To appear in the 18th IEEE International Symposium on High Performance Computer Architecture (HPCA-2012). This is the authors' edition.

Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on-Chip

Sheng Ma[†‡], Natalie Enright Jerger[‡], Zhiying Wang[†]
[†]School of Computer, National University of Defense Technology, Changsha, China
[‡]Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada
masheng@nudt.edu.cn, enright@eecg.toronto.edu, zywang@nudt.edu.cn

Abstract

Routing algorithms for networks-on-chip (NoCs) typically only have a small number of virtual channels (VCs) at their disposal. Limited VCs pose several challenges to the design of fully adaptive routing algorithms. First, fully adaptive routing algorithms based on previous deadlock-avoidance theories require a conservative VC re-allocation scheme: a VC can only be re-allocated when it is empty, which limits performance. We propose a novel VC re-allocation scheme, whole packet forwarding (WPF), which allows a non-empty VC to be re-allocated. WPF leverages the observation that the majority of packets in NoCs are short. We prove that WPF does not induce deadlock if the routing algorithm is deadlock-free using conservative VC re-allocation. WPF is an important extension of previous deadlock-avoidance theories. Second, to efficiently utilize WPF in VC-limited networks, we design a novel fully adaptive routing algorithm which maintains packet adaptivity without significant hardware cost. Compared with conservative VC re-allocation, WPF achieves an average 88.9% saturation throughput improvement in synthetic traffic patterns and an average 21.3% and maximal 37.8% speedup for PARSEC applications with heavy network loads. Our design also offers higher performance than several partially adaptive and deterministic routing algorithms. [1]

1 Introduction

Networks-on-chip (NoCs) have been proposed to meet the communication requirements of many-core computing platforms [7]. NoC performance is sensitive to the choice of routing algorithm, as the routing algorithm defines not only the packet transmission latency, but also the saturation throughput a NoC can sustain. Many novel routing algorithms have been proposed to deliver high performance in NoCs [19, 21, 24, 28, 31, 43, 50].

[1]This research was carried out while Sheng Ma was a visiting international student at the University of Toronto supported by a CSC scholarship.

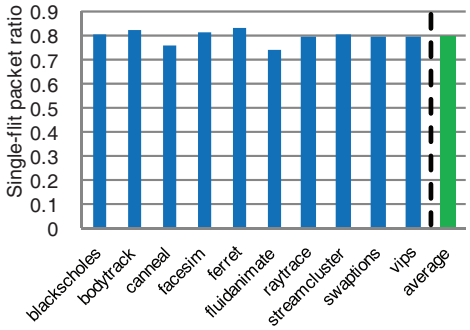


Figure 1. Single-flit packet ratio for the PARSEC benchmarks. (Flit width is 16 bytes.)

In addition to performance considerations, the routing algorithm has correctness implications for the network. Since deadlock is unacceptable, any proposed routing algorithm must be deadlock free, at both the network- and protocol-level. The guarantee of network-level deadlock freedom for a routing algorithm is generally based on deadlock-avoidance theories. There are many theories for deadlock-free fully adaptive [12, 13, 18, 29, 41, 48] and partially adaptive routing algorithm design [4, 6, 19, 20]. Although most theories were originally proposed for off-chip networks, they are widely used in today's NoCs [19, 21, 24, 28, 31, 43, 50].

However, the characteristics of packets in NoCs are quite different than those in off-chip networks. Abundant wiring resources lead to wider flits which decreases the number of flits per packet; short packets dominate traffic in NoCs. In contrast, the wiring resources in off-chip networks are limited by the pin count. For example, the flit width of a typical off-chip router is on the order of 32 bits (e.g. the Alpha 21364 router [35]), while the flit width of a NoC is typically between 128 [22] and 256 bits [10]. With such wide flits, coherence messages carrying a memory address and control information but no data can be encoded as single-flit packets in NoCs. Figure 1 shows that on average 78.7% of packets are single flits for PARSEC benchmarks [3]; the remaining packets are 5 flits long and contain a full 64B cache line.

Another noteworthy difference is that the buffer resources in NoCs are more precious than in off-chip networks due to the tight area and power budgets [17, 23], thus NoCs generally use flit-based wormhole flow control [9]. Although buffer resources are limited, several separate physical or virtual networks are leveraged for delivering different types of messages to avoid protocol-level deadlock. Table 1 lists the number of separate physical and virtual networks deployed in some industrial designs of off-chip and on-chip networks. We also show the number of required virtual networks for some cache coherence protocols in the GEMS simulator [34]. Typically, four or five virtual networks are needed to avoid protocol-level deadlock. Considering the expense of buffers in NoCs, each virtual network will be configured with a small number of VCs [5] since more VCs require more buffers and a larger allocator. For example, TILE64 [49] and TRIPS [22] have only one VC per virtual network. Thus, a NoC routing algorithm is generally running with a limited number of VCs.

