The unicast message can be expressed by Unicast=(u,d) where $u \in V$ and $d \in V$. The multicast message can be represented by Multicast=(u,D), where $u \in V$ is the source node, $D = \{d1, d2, \ldots, dx\}$ is the set of ordered destination nodes, and x is the number of destination nodes. Each node in the graph has a label (L) determined by the Hamiltonian path labeling mechanism.

Since a unicast message is the special case of a multicast message, we prove that the algorithm is deadlock-free for the multicast messages, and then it is obvious for the unicast messages.

Given a source node u and a set of destination D, let Path(u,D) denote the multicast message path from u to all destinations of D. As already mentioned, in the up channel subnetwork, the destination nodes are ordered in ascending order, so $L(u) < L(d1) < L(d2) < \ldots < L(dx)$. If we suppose that a minimal or non-minimal multicast message path is $Path(u,D) = (\mathbf{u}, a_1, a_2, \ldots, a_x, \mathbf{d1}, a_{x+1}, a_{x+2}, \ldots, a_y, \mathbf{d2}, a_{y+1}, a_{y+2}, \ldots, a_z, \mathbf{dx})$, then intermediate nodes (either in the minimal or non-minimal paths) must be selected in a way that the packet follows the path only in ascending order, so:

$$L(v_0) \le L(u) \le L(a_1) \le L(a_2) \le \ldots \le L(a_x) < \mathbf{L(d1)} \le L(a_{x+1}) \le L(a_{x+2}) \le \ldots \le L(a_y) < \mathbf{L(d2)} \le L(a_{y+1}) \le L(a_{y+2}) \le \ldots \le L(a_z) < \mathbf{L(dx)} \le L(a_{n-1})$$

Note that the Hamiltonian path guarantees the existence of at least one possible path between each pair of nodes. According to the above facts, there cannot exist any link like (a_i,a_{i+1}), where $L(a_i) > L(a_{i+1})$, so no cyclic dependency can occur between channels for a single packet. Moreover all unicast and multicast packets in the up channel subnetwork are routed in entirely ascending order, thus the traveled paths of all packets cannot create any dependency cycles. The similar proof can be applied to the down channel subnetwork.

Fig. 4-3 shows all possible turns that can be created in the network by unicast and multicast messages. No combination of the turns can form a cycle; this can be used as another proof that the proposed algorithm is deadlock-free.

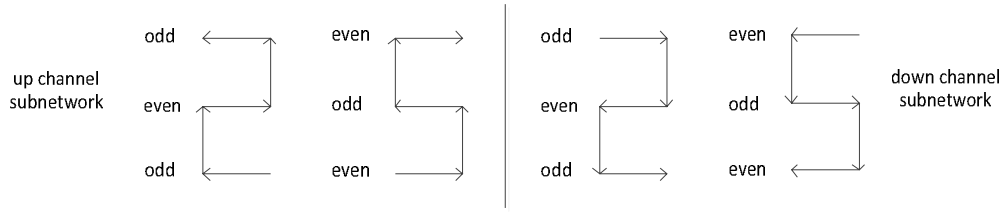


Fig. 4-3. All possible turns of HAMUM and modified HAMUM.

4.3 The AIOS Router Architecture

The underlying idea of the proposed router architecture is to spread the traffic to prevent congestion. Using adaptive input-selection and output-selection methods can improve the network performance significantly. The output-selection method utilizes both the minimal and non-minimal schemes of HAMUM. When congestion (hotspot) is formed close to a router, a non-minimal direction is selected to deliver a packet while a minimal direction is taken when there is no congestion. In AIOS, the input selection exploits the Weighted Round Robin (WRR) policy which makes the routing algorithm non vulnerable to starvation. Also, WRR increases the performance of the network by monitoring the traffic condition. We consider a $n \times n$ network of interconnected tiles with a mesh topology using wormhole scheme for the switching [68].

4.3.1 Message Format

The message format utilized in this architecture is similar to one described in Section 3.2.5.

4.3.2 Router Structure

As shown in Fig. 4-4, each input channel has a routing unit, a controller for handshaking and an input buffer. The flits of the packets are stored in the input buffer. The routing unit determines the output channel to route packets. The controller controls the buffer status including empty and full states as well as detects the sign of the rate at which the buffer is becoming occupied. A positive rate indicates that the buffer is becoming full while a negative rate reveals that the buffer is becoming empty.

Each input channel has a Congestion Flag (CF) signal (i.e. ECF, WCF, NCF, SCF and LCF corresponding to East, West, North, South and Local input channel, respectively) to inform its adjacent routers about its congestion condition so that the congested input channel should not be selected by the upstream router until the congestion condition is ceased.

The router has a crossbar to establish a connection path from an input port to an output port. For each output port the router uses an arbiter for selecting among simultaneous input requests to access the same output port. In order to detect whether the buffer status is critical or not, the flit arrival and departure rates of the buffer should be measured. For this purpose, the circuit shown in Fig. 4-5 is used. N_{new} is the number of occupied slots of the

input buffer in the current cycle of the router clock and N_{old} is the same number but in the previous cycle of the router clock. To determine the rate at which the buffer becomes full, the number of filled buffer cells at each rising edge of the router internal clock (N_{new}) is compared to that of the previous rising edge (N_{old}). If $N_{new} > N_{old}$ ($N_{old} > N_{new}$), it shows that the buffer is becoming full (empty). The sign is compared to the buffer status to activate the CF.

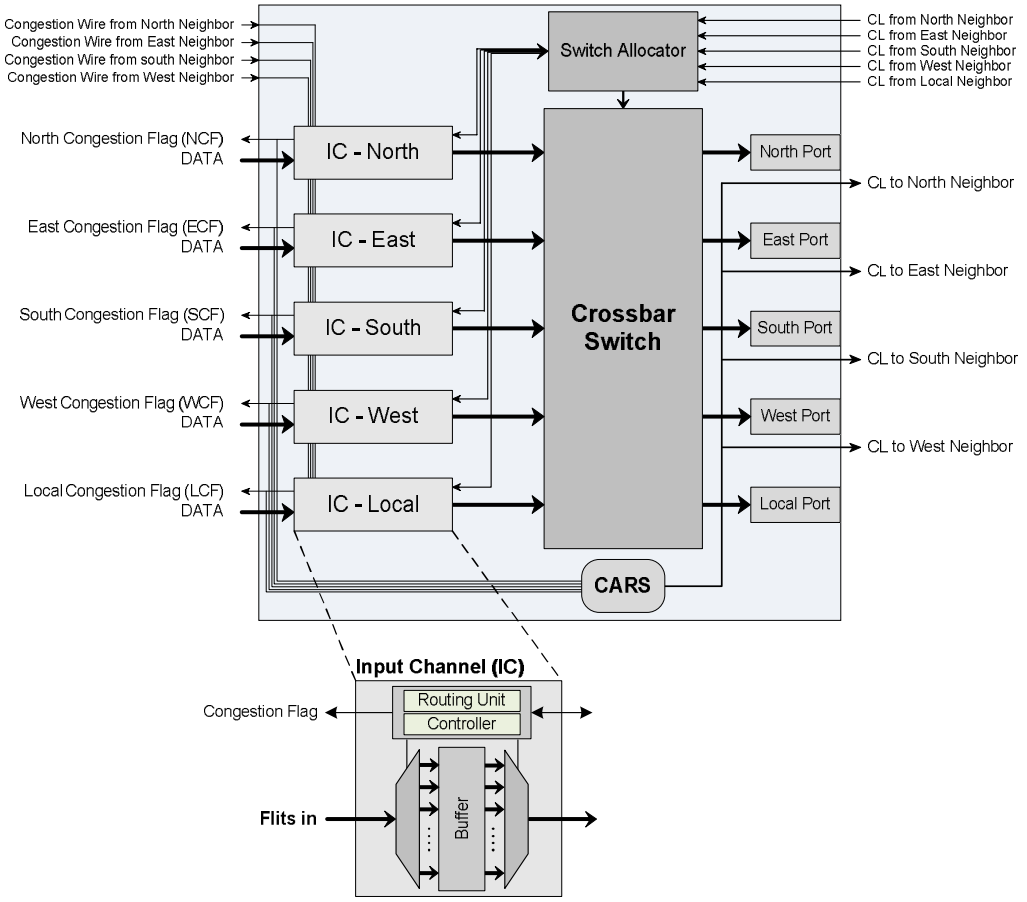


Fig. 4-4. The proposed routing structure.

The status signal of the buffer becomes full when the number of occupied cells of the buffer is more than a threshold value. In this case, for warning the full status, the signal W_Full is activated indicating that most buffer cells are full. This suggests that the congestion condition is traced using the signal W_Full to indicate the filling of the buffer.

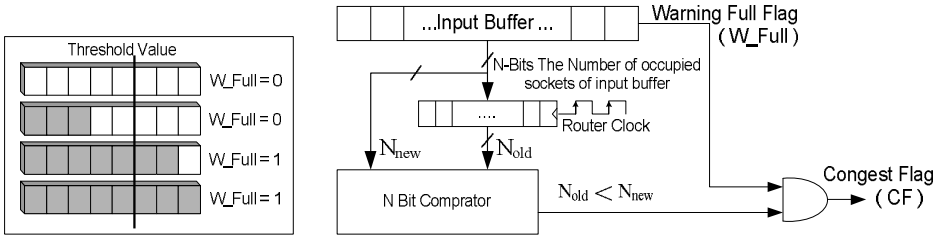


Fig. 4-5. Congestion detection circuit for the input buffer.

As shown in Fig. 4-5, CF is asserted when both the W_Full signal and the positive rate for occupying the input buffer slots ($N_{new} > N_{old}$) are detected. The Congestion Level (CL) of each router is computed by a module called Contention Aware Routing Selection (CARS). The CL is a 3-bit binary number as a result of summing up four CF values from four input ports (see Fig. 4-4 and Fig. 4-6). The CL for each router indicates its load level. For example, if the north and east input buffers of the router are congested (NCF = 1 and ECF = 1), then the CL value of the router will be equal to "010". As illustrated in Fig. 4-6, the output of the CARS module is sent to the corresponding input channels of its adjacent routers (downstream routers).

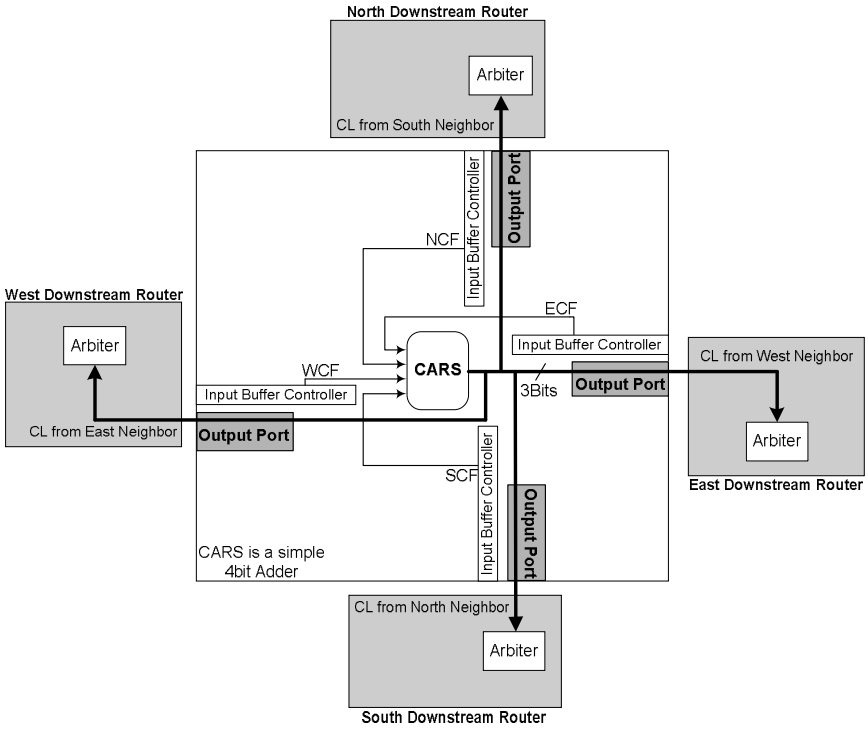


Fig. 4-6. Congestion level computation and transmission scheme.

A. Output-Selection

In the output-selection method, each input channel has a routing unit decoding the header flit of incoming packets. The modified HAMUM, based on the minimal and non-minimal paths, is used to determine the output port to deliver packets. If the route(s) determined from the minimal path routing is(are) congested, the routing unit uses instead the non-minimal path.

First, based on the modified HAMUM in Fig. 4-1, the output port(s) specified by the minimal path (MinPath1 and MinPath2) are examined and if the congestion flag of the neighboring routers of the selected output ports is active, the congestion condition of the non-minimal path is checked. If the non-minimal direction is not congested, the packet is sent to the output port determined by the non-minimal path (NonMinPath). If the neighboring routers are not congested, the packet will be sent through the first minimal path output port (MinPath1). Fig. 4-7 shows the address decoder circuit.

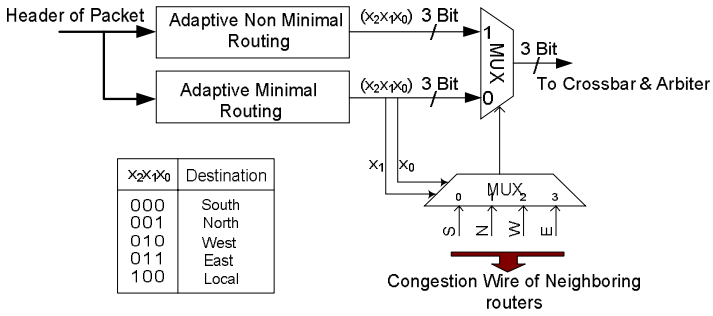


Fig. 4-7. Routing unit circuit.

The procedure of selecting the suitable output port among all output ports that have been specified by the routing unit (Fig. 4-1) is exhibited in Fig. 4-8. In fact, the routing unit chooses the direction in which the corresponding downstream router has not raised its congestion flag. For instance, if a packet with a given source and destination could be routed to both output ports p1 (CF = 1) and p2 (CF = 0), then it will be routed to p2. If p1 and p2 happen to have both their congestion flag raised, and if the routing unit has specified a non-minimal path, p3, the packet will be routed to p3 (if it is not congested), otherwise it will be routed to p1. On top of that, if both p1 and p2 are minimal output directions and the congestion flags of their corresponding downstream router have not risen, the routing unit will route the packet to p1 direction. Moreover, if the header type is a multicast message, the routing unit fetches the destination address from the header. After fetching the destination address from the header, if the destination address is the current node, the routing unit will request the local output port. Meanwhile, the routing unit fetches the next destination address from the header and runs the adaptive routing procedure to determine the output port(s) corresponding to the next destination address.

```
If ( CF(MinPath1) = '0') then
   Select <= MinPath1;
Elsif (MinPath2 /= NULL and CF(MinPath2) = '0') then
   Select <= MinPath2;
Elsif (NonminPath /= NULL and CF(NonminPath) = '0') then
   Select <= NonminPath;
Else
   Select <= MinPath1;
End if;
```

Fig. 4-8.The procedure of selecting the suitable output port.

B. Input-Selection

The proposed arbiter uses the Weighted Round Robin (WRR) scheme derived from the Round Robin (RR) policy for the input-selection method. The presented scheme allows a weight to be assigned to each input port. The weight of each input port, which specifies the number of packets to be transmitted, is proportional to the CL of the upstream router. This will assign different weights to the input channels of the routers for accessing the output channels through the arbitration process.

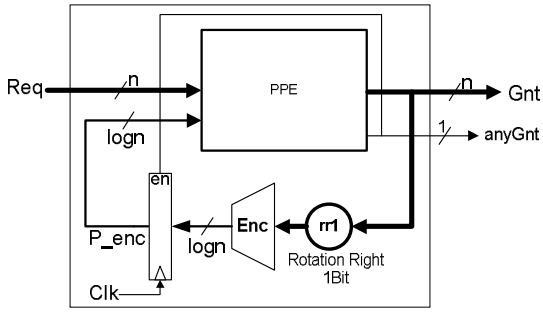


Fig. 4-9. Block diagram of a round-robin arbiter.

The arbiter provides services for each input channel in turn in the round robin order. If the input channel buffer is empty, it will be skipped without being serviced. Fig. 4-9 shows a block diagram of a round robin arbiter [102][103][104]. The arbiter uses a Programmable Priority Encoder (*PPE*) unit to choose the highest priority request from *n* incoming requests (*Req* bus). In every arbitration cycle, PPE, which takes *n* 1-bit-wide requests and the log*n*-bit-wide pointer (*P_enc*) pointing to the current highest-priority request as its inputs, chooses the first nonzero request value beyond (and including) *Req[P_enc]*. The output of the *PPE* is an n-bit-wide *Gnt* (grant) which has at most one nonzero bit and a 1-bit wide *anyGnt* signal which indicates if there has been at least one request. For updating the pointer, *Gnt* is loaded and rotated right one bit in *rr1* unit (rotate right 1-bit register) whose output is encoded using the *Enc* unit and then latched for storing the next *P_enc*.
Fig. 4-10 shows a block diagram of the Weighted Round Robin arbiter derived from the Round Robin scheme. The main difference between two schemes is that WRR serves to the input port based on its CL. There are five registers four out of five registers contain the CL

of their upstream routers and one register is for the local router. The registers have three inputs and one output. If the register enable (*En*) is set, then the new CL value, which shows the CL of the upstream router, will be loaded in the register. After loading, the register operates as the down-counter for the service provided for this input port. While the zero signal (*Zero*) is not set (i.e., the register value has not reached zero) the register value will be decremented in each packet transmission cycle. When the register value reaches zero or the register enable (*En*) is reset, then the zero signal (*Zero*) will be set and subsequently the *Enable* of the *rr1* unit is activated starting the update process for *P_enc* as was performed for the RR scheme. In the situations where there are multiple input requests to the same output channel, each output channel arbiter will service the incoming requests according to their CL (weight). This mechanism resolves any possible starvation that might occur in arbiters based on priority scheme such as in CAIS [95].

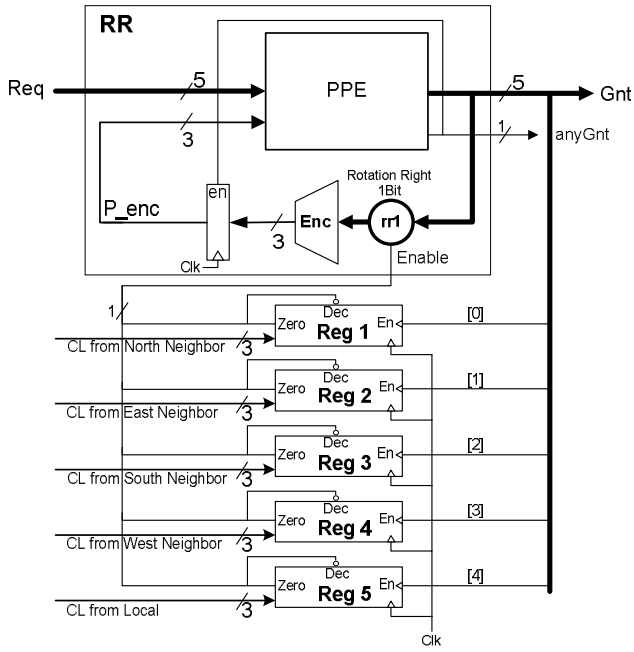


Fig. 4-10. Block diagram of a weighted round robin arbiter.

4.4 Experimental Results

To assess the efficiency of AIOS, four other routers, defined in Table 4-1, are also implemented. We have developed a flit level event driven wormhole NoC simulator implemented in C++ based on standard template libraries (STL), running under Fedora Linux OS. The simulator computes the average latency and the power consumption for the packet transmission. As a performance metric, we use latency defined as the number of

cycles between the initiation of a message operation issued by a PE and the time when the message is completely delivered to the destination PE. The request rate is defined as the ratio of the successful message injections into the network interface over the total number of injection attempts.

Table 4-1. Structure of other four routers.

Router	P-OE	P-MP	RR-OE	RR-MP
Input-Selection	Priority (CAIS)	Priority (CAIS)	Round Robin	Round Robin
Output-Selection	Odd-Even	Multi-Path	Odd-Even	Multi-Path

A 2D mesh configuration has been used for the NoC. Each router consists of 8 unidirectional channels (four incoming and four outgoing channels). The simulator inputs include the array size, the router operation frequency, the input and output selection methods, the link width, and the traffic type.

To estimate the power consumption, we have used Orion library functions [86]. Since some components such as routing unit and WRR circuits have not been modeled in Orion, we have modified the Orion library for computing their power consumptions. The data width and the frequency were set to 32 bits and 1GHz, respectively, which leads to a bandwidth of 32 Gb/s. Each input channel has a buffer (FIFO) size of 8 flits with the congestion threshold set at 75% of the total buffer capacity. The packet size was assumed to be 5 flits. The time needed to generate multicast messages (packets) is not considered, because we assumed the multicast messages are generated in the processing elements. The array size of 8×8 has been considered.

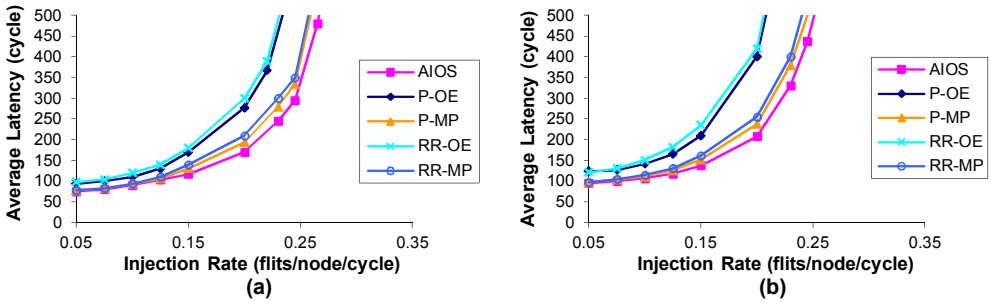


Fig. 4-11. Performance results in 8×8 2D-mesh under multicast traffic profile with (a) 10 destinations, (b) 20 destinations.

4.4.1 Performance Evaluation

A. Multicast Traffic Profile

This simulation is performed using a uniform-based multicast traffic profile pattern. Each PE generates messages and injects them into the network using the time intervals which are

obtained using the exponential distribution. In the multicast traffic profile, each PE sends a message to a set of destinations. A uniform distribution is used to construct the destination set of each multicast message [78]. The number of destinations has been set to 10 and 20. The average latency as a function of the average flit injection rate is shown in Fig. 4-11(a) and (b). As shown in the results, AIOS leads to the lowest delay particularly not only in high traffic loads but also when the number of multicast destinations increases. As described before and can be seen from Fig. 4-11(a) and (b), unicast-based routing algorithms, e.g. Odd-Even, are not efficient for multicast traffic [78][81].

B. Unicast and Multicast (Mixed) Traffic Profile

In this experiment, we have employed a mixture of unicast and multicast traffic, where 80% of injected messages are unicast messages and the remaining 20% are multicast messages. This pattern may be representative of the traffic in a distributed shared-memory multiprocessor where updates and invalidation produce multicast messages and cache misses are served by unicast messages [78]. Both unicast and multicast messages are routed using HAMUM. The number of destinations for multicast messages is set to 10 and the array size of the network is equal to the previous traffic profile. Uniform and hotspot synthetic traffic patterns [64][105] are used to generate the unicast traffic in the network. In the uniform traffic profile, each PE sends a message to any other PE with an equal probability. This probability is determined randomly using a uniform distribution. Under the hotspot traffic pattern, one or more nodes are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In Fig. 4-12(a) the average communication latency of different routers under the uniform traffic model for unicast traffic are shown. In this traffic, AIOS performs better than the other three algorithms. Under the hotspot traffic model, given a hotspot percentage of h, a newly generated message is directed to each hotspot node with an additional h percent probability. We simulate hotspot traffic with a single hotspot node. The hotspot node is chosen to be node (4, 4) in the 8×8 2D-Mesh with h=10%. As observed from Fig. 4-12(b), AIOS shows considerably smaller delays compared to the other router models.

C. Unicast Traffic Profile

For appraising the unicast efficiency of AIOS, the uniform and hotspot traffic profiles, where 100% of injected messages are unicast messages have been considered. Fig. 4-13(b) and (b) show the simulation results for the uniform and hotspot traffic profiles. As depicted, when the injection rate is increased, AIOS is superior to the other schemes. In brief, as the injection rate increases, AIOS leads to smaller average delays. This is due to the fact that the input selection uses WRR scheme which allows packet flows coming from congested paths to be serviced more often according to their congestion level. In contrast, in the RR scheme no matter how congested a path is, all packet flows are serviced equally. In the mechanism based on CAIS (priority), congested input channels which have higher numbers

of request are serviced more often while the input channels with lower traffic may not be serviced leading to the starvation problem.

