POLITECNICO DI TORINO
Master in Electronics Engineering
Department of Electronics and Telecommunications

POLITECNICO
DI TORINO

A Flow Control Mechanism for Fully Adaptive Routing Algorithms in On-Chip Networks

Supervisor: Prof. Masera GUIDO
Supervisor: Prof. William Fornaciari
Assistant Supervisor: Dott. Davide Zoni

Master Thesis of:
Tamer Ahmed Eltaras
Student n.s213732

Academic Year 2017-2018

Abstract

Routing algorithms for networks-on-chip (NoCs) typically running with a limited number of virtual channels(VCs). The design of fully adaptive routing algorithms faces several challenges due to these limited VCs. fully adaptive routing algorithms based on previous deadlock-avoidance theories require one of two schemes. First one a conservative VC re-allocation scheme which allows a VC to be re-allocated only when it is empty, but it limits the performance. Second one an updated scheme based on the previous one allows a non-empty VC to be re-allocated to a new packet if the VC had received the tail flit of the previous packet and the VC has enough space for the whole new packet, which enhances the performance. We propose a partial packet restoring (PPR) mechanism to allow the long packet to be restored to the adaptive path. With our proposal, we split the long packet into portions resulting in decreasing the packet length. By decreasing the packet length the buffer space needed for the packet to be fully accommodated in the channel becomes less, which gives a better VC utilization. Our proposal target longe packets especially when we need to restore them from escape paths to adaptive paths in that way we also, increase the network adaptivity. We prove that our mechanism does not induce deadlock. We design a novel fully adaptive routing algorithm which maintains packet adaptivity. Compared with conservative VC re-allocation, our proposal achieves an average 30% saturation throughput improvement in synthetic traffic patterns.

Acknowledgments

Foremost, I would like to thank the almighty God for guiding and blessings me all my journey of life.
To my family, my real srtrength. nothing will be sufficient to thank my parents, who have gave me the opportunites to do everything I have wanted,who have always made me feel a priviliged person and who have gave me a good life.
My deepest gratitude is to my supervisor Professor William Fornaciari who gave me the chance to write my thesis in politecnico di milano.To Davide,for his patience and his precious advices realy i learn alot from him.To all people of HEAPlab research group,for their kindness and thier continuous availability in helping me with useful advices.
My sincere thanks also goes to my advisor Professor Masera Guido, I am deeply grateful to him for encouraging me to do my thesis in politecnico di milano.
Many friends have helped me during this journey, economically and mentally. Their support helped me to overcome difficulties and stay focused all the years. I deeply appreciate their help and greatly value their friendship.
I would like to express my heart-felt gratitude to all.

Tamer Ahmed Eltaras

Contents

List of Figures

List of Tables

Acronyms

BW = Buffer Write
CMP = Chip Multi-Processor
LT = Link Traversal
LA = Link Allocation
NoC = Network-on-Chip
RC = Route Computation
SA = Switch Allocation
SaF = Store and Forward
ST = Switch Traversal
VA = Virtual-Channel Allocation
VC = Virtual Channel
VN = Virtual Network
PSF= Port-Selcetion-firts
WPF= Whole Packet Forwarding
PPR= Partial Packet Restoring
NIC= Network Interface Controller

Chapter 1

Introduction

1.1 The evolution to multi-cores

During the past several decades, the performance of microprocessors that power modern computers has continued to increase exponentially. As the transistors that are the heart of the circuits in all processors and memory chips have simply become faster over time on a course described by Moore's law [1]. Advanced chip fabrication technology and integrated circuit processing technology provide increasing integration density which has made it possible to integrate one billion transistors on a chip to improve performance as whon in Figure 1.1 [2]. Moreover, actual processor performance has increased faster than moore's law would predict. One of the most tradational techniques to improve performance is to increase the chip frequency which enables the processor to execute the programs in a much quicker time. Other methods for increasing processor performance are pipelining and instruction level parallelism (ILP). Pipelining divides the execution of an instruction into smaller parts which are allowed to run faster. ILP overlaps execution of several instructions into the same clock cycle by fully utilizing all the functional units. The combination of pipelining, ILP and faster clock rate taken together, provide significant performance improvements in uniprocessor designs.

Unfortunately, in recent years, it is becoming increasingly difficult for processor designers to continue using traditional approaches to enhance the speed of modern processors. Figure 1.2 shows the performance of an individual processor over the time period from 1978 to 2006 [3].

This is partially caused by the fact that shrinking transistor size has

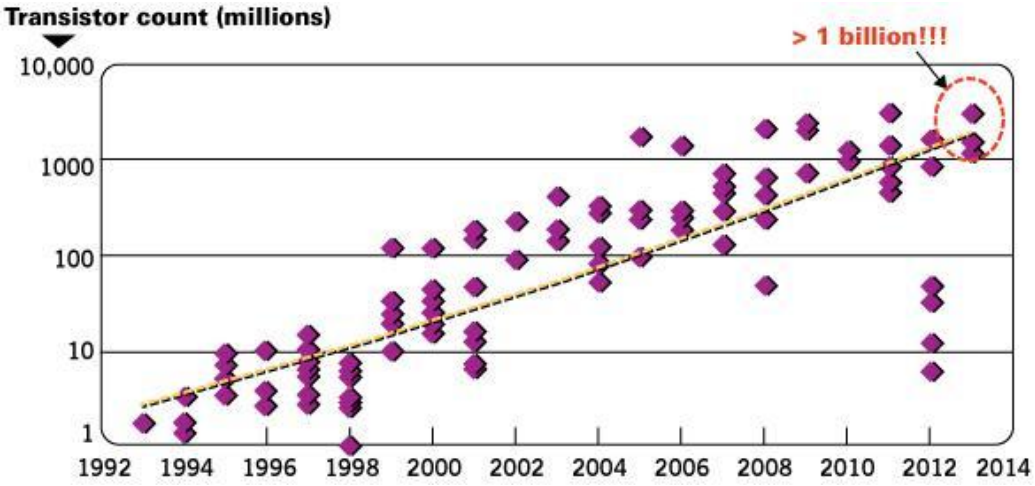


Figure 1.1: Evolution of transistor integration on a chip

increased the leakage current, crosstalk, clock skew, preventing the processor from working at higher operating frequencies. Power dissipation has become another critical factor contributing to the slower improvement rate in processor speeds. The limitations seen in uniprocessor designs and by other important considerations such chip fabrication costs, fault tolerance, power efficiency and heat dissipation led to a new era in computer architecture, where engineers switched from designing more sophisticated uniprocessors to exploring the benefits of placing multiple simpler cores on the same chip. Chip-multiprocessors (CMPs) provide a power-efficient and cost-efficient method by filling up a processor die with multiple, relatively simpler processor cores instead of just one huge core.In addition, parallel code execution, obtained by spreading multiple threads of execution across the various cores, can achieve significantly higher performance than would be possible using only a single core. Also, replicating cores on a processor is a known method for increasing processor performance within the allowed power budget of modern processors.

1.2 The evolution to on-chip networks

The future performance gains depend on solving the communication bottleneck between the processors and the memory components. As the number of on-chip cores increases the need for a high bandwidth and low-latency

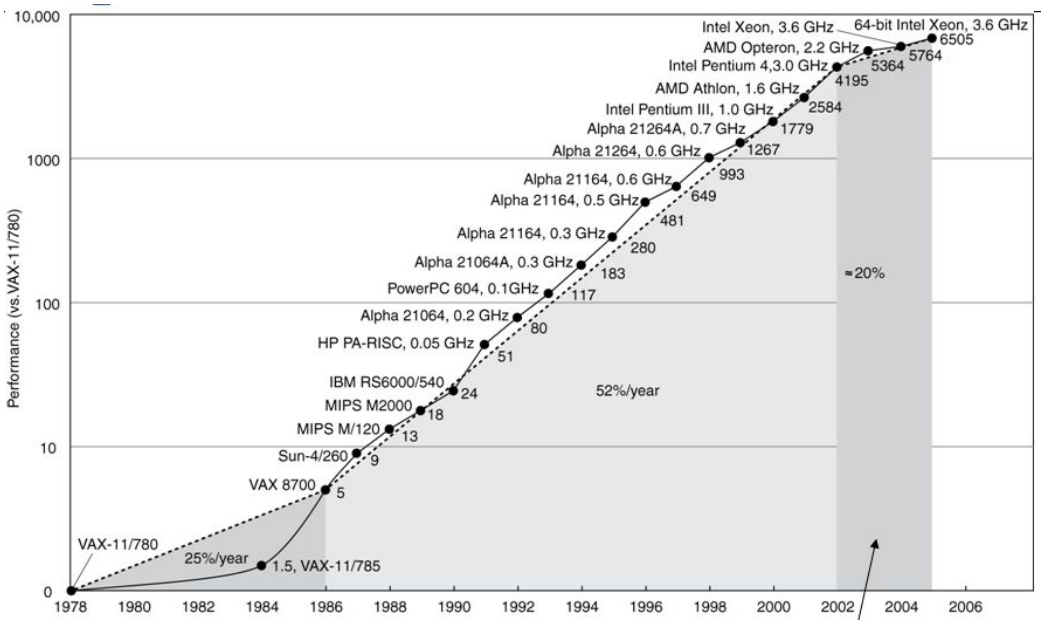


Figure 1.2: The trend of uniprocessor performance improvement since 1978

communication interconnect to connect IP cores becomes critically important.The first generation of on-chip interconnects are buses and crossbars and those are the dominant interconnect up to eight cores.