In a VC-limited network with short packets dominating traffic, the design of fully adaptive routing algorithms faces several new challenges. In a wormhole network, fully adaptive routing algorithms based on existing theories require a conservative VC re-allocation scheme: a VC can only be re-allocated to a new packet when it is empty [12, 13, 18, 29, 41, 48]. This conservative scheme prevents network-level deadlock, but it is very restrictive resulting in bandwidth and performance loss in the presence of many back-to-back short packets [19]. Figure 2 illustrates the performance of three algorithms when each virtual network is configured with two VCs[2]. Despite its flexibility and load-balancing capability, the fully adaptive routing algorithm has even poorer performance than the deterministic and partially adaptive ones, since both the deterministic and partially adaptive algorithms can apply an aggressive VC re-allocation scheme. It is imperative to improve deadlock-avoidance theories to enhance the performance of fully adaptive routing algorithms in NoCs.

We propose a novel VC re-allocation scheme: whole packet forwarding (WPF), for fully adaptive routing algorithms. This scheme is summarized as follows: if a non-empty VC has enough buffer slots to hold the whole packet, then this VC can be re-allocated to the new packet even though it is not empty. WPF can be viewed as applying packet-based flow control in a wormhole network. This hybrid flow control mechanism solves the shortcoming of conservative VC re-allocation by allowing a VC to be re-allocated before it is empty, which greatly improves the saturation throughput of fully adaptive routing algorithms. We prove that a fully adaptive routing algorithm using WPF is deadlock-free if this routing algorithm is deadlock-free with

[2]See Section 5 for detailed experimental configuration and description of the routing algorithms.

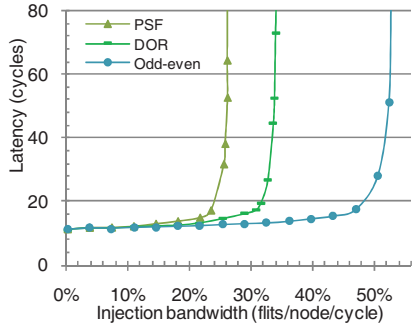


Figure 2. Routing algorithms performance with bit reverse traffic. (PSF: fully adaptive routing, DOR: deterministic routing; Odd-even: partially adaptive routing.)

conservative VC re-allocation. WPF is an important extension to previous deadlock-avoidance theories.

WPF enables the design of a fully adaptive routing algorithm with superior performance in a VC-limited network. Our novel routing algorithm achieves high VC utilization and maximal routing flexibility. Compared with conservative VC re-allocation, WPF provides an average saturation throughput improvement of 88.9% for synthetic traffic, and achieves an average 21.3% and maximal 37.8% speedup for PARSEC applications that heavily load the network. Our design also offers higher performance than several partially adaptive and deterministic routing algorithms. In summary, this paper makes the following primary contributions:

- Proposes WPF, which greatly improves the performance of fully adaptive routing algorithms, especially with limited VC resources.
- Proves WPF can be used by most previous deadlock-free fully adaptive routing algorithms; it is an important extension to existing deadlock-avoidance theories.
- Demonstrates that in a VC-limited network, maintaining packet adaptivity is very important and proposes an efficient fully adaptive routing algorithm that takes advantage of WPF.

2 Background

In this section, we discuss related work in deadlock-avoidance theories and design methodologies for fully adaptive routing algorithms.

2.1 Deadlock Avoidance Theories

Since NoCs typically use wormhole flow control [9] to reduce buffering requirements [7, 33, 39], we focus on theories for wormhole networks. Dally and Seitz proposed a seminal deadlock avoidance theory [6] which can be used to design partially adaptive and deterministic routing algorithms. Duato introduced the concept of a routing subfunction, and gave an efficient design methodology [12, 13]. Lin *et al.* [29] leveraged the message flow model, and Schwiebert and Jayasimha [41] utilized the channel waiting

Table 1. Number of physical/virtual networks. (PN: physical network; VN: virtual network)

Industrial products			Cache coherence protocols in GEMS simulator [34]		
Alpha 21364 [35]	TILE64 [49]	TRIPS [22]	MESI directory	MOESI directory	MOESI token
1 PN (7 VNs)	5 PNs (1 VN/PN)	2 PNs (4 VNs for OCN, 1 VN for OPN)	5 VNs	4 VNs	4 VNs

graph to analyze deadlock properties. Recently, Verbeek and Schmaltz [47, 48] proposed a necessary and sufficient condition for deadlock-free routing based on static conditions. These theories [12, 13, 18, 29, 41, 48] can be used to design fully adaptive routing algorithms.

