Low Latency On-Chip Networks through Compression and Multicasting

(圧縮とマルチキャストを用いた低遅延オンチップネットワーク)

Yuan He

The University of Tokyo

March 2014

Advisor: Professor Hiroshi Nakamura Author: Yuan He

Abstract

The inevitable advent of the multi-core era has driven an increasing demand for low latency on-chip interconnection networks (or NoCs). Being a critical part of the memory hierarchy for modern chip multi-processors (CMPs), these networks face stringent design constraints to provide fast communication. Modern NoC' s first order concern is clearly its latency, so we present three low latency techniques in this thesis. One is through traffic compression while the other two are low latency router designs based on multicast-able crossbar switches.

Firstly, an adaptive traffic compression scheme is proposed, taking account of the vertical bandwidth limitation of 3D NoCs and traffic compressibility. It is found that the compressibility based adaptive compression is very useful against incompressible traffic while the location-based adaptive compression is more effective with more layers.

Secondly, an improvement is made on Prediction Router, which routes packet based on predicted destinations. This improvement makes use of multiple prediction algorithms (so-called the Predict-more Router) at the same time for one packet. It helps increasing the prediction accuracy by 15% and outperforms the best PR by 3.5% in speeding-up the system.

Thirdly, to further lower the latency and to get rid of predictions, a novel low latency router (McRouter) is proposed through multicasting a packet to all possible outputs. This design allows a single cycle transfer of flits when having enough

bandwidth within the router. So it may behave like an always-hit prediction router. Evaluation shows that McRouter helps achieving system speed-ups of 1.28, 1.17 and 1.05 over the conventional router, the VSA router and the best prediction router, respectively.

Acknowledgments

Throughout this long journey, I had so many great moments to be engraved into my memory and so much great support from many to appreciate. Without these, it is impossible for me to look back with such a joy and to complete this research endeavor.

First and foremost, my heartiest gratitude goes to my advisor, Professor Hiroshi Nakamura. From the day I started my study in his laboratory, he opened up a door for me to work as a researcher by sharing his knowledge and enthusiasm. His incisive comments and valuable advices had been the key to my progress and the completion of this dissertation. Not only supported me in various aspects during my study, he also clearly taught me the real essence of research and steadily led my way through it. It had been both an honor and a joy for me to work with him.

I would also like to acknowledge Professors Hiroyuki Morikawa, Masahiro Fujita, Masahiro Goshima and Masaaki Kondo for their thoughtful advices on my dissertation in the defense. In addition, Professor Masaaki Kondo's work (while he was working with Professor Hiroshi Nakamura) was the reason for my admission to Nakamura Laboratory at the very beginning.

I am also very grateful to Professors Shinobu Miwa, Hiroshi Sasaki and Hiroki Matsutani (in addition to Professor Hiroshi Nakamura) for being co-authors of our technical papers. It had been so vital to have them when I was struggling in writing these papers. Their collaboration had made these papers materialized

which had become important building blocks of my dissertation. Professor Hiroshi Sasaki had also been the one who brought me to the field of computer architecture by sharing his thoughts and ideas and providing hints when I was stuck. Professor Hiroki Matsutani had been very kind to me as well. He helped me understand the basics of on-chip interconnections and helped me build the evaluation platforms for my work. For the same reason, I would also like to thank Professor Takashi Nakada, discussions with him had always been so helpful on improving the quality of my work.

At the early stage of my study, Professors Takashi Nanya and Masashi Imai had also provided me various support. I am in debt to them as well. I also appreciate Professors Zoran Salcic, Morteza Biglari-Abhari and Partha Roop's help for bringing me into the field of computer systems while I was studying at the University of Auckland.

During my time at Nakamura Laboratory, Ms. Setsuko Yuge, Akiko Kabayama, Yumiko Kumaoka, Remi Maehashi and Sara Noguchi had helped me through many general and administration issues. I want to express my sincere thanks to them too.

Time would not pass with such a joy without many friends during these years. They helped me make up my life other than this dissertation. I would like to thank Roberto Drebes, James Weston, Seidai Takeda, Kyundong Kim, Toshiya Komoda, Eishi Arima, all other lab members and many personal friends during my study for their company. It had been so supportive and I really appreciate it.

Lastly, I want to thank my family, especially my wife and daughter, for their care and love. Nothing is possible without these.

Contents

List of Figures

List of Tables

Well begun is half done.

Aristotle

1 Introduction

It has been nearly half a century since Moore's Law was firstly observed; and until today it is still used to guide the technology scaling in semiconductor industry. As a result of this still-ongoing technology scaling, the amount of transistors per chip continues to double roughly every 2 years [1, 40]. However, the end of Dennard scaling [23], from where the improvement on energy efficiency of semiconductor devices has retreated substantially, simply marked a historic paradigm shift to multi-core designs, for which the number of processor cores per chip is scaled up rather than spending transistor budget on core performance [8, 12, 42].

As such multi-core systems continue to scale with the technology trend, the industry and researchers are fast moving to on-chip networks (also called networks-on-chip or NoCs for short) from shared buses and dedicated wires in order to confront the growing wire delay and increasing arbitration overheads [14, 19, 20, 46]. Despite many alternatives to employ on-chip networks in multi-core systems, a lot of challenges are still left un-addressed. Performance-wise, one of the the most critical concern of on-chip networks is their communication latency which scales up with their size.

Hence, three solutions targeted at shortening the communication latency of on-chip networks are presented in this dissertation. In the next few sections, an introduction to on-chip networks will be provided and this will be followed by an overview of this dissertation including its objective, problem definitions, contributions, assumptions, scope and organization.

1.1 On-chip Networks

The concept of NoCs came out as a replacement of buses and crossbars in the multi-core era. The reason behind this is obvious. As core count scales up with technology scaling, both buses and crossbars are not sufficient enough to provide the scalability. As for buses, more cores require longer buses and more complex arbiters which add both wire delay and arbitration delay substantially. Another problem of buses is their bandwidth, with more and more numbers of cores, a bus can be easily saturated. For crossbars, their scalability is limited by their power and

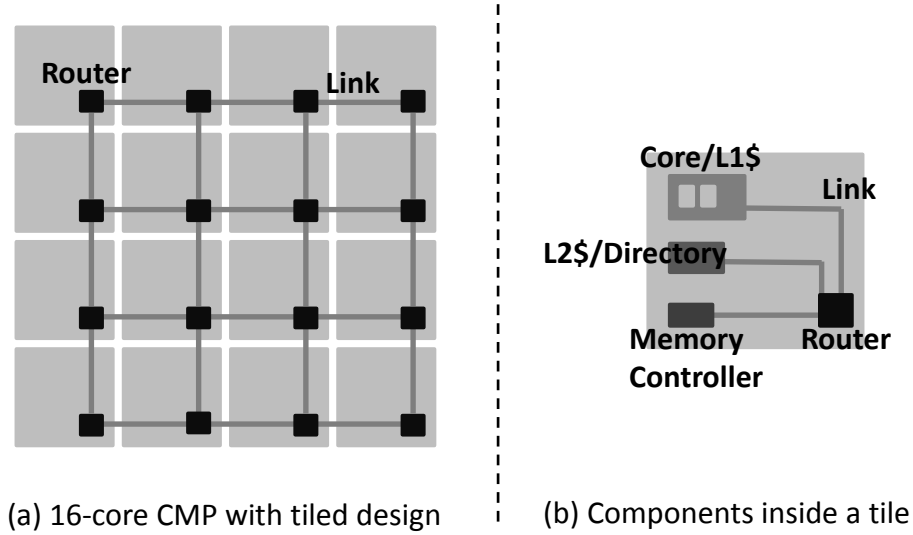


Figure 1.1.1: 16-tile CMP connected with an NoC.

area consumptions when they are used to connect more and more cores since their complexity grows rapidly. As an example, a typical on-chip network is shown in Figure 1.1.1a.

With NoCs, the benefits are as follows. Firstly, they scale much better in power and area than buses and crossbars. Secondly, they make good use of wiring since links are distributed across the network and wire length are much shorter than buses. This also enables NoCs to supply higher bandwidth. Thirdly, NoCs are modular in design.

Typical NoCs have repetitive structures and can be seen as a few basic components. Such components include network interfaces, routers and links. Each of them will be briefly introduced below.

- **Network interface** sits between network nodes (such as processor cores, cache banks and memory controllers) and the network. It is capable of

breaking packets into flits and vice versa. Flits are the minimum units of network traffic and their size is the same as link width. Network interfaces are also used to inject and receive flits to and from the network.

- **Router** is the most important part of NoCs since its delay determines the communication latency per hop. A typical NoC router is connected to network interfaces and neighbor routers through links and it has buffers, routing circuits, allocators and a crossbar inside. Depending on the topology of the network, router determines the route of a network packet through which it travels in the network. To minimize router delay in order to shorten the network communication latency, its architecture has been extensively studied.

- **Link** is a set of wires joining other components in the network. NoC links are distributed through the network and are much shorter than conventional buses.

These basic components also form the topology of the network. A topology not only reflects the layout of the network but also determines the average number of hops a packet may travel and the distance between two network nodes. For example (as in Figure 1.1.2), common on-chip network topologies include (but are not limited to) ring, mesh, torus and fat tree [19, 46, 49].

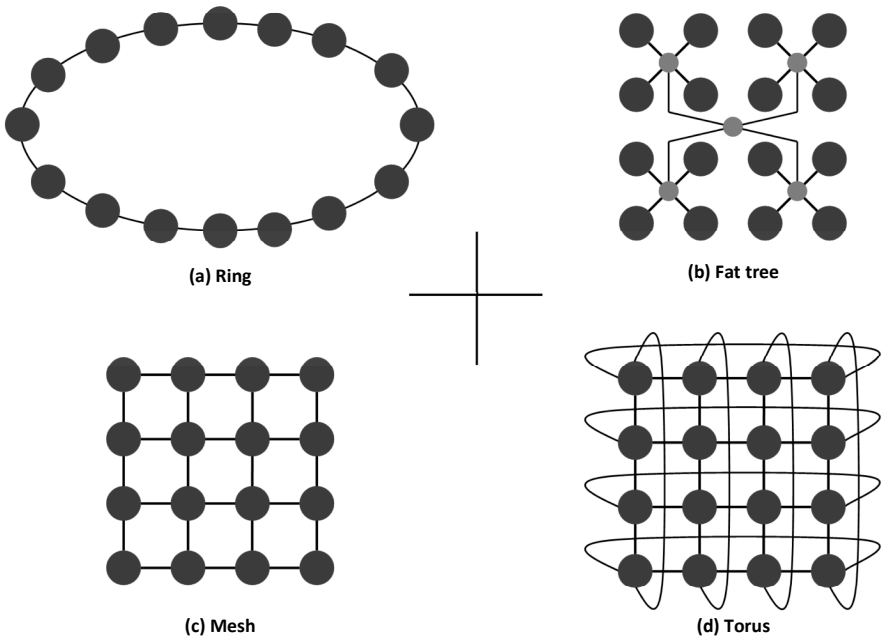


Figure 1.1.2: A few topologies for on-chip networks.

1.2 Objective, Problem Definitions and Contributions

The objective of this dissertation is to shorten the communication latency for on-chip networks. The reason behind this is that on-chip network is part of the memory hierarchy of modern multi-core chips, hence its communication latency is highly related to memory performance of the chip. Not only main memory accesses are going into such network but accesses to the L2 cache, accesses to L1 caches of other processor cores and coherence messages are also traveling through them.

Before going into problem definitions, there are two formulae[1] which details the

[1]These two formulae are derived from [25, 46].

packet latency and router delay in NoCs.

$$PacketLatency = T_{Injection} + T_{LinkPropagation} \times (h+1) + T_{Router} \times h + \frac{Packet}{LinkBandwidth} + T_{Reception} \tag{1.1}$$

In Formula (1.1), packet latency is composed of injection/reception delay at the network interfaces, link propagation delay multiplied by the number of links a packet travels, router delay multiplied by the number of hops a packet takes and the serialization delay which is the packet size divided by link bandwidth.

$$T_{Router} = T_{Buffering} + T_{Routing} + T_{Allocation} + T_{Switching} \tag{1.2}$$

Looking further into the router delay, as in Formula (1.2), there are time spent on buffering flits, computing the route, router resource allocation (failed allocations is a reflection of resource contention) and switching the flits.

So far, it is clear that minimizing any term in Formula (1.1) will help reduce the packet latency. But in this dissertation, this packet latency is optimized with two ideas. Firstly, under a fixed link bandwidth, traffic compression is employed to shrink the size of a packet in order to reduce both serialization delay and allocation delay (compression may reduce the number of flits, hence resulting in smaller pressure during router resource allocations). Secondly, two low latency router designs are proposed to minimize the router delay.

The contributions of this dissertation can be stated as follows.

- Identifying the bandwidth limitations in 3D NoCs when traffic travels be-

tween layers.

- Proposing and exploring adaptive control of traffic compression on 3D NoCs. This adaptive control invokes compression when it is beneficial and/or the traffic is encountering bandwidth limitations.

- Demonstrating the effectiveness of the three adaptive compression policies when they are applied on 3D NoCs through simulation.

- Finding that prediction accuracy can be significantly improved if multiple predictions can be carried out for one packet at once (the Wisdom of Crowds in predictive routing with multiple algorithms).

- This finding is utilized to implement a low latency router called predict-more router (PmR) which more successfully hides route computation and arbitration delays than the original prediction router with a marginal power overhead.

- Finding that a router's internal bandwidth can be very plentiful for multi-threaded workloads.

- Proposing a new low latency router named McRouter which successfully hides route computation and arbitration delays by utilizing the remaining bandwidth inside a router more productively. Different from previous techniques on speculating the route computation results (such as prediction router), multicast never misses. Also, multicast operations are both bandwidth-

dependent and speculative so that McRouter only consumes remaining bandwidth within a router and it barely degrades the routing efficiency.

- Presenting the detailed design of McRouter and demonstrating its effectiveness through performance and power evaluations against counterparts.

1.3 Assumptions and Scope

To make the evaluations in this dissertation tractable, there are some basic assumptions made for the configurations in evaluations. In this dissertation, a general-purpose chip multi-processor (CMP) system with 16 in-order cores are assumed while the network topology is 2D (for the low latency routers in Chapter 4 and Chapter 5) and 3D (for the traffic compression work in Chapter 3) mesh with 16 tiles. In addition to a processor core, each tile has a router, a bank of L2 cache and sometimes a memory controller. Figure 1.1.1 depicts such a system in 2D mesh topology and what a tile is like. Variations of this system and the tiles in 3D mesh topologies are described in Chapter 3. For 3D integration, the vertical links are set to 16-bit wide while the planar links are 128-bit wide. The reasons for having assumptions of general-purpose CMPs with in-order cores under mesh topology are popularity and simplicity. For example, available large scale CMPs such as the Tilera TILE64 Processors and the Intel Single-Chip Cloud Computer employ in-order cores and mesh topologies [13, 26]. Both in-order cores and mesh topologies are very foreseeable design decisions when considering scalability and power efficiency. The reason behind the choice of 16-bit vertical link in 3D integra-

tion follows considerations of the chip size of 16 in-order cores, current industry standard (such as JEDEC Wide I/O) and different 3D integration implementation technologies (such as wire-bonding, micro-bump and through-silicon via) [2, 22].

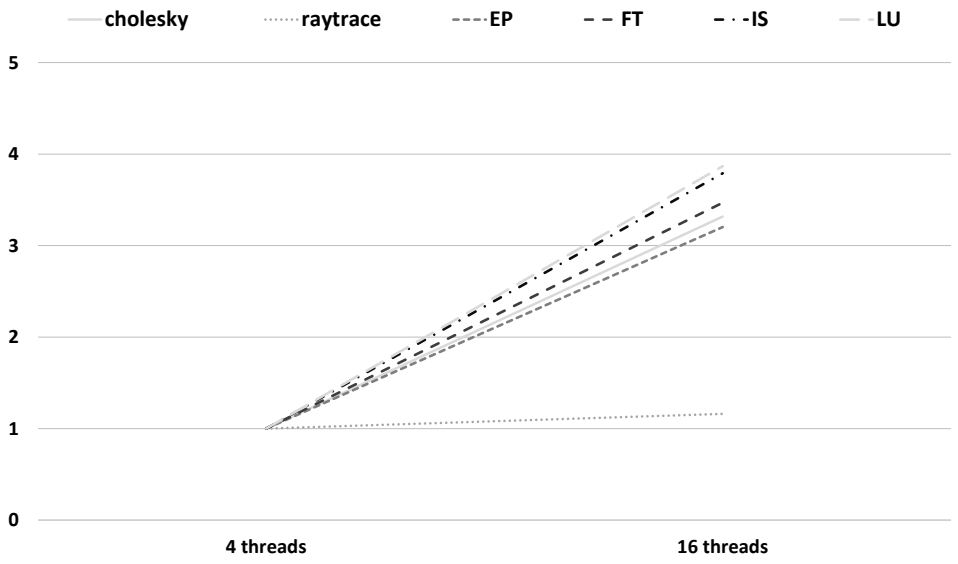


Figure 1.3.1: Parallel speed-up for some multi-threaded workloads.

The workloads used are multi-threaded parallel applications since the focus of this dissertation is on high-performance computing [3, 54]. They are parallelized with the OpenMP API [4]. All threads are doing similar job so their communication characteristics can be very similar. For some workloads (such as *EP* in NPB 3 [27, 53]) that achieve near-ideal parallel speed-up, this simply means that performance improvements on such workloads through solutions in this dissertation can be very similar regardless of the number of parallel threads. They help improving the memory access latency of every parallel thread. Otherwise for workloads whose speed-up is limited by communication overhead after parallelization,

such communication overhead can also be accelerated through the solutions proposed in this dissertation. Parallel speed-up for some workloads from 4 threads to 16 threads are shown in Figure 1.3.1. Evaluation conditions are the same as those in Section 4.3 and Section 5.4. From this figure, nearly all tested workloads scale pretty well with more threads (even better than *EP*). The reason behind this is, evaluations in this dissertation are simulating the parallel regions of these workloads. One exception is *raytrace*, its parallel speed-up with more threads scales poorly and is different from previous work [54]. A possible reason is, *raytrace* suffers a lot from the non-uniform cache access latencies. Operating system (the OS used in evaluations is Solaris 9 [6]) attaches threads to processor cores through natural task mapping. Natural mapping is chosen for its simplicity, popularity and it also is not specifically optimized for solutions proposed in this dissertation.

For the memory hierarchy, each core has its own private L1 caches while an L2 cache is shared by all cores. This shared L2 cache is exclusive. The reason behind these two assumptions on L2 cache is to have the capacity advantage with exclusive shared last level cache (LLC). Larger cache capacity through exclusive shared LLC is very good for multi-core designs since off-chip bandwidth is believed to be a serious concern for them as the number of cores scale [48]. Cache coherence is maintained through MOESI directory protocol, which means an exact location of a piece of data when L1 miss happens would direct a request to a directory sitting aside the L2 Cache banks. Such a directory based protocol is assumed for its advantage on scalability over broadcast based protocols since less traffic is needed to complete a coherence transaction [51]. This simply means that directory based

protocols are more future-proofing when having more and more cores per chip.

This dissertation is based on the performance and power models of a conventional NoC router and two of its variations [19, 45, 46]. All these router designs assume that virtual channel allocation (VCA) is the most time-consuming pipeline stage (hence the critical path). Link traversal is thus set to 1 cycle.

With the above assumptions on the memory hierarchy and network configurations, there are typically two types of packets with different sizes. Packets with larger size are data packets which are used to transmit a piece of data in the size of a cache line while packets with smaller size (1 flit in this dissertation) simply carries a control message. Based on their purposes, packets can also be categorized into 3 classes, which is either request, response or forward.

Although above assumptions do not cover the entire design space, they are relaxed in discussions to qualitatively give a picture on how different numbers of cores, topologies, coherence protocols, types of applications and so on may affect the effectiveness of proposed solutions in this dissertation.

For the scope of this dissertation, the power implications are not discussed for the first solution (traffic compression) while area overhead is left out for all proposals. These are going to be addressed in future work.

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows. Chapter 2 gives a review of related work which summarizes other low latency techniques for on-chip networks. This is followed by Chapter 3, which presents a low latency tech-

nique through traffic compression for 3D NoCs. Chapter 4 describes predict-more router, one of the two low latency on-chip routers with in-router multicasting; while the other low latency on-chip router design, which is called multicast-within-a-router are introduced in Chapter 5. Finally, this dissertation is concluded in Chapter 6.

Study the past if you would define the future.

Confucius

2

Background: Low Latency Techniques for On-chip Networks

After the introduction to NoCs and this dissertation, existing low latency techniques for NoCs will be covered in this chapter. These studies will be separated into two groups with each one has a different focus. The first group covers techniques that are compression based while the second group has most of the low latency routing techniques reviewed.

2.1 Traffic Compression

Traffic compression for NoCs, as an efficient on-chip optimization, has been extensively studied for 2D design [21, 28, 55]. In [21], the authors were the first to apply frequent pattern compression on a CMP with Network-on-Chip architecture. Their primary goal was to make a comparison between cache compression and network compression with the same algorithm, in terms of their effects on performance and energy consumption. Both [28] and [55] were about compressing data on NoCs with another candidate algorithm, frequent value compression. Although their results are showing positive feedback, it is believed that for any architecture having multiple communicating nodes, frequent value compression can be inefficient because of its overheads of area and synchronization make it scale poorly. In [28], the authors also propose a solution to the area overhead and an adaptive compression control mechanism taking into account the network congestion.

Before the study of traffic compression on NoCs was carried out, there were already many efforts of applying it on bus and cache [9–11, 52]. Moreover, a study carried out in [48] had proved that both cache and bus compression are highly efficient in terms of further scaling CMP designs.