- *Buses*: A bus is the simplest of all interconnections [4]. A single link is shared between all the components of the CMP: when two components initiate a transmission, every other component has to stop using the bus at the possibility of interfering with the in-progress transmission. Therefore, buses usually have arbiters that ensure that there is only one owner of the bus at a time called the bus master. An alternative, carrier sense/collision detection schemes are utilized where a component checks if the bus is currently in use before using it, retrying after a random amount of time if it is indeed occupied. The width of the link determines the number of bits that can traverse the bus simultaneously, and therefore, determines the bandwidth of the bus. Buses are very simple to implement, and for a low number of components, extremely cheap. Additionally, they have the interesting property that every transmission can be read by every component in the bus, which makes it interesting for certain use cases (e.g. cache coherence). As the number of components increases the disadvantages of buses become

especially noticeable:

- *Long link sizes*: a single link has to traverse each and every component in the CMP. Such a long link might require too much power to be reasonable, require lots of repeaters to maintain the adequate signal level for all segments of the bus.
- *Bandwidth*: is shared between all components in the network, and limited by the link width and clock frequency.
- *Long access times*: messages written to a bus might take a significant time to propagate over the entire link. For adequate operation, no other message can be written to the bus until a certain time has been given for such propagation.

An extension to the bus concept, the multibus, is composed of multiple parallel buses that can operate simultaneously and independently. Such an extension obviously has increased bandwidth but also increases the cost linearly as the number of buses grows.

- *Crossbars* In a crossbar, there exists a link between each pair of components. It is a peer- to-peer link. As these links are independent, a component that wants to initiate a transaction can do so at any time as long as it does not already have an active transaction with that same component. Every two components have the maximum link bandwidth between them as the link is private. A crossbar might be seen as the complete opposite to a bus: instead of a single shared link, a crossbar has no shared links at all. The disadvantages of crossbars are the while they are extremely scalable, the number of links required increases quadratically with the number of components. Therefore, they become very expensive. It is for this reason that, while crossbars have been used in commercial CMP designs, it is hard to see designs with more than 32 components [5].

As a consequence, the need of an interconnect that able to supply a reasonable bandwidth at a low area and low power overheads open the way to "On-chip networks" to be an attractive alternative to crossbars and busses. For more details about On-chip networks, the background is provided in Chapter 2.

1.3 Problem overview and proposal

Networks-on-chip (NoCs) have been proposed as an attractive alternative solution to meet the communication requirements for Multi-core computing systems [41]. The performance of the Noc is sensitive to the routing algorithm used to route the packets in the network, as the routing algorithm defines the latency of the packet transmission and also the throughput of a NoC. Several routing algorithms have been proposed to achieve high performance in NoCs [42] [44] [45] [46]. Any proposed routing algorithm has to be deadlock free, at both the network and protocol level. The need of protocol-level deadlock freedom is achieved by using multiple virtual networks for different protocol messages classes. For the network-level deadlock freedom, deadlock-avoidance theories have been proposed to solve it. There are are several theories for deadlock-free partially adaptive [17] [20] [42] and fully adaptive [25] [26] [10] [49] routing algorithm design. Another issue is that the buffer resources are limited due to the tight area and power budgets [40] [43]. Thus, a NoC is running with a limited number of VCs. In a wormhole network, the design of the fully adaptive routing algorithms based on existing theories require one of two schemes. First one a conservative VC re-allocation scheme which allows a VC to be re-allocated only when it is empty, but it limits the performance. Second one an updated scheme based on the previous one allows a non-empty VC to be re-allocated to a new packet if the VC had received the tail flit of the previous packet and the VC has enough space for the whole new packet, which enhances the performance. the later scheme targets the short packets which are single flit. we propose a novel that targets the long packets and gives better virtual channel utilization.

1.4 Our proposal

The thesis proposes a partial packet restoring mechanism (PPR), which allows decreasing the long packet length. By decreasing the packet length the buffer space needed for the packet to be fully accommodated in the channel becomes less, which gives a better VC utilization. Our proposal target longe packets when we need to restore them from escape paths to adaptive paths. In that way, we also, increase the network adaptivity. Our novel mechanism (PPR) achieves high VC utilization and maximal routing flexibility. Compared with previous VC re-allocation schemes. we make the following

primary contributions:

- *P*roposes Partial packet restoring mechanism, which improves the performance of fully adaptive routing algorithms, especially for long packets.
- *P*roves that a fully adaptive routing algorithm using PPR mechanism is deadlock-free.
- *P*roposes an efficient fully adaptive routing algorithm that takes advantage of the novel mechanism.

1.5 Structure of the thesis

The rest of the thesis is organized in 5 chapters. Chapter 2 overviews the background of the work and the state of the art. Chapter 3 provides a detailed description of our architecture and its novel contributions to the State of the Art. Chapter 4 details the methodology validation providing results with synthetic traffic and real applications. Chapter 5 points out conclusions and some future works.

Chapter 2

Background and State of the Art

2.1 Background

Network-on-Chip(NoC) has recently emerged as an attractive alternative interconnection for chip mulitprocessors. As the traditionally used interconnects have several problems as we have seen in chpater1 1. NoC is a scalabel and reliable interconnect that allowes nodes to exchage data. A typical NoC is composed of the following components: nodes,routers, links and Network Interface controllers(NICs) as shown in Figure 2.1.

- *Nodes* - The components that are participating in the network where data are exchanged between. The node can be a core or a part of memory substem.

- *Router* - The component that manages the routing of data and coherence packets into the network, Usually the presence of routers what distinguishes a NoC from a traditional interconnect. It basically receives packets from the shared links and according to the address informed in each packet, it forwards the packet to another shared link or to the Network interface controller attached to it.

- *Network Interface Controllers(NICs)* - Is the component that allows the communication between a node and a router. It makes the connection between the IP cores and the routers, converting cache messages into

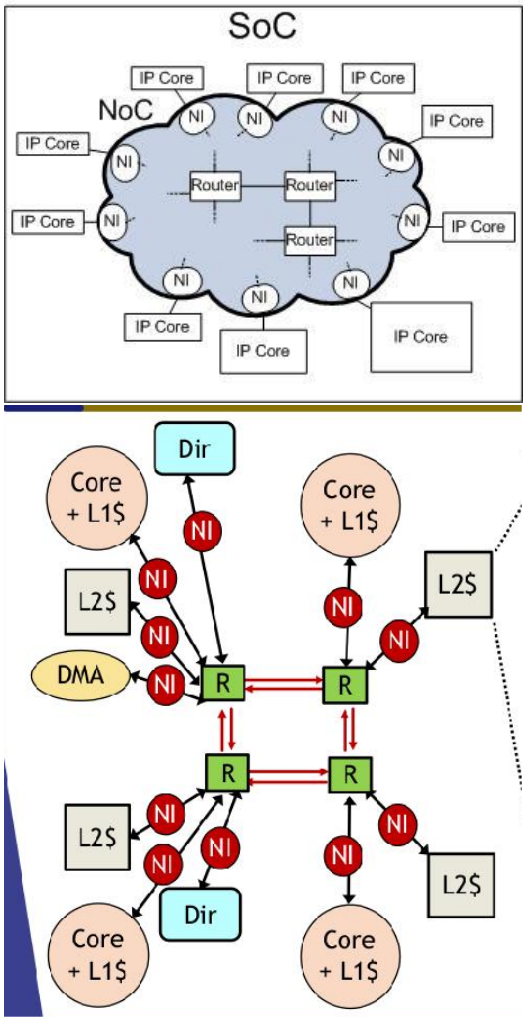


Figure 2.1

packets and injected them into the network and receiving the packets from the network and converts them into a message.

- *Links* - They are the physical wires.In NoC links are always direct, peer-to-peer connections between either two routers or a router and a terminal node. A link might be composed of more than one wire. The number of parallel wires determines the bandwidth of the link. On the other hand, the latency of a link is the time required for a bit of information to traverse it and is usually dependent on the length of the link.

Four parameters characterize a NoC: its topology, the routing algorithm, the flow control algorithm, and the router microarchitecture.

2.2 Topology

The topology of the on-chip network defines the way channels and nodes are interconnected. It determines the physical layout of the network. The choice of the topology is almost the first decision designers have to make when building an on-chip network. It has a significant impact on network cost-performance. Firts, for the cost, it determines the implementation complexity in sence of number of links at each node, number of ports at each router and the ease of laying out a topology on a chip. Second, for the performance, the topology determines the number of hops a message has to traverse to reach its destination and the lengths between hops, thus affecting network latency. Also, the number of hops affect the network energy consumption in sence that the traversing routers and links consume energy. Furthermore, the topology determines the total number of alternate paths between nodes affecting how the network can spread out the traffic which affects the maximum throughput of the network. There are several metrics used to evaluate the topology. These metrics can be divided into two types:

- *design-time metrics*
  - *Degree*: It refers to the number of the links at each node. A higher degree requires more ports at routers, which increases the complexity of implementation.
  - *Bisection bandwidth*: It is a bandwidth across a cut that partitions the network into two equal parts. This bandwidth is useful in defining the worst-case performance of the network.
  - *Diameter*: Is the maximum distance between any two nodes in the topology
- *run-time metrics*
  - *Hop count*: The number of hops a message takes from source to destination. The maximum hop count is given by the diameter of

the network.

– *Maximum channel load*: It estimates the maximum bandwidth the network can support.

$$MaximumInjectionBandwidth = 1/MaximumChannelLoad$$

– *Path diversity*: It represents the ability of the topology to provide multiple shortest paths between a given source and destination pair. Path diversity within the topology gives the routing algorithm more flexibility.

Topologies can be classified into these types:

- *Direct topologies*: Are those where each node terminal is associated with a router. All routers act as both sinks/sources of traffic and as switches for traffic from other nodes: mesh, ring, and tour.