A limitation of these theories for fully adaptive routing is that they all require that a VC be re-allocated only when it is empty [12, 13, 18, 29, 41, 48]. This requirement guarantees that all blocked packets can reach the head of a VC to gain access to the 'deadlock-free' path at every router. However, considering the large fraction of short packets in NoCs, strict adherence to this requirement strongly limits performance, especially when the number of VCs is small. To address this issue, some deadlock-recovery based designs [1] or theories [15] are proposed, which remove the constraint of conservative VC re-allocation. They allow the formation of deadlocks, and then apply some recovery mechanism [1, 15]. In contrast, WPF extends existing deadlock-avoidance theories which prohibit the formation of deadlock. To the best of our knowledge, WPF is the first proposal which allows multiple packets to reside in a VC concurrently for routing algorithms based on these previous deadlock-avoidance theories [12, 13, 18, 29, 41, 48].

Several partially adaptive algorithms based on the turn model have been proposed: negative-first, north-last, west-first [20] and odd-even [4]. The Abacus turn model is a dynamically reconfigurable routing algorithm [19]. These partially adaptive algorithms allow aggressive VC re-allocation: a VC can be re-allocated as soon as the tail flit of the last packet arrives [8]. This property can be directly deduced from Dally and Seitz's theory [6] since the channel dependency graphs of these algorithms are acyclic. However, they all suffer from limited adaptivity: packets cannot use all minimal paths between the source and destination, while fully adaptive ones can use all minimal paths.

2.2 Fully Adaptive Routing Algorithms

Duato's theory [12, 13] is widely used in the design of fully adaptive routing algorithms. In this theory, VCs are classified into two types: escape and adaptive. In the event of deadlock among the adaptive VCs, packets must have the opportunity to 'escape' to a deadlock-free set of VCs, known as escape VCs. Escape VCs are kept deadlock free by applying a more restrictive routing algorithm; dimension order routing (DOR) is typically used. An escape VC can only be used by a packet whose output port selection corresponds to a DOR path.

Many algorithms based on Duato's theory [21, 31, 35, 50] are composed of two parts: the routing function and selection strategy. If the selection strategy selects one output port, the packet can only request VCs that belong to the chosen output port. This requirement imposes a limitation on these algorithms: once a packet enters an escape VC, it can only use escape VCs until it is delivered. Otherwise, the escape VC may be involved in deadlock. In a VC-limited network, such as a cache-coherent NoC, this limitation easily results in adaptivity loss. However, Duato's theory supports the design of algorithms which can use an adaptive VC after using an escape VC if packets are always guaranteed to be able to use escape VCs [12, 13]. Based on these observations, we propose a design which maintains high adaptivity for packet routing, works well in a VC-limited environment and has low hardware overhead.

3 Motivation

In this section, we analyze the requirements of fully adaptive routing algorithms. We also illustrate how these requirements negatively affect performance.

3.1 VC Re-allocation Scheme

One limitation of fully adaptive routing algorithms is that at any time, a VC can hold at most one packet; a VC can only be re-allocated when it is empty. This is a reasonable requirement since fully adaptive routing algorithms put no limitation on the routing of some VCs and allow a cycle to form among VCs. For example, in routing algorithms based on Duato's theory, the adaptive VCs can be arbitrarily used [12, 13]. If multiple packets are allowed to reside in the same VC, a deadlock configuration is easily formed. Figure 3 illustrates a deadlock configuration [15]. Here each VN is configured with two VCs: an adaptive VC (AVC) and an escape VC (EVC). Configuring more VCs cannot eliminate this deadlock scenario since cycles are allowed to existed among adaptive VCs.

Eight packets are involved in this deadlock: P_0 - P_7. The head flit of P_0 is behind the tail flit of P_1 in AVC_1. The same is true for P_1, P_2, P_4, P_5 and P_6. Although the head flits of P_3 and P_7 are at the head of AVC_3 and AVC_6, they cannot move forward as the two valid output VCs, AVC_0 and EVC_0, are both occupied by other packets. No packet can move forward. This deadlock is due to that some head flits are not at the VC heads, resulting some packets unable to gain access to the 'deadlock-free' path. Also, the tail flits of these packets resides in other VCs, prohibiting following packets to reach the head of these VCs or even utilize these VCs. The following packets then may cyclically block aforementioned packets. For example, the tail flit of P_0 resides in AVC_0, blocking P_3 from utilizing this VC, which cyclically blocks the routing of P_0.

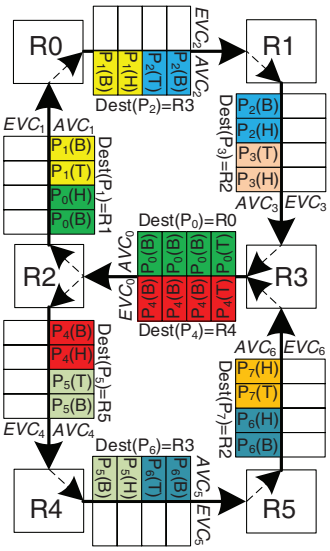


Figure 3. Deadlock in a fully adaptive routing algorithm if multiple packets are allowed to reside in one VC. (AVC: adaptive VC; EVC: escape VC; P_i(H), P_i(B) and P_i(T): the header, body and tail flit of Packet P_i, respectively; Dest(P_i): the destination of Packet P_i.)