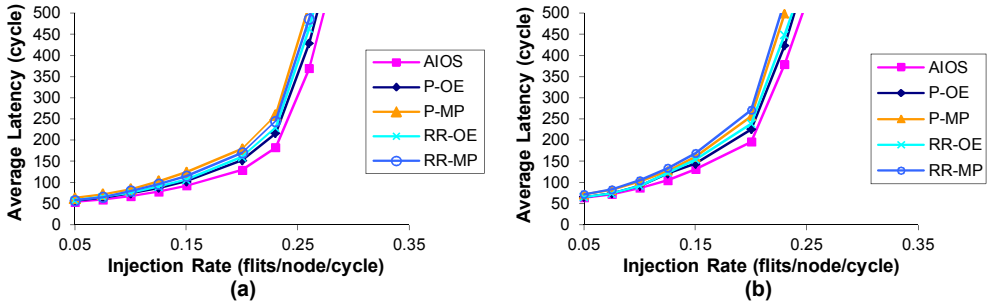


Fig. 4-12. Performance with different loads in 8×8 2D-mesh under mixed traffic (20% multicast and 80% unicast). Unicast traffic in (a) is based on the uniform pattern and in (b) is based on the hotspot pattern with h=10%.

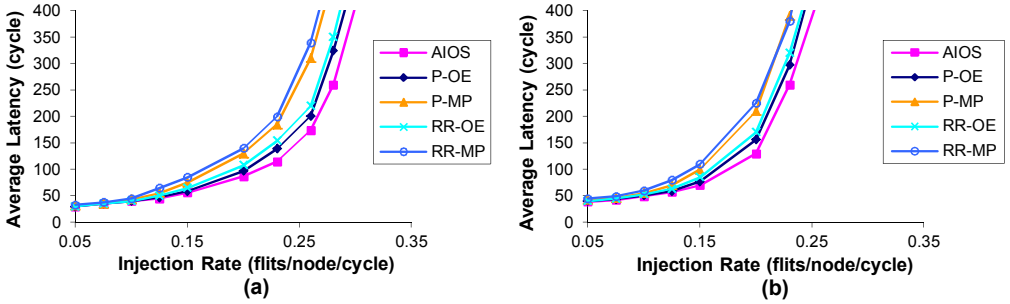


Fig. 4-13. Performance with different loads in 8×8 2D-mesh under unicast traffic: (a) the uniform pattern and (b) the hotspot pattern.

D. Video Object Plane Decoder (VOPD) Traffic Profile

To evaluate the performance of AIOS under more realistic traffic loads, we have used Video Object Plane Decoder (VOPD) traffic profile [106]. Although our algorithm is not proposed for real-time applications, we have only used VOPD traffic profile as an example of real traffic profile without considering its real-time constraints. The mesh array size was assumed to be 6×5. In Fig. 4-14, we show the core graph and its mapping onto the mesh for the VOPD. The other cores around the grey box generate uniform traffic, where each PE generates 5-flit packets and injects them into the network in the uniform manner. As the results shown in Fig. 4-15 reveal, for this traffic model, in the central areas of the chip, congestion may occur. Therefore, since the presented routers are based on the adaptive routing algorithms, they do not send packets in the central areas when these areas are

congested and thus distribute the traffic over the rest of the chip area. This strategy reduces the average delay of the packet transportation.

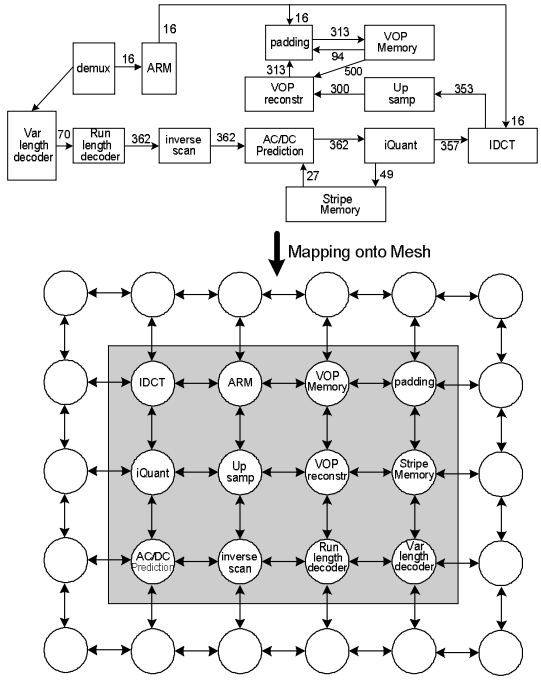


Fig. 4-14.The VOPD block diagram, with communication BW annotated (in MB/s) and its mapping onto mesh topology.

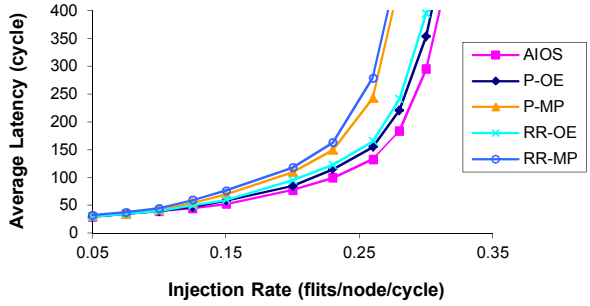


Fig. 4-15. The performance of different algorithms under VOPD traffic model.

4.4.2 Power Dissipation

Using the simulator, the power dissipation of each scheme is calculated and compared under the mixed traffic profile. The results for the average and the maximum power under mixed traffic are shown in Fig. 4-16(a) and (b), respectively. Both average and maximum power values are computed near the saturation point, 0.23 (flits/cycle), under mixed traffic. We can notice that the maximum power, compared to other routers, is considerably lowered in our proposed router. This is achieved by smoothly distributing the power consumption over the network using the output selection scheme which reduces the number of the hotspots and, hence, lowering the maximum power.

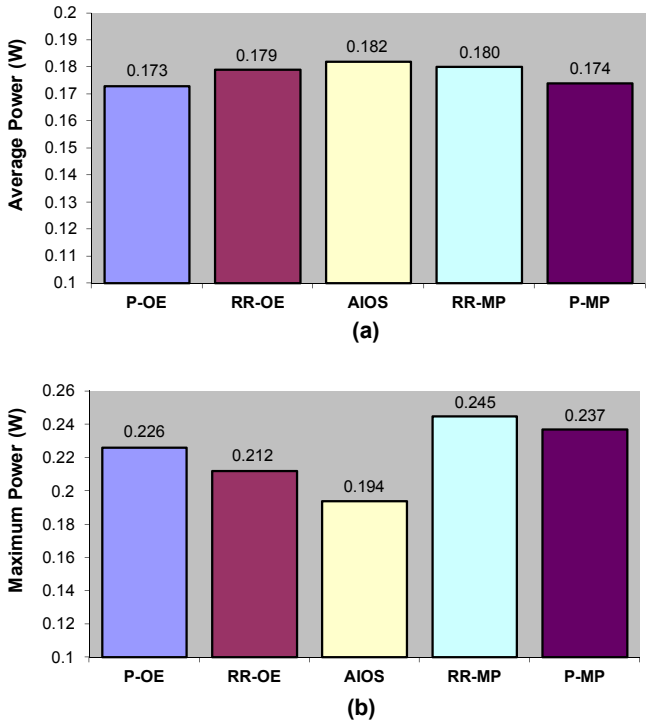


Fig. 4-16. (a) Average and (b) Maximum power dissipation results in 8×8 2D-mesh under mixed traffic profile.

4.4.3 Hardware Overhead

To evaluate the area overhead of the presented model and demonstrate the performance/area trade-off, RTL models of aforementioned routers have been implemented with four different input-output selection schemes using VHDL. The routers were described in VHDL and synthesized with Synopsys D.C. using the CMOS STMicroelectronic 65nm technology. For all routers, the data width was set to 32 bits, and each input channel has a buffer size of 10 flits. The FIFOs were implemented in our design using registers in order

to achieve better performance/power efficiency. We performed place-and-route via Cadence SoC-Encounter for more accurate area estimation. Fig. 4-17 shows the area cost of the switches. Comparing the area cost of AIOS with P-MP, P-OE, RR-MP and RR-OE introduces 2.4%, 2.1%, 1.6% and 1.3% additional overhead respectively. A test chip with a 2×2 lightweight NoC based on the presented router architecture has been fabricated which can be found in Appendix A.

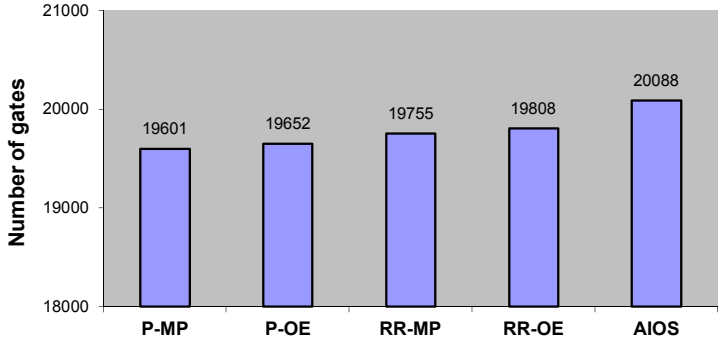


Fig. 4-17. Area cost of routers for implementing different input-output selections.

4.5 Summary

In this chapter, a router architecture based on the adaptive input and output selection is presented. The output selection of the presented router utilizes an adaptive routing algorithm supporting both unicast and multicast traffic while the input selection part of the router uses the weighted round robin arbitration. Also, the adaptive output selection algorithm supporting both minimal and non-minimal paths uses congestion flags to route packets through non-congested paths and consequently helps balance the traffic. The WRR input selection also assists in relieving nodes where congestion is formed. A simulator was used to evaluate the efficiency of the proposed router. Under the multicast, unicast, mixed, and VOPD traffic models and in high flit injection rates, the proposed architecture has the lowest average communication delay in comparison with the other router models. It also reduces the maximum power dissipation of the network compared to other models under mixed traffic model.

Chapter 5

Adaptive Network Interface Architecture

NoCs are composed of routers connecting PEs, to deliver the data (packets) from one place to another [94], and Network Interfaces (NI) acting as communication interfaces between each PE and corresponding router. The fundamental function of network interfaces is to provide data transaction between PEs and the network infrastructure. That is, one of the practical approaches of network interfaces is to translate the protocol between the PE and router based on a standard communication protocol such as AXI [13], OCP [14], and DTL [15].

In MPSoCs, in-order delivery is a practical approach which should be handled when exploiting an adaptive routing algorithm for distributing packets through the network [107], when obtaining memory access parallelization by sending requests from a master IP core to multiple slave memories [108][109], or when exploiting dynamic memory access scheduling in memory controller to reorder memory requests [110].

In this chapter, we present a memory-efficient on-chip network with adaptive interfaces not only to cope with the in-order delivery but also to improve the network performance. The key ideas are threefold.

1) The first idea is to deal with out-of-order handling in such a way that when a master IP-core sends requests to different memories, the responses might be required to return in the same order in which the master issued the addresses. Therefore, we introduce an adaptive network interface architecture using a reordering mechanism for the proposed on-chip network. In addition, resource utilization of reorder buffers, implemented in network interfaces, is significantly inefficient, inasmuch as conventional buffer management is not efficient enough for network resources. Thus, a streamlined adaptive reordering mechanism via resourceful management of buffers is implemented in the network interface.
2) As in traditional reordering mechanisms routers do not play any role in the reordering procedure, employing routers in the reordering procedure is very useful to increase utilization and reduce the average delay of on-chip networks. Thus, the second idea is

an on-chip router architecture, called priority-based router, which assigns a priority value for each packet according to the sequence number and distance between source and destination.

3) The third idea is a dynamic memory controller that is integrated into the proposed network interface. The presented memory controller is able to reorder memory requests adaptively to improve memory utilization and reduce both memory and network latencies.

Based on the introduced network interface architecture, a hybrid network interface architecture is designed to integrate both memory and processor in a tile. Furthermore, the presented on-chip network exploits the AMBA AXI protocol to allow backward compatibility with existing IP cores [13]. We also present micro-architectures of the proposed ideas, particularly the reordering mechanism.

5.1 DRAM Structure

DRAM is designed to provide high memory depth and bandwidth. Fig. 5-1 shows a simplified three dimensional architecture of an DRAM memory chip with the dimensions of bank, row, and column [110][113][114]. A DRAM chip is composed of multiple independent memory banks such that memory requests to different banks can be serviced in parallel. Hence, a benefit of a multibank architecture is that commands to different banks can be pipelined. Each bank is formed as a two dimensional array of DRAM cells that are accessed an entire row at a time. Thus, a location in the DRAM is identified by an address consisting of bank, row, and column fields. A complete DRAM access may require three commands (transactions) in addition to the data transfer: bank precharge, row activation, and column access (read/write). A bank precharge charges and prepares the bank, while a row-activation command (with the bank and row address) is used to copy all data in the selected row into the row buffer, i.e. sense amplifier. The row buffer serves as a cache to reduce the latency of subsequent accesses to that row. Once a row is in the row buffer, then column commands (read/write) can be issued to read/write data from/into the memory addresses (columns) contained in the row. To prepare the bank for a next row activation after completing the column accesses, the cached row must be written back to the bank memory array by the precharge command [110]. Also, the timing constraints associated with bank precharge, row activation, and column access are t_{RP}, t_{RCD}, and t_{CL} respectively [110][113][114]. Since the latency of a memory request depends on whether the requested row is in the row buffer of a bank or not, a memory request could be a *row hit*, *row conflict,* or *row empty* with different latencies [115]. A *row hit* occurs when a request is accessing a row currently in the row buffer and only a read or a write command is needed. It has the lowest bank access latency (t_{CL}) as only a column access is required. A *row conflict* occurs when the access is to a row different from the one currently in the row buffer. The contents of the row buffer first need to be written back into the memory array using the precharge command. Afterward, the required row should be opened and accessed

using the activation and read/write commands. The *row conflict* has the highest bank access latency (t_{RP} +t_{RCD} +t_{CL}). If the bank is closed (precharged) or there is no row in the row buffer then a *row empty* occurs. An activation command should be issued to open the row followed by read or write command(s). The bank access latency in this case is t_{RCD} +t_{CL}.

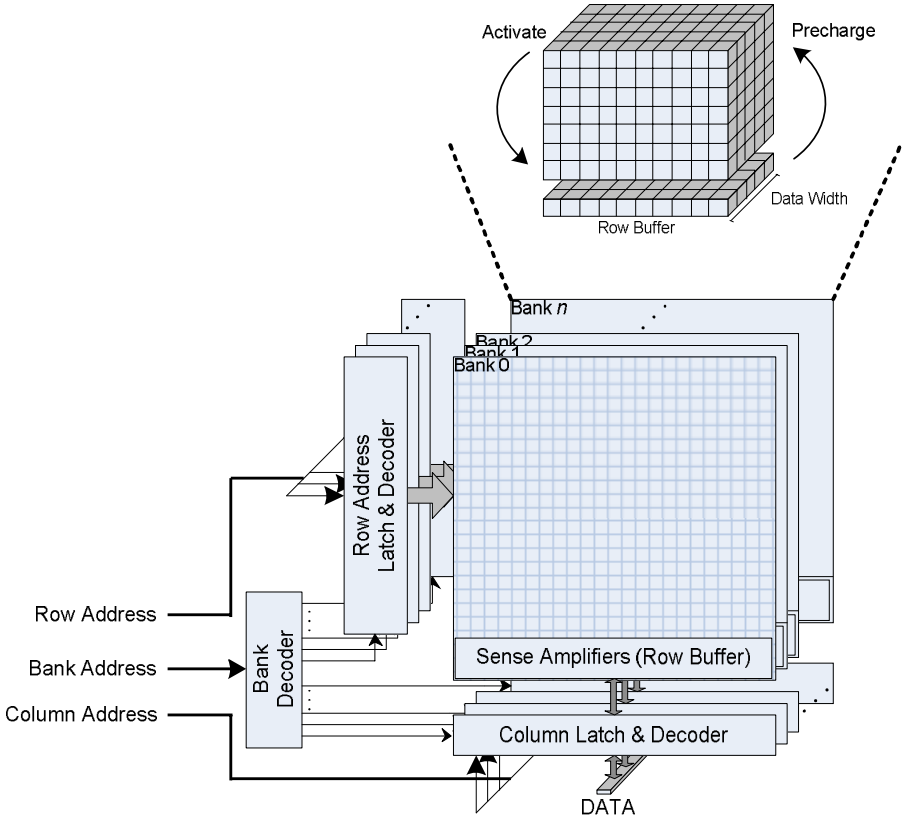


Fig. 5-1. High-level structure of an SDRAM.

5.1.1 Memory Access Scheduling

The memory controller lies between processors and the DRAM to generate the required commands for each request and to schedule them on the DRAM buses. The memory controller consists of a request table, request buffers, and a memory access scheduler. A request table is used to store the state of each memory request, e.g. valid, address, read/write, header pointer to the data buffer and any additional state necessary for memory scheduling. The data of outstanding requests are stored in read and write buffers. The read and write buffers (request buffers) are implemented as linked lists. Each memory request (read and write) allocates an entry in its respective buffer until the request is completely

serviced. Among all pending memory requests, based on the state of the DRAM banks and the timing constraints of the DRAM, the memory scheduler decides which DRAM command should be issued. The average memory access latency can be reduced and the memory bandwidth utilization can be improved if an efficient memory scheduler is employed [110][113][114]. Fig. 5-2 reveals how the memory access scheduling affects the performance. As shown in the figure, the sequence of four memory requests is considered. Request 1 and 3 are row empties, and request 2 and 4 are row conflicts. Timing constraints of a DDR2-512MB used as example throughout this chapter are 2-2-2 (t_{RP}-t_{RCD}-t_{CL}) [116]. As depicted in Fig. 5-2(a), if the controller schedules the memory requests in-order, it will take 22 memory cycles to complete them. In Fig. 5-2(b) the same four requests are scheduled out-of-order. As can be seen, request 4 is scheduled before request 2 and 3 to turn request 4 from a row conflict to a row hit. In addition, request 3 is pipelined after request 1, called *bank interleaving*, since it has the different bank address from the bank address of request 1. As a result, only 14 memory cycles are needed to complete the four requests. Thus, how the memory scheduler can improve the memory performance has been shown by this example where the memory utilization of the in-order scheduler and the out-of-order are 4(data)/22(cycle) = 18% and 4/14= 29%, respectively. In this chapter, we present an optimized memory controller that is integrated into the proposed network interface to improve the memory utilization and reduce both memory and network latencies.

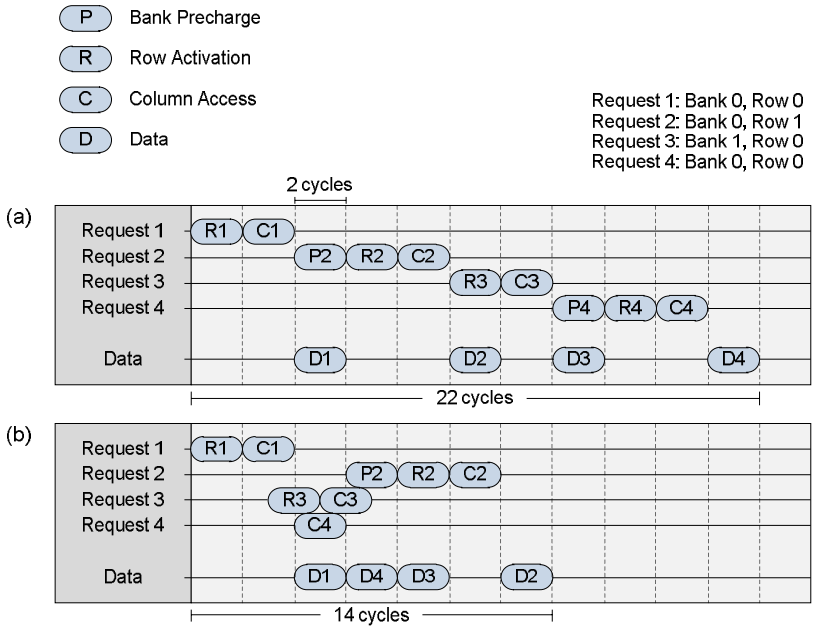


Fig. 5-2. Memory access scheduling of four memory requests with (a) in-order and (b) out-of-order access scheduling.

5.2 Related Work

Due to the fact that most of the recently published studies have focused on design and description of NoC architectures, there has been relatively little attention to network interface designs particularly when supporting out-of-order mechanisms [111]. The authors in [108] present ideas of transaction ID renaming and distributed soft arbitration in the context of distributed shared memories. In such a system, because of using global synchronization in the on-chip network, the performance might be degraded and the cost of hardware overhead for the on-chip network is too high. In addition, the implementation of ID renaming and reorder buffer can suffer from low resource utilization. This idea has been improved in [112] by moving reorder buffer resources from the network interface into network routers. In spite of increasing the resource utilization, the delay of release packets recalling data from distributed reordering buffer can significantly degrade the performance when the size of the network increases [112]. Moreover, the proposed architecture is restricted to deterministic routing algorithms, and thus, it is not a suitable method for an adaptive routing. However, neither [104] nor [112] has presented a micro-architecture of the network interface. An efficient on-chip network interface supporting shared memory abstraction and flexible network configuration is presented by Radulescu et al. [109]. The proposed architecture has the advantage of improving reuse of IP cores, and offers ordering messages via channel implementation. Nevertheless, the performance is penalized because of increasing latency, and besides, the packets are routed on the same path in the NoC, which forces routers to use the deterministic routing. Yang et al. proposed NISAR [107], a network interface architecture using the AXI protocol capable of packet reordering based on a look up table; NISAR has been implemented under the assumption of fixed message size and enjoys simple control logic and design. However such a mechanism would lead to an inefficient use of network resources for applications that generate periodic variable-size messages (burst mode). Moreover, NISAR suffers from several disadvantages described as follows. First, it permits only a limited number of transaction IDs to send several packets to the network so that some requests with different transaction IDs are prevented to be serviced for a long period. Second, NISAR uses a statically partitioned reorder buffer suffering from low resource utilization. Third, NISAR presented only a hybrid interface where the master and slave IP-cores are integrated into a single node; however it imposes significant hardware and delay overhead when the master and slave IP-cores are not integrated into a single node.

In routers, the arbitration process is performed to choose one of multiple input channels to access an output channel. The arbiter could follow either a non-priority or a priority scheme. In the non-priority method, when there are multiple input port requests for the same available output port, the arbiter uses the First-Come-First-Served (FCFS) [65], also called First-In-First-Out (FIFO), or Round-Robin (RR) [97] policy to grant access to an input port. In this way the starvation on a particular port is avoided (fair). On the other hand, in the priority method when there are multiple input port requests for the same available output port, the arbiter would grant access to the input port request which has the highest priority level [95]. The problem with the priority method is that starvation could

occur (unfair). In this chapter, we introduce a fair priority-based router to improve the network performance with low hardware overhead.
Regarding the memory scheduler, several memory scheduling mechanisms were presented to improve the memory utilization and to reduce the memory latency. The key idea of these mechanisms is on the scheduler for reordering memory accesses. The memory access scheduler proposed in [110] reorders memory accesses to achieve high bandwidth and low average latency. In this scheme, called bank-first scheduling, memory accesses to different banks are issued before those to the same bank. Shao et al. [134] proposed the burst scheduling mechanism based on the row-first scheduling scheme. In this scheme, memory requests that might access the same row within a bank are formed as a group to be issued sequentially, i.e. as a burst. Increasing the row hit rate and maximizing the memory data bus utilization are the major design goals of burst scheduling. The core-aware memory scheduler reveals that it is reasonable to schedule the requests by taking into consideration the source of the requests because the requests from the same source exhibit better locality [114]. In [113], the authors introduced an SDRAM-aware router to send one of the competing packets toward an SDRAM using a priority-based arbitration. An adaptive history-based memory scheduler which tracks the access patterns of recently scheduled accesses and selects memory accesses matching the pattern of requests is proposed in [135] and [136]. As NoCs are strongly emerging as a communication platform for chip-multiprocessors, the major limitation of presented memory scheduling mechanisms is that none of them did take the order of the memory requests into consideration. As discussed earlier, requests with the same transaction ID from the same master must be completed (turn back) in-order. While requests would be issued out-of-order in memories (slave-sides), the average network latency might be increased significantly due to the out-of-order mechanism in master sides. Therefore, it is necessary to consider the order of memory requests for making an optimal memory scheduling.
The major contribution of this chapter is to propose an adaptive network interface architecture within a dynamic buffer allocation mechanism for the reorder buffer to increase the utilization and overall performance. That is, using dynamic buffer allocation to get more free slots in the reorder buffer may lead more messages to be entered to the network. On top of that, an efficient memory scheduler mechanism based on the order of requests is introduced and integrated in our network interface to diminish both the memory and network latencies. We also present a novel router architecture for incorporating network resources to help in serializing the packets while they progress towards their destinations.

5.3 Proposed Network Interface Architecture

Since IP cores are classified into masters and slaves, the network interface is also divided into the master network interface (Fig. 5-3) and slave network interface (Fig. 5-4). Both network interfaces are partitioned into two paths: forward and reverse. The forward path transmits the AXI transactions received from an IP core to a router; and the reverse path

receives the packets from the router and converts them to AXI transactions. The proposed network interfaces for both master and slave sides are described in detail as follows.