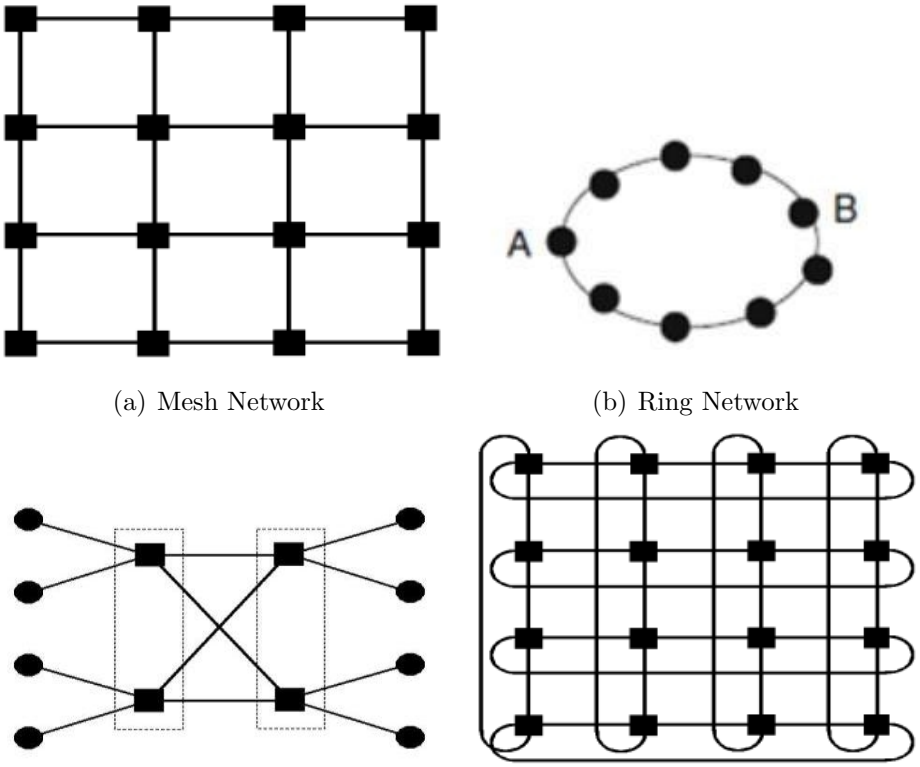
- *Indirect topologies*: They connect terminal nodes via one or more intermediate stages of switch nodes. the terminal node only acts as sources and destination of the traffic, intermediate nodes switched traffic to and from terminal nodes.

- *Hierarchical topologies*: Multiple nodes are clustered together in one topology, and these clusters connected together via another topology, building a hierarchical design.

Here we give examples of the topologies that usually used in NoCs:

- *Mesh*: Mesh topology is mostly used on in on-chip networks due to its layout efficiency. It has good electrical property. A mesh-shaped network consists of m columns and n rows. Figure2.2(a) show howt the network organized using mesh topology. the two wires.

- *Torus*: A simple torus network is a mesh in which the heads of the columns are connected to the tails of the columns and the left sides of the rows are connected to the right sides of the rows. The path diversity of torus is better than the mesh network, and it also has more minimal routes. Figure2.2(d) show howt the network organized using mesh topology.



(a) Mesh Network (b) Ring Network

(c) Butterfly Network (d) Torus Network

Figure 2.2: Noc Topologies

- *Butterfly*: A butterfly network is a uni- or bidirectional and butterfly-shaped network typically uses a deterministic routing. Packets arriving at the inputs on the left side of the network are routed to the correct output on the right side of the network. Figure2.2(c) show howt the network organized using mesh topology.

This section summarizes the state of the art that is of interest in this thesis. The literature is huge and here we present the related work in deadlock avoidance theories and designs for fully adaptive routing.

2.2.1 Deadlock Avoidance Theories

As second generation multi-computers use wormhole flow control [19, 22, 34, 39] , the focus becomes on theories for wormhole networks.Deadlock-free routing strategies have been developed, allowing the implementation

of fast hardware routers that reduce the communication bottleneck. Designing deadlock-free routing algorithms for wormhole routing was simplified by Dally and Seitz [21] proposing a necessary and sufficient condition for deadlock-free routing. This condition states that a routing function is deadlock-free if and only if its channel dependency graph is acyclic. Also, adaptive routing algorithms with deadlock-avoidance or deadlock-recovery strategies have been proposed to be an effective strategy. Duato was the first to propose theorems for deadlock-free routing in adaptive networks [25, 26]. He proved that a cyclic dependency is not a sufficient condition to create a deadlock As alternative paths could be used to escape deadlock situations. Duatos theorem holds for a large class of wormhole networks. Beside his theorem for deadlock-free, he provided the corresponding design methodology. Loren Schwiebert and D. N. Jayasimha [49] proposed a necessary and sufficient condition that can be used for any adaptive or nonadaptive routing algorithm for wormhole routing, as long as the only local information is required for routing. The underlying proof technique introduces a new type of dependency graph, which removes most channel dependencies that cannot be used to create a deadlock. X.Lin and P.Mckinely [32]introduced a new approach to deadlock-free routing in wormhole-routed networks called the message flow model. This method can be used to design deterministic, partially-adaptive, and fully-adaptive routing algorithms. S.Tektak and E.Encrenaz [51]proposed a new sufficient and necessary condition associated with a polynomial algorithm to check if a given network is deadlock-free. The proposed algorithm has been implemented in a tool for automatic detection of deadlocks in wormhole networks. F.vebeek and j.Schmaltz [52] presented a novel algorithm for the automatic verification that a routing function is deadlock-free in wormhole networks. Verbeek and Schmaltz [53, 54] gave the first static necessary and sufficient condition for deadlock-free routing. There are several partially adaptive routing algorithms based on turn models [17], [11], [12]. However, partially adaptive routing algorithms complain of limited adaptivity as packets cannot use all minimal paths between the source and destination. fully adaptive routing can be designed with these theories [25], [26], [10], [32], [49], [51], [52], [54]. This theories of fully adaptive guarantee that all blocked packets can reach VC heads to access deadlock-free paths as they only allow only empty VCs to be re-allocated. Some deadlock-recovery theories [15], [7] have been proposed to remove the limitation of performance when only empty VCs allowed to be re-allocated as they allow the formation

of deadlock, and then invoke some recovery mechanisms.

2.2.2 Fully Adaptive Routing Algorithms

Duatos theory [12, 13] is widely used in the design of fully adaptive routing algorithms. Duato classified VCs into two types: escape and adaptive. When deadlock among the adaptive VCs occurs, Packets must have the chance to escape through a deadlock-free set of VCs, Which defined as escape VCs. Escape VCs are always deadlock-free by applying a restrictive routing algorithm which ensures that the channel dependency graph is acyclic. Many algorithms based on Duatos theory [20, 27, 31,43] select the physical port first. Once selecting a port, packets can only request VCs of this chosen port.This requirement imposes a limitation on these algorithms as once a packet enters an escape VC, it can only use escape VCs until it is delivered. Otherwise, the escape VC may be involved in a deadlock. This limitation causes adaptivity loss. Duatos theory supports the design of algorithms which can use an adaptive VC after using an escape VC if packets are always able to reach the head of the channel.

Chapter 3

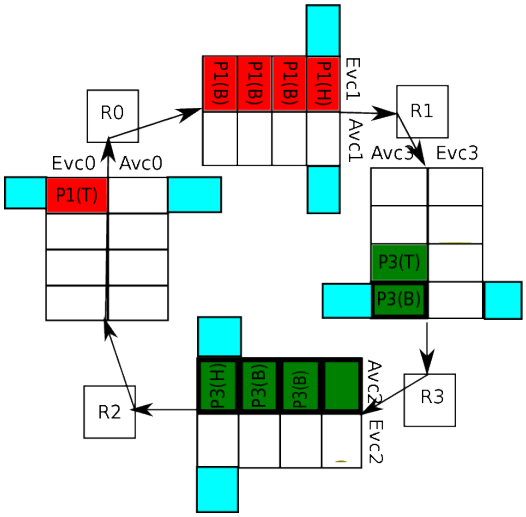
Proposed methodology

In this chapter we present our partial packet restoring mechanism scheme and prove it is deadlock-free. Next, we design a routing algorithm which maintains packet adaptivity using this scheme. Finally, we describe in details how we implemented our proposal.

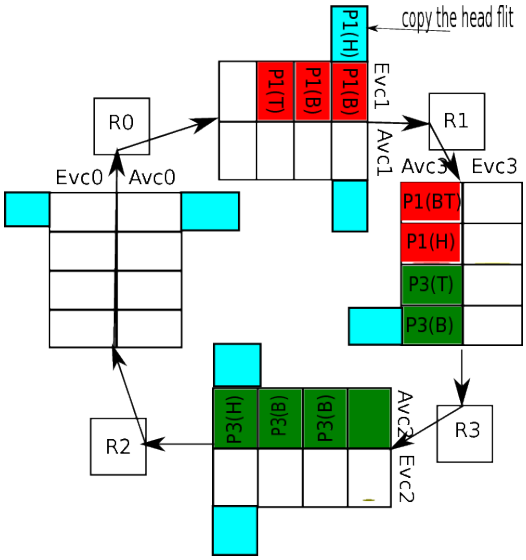
3.1 Partial packet restoring

As described before Existing fully adaptive routing algorithms use conservative VC re-allocation to prevent deadlock. However, this results in poor VC utilization. Therefore, we propose a novel VC re-allocation scheme: partial packet restoring, which improves VC utilization and it is a deadlock free. Suppose a packet P_m with length(m) resides in the upstream escape virtual channel EVC_i, and the escape virtual channel EVC_j and the adaptive AVC_j are the downstream channel. Assume that the routing algorithm allows packet P_m to forward to downstream router. With conservative VC re-allocation scheme the downstream virtual channels can be re-allocated to P_m only if they are empty [7]. With whole packet forwarding (WPF) scheme we can re-allocated a non-empty virtual channel if the channel is already received the tail of the currently allocated packet and there are enough spaces to accommodate the incoming packet totaly [6]. In both previous scheme we can't restore the long packet P_m to downstream adaptive virtual channel AVC_j and the packet has to continue in the escape path using EVC_j which results of losing the adaptivity of the packet. For our proposed VC re-allocation scheme, AVC_j can be re-allocated to P_m If AVC_j is empty or, has some free

spaces and AVC_j holds the tail flit of the currently allocated packet. We can split the long packet P_m according to the number of free slots in AVC_j which has to be greater than one to make it significant. After splitting the packet, we can restore the portion of the packet which its length becomes equal to free slots in AVC_j. We call this scheme partial packet restoring (PPR). Figure 3.1 shows a PPR example. Here, in the upper Figure 3.1(a) the packet P1 with 5 flits long is resides in escape path and the routing algorithm allows the packet to use the channels in router R1. As the length of the packet p1 is greater than the channel depth, so with traditional schemes, the packet will continue in escape path. In router R1 the adaptive channel Avc3 has two free buffer slots and already received the tail flit of the currently allocated packet. In order to restore the packet P1 to the adaptive path we split it into two halves, first half is 2 flits long which is equal to the number of free buffer slots and the second half is 4 which will be now equal to the number of the channel depth. The lower Figure 3.1(b) is the result of applying PPR mechanism. PRR mechanism solves the shortcoming of conservative VC re-allocation and WPF in restoring long packets to the adaptive path