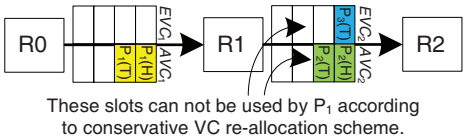
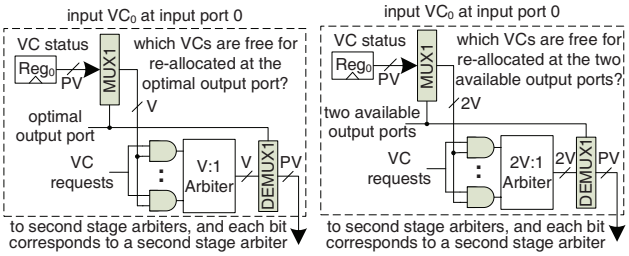


Figure 4. A VC underutilization scenario with conservative VC re-allocation. (AVC: adaptive VC; EVC: escape VC; P_i(H) and P_i(T): the head and tail flit of Packet P_i.)

If the packet length is greater than the VC depth, allowing multiple packets to reside in one VC easily results in deadlock. When a long packet enters a non-empty VC, its header flit is not at the head of a VC, while its tail flit blocks the head of another VC. However, due to the abundant wiring on chip, NoC traffic is dominated by short packets. With many short or single-flit packets, strictly adhering to the requirement that a VC can only hold one packet reduces throughput and results in VC underutilization. In Figure 4, neither EVC_2 nor AVC_2 are available for re-allocation; P_1 must wait in AVC_1 until either EVC_2 of AVC_2 becomes empty. However, since P_1 consists of only two flits, and both EVC_2 and AVC_2 have enough slots to hold this packet, forwarding P_1 into these VCs will not prevent following packets from getting to the head of AVC_1, as was the case in Figure 3. This is an opportunity for performance optimization. We will prove that P_1 can be forwarded into EVC_2 or AVC_2 without leading to deadlock.

3.2 Packet Adaptivity

In this section, we focus on maintaining packet adaptivity in VC-limited environments. Many fully adaptive routing algorithms based on Duato's theory are composed of



(a) Port-selection-first design. (b) Naive design.

Figure 5. The structure of a first stage arbiter in a VC allocator for one VN. (Each VN is configured with V VCs and P input/output ports.)

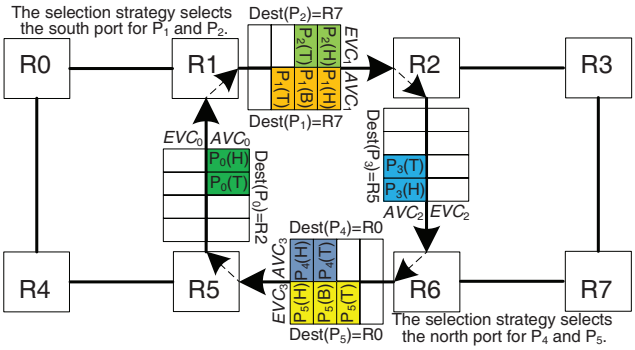


Figure 6. A deadlock configuration if packets in EVCs can apply for AVCs in port-selection-first algorithms.

two parts: routing function and selection strategy [21, 31, 35, 50]. The routing function computes all available output ports, then the selection strategy selects one of them. Once the selection strategy makes a choice, the packet can only request VCs for this particular port. This type of routing algorithm is called *port-selection-first*. Assuming a separable VC allocator consisting of two stages of arbiters [2, 36, 40], a port-selection-first algorithm only requires $V : 1$ arbiters in the first stage as shown in Figure 5(a).

A limitation of these algorithms is that once a packet enters an escape VC, it must continue to use escape VCs; the packet will lose adaptivity in subsequent hops. Violating this limitation results in deadlock as shown in Figure 6. In this example, both south and east ports are available for P_1 and P_2. If the selection strategy chooses the south port, P_1 and P_2 can only apply for AVC_2. They cannot request EVC_2 because escape VCs can be only used when the chosen port adheres to DOR. Similarly, the selection strategy chooses the north port for P_4 and P_5; they can only apply for AVC_0. No packet can move forward. Thus, the limitation that once a packet enters an escape VC, it can only use escape VCs until delivered is necessary for port-selection-first algorithms. However, this requirement results in significant adaptivity loss with limited VCs, since packets have a high probability of going into escape VCs.