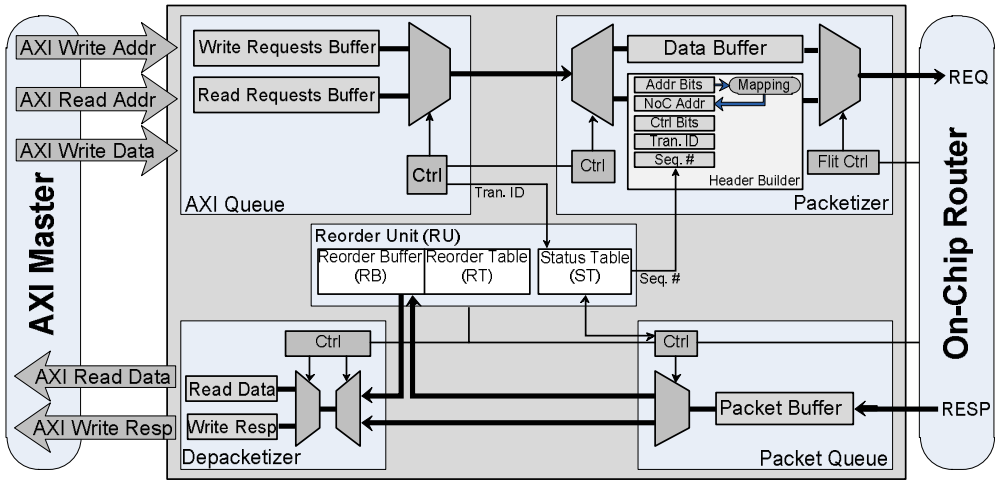


Fig. 5-3. Master-side network interface architecture.

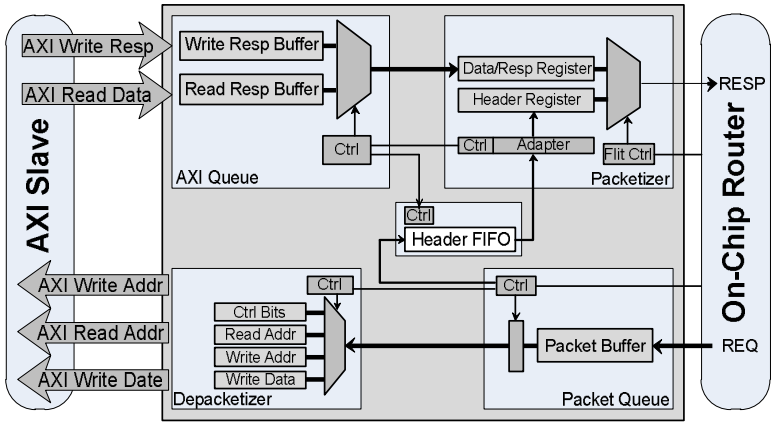


Fig. 5-4. Slave-side network interface architecture.

5.3.1 Master-side Network Interface

As shown in Fig. 5-3, the forward path of the master network interface transferring requests to the network is composed of an AXI-Queue, a Packetizer unit, and a Reorder unit, while the reverse path, receiving the responses from the network, is composed by a Packet-

Queue, a Depacketizer unit, and the Reorder unit. The Reorder unit is a shared module between the forward and reverse paths.

AXI-Queue: the AXI master transmits write address, write data, or read address to the network interface through channels. The AXI-Queue unit performs the arbitration between write and read transaction channels and stores requests in either write or read request buffer. The request messages are sent to the packetizer unit if admitted by the reorder unit, and on top of that a sequence number (*SN*) for each request should be prepared by the reorder unit after the admittance.

Packetizer: it converts incoming messages from the AXI-Queue unit into header and data flits, and delivers the produced flits to the router. Since a message is composed of several parts, the data is stored in the data buffer and the rest of the message is loaded in corresponding registers of the header builder unit. After the mapping unit converts the AXI address into a network address by using an address decoder, based on the request information loaded on related registers and the sequence number provided by the reorder buffer, the header of the packet can be assembled. Afterward, the flit controller wraps up the packet for transmission.

Packet-Queue: this unit receives packets from the router; and according to the decision of the reorder unit a packet is delivered to the depacketizer unit or reorder buffer. In fact, when a new packet arrives, the sequence number and transaction ID of the packet are sent to the reorder unit. Based on the decision of the reorder unit, if the packet is out-of-order, it is transmitted to the reorder buffer, and otherwise it is delivered to the depacketizer unit directly.

Depacketizer: the main functionality of the Depacketizer unit is to restore packets coming from either the packet queue unit or reorder buffer into the original data format of the AXI master core.

Reorder unit: it is the most influential part of the network interface including a Status-Table, a Reorder-Buffer, and a Reorder-Table. In the forward path, preparing the sequence number for corresponding transaction ID and avoiding overflow of the reorder buffer (by an admittance mechanism), are provided by this unit. On the other side, in the reverse path, this unit determines where the outstanding packets from the packet queue should be transmitted (reorder buffer or depacketizer), and when the packets in the reorder buffer should be released to the depacketizer unit.

Status-Table: the state of outstanding messages is kept in a table named Status-Table. The Status-Table has n entries where each entry corresponds to a transaction ID and n is the number of AXI transaction IDs. Each entry contains the information of outstanding messages associated with that transaction ID and includes *NM, ES,* and *LMS* fields. The *NM* (Number of outstanding Messages) field reveals that how many messages of the given *T-ID* are inside the network. This value is incremented when a new message with the same *T-ID* enters the network, and is decremented when the response message comes back to the master core. The *ES* (Expecting Sequence number) field points out the sequence number of the message expected to be delivered to the master core.

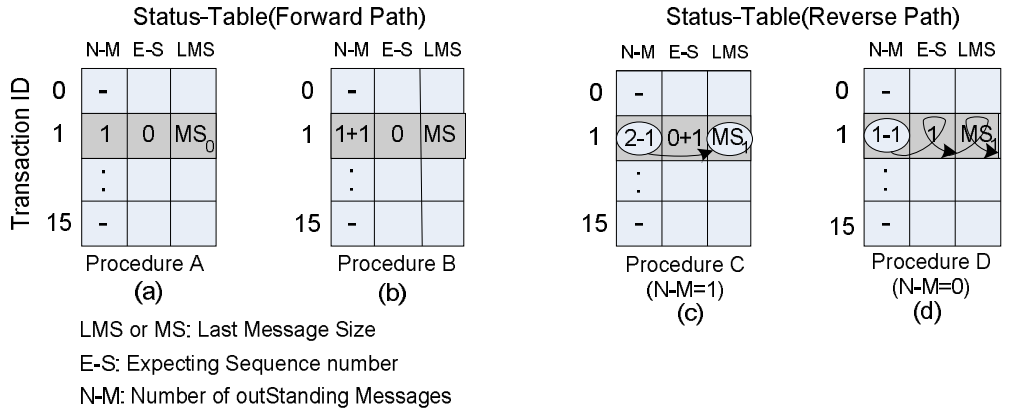


Fig. 5-5. Status-Table of the reorder unit.

As the master core expects to receive the first message first, the *ES* field is set to 0 at the initialization time and it is increased by receiving in-order messages. As already mentioned, each message has a sequence number (*SN*) indicating the order of the message within the transaction ID. This value is produced by the reorder unit, if the admittance is given. Finally, the *LMS* field defines the reserved buffer space for the last message. The Status-Table might be updated in both forward and reverse paths described as follows. Suppose that in the forward path, the first message of a transaction ID requests to enter the network. The corresponding row of the transaction ID is initiated such that the *NM, ES* and *LMS* fields are set to 1, 0, and the message size, respectively, and the value of *SN* is initialized to 0. However, as no ordering mechanism is required for a single outstanding message of the given transaction ID, no buffer space needs to be reserved for this message (Procedure A, Fig. 5-5(a)). For the second (or the rest of) admitted requests of the given transaction ID, the *NM* field is increased by +1, the *ES* field remains unchanged, and the *LMS* field is set to the required buffer size of the new message. Subsequently, the value of *SN* is obtained by adding the values of *NM* and *ES*. Since more than one message with the same transaction ID is issued ($NM \geq 2$), the out-of-order handling mechanism is required. Therefore, in order to prevent overflow of the reorder buffer, the buffer space required by the new message is compared with the available space of the reorder buffer. If there is enough space for the new message (*MsgSize*), the required space is allocated in the reorder buffer (Procedure B, Fig. 5-5(b)). The *RsrvSize* indicates the required space of all outstanding transactions in the network. Indeed, this register reserves the number of buffer slots required by outstanding messages of different transaction IDs.

Procedure A:(sending first msg. of T_ID to network)
1 S_Table(T_ID)(NM) <= “0001”;
2 S_Table(T_ID)(ES) <= (others =>’0’);
3 S_Table(T_ID)(LMS) <= MsgSize;
4 SN <= (others =>’0’);
5 RsrvSize <= RsrvSize;

Procedure B:(sending other msgs. of T_ID to network)
1 S_Table(T_ID)(NM) <= S_Table(T_ID)(NM) + 1;
2 S_Table(T_ID)(ES) <= S_Table(T_ID)(ES);
3 S_Table(T_ID)(LMS) <= MsgSize;
4 SN <= S_Table(T_ID)(NM) + S_Table(T_ID)(ES);
5 RsrvSize <= RsrvSize + MsgSize;

In the reverse path, the transaction ID and sequence number of the arriving response message are sent to the reorder unit to find the related row in the Status-Table. In the corresponding row of the transaction ID, if the sequence number of the incoming packet is equal to the value of *ES*, the packet is an expected packet (in-order) and it should be delivered to the depacketizer unit. Thereafter, the received message size (*RecvMsgSize*) is reduced from the *RsrvSize,* and the values of *ES* and *NM* are added by +1 and -1, respectively (Procedure C, Fig. 5-5(c)). However, if the sequence number of the packet is not equal to the value of *ES,* the packet is out-of-order and should be delivered to the reorder buffer. In case that the message is delivered to the depacketizer unit and the value of *NM* becomes 1, the reserved buffer space for the last message (i.e. *LMS*) can be deallocated. If the value of *NM* reaches 0, the transaction is terminated (Procedure D, Fig. 5-5(d)).

Procedure C: (arriving expected packet)
1 S_Table(T_ID)(NM) <= S_Table(T_ID)(NM) - 1;
2 S_Table(T_ID)(ES) <= S_Table(T_ID)(ES) + 1;
3 S_Table(T_ID)(LMS) <= S_Table(T_ID)(LMS);
4 RsrvSize <= IF NM/=1 THEN RsrvSize – RecvMsgSize;
ELSE RsrvSize – RecvMsgSize - LMS;

Procedure D: (arriving last packet)
1 S_Table(T_ID)(NM) <= S_Table(T_ID)(NM) - 1;
2 S_Table(T_ID)(ES) <= (others =>'0');
3 S_Table(T_ID)(LMS) <= (others =>'0');
4 RsrvSize <= RsrvSize;

Reorder-Table and Reorder-Buffer: As shown in Fig. 5-6, each row of the Reorder-Table corresponds to an out-of-order packet stored in the Reorder-Buffer. This table includes the valid tag (*v*), the transaction ID (*T-ID*), the sequence number (*SN*) as well as the head pointer (*P*). In the Reorder-Buffer, the flits of each packet are maintained by a linked list structure providing high resource efficiency with a little hardware overhead. On top of that, the goal of using the shared Reorder-Buffer is to support variable packet sizes and improve the buffer utilization which can also increase the performance by feeding more packets into the network. Fig. 5-6 exhibits a pointer field adopted to indicate the next flit position in the Reorder-Buffer. Using the proposed structure in Fig. 5-6, each out-of-order packet updates the Reorder-Table and Reorder-Buffer according to the procedure E and F.

Procedure E: (updating Reorder-Table)
1 ReorderTable(FreeRow)(V) <= '1';
2 ReorderTable(FreeRow)(T-ID) <= HeaderFlit(T-ID);
3 ReorderTable(FreeRow)(SN) <= HeaderFlit(SN);
4 ReorderTable(FreeRow)(P) <= Current_Free_Slot;

The first three operations in the procedure E, stores the transaction ID and sequence number from the header flit of the out-of-order packet to the available slot indicated by *FreeRow* in the Reorder-Table; and the last operation in the procedure E updates the pointer to point to the available slot in the Reorder-Buffer.

Procedure F: (updating Reorder-Buffer)
1 ReorderBuf(Current_Free_Slot)(V) <= '1';
2 ReorderBuf(Current_Free_Slot)(Data) <= flit;
3 ReorderBuf(Current_Free_Slot)(P) <= Next_Free_Slot;
4 Current_Free_Slot <= Next_Free_Slot;

The procedure F is intended to store the incoming flits into the Reorder-Buffer. While *Current_Free_Slot* shows the current free location in the Reorder-Buffer to store the current flit, *Next_Free_Slot* returns an available slot for the next flit. By repeating the operations in the procedure F, all the payload flits are stored in the Reorder-Buffer. The tail flit can be determined by extracting header flit information. Whenever an in-order packet delivered to the depacketizer unit, the depacketizer controller checks the Reorder-Table for the validity of any stored packet with the same transaction ID and next sequence number. If so, the stored packet(s) is (are) released from the reorder unit to the depacketizer unit.

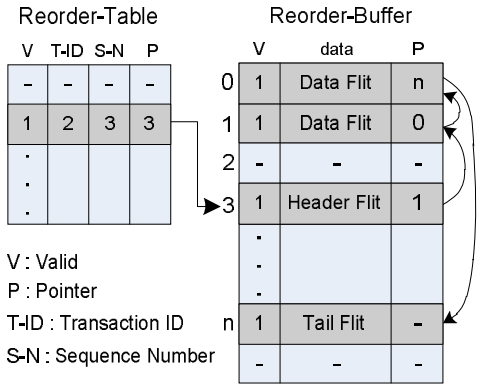


Fig. 5-6. Dynamic buffer allocation.

If master cores, slave cores, and the network operate at different frequencies, bi-synchronous FIFOs are deployed between network interfaces and cores. Bi-synchronous FIFOs are widely used in multi-clock systems to synchronize signals from different

clock/frequency domains. Each domain is synchronous to its own clock signal but can be asynchronous with respect to others in either clock frequency or phase [117]. The challenges of designing bi-synchronous FIFOs include the enhancement of reliability and reducing latency and power/area cost. We identify the bi-synchronous FIFOs structure presented in [118] suitable to be used in the interfaces.

5.3.2 Slave-side Network Interface

A slave IP core cannot operate independently. It receives requests from master cores and responds to them. Hence, using reordering mechanism in the slave network interface is completely meaningless. To avoid losing the order of header information (transaction ID, sequence number, and etc.) carried by arriving requests, a FIFO has been considered. After processing a request in the slave core, the response packet should be created by the packetizer. As can be seen from Fig. 5-4, to generate the response packet, after the header content of the corresponding request is invoked from the FIFO, and some parameters of the header (destination address, and packet size, and etc) are modified by the adapter, the response packet can be formed. However, the components of slave-side interface in both forward and reverse paths are similar to the master-side interface components, except the reorder unit.

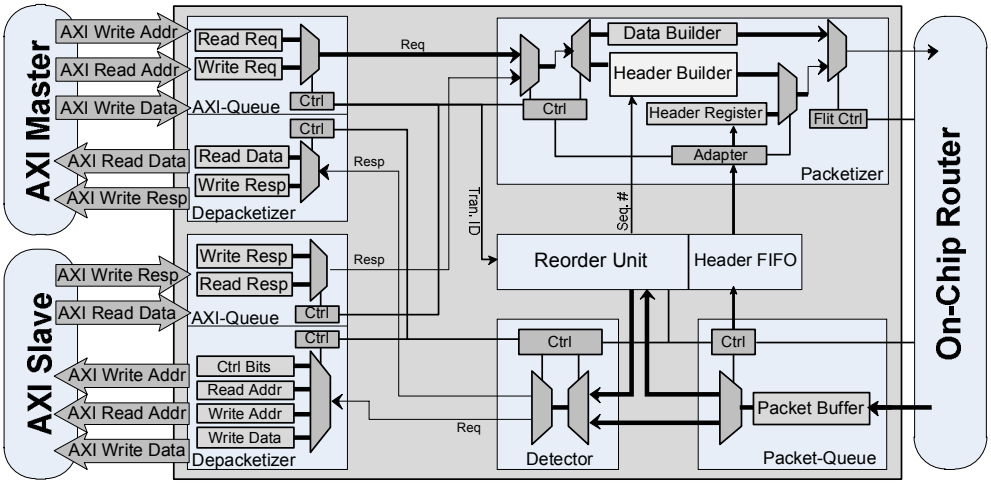


Fig. 5-7. Hybrid network interface architecture.

5.3.3 Hybrid Network Interface

The hybrid model is formed by combining the *master-side* and *slave-side* network interfaces. As illustrated in Fig. 5-7, based on the type of incoming packet (Req/Resp) the detector unit determines the target unit (Slave-side Queue/Master-side Queue). Regarding the MPSoC's configuration, if each node is supposed to integrate a dedicated processor and

memory, instead of using two network interfaces (master and slave), the hybrid model is more beneficial, particularly in terms of area and power costs. This architecture also prevents the local requests to enter the network such that local requests can access the local memory directly. A round-robin arbitration scheme is used between the local requests and global requests coming from the network.

5.4 Priority-based Router Architecture

In this part, we present a novel method for incorporating network resources in serializing the packets while they traverse inside the network. Fig. 5-8 shows a 4×4 tile architecture where the master core 0 accesses three memory modules 6, 13 and 15. Assume that the master core generates three requests, A, B and C, with a same transaction ID and sends request A to memory 15, request B to memory 13 and request C to memory 6. Due to the in-order requirement of the AXI protocol, response A needs to be delivered to the master core first and then responses B and C, respectively. For simplicity, we assume that the memory modules return responses with zero latency and we also assume that a round-robin arbiter is used in each router such that on average three cycles are needed for a packet to win arbitration in a router.

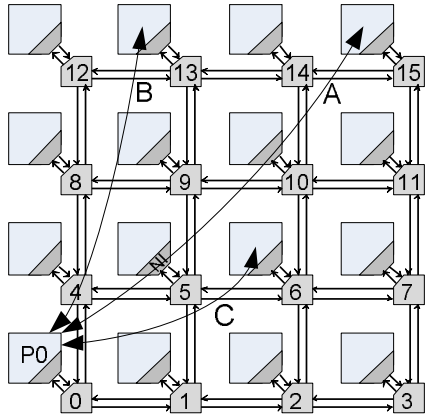
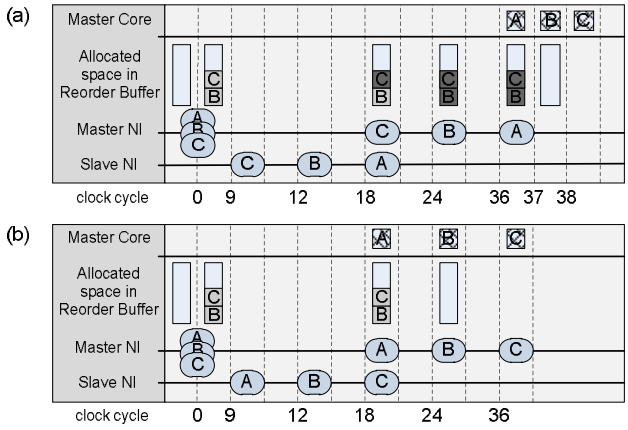


Fig. 5-8. 4×4 NoC where master core 0 sends requests A, B and C to memories 6, 13 and 15, respectively.

As shown in Fig. 5-9(a), the master network interface sends requests A, B and C at time 0 to the network. According to the proposed network interface architecture, buffer space should be reserved for requests B and C. By considering three cycles waiting time at each router, requests C and B access the memory modules 6 and 13, respectively, at cycles 9 and 12. At cycle 18, request A accesses the memory 15 and meanwhile the response C reaches the master network interface. Response C cannot be served by the master core before responses A and B, so it has to be stored in the reorder buffer. At cycle 24, response B is received by the master network interface and stored in the reorder buffer as it cannot be served earlier than response A. Response A is received by the master network interface at

time 36 and it can be sent directly to the master core. Finally, the stored responses B and C are released from the reorder buffer and delivered to the master core at cycles 37 and 38, respectively. However, when response B is delivered to the master core, the allocated buffer space for both requests B and C is released.

The idea behind our method is to give better chance to the long-distance packets with low ordering values to win arbitration in routers. By this approach, in the same example as Fig. 5-9(a) the number of waiting times of request A in arbitration phases is probably less than that of requests B and C (similarly the number of waiting cycles of request B is probably less than request C). The possibility of benefits from the idea of priority-based router can be found in the case of Fig. 5-9(b) in which the requests experience different waiting periods at routers and they are supposed to be 1.5, 3 and 6 cycles for requests A, B and C, respectively (these values are chosen such that the results could be compared with the example shown in Fig. 5-9(a)).



Req	Average cycles waiting in each router	hops between master & slave	arriving time to slave NI	arriving time to master NI
A	3 cycles	6 Hops	18 cycles	36 cycles
B	3 cycles	4 Hops	12 cycles	24 cycles
C	3 cycles	3 Hops	9 cycles	18 cycles

Req	Average cycles waiting in each router	hops between master & slave	arriving time to slave NI	arriving time to master NI
A	1.5 cycles	6 Hops	9 cycles	18 cycles
B	3 cycles	4 Hops	12 cycles	24 cycles
C	6 cycles	3 Hops	18 cycles	36 cycles

Fig. 5-9. Comparing (a) round-robin and (b) priority-based arbitration schemes in serializing the packets.

As illustrated in Fig. 5-9(b), requests A and B access their memories at cycles 9 and 12, respectively. At cycle 18, request C accesses the memory while response A is received by the master network interface. Since response A is arrived in-order, it can be served immediately and delivered to the master core. At cycle 24, response B can also be directly sent to the master core. Upon arrival of this response not only the required space for request B is released but also the reserved buffer space for request C is released. Finally, the response C reaches the master network interface at cycle 36 and it is delivered to the master core directly. According to the examples in Fig. 5-9(a) and Fig. 5-9(b), latencies can be considerably reduced by applying the idea of priority-based router, i.e. in Fig. 5-9(b) responses A, B and C are delivered to the master core at cycles 18, 24 and 36 which are earlier than in Fig. 5-9(a) where responses are served at cycles 36, 37 and 38, respectively. This reduction is mainly from the fact that responses arrive at the master network interface in-order, and thus can be served immediately. Another advantage of using priority-based

router is that the corresponding reserved buffer space in the reorder buffer can be released sooner as the responses are mainly reaching the master network interface in-order, i.e. in Fig. 5-9(b) the reserved buffer space for requests B and C are released at cycle 24 while in Fig. 5-9(a) they are freed at cycle 37. The idea of priority-based router can further improve the performance by allowing more pending requests to enter the network, thereby reducing the overall latencies of packets.

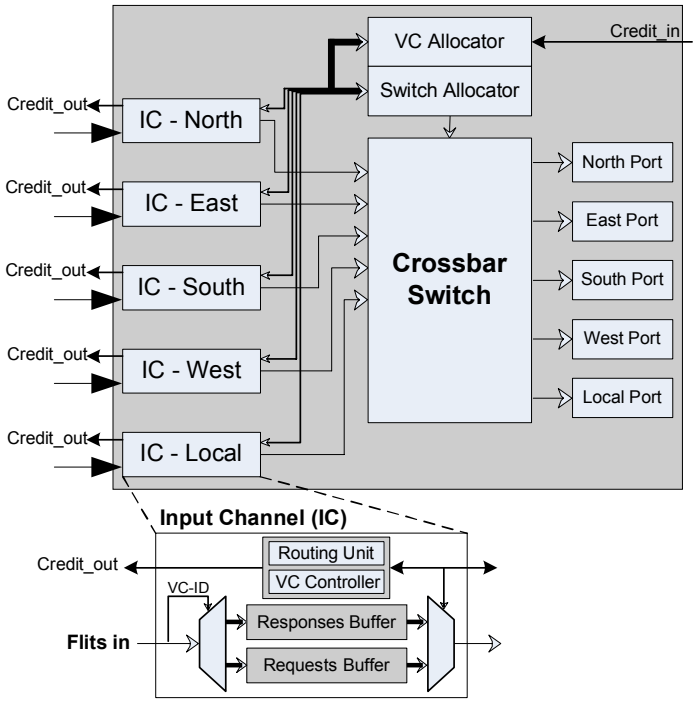


Fig. 5-10. The router architecture.