2.2 Low Latency Routers

Over the years after the introduction of NoCs, there are many existing works focusing on shortening the latency of a router [24, 33, 39, 41, 45]. Major improve-

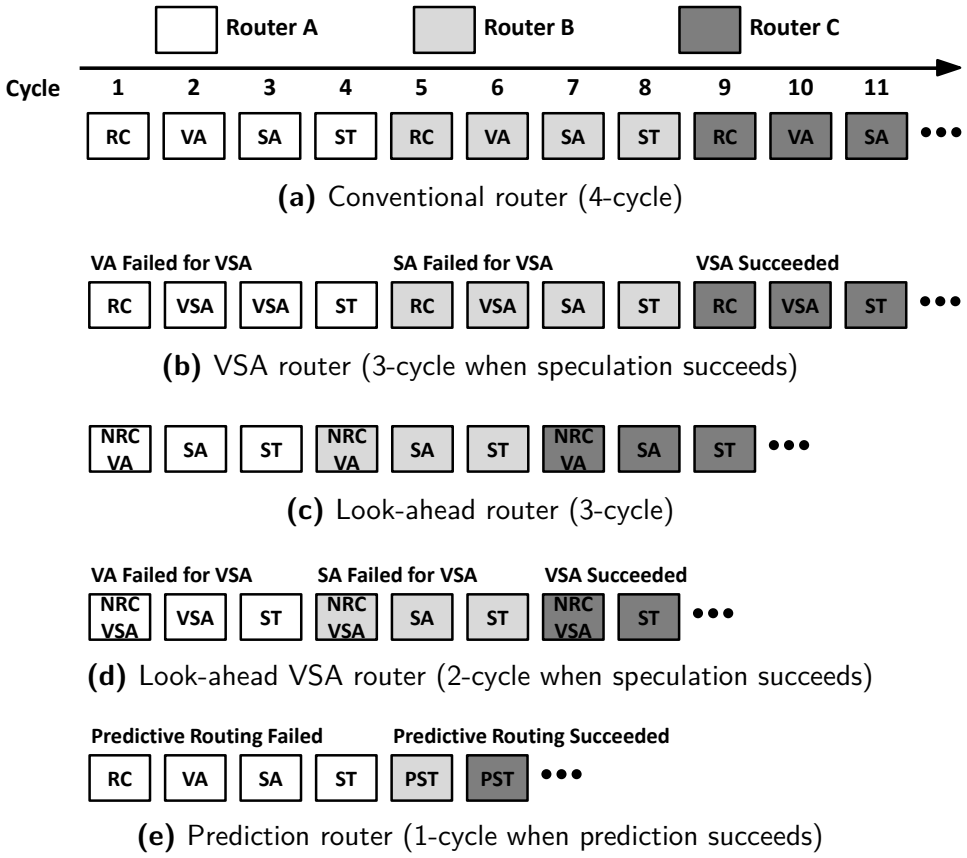


Figure 2.2.1: Pipeline stages of various router designs.

ments on router designs will be covered in this section by starting with the conventional router.

2.2.1 Conventional Router

The structure of a conventional virtual channel router (CR) with a 4-stage pipeline [19] is shown in Figure 2.2.1a. When a packet is transferred through this router, its header flit will serially invoke these 4 pipeline stages which are route computation

(RC), virtual channel allocation (VA), switch allocation (SA) and switch traversal (ST); if there exists any body flit for the same packet, they will instead be processed by SA and ST stages only. As their names suggest, RC is capable of finding the output port for a packet through decoding its header flit and computing its route; VA is used to locate a proper group of buffers for this packet to be stored at the next hop; SA helps allocating a proper time slot for flits of a packet to traverse the crossbar switch; while at last, flits traverse the crossbar switch at the ST stage.

2.2.2 Router Optimizations

The latency of a router determines the transmission delay of a packet. Figure 2.2.1b to Figure 2.2.1e demonstrate the evolution of router pipeline designs with each design resulting in different per-hop latency.

VSA router (VSAR) [45] is proposed that VA and SA stages are overlapped to produce a 3-cycle router as shown in Figure 2.2.1b. The key point is, if both operations succeed while being performed at the same time, VA and SA together only cost 1 cycle (as for Router C). This proposal acts as a CR if VA fails to return a free virtual channel regardless of the result of the speculative attempt of SA (as for Router A).

Look-ahead routing (LAR) shown in Figure 2.2.1c is a non-speculative technique which effectively shortens the router latency to 3 cycles. A router with LAR always performs RC for the next hop. This successfully helps hiding the RC delay. Moreover, as illustrated in Figure 2.2.1d, VSA and LAR can be merged to form a look-ahead VSA router which helps shortening the router latency to 2 cycles when

speculation succeeds (as for Router C).

There are two single cycle routers both utilizing LAR. One is a router whose flit is able to speculatively traverse a crossbar switch if the network is not busy by assuming that arbitration is not needed for a router in such a network condition [41]. Another one is a non-speculative wormhole router named NoX [24]. Instead of arbitrations on the crossbar switch, NoX router relies on a new XOR-based switch which helps hiding the arbitration delay by XORing contending flits (so that routed flits may be XORed).

So far the most aggressive speculation found in routers is the prediction router (PR) [39] whose speculative switch traversal is enabled with predictions of output ports before a packet actually comes to a router. As depicted in Figure 2.2.1e, if a predictive routing succeeds (as for Routers B and C), RC, VA and SA are all hidden since they are already carried out with a predicted RC result.

Kumar *et al.* have proposed another two low latency techniques [33, 34]. The first one introduces a single cycle router with an aggressively optimized control path [34]. It is able to route in one cycle by sending an advanced bundle to help setting up the control before a flit actually comes. The second one proposes an approach called express channels which enable a multi-hop packet to bypass intermediate routers [33].

Size is not a reality, but a construct of the mind; and space a construct to contain constructs.

Robert Anton Wilson

3

Latency Reduction through Traffic Compression

It is not long since the concept of NoCs is being extended to ICs that have three-dimensional structures, namely the 3D NoC [50], in order to mitigate the wire delay and wire energy which are increasingly posing severe problems to modern VLSI design. Traditionally, the wire delay can be mitigated by inserting inverting buffers (i.e., repeaters) on long wires, but the buffers themselves add gate delay and consume energy; thus repeater insertion is not a fundamental solution to the problem. With 3D ICs, a number of wafers or dies are stacked very

closely (e.g., 5μm to 50μm); thus a 3D structure significantly reduces wire length, wire delay, and wire energy compared to 2D counterparts.

For these reasons, 3D NoC is an emerging research topic, and its network topology [44], router architecture [31, 43], and routing algorithms [47] have already been extensively studied.

However, many studies on 3D IC architectures have underestimated the negative impact of vertical interconnects, as reported in [30]. Unfortunately, these vertical interconnects, such as through-silicon vias (TSVs) and microbumps, also consume a certain amount of area. In addition, they affect the routability of wires negatively, because some vertical interconnects interfere with metal layers. Thus, although 3D IC technologies are believed sound beyond Moore's Law, their vertical bandwidth is still a major concern. In practice, such vertical bandwidth limitation can significantly exacerbate the system performance (see Section 3.1).

Since vertical bandwidth limitations come from the physical design constraints mentioned above, to mitigate the performance degradation, there is no other choice but to reduce the amount of communication data, especially for those data moving vertically. In this solution, therefore, a study of traffic compression on 3D NoC architectures is presented with a comprehensive set of scientific workloads.

3.1 Motivation

3D ICs bring many benefits like increased system integration, reduced wire length and increased data locality, but how different wafers or dies are stacked vertically remains an open question for the research community and the industry. Various

interconnection technologies of 3D ICs have been developed for the purpose of vertical stacking, such as wire-bonding, micro-bump [16, 32] and through-silicon via (TSV) [17, 22].

- **Wire-bonding** is a die-to-die interconnection formed with bonding wires. It has a footprint recorded from 35 to 100 um. It is the most common approach and has been highly utilized by System-in-Package designs. The limitation is the number of wires and their density as only edges of a chip is used for the purpose of bonding. Obviously, the bonding wire length can be the cause of a considerable communication delay.

- **Micro-bump** forms a die-to-die interconnection through solder balls. It has a footprint known to be from 10 to 100 um. This approach is generally limited to stack only two dies with face-to-face connections but it can also be used to form connections of more than two dies with face-to-back design although this is believed inefficient because of factors like heat.

- **Through-silicon via (TSV)** is a wafer-level interconnection making use of via-holes formed through multiple wafers. The footprint of TSV is 5 to 50 um thus it has the potential of offering a better interconnection density than wire-bonding and micro-bump. However, it suffers from high manufacturing cost due to the fact that an extra process to form these interconnects. Another constraint of TSV comes from routing, as TSV interconnects interfere with gates and wires. So considering yield and cost, the number of TSV interconnects has major impact in design and it should be considered

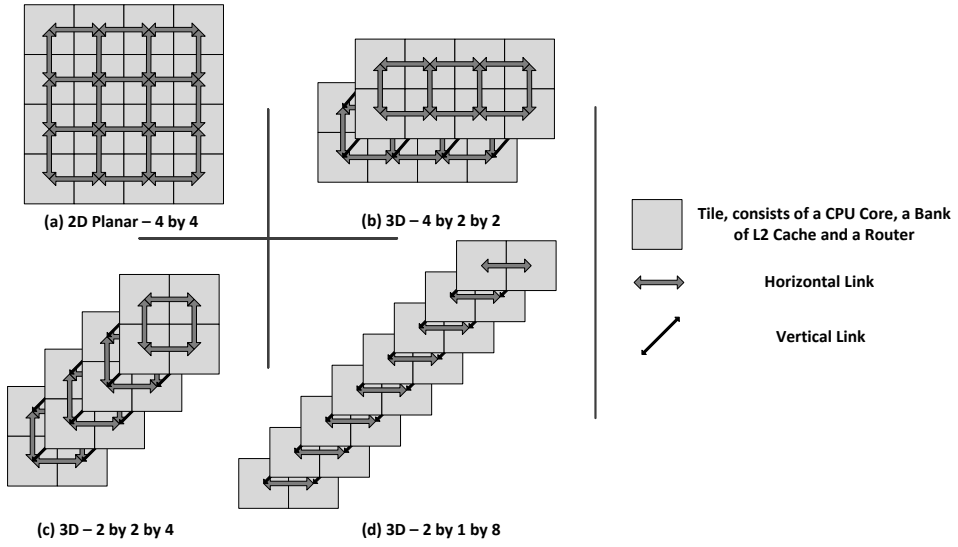


Figure 3.1.1: 2D and 3D NoC topologies.

carefully ahead of manufacturing [30].

As briefly explained above, all three interconnection technologies of 3D ICs have a limitation of going vertical, that is, the die-to-die or wafer-to-wafer interconnection can become a bandwidth bottleneck. With larger numbers of such interconnects, the difficulty of design complexity and the cost of manufacturing are also severe. To depict this vertical bandwidth limitation, a 3D NoC model with heterogeneous link widths are evaluated, which is, for vertical links that are used to move data between dies or wafers, they are modeled as having smaller bit widths compared to horizontal links. In this chapter, the effects for having different numbers of layers (dies/wafers) are also evaluated with the 3D NoCs modeling 2, 4 and 8 layers. An example of the baseline 2D NoC and three 3D NoC configurations are

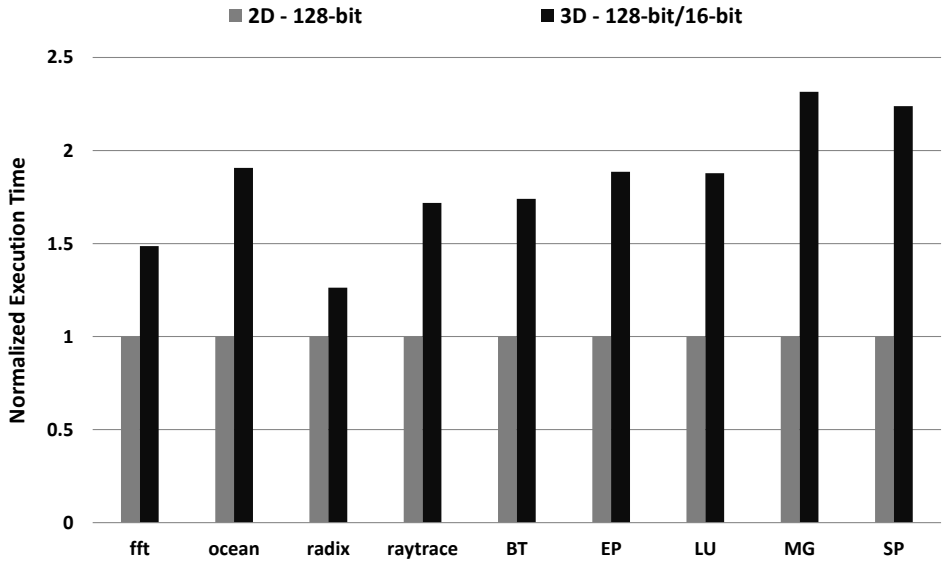


Figure 3.1.2: System performance degradations under link limitations for 3D NoC.

illustrated in Figure 3.1.1. A square represents a tile of the modeled NoCs while the thick and thin arrows denote horizontal and vertical links, respectively.

Moreover, Figure 3.1.2 presents an example of how this link limitation can affect the system performance. Please note the detailed evaluation conditions and environment will be shown in Section 3.3. Both 2D and 3D NoCs are configured in the same way except their link widths. For this particular evaluation, we tested a 2D NoC having 128-bit links and an 8-layer 3D NoC. For the 3D NoC, its horizontal links are set to 128-bit while its vertical links are 16-bit wide. Both configurations assume a total of 16 cores. In this evaluation, the execution time of the same workload is being increased by up to 130%. As shown in Figure 3.1.2, these numbers are far larger than the the 2D NoC with 128-bit links. Thus, vertical link bandwidth

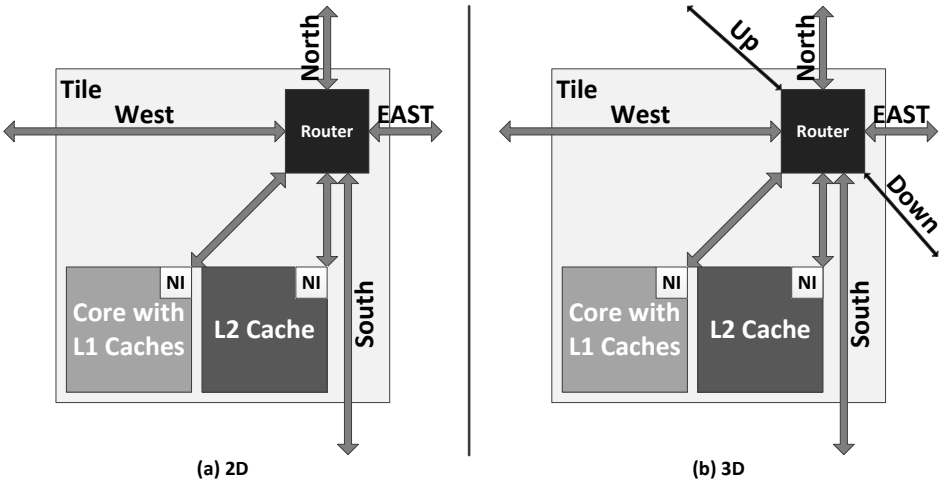


Figure 3.1.3: Tiles of 2D and 3D NoCs.

limitation can be a major bottleneck for any system moving to 3D design.

In the network model shown in Figure 3.1.3b, basic building blocks (tile) of the 3D NoCs are connected with each other by routers and links. For comparison purpose, the tile of a 2D design is shown in Figure 3.1.3a, whose router is at most having six ports and two of them are used to connect to a processor core and an L2 cache bank. For 3D NoCs, two more ports may be added to the router and through two additional links, different dies/wafers are connected. The network routing scheme is also re-defined since X-Y routing for 2D is not sufficient for the 3D design. As shown in the last paragraph, because of the layer-to-layer bandwidth limitation, the 3D NoC models narrower vertical links. More details on configurations of the 3D NoC model are covered in Section 3.3 where the simulation methodology is described.

3.2 Traffic Compression on NoCs

Traffic compression is a popular architectural technique and it has been applied in many fields to conserve on-chip/off-chip bandwidth, to enlarge cache/memory capacity or to reduce communication latency. In this solution, traffic compression is used to conserve bandwidth and to reduce latency for 3D NoCs. In this section, the compression technique is discussed in details. Firstly, an introduction to the compression algorithm, frequent pattern compression, will be briefly covered. After which, the focus is shifted to implementation issues of this compression algorithm. And finally, the proposal of adaptive control on compression will be presented.

3.2.1 Compression Algorithm and Implementation

There are several state-of-the-art traffic compression algorithms which have been applied on NoCs, including frequent pattern compression (FPC) [21] and frequent value compression (FVC) [28, 55]. In this work, FPC is chosen because of its simplicity and effectiveness. FPC is a significance-based compression scheme having small compression/de-compression overheads; unlike FVC, it has no synchronization overhead. FPC compresses frequent patterns appeared in data packets. In this solution, there are seven such patterns with which each 32-bit of data is being compressed and a full description of these patterns are presented in Figure 3.2.1. Of all these patterns, the selection was made upon their frequencies. In [21], it is found that zero words, words with 8-bit data and words with 16-

bit data are the most frequent patterns for workloads from SPLASH-2 [54] and NPB 3 [3]. Therefore, these patterns are selected as shown in Figure 3.2.1. For all seven data patterns, a 3-bit index is assign to each of them. Along with another index for uncompressed data words, there are in total eight indexes which are the compression overhead. For example, a data word of 32 zeros will be replaced with an index of 000 after compression, while an 8-bit sign-extended data word will be replaced with an index of 001 plus the 8-bit data. Please note that although indexes are fixed to 3-bit, the actual data appended to the index may be different in size. For the last index which is "111", the data is uncompressed which results in a negative effect after the combination of index and data. FPC has advantages of high compression ratio and parallel compression. For 128-bit data, it can always be split into 4 parts and each part is compressed with a separate compression circuit. But since FPC employs variable length compression, the de-compression may have to be done in a serial manner.

Regarding the implementation, similar to [21, 28, 55], traffic compression/de-compression circuits in this solution are assumed to be implemented in network interfaces (NI) of the 3D NoCs. At NIs, any injecting data traffic will be compressed and receiving data traffic will be de-compressed; but it is important to note that the enhanced NIs will also have area, latency and energy overheads. The compression and de-compression processes are carried out for data packets only. In the evaluation, any data packet has a 512-bit body which is the size of a cache line. When compression is applied, the 512-bit data is broken into 32-bit pieces, which

```
Pattern 1: Full Zero Runs
        00000000000000000000000000000000 => 000

Pattern 2: 8-bit Sign-extended
        00000000000000000000000???????? => 001 + ????????
        11111111111111111111111???????? => 001 + ????????

Pattern 3: 16-bit Sign-extended
        0000000000000000???????????????? => 010 + ????????????????
        1111111111111111???????????????? => 010 + ????????????????

Pattern 4: 8-bit Data Padded with Zeros
        ????????00000000000000000000000 => 011 + ????????

Pattern 5: 16-bit Data Padded with Zeros
        ????????????????0000000000000000 => 100 + ????????????????

Pattern 6: Two Half-words, Each as a Sign-extended Byte
        00000000????????00000000???????? => 101 + ????????????????
        00000000????????11111111???????? => 101 + ????????????????
        11111111????????00000000???????? => 101 + ????????????????
        11111111????????11111111???????? => 101 + ????????????????

Pattern 7: Word Consisting of Repeated Bytes
        [Byte X][Byte X][Byte X][Byte X] => 110 + [Byte X]

Non-compressible Word
        [Word X] => 111 + [Word X]
```

Figure 3.2.1: Patterns of the frequent pattern compression.

are then encoded with the eight patterns shown in Figure 3.2.1. If the flit size is 128-bit and the compression ratio is between 4:3 and 2:1 (like the one shown in Figure 3.2.2), the original packet is composed of a header flit and 4 body flits while

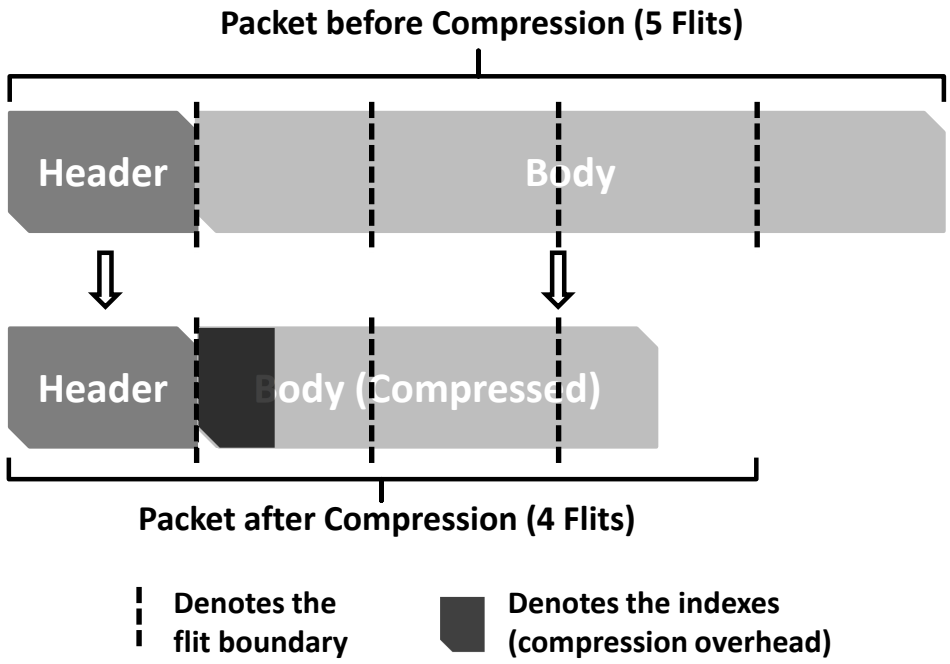


Figure 3.2.2: An example of the frequent pattern compression.

the compressed packet carries a header flit and 3 body flits. This results in a 5-flit to 4-flit packet size reduction.