Duato's theory supports the design of algorithms which allows a packet to use adaptive VCs after using escape VCs,

if it satisfies the following condition: a packet must be able to request an escape VC at any time. Once satisfy this condition, packets can always find a path whose VCs are not involved into cyclic dependencies since the extended channel dependency graph of escape VCs is acyclic [12, 13]. To achieve this target, a packet could be allowed to request all available output VCs, since at least one output port must adhere to DOR and the packet can use the escape VC of this port [12, 13]. However, this naive design results in additional hardware overhead. As shown in Figure 5(b), the VC allocator must use $2V : 1$ arbiters in the first stage to cover the at most two available output ports for minimal routing algorithms. Based on these observations, we propose a novel design which maintains significant packet adaptivity with only minor additional hardware.

4 Whole Packet Forwarding and Fully Adaptive Routing

In this section, we present our whole packet forwarding scheme and prove it is deadlock-free. Next, we design a routing algorithm which maintains packet adaptivity without significant hardware costs. Finally, we describe the hardware design and overhead.

4.1 Whole Packet Forwarding

As described in Section 3.1, existing fully adaptive routing algorithms use conservative VC re-allocation to prevent deadlock. However, this results in poor VC utilization. Therefore, we propose a novel VC re-allocation scheme: whole packet forwarding, which greatly improves VC utilization while not inducing deadlock. Suppose a packet P_k with length of $length(P_k)$ currently resides in VC_i, and VC_j is downstream of VC_i. Assume that the routing algorithm allows packet P_k to use VC_j. With conservative VC re-allocation, VC_j can be re-allocated to P_k only if the tail flit of its most recently allocated packet has been sent out, i.e., it is currently empty [8]. For our proposed VC re-allocation scheme, VC_j can be re-allocated if it already holds the tail flit of the most recently allocated packet, and the current free buffer count ($free\_slots(VC_j)$) is greater than or equal to $length(P_k)$. If $free\_slots(VC_j) \geq length(P_k)$, then all flits of P_k are guaranteed to be sent to VC_j after a limited time[3]. We call this VC re-allocation scheme *whole packet forwarding (WPF)*.

Figure 7 shows a WPF example. Here, the routing algorithm allows P_1 to use VC_2. VC_2 has already received the tail flit of P_2 and its free buffer count is two; this space is sufficient to hold the whole packet P_1 which consists of two flits. As a result, if we use WPF to re-allocate VC_2 to P_1, all flits of P_1 will be sent to VC_2 in a limited time. WPF forwards a packet into a non-empty VC

[3]The time to send all flits of P_k to VC_j will be determined by the congestion and switch allocation but all flits of P_k are guaranteed to advance.

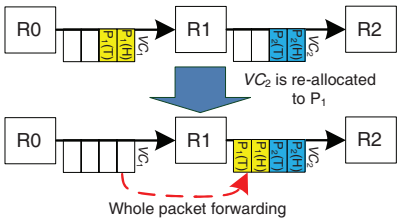


Figure 7. An example of whole packet forwarding.

if it has enough free buffers to hold the whole packet. In this case, WPF works similarly to packet-based flow controls such as store-and-forward (SAF) [16] and virtual cut-through (VCT) [26]. However, if the downstream VC is empty, we still use wormhole flow control, which does not require the empty VC to have enough slots to hold the whole packet; this reduces the buffering requirements compared to SAF and VCT. WPF can be viewed as applying packet-based flow control in a wormhole network. This hybrid flow control mechanism solves the shortcoming of conservative VC re-allocation for fully adaptive routing algorithms.

Our contention is that if the routing algorithm with conservative VC re-allocation is deadlock-free, then applying WPF to forward packets into non-empty VCs will not lead to deadlock. If the VC depth is larger than the maximum packet length, and the network applies packet-based flow controls, multiple packets are allowed to reside in one VC for fully adaptive routing algorithms [14]. However, for the wormhole network, a blocking packet may reside in multiple VCs, introducing two additional dependencies, indirect and cross indirect dependency, between non-neighboring channels [12, 13, 14]. With these additional dependencies, it is difficult to prove the deadlock-free property of WPF based on existing theories.

We first give a qualitative proof for algorithms based on Duato's theory: using WPF will never allow a packet to get stuck 'mid-way' between two routers, as packets will always either be able to be fully transmitted to non-empty VCs or otherwise they will be able to use the escape VCs. However, the 'escape VC' is only defined in Duato's theory, and other theories may not have this definition. Thus, we provide a general proof. For convention, we label the routing algorithm with conservative VC re-allocation as Alg; $Alg + WPF$ is Alg with WPF applied to allow forwarding of entire packets to non-empty VCs.