5.4.1 The Proposed Priority-based Router

The architecture of the priority-based router (PR), depicted in Fig. 5-10, has a typical state-of-the-art structure including input buffers, a VC (Virtual Channel) allocator, a routing unit, a switch allocator and a crossbar. Each router has 5 input/output ports, and each input port of the router has 2 VCs. Packets of different message types (request and response) are assigned to corresponding VCs to avoid message dependency deadlock [119]. The arbitration scheme of the switch allocator in the typical router (TR) structure is round-robin. The round-robin scheme is a fair policy when all packets have the same priority; otherwise priority-based methods are more beneficial. As already mentioned, each packet is assigned a sequence number and packets might be returned back out-of-order due to the different path length and different memory response time. The priority-based router assigns a

priority to each packet such that long-distance packets with low sequence numbers have better chance to win the arbitration in routers. Accordingly, the packet priority is computed by summing up two values. The first one is the distance the packet must traverse between the master and slave while the second one is obtained by subtracting the MaxSeqNum value (the maximum sequence number value that can be generated by the network interface) from the PacketSeqNum value (packet sequence number). The result is stored in the packet's header and used by router arbiters. By using the priority-based router, packets can proceed inside the network with different speeds according to their priority values.

```
   i    : i(th) input channel
   P(i) : priority value of i(th) input channel
          = (MaxSeqNum - PacketSeqNum + Distance)
   WP(i): waiting periods + P(i)
----------------
process Find_MaxPriority is
begin
  MaxValue <= 0;
  for 'i=0 to all Reqs in the output port' loop
   if Req is a new packet then
        WP(i) <= P(i);
   else
        WP(i) <= WP(i) + 1;
    end if;
    if WP(i) > MaxValue then
     MaxValue <= WP(i);
      select   <= i;
    end if;
  end loop;
end process;
```

Fig. 5-11. Pseudo VHDL code of the priority-based router.

Fig. 5-11 shows the algorithm in which the process, *Find_MaxPriority*, is activated when the output channel is available and there are multiple messages. It examines all messages and the priority value of the corresponding input packets and grants a message with the highest priority value. In order to prevent starvation, each time after finding the highest value, the priorities of defeated packets are incremented.

5.5 Order Sensitive Memory Scheduler

The architecture of the proposed memory controller, dubbed OS from Order Sensitive, is depicted in Fig. 5-12. As illustrated in the figure, the proposed memory controller is integrated in the slave-side network interface. After arriving to the network interface on the edge of the network, requests are stored in the respective queues based on their target banks. The data associated with write requests is stored in the write queue. The queues are implemented as the linked list structure which has been described earlier. Depending on the sequence number, received requests in each bank queue obtain a priority value to access the

memory. Here we have the same scenario as in the priority-based router. The priority value of each packet is based on the sequence number, which helps to deliver the packets to the master network interface in-order such that the corresponding reserved buffer space in the reorder buffer can be released sooner.

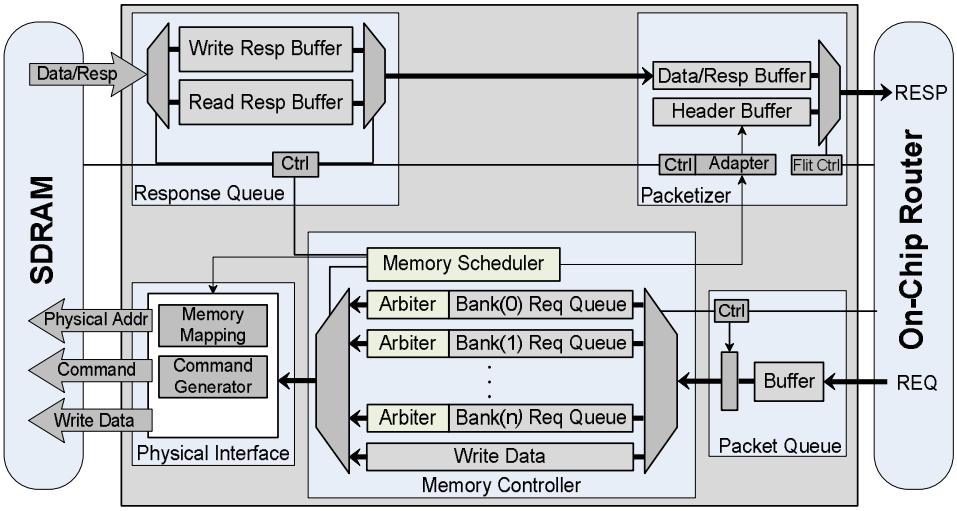


Fig. 5-12. The proposed memory controller integrated in the slave-side network interface.

Once a new request enters a queue, the process input_queue, shown in Fig. 5-13, updates the priority value of each request in the queue. The packet's sequence number of received request is assigned as a priority value for this request. In addition, to prevent starvation, the priority values of existing requests in the queue are incremented at every *input_queue* event. As mentioned earlier, each bank arbiter selects a request from the queue with the highest priority value based on the bank timing constraints as the first level of scheduling procedure. Since the *row-first* policy has better memory utilization in comparison with the other bank arbitration policies, the bank arbiters of the presented memory controller also takes advantage of the row-first policy. The bank arbitration policy in our memory controller is shown in Fig. 5-13. Whenever the *arbiter* process is activated, it tries to find a request which is a row hit and has a higher priority value. If there are not any row hits, the bank arbiter selects the highest priority request which is a row conflict from the queue and issues the SDRAM commands to service the selected request. Fig. 5-14 depicts the circuit of finding the suitable request in the request buffer which is described in the *arbiter* process of Fig. 5-13. In the second level of the scheduling procedure, at each memory cycle the memory scheduler decides which request from all bank arbiters should be issued. To simplify the hardware implementation and provide the *bank interleaving*, round-robin mechanism is utilized by the memory scheduler.

```
[1] RA(i) : row address of i(th) request.
[2]  CRA   : current row address issued prior.
[3] P(i)  : priority value of i(th) request
[4]         =(MaxSeqNum - PacketSeqNum)
[5] W(i)  : waiting periods + prioriy
[6]          of i(th) request in the queue.
[7] ----------------
[8] Process(input_queue)
[9] Begin
[10] For 'i:1 to number of Reqs in input queue' loop
[11]            If Req is a new packet then
[12]               W(i) <= P(i);
[13]     Else
[14]               W(i) <= W(i)+1;
[15]     End if;
[16]  End loop;
[17] End process;
[18] ----------------
[19] Process(arbiter)
[20] Begin
[21]  MaxValue1 <=0; select1 <=0;
[22]  MaxValue2 <=0; select2 <=0;
[23]           For 'i:1 to all requests in input queue' loop
[24]             If RA(i)= CRA then
[25]               If W(i)>= MaxValue1 then
[26]                 select1 <= i;
[27]                 MaxValue1     <= W(i);
[28]               End if;
[29]             Else
[30]               If W(i)>= MaxValue2 then
[31]                 select2 <= i;
[32]                 MaxValue2     <= W(i);
[33]               End if;
[34]     End if;
[35]           End loop;
[36]           If select1 /= 0 then
[37]              select <= select1;
[38]           Else
[39]     select <= select2;
[40] End if;
[41]End process;
```

Fig. 5-13. Pseudo VHDL code of the arbiter in the memory controller.

5.6 Experimental Results

In this section, we evaluate the proposed on-chip network architecture in terms of average network latency, memory latency, and memory utilization compared with the baseline architecture under different traffic patterns. Also, we discuss the area and power

consumption of the proposed NoC components: network interface, priority-based router, and order sensitive memory scheduler. Consequently, a 2D NoC simulator is implemented with VHDL to model all major components of the NoC.

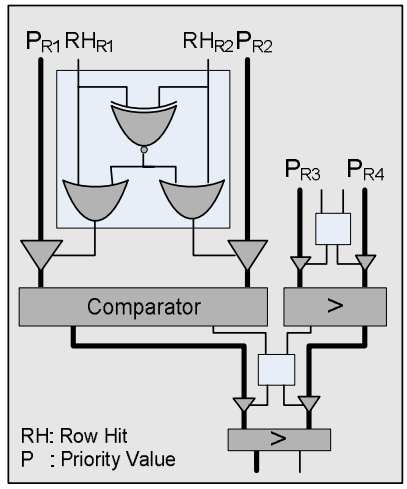


Fig. 5-14. Request selector circuit.

5.6.1 System Configuration

We use a 25-node (5×5) 2D mesh on-chip network with two different configurations for the entire architecture. In the first configuration (A), illustrated in Fig. 5-15, out of 25 nodes, 10 nodes are assumed to be processors (master cores, connected by master NIs) and remaining 15 nodes are memories (slave cores, connected by slave NIs). For the second configuration (B), each node is considered to have a processor and a memory (master and slave cores, connected by a hybrid NI). The processors are 32b-AXI and the memories specified in subsection 5.1, are DDR2-256MB (t_{RP}-t_{RCD}-t_{CL}=2-2-2, 32b, 4 banks) [116]. We assume that the memories are integrated on a separate die which is stacked on top of the processor layer [35][120][121]. Inasmuch as the memories are now stacked on top of the processors layer, the front-side bus and memory controller operate at the same speed as the processors. The timing of each stacked DRAM module is still the same as in a traditional DRAM memory (t_{CAS}, t_{RAS}, etc. are unchanged) [35][120][121]. We adopt a commercial memory controller with memory interface, DDR2SPA module from Gaisler ip-cores [122]. Along with the proposed order sensitive (OS) memory scheduler, another memory scheduler with row-first (RF) policy is also implemented as the default scheduler for the memory controller. The network of each configuration that has been considered for experimental results is formed either by Typical Router (TR) or by Priority-based Routers (PR).

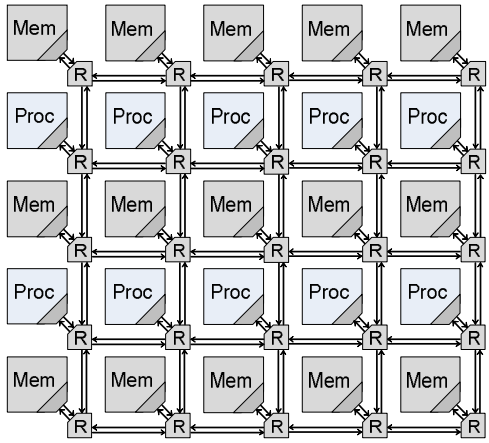


Fig. 5-15. The layout of the system configuration A.

The array size, routing algorithm, link width, number of VCs, buffer depth of each VC, and traffic type are the other parameters which must be specified for the simulator. The routers adopt the XY routing algorithm [64][60] and utilize wormhole switching. For all routers, the data width (flit size) is set to 32 bits, and the buffer depth of each VC to 5 flits. Message structures for the AXI protocol are defined in [109][115]. For the request, the command and all its control bits (flags) are included in the first flit of the packet, the memory address is set in the second flit, and the write data (in the case of a write command) is appended at the end. For the response message, the control bits are included in the first flit while the read data is appended at the end if the response relates to a read request. Hence, the packet length for write responses and read requests is 1 flit and 2 flits, respectively, while the packet length for data messages, representative of read responses and write requests, is variable and depends on the write request/read response length (burst size) produced by a master/slave core. As a performance metric, we use latency defined as the number of cycles between the initiation of a request operation issued by a master (processor) and the time when the response is completely delivered to the master from the slave (memory). The request rate is defined as the ratio of the successful read/write request injections into the network interface over the total number of injection attempts. All the cores and routers are assumed to operate at 1GHz; and for fair comparison, we keep the bisection bandwidth constant in all configurations. All memories (slave cores) can be accessed simultaneously by each master core continuously generating memory requests. Furthermore, the size of each queue (and FIFO) in the network is set to 8×32 bits and the size of the reorder buffer is set to 48 words. If the maximum burst size is set to 8, the baseline architecture utilizing a statically partitioned reorder buffer [107] can support at most 6 outstanding read requests in a 48-word reorder buffer (regardless of the exact size of the requests), while the proposed approach is able to embed as many requests as can be reserved in the reorder buffer, i.e. at most 48 and at least 6 outstanding read requests.

5.6.2 Performance Evaluation

To evaluate the performance of the proposed schemes, uniform and non-uniform/localized synthetic traffic patterns are considered separately for both configurations (A and B). These workloads provide insight into the strengths and weaknesses of the different buffer management mechanisms in the interconnection networks, and we expect applications stand between these two synthetic traffic patterns [123][124]. The random traffic represents the most generic case, where each processor sends in-order read/write requests to memories with a uniform probability. Hence, the target memory and request type (read or write) are selected randomly. Eight burst sizes, from 1 to 8, are stochastically chosen according to the data length of the request. In the non-uniform mode, 70% of the traffic is local requests, where the destination memory is one hop away from the master core, and the rest 30% of the traffic is uniformly distributed to the non-local memory modules. We also consider the hotspot traffic pattern where four memory nodes are chosen as hotspots receiving an extra portion of the traffic (10%) in addition to the regular uniform traffic [64][60]. For the uniform and hotspot traffic profiles, we obtained very similar performance gains in each configuration so that they are not presented.

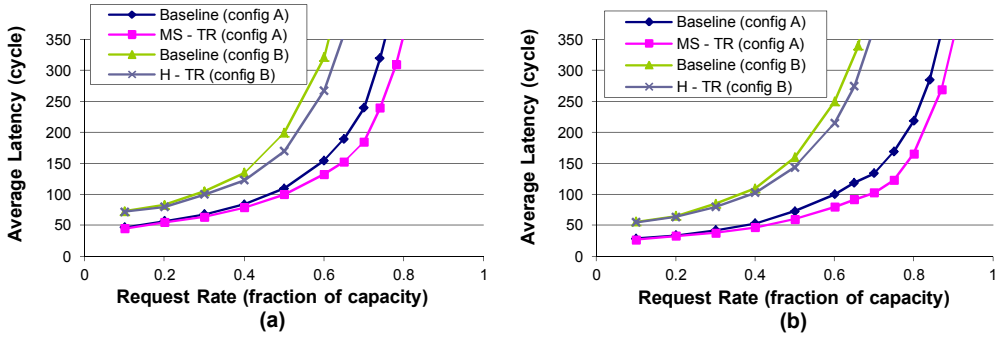


Fig. 5-16. Performance evaluation of both configurations under (a) uniform and (b) non-uniform traffic models.

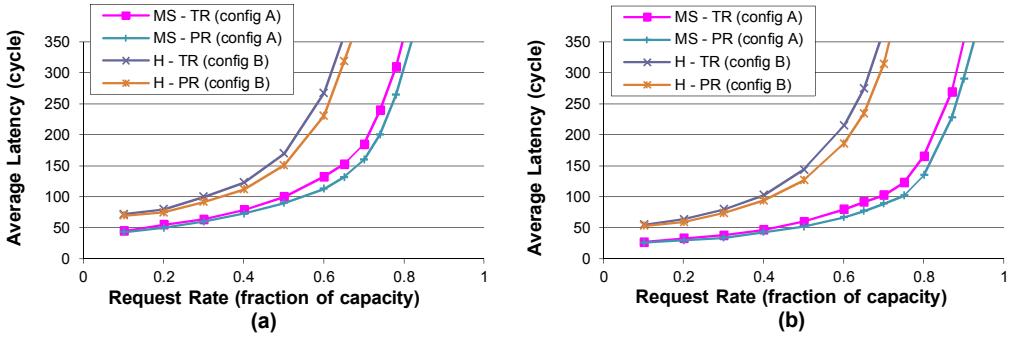


Fig. 5-17. Performance impact of using the priority-based router under the (a) uniform and (b) non-uniform traffic models.

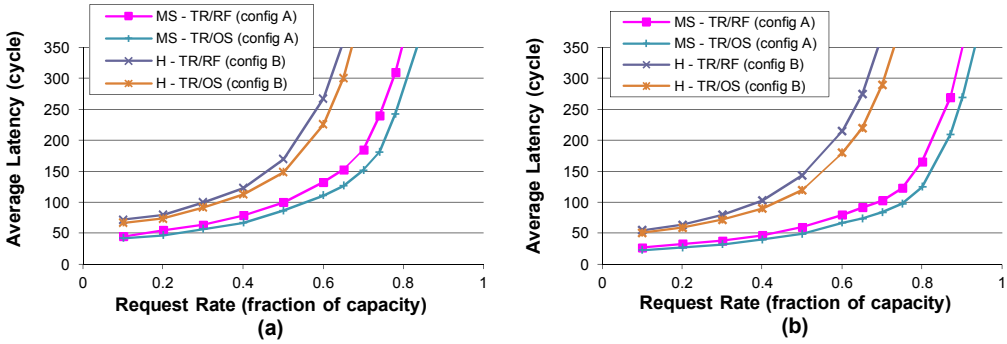


Fig. 5-18. Performance impact of using the order sensitive memory controller under the (a) uniform and (b) non-uniform traffic models.

Fig. 5-16(a) and (b) show the simulation results under the uniform and non-uniform traffic models, respectively. In each configuration, the on-chip network utilizing the proposed network interface, denoted by MS (Master/Slave NI) and H (Hybrid NI), is compared with the network equipped with the baseline network interface. As demonstrated in both figures, compared with the baseline architecture, the NoC using the proposed network interface reduces the average latency when the request rate increases under the uniform and non-uniform traffic models. The foremost reason for such an improvement is due to employing the shared reorder buffer in the network interface which allows more messages to enter the network, i.e. this leads more requests to be released from the injection queue.

Regarding the configuration B, using the master-side and slave-side network interfaces instead of the hybrid network interface when each node is composed of a dedicated processor and memory gives a better performance but as discussed in the next subsection it is not a cost efficient approach. The Hybrid structure deteriorates the performance because the buffer resources are shared. However, the performance given by the hybrid structure close to the saturation point is around 34% and 18% less than the other structure under the uniform and non-uniform traffic models, respectively. The performance penalty under the non-uniform traffic model is not significant compared to the uniform traffic model because the hybrid network interface allows the local requests to access the local memory directly.

Fig. 5-17(a) and (b) depicts the performance gain of the presented priority-based router architecture under uniform and non-uniform traffic models, respectively. In each configuration the network formed by the priority-based routers (PR) reduces the average latency as compared with the network formed by typical routers (TR). The performance gain near the saturation point (0.6) under the uniform traffic model for configuration A and B is about 15% and 13%, respectively, while the hardware overhead of this router is less than 2% in comparison with the typical router. This reveals that giving priority to packets according to their sequence number and remaining distance helps to deliver the packets to the master network interface in-order. Therefore, the corresponding reserved buffer space in the reorder buffer can be released sooner and master cores can receive responses earlier than using the typical routers.

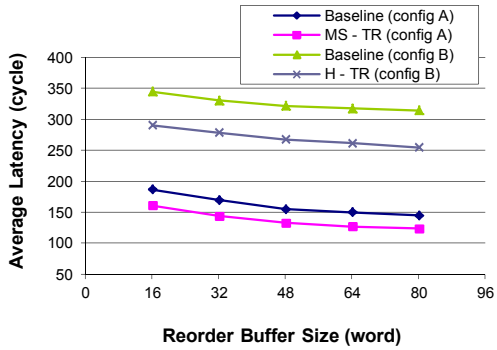


Fig. 5-19. Effect of reorder buffer size on the performance under the uniform traffic model.

To explore the impact of the proposed Order Sensitive (OS) memory scheduler, we compare the network equipped with the OS scheduler with the one using the default scheduler (Row-First (RF)) where both networks are formed by the typical routers. Fig. 5-18(a) and (b) present the performance comparison between the two networks for each configuration under the uniform and non-uniform traffic models. Compared with the RF scheme, the network utilizing OS scheduler gives significant improvements in average network latency. The performance gain of the OS scheduler close to the saturation rate under the uniform traffic model for configuration A and B is up to 17% and 16%, respectively. The average memory utilization and average memory latency are also computed near the saturation rate under the uniform traffic profile for the configuration A. According to our observation, at least half of the request buffers of each memory controller are occupied under the uniform traffic model with the given injection rate, which keeps the memory controller busy all the time. As a result, compared with the RF scheme, the average utilization of memories is improved by 22% while the average memory latency is reduced by 19%. Compared with RF, the hardware overhead of the OS scheme is negligible since both of them have similar request and data queues, buffer management, and bank interleaving arbiter. In addition, near the saturation rate under the uniform traffic profile for the configuration A the number of average and maximum buffer occupancy of reorder buffers is around 70% and above 90%, respectively. The number of average and maximum outstanding messages in the system is around 160 and 230, respectively.

We also vary the reorder buffer size to show how relative reorder buffer size affects the performance. Fig. 5-19 illustrates the average network latency of both configurations near the saturation point (0.6) under the uniform traffic profile. It reveals that as the reorder buffer size increases, the average network latency reduces. Given the same reorder buffer size, the proposed network interface achieves better performance gain, e.g. when the reorder buffer size is 48, the performance gain for the configuration A and B is up to 16% and 21%, respectively. The proposed scheme not only achieves significant performance gain but also enables reducing the area overhead of reorder buffer by more than 60%. For instance, the proposed scheme in the configuration A with a reorder buffer size of 32 offers a better performance than a reorder buffer size of 80 in the baseline method.

Table 5-1. Hardware implementation details.

Components	Area (mm^2)	Power (mW)
Slave-side	0.0428	17
Master-side	0.0755	26
Hybrid	0.1014	37
Memory controller (OS)	0.0807	31
Memory controller (RF)	0.0798	27
Typical Router	0.1853	65
Priority-based Router	0.1881	68

5.6.3 Hardware Overhead

For appraising the area overhead of the proposed architectures, each scheme is synthesized by Synopsys D.C. using the UMC 90nm technology with an operating point of 1GHz and supply voltage 1V. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation of each scheme, including both dynamic and leakage power, is also calculated near the saturation point (0.6) under the uniform traffic model using Synopsys PrimePower. In addition to the aforementioned configuration of the network interface, the T-ID and SN are set to 4-bit and 3-bit, respectively. The layout areas and power consumptions of the master-side, slave-side, hybrid interfaces, different memory controller and routers are listed in Table 5-1. As can be seen from the table, using the hybrid architecture for the latter configuration (B) is more beneficial (in terms of power and area) than using the master-side and slave-side models when each node is composed of a dedicated processor and memory. That is, using a hybrid NI model reduces 14.3% and 13.8% in hardware area and power dissipation respectively. Because all queues (and FIFOs) are equal in size, they do not affect the comparison. On the other hand, the master-side and slave-side network interface architectures are more cost efficient if each node consists of a dedicated processor or memory as in the former configuration (A). Also, comparing the area cost of the baseline model to each proposed network interface indicates that the hardware overheads of implementing the proposed network interface schemes are less than 0.5%. Furthermore, for the slave-side interface within the memory controller, since each of the memory modules utilized in this thesis has 4 banks, 4 bank queues have been implemented in the memory controller.

5.7 Summary

In this chapter, we presented a high performance network interface with a novel dynamic buffer allocation and a priority-based router model to improve the resource utilization, and overall on-chip network performance. In addition to the resource utilization of the network interface and on-chip network, the utilization of memories considerably affects the network

latency. Therefore, we have developed an optimized scheduling method for the DRAM memories and integrated it in the network interface such that the network and memory latencies were reduced significantly in comparison with the baseline architecture. The micro-architectures of the proposed network interfaces which are compatible with the AMBA AXI protocol have been presented. A cycle-accurate simulator was used to evaluate the efficiency of the proposed architecture. Under both uniform and non-uniform traffic models, in high traffic load, the proposed network interface architecture has lower average delay in comparison with the baseline architecture.

Chapter 6

Three-Dimensional Networks-on-Chip

As mentioned earlier, two-dimensional (2D) chip fabrication technology is facing several challenges in the deep submicron regime such as designing the clock-tree network for a large chip, limited floor-planning choices, increasing the wire delay and power consumption, integrating diversity components that are digital, analog, MEMS RF, etc.
The Three Dimensional (3D) integration has emerged as a potential solution to address these problems and the design complexity of MPSoC in 2D Integrated Circuits (IC). 3D ICs reduce the interconnect delay by stacking vertically active silicon layers as well as offering a number of advantages over the traditional 2D chip [19][20][21][22]: (1) shorter global interconnects; (2) higher performance; (3) lower interconnect power consumption due to wire-length reduction; (4) higher packing density and smaller footprint; and (5) support for the implementation of mixed-technology chips. In this chapter we focused on wafer stacking technology. In wafer-to-wafer bonding technology, one of the popular options for 3D integrations, dies are vertically stacked. Short, fat, and vertical Through-Silicon-Vias (TSVs) are exploited for inter-layer communication. The distance between wafers can range from 5μm to 50μm [22][24], which is much shorter than the wire length between cores on a tier, and the pitches of TSVs can range from 1μm to 10μm square [22][24]. That is, the wire delay, power consumption and chip form factor are significantly reduced [25][26][28].
3D ICs have emerged as a viable candidate to achieve better performance and packaging density as compared to traditional two dimensional (2D) ICs. 3D NoC topologies not only enable scalable networks to provide communication requirements in 3D ICs [19]-[23] but also are a crucial factor of 3D chips in terms of performance, cost, and energy consumption [19]. Various on-chip network topologies have been studied for 3D NoCs [19]-[23][25][27][29]. Mesh-based structures are popularly used in 3D systems, because their grid-based regular architecture is intuitively considered to be matched to the 2D VLSI layout for each stack layer [19]-[23]. Nevertheless, if the number of IP-cores and memories increases in each layer, more TSVs are necessitated to handle the inter-layer communication. Inasmuch as each TSV employs a pad for bonding, the area footprint of TSVs in each layer is augmented significantly [22][29]. The main contributions of this chapter are twofold.