(a) config0



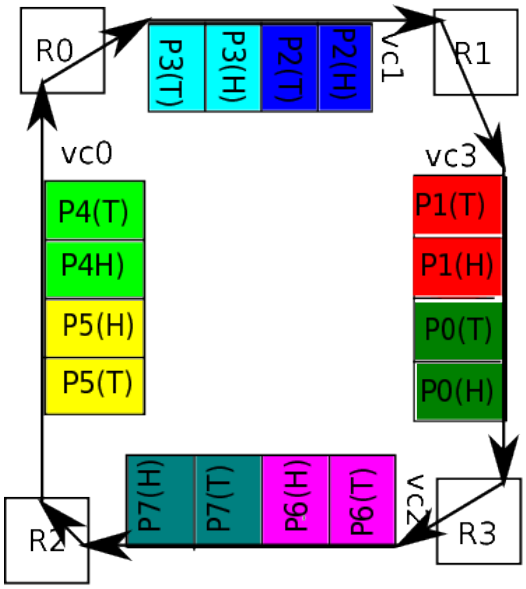
(b) partial packet restoring

Figure 3.1: example of partial packet restoring

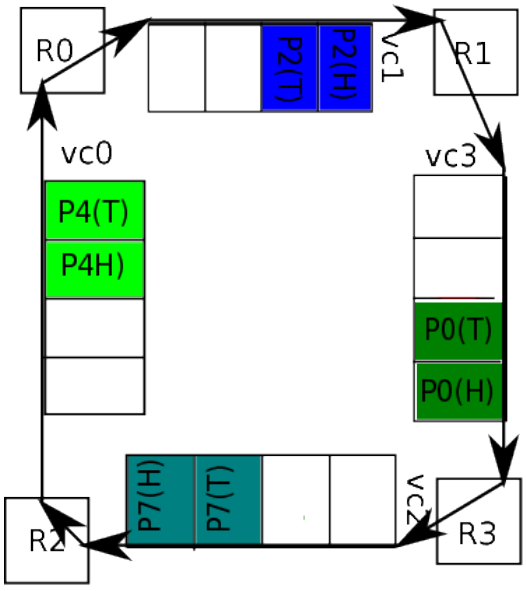
3.2 Proof dead-lock freedomness

Our strategy is that if the routing algorithm with conservative VC re-allocation is deadlock-free, then applying PPR will not cause any deadlock. As when we splitting the packet we split in the way that the split portion will be fully accommodated in the downstream channel. So the packet will either be at

the head of the VC or behind some packets whose head flits are at VC heads. Thus the packets can access the deadlock-free path. As after applying partial packet restoring the flow control of the packet will follow the same approach of whole packet forwarding, we use the same proof to verify that PPR is dead-lock free [6]. we built the proof in an inverse way, we prove that if the scheme after applying the PPR has a deadlock configuration then the scheme before applying the PPR has a deadlock configuration as well. Using dead-lock configuration in Figure 3.2(a) as an example, we remove packets whose head flits are not in the head of the VC and get a new configuration Figure 3.2(b). The configuration in the first figure represents the possible configuration that may appear after applying PPR. The configuration in the second figure represents the scheme before applying PPR that using VC conservative scheme. The configuration in the second figure is has a deadlock. However, the VC conservative scheme is a deadlock free, thus there is no such configuration.



(a) deadlock-1



(b) deadlock-2

Figure 3.2: proof of deadlock-freedomess

3.3 Fully adaptive routing algorithm with novel mechanism

Fully adaptive routing algorithms in VC-limited networks yield poor VC utilization which affects the performance. To solve this problem, we leverage to design a novel fully adaptive routing algorithm. Our design is based on Duatos theory [7]. In a NoC where limited VCs are used, the routing algorithm has to provide maximum routing flexibility. It should allow the use of adaptive VCs after using escape VCs for all packets (short and long). Otherwise, once a packet goes into escape VCs, it loses its adaptivity. The design must guarantee that at any time a packet is able to request an escape VC [25]. we made a simple modification on in port-selection-first algorithms. In PSF algorithm the packet can only use the channels in the selected port. As our mechanism allowing long packets to be restored in the adaptive path after using escape path, we can use also the escape channels of non-selected the port without worrying about not to restore long packets as long as the selection doesn't violate DOR. The packet can apply for all the channels in selected one and if it is failed to be allocated, it can apply for all adaptive channels in other port and all escape channels with a condition not to violates DOR in escape path.

3.4 Design

In this section, we will explain in details how we implemented our proposal. Our baseline architecture is Garnet2.0, a detailed on-chip network model inside a gem5 simulator [58]. Inside garnet2.0 we have the following components for each component we will show the modification we added to implement our proposal.

- **VirtualChannel**:
  - *new type*: In baseline implementation, there is one type of virtual channel. To implement the fully adaptive algorithm and our proposal, We add a new type of virtual channels, escape virtual channels (EVCs). Now we can classify the virtual channels as, adaptive virtual channels and escape virtual channels.

- *new state*: In virtual channel class we have states to indicate the state of the channel. In baseline implementation, we have two states, active and idle. we add one state more, reusable state. reusable state indicates that the tail flit of the packet currently allocated in the channel has been received and we can use the channel for new re-allocation.
- *new buffer slot*: We added a new buffer slot for each virtual channel. As when we split the packet, We copy the head flit of the split packet in this slot before it leaves the channel. We use this slot as additional flit to guide the second portion the split packet.
- *split flag and split counter*: We add a flag to indicate that we split the packet currently resides in the virtual channel. to organize the flits. We use the split counter to count the number of flits ejected from the virtual channel after the split. When the counter becomes equal to the length of the first portion of the split packet, it means that the first portion left the virtual channel and we have to change the last flit of the first portion as tail flit. moreover, when the first potion left we have to read the next flit from the new buffer slot to become the head flit of next portion of the packet.
- *change path flage*: As we add escape virtual channels, so we have the possibility to change from escape to adaptive and vice-versa. Change path flag indicates when we want to change the path.

- **Routing unit**:
  - *New Routing function*: In baseline implementation, there is a routing function which uses deterministic XY routing. We implemented a new routing function besides the original one. We used the original function to route packets in escape path. For the adaptive path, we use the new function that selects the out-port depending on the number of free virtual channels. If both ports have the same number of free channels, It randomly selects one of them. moreover, the new function provides the two ports, selected port and the other one. As our proposal give the possibility to use the channels from the non-selceted port.

- **Input unit**:

- *Delay stages*: In baseline architecture, when the new flit is received and inserted in the corresponding channel, it immediately goes to route computation stage. As in our proposal, more than one packet can reside in one channel, we have to delay the route computation stage until the flit reaches the head of VC.
- *New functions*: The input unit embedded the virtual channels inside it. We add some features in virtual channel class. We added in input unit some functions to support these features. These functions are read split flag, set the split flag, read split counter, increment split counter, reset the split counter, and route compute function which used to delay the route computation stage.

- **Output unit**:
  - *New function for EVC*: In baseline architecture, there are functions to check if the downstream has free VCs and make a selection. As we added another type of virtual channel (EVCs), we have to add their corresponding functions. we added functions for the escape path beside the original functions that used for the adaptive path.
  - *Num of free channels*: We add a new function to get the number of free channels in the downstream router. We use this function in routing computing unit, as we added a new routing function which makes a selection depend on num of free channels.
  - *Num of free buffer slot*: We add a new function to get the number of free buffer slots in the downstream channel. We use this function in our proposal as we depend on the number of free buffer slots in the channel to make the split.

- **SwitchAllocator**:
  - *Resotre function*: We added this function to restore packet from escape path to adaptive path. When the packet is in the escape path and tries to re-allocate the channel in the downstream router it calls the restore function. The restore function checks if the packet is a short or long packet. If it is short it checks if there is a reusable or idle adaptive virtual channel with one free slot in the downstream router, if there is available VC, the restore function

return true and set the change-path-flag. If the packet is long, the restore function checks if there is a reusable or idle adaptive virtual channel in the downstream adaptive virtual channels has free slots buffer to accommodate the portion of the packet. If we have space the restore function return true and set split-flag.

- *Modifying arbitrate inport function*: In this function, we have to check if the split flag is set or not. If the split flag is set and the split counter equal to the number the first portion of the split packet, then we have to peek the top flit from the slot buffer we already added instead from the virtual channel.
- *Modifying arbitrates outport function*: In this function, the actual switch allocation occurs and the flit leaves the channel. In this function, we have to check the split flag. If the split flag is set we have to shape the split packet, when its flits are leaving the channel.