***Theorem* 1**: If Alg is deadlock-free, then $Alg + WPF$ is also deadlock-free.

Informal Description: Our proof is by contradiction. We prove that if there is a deadlock configuration for $Alg + WPF$, then there is a deadlock configuration for Alg as well. Using the deadlock configuration $Config_0$ shown in Figure 8 as an example, we remove these packets whose head flits are not at the heads of VCs, and get a new configuration $Config_1$. We prove that Alg can achieve $Config_1$, and $Config_1$ is a deadlock configuration. However, Alg is

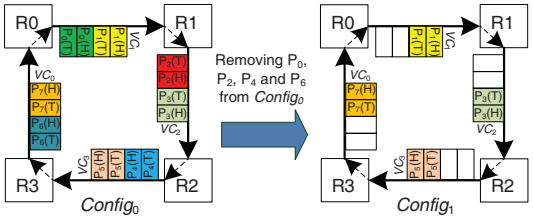


Figure 8. The construction of a new configuration based on $Config_0$ of $Alg + WPF$.

deadlock-free, thus there is no such configuration.

Proof: By contradiction. If $Alg + WPF$ is not deadlock-free, then there is a deadlock configuration ($Config_0$) in which a set of packets, P_{set_0} are waiting on VCs held by other packets in P_{set_0}. We prove that a deadlock configuration also exists for Alg. Our proof consists of three steps.

Step 1: We build a new configuration based on $Config_0$. Consider each packet P_i in P_{set_0}. If P_i is a packet whose header flit is not at the head of a VC, then this VC was allocated to P_i using WPF; therefore, all flits of P_i must reside in this VC in $Config_0$. We remove P_i from the network and label these removed packets as P_{subset_0}. We label the new configuration as $Config_1$, and the set of packets remaining in this configuration as P_{subset_1}.

Step 2: We prove that when the network is routed by Alg, all packets in P_{subset_1} can be forwarded into their current VCs in $Config_1$. We consider each packet P_j in P_{subset_1}. We further consider each hop hop_k of P_j when the network is routed by $Alg + WPF$. Without loss of generality, we assume the head flit of P_j is forwarded from VC_k to VC_{k+1} during hop_k. There are two situations for VC_{k+1}.

2.1) VC_{k+1} is empty when the head flit of P_j is forwarded into it; therefore, VC_{k+1} is allocated to P_j using conservative VC re-allocation. Thus, if the network is routed by Alg, P_j can use VC_{k+1}.

2.2) VC_{k+1} is not empty when the head flit of P_j is forwarded into it, thus VC_{k+1} is allocated to P_j using WPF. Since P_j can be forwarded into VC_{k+1}, the routing algorithm allows P_j to use VC_{k+1}. However, if the network is routed by Alg, P_j cannot be forwarded into VC_{k+1} until it is empty. Since Alg is deadlock-free, the packet currently residing in VC_{k+1} must be sent out in a limited time. Then VC_{k+1} can be re-allocated to P_j using conservative VC re-allocation. Thus, if the network is routed by Alg, P_j can use VC_{k+1}.

Considering 2.1) and 2.2) together, for each hop, if a VC is used by P_j when the network is routed by $Alg + WPF$, this VC can be also used by P_j when the network is routed by Alg. Thus, P_j can be routed to its current VC(s) in $Config_1$ by Alg.

Step 3: We prove that $Config_1$ is a deadlock configuration for Alg. For each P_i in the removed packet set P_{subset_0}, all flits of P_i reside in one VC but the head flit of P_i is not at the head of its VC. Thus, removing P_i from the network does not create an empty VC; each VC now holds flits of only one packet. Alg utilizes conservative VC re-allocation which only allows empty VCs to be re-allocated. Therefore all packets in the remaining packet set P_{subset_1} still wait for VCs held by other packets in P_{subset_1}. Thus, $Config_1$ is a deadlock configuration for Alg, but Alg is deadlock-free, so there is no such deadlock configuration. Thus, $Alg+WPF$ is deadlock-free as well. □

Note that our proof does not make any assumption about the routing algorithm; WPF can be utilized with any fully adaptive routing algorithm if it is deadlock-free using conservative VC re-allocation. WPF removes the constraints of conservative VC re-allocation. Thus, it is an important extension of these theories.

4.2 Fully Adaptive Routing Algorithm

As demonstrated in Section 1, fully adaptive routing algorithms can yield worse performance than deterministic and partially adaptive ones in VC-limited networks. To combat this problem, we leverage WPF to design a novel fully adaptive routing algorithm with superior performance. Our design is based on Duato's theory [12, 13]. In a VC-limited NoC, the routing algorithm should maintain maximum routing flexibility; it should allow the use of adaptive VCs after using escape VCs. Otherwise, once a packet goes into escape VCs, it loses adaptivity in subsequent routing. The design must guarantee that at any time a packet is able to request an escape VC [12, 13].

We make a simple modification. In port-selection-first algorithms, the only time a packet cannot use an escape VC is when the selection strategy chooses an port that violates DOR. Our design allows the packet to violate the selection in this case; the packet can apply for the escape VC of the other port that was not selected in addition to adaptive VCs of the selected one. Using P_1 in Figure 6 as an example, if the selection strategy chooses the south port, our algorithm allows P_1 to request the escape VC of the east output port as well. If there is only one available port, this port must adhere to DOR, and the packet can request its escape VC. Our design guarantees that a packet always has an opportunity to use an escape VC. Thus, it allows a packet to move back into an adaptive VC after using an escape VC. It only needs $V : 1$ arbiters in the first stage of the VC allocator. Large arbiters result in more hardware overhead and introduce additional delay on the critical path.