First, in order to improve performance of vertical channels, a novel pipeline bus structure is introduced. The proposed bus structure overcomes the drawbacks of previously presented bus structures for vertical channels and improves the performance by reducing the delay and complexity of traditional bus arbitration which is the foremost impediment in bus communications. In addition, the presented pipeline bus structure can utilize bi-synchronous FIFO for synchronization between stacked layers, if each layer is fabricated by different technologies.
Second, we present two novel stacked mesh topologies to reduce the area overhead of TSVs and power dissipation with a small performance penalty. The proposed stacked mesh topologies, named Clustered Mesh Inter-layer Topology (CMIT) and Concentrated Inter-layer Topology (CIT), benefit of clustering the mesh topology for each layer. Each cluster of the presented topologies has its dedicated vertical channel, composed of a set of TSVs. CMIT and CIT preserve the advantages of the clustered mesh topology and mitigates both power density and TSV area footprint on each layer.

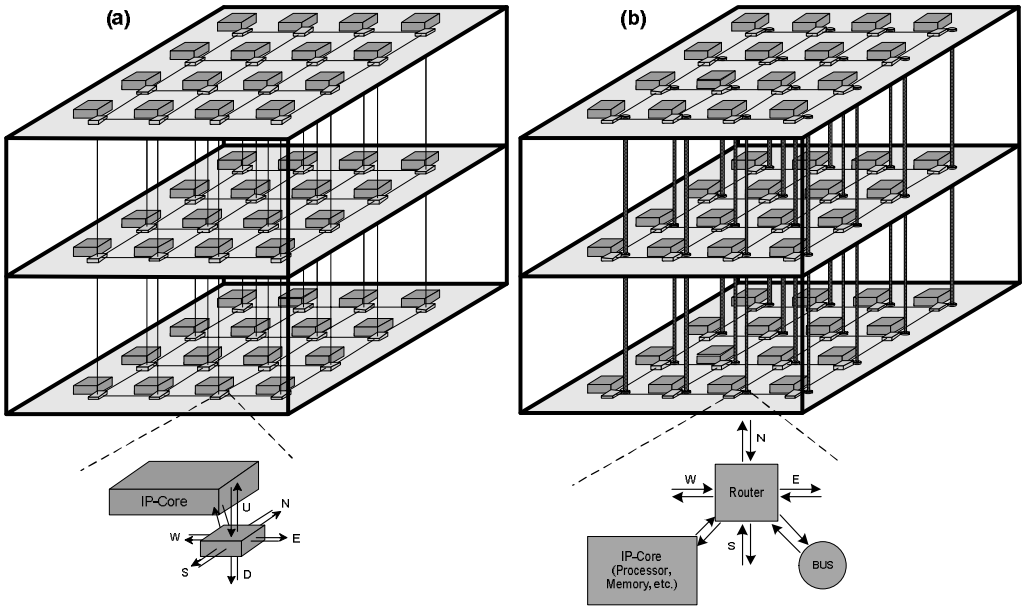


Fig. 6-1. Mesh-based NoC architectures: (a) 3D-symmetric NoC (b) 3D NoC-Bus Hybrid structures.

6.1 3D NoC Architecture

3D-Symmetric NoC and 3D NoC-Bus Hybrid (stacked mesh) structures are popularly used in 3D systems, because their grid-based regular structure is intuitively considered to match the 2D VLSI layout for each layer [19]-[23][25][27][29]. The 3D-symmetric NoC structure,

shown in Fig. 6-1(a), is an extension of 2D mesh by adding two additional physical ports to each baseline-router (one for up and one for down) in the popular 2D mesh-based system [19][22]. Adding two additional ports requires larger crossbar incurring significant area and power overhead and increases the blocking probability occurring inside the router. Since TSVs are shorter and wider than intra-layer interconnects, they have lower resistance and can support higher signaling speeds [19][22]. As router latencies may dominate the fast vertical interconnects, this has led the researchers to propose 3D NoC-Bus Hybrid structures using a bus with a centralized arbiter for each vertical channel, which allows single hop latency for packets between any layers [19]-[22]. As depicted in Fig. 6-1(b), on-chip routers in this structure have at most 6 ports, one to the IP-core, one to the bus, and four for cardinal directions. According to [19] the 3D-hybrid structure was observed to be better than the 3D symmetric for the vertical interconnection as long as the number of device layers was less than 9. This has motivated us to present an efficient pipeline bus to overcome drawbacks of the conventional bus that has been employed for inter-layer communication.

6.2 Constraint on the number of TSVs

A relatively high area penalty due to via blockage may impose limitations on the number of TSVs that can be utilized for inter-layer communication due to the following reasons. First, the move from 2D to 3D architecture could accentuate the thermal concerns due to the increased power densities resulting from placing one logic block over another in the multilayered 3D stack. An efficient solution for cooling 3D ICs is to employ either thermal TSVs [29][125], establishing a thermal path from the core of a chip to the heat sink or liquid cooling based on fluidic TSVs [125]. This can take at least 10-20% of total chip area to create thermal or liquid -efficient 3D ICs [29][125][126][127]. Second, conventionally in 3D ICs the input clock signal, at the center of the clock tree, is fed to each layer via TSV and each layer has its own clock tree with associated clock buffers implemented in the corresponding active layer. However, the obvious disadvantage of this scheme is the design overhead, both in terms of resources and design efforts required for the layer customization. Moreover, because of the separate customization of the different layers, the skew between terminals in different layers may be high even if the skew is low in the same layer. The alternative scheme, named via topology, implements the clock tree with the clock buffers on a single layer and using TSVs the clock signals from the terminals of the clock tree are passed to all other layers [128]. This scheme provides uniform skew compensation across layers since the same terminal clock signals are transmitted across layers with less design overhead, i.e. approximately N times less area and power than the conventional scheme, where N is the number of layers in the 3D clock tree. The only shortcoming of this scheme is the high number of TSVs required for passing the clock signals from terminals to all other layers. Third, if we consider signal TSVs and power/ground TSVs separately, since each signal TSV uses the minimal allowable TSV size and microprocessors typically require a few hundred I/O signals, signal TSVs occupy a very small area on stacked dies.

On the other hand, microprocessors dies typically need tens or few hundreds of amperes current, which causes power consumption overhead for power TSVs due to the resistance of TSVs. Thus, we have to increase the aggregate size of those power TSVs so that the area overhead induced through power TSV network is considerably high [129]. However, not only the area overhead of TSVs is quite high, but also floor planning and routing is extremely challenging since TSVs are distributed in each layer. In this chapter, we present two area-efficient stacked architectures to reduce the TSV footprint with a small performance overhead though the presented bus architecture diminished the TSVs overhead in compared to conventional buses.

6.3 Related Work

Design techniques and methodologies for 3D architectures have been investigated to efficiently exploit the benefits of 3D technologies. Several NoC topologies for 3D systems have been exhaustively investigated in [19]-[22][27][133]. The authors in [19] demonstrate that besides reducing the footprint in a fabricated design, 3D systems provide a better performance compared to traditional 2D systems. They have also demonstrated that both mesh and tree topologies for 3D systems achieve better performance compared to traditional 2D systems. However, the mesh topology shows significant performance gains in terms of throughput, average latency, and energy dissipation with a small area overhead [19]. In [133] different 3D mesh-based architectures have been compared in the zero-load latency, but the performance of the network with different traffic patterns and loads is also necessary to be evaluated.

To construct an optimistic 3D mesh-based system, several 3D structures have been presented. Baseline-routers in 2D mesh-based systems have 5 ports, i.e. 4 ports to adjacent routers and one for the resource node. The straightforward extension for 3D mesh-based systems (*3D-symmetric NoC*) is to utilize routers with two additional inter-layer links by adding two physical ports to baseline-routers (one for up and one for down) [19][22][25][27][137]. As mentioned earlier, the 3D structure using such routers, not only increases the area and power overhead of the routers but also contention in the routers may arise. The electrical behavior of the relatively short and wide TSV, i.e. the low resistance, and supporting much higher signaling speeds led the authors of [22] to propose the 3D-hybrid structure. This 3D structure exploits the Dynamic Time Division Multiple Access (dTDMA) bus [132] with a centralized arbiter for the vertical communication link. Thus, moving from one layer to any of the other layers takes only one hop. However, contention issues in the bus limit the attainable performance gains [19]. That is, such structures inherently suffer from the limitation of buses since only one transmission is allowed each time over a vertical bus.

In [21], the DimDe router for 3D architectures has been proposed. The presented router uses a full 3D crossbar and a simple bus structure spanning all layers of the chip and fusing them into a single router entity. This router can minimize vertical traversal to one hop

between any layers, but requires huge number of vertical connections and significantly complicates the control and arbiter of the router.
A multilayered 3D router architecture, named MIRA, is introduced for 3D systems by D. Park et al [20]. The router components are classified as separable components (buffers, crossbar, and inter-router links) and non-separable components (arbiter and routing modules). The separable components are laid out across multiple layers to save chip area and reduce power by dynamically shutting down some inactive layers. However, such routers are too aggressive in the current technology [138].
To reduce the area footprint of TSVs, a serialization scheme for vertical channels has been presented in [29], but this scheme is only applicable with 3D-hybrid structures where each node has a dedicated vertical channel.
Due to the above concerns, in this chapter, we have focused on both the 3D-symmetric structure (7-port switch design) and the 3D-hybrid structure (bus-based vertical interconnect). As described in [27], the 3D-hybrid structure is shown to perform the worst compared to the other structures in terms of scalability under local traffic. Although shown to be weak in [27][138], the bus may be appropriated for hotspot traffic injection where many packets may need to be sent through several layers to a hotspot frequently. This may be akin to a processor on one layer, and a memory stack directly above it. Hence, in 3D architectures, the 3D-hybrid structure performance degrades as the number of layers and number of processing nodes increase [27], thereby the 3D-symmetric structure is more feasible, mature, and more efficient than the 3D-hybrid structure as network size increases [139]. However, our proposed stacked architectures are applicable for both 3D-hybrid and 3D-symmetric structures where a group of nodes can share a vertical channel as an inter-layer interconnection. We also introduce a novel bus architecture which is more efficient than the conventional buses utilized for inter-layer communications in terms of scalability and performance.

6.4 Pipeline bus Architecture

Traditionally, a bus is described as a shared link which can be owned by one attached subsystem at a time. Parallelism can be added to the structure by partitioning the bus into segments with bridges and allowing these segments to operate concurrently [130]. However, on one side, the overall system performance in such designs is still limited by the lack of parallel bus transactions, and on the other hand, because of using many control wires for the central arbitration in such segmented buses, it is not a suitable approach for vertical bus in 3D ICs. Our solution for these bottlenecks for vertical buses is to consider the system bus with a bidirectional pipeline which is capable of transferring data concurrently from one or more sources to several destinations. As the proposed architecture is illustrated in Fig. 6-1(a), the system is partitioned into a set of modules each of which is used to connect the corresponding layer to the pipeline bus. As the system is based on Globally Asynchronous Locally Synchronous (GALS) design paradigm, the layers can internally operate at different clock frequencies. The layers are independent of each other,

in case there are some inter-layer transactions, the layers exchange data synchronously or asynchronously through the pipelined system bus, a segmented communication link which allows simultaneous transfer in both directions. The layers can concurrently access the bus without waiting for any grant signals, because of the pipelined structure of the proposed bus architecture. The interface module acts as a synchronizer between the router and the pipeline bus. To construct the pipelined bus, the physical wires that implement the bus are divided into a set of segments separated from each other by Transfer Stages (TS), one attached to each layer (Fig. 6-1(a)).

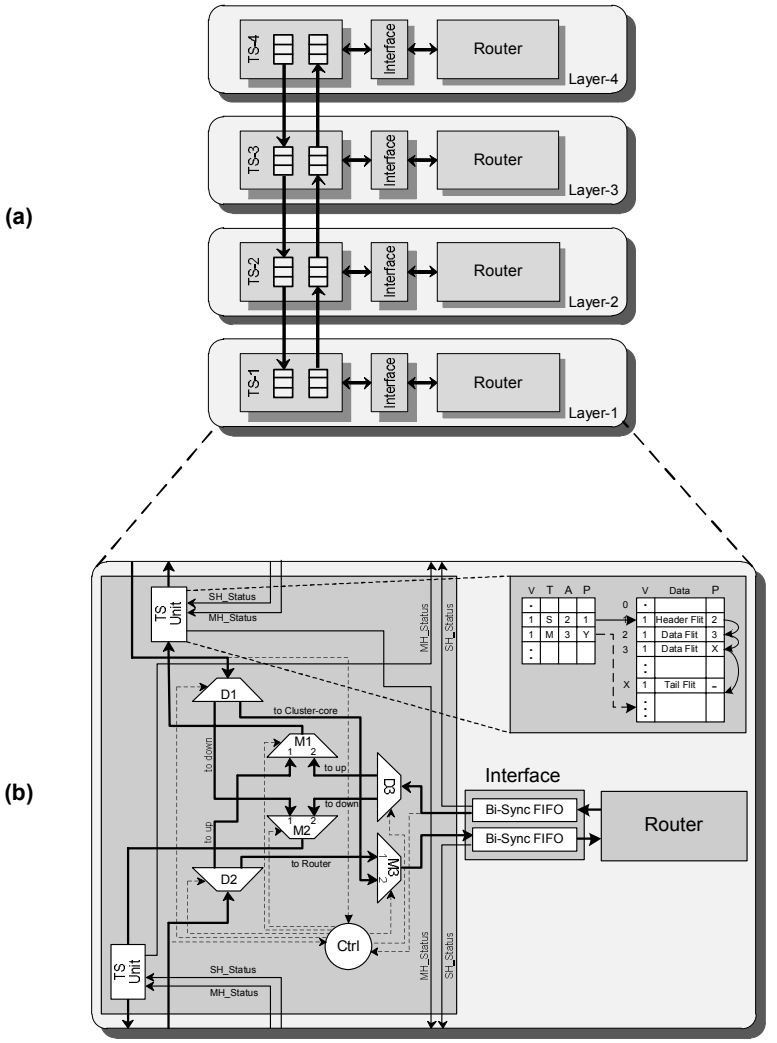


Fig. 6-2. (a) Proposed bus architecture and (b) the micro-architecture of the transfer stage.

Each transfer stage contains internal FIFO queues for pipelining the data flow, and a bus segment between adjacent stages consists of two separate unidirectional point-to-point interconnects which transfer data synchronously (or asynchronously) between the stages in opposite directions. These two links of a segment can operate in parallel, and due to pipelining, all segments of the bus can transfer data simultaneously. Each layer has a unique address for inter-layer communication. Furthermore, each IP-core/memory in a layer has its own address which makes addressing of a specific module in a given layer possible. Hence, a datagram propagating along the bus has a header containing both the layer address and the IP-core/memory address. The former is analyzed at each transfer stage, and the latter is decoded by routers in each layer.

6.4.1 Transfer Stage Micro-Architecture

The micro-architecture of the transfer stage is illustrated in Fig. 6-1(b) where it includes two identical pipelines transferring data to the opposite directions. Each pipeline contains multiple slots to pipeline packets between slots. Apart from the pipelines the interface contains FIFO queues used as input and output buffers of the host port. Their capacity has to be chosen according to the speed of the bus interface and the estimated data rate of the attached router. Each transfer stage also contains three multiplexers (M1, M2 and M3) and three de-multiplexers (D1, D2 and D3) to establish a communication between inputs and outputs of the transfer stage. On top of that, each transfer stage has the following functions:
1- It forwards incoming packets from the preceding stage to the next stage through a buffer, in both directions. That is, if the incoming packet from the upper stage (lower stage) is intended to be forwarded to the lower stage (upper stage), D1 and M2 (D2 and M1) will provide the required connections.
2- If an incoming packet from an adjacent transfer stage is intended to be processed by the router, the transfer stage delivers the packet to the interface module of the layer through a FIFO queue, i.e. D1 and M3 (D2 and M3) establishes the required connections to deliver the incoming packet from the upper stage (lower stage) to the router.
3- When a data is sent to another layer, the transfer stage operates as an output buffer. This means that it takes care of first receiving data from the interface module of the attached layer through a FIFO queue and then sending this data to one of the two adjacent transfer stages, depending on the direction in which, the target layer is located. Namely, D3 and M1 (D3 and M2) will be responsible for the required connections when the router decides to send a packet to upper layer (lower layer). When a packet arrives at a transfer stage, the header flit is sent to the controller unit to determine in which direction the packet should be sent. Based on the controller decision, it will be either forwarded to the next stage or transferred to the host router via the interface. Also, an arbitration in the controller module has to be performed to prevent the two parallel operating pipelines from writing simultaneously to the FIFO in the interface. In addition, because the electrical behavior of short and wide TSVs provides much higher signaling speeds, the credit-based flow control [2] has been implemented for the transmission protocol on a segment between transfer stages.

```
-- Multiplexers M1
w1= number of lower layers
w2= 1 (connected to one layer)
Process
Begin
    If input1='1' and counter ≤ w1 then
        service <= input1;
        counter <= counter + 1;
    Elsif input2='1' then
        service <= input2;
        counter <= 1;
    Else
        service <= input1;
        counter <= 1;
    End If;
End;
```

```
-- Multiplexers M2
w1= number of upper layers
w2= 1 (connected to one layer)
Process
Begin
    If input1='1' and counter ≤ w1 then
        service <= input1;
        counter <= counter + 1;
    Elsif input2='1' then
        service <= input2;
        counter <= 1;
    Else
        service <= input1;
        counter <= 1;
    End If;
End;
```

```
-- Multiplexers M3
w1= number of lower layers
w2= number of upper layers
Process
Begin
    If input1='1' and counter ≤ w1 then
        service <= input1;
        counter <= counter + 1;
    Elsif input2='1' then
        service <= input2;
        If counter =w1+w2 then
            counter <= 1;
        Else
            counter <= counter+1;
        End If;
    Else
        service <= input1;
        counter <= 1;
    End If;
End;
```

Fig. 6-3. Pseudo VHDL code of the weight-based arbitration for multiplexers: M1, M2, and M3.

6.4.2 Weight-based Arbitration

Arbiters in the controller may use a round-robin policy to arbitrate between the inputs of the multiplexers. However by using round-robin arbitration policy in the transfer stage of the pipeline bus architecture, fairness can become a problem. Fairness is not an issue when the traffic load is low, but as the traffic load approaches saturation, fairness can become a bottleneck. Let us consider an example to illustrate the unfairness problem where three

layers 1, 2 and 3 have large amounts of packets to be sent to the layer 4. Contention might occur in layers 2 and 3 as the packets must share the same channel resources. For instance, in the layer 3 there are two flows of packets competing for the bandwidth of the output. One flow is comprised of the packets being generated in the layer 3, and the packets in the other flow are arrived from the layer 1 and layer 2.
In other words, the layer closest to the destination layer, layer 3, will get the most bandwidth, 1/2 of the available bandwidth. The remaining half of the bandwidth is allocated to the layers 1 and 2, so each can receive 1/4 of the total bandwidth. Therefore, the allocation of the available bandwidth to the competing flows is not fair if round-robin policy is used. To overcome this limitation, a weight-based arbitration is employed so that the weight of each input port is determined by the number of upstream layers connected to that input port through the pipeline bus. In the above example at the layer 3, the weight of one input port is two as it can accept packets from two lower layers (*w1=2*), and the weight of the other input port is one since it can receive data only from the current layer (*w2=1*). As a result, the arbiter transmits the maximum of *w1* packets (if any) from the first flow and then allows the other flow to forward up to *w2* packets (if any) to the output and the process is repeated for the rest of the packets. By this approach, in each layer the total bandwidth is equally shared among packets from different layers. This simple weighted round robin arbitration achieves a fair forwarding policy with a very low hardware overhead. Fig. 6-3 is the pseudo code of the weight-based arbitration for multiplexers M1, M2 and M3.

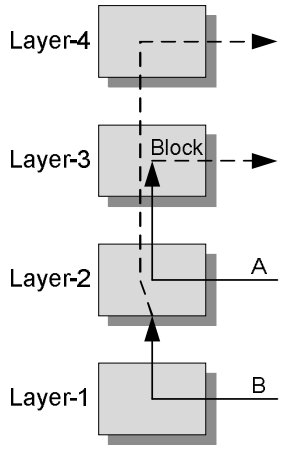


Fig. 6-4. A blocking situation.

6.4.3 Non-blocking Scheme

The proposed pipeline structure allows simultaneous transmissions without using centralized bus arbitration, which considerably reduces arbitration complexity and improves bandwidth. However in the pipeline bus architecture, a single blocked packet

might obstruct the subsequent packets so that increasing the communication latency. A blocking situation is shown in Fig. 6-4 where the packet A and packet B destined for the layer 3 and layer 4, respectively. When the packet A is blocked, the packet B can be obstructed behind the packet A. In order to prevent this blocking situation, we introduce two types of packet, Single-Hop (SH) packet and Multiple-Hop (MH) packet. SH packet destinations are located in one of the neighboring layers while MH packets require passing several layers. However, a MH packet changes its type to SH once the destination is one layer away.

In each transfer stage, the incoming packets are de-multiplexed into two separate paths: one path delivers the SH packets to the interface (SH path) while the other path forwards the MH packets to the next stage (MH path). That is, packets are de-multiplexed to either the TS unit or interface. The point is that if one of these two paths gets blocked, the remaining flows from the upstream stage cannot pass through the other path if it is idle. This blocking probability can be considerably reduced by considerably increasing the size of both the interface and TS unit buffers, which is an expensive solution for such systems. The idea is to reduce the blocking probability with a low hardware cost. Therefore, each stage adopts congestion condition of its downstream transfer stage buffers (interface and TS unit buffers) so that it can decide to deliver a packet to the less congested path of the next stage. The congestion condition of SH and MH paths, indicating the stress value of the interface and TS unit buffers, can be transmitted from one layer to another through two separate inter-layer signals (MH_Status and SH_Status). These signals are employed by the TS unit of the upstream transfer stage to forward a non-blocking packet, i.e. send a MH packet if the SH path is congested/blocked or send a SH packet if the MH path is congested/blocked.

As depicted in Fig. 6-2(b), each TS unit is composed of a table and a buffer. Each row of the table corresponds to a packet and includes a valid tag (v), a packet type (T), a packet age (A) and a header pointer (P). In the buffer, the flits of each packet are stored with a linked list structure providing high resource efficiency with a little hardware overhead. Fig. 6-2(b) exhibits a pointer field adopted to indicate the next flit position in the buffer. As multiple packets might be stored in the buffer, an arbitration mechanism is needed to determine which packet is allowed to be transmitted. The TS unit arbitration decision is based on the stress value of MH and SH paths. This arbiter selects the oldest packet (highest age) requesting an available path with the lowest stress value. Afterward, the age value of each packet having the same type as the selected one is increased by the arbiter to prevent starvation.

6.4.4 Synchronizing FIFO

Bi-synchronous (Bi-Sync) FIFOs are widely used in multi-clock systems to synchronize signals from different clock/frequency domains. Each domain is synchronous to its own clock signal but can be asynchronous with respect to others in either clock frequency or phase [117]. The challenges of designing Bi-Sync FIFOs include the enhancement of reliability and reducing latency and power/area cost. We identify the Bi-Sync FIFO structure presented in [118] as a suitable synchronizer to be used in the interfaces.

The structure of the Bi-Sync FIFO is depicted in Fig. 6-5. The FIFO implementation uses two pointers, one defining the next writing position and another defining the next reading position. The FIFO state is either full or empty when both pointers refer to the same address. Thus, it is necessary to compare the pointers. Although this procedure is trivial in synchronous circuits, it implies some complexity in the Bi-Sync FIFO, because the pointers are generated by different clocks.

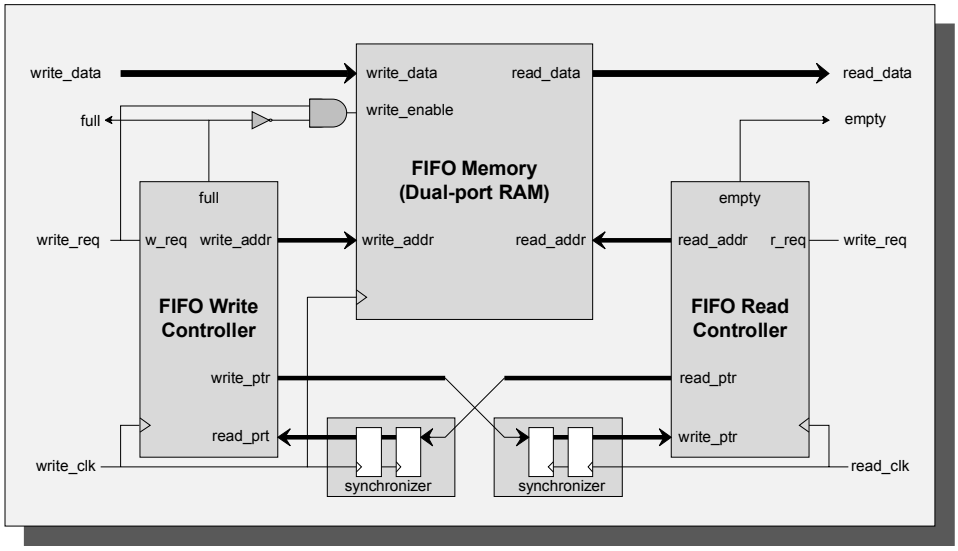


Fig. 6-5. Bi-Sync FIFO structure.