Chapter 4

Experimental results

In this chapter, we present the assessment of the proposed Partial Packet Restoring(PPR) mechanism, from the performance viewpoint. It will be shown its behavior with a synthetic traffic, discussing for each pattern how the applying of the proposed mechanism improves the performance of the network. Besides the synthetic traffic, we evaluate our proposal with real applications using a subset of the Splash-2x benchmark suite.

The rest of the chapter is organized as follows. Section 4.1 describes the simulation setup, the target architecture, synthetic traffic patterns and benchmarks used.Section 4.2 describes the obtained results with both synthetic traffic and real application. Finally, Section 4.3 describes some design-space exploration, showing how the behavior of the patterns change with different percentage of long packets.

4.1 Simulation platform and setup

For computer-system architecture research, the gem5 simulator provides a modular platform, encompassing system-level architectures and microarchitectures. The novel mechanism has been integrated into the enhanced version of the gem5 using Garnet2.0: A detailed on-chip network model inside gem5 simulator [57]. In our evaluation, we use both synthetic traffic and real applications.

- **Synthetic traffic**
  In the gem5 simulator, there is a garnet synthetic traffic injector. Garnet synthetic traffic provides a framework for simulating the Garnet

network with controlled inputs. Gem5 has a dummy cache coherence protocol to work in a proper way with the Garnet synthetic traffic.

Different synthetic traffic patterns have been used for evaluating NoCs. In this thesis, we used the following synthetic traffic patterns to evaluate our mechanism: uniform-random, tornado, bit-reverse, bit-rotation, neighbor, shuffle,transpose-1, and transpose-2. For each pattern we evaluate the network with different values of injection bandwidth. Here we give a brief describtion about how these patterns work.

- *Uniform-random*: Each node in the network sends messages to other nodes with an equal probability, the destination nodes are chosen randomly.
- *Bit-reverse*:Each node sends to a distination whose address is bit reversal of the source address.
- *Bit-rotation*:Each node sends to a distination whose address is bit rotation of the source address.
- *neighbor*:Each node sends to a distination whose address is remaining the upper half of the source address and shift the lower half by one positon.
- *shuffle*:Each node sends to a distination whose address is arthimetic shift left of the source address.
- *tornado*:Each node sends to a distination whose address is remaining the upper half of the source address and arthimetic shift right lower half by number of position equal to radix-1.
- *Transpose-1*: Each node sends messages to a destination with the exchange of the upper and lower halves of the source address.
- *Transpose-2*: Each node sends messages to a destination with the exchanege of the upper and lower halves of the source address shifted by one position.

Table 4.1 lists all the previous patterns, showing for each pattern the crosspending destiantion address(binary coordiaties) strarting from a given source address. The example use 4x4 mesh topology.

- **Real appliction**
  Splash-2x benchmark suite [56] includes applications and kernels mostly

Source (binary coordinates):(y_3,y_2,y_1,y_0, x_3,x_2,x_1,x_0)	
Traffic Pattern	Destination(bnary coordinates)
Bit-reverse	(x_0,x_1,x_2,x_3,y_0,y_1,y_2,y_3)
Bit-rotation	(x_0,x_1,x_2,x_3 , y_3,y_2,y_1,y_0)
Shuffle	(y_2,y_3,y_0,x_3,x_2,x_1,x_0,y_1)
Tornado	(y_3,y_2,y_1,y_0,x_1,x_0,x_2,x_3)
Transpose-1	(x_3,x_2,x_1,x_0 , y_3,y_2,y_1,y_0)
Transpose-2	(x_2,x_1,x_0,x_3 , y_2,y_1,y_0,y_3)
Negibhor	(y_3,y_2,y_1,y_0,x_0,x_3,x_2,x_3)
Uniform_random	random()

Table 4.1: Synthetic traffic patterns for 4x4 mesh

in the area of high performance computing (HPC). It has been widely used to evaluate multiprocessors and their designs. years.

4.1.1 System configuration

For the evaluation we compared our proposed fully adpative routing algorithm which applying both whole packet forwarding and partial packet restoring mechanisms (PSF+WPF+PPR) against three routing algorithms. we implemented a port-selection-first fully adpaprive routing algorithm with conservative Vc re-allocation (PSF) and with whole packet forwarding (PSF+WPF). The determinstic routing algorithm is (DOR) as a basline. We use a local selection strategy for all adaptive routing; when there are two premissible ports, it chooses the one with more free buffer, other wise if the ports have the same free buffers, it randomly chooses one of them.

For synthetic traffic evaluation, we used 4x4 mesh with 3 VNs. we used 2 VCs per each VN. The VC depth is 4 flits deep. There are single-flit and five-flit packets. The simulator is warmed up for 10,000 cycles. Table 4.2 summarizes the network configuration.

For real applications we run splash-2x benchmarks with 16 threads on a 16-core CMP. Each core is connected to private inclusive L1 cache and shared L2 cache. long packets are 5 flits wide with a 16-byte flit width. We use MESI coherence protocol whcih needs 3 VNs for protocol-level deadlock freedom. Each VN has 2 VCs:each VC is a 4 flits deep. All benchmarks use the simsmall input sets. The total execution time is used as the metric for

Num_cores	16
Num_directories	16
coherence protocol	Garnet standalone
Topology	2D-mesh 4x4 at 16 cores
Vitrual channels	2 VCs per vnet (1 EVC- 1Avc)
Cahnnel width	128 bits
Channel depth	4 flits deep
Flit size	16 byte
Router latency	2 cycles
Link latecny	1 cycle
Total cycles	10,000

Table 4.2: synthetic traffic simualtion configuration

the performance. Table 4.3 summarizes the system configuration.

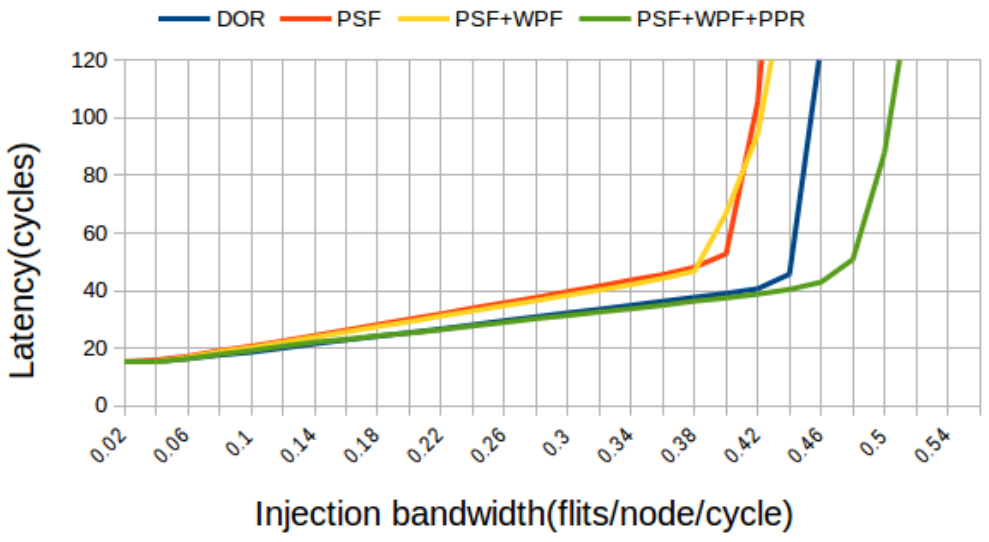
Processor Core	1GHz,In-Order X86 core, 1 cycle per execution phase
L1I Cache	32KB 8-way Set Associative
L1D Cache	32KB 8-way Set Associative
L2 Cache	256KB per bank, 16-way set Associative
coherence protocol	MESI- 3 Virtual network
Topology	2D-mesh 4x4 at 16 cores
Num_directories	4
Vitrual channels	2 VCs per vnet (1 EVC- 1AVC)
Cahnnel width	128 bits
Channel depth	4 flits deep
Flit size	16 byte
Router latency	2 cycles
Link latecny	1 cycle
Real traffic	Subset of Splash-2x bechmarks

Table 4.3: full system simulation configuration

4.2 Experimental results

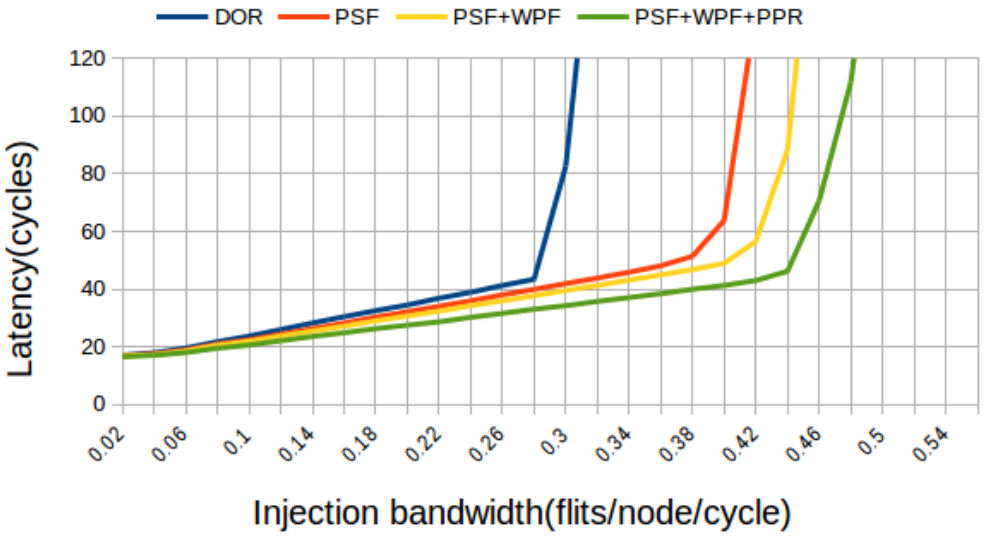
4.2.1 Synthetic traffic

We provided the performance with seven synthetic traffic patterns. We compared our proposal (PSF+WPF+PPR) against three other implementations, Port-selection-first(PSF), whole packet forwarding (PSF+WPF) and DOR which present the deterministic routing. for all the patterns, our proposal gave the heights performance. This increase in performance due to the ability of our proposal to restore long packets. Figure 4.5 shows how our proposal succeeds to restore a high ratio of long packets resulting increasing the packets adaptivity. Figure 4.5(a) shows that our proposal restores a high ratio of short packets near to other implementation, while Figure 4.5(b) shows how we succeed to restore a high portion of long packets.