4.3 Router Microarchitecture

The pipeline of a canonical NoC Router [8, 16, 36, 40] consists of four stages: routing computation (RC), VC allocation (VA), switch allocation (SA) and switch traversal (ST). Several optimizations are applied to achieve high baseline performance. The speculative switch allocation is used to parallelize VA and SA [40]. Look-ahead routing removes RC from the critical path; the adaptive routing al-

gorithm calculates at most two available output ports one hop ahead and applies a selection strategy to choose an optimal one [21, 27, 31]. The delay of the baseline router is 2 cycles plus an additional cycle for link traversal.

Both WPF and our routing algorithm only require simple modifications to the baseline VC allocator. They can be used with any type of VC allocator; we assume a separable VC allocator which is widely used due to its low complexity and high frequency [2, 36, 40]. In a separable VC allocator, each input VC determines which output VC of the selected output port to bid on in the first stage. The winning requests from the first stage then arbitrate for an output VC in the second stage. We modify the first stage arbiters.

First, we need to monitor whether a downstream VC is free to be re-allocated with WPF. The criterion is that the downstream VC holds the tail flit of its most recently allocated packet and still has enough free slots to hold the entire new packet. Calculating whether there are enough free buffer slots for a new packet introduces some hardware overhead. However, considering that cache coherence packets exhibit a bimodal distribution and long packets are generally longer that the VC depth, we focus on applying WPF to single-flit packets. Thus, if a downstream VC receives the tail flit of the most recently allocated packet, and it still has free slots, it can be re-allocated to a single-flit packet.

Figure 9 depicts our proposed VC allocator. Reg_0 records if a downstream VC is free to be re-allocated with conservative re-allocation; the downstream VC is currently empty. An additional register Reg_1 is needed to record whether a downstream VC is free to be re-allocated with WPF. Based on the incoming packet type, Reg_0 or Reg_1 is chosen as the input to MUX0. If the incoming packet is a single-flit packet, we apply WPF, choosing the contents of Reg_1 as the input for MUX1. Otherwise, the contents of Reg_0 are sent to MUX0. Updates to Reg_0 and Reg_1 are off the critical path since a router monitors the status of downstream VCs using credits [8]. The only increase in delay for WPF is an additional 2-input multiplexer: MUX0.

To support our new fully adaptive routing algorithm, we modify MUX1 and DEMUX1, as shown in Figure 9. MUX1 needs two additional input signals: *DOR* and *the other output port*. The *DOR* signal indicates if the chosen optimal output port obeys DOR or not. *The other output port* signal records the available output port that was not chosen. The routing computation logic produces these two signals. If the *DOR* signal is '0', then the selected output port violates DOR path. In this case, the status of the escape VC for *the other output port* rather than the chosen optimal one will be sent to the $V:1$ arbiter. This is accomplished with a 2-input multiplexer whose select signal is *DOR*. DEMUX1 also needs these two additional signals. If the *DOR* signal is '0', the result of $V:1$ arbiter is de-multiplexed to the second stage arbiter for the escape VC of *the other output port* instead of the chosen port. A 2-input demultiplexer implements this function. The increased delay for our new fully adaptive routing algorithm is an additional 2-input multiplexer and demultiplexer.

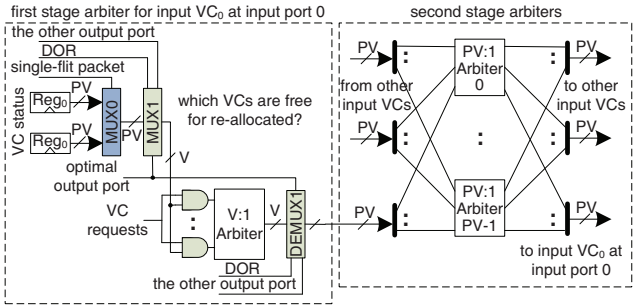


Figure 9. The proposed VC allocator for one VN.

Table 2. The critical path delay and area results.

Design	**Delay** (ns)	**Area** (μm^2)
Port-selection-first (Figure 5(a))	1.78	49437.4
Naive design (Figure 5(b))	1.92	56045.4
Proposed design (Figure 9)	1.79	49512.6

To analyze the hardware overhead, we implement the three VC allocators (Figures 5 and 9) in RTL Verilog for an open-source NoC router [2] and synthesize in Synopsys Design Compiler with a TSMC 65nm standard cell library. The designs operate at 500 MHz under normal conditions (1.2V, 25°C). We use simple round-robin arbiters [8]. This router has 5 ports ($P = 5$) and supports 4 VNs; each VN has 2 VCs ($V = 2$). Table 2 presents the area and critical path delay estimates. The naive design uses *4:1* arbiters in the first stage, resulting in a 7.9% longer critical path and 13.4% more area than the port-selection-first design. Our design uses *2:1* arbiters in the first stage and only increases the critical path by 0.5% and area by 0.2%. An allocator's power consumption is largely decided by the arbiter size [2, 50]; given the small arbiters in our design, there should be negligible power overhead compared with the port-selection-first design. However, we omit a detailed power evaluation as it depends on the activity factor of each signal.