Table 6-1. Description of Bi-Sync FIFO signals.

Signal	Description
write_data	Data to be written in the FIFO
write_req	Write request
write_clk	Clock signal in the write domain
full	Signal to indicate the FIFO is full and no more data can be received
read_data	Data to be read from the FIFO
read_req	Read request
read_clk	Clock signal in the read domain
empty	Signal to indicate the FIFO is empty and henceno data can be read

A common solution to this problem is to transfer and synchronize the writing pointer (reading pointer) with the receiver clock domain (the sender clock domain) which generates the empty signal (the full signal). Exchanging the pointers (write_ptr and read_ptr) via a

handshake protocol implies additional latency. Therefore, two synchronizers are utilized for exchanging the pointers [118]. The addresses are converted to the gray code which guarantees that consecutive addresses are at a Hamming distance of 1. In this way, the metastability problem is confined to a single bit and synchronizers can be employed without handshake. Utilizing the Bi-Sync FIFO in the interfaces, allows each layer to work with its own clock source. Table 6-1 lists the input and output signals and their functionalities of the Bi-Sync FIFO.

6.5 Cluster Architectures

As mentioned earlier, both 3D-symmetric and 3D-hybrid structures require a large number of TSV interconnections for inter-layer communication. In addition, each TSV requires a pad (around 5µ×5µ) with the pitch of around 8µ for bonding to a wafer, thereby, the area overhead of TSVs impose constraints on the number of TSVs [22][29][131]. In order to reduce vertical channels, we present two novel topologies, named CIT (Concentrated Inter-layer Topology) and CMIT (Clustered Mesh Inter-layer Topology). Although both of the presented topologies can be implemented as the 3D-symmetric (7-port router with vertical packet switched interconnection) and 3D-hybrid (vertical bus with an interface at each 6-port router) structures, we describe these topologies based on the 3D-hybrid scheme which is more efficient than the 3D-symmetric structure [22][29]. To compensate the performance loss due to using the bus as the vertical interconnect, each vertical channel is composed of two unidirectional channels in opposite directions to propagate the inter-layer data.

6.5.1 CIT (Concentrated Inter-layer Topology)

Unlike the mesh topology where each IP-core is connected to a router, CIT forms a scalable architecture by sharing a router between multiple nodes (IP-cores and memories). CIT reduces the number of routers decreasing the number of vertical channels and hop counts. A 3×3 CIT with 36 nodes is shown in Fig. 6-6(a), where four nodes are grouped into a cluster, thereby forming 9 clusters in the network. Each cluster has a router with at most 9 ports (10 ports for 3D-symmetric structure): one connected to the bus, four to IP-cores/memories and the other four ports to neighbor routers. Communication channels in CIT can be classified as intra-layer channels (horizontal channels) and inter-layer channels (vertical channels). As illustrated in Fig. 6-6, the inter-layer communication is achieved by the cluster nodes. Each cluster node has a cluster core to establish the vertical connection via an interface to the vertical bus.

Due to the fact that each CIT router has larger number of input ports than the symmetric and hybrid routers, it consumes more area and power in comparison with conventional routers in the two other structures. In addition, the larger number of input ports becomes a performance bottleneck in terms of increased router complexity and contention probability inside the router (i.e. there are more input ports competing for an output port) [22][27]. Nonetheless, as the number of routers is decreased by the clustering approach in CIT, it not only reduces the area and power dissipation of the network but also the TSV area footprint

is considerably diminished on each layer. Besides, the distance between two nodes of the same cluster in CIT is only one router so that the data transmission between nodes of the same cluster can be very fast. That is, the latency in CIT for distant nodes is more than that in the mesh-based 3D structures, but for nearby nodes the latency in CIT is smaller.

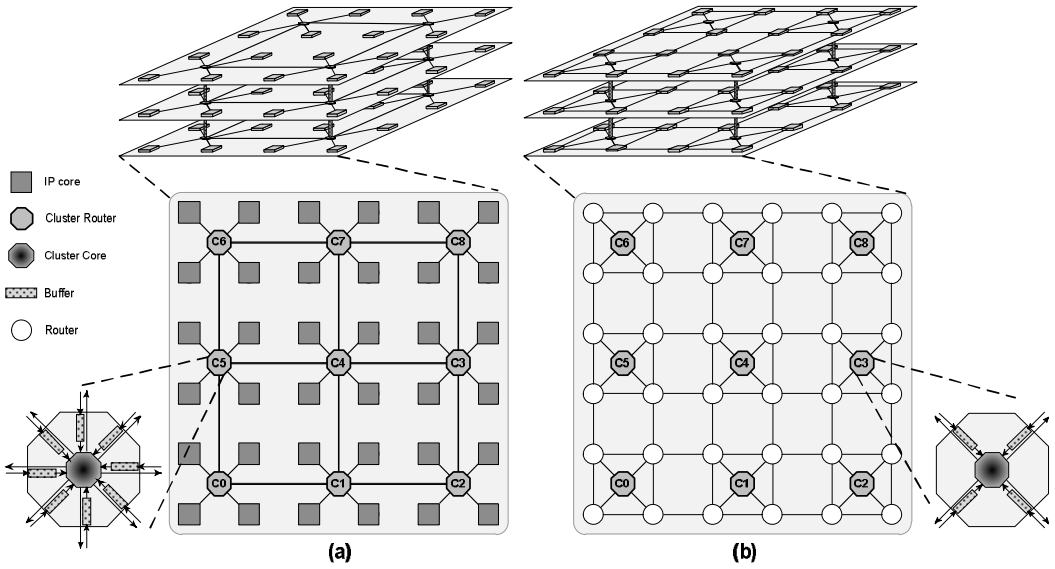


Fig. 6-6. Clustering approaches: (a) CIT and (b) CMIT.

Table 6-2. Number of routers, cluster routers and vertical channels of the described topologies in a (4×4×3) 3D architecture.

Topology	# of routers	# of cluster routers	# of vertical channels
3D-hybrid	64	0	64
CIT	0	16	16
CMIT	64	16	16

6.5.2 CMIT (Cluster Mesh Inter-layer Topology)

The structure of CMIT, depicted in Fig. 6-6(b), is basically similar to that of the mesh topology, except that for every layer the number of vertical channels has been reduced by sharing a vertical bus among routers of each cluster. That is, even preserving the advantage of the mesh on each layer, CMIT diminishes the number of inter-layer interconnection to meet constraints on the number of TSVs.

In CMIT, each router has at most 6 ports: one to the node (IP-core/memory), one to the bus (cluster router), and four for neighbors. Fig. 6-6(b) exhibits CMIT with 64 nodes, in which

every four routers are grouped into a cluster on each layer. Even though CMIT achieves better area and power efficiency than the typical 3D mesh structure due to reducing the number of vertical channels, since several routers are connected to a shared vertical bus, the performance may be degraded when the inter-layer traffic is augmented. The specification of the three described architectures has been summarized in Table 6-2. The arbiter should be placed in the middle layer of the chip to keep wire lengths as uniform as possible. The number of control wires of each arbiter increases with the number of nodes attached to the vertical channel (bus). As a result, the presence of a centralized arbiter is the reason why the number of vertical channels in the chip should be kept low [22][29]. We believe that, the proposed topology can keep the number of vertical channels low with a negligible performance penalty.

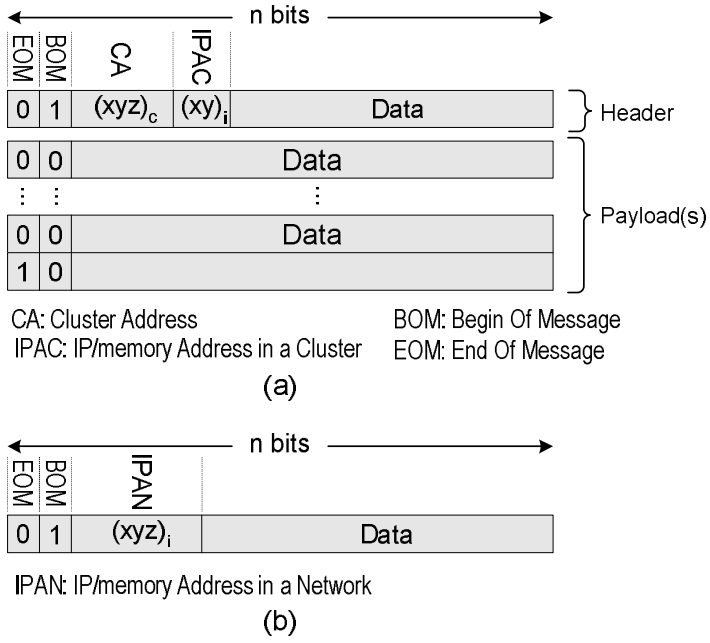


Fig. 6-7. (a) The packet format in CIT and (b) the header format in CMIT.

6.5.3 Routing Algorithm

For the presented CIT, we employ the dimension order routing (DOR) algorithm which guarantees the network is deadlock-free. DOR is a minimal deterministic routing scheme in which the message is first forwarded along the X-dimension, then along the Y-dimension, and, finally, along the Z-dimension. Fig. 6-7(a) shows the packet format of the CIT network. The header flit is n-bit wide and the n^{th} bit is the EOM (End Of Message) sign and

the $(n\text{-}1)^{th}$ bit is the BOM (Begin Of Message) sign. The third field indicates the address of the destination cluster and the next one is used for the IP-core/memory address inside the cluster. The content of the message is located in the rest of the flits (Payload). As can be seen from Fig. 6-7(b), the packet format of the CMIT network only has one destination address field (i.e. IPAN) similar to the mesh network. In CIT, the cluster routers perform the routing mechanism, while in CMIT the routing is performed by typical routers. That is, cluster routers are only employed for the purpose of inter-layer communication in CMIT.

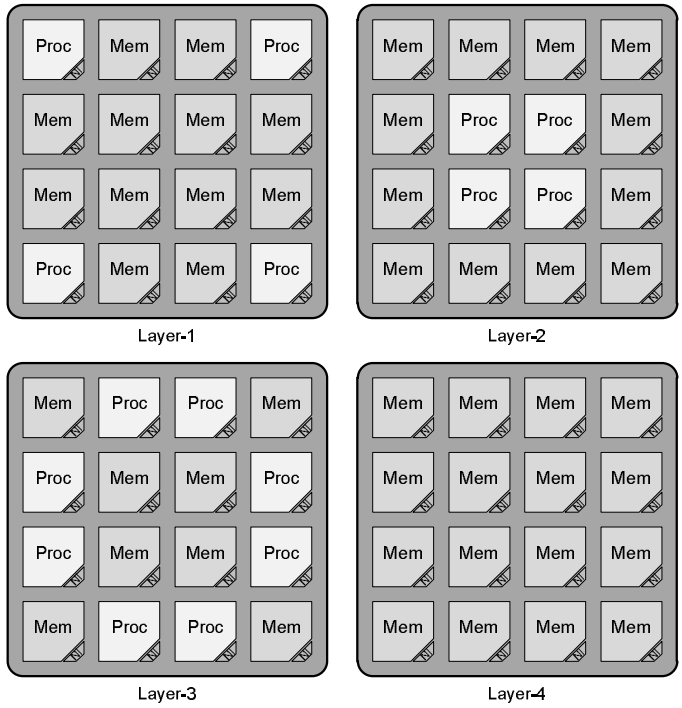


Fig. 6-8. 4×4×4 stacked mesh layout.

6.6 Experimental Results

In this section, we compare the presented topologies with the conventional structures in terms of latency, power consumption, and area cost. Also, the impact of using the novel pipelined bus has been explored. Hence, a cycle-accurate 3D NoC simulator is developed to assess the efficiency of the proposed architectures. The simulator models all major components of the NoC such as network interfaces, routers, and wires along with vertical channels.

6.6.1 System Configuration

We configure a 64-node (4×4×4) 3D stacked architecture. In this configuration, illustrated in Fig. 6-8, out of 64 nodes, 16 nodes are assumed to be processors and other 48 nodes are memory blocks, i.e. DRAMs. The processors are 32b AXI compatible core and the memory blocks are DDR2-256MB (t_{RP}-t_{RCD}-t_{CL}=2-2-2, 32b, 4 banks) [116]. Three different 3D on-chip network topologies are considered for experiment: 3D-hybrid structure, CIT, and CMIT. The 3D-hybrid and CMIT networks are formed by a typical state-of-the-art router structure including input buffers, a VC (Virtual Channel) allocator, a routing unit, a switch allocator, and a crossbar as well as an interface unit connecting the router to either a vertical channel (bus) or a cluster router. Typical routers of 3D-hybrid and CMIT have at most six input/output ports. Every cluster router of CMIT has five input/output ports, i.e. four for local routers and one for the vertical channel interface, while cluster routers of CIT have at most nine input/output ports, i.e. four for local IP-cores/memories connections, at most four for neighboring cluster routers, and one for the vertical channel interface. Each input port of router has 2 VCs where packets of different message types (request and response) are assigned to corresponding VCs to avoid message dependency deadlock [119]. The arbitration scheme of the switch allocator in the typical router structure is round-robin. The array size, routing algorithm, link width, number of VCs, buffer depth of each VC, and traffic type are the other parameters which must be specified for the simulator. The routers adopt the DOR routing scheme and utilize wormhole switching. For all routers, the data width (flit) was set to 32 bits, and the buffer depth of each VC is 5 flits. As mentioned earlier, to compensate the performance loss due to using the bus as the vertical interconnect, each vertical channel is composed of two unidirectional channels in opposite directions to propagate the inter-layer data. Thus, 32 bits of the channel is allocated to upward direction and the other 32 bits of the channel is employed for the downward direction. Each channel has its arbiter module and bus controller [22][132]. The depth of buffers in the transfer stage is 6 flits.

The presented configuration uses 1 flit for messages related to read requests and write responses, and the size of read request messages typically depends on the network size and memory capacity of the configured system. The message size of the read responses and write requests is variable and depends on the request/response length produced by a master/slave core (burst size 1:8). As for the performance metric, we use latency defined as the number of cycles between the initiation of a request operation issued by a master (processor) and the time when the response is completely delivered to the master from a slave (memory). The request rate is defined as the ratio of the successful read/write request injections into the network interface over the total number of injection attempts. All the cores and routers are assumed to operate at 1 GHz. For fair comparison, we keep the bisection bandwidth constant in all configurations. All memories (slave cores) can be accessed simultaneously by each master core with continuously generating memory requests. To estimate the power consumption of networks, we have used Orion [86] (estimate both dynamic and static power) as well as the power and delay values of vertical links in [133].

6.6.2 Performance Comparison

To assess the performance of the presented pipeline bus architecture, the uniform and non-uniform synthetic traffic patterns have been considered separately and we expect realistic applications stand between these two synthetic traffic patterns. The random traffic represents the most generic case, where each processor sends in-order read/write requests to memories with the uniform probability, and the memories and request type (read or write) are selected randomly. Eight burst sizes, from 1 to 3, are stochastically chosen according to the data length of the request. In the non-uniform mode, 70% of the traffic is local requests, where the destination memory is one hop away from the master core, and the rest 30% of the traffic is uniformly distributed to the non-local memory modules.

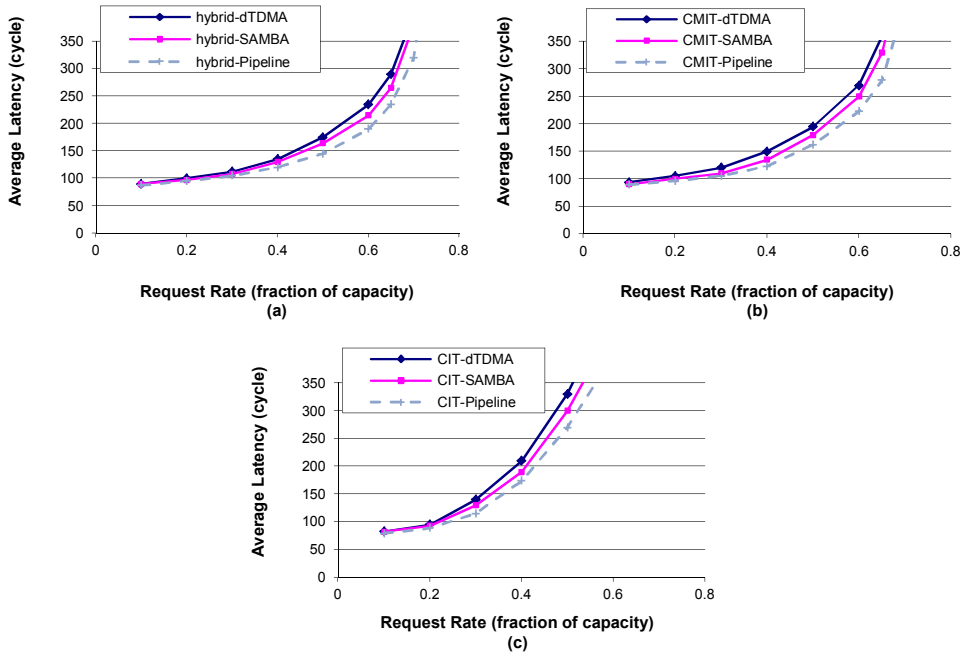


Fig. 6-9. Performance impact of using the presented bus in (a) hybrid, (b) CMIT, and (c) CIT networks under uniform traffic profile.

Here, we explore the average latency of using the presented pipeline bus architecture. Two conventional baseline buses, dTDMA [22][132] and SAMBA [130], have been considered to be used for vertical channels. Fig. 6-9 and Fig. 6-10 show the performance gain of employing the pipeline bus architecture for vertical channels in the 3D-hybrid, CMIT and CIT networks under uniform and non-uniform traffic profiles, respectively. This is achieved due to having a small local arbiter in the transfer stage such that the arbitration delay is reduced significantly.

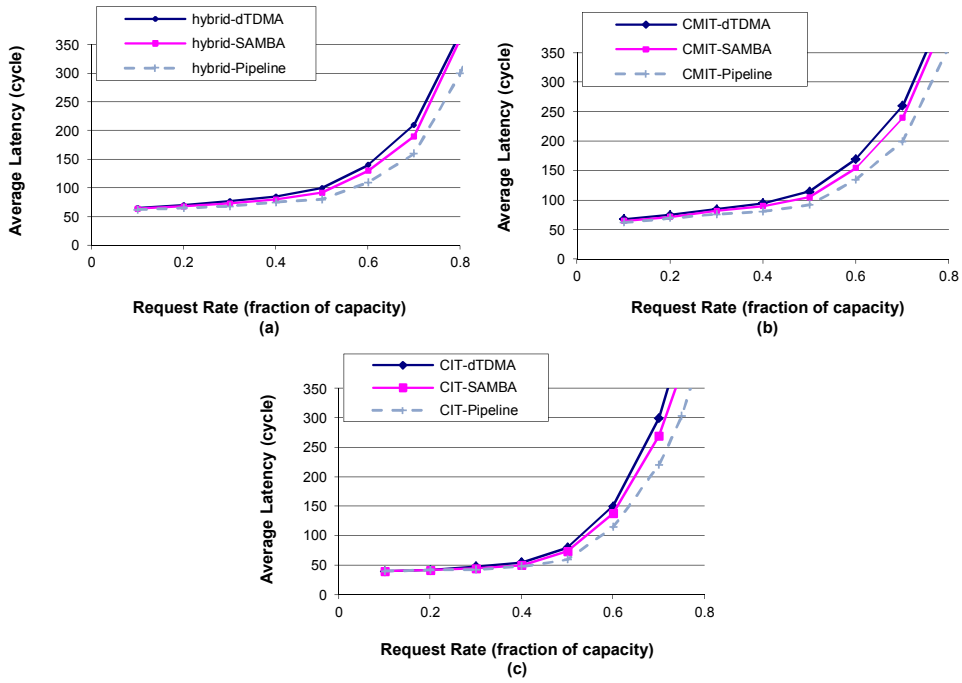


Fig. 6-10. Performance impact of using the presented bus in (a) hybrid, (b) CMIT, and (c) CIT networks under non-uniform traffic profile.

To explore the efficiency of the two presented topologies without considering the pipeline bus (employing dTDMA), the simulation results under the uniform, non-uniform, and hotspot traffic profiles are depicted in Fig. 6-11. In the hotspot traffic pattern, one or more nodes are designated as hotspot nodes receiving an extra portion of the traffic in addition to the regular uniform traffic. Newly generated packets are directed to each hotspot node with an additional H percent probability. We simulate hotspot traffic with four hotspot nodes. Four hotspot nodes are chosen at the center of each layer, (2, 2, 1), (3, 3, 2), (2, 3, 3), (3, 2, 4), with equal probability of H=20%.

As demonstrated in Fig. 6-11(a) and (c), CIT has the lowest average latency in the low traffic load (<0.2), one of the foremost reasons for such an improvement is that CIT reduces the average hop count and improves load balance across the channels. But in high traffic load the performance of CIT degrades considerably since the network bandwidth in CIT is lower than that of mesh-based structures. That is, the number of links in CIT is much smaller than that of mesh-based structures. Therefore, in the high traffic load, the traffic in CIT links is much higher than in mesh-based structures. Another subtle point regarding clustered topology is that the latency in CIT for distant destinations is significantly larger than that of mesh-based structures due to the high router complexity and contention probability, while for nearby destinations the latency of CIT is smaller. Thus, CIT might have better performance in applications where most of requests are issued among

neighboring nodes under low traffic load. This can be seen from the results in Fig. 6-11(b) where each processor sends requests to the memories based on the non-uniform traffic profile. CIT outperforms the others in terms of latency when the request rate is below the saturation point and most of the traffic is local. The average latency of each presented topology has been computed near saturation point (0.5) under the non-uniform traffic profile. As a result, compared with the 3D-hybrid and CMIT, the average latency of CIT is reduced by 20% and 30%, respectively.

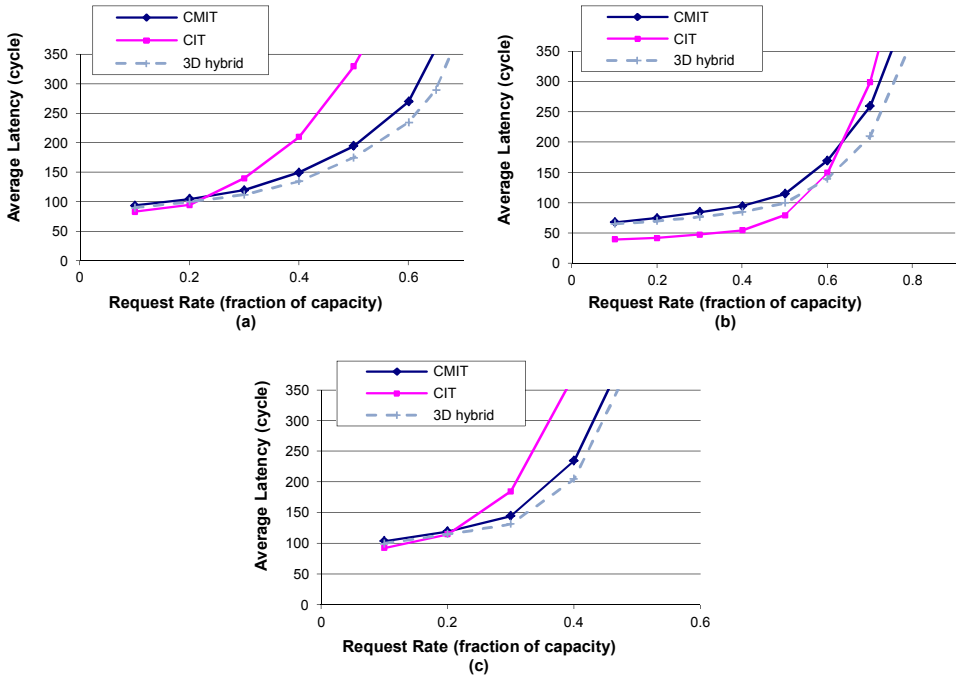


Fig. 6-11. Performance comparison of different 3D structures for (a) the uniform, (b) non-uniform, and (c) hotspot traffic profiles using dTDMA.

Fig. 6-12(a)-(c) demonstrate the performance impact of the pipeline bus on different topologies for uniform, non-uniform, and hotspot traffic profiles. As illustrated, employing the pipeline bus for the hybrid structure outperforms the 3D symmetric structure where each router includes 7 ports. This performance gain is due to the fact that the pipeline scheme is inspired from typical router which is optimized for inter-layer communication and can benefit from the weight-based arbitration and the non-blocking scheme.

Additionally, to illustrate how local traffic under the non-uniform profile can affect the performance, we scale the amount of local traffic from 0 to 100%. The results obtained at rate 0.5 (near the saturation point) are shown in Fig. 6-13 where CIT, CMIT, and hybrid utilize the proposed pipeline bus. Like in the previous non-uniform experiments, CIT

achieves significant latency reduction when the amount of local traffic is increased, particularly from 40%.