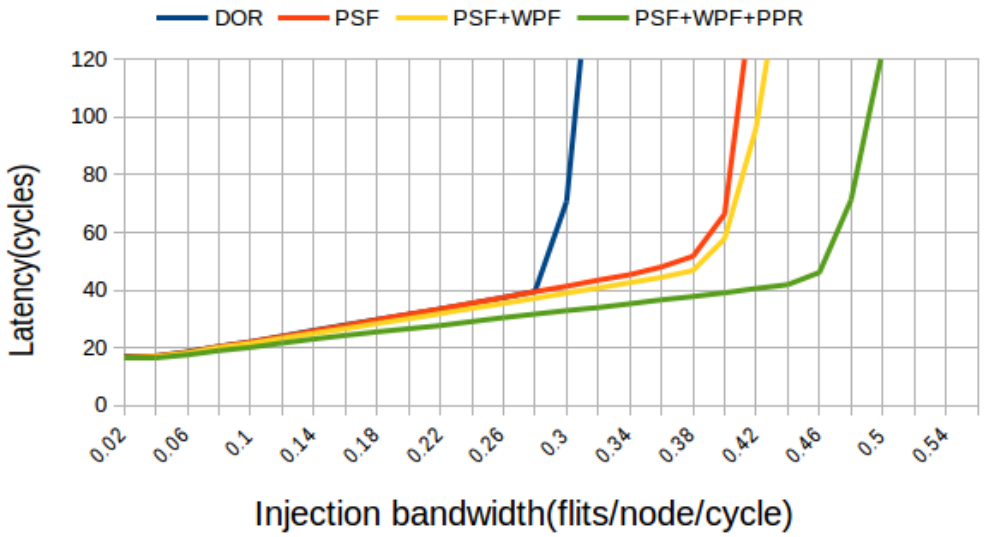


(a) Shuffle

Figure 4.4: Routing algorithm performance using synthetic traffic

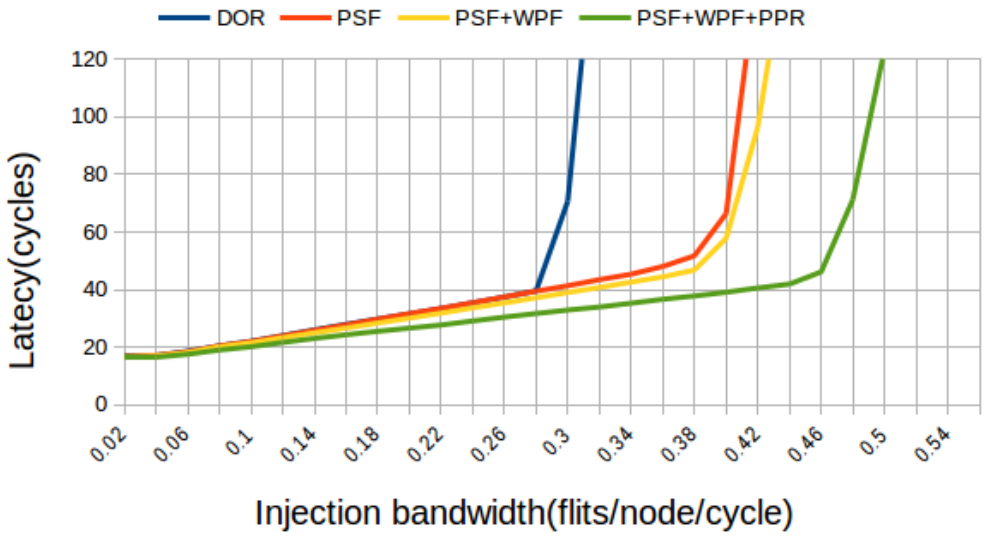


(a) Bit-reverse

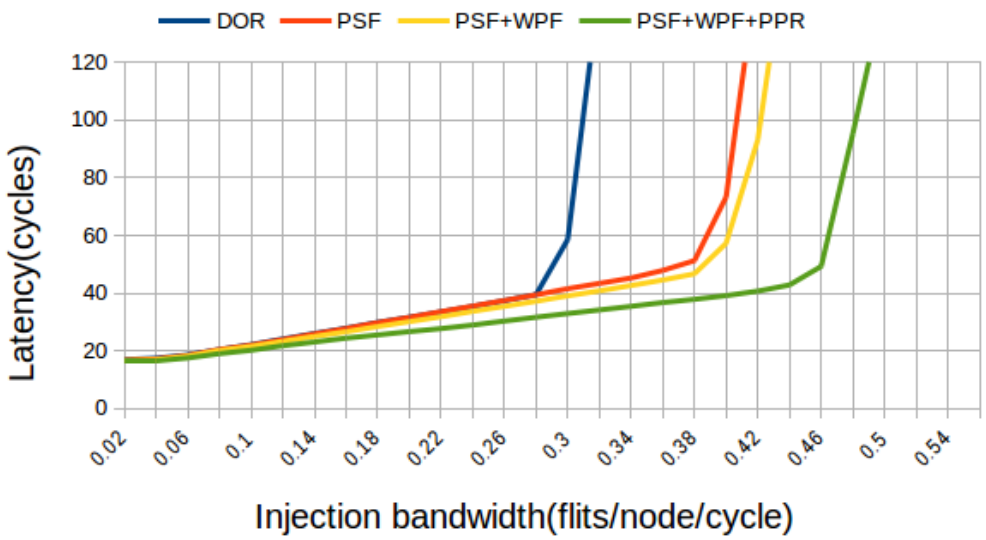


(b) Bit-rotation

Figure 4.1: Routing algorithm performance using synthetic traffic

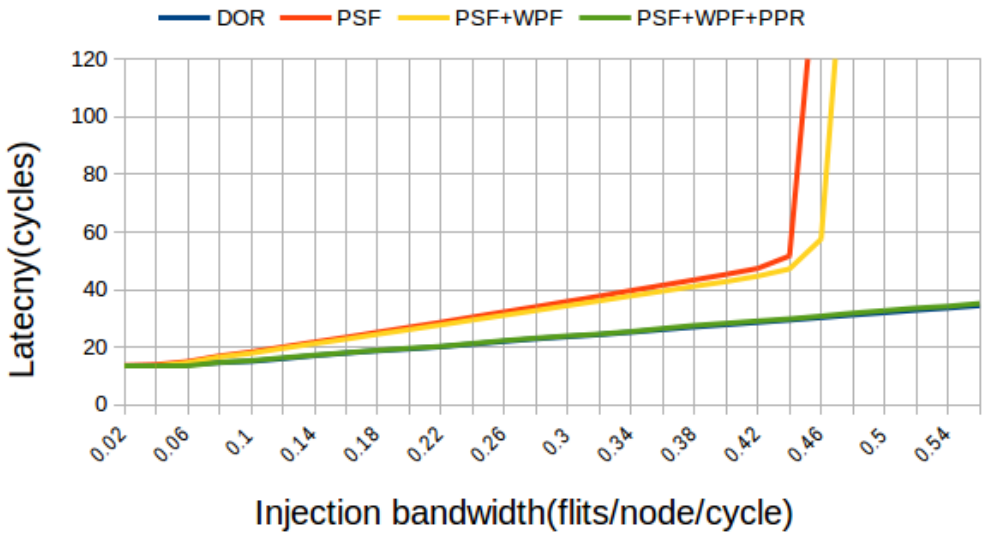


(a) Transpose-1

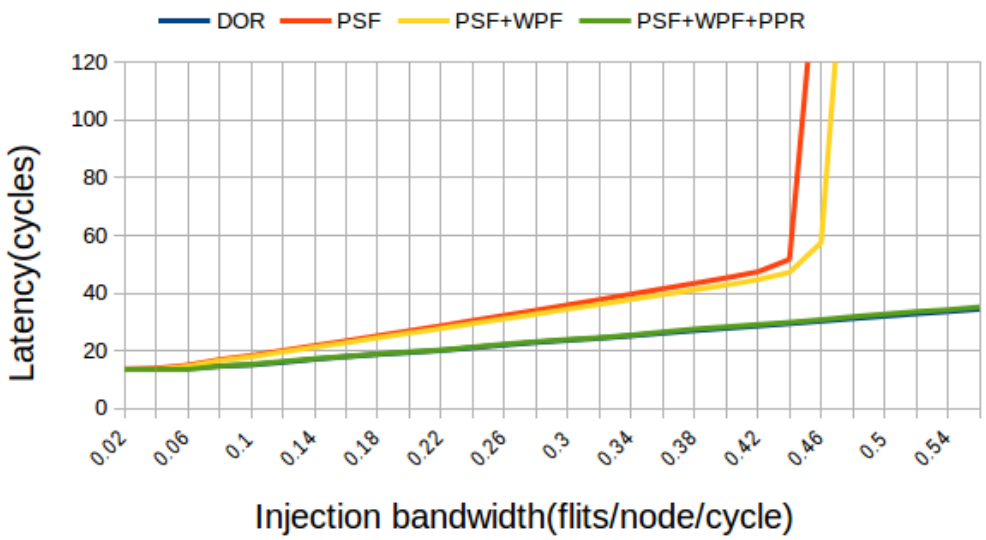


(b) Transpose-2

Figure 4.2: Routing algorithm performance using synthetic traffic

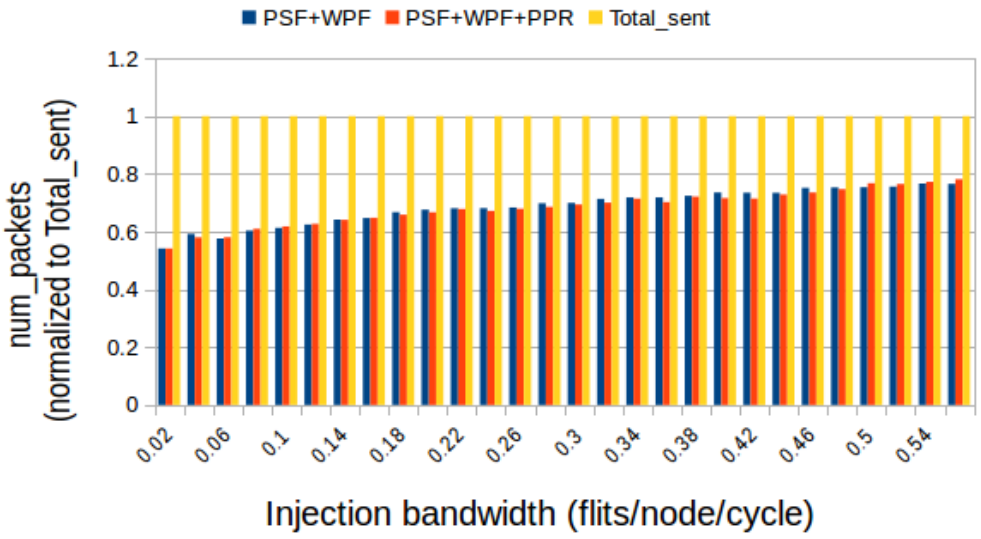


(a) tornado

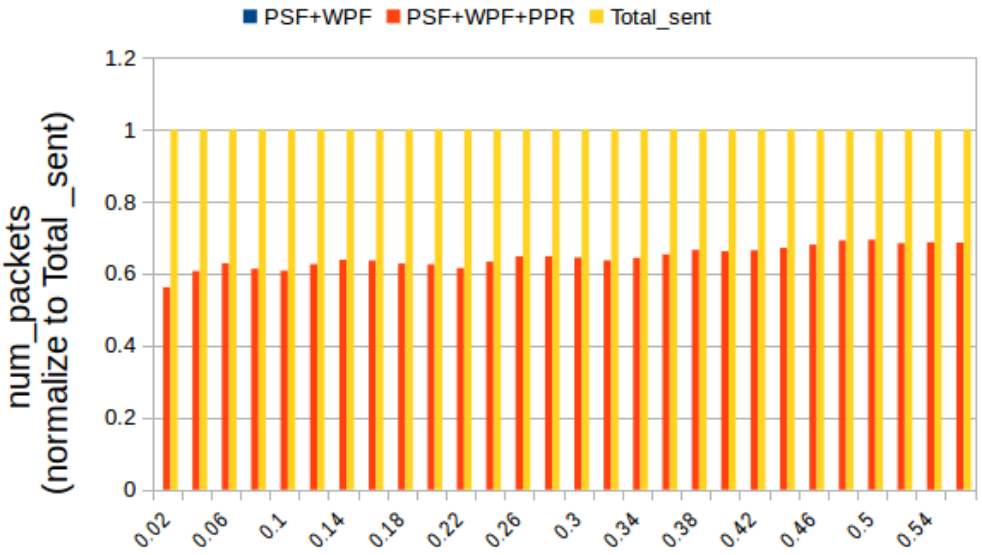


(b) neighbor

Figure 4.3: Routing algorithm performance using synthetic traffic