As mentioned earlier, the compression process of FPC can be done in parallel for several data words at a time. As stated in [9], this compression process is only taking one cycle per data word, thus with multiple parallel encoders, the timing overhead of compression is one cycle per packet. For de-compression, since FPC is a variable length compression scheme, it is unable to carry out the de-compression in parallel. But as proposed in [21], it is able to overlap the network latency with part of this de-compression latency. In details, the receiving and de-compression pipeline is designed to work with only a fraction of a packet received.

After the first body flit containing indexes of all compressed words (the compression overhead) is received, there is a pre-computation process in order to obtain the length of compressed data before its arrival. Hence, the de-compression does not need to rely on receiving the entire compressed packet. By applying this improvement, the de-compression timing overhead can be kept within two cycles per packet. Thus, in the evaluation, one cycle of compression delay and two cycles of de-compression delay are assumed for any data packet. However, for incompressible packets, their sizes, in terms of number of flits, will be the same or even increased after the compression. This opens up another opportunity for adaptive control in order to avoid negative effects, such as increased packet latency due to having more flits or effortless de-compression.

In [21], it is recorded that with 45 nm process, the area overhead and dynamic power consumption of compressor/de-compressor circuits are 0.183 mm^2 and 0.273 W, respectively. In this work, since both the packet size and the compression/de-compression algorithm and process are the same as [21], a similar area overhead is expected.

3.2.2 Proposed Adaptive Compression for 3D NoCs

In Section 3.1 and Section 3.2.1, the 3D NoC model and its vertical bandwidth limitation are discussed. To help mitigating the vertical bandwidth limitation and making better use of FPC, an adaptive compression technique is presented for 3D NoCs. Based on FPC, this adaptive compression scheme utilizes compressibility and location based mechanisms to control the compression process while static

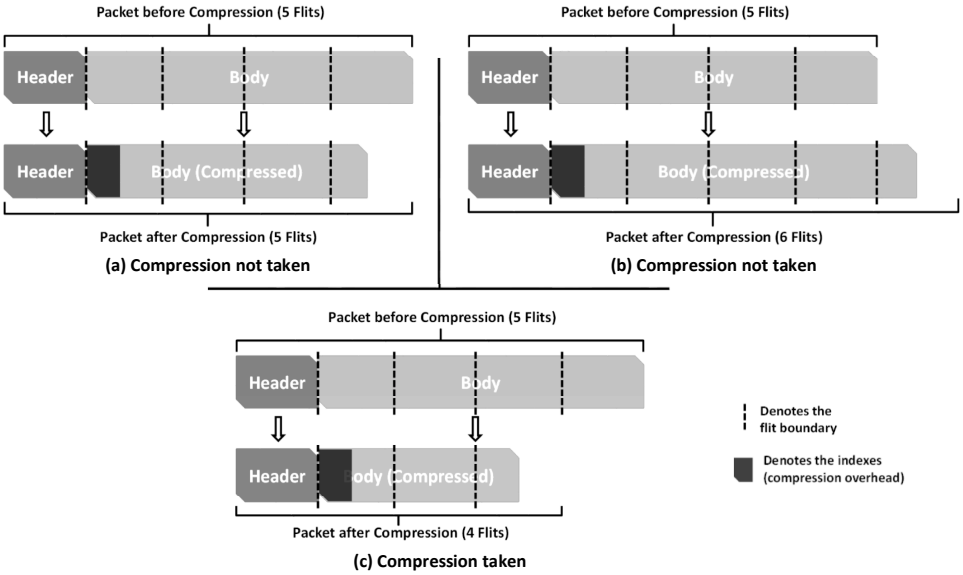


Figure 3.2.3: Compressibility-based adaptive control.

FPC employs a constant-on rule that every data packet gets compressed. For any data packet waiting to be injected to the network, two policies have been set up to determine whether the compressor should be invoked or not. There is also a third policy which aggregates these two proposed policies. These 3 adaptive policies are described in details below and their characteristics, advantages and disadvantages are summarized in TABLE 3.2.1.

- **Compressibility** based control requires the compression process, which incurs overhead of compression. The reason for proposing this policy is that negative compressibility and effortless de-compression should always be avoided. After the actual compression process, the size of the compressed packet will be known. If it is known that the compressed packet cannot

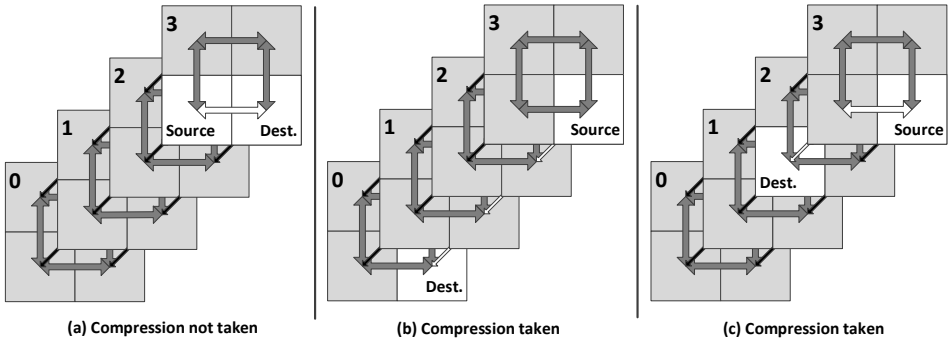


Figure 3.2.4: Location-based adaptive control.

derive any flit reduction from the original packet, then the network interface disregards the compressed packet and instead it splits and injects the original packet. With this policy, for packets whose compressibility is not good enough for any flit reduction, the timing overhead of sending more flits or carrying out an effortless de-compression can be saved when compared to static compression. However, if the data is incompressible, one cycle per packet is lost when compared to no compression. When this is the only adaptive control implemented, the compressibility is always checked in spite of the packet direction. Figure 3.2.3 gives three examples of this adaptive control and only the third case has the compression incurred since that packet has less number of flits after compression.

- **Location** based control is simple. It does not require the compression process. As shown in Figure 3.2.4, this method detects packets going across layers, such as Figure 3.2.4b and Figure 3.2.4c, and compresses them. Layer

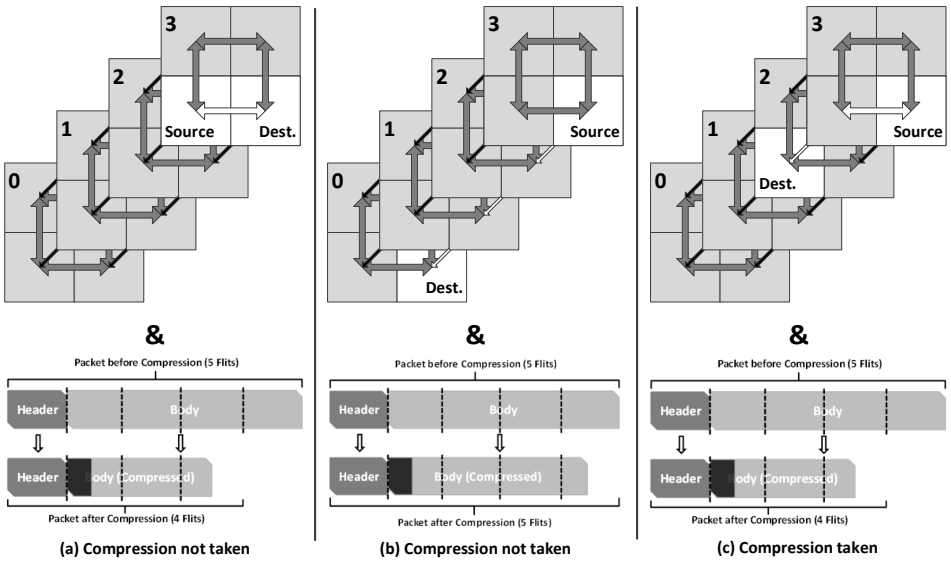


Figure 3.2.5: Compressibility- and location-based adaptive control.

crossing packets can be easily detected by checking several bits of the packet header indicating the destination node. There are two reasons for proposing this policy. Firstly, it is believed that 3D NoC will grow with increasing number of layers, which means more traffic will be layer-crossing. Secondly, if most of the compressible packets are crossing layers, compressing these traffic is more promising since they also suffer from the vertical bandwidth limitation as described in Section 3.1.

- **Compressibility and Location** based control is the logical conjunction of the above two policies. A layer-crossing packet will be examined for compressibility to determine if its compressed form is going to be injected to the network. Please note that packets traveling within the same layer will

neither be checked for compressibility nor be compressed. Like the second policy, this policy also targets at the vertical bandwidth limitation. However, it removes any negative compressibility or effortless de-compression for these layer-crossing packets and it also removes the timing overhead of the compressibility check for packets traveling in the same layer. It has one cycle of timing overhead if a layer-crossing packet is incompressible when compared to no compression. Three examples are shown in Figure 3.2.5, while only the third one has compression incurred since both conditions are satisfied.

To successfully implement this adaptive control on FPC, it is necessary to have a bit in the header indicating the compression status for all data packets. When compressed, this bit in the packet header will be set to "1", or this bit is set to "0" when the packet is not compressed.

3.3 Methodology

To quantify the effects of applying the 3 adaptive compression policies, full system simulation is employed. In this section, the simulation platform will be explained in details. Firstly, parameters of the simulation model will be covered; and secondly, a brief introduction to the workloads in simulation will be made.

For the 3D NoC model, simulation is carried out for a 16-core CMP system with shared L2 cache using the Multifacet GEMS simulator [37] based on Simics [36]. To correctly simulate traffic compression and its effect on NoCs, the de-

Table 3.2.1: Qualitative comparisons of adaptive compression policies.

	Adaptive compression policies			
	SC	AC1	AC2	AC1+2
Compression overhead per packet	3 cycles	1 or 3 cycles	0 or 3 cycles	1 or 3 cycles
Packets to compress	All	Compressible ones	Die-crossing ones	Compressible die-crossing ones
Ignored beneficial packets	None	None	Compressible intra-die ones	Compressible intra-die ones
Compressed harmful packets	Incompressible ones	None	Incompressible intra-die ones	None
Favored cases	When all packets are compressible	When there exists incompressible packets	When most of the compressible traffic are die-crossing	When there exists incompressible die-crossing packets

tailed network model of GEMS is modified. Each core has a pair of dedicated instruction/data L1 caches and the L2 cache is divided into 16 banks. The coherence model of caches includes MOESI protocol with 2 distributed on-chip directories implemented on the bottom layer. Directories are used to maintain coherence of memory hierarchies and served as memory controllers; in simulation, directory entry access costs 6 cycles, same as the L2 cache. So any L2 cache miss at a core will result in a directory access to locate the needed data, which is either in another core's L1 cache or in the main memory. The whole memory address space is interleaved across these two directories, each of which is also a channel to the main memory. The router has a fixed 3-stage pipeline, wormhole switching and 3 virtual channels; the network interface is implemented with a 2-stage pipeline. Compression always consumes one cycle of latency while de-compression takes two cycles.

The simulation parameters also assume each core has 64KB of L1 cache split for instruction and data. Each L2 cache bank is 256 KB. Three 3D topologies are evaluated. One is having eight cores per die and two stacked dies which forms a 4 by 2 by 2 3D Mesh network. The other two are 4 cores stacked as 4 layers and 2 cores stacked as 8 layers, respectively. They form a 2 by 2 by 4 and a 2 by 1 by 8 3D Mesh topologies, one by another. Note that all planar links for 3D NoCs are 128-bit wide and all vertical links are 16-bit wide. These two link widths are picked up after considering the footprint of TSVs. Footprint of a TSV is much larger than that of a wire or a driver cell. For example, a typical size of via-last TSVs ranges from 5um to 20um [30], while that of an inverter cell is only 0.57um by 2.47um in

the case of OSU's free 45nm standard cell library. Furthermore, wire-bonding and microbump are believed to be more area hungry according to [22] as mentioned in Section 3.1.

Routers in this 3D NoC model employ deterministic X-Y-Z routing and 2 more ports are needed as connections to routers at neighbor dies/wafers. Packet communication between layers assumes that each 128-bit flit is transferred over 16-bit links in 8 cycles; however, routing and arbitration for vertical going flits are not different from non-vertical going ones.

For simplicity, configurations are summarized in TABLE 3.3.1. Wormhole switching with credit-based flow control are used for both horizontal and vertical transfers. It is also assumed that the flow control signals for vertical transfers are implemented with TSVs, while those for horizontal are implemented with metal wires within the die.

In order to have a diverse performance evaluation, nine workloads with 16-core input from SPLASH-2 and NPB 3 suites are used for simulations [3, 54]. Both benchmark suites are implemented with OpenMP and their input sizes are stated in TABLE 3.3.2.

3.4 Results

Depending on different 3D topologies, memory access characteristics, on-chip bandwidth requirements and compressibility of workload, traffic compression on

Table 3.3.1: System parameters.

Component	Parameter
Processors:	16
L1 Cache:	Each core has a total of 64KB of private L1 cache (split I and D), which is 4-way set-associative and has 64 bytes per line and 1 cycle of access latency.
L2 Cache:	Shared L2 cache divided into 16 banks. Each bank is 256KB, 16-way set-associative and has 6 cycles of access latency.
Memory:	4GB of DRAM with 160 cycles of access latency.
Topology:	16 nodes organized in three 3D Mesh topologies, 4 by 2 by 2 layers, 2 by 2 by 4 layers and 2 by 1 by 8 layers.
Network Interface:	2-stage pipeline for splitting packets into flits and flit injection; and 2-stage pipeline for flit reception and combining flits into a packet. The compression/de-compression circuits are implemented here.
Router:	3-stage pipeline with X-Y-Z routing, wormhole switching and 3 virtual channels.
Link:	Uneven link width is implemented; the planar link width is 128-bit and the vertical link width is 16-bit.
Compression Overhead:	For all compression methods, compression takes 1 cycle while de-compression takes 2 cycles. For compressibility based adaptive policy, the compressibility check is 1 cycle. For location-based adaptive policy, the destination node detection does not cost any additional cycle. Similarly for compressibility and location based adaptive policy, the compressibility check takes 1 cycle but it is only for packets which travel across layers and this destination node detection does not take any additional cycle.

Table 3.3.2: Benchmark programs and inputs.

Application	Input
fft	2^{18} complex data points
ocean, contiguous	grid of 18×18
radix	1048576 keys, radix of 1024
raytrace	head, scaled down by 16
BT	grid size of $12 \times 12 \times 12$, 60 iterations, time step of 0.01
EP	2^{24} random number pairs
LU	grid size of $12 \times 12 \times 12$, 50 iterations, time step of 0.5
MG	grid size of $32 \times 32 \times 32$, 4 iterations
SP	grid size of $12 \times 12 \times 12$, 100 iterations, time step of 0.015

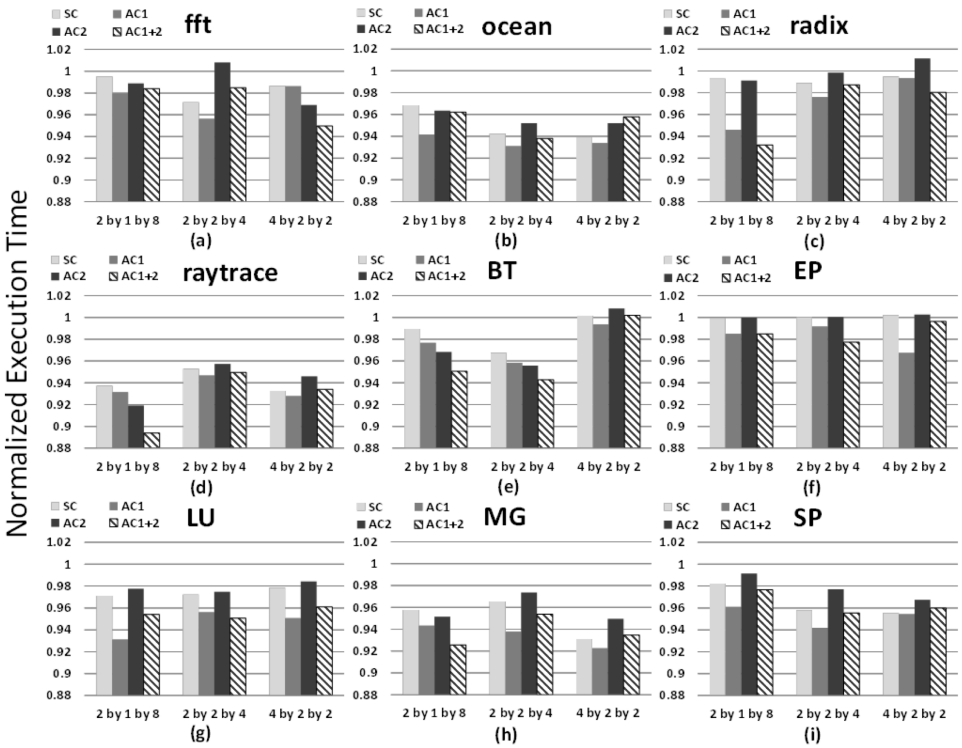


Figure 3.4.1: Normalized execution time with static/adaptive compression on 3D NoCs.

3D NoCs can bring several benefits. In this section, how these benefits look like in practice is going be made clear. The normalized execution time for 3D NoCs under two compression schemes, static and adaptive will be quantified and discussed. Note that for adaptive compression, each policy is applied separately. In total, there are four sets of results under static compression (SC), compressibility-based compression (AC1), location-based compression (AC2) and the conjunction of AC1 and AC2 (AC1+2). These results are obtained with normalization to execution time under no compression (NC) and they are presented in Figure 3.4.1 with each histogram representing a workload.

Firstly, static traffic compression on 3D NoCs is fairly effective. Of the 27 cases (9 workloads with 3 topologies) simulated, only 4 of them show zero or negative performance improvement, which means for these cases, the overhead of compression is not well covered by the amount of network latency reduced. These 4 cases are, SC of *BT* on 4 by 2 by 2 in Figure 3.4.1e and SCs of *EP* on 2 by 1 by 8, 2 by 2 by 4 and 4 by 2 by 2 in Figure 3.4.1f.

Secondly, adaptive control of traffic compression is more effective than SC. AC1 outperforms SC for all tested workloads and configurations. This is supported by the fact that if compression is beneficial, then AC1 is the same as SC, while if compression is not carried out because of it results in more flits or no flit reduction, then one cycle is wasted at the compressibility check, but 2 cycles are saved at decompression and maybe more latency are saved at the network. It is found that the improvement ranges from 1 to 5%, thus avoiding incompressible packets is very useful. As proposed, AC2 performs better than SC with more layers. With topology of 4 by 2 by 2, AC2 outperforms SC in only one case, between AC2 and SC of *fft* on 4 by 2 by 2 in Figure 3.4.1a; but this number climbs up to 6 with topology of 2 by 1 by 8. The 6 cases are, *fft* in Figure 3.4.1a, *ocean* in Figure 3.4.1b, *radix* in Figure 3.4.1c, *raytrace* in Figure 3.4.1d, *BT* in Figure 3.4.1e and *MG* in Figure 3.4.1h. Similarly, AC1+2 also outperforms SC with more layers; it can also be noted that because of avoiding unnecessary compression which is harmful on layer-crossing traffic, AC1+2 is better than SC for all workloads with topology 2 by 1 by 8.

Thirdly, between AC1 and AC2, AC2 misses chances of compression for traffic travels within layer and it also suffers from unnecessary compression for packets

going across layers. For these two reasons, AC1 is generally better than AC2 but with topology of 2 by 1 by 8, it is observed that AC2 outperforms AC1 in two cases as for *raytrace* and *BT*. This means the benefit gained by compressing layer-wise packets with AC1 does not compensate for its compression and de-compression overhead, while AC2's gain from compressing layer-crossing packets well exceeds its unnecessary compression. Another reason is that with more layers, it is less possible for AC2 to lose chances to compress data within a layer.

Finally, after combining the two policies, it is seen that AC1+2 outperforms AC2 in almost all cases with the same reason as AC1 outperforms SC. This means layer-crossing packets also favor the compressibility check, which improves AC2 by denying all incompressible layer-crossing packets. Another important observation is AC1+2 outperforms AC1 in two cases under topology of 4 by 2 by 2, which are *fft* and *radix*. However, this number grows to 3 for topology of 2 by 2 by 4 with *BT*, *EP* and *LU*; and it further grows to 4 for topology of 2 by 1 by 8 with *radix*, *raytrace*, *BT* and *MG*. It can be seen that AC1+2 also performs better while the chip is implemented with more layers. This is the same as AC2; if having more layers, AC1+2 also loses less chances of traffic within a layer. One more observation is, in some cases, performance improvement is not larger with configurations having more layers under route-based adaptive policies. For example, *ocean* with 4 layers under AC1+2 obtains the best performance improvement when compared to the cases with 2 and 8 layers. An explanation behind this is, more critical packets are well compressed in the 4 layer case since it is the criticality of a packet which determines if compression on such a packet is going to benefit the performance.

More specifically, for both 2 by 2 by 4 and 4 by 2 by 2, AC1 has been recorded a performance improvement of up to 7% over NC, and is better than SC, AC2 and AC1+2. This 7% of improvement with AC1 is seen in Figure 3.4.1b, Figure 3.4.1d and Figure 3.4.1h for *ocean* on 2 by 2 by 4, *raytrace* on 4 by 2 by 2 and *MG* on 4 by 2 by 2. However, with 2 by 1 by 8, AC1+2 is seen to have the best performance improvement of up to 11% over NC in Figure 3.4.1d for *raytrace*.