In order to explore the real impact of the proposed inter-layer scheme, we use traces generated using the GEMS simulator from SPLASH-2. We use the Radix, Ocean, and FFT applications from SPALSH-2 for our simulations.

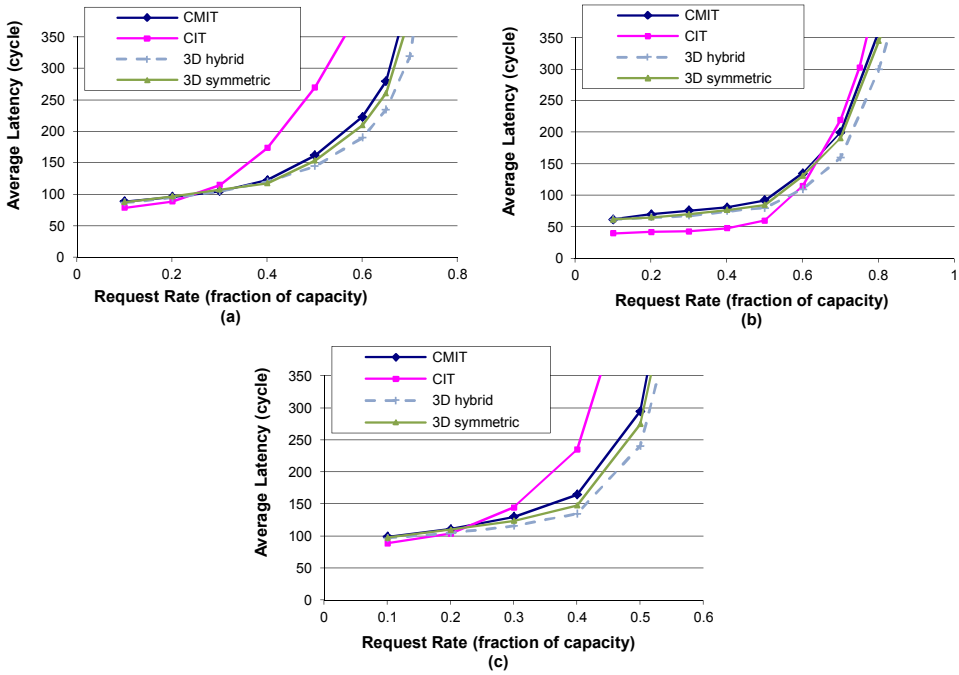


Fig. 6-12. Performance comparison of different 3D structures for (a) the uniform, (b) non-uniform, and (c) hotspot traffic profiles using the presented bus.

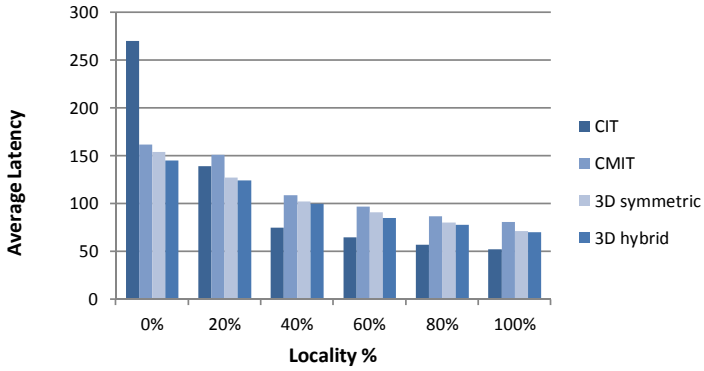


Fig. 6-13. Performance impact of topologies using presented bus with different local loads.

Table 3-2 summarizes our full system configuration where the cache coherence protocol is MESI. We configure a 64-node on-chip network (4×4×4) that four layers are stacked on top of each other, i.e. out of the 64 nodes, 16 nodes are processors and other 48 nodes are L2 caches. L2 caches are distributed in the bottom three layers, while all the processors are placed in the top layer close to a heat sink so that the best heat dissipation capability is achieved. The simulator produces, as output, the communication latency for cache access. The CIT, CMIT, and hybrid configurations are equipped with the pipeline bus. Fig. 6-14 shows the average network latency of the real workload traces collected from the aforementioned system configurations. We can see that the hybrid configuration consistently reduces the average network latency across all tested benchmarks. It shows a steady reduction amount: 12%~25% (hybrid/CIT), 4%~16% (hybrid/CMIT), and 3%~13% (hybrid/symmetric) with the average of 19%, 10%, and 7%, respectively.

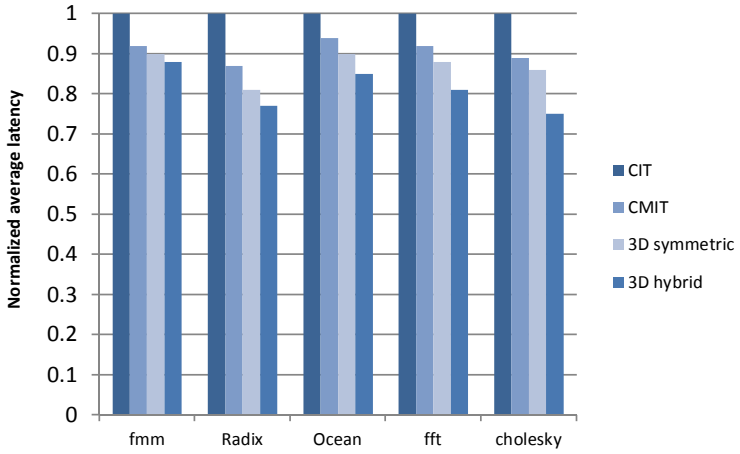


Fig. 6-14. Performance for application traces normalized to CIT.

6.6.3 Power Analysis

Using the simulator, the average power consumption of the presented topologies was calculated and compared under uniform and non-uniform traffic patterns close to the saturation point. The results are shown in Fig. 6-15(a) and (b). According to Fig. 6-11(a) and (b), the saturation points that have been considered for computing the average power values are 0.3 for uniform traffic and 0.5 for non-uniform traffic. As presented in Fig. 6-15(a), under the uniform traffic profile, the average power dissipation of the CIT scheme is 30%, 16%, and 10% less than those of the 3D-symmetric, 3D-hybrid and CMIT schemes, respectively. Furthermore, the results in Fig. 6-15(b) indicate that the average power of CIT, under the non-uniform traffic profile, is 39%, 35%, and 30% less than that of

the 3D-symmetric, 3D-hybrid, and CMIT schemes, respectively. We can notice that although the power consumption of every cluster router is about 1.5 times more than the power consumption of a typical router, the average power in the CIT network, compared to other schemes, is considerably lower under non-uniform traffic profile since the average number of hops between two arbitrary nodes is less than in the other presented schemes. Also, to illustrate how the proposed pipeline bus affects the power dissipation, we compute the average power of each network close to its saturation point under the uniform traffic profile. Based on the achieved results, the average power consumption of the pipeline bus in the hybrid network is diminished by 10% and 12% compared with SAMBA and dTDMA, respectively. The average power reduction of using pipeline bus in the CMIT network is 6% and 9% compared with SAMBA and dTDMA, respectively, while in the CIT network, it is 5% and 10%. In fact, this power saving is obtained because of the following reasons. First, the hardware overhead of the pipeline bus is smaller than that of SAMBA and dTDMA. Second, central arbiters, employed in SAMBA and dTDMA, cause a lot of switching compared to small local arbiters used in pipeline bus so that the power dissipation of those two buses are higher than that of the pipeline bus.

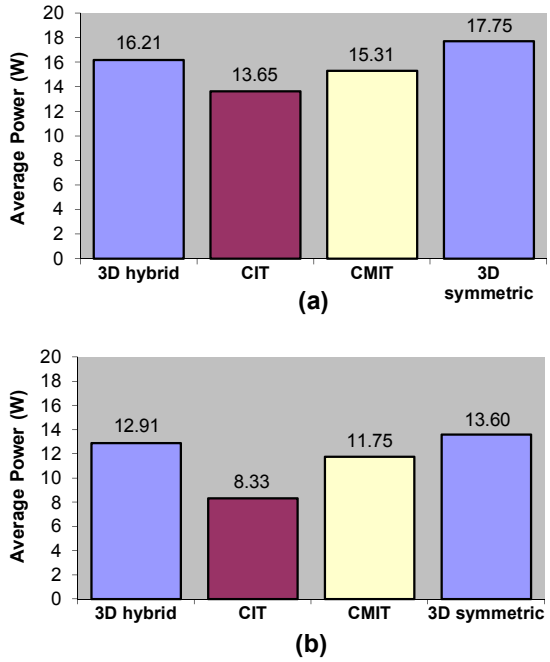


Fig. 6-15. Average power dissipation results under (a) uniform and (b) non-uniform traffic profiles.

6.6.4 Physical Analysis

The number of routers and vertical channels in a chip affects the area and implementation cost. Thus, to compute the network area for each topology, we estimate the area of routers, cluster routers, and vertical channels. The network platform of each topology with the aforementioned configuration is synthesized using Synopsys Design Compiler with the UMC 0.09µm technology, while the backend is performed with the Cadence Encounter tool. Depending on the technology and manufacturing process, the pitch of TSVs can range from 1µm to 10µm [22][23][28]. The pad size for TSVs is assumed to be 5µm square with pitch of around 8µ, the flit-width is set to 32 bits, and each vertical channel requires 3×14 control wires for arbitration in four-stacked layers [22]. Hence, after the TSV area is calculated with the given values, the TSV footprint has been reduced from 0.41 mm^2 in 3D-hybrid to 0.1 mm^2 in CIT and CMIT, resulting in about 75% area saving for the TSV footprint. 3D-hybrid occupies a larger network area than CIT and CMIT, because each router in 3D-hybrid has a transceiver module to interface with the vertical channel (bus) [22] and each bus should have its own arbiter module. Like 3D-hybrid, the transceiver and arbiter modules are only integrated in cluster routers of CIT and CMIT. On the other hand, the total network area used by CIT is significantly smaller than that of the other architectures (37% and 42% less than that of CMIT and 3D-hybrid respectively) since the network is formed only by cluster routers.

Using the pipeline bus reduces the TSV footprint for each vertical channel. As each vertical channel, i.e. two unidirectional 32-bit dTDMA buses, occupies 6400μm^2 and the required area for each vertical channel using the pipeline bus is 4096μm^2, the proposed bus scheme can save more than 35% of the TSV area footprint with a performance gain. Hence, after considering the TSV footprint area, the hardware overhead of the pipeline bus is approximately 10% and 8% less than that of the conventional dTDMA and SAMBA buses, respectively. In addition, comparing the cost of the network using the presented pipeline bus with the 3D symmetric network reveals that the area overhead of the 3D symmetric network is about 10%. This is because a 7-port router is larger than a 6-port router.

6.7 Summary

3D stacked architectures provide significant benefits in performance, footprint and yield. It has been demonstrated that combining 3D ICs and on-chip networks can be a promising option for designing large multiprocessor architectures. One critical issue in 3D design is that the vertical interconnections are very fast and fat such that the area overhead of TSVs impose constraints on the number of TSVs for existing 3D architectures. In this chapter, two cluster-based topologies have been presented to deal with constraints on the number of TSVs. Also, a novel pipeline bus structure for vertical channels is introduced not only to mitigate the drawbacks of existing bus structures in terms of power and performance, but also to reduce the number of required inter-layer arbiter control signals. Experimental results revealed that the on-chip network formed by the two presented topologies (CIT and

CMIT) reduces the number of TSVs significantly with low performance penalty under uniform traffic, but under non-uniform traffic which is more realistic case, CIT outperformed the other network structures in terms of the average network latency.

Chapter 7

Conclusion

In the last chapter of this thesis, it is the intention of the author to summarize and integrate all chapter concluding remarks in order to present a consistent overall picture of the achievements. A few remaining open problems and some interesting future research ideas are also detailed here. This is done to show that there are options to continue this thesis.

7.1 Thesis Contributions

As one of the main contribution of this thesis, three adaptive routing protocols have been presented where the first one is a unicast-based routing while the other two are unicast- and multicast-based routing schemes. The unicast-based scheme is a congestion-aware adaptive routing protocol in which two congestion wires (one in each direction) between any two routers are added to indicate the existence of congestion in a row (column). That is, two congestion wires are added to each router to flag a row or column congestion further away from the current switch. These signals enable the routing protocol to avoid these paths when there are other paths between the source and destination pair, and thus decreasing the latency of the routing protocol. Exploiting the unicast routing protocols for multicast communication increases the likelihood of deadlock and congestion. In order to avoid deadlock for multicast communication, the Hamiltonian path strategy was introduced. The traditional Hamiltonian path routing protocols supporting both unicast and multicast traffic are based on deterministic models, leading to lower performance. In this thesis, two adaptive routing schemes for both unicast and multicast communications without using virtual channels have been proposed. The presented routing schemes invoke non-congested paths for routing the messages to prevent creating highly congested areas.
In this thesis, a router architecture based on the adaptive input and output selection is proposed. The output selection of the router utilizes an adaptive routing algorithm supporting both unicast and multicast traffic while the input selection part of the router uses the weighted round robin arbitration. Also, the adaptive output selection algorithm supporting both minimal and non-minimal paths uses congestion flags to route packets

through non-congested paths and consequently helps balance the traffic. The weighted round robin input selection also assists in relieving nodes where congestion is formed.
In order to increase the memory bandwidth in network-based multiprocessor architectures multiple memory modules can be accessed in parallel. On top of resource utilization and latency, a reordering mechanism is required to deliver the response transactions of concurrent memory accesses in-order. Therefore, in this thesis, we presented a high performance network interface with a novel dynamic buffer allocation and a priority-based router model to improve the resource utilization, and overall on-chip network performance. In addition to the resource utilization of the network interface and on-chip network, also the utilization of memories considerably affects the network latency. Accordingly, an optimized scheduling method for the DRAM memories is developed and integrated in the network interface such that the network and memory latencies were reduced significantly in comparison with the baseline architecture. The micro-architectures of the proposed network interfaces which are compatible with the AMBA AXI protocol have been presented.
Three-Dimensional (3D) stacked architectures provide significant benefits in performance, footprint and yield. It has been demonstrated that combining 3D ICs and on-chip networks can be a promising option for designing large multiprocessor architectures. One critical issue in 3D design is that the vertical interconnections are very fast and thick such that the area overhead of Through-Silicon-Vias (TSVs) imposes constraints on the number of TSVs for existing 3D architectures. In this dissertation, two cluster-based topologies have been presented to deal with constraints on the number of TSVs. Also, a novel pipeline bus structure for vertical channels is introduced not only to mitigate the drawbacks of existing bus structures in terms of power and performance, but also to reduce the number of required inter-layer arbiter control signals. Experimental results revealed that the on-chip network formed by the two presented topologies, Concentrated Inter-layer Topology (CIT) and Cluster Mesh Inter-layer Topology (CMIT), reduces the number of TSVs significantly with a low performance penalty under uniform traffic, but under non-uniform traffic which is more realistic case, CIT outperformed the other network structures in terms of the average network latency.

7.2 Future Directions

Some interesting open problems that are tightly related to the work in this thesis are as follows.
3D chip stacking technology is emerging as a viable candidate to address the memory bandwidth problem, memory wall, by stacking multiple DRAM layers on top of a multiprocessor layer (logic layer) to reduce wire delay and energy consumption between them. In addition, combining the benefits of 3D memory-on-processor stacking architecture and on-chip networks provides a significant performance gain. To fully exploit the benefits of the 3D stacked memory-on-processor architectures, efficient on-chip communication platforms are required to be explored for different stacked layers. Also, to guarantee low-

latency access to the stacked DRAM layers, adaptive memory controllers may need to be designed.

Achieving higher performance along with reducing the network latency can be obtained by applying efficient communication protocols in 3D NoC-based CMPs. As the multicast communication is utilized commonly in various parallel applications, the performance can be significantly improved by supporting multicast operations at the hardware level. Various partitioning methods can be developed to distribute the multicast traffic among several subsets. Furthermore, several factors of efficiency such as average unicast latency, average multicast latency and average startup latency can be studied by analytical models.

References

[1] L. Benini, G. De Micheli, "Networks on Chips: A New SoC Paradigm," IEEE Computer, pp. 70-78, January 2002.

[2] A. Jantsch and H. Tenhunen, "Networks on Chip," Kluwer Academic Publishers, 2003.

[3] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," In Proc. of Design Automation Conference (DAC), pp. 684-689, June 2001.

[4] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist. "Network on chip: An architecture for billion transistor era," In Proc. of the IEEE Norchip Conf., pp. 120-124, November 2000.

[5] S. Kumar, A. Jantsch, M. Millberg, J. Oberg, J. Soininen, M. Forsell, K. Tiensyrj, and A. Hemani. "A network on chip architecture and design methodology," In Proc. Symposium on VLSI, pp. 117-124, April 2002.

[6] S. Vangal et al., "An 80-tile 1.28TFlops Network-on-Chip in 65nm CMOS," In Proceedings of ISSCC'07, pp. 98-100, 2007.

[7] Tilera. http://www.tilera.com, 2008.

[8] Semiconductor Association. The International Technology Roadmap for Semiconductors (ITRS).

[9] Semiconductor Association. Semiconductor Industry Association (SIA).

[10] G. D. Micheli, L. Benini, "Powering Networks on Chips: Energy-Efficient and Reliable Interconnect Design for SoCs," in Proceedings of the 14th international ieee symposium on Systems synthesis (ISSS '01), pp.33-38, 2001.

[11] F. Moraes , A. Mello, L. Moller, L. Ost, and N. Calazans, "A Low Area Overhead Packet-switched Network on Chip: Architecture and Prototyping," in Proc. of International Conference on Very Large Scale Integration (VLSI-SoC), Germany, pp. 318-323, 2003.

[12] A. A. Chen, and J. H. Kim, "Planar-Adaptive routing: Low-cost adaptive networks for multiprocessors," In Proc. of 19th Ann Int'l Symp Computer Architecture, pp. 268–277, 1992.

[13] ARM, AMBA AXI Protocol Specification, Mar. 2004.

[14] OCP International Partnership, Open Core Protocol Specification. 2.0 Release Candidate, 2003.

[15] Philips Semiconductors, Device Transaction Level (DTL) Protocol Specification. Version 2.2, Jul. 2002.

[16] D. Atienza, F. Angiolini, S. Murali, A. Pullini, L. Benini, G. De Micheli, "Network-On-Chip Design and Synthesis Outlook," Integration-The VLSI journal, vol. 41, no. 3, pp. 340-359, 2008.

[17] J. Henkel, W. Wolf, S. Chakradhar, "On-chip networks: A scalable, communication-centric embedded system design paradigm," in proc. of 17th International Conference on VLSI Design (VLSID), pp.845, India, 2004.

[18] K. Banerjee, S. J. Souri, P. Kapur, K. C. Saraswat, "3D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration", Proc. of the IEEE, 89(5):602-633, May 2001.

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

[20] D. Park, S. Eachempati, R. Das, A. K. Mishra, Y. Xie, N. Vijaykrishnan, and C. R. Das, "MIRA: A Multi-Layered On-Chip Interconnect Router Architecture", ISCA 2008, pp. 251-261, Pennsylvania State, USA.

[21] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, and C. R. Das, "A novel dimensionally-decomposed router for on chip communication in 3D architectures," in Proc. of the ISCA, pp. 138-149, Boston, USA, 2007.

[22] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir, "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," In 33rd International Symposium on Computer Architecture (ISCA), pp. 130–141, 2006.

[23] I. Loi and L. Benini, "An Efficient Distributed Memory Interface for Many-Core Platform with 3D Stacked DRAM," in Proc. of the DATE Conference, Germany, pp. 99-104, 2010.

[24] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, "Demystifying 3D ICs: The Pros and Cons of Going Vertical," IEEE Design and Test of Computers, 22(6):498–510, Nov. 2005.

[25] I. Loi, F. Angiolini, L. Benini, "Supporting vertical links for 3D networks on chip: toward an automated design and analysis flow," in Proc. Nanonets, 2007.

[26] K. Snoeckx, E. Beyne, and B. Swinnen, "Copper-nail TSV technology for 3D-stacked IC integration," Solid State Technology, Vol. 50, No. 5, pp.53-55, 2007.

[27] A. Y. Weldezion, M. Grange, D. Pamuwa, Z. Lu, A. Jantsch, R. Weerasekera, and H. Tenhunen. Scalability of the Network-on-Chip communication architecture for 3D meshes. In International Symposium on Networks-on-Chip (NoCS), pp. 114-123, 2009.

[28] I. Savidis, S. M. Alam, A. Jain, S. Pozder, R. E. Jones, R. Chatterjee, "Electrical modeling and characterization of through-silicon vias (TSVs) for 3D integrated circuits," Microelectronics Journal, Vol. 41(1), pp. 9-16, 2010.

[29] S. Pasricha, "Exploring Serial Vertical Interconnects for 3D ICs," in Proc. IEEE/ACM DAC, pp. 581-586, 2009.

[30] S. Das, A. Chandrakasan, R. Reif, "Three Dimensional Integrated Circuits: Performance, Design, Methodology and CAD tools," in proc. of ISVLSI, pp. 13–18, 2003.

[31] S. Das et al., "Technology, Performance, and Computer Aided Design of Three-Dimensional Integrated Circuits," In Proc. International Symposium on Physical Design, USA, pp. 108-115, 2004.

[32] Y. Xie, G. H. Loh, B. Black, K. Bernstein, “Design Space Exploration for 3D Architectures,” ACM Journal on Emerging Technologies in Computing Systems 2, pp. 65–103, 2006.

[33] S.-M. Jung et al., “The revolutionary and truly 3-dimensional 25F2 SRAM technology with the smallest S3 cell, 0.16um2, and SSTFT for ultra high density SRAM,” in IEEE Symp. VLSI Tech., pp. 228-229, 2004.

[34] B. Black, M. Annavaram, N. Brekelbaum, J. Devale, L. Jiang, G. H. Loh, D. Mccauley, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, C. Webb, “Die-Stacking (3D) Microarchitecture,” in Proceedings of MICRO-39, pp. 469-479, 2006.

[35] G. H. Loh, “3D-Stacked Memory Architectures for Multi-core Processors,” in proc. of International Symposium on Computer Architecture (ISCA), pp. 453–464, 2008.

[36] P. Lotfi-kamran, A. Rahmani, M. Daneshtalab, A. Afzali-Kusha, Z. Navabi, “EDXY - A Smart Congestion-Aware and Link Failure Tolerant Routing Algorithm for Network-on-Chips,” Journal of Systems Architecture (JSA-elsevier), Vol. 56, No. 7, pp. 256-264, Jul 2010.

[37] M. Daneshtalab, M. Ebrahimi, S. Mohammadi, A. Afzali-Kusha, “Low distance path-based multicast algorithm in NOCs,” IET (IEE) Special issue on NoC, Vol. 3, Issue 5, pp. 430-442, Sep 2009.

[38] M. Ebrahimi, M. Daneshtalab, S. Mohammadi, Juha Plosila, H. Tenhunen, “An Efficent Dynamic Multicast Routing Protocol for Distributing Traffic in NOCs,” in Proc. of 12th IEEE/ACM Design, Automation, and Test in Europe (DATE), pp. 1064-1069, April 2009, France.

[39] M. Daneshtalab, M. Ebrahimi, T. C. Xu, P. Liljeberg, and H. Tenhunen, “A Generic Adaptive path-based routing method for MPSoCs," Journal of Systems Architecture (JSA-elsevier), Vol. 57, No. 1, pp. 109-120, 2011.

[40] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, H. Tenhunen, “HAMUM – A Novel Routing Protocol for Unicast and Multicast Traffic in MPSoCs,” in Proceedings of 18th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP), pp. 525-532, February 2010, Italy.

[41] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, “Input-Output Selection Based Router for Networks-on-Chip,” in Proceedings of 9th IEEE International Symposium on VLSI (ISVLSI), pp. 92-97, July 2010, Greece.

[42] M. Daneshtalab, M. Kamali, M. Ebrahimi, S. Mohammadi, A. Afzali-Kusha, and J. Plosila, “Adaptive Input-output Selection Based On-Chip Router Architecture,” Journal of Low Power Electronics (JOLPE) - (To appear).

[43] M. Rahmani, M. Daneshtalab, P. Liljeberg, H. Tenhunen, “Power-Aware NoC Router Using Central Forecasting-Based Dynamic Virtual Channel Allocation,” in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), pp. 3224-3227, May 2010, France.

[44] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, Juha Plosila, and H. Tenhunen, “Memory-Efficient On-Chip Network with Adaptive Interfaces,” To appear in IEEE Transaction on Computer Aided Design (IEEE TCAD).

[45] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, Juha Plosila, and H. Tenhunen, "A Low-Latency and Memory-Efficient On-hip Network," in Proceedings of 4th IEEE/ACM International Symposium on Network-on-Chip (NOCS), pp. 99-106, May 2010, France.

[46] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, H. Tenhunen, "A High-Performance Network Interface Architecture for NoCs Using Reorder Buffer Sharing," in Proceedings of 18th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP), pp. 547-550, February 2010, Italy.