(a) short-packet-restored



(b) long-packet-restored

Figure 4.5: Packt adaptivity

4.2.2 Real application

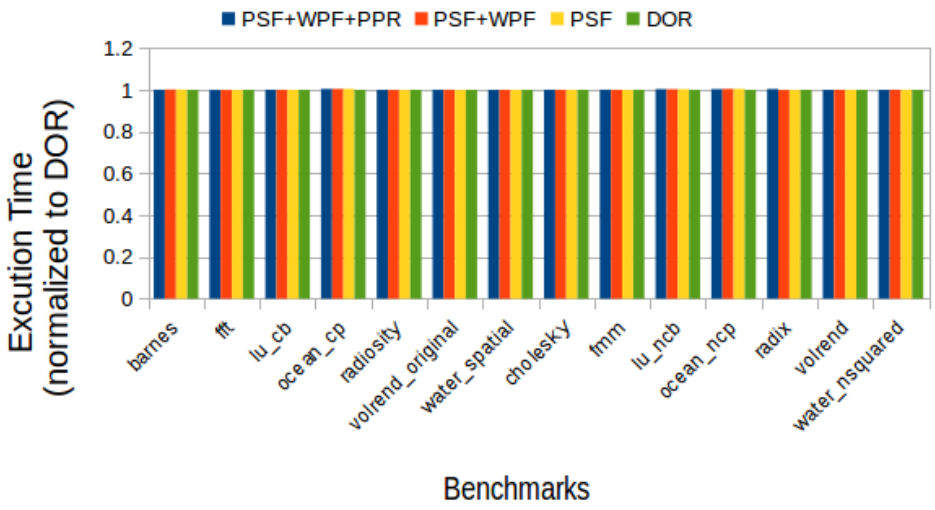
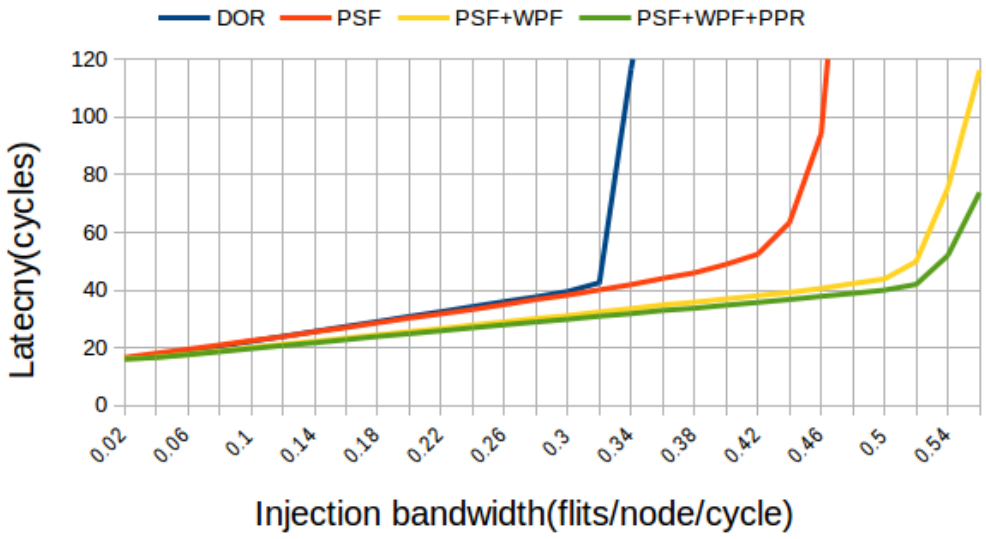


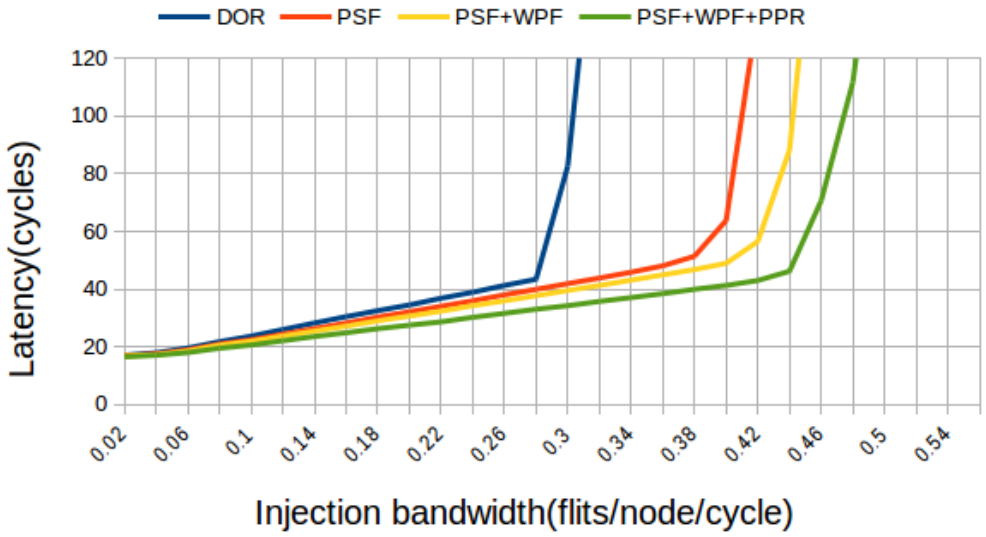
Figure 4.6: system simulation for splash-2x benchmarks

4.3 Design space exploration

In order to strengthen the assessment of the proposed methodology this section explores the effect of increasing the ratio of long packets, as our proposal targets those kinds of packets. Figure 4.7 shows the effect increasing the long packet ration from 20% to 60%. The figure shows that our proposal give more gab when long packets ration increases.