3.5 Summary and Discussions

In this solution, it is evaluated that how adaptive traffic compression affects system performance for CMPs implemented with 3D NoCs. It is also presented what difference on performance is made with adaptive schemes of traffic compression proposed in this work. In a bandwidth limited situation like a CMP with 3D NoCs having multiple connected layers, adaptive traffic compression with location-based control or with both compressibility and location based control is very promising if the number of layers continues to grow.

Furthermore, according to the evaluation result, if frequent pattern compression is to be utilized, then compressibility check is a must since it is always better than static compression. Secondly, if a 3D implementation has many layers and few cores per layer, AC1+2 is very efficient since it targets specifically at the vertical bandwidth limitation and most of the traffic are layer-crossing. Finally, although the improvements vary case by case, these results are quite conservative since the simulation are carried out with Simics whose processor model is in-order and a relatively smaller input size is used for the workloads. In practice, modern pro-

cessor cores are generally more advanced with a higher bandwidth requirement. In consequence, a more promising improvement than these shown results can be expected if a similar 3D design has the adaptive FPC implemented.

Regarding the coherence protocols, traffic compression works better with directory based protocols, since they result in less control packets which are not compressible at all. As for having more numbers of cores, it is obvious that stacking more processor cores will benefit from adaptive traffic compression since more traffic will suffer from the bandwidth limitation.

The council of three gave good counsel.

Anonymous

4

Latency Reduction through In-router Multicasting: Predict-more Router

Of all parameters that affect system performance and power consumption, router latency is very critical as inter-node communications in NoCs are carried out on a hop-by-hop basis through routers. Many attempts have thus been made to efficiently shorten router latency. The technique with the most aggressive speculation so far (prediction router or PR) [38] works by predictively routing packets to pre-determined outputs before route computation is finished. For the application traffic we test, about 65% of the prediction this technique em-

ploys matches the computed route even with the best algorithm. This means, on average, only about 65% of the packets may succeed predictive routing before taking contentions into consideration yet.

In this chapter, a new low latency router is proposed for NoCs by improving PR's prediction accuracy through "the Wisdom of Crowds". The essence of this new router (predict-more router or PmR) is to carry out multiple route predictions for one incoming packet with different prediction algorithms. This simply increases the prediction accuracy (for more than 15%, on average) and in consequence helps more packets to succeed predictive routing which results in more opportunities of latency reduction. Different from PR, PmR has multiple predictors under different algorithms working at the same time and a switch crossbar which allows one-to-many traversals of the same flit. This simple change enables multiple predictive switch traversals of a packet following multiple predictions. Like PR, PmR also maintains modularity and portability as a standalone design and fits well in any NoC with wormhole or virtual channel routers.

4.1 Motivation

Although PR is a capable technique of achieving single cycle flit transfer, it can be found that, for different applications, its predictions hit from 46% to 80% under algorithms Matsutani *et al.* [38] proposed and evaluated (more details in Figure 4.1.1, evaluation conditions are stated in Section 4.3 while the prediction al-

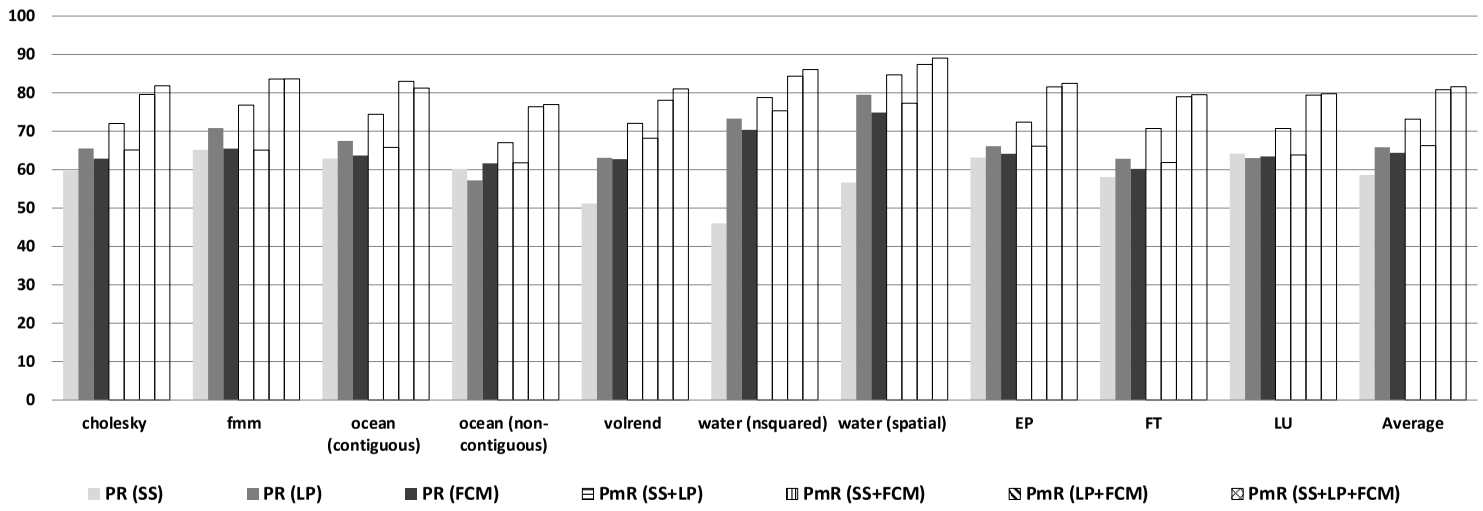


Figure 4.1.1: Prediction accuracy.

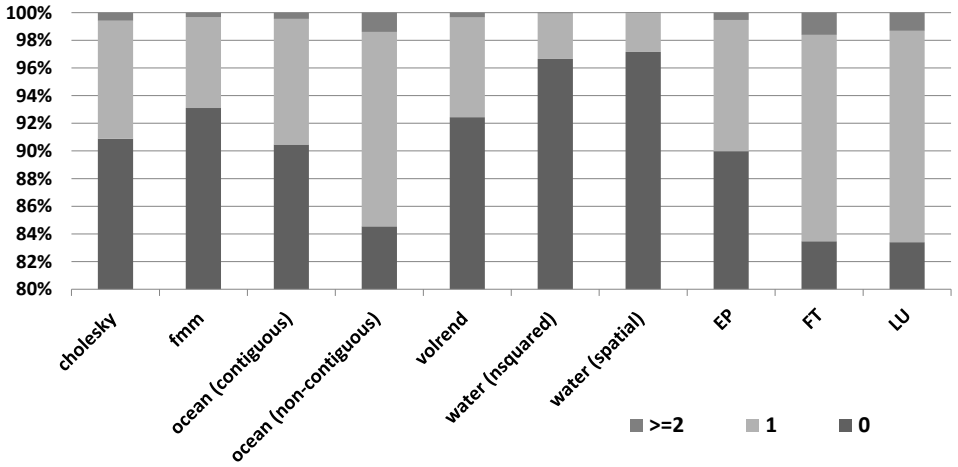


Figure 4.1.2: Fractions for different numbers of concurrent flits arriving at routers each cycle.[2]

gorithms are described in Section 4.2.1). With the two better-performing algorithms, LP and FCM, prediction on average hits around 65% of the time, so there are still 35% of the packets which have no chance to be accelerated. This leads to the potential of improving the prediction accuracy. A very simple approach is, to predict with multiple algorithms at the same time. According to evaluation (see Figure 4.1.1), this can increase the prediction accuracy by more than 15% on average with the best combinations of algorithms (LP+FCM or SS+LP+FCM). Although this improvement looks good so far, it does not necessarily mean that 15% more packets are able to get accelerated. In fact, the more predictions are carried out, the more contentions may be created within the router. More contentions will simply degrade the efficiency on arbitration so that this desired improvement may not be realized. Fortunately, with another evaluation (see Figure 4.1.2), it is

[2]The y-axis of this figure starts from 80%.

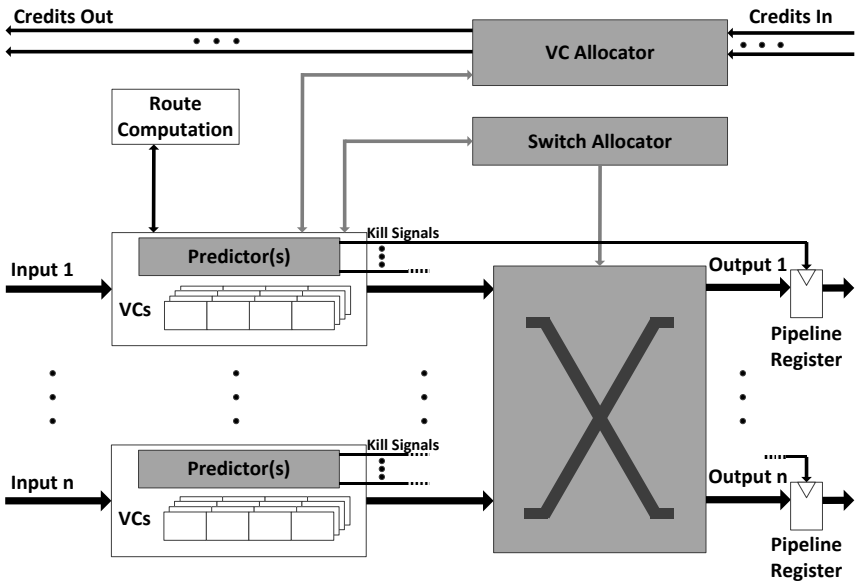


Figure 4.2.1: Architectures of the prediction router and the predict-more router.

found that at most of the time during execution, a router is either idle or transmitting only a single flit for the workloads evaluated. This means most of the time the benefit provided by having concurrent predictions on a packet can be simply realized. In a word, the primary goal of PmR is to accelerate more packets through more accurate predictions if the internal bandwidth of a router allows.

4.2 The Predict-more Router

In this section, how PmR is motivated and designed are covered. Discussions are also made on the architecture of PR and PmR, qualitatively.

4.2.1 Design

PmR is basically a PR with three enhancements. The first one is to enable multiple predictions on one packet. The second one is to enable one-to-many traversals on the crossbar switch. The last one is about how contentions are resolved in prediction. More details of these enhancements are presented when looking at how individual components of a PmR are designed. In Figure 4.2.1, the architectures of both PR and PmR are shown. Components that are in gray are the ones which differentiate PmR from PR.

- **Input units:** Predictors reside in the input units. In order to carry out multiple predictions at a time, multiple predictors per input port are necessary. The three prediction algorithms are static straight (SS), latest port (LP) and finite context method (FCM). SS is a simple algorithm targeting at dimension-ordered routing. It simply predicts that a packet travels along the same dimension. At corner nodes, SS does not work; while at edge nodes, prediction can only be made along one dimension. For this reason, not all packets can be accelerated with SS and the efficiency of SS is the lowest of the three. Because of its simplicity, the power overhead of SS is also the lowest. LP simply predicts that an incoming packet from an input is going to be routed to the same output as the previous packet comes from this input. LP works well if there is a lot of repeated traffic. A single history buffer is required to implement LP as it is used to store the previous routing result. FCM predicts by taking the most frequently used output (oth-order finite context

method [18]). For an N-radix router, an FCM predictor requires a history table of N-1 items to keep a record of the routing history. It has the most complex and expensive design of the three.

- **Virtual channel allocator and switch allocator:** Both allocators in PmR are very similar to PR. In PR, when a VA or SA request is placed following a prediction, it is considered a request with low priority since it is speculative. This rule is simply inherited from PR to PmR. So a virtual channel (VC) or a slot of the crossbar switch can be reserved if it belongs to an output predicted at any input port and it is not allocated through non-speculative VA or SA operation, but it is not allocated before any actual packet comes. If there is only one packet, then only one input port's prediction is meaningful at this cycle, so allocations are simply written to this packet as it is the only winner of this VA or SA operation. One thing that is not discussed in the work by Matsutani *et al.* [38] is how contention from predictions is resolved. The solution is, if two or more packets come to the router from different inputs which have an overlapped prediction, prediction accuracy from the past is used to resolve such a contention caused by prediction. For example, if this output is predicted with algorithm A (or combination A of multiple algorithms) at input 1 while it is also predicted by algorithm B (or combination B of multiple algorithms) at input 2 and if A at input 1 performs better than B at input 2 in the past, then this contention is resolved so that the packet at input 1 wins this allocation. This process can be complex but it can actually be done before any packet comes since past prediction

accuracy is always known.

- **Crossbar switch:** Multicast support needs to be implemented in the crossbar switch, which means, with an N by N crossbar switch, N^2 control signals are needed to help invoking a multi-destination traversal.

- **Kill circuit:** PmR may result in multiple copies of a head flit traversing the crossbar switch to reach multiple predicted outputs and it is necessary that only the correctly routed one leaves the router. So the same kill circuit of PR is also required by PmR. The difference is, multiple flit killing operations may be needed for one prediction in PmR.

4.2.2 Architectural Discussions

In this subsection, architectural discussions on PmR and PR are presented. Purpose of such discussions is to identify the pros and cons of PmR, qualitatively. Discussions are made in three aspects: router timing, routing efficiency and critical path delay.

- **Router timing:** The timing of PmR is the same as PR as shown in Figure 2.2.1e. If a prediction hits and a VC and a time slot of the crossbar switch are successfully reserved, it takes 1-cycle for both PR and PmR to transmit a flit. Otherwise, if a prediction misses or a reservation fails because of contention, a flit is then processed through the conventional datapath.

- **Routing efficiency:** Talking about routing efficiency, it is obvious that PmR causes more prediction-incurred contentions since multiple predictions by

all input ports create more overlapped predictions. But the enhancement added to the allocators has very much alleviated this problem. Firstly, in case if the network load is low (as in Figure 4.1.2), there will be very few contentions since only one prediction is meaningful at such a moment. Secondly, if two or more flits come concurrently, prediction accuracy from the past helps the most potential packet to win the arbitration. In terms of PR, there also exist overlapped predictions, but much fewer. The original PR work did not discuss this issue so a convention is chosen in both PR and PmR, that is, prediction accuracy is used to resolve such contentions. Furthermore, the way on how contention is handled by PmR reveals an important insight, that is, PmR's improvement on prediction accuracy by having multiple predictions can be effectively utilized when network load is low and PR should be more efficient than PmR in routing efficiency when network load is high. Additionally, predictions with two algorithms should be more efficient than predictions with three algorithms when network load is high, since the former create less overlapped predictions.

- **Critical path delay:** In terms of critical path delay, PmR is very similar to PR if only considering gate delay. For PR with SS, it is evaluated that its critical path delay is longer than a CR by 5.6% mainly because of more state controls following predictions [38]. For the same reason, the predictive switch traversal stage of PmR is also longer. However, inside a virtual channel router, VA is actually the longest stage [19, 45]. Since half of the VA operation is already done through reservation for both PR and PmR follow-

Table 4.3.1: System parameters.

Component	Parameter
Number of cores:	16
Topology:	4×4 mesh
Processor:	4 GHz, in-order
L1 I/D cache:	32 KB per core, 4-way set associative, 1 cycle access latency
L2 cache:	256 KB per Bank, 16-way set associative, 6 cycles access latency
Cache line size:	64 Bytes
Main memory:	4 GB, 160 cycles access latency
Coherence protocol:	MOESI, directory
Link:	128-bit, 1 cycle traversal
Packet:	128-bit control, 640-bit data
Router:	1 GHz, virtual channel router
Virtual channel:	4 per virtual network
Virtual network:	3 per physical link
Routing algorithm:	X-Y routing
Process technology:	32 nm
Vdd:	1 V

ing a prediction, the critical path delay of PR and PmR should be the same as a conventional router.

4.3 Methodology

In this solution, various evaluations on performance and power are carried out with GEMS [37] and Simics [36] extended with the network model from Garnet [7] and the network power model from Orion [29]. To evaluate performance, the source code of GEMS and Garnet are modified to provide cycle-accurate timing models of PR and PmR. PR is evaluated with the three prediction algorithms as mentioned in Section 4.2.1. For PmR, predictions are set up with these three algorithms being used at the same time. For example, SS+LP means a prediction with both SS and LP while SS+LP+FCM uses all three algorithms in one predic-

Table 4.3.2: Benchmark programs and inputs.

Application	Input
cholesky	tk29.O
fmm	16384 particles
ocean, contiguous	grid of 258×258
ocean, non-contiguous	grid of 258×258
volrend	head
water, nsquared	512 molecules
water, spatial	512 molecules
EP	2^{25} random number pairs
FT	grid size of 128×128×32, 6 iterations
LU	grid size of 64×64×64, 250 iterations, time step of 2.0

tion. For the power evaluation, the kill circuit is not included. For predictor power, only the memory components inside them are considered. For each LP predictor, a power model of a 3-bit register is used while for each FCM predictor, an 8-bit register file is taken. The number of registers in the register file equals N-1 if the predictor is implemented in an N-radix router. The evaluation conditions are summarized in TABLE 4.3.1.

In all evaluations, a 16-tile mesh network with 128-bit links are assumed. Each tile has an in-order processor core, a bank of L2 cache/a directory. Each corner node also has a memory controller. The access latencies for L1 cache, L2 cache and main memory are 1 cycle, 6 cycles and 160 cycles, respectively. These components are connected to a router individually and network traffic travels through routers and links. So low latency routers such as PmR can accelerate remote L1, L2 and main memory accesses; and the former two are best candidates since main memory access latency is much larger than network latency. Figure 1.1.1 in Section 1.1 illustrates the schematic view of this simulated system and what a tile is composed

of. The entire network is set to have three virtual networks to support the MOESI directory coherence protocol which has three classes of traffic. Each router has a maximum of 6 ports and each port has four virtual channels while each virtual channel has four 128-bit buffers. More details are presented in TABLE 4.3.1. The evaluations are based on both synthetic and application traffic. The types of synthetic traffic used are uniform random, bit compliment and tornado. All packets in synthetic traffic consist of 5 flits. Applications are chosen from the NPB 3.3 [3] and SPLASH-2 [54] benchmark suites. TABLE 4.3.2 lists these applications and their inputs.

4.4 Results

In this section, evaluation results on various router designs including PmR are presented and discussed in terms of their performance and power consumption. Two specific points on performance evaluations with PmRs that will be concentrated are if the prediction accuracy improvement is well compensated for and how the bandwidth consumption of workloads makes a difference for PmR's effectiveness.

4.4.1 Prediction Accuracy and Routing Efficiency

The prediction accuracy is briefly discussed in Section 4.1 already. Looking at Figure 4.1.1, in more details, there are four other findings about prediction accuracy. Firstly, for all workloads evaluated, LP+FCM and SS+LP+FCM are the best performing combinations of algorithms. Secondly, SS+LP provides a mod-

erate improvement in prediction accuracy. Thirdly, the three combinations of algorithms mentioned above are always better than a single prediction algorithm in prediction accuracy. Finally, SS+FCM is only marginally better than FCM and it is even outperformed by LP for half of the workloads. The reason is, prediction results of SS and FCM mostly overlap, which means, outputs on the same dimension of inputs are also the most frequently routed targets.

Figure 4.4.1 provides another angle of looking at the prediction accuracy. It records the amount of routing that is successfully accelerated with predictions, which means, the prediction does hit and the required router resources are also successfully allocated to this prediction. One observation in this figure is, LP+FCM is the most effective combination of algorithms in routing efficiency since not only providing the best prediction accuracy, it also enables the most accelerations through prediction. Hence, it is the most productive at utilizing router's internal bandwidth. A second observation is, although SS+LP+FCM is the best performing combination of algorithms in accuracy, it does not hold the top position in terms of routing efficiency, since it creates the largest number of contentions in prediction by having all three algorithms. Thirdly, the numbers in this figure are always smaller than the corresponding prediction accuracy in Figure 4.1.1, and this difference comes from the fact that some predictions do hit but they fail to acquire necessary router resources to speed up the routing process.

PR with SS performs poorly in terms of routing efficiency despite of its prediction accuracy is about 60% on average. The reason is, not all packets are able to be predicted with SS (as described in Section 4.2.1).

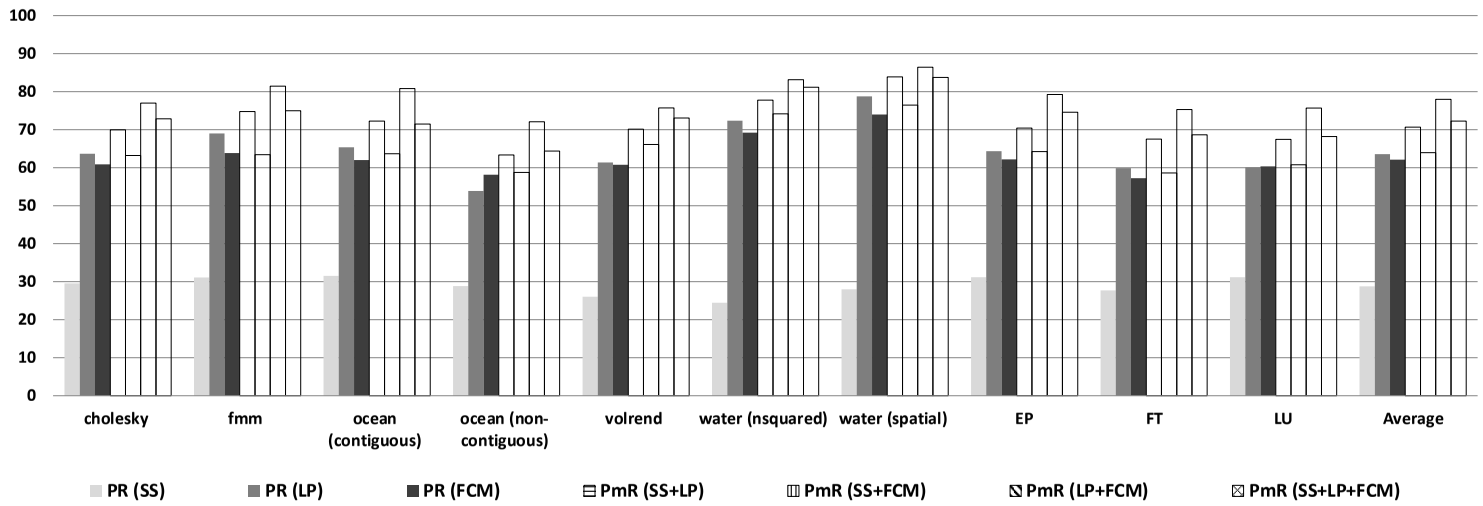


Figure 4.4.1: Percentage of packets successfully accelerated with predictions.