[47] M. Ebrahimi, M. Daneshtalab, N. Sreejesh, P. Liljeberg, Juha Plosila, H. Tenhunen, "Efficient Network Interface Architecture for Network-on-Chips," in Proc. of 27th IEEE Norchip, pp. 1-4, Nov 2009, Norway.

[48] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, Juha Plosila, H. Tenhunen, "CMIT- A Novel Cluster-based Topology for 3D Stacked Architectures," in Proc. Of 2nd IEEE International 3D System Integration Conference (3DIC), Nov 2010, Germany.

[49] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, Juha Plosila, H. Tenhunen, "Efficient Inter-layer Bus Architecture for 3D Cluster-based Networks-on-Chip," submitted The Journal of Computer and System Sciences (JCSS-elsevier).

[50] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Pipeline-Based Interlayer Bus Structure for 3D Networks-on-Chip," in Proceedings of 15th International Symposium on Computer Architecture & Digital Systems (CADS), IEEE Press, pp. 41-47, Sept 2010, Iran.

[51] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "High-Performance TSV Architecture for 3-D ICs," in Proceedings of 9th IEEE International Symposium on VLSI (ISVLSI), PhD-Forum, pp. 467-468, May 2010, Greece.

[52] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, Juha Plosila, H. Tenhunen, "A Novel Interlayer Bus Architecture for Three-Dimensional Network-on-Chips," 3D Integration Workshop, The Design, Automation, and Test in Europe (DATE) conference, March 2010, Germany.

[53] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, H. Tenhunen, "Partitioning Methods for Unicast/Multicast Traffic in 3D-NoC Architecture," in Proceedings of 13th IEEE International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS), pp. 127-132, April 2010, Austria.

[54] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "Performance Analysis of 3D NoCs Partitioning Methods," in Proceedings of 9th IEEE International Symposium on VLSI (ISVLSI), PhD-Forum, pp. 467-468, May 2010, Greece.

[55] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, H. Tenhunen, "Exploring Partitioning Methods for 3D Networks-on-Chip Utilizing Adaptive Routing Model," in Proceedings of 5th ACM/IEEE International Symposium on Networks-on-Chip (NOCS), pp. 73-80, May 2011, USA.

[56] W. O. Cesario, D. Lyonnard, G. Nicolescu, Y. Paviot, S. Yoo, A. A. Jerraya, L. Gauthier, M. Diaz-Nava, "Multiprocessor SoC platforms: a component-based design approach," in proc. of IEEE Design and Test of Computers, pp. 52–63, 2002, France.

[57] E. Rijpkema, K.G.W. Goossens, A. Radulescu, 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," in proc. of IEEE Design Automation and Test Conference in Europe (DATE), pp. 350–355, Germany 2003.

[58] Z. Lu, B. Yin, and A. Jantsch, "Connection-oriented Multicasting in Wormhole-switched Networks on Chip," in proc. of IEEE Symposium on VLSI (ISVLSI), pp. 205-211, Germany 2006.

[59] N. E. Jerger, L. S. Peh, and M. H. Lipasti, "Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support," in proc. of International Symposium Computer Architecture (ISCA), pp. 229-240, China 2008.

[60] J. Duato, S. Yalamanchili, N. Lionel, "Interconnection networks: an engineering approach" (Morgan Kaufmann Publishers, 2003)

[61] W. J. Dally and B. Towles, "Principles and Practices of Interconnection Networks," Morgan Kaufmann, 2004.

[62] D. Bertsekas and R. Gallager, Data Networks, Prentice Hall, 1992.

[63] Intel Corporation, A touchstone delta system description, in: Intel Advanced Information, 1991.

[64] G. M. Chiu, "The odd-even turn model for adaptive routing," IEEE Trans. on Parallel and Distributed Systems, vol. 11, pp. 729 – 738, 2000.

[65] J.C. Hu, R. Marculescu, "DyAD–smart routing for networks-on-chip," in proc. of the Design Automation Conference (DAC), pp. 260–263, USA 2004.

[66] J. Kim, D. Park, T. Theocharides, N. Vijaykkrishnan, and C.R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in proc. of International Symposium on Computer Architecture, pp. 150–161, USA 2005.

[67] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in Proc. of Design Automation Conference, pp. 849–852, 2006.

[68] L.M. Ni, P.K. McKinley, "A survey of wormhole routing techniques in direct networks," IEEE Computer, Vol. 26, No. 2, pp. 62–76, 1993.

[69] P. Gratz, B. Grot, S.W. Keckler, "Regional congestion awareness of load balance in network-on-chips," in proc. International Symposium on High Performance Computer Architecture, pp. 203–214, February 2008.

[70] P. K. McKinley, H. Xu, E. Kalns, and L. M. Ni, "CompaSS: Efficient communication services for scalable architectures," in proc. of int. conf. supercomputing, pp. 478-487, USA 1992.

[71] H. Xu, P. K. McKinley, E. Kalns, and L. M. Ni, "Efficient implementation of barrier synchronization in wormhole-routed hypercube multicomputers," Journal of Parallel and Distributed Computing, Vol. 16, pp. 172-184, 1992.

[72] K. Li, and R. Schaefer, "A Hypercube Shared Virtual Memory," in proc. of int. conf. ICPP, pp. 125-132, USA 1989.

[73] M. Azevedo, and D. Blough, "Fault-Tolerant Clock Synchronization of Large Multicomputers via Multistep Interactive Convergence," in proc. of int. conf. ICDCS, pp. 249-257, Hong Kong 1996.

[74] X. Lin, and L. M. Ni, “Multicast Communication in Multicomputer Networks,” IEEE Transactions on Parallel and Distributed Systems, 4, pp. 1105-1117, 1993.

[75] P. McKinely, H. Xu, A. H. Esfahanian, and L. Ni, “Unicast-based multicast communication in wormhole-routed networks,” IEEE Trans. of Parallel and Distributed Systems, Vol. 5, pp. 1252-1265, 1994.

[76] M.P. Malumbres, J. Duato, and J. Torrellas, “An Efficient Implementation of Tree-Based Multicast Routing for Distributed Shared-Memory Multiprocessors,” in proc. of SPDP, pp. 186-190, USA, 1996.

[77] A. Al-Dubai, I. Romdhani, “A Performance Study of Path Based Multicast Communication Algorithms,” in proc. int. conf. PARELEC, Bialystok, Poland, pp. 245-250, 2006.

[78] R. V. Boppana, , S. Chalasani, C.S. Raghavendra, “Resource deadlock and performance of wormhole multicast routing algorithms,” IEEE Transactions on Parallel and Distributed Systems, 9, pp. 535-549, 1998,

[79] A. Al-Dubai, M. Ould-Khaoua, and L. M. Mackenzie “An Efficient Path-Based Multicast Algorithm for Mesh Networks,” in proc. of IPDPS, pp. 1—8, 2003.

[80] F. Harary, ‘Graph Theory’ (Readings, Massachusetts: Addson-Wesley, 1972)

[81] P. Mohapatra, V. Varavithya, “A Hardware Multicast Routing Algorithm for Two-Dimensional Meshes,” in proc. int. conf. SPDP, New Orleans, pp. 198-205, 1996.

[82] W.C. Tsai, K.C. Chu, S.J. Chen, and Y.H. Hu, “TM-FAR: Turn-Model based Fully Adaptive Routing for Networks on Chip,” in proc. of VLSI_SoC, pp. 19-24, 2010.

[83] J. Liang, S. Swaminathan, and R. Tessier, “aSOC: a scalable, single-chip communication architectures,” in proc. int. conf. PACT, Oregon, USA, pp. 37–46, 2000.

[84] E. Carara, and F. G. Moraes, “Deadlock-Free Multicast Routing Algorithm for Wormhole-Switched Mesh Networks-on-Chip,” in proc. int. conf. ISVLSI, France, pp. 341-346, 2008.

[85] D. K. Panda, S. Singal, and R. Kesavan, “Multidestination message passing in wormhole k-ary n-cube networks with base routing conformed paths,” IEEE Transactions on Parallel and Distributed Systems, 10, pp. 76–96, 1999.

[86] A. 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 proc. of DATE, pp. 423-428, 2009.

[87] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, “The splash-2 programs: Characterization and methodological considerations,” in Proceedings of the 22nd International Symposium on Computer Architecture (ISCA), pp. 24–36, 1995.

[88] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The parsec benchmark suite: characterization and architectural implications,” in Proceedings of the 17th international conference on Parallel architectures and compilation techniques, pp. 72–81, 2008.

[89] C. Bienia, S. Kumar, and K. Li, “Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chipmultiprocessors,” in IEEE International Symposium on Workload Characterization, pp. 47–56, 2008.

[90] M.M.K. Martin, et al. “Multifacet's general executiondriven multiprocessor simulator (GEMS) toolset,” SIGARCH Computer Architecture News, Vol. 33, No. 4, pp.92-99. November 2005.

[91] A. Patel and K. Ghose, "Energy-efficient mesi cache coherence with pro-active snoop filtering for multicore microprocessors," in Proceeding of the thirteenth international symposium on Low power electronics and design, pp. 247–252, August 2008.

[92] M. Martin, M. Hill, and D. Wood, "Token coherence: decoupling performance and correctness," in proceeding of 30th Annual International Symposium on Computer Architecture (ISCA), pp. 182-193, 2003.

[93] A. Demers, S. Keshav and S. Shenkar, "Analysis and Simulation of a Fair Queuing Algorithms," Proceedings of SIGCOMM '89, pp. 3-12, August 1989.

[94] C. A. Zeferino, M. E. Kreutz, and A. A. Susin, "RASoC: A router soft-core for Networks-on-Chip," Designers Forum - DATE, pp. 198-203, France, 2004.

[95] D. Wu, B. M. Al-Hashimi, and M. T. Schmitz, "Improving Routing Efficiency for Network-on-Chip through Contention-Aware Input Selection," In Proc. of 11th ASP-DAC, pp. 36 – 41, 2006.

[96] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the proximity congestion awareness in a network on chip," DATE, pp. 1126-1127, Germany, 2003.

[97] C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," Proc, Symp, Computer Architecture, pp. 278-287, May 1992.

[98] T. T. Ye, L. Benini, and G. De Micheli, "Packetization and routing analysis of on-chip multiprocessor networks," Journal of Systems Architecture, vol. 50, pp. 81-104, 2004.

[99] U. Feige, P. Raghavan, "Exact analysis of hot-potato routing," in Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, pp. 553–562, 1992.

[100] P. Abad, V. Puente and J. Á. Gregorio, "MRR: Enabling Fully Adaptive Multicast Routing for CMP Interconnection Networks," High Performance Computer Architecture (HPCA), pp. 355-366, 2009.

[101] X. Li, P.K. Mckinley, and L.M. Ni, "Deadlock-free multicast wormhole routing in 2-D mesh multicomputers", IEEE transactions on Parallel and Distributed Systems, v.5, i.8,pp. 793-804, 1994.

[102] P. Gupta, N. McKeown, "Designing and Implementing a Fast Crossbar Scheduler," IEEE Computer Society Press, pp 20-28, Jan. 1999.

[103] K. Lee, S. Lee, and H. Yoo, "A distributed crossbar switch scheduler for on-chip networks," in IEEE Int. Conf. on CICCS, pp. 671–674, Sept. 2003.

[104] D. C. Stephens, J. C.R. Bennet, and H. Zhang, "Implementing scheduling algorithms in high-speed networks," IEEE Journal on Selected Areas in Communications, vol. 17, pp. 1145-1158, June 1999.

[105] R. V. Boppana and S. Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms," Proc. Int'l. Symp. Computer Architecture, pp. 351–360, May 1993.

[106] E.B. Van der Tol and E.G.T. Jaspers, "Mapping of MPEG-4 Decoding on a Flexible Architecture Platform," Proc. SPIE 2002, pp. 1–13, Jan. 2002.

[107] X. Yang, Z. Qing-li, F. Fang-fa, Y. Ming-yan, L. Cheng, "NISAR: An AXI compliant on-chip NI architecture offering transaction reordering processing," in Proc. ASICON, pp. 890-893, 2007, Greece.

[108] W. Kwon, S. Yoo, S. Hong, B. Min, K. Choi, and S. Eo, "A Practical Approach of Memory Access Parallelization to Exploit Multiple Off-chip DDR Memories", Proc. DAC, 2008.

[109] A. Radulescu, and et al., "An Efficient On-Chip NI Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration", in Proc IEEE TCAD, 24(1), January 2005.

[110] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," In Proc. of ISCA'00, pp. 128-138, US, 2000.

[111] S. E. Lee, J. H. Bahn, Y. S. Yang, and N. Bagherzadeh, "A Generic Network Interface Architecture for a Networked Processor Array (NePA)", In proc. ARCS'08, pp. 247-260, 2008.

[112] W. Kwon, S. Yoo, J. Um, and S. Jeong, "In-network reorder buffer to improve overall NoC performance while resolving the in-order requirement problem", In proc. DATE'09, pp. 1058 – 1063, France, 2009.

[113] W. Jang and D. Z. Pan, "An SDRAM-Aware Router for Networks-on-Chip," in proc. of DAC'09, pp. 800-805, US, 2009.

[114] Z. Fang, X. H. Sun, Y. Chen, and S. Byna, "Core-aware memory access scheduling schemes," In Proc. of IEEE International Symposium on Parallel & Distributed Processing (IPDPS'09), pp. 1-12, Italy, 2009.

[115] H. G. Rotithor, R. B. Osborne, and N. Aboulenein, "Method and Apparatus for Out of Order Memory Scheduling," United States Patent 7127574, Intel Corporation, October 2006.

[116] Micron Technology, Inc. Micron 512Mb: x4, x8, x16 DDR2 SDRAM Datasheet, 2006.

[117] T. Ono, M. Greenstreet, "A modular synchronizing FIFO for NoCs," in proc. of the 3rd ACM/IEEE International Symposium on Networks-on-Chip, USA, pp. 224-233, 2009.

[118] C. E. Cumings, "Simulation and Synthesis Techniques for Asynchronous FIFO Design". In: SNUG, 2002.

[119] S. Murali, and et al. "Designing message-dependent deadlock free networks on chips for application-specific systems on chips," In Proc. VLSI-SoC, pages 158-163, 2006.

[120] T. Kgil, A. G. Saidi, N. L. Binkert, S. K. Reinhardt, K. Flautner, and T. N. Mudge, "PicoServer: Using 3D stacking technology to build energy efficient servers," in ACM Journal on Emerging Technologies in Computing Systems (JETC), Vol. 4, No. 4, 2008.

[121] G. L. Loi, B. Agarwal, N. Srivastava, S.-C. Lin, and T. Sherwood, "A Thermally-Aware Performance Analysis of Vertically Integrated (3-D) Processor-Memory Hierarchy," In Proceedings of the 43rd Design Automation Conf., pp.991-996, 2006.

[122] Gaisler IP Cores, http://www.gaisler.com/products/grlib/, 2009.

[123] R. Das, S. Eachampati, A. K. Mishra, N. Vijaykrishnan, and C. R Das, "Design and Evaluation of a Hierarchical On-Chip Interconnect for Next-Generation CMPs," in proc. of 15th International Symposium on High-Performance Computer Architecture (HPCA), pp. 175-186, 2009.

[124] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, "Performance Evaluation and Design Trade-offs for Network on Chip Interconnect Architectures," IEEE Transactions on Computers, vol. 54, no. 8, pp. 1025-1040, 2005.

[125] Young-Joon Lee, Yoon Jo Kim, Gang Huang, Muhannad Bakir, Yogendra Joshi, Andrei Fedorov, and Sung Kyu Lim, "Co-Design of Signal, Power, and Thermal Distribution Networks for 3D ICs," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 610-615, 2009.

[126] B. Goplen and S. Sapatnekar, "Thermal via placement in 3D ICs", Proc. ISPD 2005, pp. 167-174, USA.

[127] Z. Li, et al., "Efficient thermal-oriented 3D floorplanning and thermal via planning for two-stacked-die integration", ACM TODAES 11:2, pp. 325-345, 2006.

[128] M. Mondal, A. Ricketts, S. Kirolos, T. Ragheb, G. Link, V. Narayanan and Y. Massoud, "Thermally Robust Clocking Schemes for 3D Integrated Circuits," Proceedings of the IEEE Design Automation and Test in Europe (DATE), Nice, France, pp. 1-6, 2007.

[129] Q. Wu, K. Rose, J.-Q. Lu, and T. Zhang, "Impacts of though-DRAM vias in 3D processor-DRAM integrated systems," in IEEE Int' 3D System Integration Conf. (3DIC), pp. 1–6, 2009.

[130] R. Lu, A. Cao, and C. Koh, "SAMBA-Bus: A High Performance Bus Architecture for System- on-Chips", IEEE Transactions on VLSI Systems, Vol. 15, Issue 1, pp. 69-79, Jan 2007.

[131] M. Grange et al., "Physical mapping and performance study of a multi-clock 3-Dimensional Network-on-Chip mesh," in Proc. IEEE International Conference on 3D System Integration (3D IC), 2009, San Francisco, USA, pp. 1-7, 2009.

[132] T. Richardson, C. Nicopoulos, D. Park, V. Narayanan, Y. Xie, C. Das, and V. Degalahal, "A hybrid SoC interconnect with dynamic TDMA-based transaction-less buses and on-chip networks", In Proc. VLSID, pp. 8-15, 2006.

[133] V.F. Pavlidis and E.G. Friedman, "3D Topologies for Networks-on-Chip," IEEE Transactions on Very Large Scale Integration Systems, 15(10):1081, 2007.

[134] J. Shao and B. T. Davis, "A Burst Scheduling Access Reordering Mechanism", Proceedings of the 13th International Symposium on High-Performance Computer Architecture, pp. 285-294, 2007.

[135] D. E. Culler, J. P. Singh, and A. Gupta, "Parallel Computer Architecture: A Hardware/Software Approach," Morgan Kaufmann Press, 1998.

[136] I. Hur and C. Lin, "Memory scheduling for modern microprocessors," ACM Trans. on Computer Systems, vol. 25, no. 4, Dec. 2007.

[137] S. Murali, C. Seiculescu, L. Benini, and G. D. Micheli, "Synthesis of networks on chips for 3D systems on chips," In Proceedings of the 14th Asia and South Pacific Design Automation Conference (ASPDAC'09), pp. 242-247, Jan. 2009.

[138] Y. Qian, Z. Lu and W. Dou, "From 2D to 3D NoCs: A Case Study on Worst-Case Communication Performance," In. Proc. of the International Conference on Computer-Aided Design (ICCAD), pp. 555-562, 2009.

[139] A. Y. Weldezion, Z. Lu, R. Weerasekera, and H. Tenhunen, "3D Memory Organization and Performance Analysis for Multi-processor Network-On-Chip Architecture," in Proc. of IEEE International 3D System Integration Conference, San Francisco, USA, pp. 1-7, 2009.

Appendix A

Hardware Prototyping

A configurable on-chip network platform is developed with VHDL during this thesis work. Different hardware parts and their functionalities are already discussed in Chapters 3, 4, and 5. The platform exploits the adaptive router described in Chapter 4 where the router is configurable to use different routing protocols introduced in Chapter 3. It employs the network interface described in Chapter 5 to connect AXI-based processors to the network. Here, a lightweight prototype of the platform is presented. As illustrated in Fig. A-1, the prototype is composed of four routers each of which contains an AXI-based network interface. The agent cluster adjusts the frequency and voltage based on the congestion value.

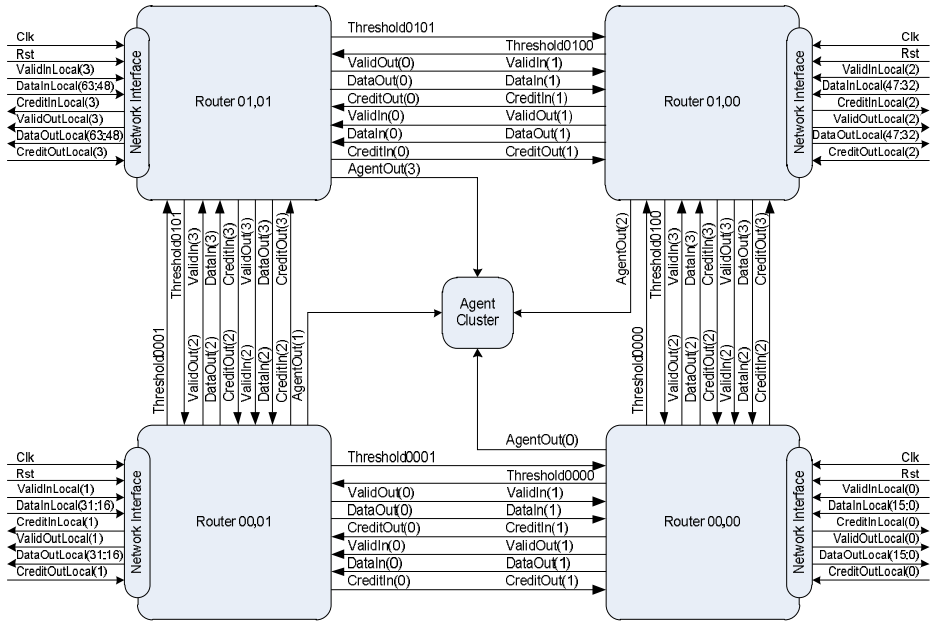


Fig. A-1. The test chip architecture.

The VHDL implementation of the platform is simulated using the ModelSim simulator from Mentor graphic and synthesized by Synopsys Design Compiler. Different tools from Synopsys (for frontend) and Cadence (for backend), e.g. Synopsys Design Compiler, Synopsys PrimePower, Cadence Encounter, and Cadence Virtuoso, are used for hardware analysis and layout. A test chip from the lightweight platform (Fig. A-1) was fabricated using the ST 65nm CMOS technology with Low Power Low Voltage Standard Cell Library at 1 V supply voltage. The hardware layout of the platform is shown in Fig. A-2. The $0.07mm^2$ design contains 5520 cells along with 146 IOs with chip maximum operating frequency of 2 GHz. Finally, Fig. A-3 depicts the targeted prototype board. The platform is supposed to connect four AXI-based processors while voltage and frequency are adapted based on the agent decision integrated in the network.

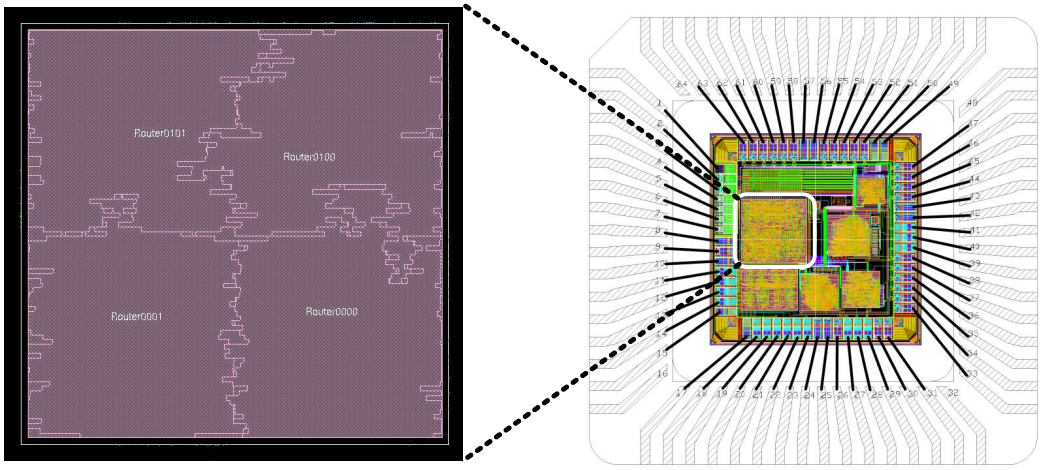


Fig. A-2. The test chip layout.

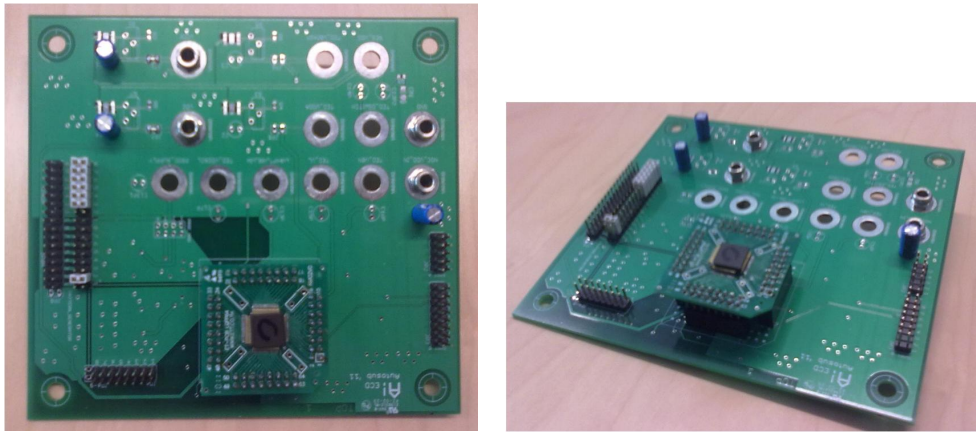


Fig. A-3. The prototype board.