(a) latency-20%



(b) latecny-60%

Figure 4.7: The effect of increasing ratio of long packets

Chapter 5

Conclusions and Future Works

Partial packet restoring is a novel mechanism for fully adaptive routing algorithms in wormhole networks. This mechanism allows long packets to be restored again in the adaptive path. It improves VC utilization in VC-limited networks. With our mechanism, we split the long packet into portions resulting in decreasing the packet length. By decreasing the packet length the buffer space needed for the packet to be fully accommodated in the channel becomes less, which gives a better VC utilization. Our proposal target longe packets especially when we need to restore them from escape paths to adaptive paths in that way we also, increase the network adaptivity. We prove that our mechanism does not induce deadlock. We design a novel fully adaptive routing algorithm which maintains packet adaptivity. Compared with conservative VC re-allocation, our proposal achieves an average 30% saturation throughput improvement in synthetic traffic patterns.

The present work proposed a new mechanism which allows splitting the packets, the performance can be improved if we allow more than one split per packet. The policy to choose between the available paths enables further optimizations that are left as future work.

Bibliography

[1] Schaller, R.R. 1997, "Moore's law: past, present and future", Spectrum, IEEE, vol. 34, no. 6, pp. 52-59.

[2] Lizhe Wang, Jie Tao, Gregor von Laszewski, Holger Marten. 2010, Multicores in Cloud Computing: Research Challenges for Applications, Journal of Computers, Vol 5, No 6 (2010)

[3] J. Hennessy and D. Patterson, Computer Architecture - A Quantitative Approach,D. Penrose, Ed. Morgan Kaufmann, 2003

[4] William Dally and Brian Towles. Principles and Practices of Interconnection Networks. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[5] Giorgos Passas, Manolis Katevenis, and Dionisios N. Pnevmatikatos. Crossbar nocs are scalable beyond 100 nodes. IEEE Trans. on CAD of Integrated Circuits and Systems, 31(4):573585, 2012.

[6] Sheng Ma,Natalie Enright Jerger,Zhiying Wang,"Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip"High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on,feb.2012

[7] J. Duato, T. Pinkston, "A general theory for deadlock-free adaptive routing using a mixed set of resources", IEEE Trans. Parallel Distrib. Syst., vol. 12, no. 12, pp. 1219-1235, Dec. 2001.

[8] N.E. Jerger, L. Peh, On-Chip Networks, USA, CA, San Rafael:Morgan and Claypool Publ., 2009.

[9] C. Fallin, C. Craik, O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router", Proc. HPCA, pp. 144-155, 2011.

[10] E. Fleury, P. Fraigniaud, "A general theory for deadlock avoidance in wormhole-routed networks", IEEE Trans. Parallel Distrib. Syst., vol. 9, no. 7, pp. 626-638, July 1998.

[11] B. Fu, Y. Han, J. Ma, H. Li, X. Li, "An abacus turn model for time/space-efficient reconfigurable routing", Proc. ISCA, pp. 259-270, 2011.

[12] C. Glass, L. Ni, "The turn model for adaptive routing", Proc. ISCA, pp. 278-287, 1992.

[13] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shivakumar, S.W. Keckler, D. Burger, "On-chip interconnection networks of the TRIPS chip", IEEE Micro, vol. 27, no. 5, pp. 41-50, Sept.-Oct. 2007.

[14] P. Gratz, B. Grot, S. Keckler, "Regional congestion awareness for load balance in networks-on-chip", Proc. HPCA, pp. 203-214, 2008.

[15] K. Anjan, T. Pinkston, "An efficient fully adaptive deadlock recovery scheme: Disha", Proc. ISCA, pp. 201-210, 1995.

[16] D. Becker, W. Dally, "Allocator implementations for network-on-chip routers", Proc. SC, pp. 52, 2009.

[17] G. Chiu, "The odd-Even turn model for adaptive routing", IEEE Trans. Parallel Distrib. Syst., vol. 11, no. 7, pp. 729-738, July 2000.

[18] W. Dally, "Virtual-channel flow control", IEEE Trans. Parallel Distrib. Syst., vol. 3, no. 2, pp. 194-205, Mar. 1992.

[19] W. Dally, C. Seitz, "The torus routing chip", Distrib. Comput., vol. 1, no. 4, pp. 187-196, 1986.

[20] W. Dally and C. Seitz. Deadlock-free message routing in multipro-

[21] W. Dally, C. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", IEEE Trans. Comput., vol. C-36, no. 5, pp. 547-553, May 1987.

[22] W. Dally, B. Towles, "Route packets not wires: On-chip interconnection networks", Proc. DAC, pp. 684-689, 2001.

[23] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, USA, CA, San Francisco:Morgan Kaufmann, 2003.

[24] S. Eachempati, A.K. Mishra, V. Narayanan, C.R. Das, "Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs", Proc. HPCA, pp. 175-186, 2009.

[25] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks", IEEE Trans. Parallel Distrib. Syst., vol. 4, no. 12, pp. 1320-1331, Dec. 1993.

[26] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks", IEEE Trans. Parallel Distrib. Syst., vol. 6, no. 10, pp. 1055-1067, Oct. 1995.

[27] M. Hayenga, N.E. Jerger, M. Lipasti, "SCARAB: A single cycle adaptive routing and bufferless network", Proc. MICRO, pp. 224-254, 2009.

[28] J. Hu, R. Marculescu, "DyADSmart routing for networks-on-chip", Proc. DAC, pp. 260-263, 2004.

[29] P. Kermani, L. Kleinrock, "Virtual cut-through: A new computer communication switching technique", Comput. Netw., vol. 3, no. 4, pp. 267-286, Sept. 1979.

[30] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, C.R. Das, "A low latency router supporting adaptivity for on-chip interconnects", Proc. DAC, pp. 559-564, 2005.

[31] M. Li, Q. Zeng, W. Jone, "DyXYA proximity congestion-aware deadlock-free dynamic routing method for network on chip", Proc. DAC, pp. 849-852, 2006.

[32] X. Lin, P. McKinley, L. Ni, "The message flow model for routing in wormhole-routed networks", IEEE Trans. Parallel Distrib. Syst., vol. 6, no. 7, pp. 755-760, July 1995.

[33] S. Ma, N.E. Jerger, Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip", Proc. ISCA, pp. 413-424, 2011.

[34] R. Marculescu, U.Y. Ogras, L.-S. Peh, N.E. Jerger, Y. Hoskote, "Outstanding research problems in NoC design: System microarchitecture circuit perspectives", IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 28, no. 1, pp. 3-21, Jan. 2009.

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

[36] Y. Xu, B. Zhao, Y. Zhang, J. Yang, "Simple virtual channel allocation for high throughput and high frequency on-chip routers", Proc. HPCA, pp. 1-11, 2010.

[37] S. Mukherjee, P. Bannon, S. Lang, A. Spink, D. Webb, "The Alpha 21364 network architecture", Proc. Hot Interconnects, pp. 113-117, 2001.

[38] R. Mullins, A. West, S. Moore, "Low-latency virtual-channel routers for on-chip networks", Proc. ISCA, pp. 188, 2004.

[39] U.Y. Ogras, J. Hu, R. Marculescu, "Key research problems in NoC design: A holistic perspective", Proc. CODES + ISSS, pp. 69-74, 2005.

[40] C. Fallin et al. CHIPPER: A low-complexity bufferless deflection router. In HPCA 2011.

[41] W. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In DAC 2001.

[42] B. Fu et al. An abacus turn model for time/space-efficient reconfigurable routing. In ISCA 2011.

[43] M. Hayenga et al. SCARAB: A single cycle adaptive routing and bufferless network. In MICRO 2009.

[44] P. Gratz, B. Grot, and S. Keckler. Regional congestion awareness for load balance in networks-on-chip. In HPCA 2008.

[45] J. Hu and R. Marculescu. DyAD - smart routing for networks-on- chip. In DAC 2004.

[46] Y. Xu et al. Simple virtual channel allocation for high throughput and high frequency on-chip routers. In HPCA 2010.

[47] L. Peh, W. Dally, "A delay model and speculative architecture for pipelined routers", Proc. HPCA, pp. 255-266, 2001.

[48] A. Roca, J. Flich, F. Silla, J. Duato, "VCTlite: Towards an efficient implementation of virtual cut-through switching in on-chip networks", Proc. HiPC, pp. 1-12, 2010.

[49] L. Schwiebert, D.N. Jayasimha, "A necessary and sufficient condition for deadlock-free wormhole routing", J. Parallel Distrib. Comput., vol. 32, no. 1, pp. 103-117, Jan. 1996.

[50] A. Singh, W.J. Dally, A.K. Gupta, B. Towles, "GOAL: A load-balanced adaptive routing algorithm for Torus networks", Proc. ISCA, pp. 194-205, 2003.

[51] S. Taktak, E. Encrenaz, J. Desbarbieux, "A polynomial algorithm to prove deadlock-freeness of wormhole networks", Proc. PDP, pp. 121-128, 2010.

[52] F. Verbeek, J. Schmaltz, "Automatic verification for deadlock in networks-on-chips with adaptive routing and wormhole switching", Proc. NoCS, pp. 25-32, 2011.

[53] F. Verbeek, J. Schmaltz, "A Comment on A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks", IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 10, pp. 1775-1776, Oct. 2011.

[54] F. Verbeek, J. Schmaltz, "On necessary and sufficient conditions for deadlock-free routing in wormhole networks", IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 12, pp. 2022-2032, Dec. 2011.

[55] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Carl, M. Mattina, C.-C. Miao, J.F. Brown, A. Agarwal, "On-chip interconnection architecture of the tile processor", IEEE Micro, vol. 27, no. 5, pp. 15-31, Sept.-Oct. 2007.

[56] Steven Came ron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. 1995. The SPLASH-2 programs: characterization and methodological considerations.In Proceedings of the 22nd annual international symposium on Computer Architecture, June 1995.

[57] Gem5 introduction
`http://gem5.org/Main_Page`

[58] Garnet2.0: An On-Chip Network Model for Heterogeneous SoCs
`http://www.gem5.org/Garnet2.0`