4.4.2 Synthetic Performance

Figure 4.4.2 presents the average latency per flit with varying injection rates under different synthetic traffic patterns. For simplicity, only four cases are shown, which are CR, PR with FCM, PmR with LP+FCM and PmR with SS+LP+FCM. This figure has brought three observations. Firstly, all three cases with prediction work pretty well at reducing the latency with low injection load. Secondly, it can seen that all prediction cases still work with high injection load as they only saturate with more injections than CR. Thirdly, PmR is at least as good as PR in terms of both latency reduction (marginally better, but hard to see in Figure 4.4.2 since the best algorithms are chosen) and bandwidth consumption (nearly the same as PR or even smaller). Although PmR creates more contentions in prediction, it seems that the prediction accuracy based approach on resolving these contentions works pretty well (at least, for these synthetic traffic patterns).

4.4.3 Application Performance

For network latency as shown in Figure 4.4.3, it can be found that PmR with LP+FCM is the best at latency reduction, on average. It outperforms all PRs and while compared to other PmRs, there are only two cases (*cholesky* and *ocean (non-contiguous)*) it is outperformed by SS+LP+FCM. On average, the best performing PmR (with LP+FCM) outperforms the best PR (with LP) by 3.8% in latency reduction.

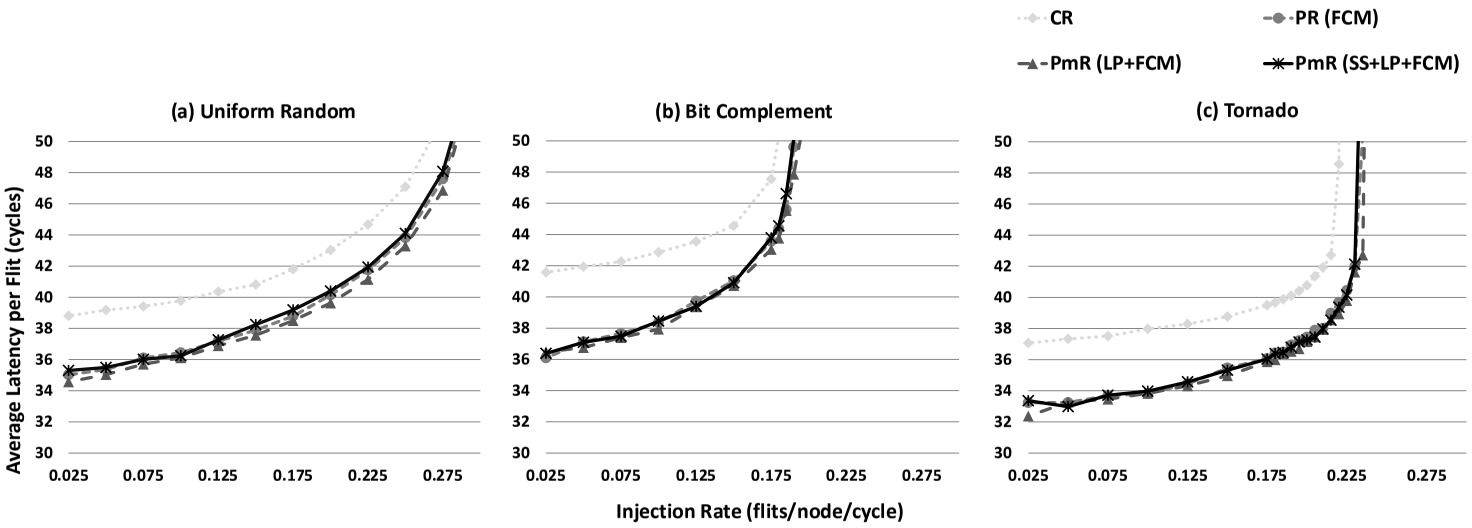


Figure 4.4.2: Network latency with synthetic traffic.

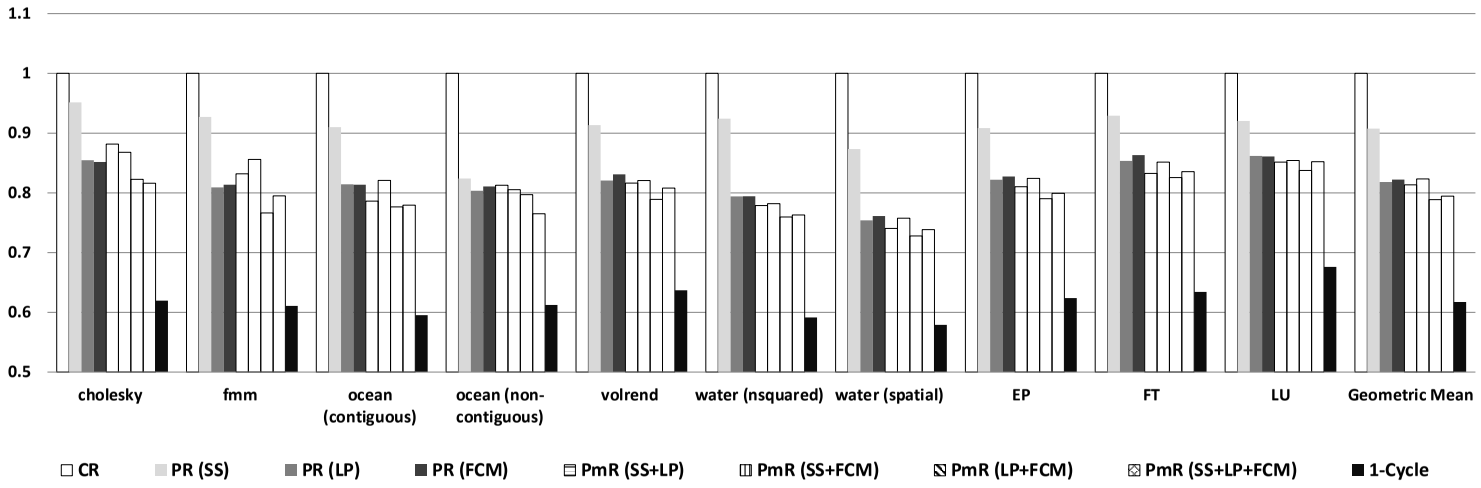


Figure 4.4.3: Normalized per-flit latency.

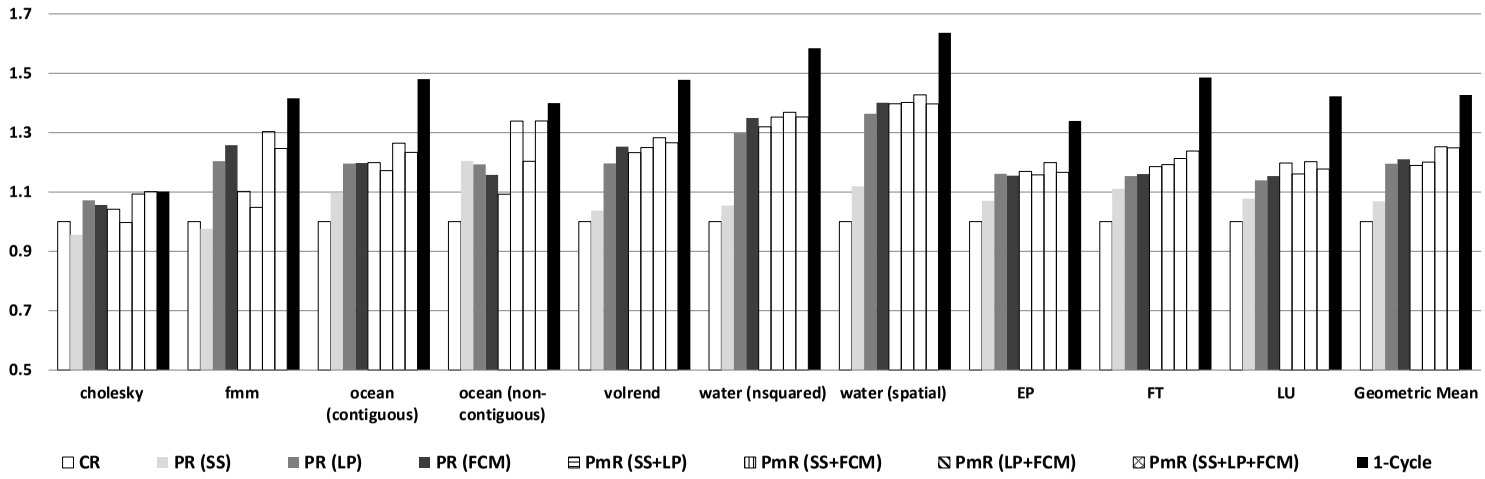


Figure 4.4.4: Normalized system speed-up.

For system speed-up as shown in Figure 4.4.4, the result is very similar to network latency. PmR with LP+FCM is the best performing combination of algorithms on average. It outperforms all other PRs and PmRs except 3 cases (two with SS+LP+FCM for workloads *cholesky* and *FT*; one with both SS+FCM and SS+LP+FCM for workload *ocean (non-contiguous)*). On average, the best performing PmR (with LP+FCM) outperforms the best PR (with FCM) by 3.5% in system speed-up.

Results in Figure 4.4.3 and Figure 4.4.4 do not match each other strictly, this is not odd since another factor, the criticality of packets, actually determines if network latency reduction affects the system performance [35].

Working set size also plays an important role here. For workloads that consume more memory (such as *EP*, *FT* and *LU* [5, 27, 53, 54]), improvement with PmRs on system performance is relatively smaller. More L2 cache misses results in more main memory accesses whose latencies are much larger than network latencies. Workload *cholesky* is an exception. The system speed-up is relatively low for it with even ideal 1-cycle router. This means, *cholesky* is not as sensitive on network performance as others.

One more observation is, PmR is well behind the ideal 1-cycle router in both network and system performance. This means, although multiple predictions help improving the prediction accuracy, there is still some traffic that is not benefited from better prediction.

As shown in Figure 4.4.5, PmR in general consumes more power than CR and PR. But the difference varies considering what algorithms are used. For both PR

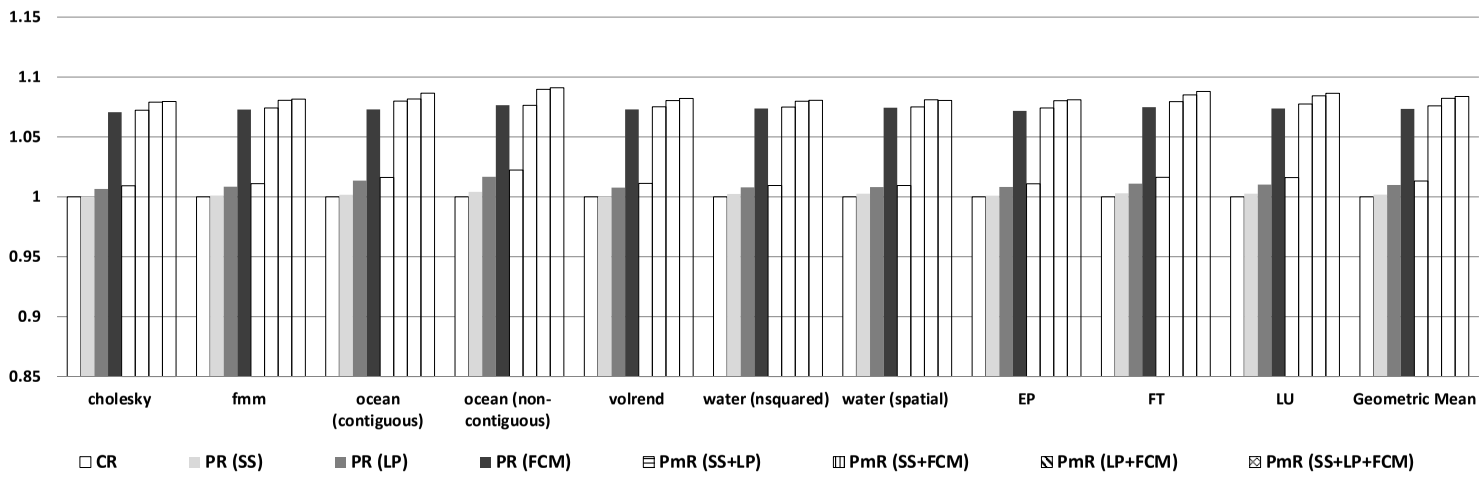


Figure 4.4.5: Normalized network power consumption.

and PmR, power overhead actually comes from the dynamic power spent at the arbitrators and the crossbar switch following predictions and the memory components inside LP and FCM predictors. PmR consumes more power since it incurs more utilization on router components and predictors. In evaluation, the best PmR (with LP+FCM) only consumes marginally more power than the best PR (with FCM) since LP has very little power overhead. Finally, when compared to CR, PmR with LP+FCM outperforms it in speeding up the system by 25.2% with a power overhead of only 8.2% on the network.

4.4.4 Summary and Discussions

In this chapter a new low latency on-chip router named predict-more router is proposed which simply improves the prediction accuracy and the number of successful predictive routing of the original prediction router by allowing multiple predictions on one packet.

As a summary, PmR is better at latency reduction than PR while it also incurs more contentions in prediction. But there are two reasons that these extra contentions do not hurt bandwidth. Firstly, link bandwidth consumption is not affected since extra bandwidth consumption only happens inside a router for both PR and PmR. Secondly, resolving these contentions with prediction accuracy works well so internal bandwidth of the router is not a concern, too.

With the evaluations carried out, it is found that PmR (with the best combination of algorithms) improves the prediction accuracy by more than 15%, versus PR with its best algorithm. Along with this higher prediction accuracy, significant

improvement on routing efficiency can also be spotted, over 14% more packets are seen accelerated through more predictions. Finally, 3.8% improvement on latency reduction and 3.5% more speed-up on system performance are also identified, on average. These benefits are brought with a nearly negligible cost in power consumption if the best PmR (LP+FCM) is compared with the best PR (FCM).

To extend the above summary, a few qualitative discussions will be made, regarding network topology and core scaling.

Firstly, topology makes difference for the radix of routers since the radix of routers can affect the prediction accuracy. According to the PR work [39], prediction accuracy with fat tree topology is much lower than with mesh.

Secondly, there are two forms of core scaling which have opposite influences on PmR. If the number of tiles scales with the number of cores, PmR should still be effective or even better since with such a scaling traffic travels longer distance in the network, which means more chance to be accelerated. However, if the size of the network stays while the number of cores scales up (a denser design), it is hard for PmR to maintain its effectiveness since the network is simply more stressed and with more cores prediction accuracy can be exacerbated as the radix of routers will be increased.

When in doubt, use brute force.

Kenneth Lane Thompson

5

Latency Reduction through In-router Multicasting: McRouter

Going further from Predict-more Router, this chapter introduces another low latency on-chip router design utilizing a technique called multicast-within-a-router or ***McRouter*** for short.

The essence of McRouter is to achieve latency reduction by allowing itself to consume some additional bandwidth within the router (not the on-chip network). Different from any on-chip routers designed so far, McRouter has a switch crossbar which allows multicast operations. This simple change together with control cir-

cuitry to check if the internal bandwidth of a router is sufficient to carry out a multicast, helps hiding the route computation and arbitration delays which enables a single cycle transfer of flits. Different from most speculative routers [39, 45], McRouter more aggressively utilizes its internal bandwidth in order to shorten the router's per-hop latency. In contrast to single cycle routers employing look-ahead routing [41], McRouter also maintains its modularity and portability as a standalone design and fits well in any NoC having wormhole or virtual channel routers with virtually any routing algorithm.

5.1 Motivation

Route computation or RC is important within a NoC router. This importance does not come from its complexity, which is actually based on the routing algorithm. It is important since all other operations like VA, SA and ST depend on the result of RC. With the low latency routing techniques reviewed in Subsection 2.2.2, two have successfully helped hiding RC delay and getting RC result earlier. The first one is LAR. By computing the route at preceding routers, it allows one to remove the RC delay from the per-hop latency of a router. This technique has been extended to allow single cycle flit transfer in [24, 41]. However, these LAR based ideas all suffer from a common design issue. The portability and modularity of such routers are violated, since a router with LAR requires assistance from the upstream routers or network interfaces to carry out RC. Moreover, in some single cycle router with look-ahead routing [41], to even remove the decoding of RC results from the critical path (so that RC delay is completely hidden), extra wiring

between routers are required to carry one-hot encoding of the RC results. Similarly, another low latency router proposed also has such design issue since it depends on sending a packet called advanced bundle to set up the control for a flit to traverse a future router [34].

Another technique, which forms a standalone router design (free from above mentioned design issue) and is able to mostly hide RC delay, is PR [39]. It is so far the most aggressive speculation found in router designs. By predicting RC result well ahead of an arriving packet, it also helps removing the arbitration delays from the per-hop latency of a router if prediction hits. When the prediction misses, extra bandwidth is consumed as a mis-routed flit has to traverse its data path. In evaluation, it is found that prediction hits around 65% even with the best algorithm; therefore, about 35% of the packets cannot be conveyed in one cycle at all.

Regarding the portability and modularity of a router design, LAR is not a good option. This causes problems when one replaces CR with LAR in a design. For speculation on RC (such as prediction), the following two important concerns appear: (1) whether such a speculation is necessary and (2) if it is possible to trade in more bandwidth to make the speculation more accurate.

Before going further into these particular concerns mentioned above, an evaluation is set up with application traffic from NPB-3.3 [3] and SPLASH-2 [54] benchmarks to understand how internal bandwidth of a router, especially for its crossbar switch, is utilized. A 16-tile mesh NoC based CMP is used where each tile is composed of a core, an L2 cache bank and sometimes a memory controller (see Figure 1.1.1). The result is shown in Figure 5.1.1 where the detailed condi-

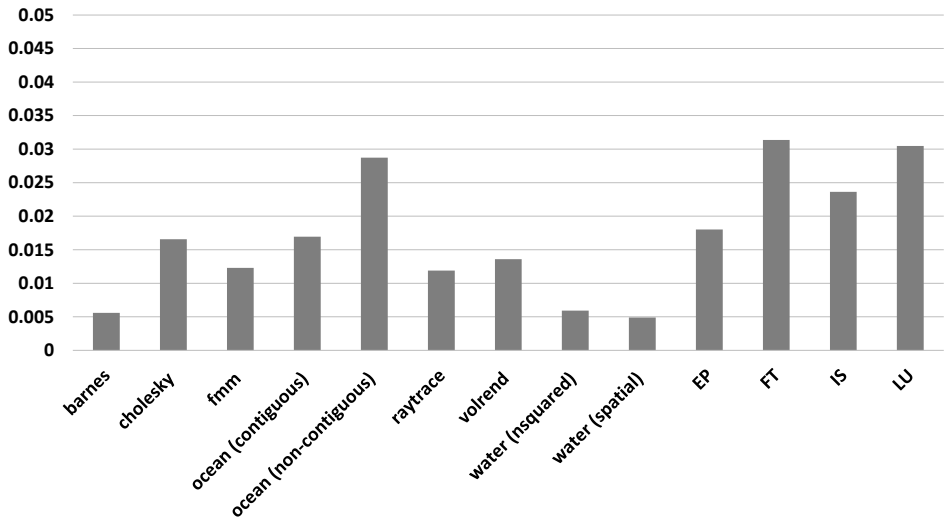


Figure 5.1.1: Average link utilization on a 16-core CMP connected with 4 by 4 mesh network.

tions of these evaluations are covered in Section 5.4. As can be seen from the figure, even for the worst case, a link is only injected with roughly 0.031 flits/cycle on average (this is equal to 470 MBytes/link/sec). Considering a router with 6 input/output ports (4 ports connected to 4 neighboring routers, 1 to a core, and 1 to a bank of L2 cache), it roughly translates to 0.2 flits/crossbar/cycle. This means there is plenty of bandwidth inside a router which could in turn be utilized to help shortening its latency for these parallel workloads. Such an observation is also supported by a recent work [49], although its evaluation has 64 processors connected with a 16-tile mesh topology and is taken from another set of workloads (PARSEC benchmark suite [15]).

As a consequence, this observation tells that, like many other components in a computer system, a NoC router is designed for the worst case. Following this,

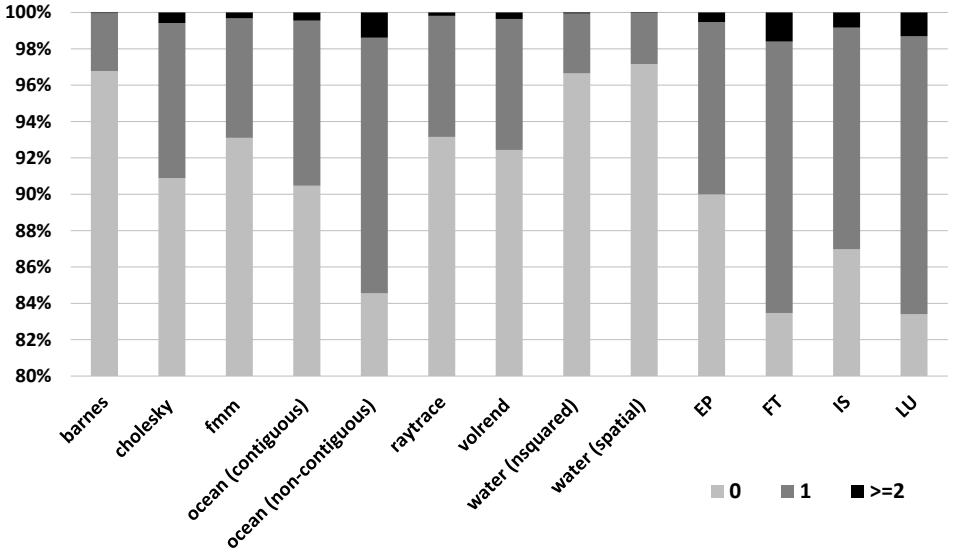


Figure 5.1.2: Fractions for different numbers of concurrent flits arriving at routers each cycle.[3]

there are simply two facts that can be identified from Figure 5.1.2 that is retrieved with the same evaluation parameters as Figure 5.1.1. This figure presents the fraction of different numbers of concurrent flits arriving at routers (the cases for more than 2 concurrent flits arriving at routers are aggregated). Firstly, it is shown in this figure that a router can be inactive for most of the cycles. Secondly, when active, a router may be transmitting no more than a single flit at most cycles. Even for workload *FT* that stresses the network the most, there is only a single flit crossing a router for 15% of the time while about 83% of the time a router is receiving no flit at all. Now, back to the concerns on speculation, if the internal bandwidth of a router (observed in Figure 5.1.1) allows a packet to be arbitrarily routed to all possible outputs (multicast), there is no need to have RC delay in the per-hop latency

[3]The y-axis of this figure starts from 80%.

anymore while RC only exists for the purpose of acknowledging a packet that is correctly routed. This puts RC, VA and ST operations parallelizable while SA is no longer needed. To summarize, speculation on RC is not necessary anymore with multicast where it trades in more bandwidth for lower latency. Last but not least, relying on multicast also gets rid of the design issues related to LAR since either extra wiring or assistance from other routers is not needed.

5.2 Multicast within a Router Approach and Architecture

Following Section 5.1, the proposed multicast within a router architecture or McRouter is presented in this section. It is overviewed in Subsection 5.2.1 while details of its design, timing and critical path delay are covered in later subsections.

5.2.1 Overview

As mentioned before, McRouter's potential of having 1-cycle latency comes from the multicast operation within the router. It multicasts when there is remaining bandwidth on the crossbar switch; in other words, a packet may be transmitted to all possible output ports, which always includes its target output port, only if the remaining bandwidth is able to support such an operation within a router. When multicast operation is taken for a packet, its head flit immediately traverses the entire crossbar switch (except where it comes from) while at the same time, RC is carried out and its result acknowledges the *one* correctly routed head flit. Misrouted head flits are discarded while this RC result can then be used for any body flits of this packet. If the internal bandwidth of a router is sufficient for a multicast

operation, SA for the head flit is not needed any more. The router is able to serve a multicast operation for any single flit coming if the entire crossbar switch is not going to serve any traversal. Multicast only happens when the remaining bandwidth within a router can afford it. Flits being multicast also have lower priority when acquiring router resources. These two reasons have made McRouter less demanding than it looks.

Same as LAR and PR, McRouter also works for virtually any routing algorithm, since multicasting simply covers all possible RC outcomes. In this work, X-Y routing is used since it is the most popular and straightforward routing algorithm for a NoC router.

In a multicast operation, the entire crossbar switch of McRouter is almost occupied (not always true, more details in the next subsection), thus a conflict occurs if more than one packet is traversing McRouter at the same cycle. In such a situation (rare as seen from Figure 5.1.2), a contention-free policy is followed. No multicast is allowed and all packets are going to traverse the conventional data and control paths.

Another thing worth mentioning is, McRouter handles multi-flit packets well. To retain the single-cycle operation, McRouter carries out SA for body flits before they actually show up. More details are covered in Subsection 5.2.3.

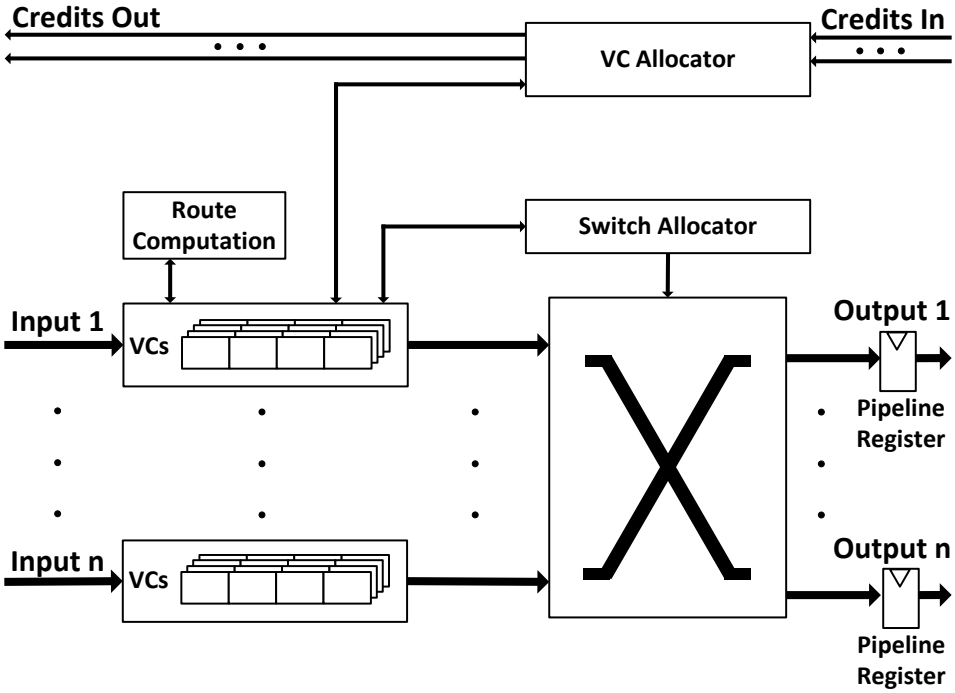


Figure 5.2.1: Architecture of the conventional router.

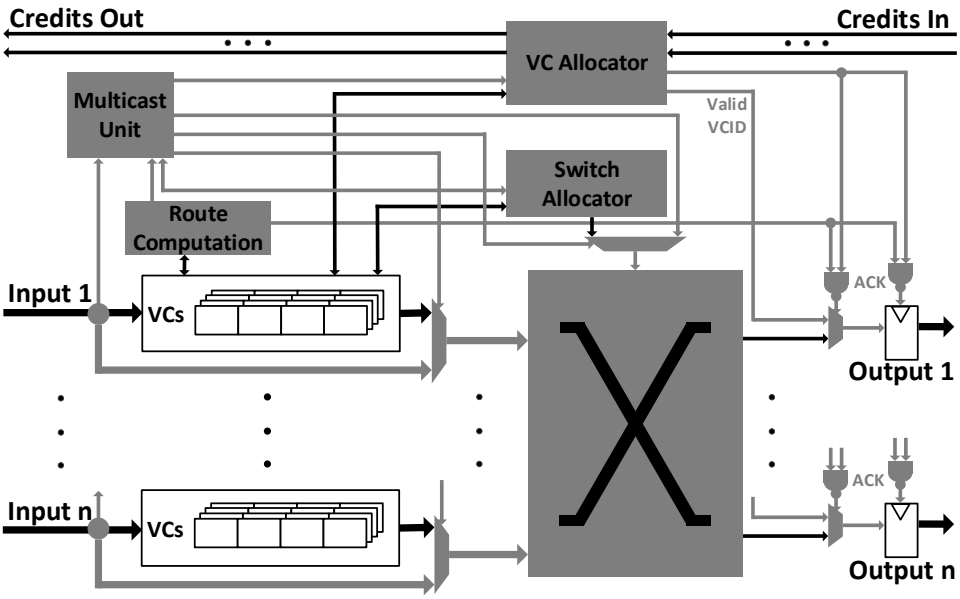


Figure 5.2.2: Architecture of McRouter.

5.2.2 Architectural Changes and the Multicast Operation

To support multicast operations inside a router, there are several architectural changes necessary. Such changes are going to be detailed in this subsection and how these components function when multicasting will also be presented. Starting from a CR, each component of McRouter will have its modifications described. In Figure 5.2.1, the architecture of CR is illustrated while the proposed McRouter has Figure 5.2.2 for the same purpose. In Figure 5.2.2, changes made from Figure 5.2.1 are shown in grey.

Input units: Direct connections from the input links to the crossbar switch are required here since when a packet is being multicast, all of its flits may traverse the crossbar switch immediately.

Multicast unit: This unit determines if a multicast operation is going to be taken, and if so, how other parts of the router need to function. Two sets of signals from the input units are required to be supplied to this unit, the first is flit type and the second is the virtual channel ID (VCID). Another two sets of input signals come from the switch allocator and the route computation unit. The input signals from the switch allocator tell which input and output ports of the crossbar switch are granted at the last cycle, so they are occupied at this cycle. The input signals from the route computation unit supply the RC result of a packet being multicast. Flit type helps determining if a multicast operation is going to be taken. For example, if only one head flit enters the router, it is able to be multicast if there is no granted SA requests at the last cycle. It also helps invoking the speculative VAs needed when multicasting a head flit. Flit type, VCID, SA granting and RC result

are used to help traversing a body flit in a single cycle if a multi-flit packet is multicast. In more details, before a body flit of a multi-flit packet enters the McRouter, if the input and output required by this body flit are not granted to other SA requests, then this flit will be able to traverse the crossbar switch immediately. Last but not least, in the case of multicasting, this unit controls the crossbar switch directly.

Route computation unit: For McRouter to work correctly, the route of a packet still needs to be computed in order to acknowledge the correctly routed head flit while the packet is being multicast. Therefore, route compute unit is required to have a set of additional input and output signals to accomplish these requirements. Firstly, a group of input signals come from the input links directly, carrying the information encoded in the head flit for immediate RC. Secondly, a 1-bit output from this unit is required to form an "ACK" signal which is used for the acknowledgment of multicast. The time required to generate this signal depends on which routing algorithm is used and it is much shorter than ST delay in practice [19, 45]. Finally, the RC result is also supplied to the multicast unit to help traversing body flits in the case of multicasting a multi-flit packet.

VC allocator: Apart from its normal operation, the VC allocator is required to allocate a VC to the correctly routed head flit in case of a multicast. Although there is only one correctly routed head flit, this VA has to be done for all potential outputs for this head flit since RC result is yet to know at this point. Additionally, since this head flit is going to leave the router in 1-cycle, modifying the "VCID" field at the input buffers is not sufficient in time and this has to be done at the pipeline registers of the ST stage of the router. If a valid "VCID" is not successfully acquired

for the correctly routed head flit in the case that all VCs are fully occupied, a multicast operation fails and the packet enters the conventional data path. Thus, a 1-bit output to acknowledge the process of obtaining a valid "VCID" is also required to form the "ACK" signal mentioned in the last paragraph. To invoke VA immediately while a head flit is being multicast, a 1-bit signal from each input is needed where the flit type field is picked up since it is "1" when a head flit enters a router and it will help determining if a multicast is about to be taken and which output has to be allocated the VCs. Note that there is in fact no input arbitration needed when obtaining VCs for a multicast head flit, since it (the input VC) requires multiple output VCs (one for each potential output) allocated before RC is out. Moreover, output arbitration for such a head flit in VA should be considered lower in priority than VA carried out for non-multicast head flits in conventional data and control paths. In practice, McRouter is implemented with a separable output-first VC allocator. After the output arbitration, remaining VCs that are least recently allocated are used to supply the multiple VCs that a multicast head flit needs. Note that only the VC allocated to the correctly routed head flit is actually meaningful while all the other VC allocations related to this multicast operation are simply reverted.

Switch allocator: If switch allocator possesses no request to occupy the crossbar switch, the next cycle it will be able for the crossbar switch to multicast a head flit. This information is passed on to the multicast unit along with which input and output ports of this crossbar are occupied if any request is granted.

Table 5.2.1: Destination output ports when multicasting considering incoming ports.[4]

		Destination			
		North	South	West	East
Source	North	×	◯	×	×
	South	◯	×	×	×
	West	◯	◯	×	◯
	East	◯	◯	◯	×

Table 5.2.2: Destination output ports when multicasting considering packet types.[4]

		Destination		
		Core	L2$/Dir	MC
Type	Request	◯	◯	×
	Forward	×	×	◯
	Response	◯	◯	◯

Crossbar switch: Multicast support is implemented in the crossbar switch, which means, with an N by N crossbar switch, N^2 control signals are required to help invoking a proper multicast. A multicast operation does not always occupy the entire crossbar switch. Routing algorithm and packet types may help minimizing the usage of the crossbar while multicasting. Firstly, taking X-Y routing (used in all evaluations) as an example, a flit turns its direction when its movement on the X-dimension has finished. This means that if a flit comes from the northern or southern port, it will never be routed to the western or eastern port. So this helps an already-turned flit to reduce its cost of multicast. This is summarized in TABLE 5.2.1. Secondly, coherence protocols such as MOESI have three types of packets to maintain cache coherence which are *request*, *forward* and *response*.

[4] ◯ denotes "Yes" while × denotes "No".

Request messages target at cores or L2 cache banks, which means that McRouter needs not consider a memory controller as routing destinations for request messages. Similarly, *forward* messages target at memory controllers only, so cores and L2 cache banks are not considered as routing destinations for forward messages. *Response* messages can target at all types of destinations, so it does not help. This is summarized in TABLE 5.2.2.

Head flit acknowledgment circuit: Multicast operation results in multiple copies of a head flit traversing the crossbar switch to reach all potential outputs. It is necessary that only the correctly routed one gets acknowledged. This requires two NAND gates and a multiplexer. In the case of multicasting, one NAND gate is used to acknowledge the correctly routed head flit while the multiplexer and another NAND gate is used to write the VCID in time.

5.2.3 Timing

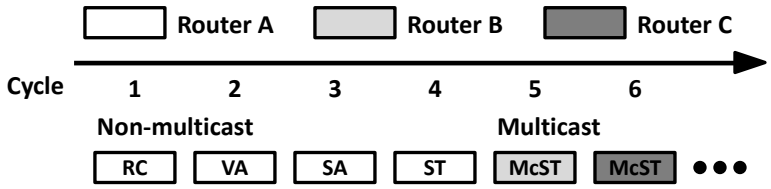


Figure 5.2.3: Pipeline stages of McRouter.

This subsection presents the best case timing of McRouter. Figure 5.2.3 illustrates how a head flit of a packet is transmitted through the pipeline stages of McRouter. When a head flit is being multicast (shown as McST at Routers B and C), it takes

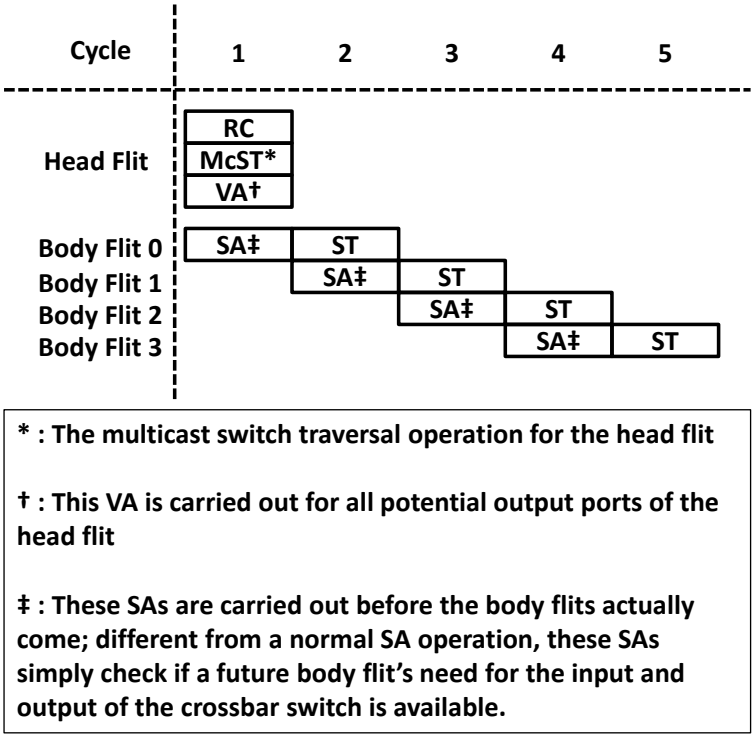


Figure 5.2.4: Best case transmission of a multi-flit packet in McRouter.

one cycle to reach the links; if it is not being multicast (at Router A), the conventional data path of the router is invoked and it spends four cycles to finish a routing. Figure 5.2.4 shows how an entire packet is transmitted in the case of multicasting. For any packet whose head flit is being multicast, the first cycle at McRouter is the key. Not only the head flit is being multicast, but a route computation is also carried out to acknowledge the head flit correctly transmitted. In addition, multiple VCs are required to be allocated for this head flit before RC is done. Meantime, SA (mainly to check which inputs and outputs of the crossbar are vacant) is carried out to see if the switch allocator grants anything; a body flit of the packet is able to traverse the crossbar if the input and output it needs are not going to be occupied.

5.2.4 Critical Path Delay

The longest pipeline stage determines the critical path delay of a router. According to literature [19, 45], VA is generally the longest pipeline stage in a router. As described in Subsection 5.2.2, the operation of the multicast unit is mostly in parallel with VA so its delay can be hidden. While the first rank of arbiters of the VC allocator is arbitrating the output VCs for non-multicast packets, the multicast unit is deciding if a multicast operation is going to be taken and if yes, it then sets up the crossbar switch. This decision on multicasting will then be used at the VC allocator to allocate more output VCs needed to accomplish this multicast operation. At the same time, the VC allocator's second rank of arbiters are arbitrating the input channels for non-multicast packets. Thus, the delay of the multicast unit is in parallel with the delay of the VA stage. However, the operation and existence of the multicast unit add additional delay to the ST stage since the multicast unit needs to control the switch crossbar directly in terms of having multicasting, the simplicity of the multicast unit, by having pure logic, guarantees that this delay is well hidden since ST delay is much shorter than VA delay in practice [19, 45].

A delay which cannot be hidden in McRouter's data and control paths is at its head flit acknowledgment circuit. In Figure 5.2.2, a NAND gate has to be placed between an output of the VC allocator and the pipeline registers of the ST stage. However, this NAND gate is going to add roughly one FO4 delay to McRouter's critical path, which is negligible.

5.3 Architecture Discussions and Qualitative Comparisons

In this section, discussions on McRouter and comparisons against VSAR [45] and PR [39] are presented. Purposes of such discussions and comparisons are to identify the pros and cons of McRouter. VSAR and PR are chosen as counterparts because both of them are standalone designs like McRouter which maintains portability and modularity and do not require any assistance from upstream routers or any change in top-level wiring; this means, VSAR, PR and McRouter can simply replace any CR in NoCs. Before going into details, one thing in common for all routers discussed in this section is that they are all based on CR, so when speculation, prediction or multicast fails, they all follow the control and data paths of CR.

As reviewed in Section 2.2, CR is a 4-cycle router when free of contention, VSAR is a 3-cycle router if speculation succeeds and PR is a 1-cycle router if prediction succeeds; while as introduced in Section 5.2, McRouter is a 1-cycle router if multicast is successfully carried out. Despite how latencies are hidden, differences can be identified in four aspects: control dependency, speculation, routing efficiency and power overhead. These comparisons are summarized in TABLE 5.3.1.

5.3.1 Control Dependency

Every neighboring pipeline stage in a CR has control dependencies among each other (RC and VA, VA and SA, and SA and ST). VSAR hides the control dependency between VA and SA by speculating that SA can be successfully performed in

Table 5.3.1: Qualitative comparisons of low latency routers including McRouter.

	Low latency routers		
	VSAR	PR	McRouter
Control dependency	Control dependency between VA and SA can be hidden with speculation	No control dependency hidden	Control dependencies between RC and VA/ST are broken while dependency between VA and ST is hidden
Speculation	VA/SA can succeed at the same cycle	RC is predicted	A valid VC can be obtained for the head flit while a valid time slot of the crossbar switch is available for the first body flit
Routing efficiency	3-cycle routing when contention is low at VA/SA	1-cycle routing when prediction hits	1-cycle routing when router bandwidth allows multicasting
Power overhead	Static: an extra switch allocator; dynamic: additional accesses to switch allocator when speculation fails	Static: predictors and kill circuit; dynamic: predictor activities, additional access to the virtual channel allocator, the switch allocator and the crossbar switch when prediction fails, and activity at the kill circuit in case of mis-prediction	Static: multicast unit and the acknowledgment circuit; dynamic: multicast unit activity, additional accesses to the virtual channel allocator, the switch allocator and the crossbar switch when multicast is taken, and activity at the flit acknowledgment circuit in the case of a multicast operation

parallel with VA if VA returns a valid VCID, otherwise carrying them out in parallel is meaningless. PR does not hide any control dependency by prediction. With predicted routing available every cycle, VA and SA are carried out in parallel to shorten the preparation time for a potential predictive switch traversal.

For McRouter, dependencies between RC and VA/ST are broken when a head flit is being multicast as going to every possible output means routing is always correct. Dependency between VA and ST is also hidden since VA is carried out for every participating output in a multicast operation. For a multicast head flit, SA is no longer needed and dependency between SA and multicast ST (or McST) does not exist, since multicast operations only take place when the crossbar has enough bandwidth and there is no need to allocate the crossbar switch in such a situation. However, dependency between SA and ST still exists for a body flit whose head flit is being multicast. After route computation acknowledges a head flit, any body flit which follows this head flit requires two stages, SA and ST, to traverse the crossbar. Hence, SA should be carried out before the actual body flit comes, which helps McRouter maintain its one flit per cycle performance.

5.3.2 Speculation

CR has no speculation while VSAR speculates that VA/SA both return valid results at the same cycle. For PR, three speculations are made if a head flit is going to be routed predictively. First, RC is speculated with predictions. Secondly, VA and SA have to be performed in parallel based on this predicted RC result. Thirdly, while the head flit predictively traverses the crossbar switch, the first body

flit which follows it has to enter SA speculatively with the predicted RC result.

McRouter has two sources of speculation. Firstly, for VA, every output involved in a multicast operation has to have a valid VC obtained and a speculation is needed on that this VA is done for the correctly routed output to guarantee the success of this multicast. Secondly, while the head flit is multicast, the first body flit which follows it has to enter SA speculatively (at every output involved in this multicast operation). The success of this SA helps guaranteeing that the first body flit traverses McRouter in one cycle. However, there is no speculation on RC because of multicasting (always correct). Flits being multicast always reach the target output port without RC, and RC is only used to acknowledge the correctly routed head flit.

5.3.3 Routing Efficiency

Amount of contention determines the efficiency of VSAR, which means, the more flits VSAR has in its data path competing in VA/SA, the less chance it can successfully speculate. In terms of PR, other than the amount of contention because of having more flits, another problem is that contention also comes from predictions. VA/SA following a predicted RC may fail because of two reasons. First, predictions for two or more input ports may overlap. Secondly, VA and SA following a prediction may overlap with flits that are not predictively routed, for example, the body flits. In a word, having prediction is similar to having more flits in the router. Furthermore, results of predictions also affect the efficiency because mis-predictions result in a head flit being re-routed with the conventional pipeline.

McRouter's efficiency is determined by how many flits can be multicast and this is related to how plentiful the bandwidth is within the router, which means, McRouter is more efficient when the network load is low (like Figure 5.1.1 and Figure 5.1.2). When the network load is too high, multicast can not be taken and its routing efficiency is as low as a CR.

5.3.4 Power Overhead

The power overhead of VSAR, PR and McRouter comes as both static and dynamic power. For VSAR, its static power overhead is from the extra switch allocator which handles speculative SA requests while its dynamic power overhead is from the additional accesses to the switch allocators when speculation fails. For PR, more static power is consumed by the predictors and the kill circuit. More dynamic power is consumed by PR with three purposes. Firstly, predictors are activated when a packet arrives at a PR (predicted RC result may be refreshed). Secondly, additional accesses to the virtual channel allocator, the switch allocator and the crossbar switch exist when prediction fails. Thirdly, the kill circuit is also activated in case of mis-prediction.

For McRouter, the static power overhead comes from the multicast unit and the acknowledgment circuit. Dynamic power overhead of McRouter is caused when multicast succeeds. Firstly, multicat unit is activated when any packet arrives at a McRouter. Secondly, extra accesses to the virtual channel allocator, the switch allocator and the crossbar switch exist when multicast is taken. Thirdly, the flit acknowledgment circuit is activated to allow the correctly routed flit to pass in the

Table 5.4.1: System parameters.

Component	Parameter
Number of cores:	16
Topology:	4×4 mesh
Processor:	4 GHz, in-order
L1 I/D cache:	32 KB per core, 4-way set associative, 1 cycle access latency
L2 cache:	256 KB per Bank, 16-way set associative, 6 cycles access latency
Cache line size:	64 Bytes
Main memory:	4 GB, 160 cycles access latency
Coherence protocol:	MOESI, directory
Link:	128-bit, 1 cycle traversal
Packet:	128-bit control, 640-bit data
Router:	1 GHz, virtual channel router
Virtual channel:	4 per virtual network
Virtual network:	3 per physical link
Routing algorithm:	X-Y routing
Process technology:	32 nm
Vdd:	1 V

case of a multicast operation.

Apart from the power overhead, low latency routers like VSAR, PR and McRouter can help reducing the total energy consumed by the system since these router designs are able to speed-up the system thus shortening the execution time.

5.4 Methodology

Evaluations are carried out on performance and power by using GEMS [37] and Simics [36] extended with the network model from GARNET [7] and the power model from Orion [29]. To evaluate performance, the source code of GARNET are modified to provide cycle-accurate timing models of VSAR, PR and McRouter. We evaluate PR with two prediction algorithms, which are latest port (LP) and finite context method (FCM, the same as most-frequently-used). In the power

Table 5.4.2: Benchmark programs and inputs.

Application	Input
barnes	4096 particles
cholesky	tk29.O
fmm	16384 particles
ocean, contiguous	grid of 258×258
ocean, non-contiguous	grid of 258×258
raytrace	teapot
volrend	head
water, nsquared	512 molecules
water, spatial	512 molecules
EP	2^{25} random number pairs
FT	grid size of $128 \times 128 \times 32$, 6 iterations
IS	2^{23} keys with a max key of 2^{19}
LU	grid size of $64 \times 64 \times 64$, 250 iterations, time step of 2.0

evaluation, power consumption of low latency routers are quantified by looking at the component power models in Orion. For VSAR, the power consumed by the extra switch allocator is added. For PR, both the power consumption of memory components inside the predictors and the power consumption from extra router component accesses incurred by mis-predicted flits are considered. For each LP predictor, the power model of a 3-bit register is used while for each FCM predictor, an 8-bit register file is used. The number of registers in the register file equals N-1 if the predictor is implemented in an N-radix router. Similarly, the power model of McRouter includes power overheads from the extra accesses to the router components resulted from multicasting. The evaluation conditions are summarized in TABLE 5.4.1.

In all evaluations (except some sensitivity studies), a 16-tile mesh network with 128-bit links is assumed. Each tile has an in-order processor core, a bank of L2 Cache/Directory. Each corner tile also has a memory controller. Similar to other

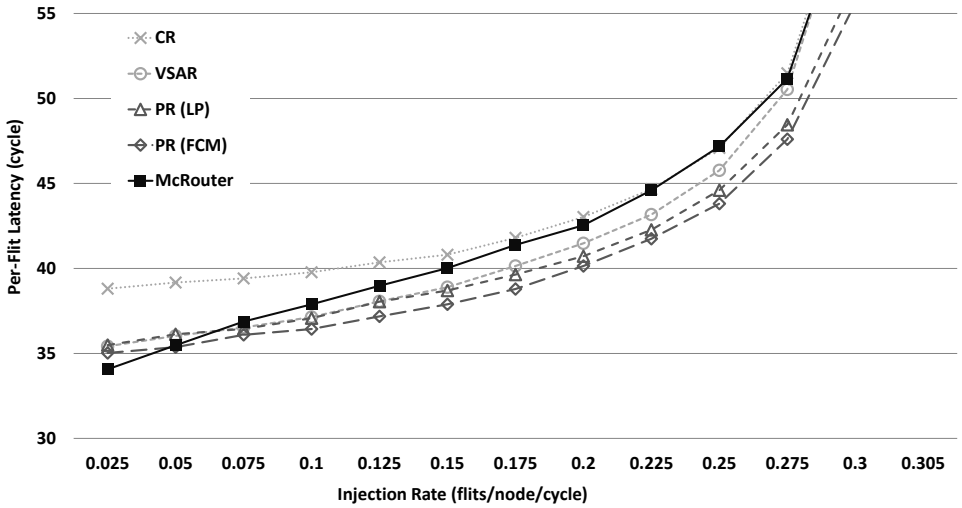
low latency routers, McRouter can accelerate remote L1, L2 and main memory accesses; and the former two are best candidates since main memory access latency is much larger than network latency.The assumption of using in-order cores in evaluation is reasonable since these cores run at 4 GHz and are 4 times the frequency of the network (so the network is properly stressed). The schematic view of this simulated system and what a tile is composed of are illustrated in Figure 1.1.1 in Section 1.1. The entire network is set to have three virtual networks to support the MOESI directory coherence protocol which has three classes of packets. Each router has a maximum of six ports and each port has four virtual channels while each virtual channel has four 128-bit buffers. More details are presented in TABLE 5.4.1. Evaluations are carried out with both synthetic and application traffic. The synthetic traffic pattern is *uniform random* and the traffic is made of data packets only (each data packet has 5 flits). The application traffic is based on workloads chosen from NPB 3.3 [3] and SPLASH-2 [54] benchmark suites. TABLE 5.4.2 lists these applications and their inputs. Last but not least, in Subsection 5.5.3, some parameters are downscaled to see how McRouter behaves under different situations.

5.5 Results

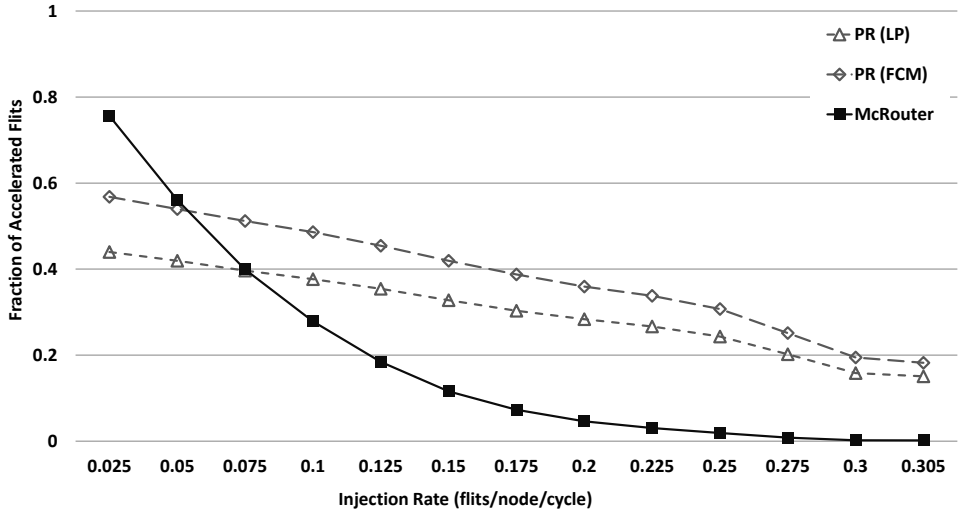
In this section, the evaluation results on McRouter are presented and discussed in terms of its performance and power consumption. As stated in Section 5.4, McRouter is compared to CR, VSAR and PR in evaluations to test its effectiveness and how bandwidth consumption makes it differ. Sensitivity studies are also set

up to clarify McRouter's effectiveness in a few bandwidth-constrained situations.

5.5.1 Synthetic Traffic



(a) Per-flit latency versus injection rate.



(b) Fraction of accelerated flits versus injection rate.

Figure 5.5.1: Evaluations with synthetic traffic.

In Figure 5.5.1a, average per-flit latency versus injection rate per node is presented while the fraction of accelerated traffic is reported in Figure 5.5.1b. All routers mentioned above are included in Figure 5.5.1a while only PR with LP/FCM algorithms and McRouter are shown in Figure 5.5.1b, since the latter only covers the amount of traffic accelerated to pass a router in 1 cycle.

In Figure 5.5.1a, when different amount of traffic is injected into the network, the routers perform very differently. It is observed that when injection rate is low (up to 0.05 flits/node/cycle; this is roughly equal to 760 MBytes/node/sec and it results in a link utilization of 0.07 flits/link/cycle), McRouter outperforms all counterparts. This also matches Figure 5.5.1b that more flits can be accelerated with McRouter than PR with FCM when injection rate is under 0.05 flits/node/cycle. By looking at Figure 5.1.1, there is no such application which is going to inject traffic to form a link utilization of 0.07 flits/link/cycle and this is going be proved again by evaluations with application traffic in the next subsection.

From Figure 5.5.1a, it can also be seen that McRouter maintains its advantage over CR until injection reaches 0.225 flits/node/cycle (it is roughly equal to 3430 MBytes/node/sec and this results in a link utilization of 0.34 flits/link/cycle). This means, the performance of McRouter is better than CR unless the injection is very high (already near saturation).

5.5.2 Application Traffic

Figure 5.5.2 shows the network performance (note that lower is better) under different router designs. Except for *ocean (non-contiguous)* where PR with FCM

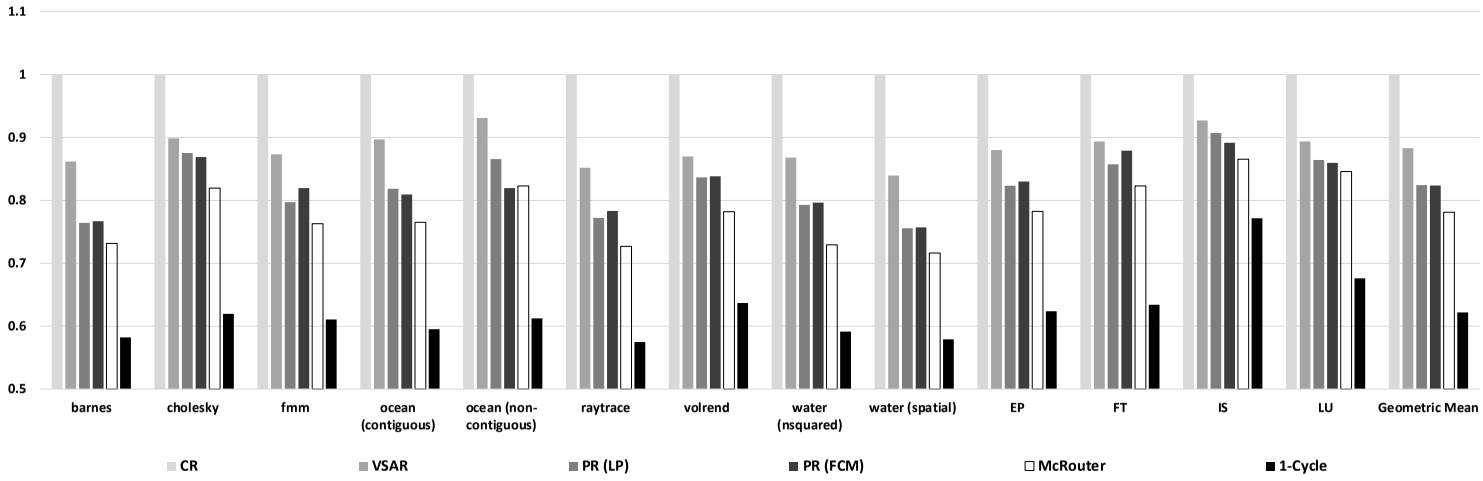


Figure 5.5.2: Normalized per-flit latency.

algorithm is 0.4% better than McRouter, McRouter clearly outperforms all other router designs. On average, McRouter shortened the per-flit latency by 22% over CR while it also provides an additional 4% reduction over the best PR. At best, McRouter achieves 28% latency reduction over CR for *water (spatial)* and 6% latency reduction over PR for *water (nsquared)*.

Figure 5.5.3 presents the system performance (note that higher is better). A similar outcome is observed that McRouter outperforms every other router design except workload *cholesky* where PR with LP algorithm is 1% better. Looking at the geometric mean, McRouter helps the system achieve a speed-up of 1.28 over CR while on average it also provides a speed-up of 1.05 over the best PR. In the best case, a speed-up of 1.48 can be spotted for McRouter over CR for workloads *barnes* and *water (spatial)* while a speed-up of 1.08 can be identified for *raytrace* over the best PR.

These performance numbers clearly demonstrate that McRouter is the optimal router design given the application workloads evaluated. Although both McRouter and PR are 1-cycle routers, the number of times McRouter successfully multicasts a packet is higher than the number of successful predictions PR has, under the evaluated workloads.

Another observation is, as what it is designed for, on-chip bandwidth is a determining factor on how well McRouter can perform. For workloads that consume more on-chip bandwidth (such as *ocean(non-contiguous)*, *EP*, *FT*, *IS* and *LU*), improvement with McRouter on both network and system performance are relatively smaller. Another reason behind this is, similar to PmR, working set size also plays

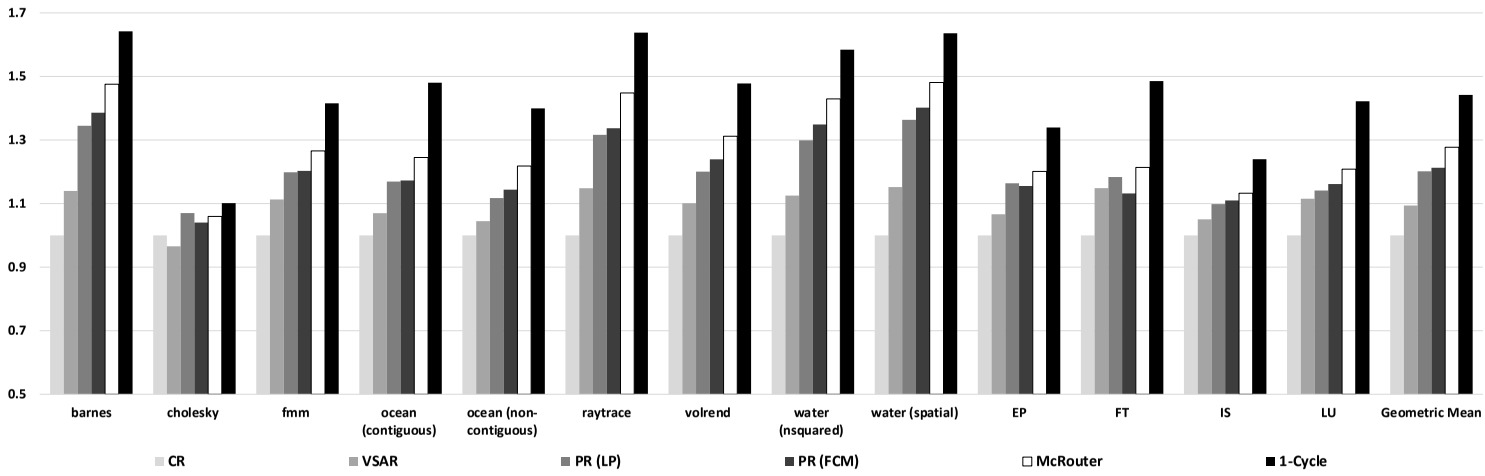


Figure 5.5.3: Normalized system speed-up.

an important role here. For the NPB workloads that consume more memory, improvement with McRouter on system performance is smaller since more L2 cache misses results in more main memory accesses whose latencies are far larger than network latencies. Workload *cholesky* is an exception since all low latency router designs (including McRouter and ideal 1-cycle router) are not efficient for it. It seems that it is not much affected by the network performance.

In addition, by looking at Figure 1.3.1 in Section 1.3, for those workloads that scale well with more threads, improvements on their performance mostly come from speed-up on the serial execution time of each individual thread. *raytrace* scales poorly with more threads but it has larger performance improvement since slower cache accesses are well accelerated with low latency routers studied in this dissertation.

McRouter is also well behind the ideal 1-cycle router. Although average link utilization is very small for these workloads, it seems that network traffic tends to burst so that in reality the amount of multicasting taken is smaller than expected because of higher crossbar switch utilization temporally.

Demonstrated in Figure 5.5.4, McRouter consumes on average 1% more power than CR and this is similar to VSAR and PR with LPM. This simply means that McRouter is the most power efficient design when compared to its counterparts. With merely 1% more power consumption, McRouter outperforms CR by 28% in speeding-up the system. Similarly, with roughly the same amount of power consumption, McRouter outperforms VSAR by 17% in speeding-up the system. When compared to PR, McRouter outperforms both PRs (regardless of the pre-

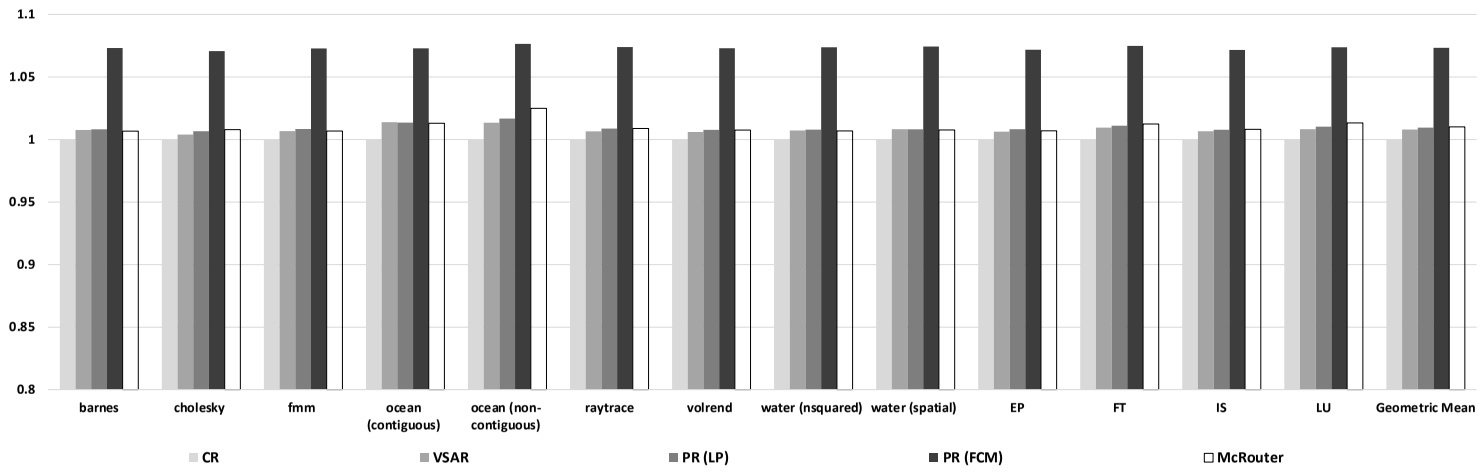
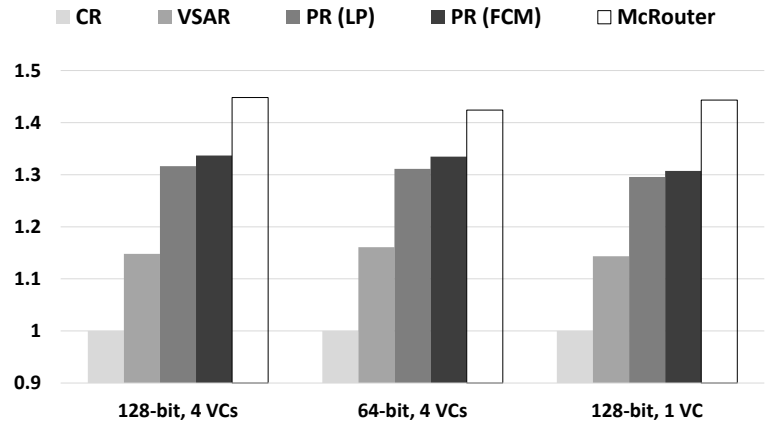


Figure 5.5.4: Normalized network power consumption.

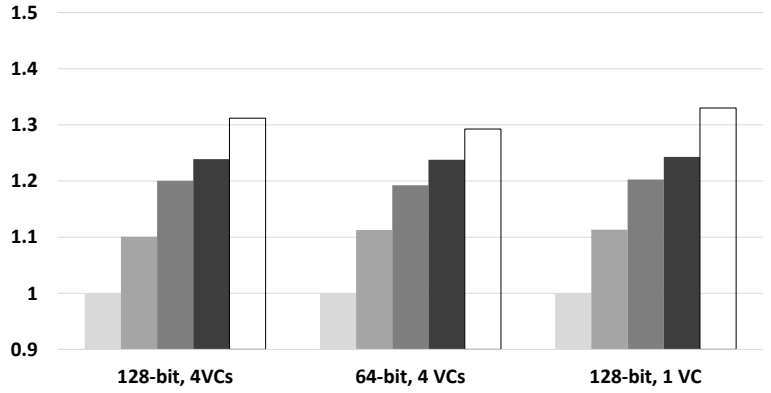
diction algorithm) by 5% while it consumes roughly the same amount of power as PR with LP and even better, it consumes 6% less power than PR with FCM. Although multicasting seems a power hungry operation, this is not the case for McRouter. Since McRouter does not incur any additional link traversal and buffer accesses; while at the same time, not every output is needed to carry out a multicast operation. Another reason that McRouter is more power efficient is because of the fact that it is bandwidth driven. Only if enough router bandwidth is present, multicast happens and it accelerates the traffic; so a multicast operation is very productive that unlike mis-predictions from PR, there is rarely any case that it consumes power but accelerates none.

5.5.3 Sensitivity Studies

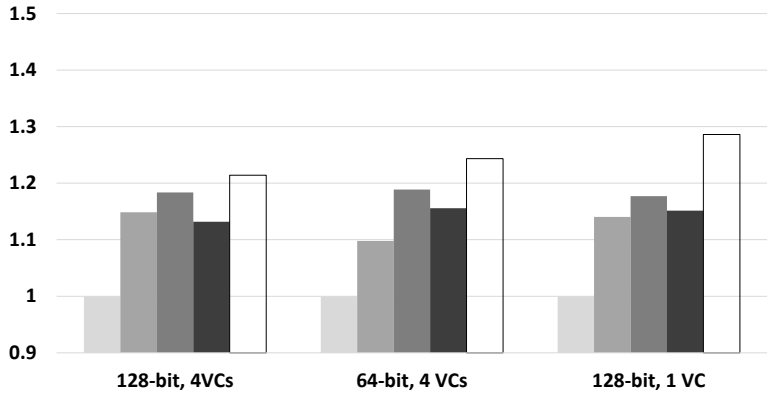
In this subsection, McRouter's efficiency is unveiled with smaller on-chip bandwidth available. The purpose to have such studies is to clear any doubt that McRouter's advantage shown in Subsection 5.5.2 comes from too plentiful on-chip bandwidth (in other words, over-designing). To achieve this goal, 2 parameters are chosen to be downscaled in the evaluation. The 2 parameters chosen to be varied are flit size and number of VCs. By changing these parameters, the available bandwidth inside a router is decreased. All of these evaluations are set up to test the speed-up on system performance with different routers when these two parameters are downscaled. We select 3 workloads (*raytrace*, *volrend* and *FT*) to carry out these evaluations. *FT* is one of the most on-chip bandwidth demanding workloads while *raytrace* and *volrend* have relatively lower on-chip bandwidth demands where two dis-



(a) Workload: *raytrace*



(b) Workload: *volrend*



(c) Workload: *FT*

Figure 5.5.5: System speed-up with router parameter downscaling.

tinct types of applications are covered in evaluations.

As shown in Figure 5.5.5, the flit size is halved; 64-bit flit is the minimum size found in literature. Another downscaling has the number of VCs reduced from 4 to 1; 1 VC is the minimum number of VCs possible (and all routers actually turn into wormhole routers in this case).

What is seen from Figure 5.5.5a and Figure 5.5.5b tells that McRouter's efficiency merely changes with either case of downscaling. This proves that even with smaller on-chip bandwidth available, McRouter is still the best design to consider.

With Figure 5.5.5c, things are more interesting. McRouter performs even better with less available on-chip bandwidth for *FT*. The speed-up when having downscaled parameters are 1.24 (when flit size is halved) and 1.29 (when the number of VCs is decreased from 4 to 1) while with the original parameters, the recorded speed-up is only 1.21. Since *FT* is the most on-chip bandwidth demanding workload in evaluation, this not only tells that McRouter still works well with smaller on-chip bandwidth, but this also means that even for the most on-chip bandwidth demanding workload, the router bandwidth is still plentiful for McRouter to utilize.

5.6 Summary and Discussions

In this chapter a novel low latency on-chip router named McRouter is proposed and is designed for high performance NoCs. By allowing a head flit of a packet to traverse the crossbar switch in the manner of multicasting, the RC delay is completely removed from the per-hop latency, and VC allocation and switch traversal

are parallelizable which enables a single-cycle transfer of flits. Compared to low latency routers employing LAR, McRouter excels by preserving its portability and modularity in design. Furthermore, compared to low latency routers with aggressive speculation like PR, McRouter is better at low network load as its multicast nature makes it an "always-hit" prediction router whenever multicast is able to be taken. With the detailed evaluations carried out for McRouter, it is found that McRouter outperforms its counterpart designs with a negligible power overhead. On average, system speed-ups of 1.28, 1.17 and 1.05 are observed over CR, VSAR and PR with the best prediction algorithms, respectively. These are achieved with a merely 1% more power over CR. From the sensitivity study, McRouter is found to work well under tighter bandwidth budget, too.

Another observation is, with a direct comparison between PmR and McRouter with 10 workloads[5], McRouter is definitely better in *volrend*, *water (nsquared)* and *water (spatial)* which have relatively low link utilization (hence low bandwidth utilization). This simply means that McRouter is more bandwidth sensitive than PmR. It should also noted that McRouter is more power efficient than PmR for the reason that it does not have any predictors.

Following the above summary, there are also a few qualitative discussions to be made, regarding network topology, core scaling, coherence protocols and parallel speed-up of application workloads.

Firstly, topology makes difference in two aspects, the number of links and the radix of routers. For example, McRouter should be more efficient with torus than

[5]There are 3 workloads failed for PmR in simulation.

mesh, since there are more links (and the same radix for routers) with torus which can result in a smaller link utilization. Conversely, with ring topology, McRouter should be less efficient since the amount of links are much lower while the radix of routers is also slightly lower. This means, the available bandwidth (in terms of network resources) is less and the link utilization can be much higher.

Secondly, there are two forms of core scaling which have opposite influences on McRouter. If the number of tiles scales with the number of cores, McRouter should still be effective or even better since with such a scaling traffic travels longer distance in the network. However, if the size of the network stays while the number of cores scales up (a denser design), it is hard for McRouter to maintain its effectiveness since the network is simply more stressed in this case.

Finally, McRouter should work better with directory based cache coherence protocols because of the fact that such protocols are less bandwidth stressful than broadcast based ones.

The end of a melody is not its goal: but nonetheless, had the melody not reached its end it would not have reached its goal either.

Friedrich Wilhelm Nietzsche

6 Conclusions

It is clear that this shift to multi-core designs is somehow inevitable. Replication of processor cores on chip in such a modular manner does not only favor on-chip networks but also puts a few challenges on them. A key challenge this dissertation has contributed to is to optimize the communication latency of such networks.

First and foremost, there are three findings which motivates this dissertation. The first one is the bandwidth limitation identified in 3D NoCs. This physical limitation can be spotted when traffic crosses layers. It has been the starting point of the first solution described in this dissertation, where traffic compression is ap-

plied adaptively on 3D NoCs. The second finding is about the utilization of multiple prediction algorithms for speculating the routing result of a packet in an on-chip router. Multiple prediction algorithms lead to better prediction accuracy so this is where predict-more router is motivated. The third finding simply tells a fact that with multi-threaded workloads, there is plentiful of internal bandwidth for a router to possibly multicast an incoming packet to all possible outputs without knowing this packet's route computation result. And this simply leads to the idea of multicast-within-a-router design. With these findings, three solutions are simply designed to minimize the communication latency for NoCs and their effectiveness has been proved through cycle-by-cycle simulations.

6.1 Further Discussions and Future Work

This section covers further discussion beyond the scope of this dissertation and identifies possible future work to improve the three solutions presented.

The application traffic used in this dissertation are generated from multi-threaded workloads which have been optimized to minimize the amount of communications between threads. But if the target applications have been changed to multi-programmed workloads or data-parallel workloads, the effectiveness of McRouter will degrade but PmR should still work. Traffic compression, on the other hand, should be a more effective solution . The reason behind this is, the latter two types of workloads may generate more traffic than the multi-threaded ones, hence they are more bandwidth and network resource consuming, especially in the case that memory-intensive applications are executed.

As a future work, power evaluation is a good candidate for the traffic compression solution, since it is interesting to find the trade-off between the power saved from reducing the amount of network traffic through compression and the power overhead consumed by the compression/de-compression circuits. For the low latency router proposals, HDL synthesis is a promising extension to improve the accuracy of the power model and to provide an evaluation on area overhead of these two solutions. It may also be interesting to apply these solutions to networks with different scales whose latency and bandwidth requirement differ from NoCs.

References

[1] International technology roadmap for semiconductors. http://www.itrs.net/reports.html.

[2] Mobile memory: LPDDR2 & 3, Wide I/O, Memory MCP. http://www.jedec.org/category/technology-focus-area/mobile-memory-lpddr2-3-wide-io-memory-mcp.

[3] NAS parallel benchmarks 3.3. http://www.nas.nasa.gov/Resources/Software/npb.html.

[4] OpenMP specifications. http://openmp.org/wp/openmp-specifications.

[5] A Memo on Exploration of SPLASH-2 Input Sets. http://parsec.cs.princeton.edu/doc/memo-splash2x-input.pdf.

[6] Oracle Solaris releases. http://www.oracle.com/technetwork/server-storage/solaris/overview/releases-jsp-140987.html.

[7] N. Agarwal, T. Krishna, Li-Shiuan Peh, and N.K. Jha. GARNET: a detailed on-chip network model inside a full-system simulator. In *ISPASS '09: Proceedings of the 2009 IEEE international symposium on Performance analysis of systems and software*, pages 33–42, 2009.

[8] V Agarwal, M.S Hrishikesh, S.W Keckler, and D Burger. Clock rate versus IPC: the end of the road for conventional microarchitectures. In *ISCA '00: Proceedings of the 27th annual international symposium on Computer architecture*, pages 248–259, 2000.

[9] Alaa R. Alameldeen and David A. Wood. Frequent Pattern Compression: A Significance-Based Compression Scheme for L2 Caches. Technical Report TR-1500, University of Wisconsin-Madison, April 2004.

[10] Alaa R. Alameldeen and David A. Wood. Adaptive cache compression for high-performance processors. In *Proceedings of the 31st annual international symposium on Computer architecture*, ISCA '04, pages 212–, 2004.

[11] Alaa R. Alameldeen and David A. Wood. Adaptive cache compression for high-performance processors. *SIGARCH Comput. Archit. News*, 32(2): 212–, March 2004.

[12] Luiz André Barroso, Kourosh Gharachorloo, Robert McNamara, Andreas Nowatzyk, Shaz Qadeer, Barton Sano, Scott Smith, Robert Stets, and Ben Verghese. Piranha: A scalable architecture based on single-chip multiprocessing. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, ISCA '00, pages 282–293, 2000.

[13] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, Liewei Bao, J. Brown, M. Mattina, Chyi-Chang Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook. Tile64 - processor: A 64-core soc with mesh interconnect. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 88–598, Feb 2008.

[14] Luca Benini and Giovanni De Micheli. Networks on chip: a new paradigm for systems on chip design. In *In Proceedings of Conference on Design, Automation and Test in Europe*, pages 418–419, 2002.

[15] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The PARSEC benchmark suite: characterization and architectural implications. Technical Report TR-811-08, Princeton University, January 2008.

[16] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, Lei Jiang, G.H. Loh, D. McCauley, P. Morrow, D.W. Nelson, D. Pantuso, P. Reed, J. Rupley, Sadasivan Shankar, J. Shen, and C. Webb. Die stacking (3d) microarchitecture. In *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, pages 469–479, 2006.

[17] J. Burns, L. McIlrath, C. Keast, C. Lewis, A. Loomis, K. Warner, and P. Wyatt. Three-dimensional integrated circuits for low-power, high-bandwidth systems on a chip. In *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*, pages 268–269, 2001.

[18] Martin Burtscher and Benjamin G. Zorn. Hybrid load value predictors. *IEEE Transactions on Computers*, 51:759–774, 2000.

[19] William Dally and Brian Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann Publishers Inc., 2003.

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

[21] R. Das, A.K. Mishra, C. Nicopoulos, Dongkook Park, V. Narayanan, R. Iyer, M.S. Yousif, and C.R. Das. Performance and power optimization through data compression in network-on-chip architectures. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 215–225, 2008.

[22] W.R. Davis, J. Wilson, S. Mick, J. Xu, Hao Hua, C. Mineo, A.M. Sule, M. Steer, and P.D. Franzon. Demystifying 3d ics: the pros and cons of going vertical. *Design Test of Computers, IEEE*, 22(6):498–510, 2005.

[23] R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, and A.R. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256–268, 1974.

[24] Mitchell Hayenga and Mikko Lipasti. The NoX router. In *MICRO 44: Proceedings of the 44th annual IEEE/ACM international symposium on Microarchitecture*, pages 36–46, December 2011.

[25] John L. Hennessy and David A. Patterson. *Computer Architecture, Fifth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011. ISBN 012383872X, 9780123838728.

[26] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, R. Van der Wijngaart, and T. Mattson. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 108–109, Feb 2010.

[27] H. Jin, M. Frumkin, and J. Yan. The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance. Technical Report NAS-99-011, NAS System Division, NASA Ames Research Center, October 1999.

[28] Yuho Jin, Ki Hwan Yum, and Eun Jung Kim. Adaptive data compression for high-performance low-power on-chip networks. In *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*, pages 354–363, 2008.

[29] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration. In *DATE '09: Proceedings of the conference on Design, automation and test in Europe*, pages 423–428, April 2009.

[30] Dae Hyun Kim, K. Athikulwongse, and Sung-Kyu Lim. A study of through-silicon-via impact on the 3d stacked ic layout. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 674–680, 2009.

[31] Jongman Kim, Chrysostomos Nicopoulos, Dongkook Park, Reetuparna Das, Yuan Xie, Vijaykrishnan Narayanan, Mazin S. Yousif, and Chita R. Das. A novel dimensionally-decomposed router for on-chip communication in 3d architectures. In *Proceedings of the 34th annual international symposium on Computer architecture*, pages 138–149, 2007.

[32] K. Kumagai, Changqi Yang, H. Izumino, N. Narita, K. Shinjo, S. Iwashita, Y. Nakaoka, T. Kawamura, H. Komabashiri, T. Minato, A. Arnbo, T. Suzuki, Zhenyu Liu, Yang Song, S. Goto, T. Ikenaga, Y. Mabuchi, and K. Yoshida. System-in-silicon architecture and its application to h.264/avc motion estimation for 1080hdtv. In *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, pages 1706–1715, 2006.

[33] Amit Kumar, Li-Shiuan Peh, Partha Kundu, and Niraj K Jha. Express virtual channels: towards the ideal interconnection fabric. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 150–161, June 2007.

[34] Amit Kumar, Li-Shiuan Peh, Partha Kundu, and Niraj K. Jha. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *ICCD '07: Proceedings of 2007 IEEE international conference on Computer design*, pages 63–70, September 2007.

[35] Zheng Li, Jie Wu, Li Shang, Robert P. Dick, and Yihe Sun. Latency criticality aware on-chip communication. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, pages 1052–1057, 2009.

[36] P.S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: a full system simulation platform. 35(2):50–58, 2002.

[37] Milo M.K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *SIGARCH Computer Architecture News*, 33(4), November 2005.

[38] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga. Prediction router: yet another low latency on-chip router architecture. In *HPCA '09: Proceedings of the 2009 IEEE 15th international symposium on High performance computer architecture*, pages 367–378, 2009.

[39] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga. Prediction Router: a low-latency on-chip router architecture with multiple predictors. 60(6): 783–799, 2011.

[40] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[41] Robert Mullins, Andrew West, and Simon Moore. Low-latency virtual-channel routers for on-chip networks. In *ISCA '04: Proceedings of the 31st annual international symposium on Computer architecture*, pages 188–197, June 2004.

[42] Kunle Olukotun, Basem A. Nayfeh, Lance Hammond, Ken Wilson, and Kunyung Chang. The case for a single-chip multiprocessor. *SIGOPS Oper. Syst. Rev.*, 30(5):2–11, September 1996.

[43] Dongkook Park, S. Eachempati, R. Das, A.K. Mishra, Yuan Xie, N. Vijaykrishnan, and C.R. Das. Mira: A multi-layered on-chip interconnect router architecture. In *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pages 251–261, 2008.

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

[45] Li-Shiuan Peh and William J. Dally. A delay model and speculative architecture for pipelined routers. In *HPCA '01: Proceedings of the 7th international symposium on High-performance computer architecture*, pages 255–255, 2001.

[46] Li-Shiuan Peh and Natalie Enright Jerger. *On-Chip Networks*. Morgan and Claypool Publishers, 1st edition, 2009.

[47] R.S. Ramanujam and Bill Lin. Randomized partially-minimal routing on three-dimensional mesh networks. *Computer Architecture Letters*, 7(2): 37–40, 2008.

[48] Brian M. Rogers, Anil Krishna, Gordon B. Bell, Ken Vu, Xiaowei Jiang, and Yan Solihin. Scaling the bandwidth wall: challenges in and avenues for cmp scaling. In *Proceedings of the 36th annual international symposium on Computer architecture*, pages 371–382, 2009.

[49] Daniel Sanchez, George Michelogiannakis, and Christos Kozyrakis. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. *ACM Transactions on Architecture and Code Optimization (TACO)*, 7(1), April 2010.

[50] A. Sheibanyrad, F. Petrot, and Janstch A. *3D Integration for NoC-Based SoC Architectures*. Springer, 2010.

[51] Daniel J. Sorin, Mark D. Hill, and David A. Wood. *A Primer on Memory Consistency and Cache Coherence*. Morgan & Claypool Publishers, 1st edition, 2011.

[52] M. Thuresson, L. Spracklen, and P. Stenstrom. Memory-link compression schemes: A value locality perspective. *Computers, IEEE Transactions on*, 57 (7):916–927, 2008.

[53] Frederick C. Wong, Richard P. Martin, Remzi H. Arpaci-Dusseau, and David E. Culler. Architectural requirements and scalability of the nas parallel benchmarks. In *Proceedings of the 1999 ACM/IEEE Conference on Supercomputing*, 1999.

[54] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. The SPLASH-2 programs: characterization and methodological considerations. In *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*, pages 24–36, 1995.

[55] Ping Zhou, Bo Zhao, Yu Du, Yi Xu, Youtao Zhang, Jun Yang, and Li Zhao. Frequent value compression in packet-based noc architectures. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, pages 13–18, 2009.

List of Publications by the Author

Refereed Journal Publications

[J-1] **Yuan He**, Hiroki Matsutani, Hiroshi Sasaki, and Hiroshi Nakamura, "Adaptive Data Compression on 3D Network-on-Chips," IPSJ Transactions on Advanced Computing Systems, Vol.5, No.1, pp.80-87, January 2012.

Refereed Conference and Workshop Publications

[C-1] **Yuan He**, Hiroshi Sasaki, Shinobu Miwa, and Hiroshi Nakamura, "McRouter: Multicast within a Router for High Performance Network-on-Chips," In Proc. of the 22nd International Conference on Parallel Architectures and Compilation Techniques, pp.319-329, September 2013.

[C-2] **Yuan He**, Hiroshi Sasaki, Shinobu Miwa, and Hiroshi Nakamura, "Predictmore Router: A Low Latency NoC Router with More Route Predictions," In Proc. of the 2013 IEEE International Parallel and Distributed Processing Workshops and Phd Forum (the 3rd Workshop on Communication Architecture for Scalable Systems), pp.842-850, May 2013.

[C-3] **Yuan He**, Hiroki Matsutani, Hiroshi Sasaki, and Hiroshi Nakamura, "Data Compression on 2D and 3D Network-on-Chips for CMP," In Proc. of the 9th Symposium on Advanced Computing Systems and Infrastructures, pp.391-398, May 2011.

Patents

[P-1] 和 遠, 三輪 忍, 中村 宏, 「ルータ」, 特願 2013-111244, 出願日 2013 年 5 月 27 日.