5 Evaluation

We modify the cycle-accurate Booksim simulator [8] to model the microarchitecture discussed in Section 4. We compare the performance of our proposed fully adaptive routing algorithm with conservative VC re-allocation (FULLY) and with WPF (FULLY+WPF) against several routing algorithms. We implement a port-selection-first fully adaptive routing algorithm with conservative VC re-allocation (PSF) and with WPF (PSF+WPF). The deterministic routing algorithm is DOR. West-first, negative-first and odd-even represent partially adaptive algorithms. Since the design of selection strategy is orthogonal to this paper, we use a local selection strategy for all adaptive algorithms; when there are two available output ports, the selection strategy

Table 3. Baseline configuration and variations.

Characteristic	Baseline	Variations
Topology (mesh)	4×4	8×8
VCs/VN	2	4
Flit buffers/VC	4	3, 2
Packet length (flits)	long: 5, short: 1	-
SFP ratio	80%	60%, 40%
Warmup cycles, Total Cycles	10000, 100000	-

Table 4. Full system simulation configuration.

# of cores	16
L1 cache (D & I)	private, 4-way, 32KB each
L2 cache	private, 8-way, 512KB each
Cache coherence	MOESI distributed directory
Topology	4×4 Mesh

chooses the port with more free buffers.

We evaluate both synthetic traffic and real applications. For synthetic traffic patterns, we use one VN since each VN is independent. Our baseline configuration uses a 4×4 mesh with 2 VCs that are each 4 flits deep. The packet lengths exhibit a bimodal distribution; there are single-flit and five-flit packets. The baseline single-flit packet (SFP) ratio is 80%. Table 3 summarizes the baseline network configuration and the variations used in the sensitivity studies.

To measure full-system performance, we leverage two simulation frameworks: FeS2 [37] for x86 simulation and BookSim for NoC simulation. FeS2 is a timing-first, multiprocessor, x86 simulator, implemented as a module for Virtutech Simics [32]. We run PARSEC benchmarks [3] with 16 threads on a 16-core CMP. We assume cores optimized for clock frequency; they are clocked at a frequency 5× higher than the network. Prior research shows the frequency of simple cores in many-core platform can be optimized to 5∼10 GHZ, while the frequency of NoC router are limited by the allocator speed with a large number of VCs [11]. As we consider several VNs, more VCs are needed. Thus, it is reasonable to assume cores will be clocked faster than the network. Each core is connected to private, inclusive L1 and L2 caches. Cache lines are 64 bytes; long packets are 5 flits wide with a 16-byte flit width. We use a distributed, directory-based MOESI coherence protocol which needs 4 VNs for protocol-level deadlock freedom. Each VN has 2 VCs; each VC is 4 flits deep. All benchmarks use the *sims-mall* input sets to reduce simulation time. The total runtime is used as the metric for full-system performance. Table 4 presents the system configuration.

5.1 Performance of synthetic workloads

Figure 10 illustrates the performance of several routing algorithms in our baseline configuration using four synthetic traffic patterns: bit reverse, hotspot and 2 transpose patterns. Across these four patterns, the fully adaptive routing algorithms (PSF and FULLY) show the poorest performance. Although PSF and FULLY offers adaptiveness for all traffic, conservative VC re-allocation significantly limits their performance. In contrast, DOR and partially adaptive routing algorithms use aggressive VC re-allocation. For all four patterns, PSF's performance is worse than FULLY. PSF's performance is further limited by its poor flexibility: once a packet enters an escape VC, the packet can only be routed by DOR using escape VCs in subsequent hops.

For bit reverse traffic, a source node with bit address $\{s_3, s_2, s_1, s_0\}$ sends traffic to destination $\{s_0, s_1, s_2, s_3\}$. 62.5% of this traffic is between the north-east and south-west quadrants; negative-first offers adaptiveness for this traffic. Only 37.5% of the traffic is eastbound; west-first offers adaptiveness for this traffic, which leads to poorer performance than negative-first. The adaptiveness offered by odd-even is lower than negative-first, thus its performance is worse than negative-first. Although WPF improves the VC utilization for PSF+WPF, its saturation throughput is lower than odd-even and negative-first. PSF+WPF is still limited by its poor flexibility. FULLY+WPF provides high VC utilization and significant routing flexibility leading to the highest saturation throughput[4].

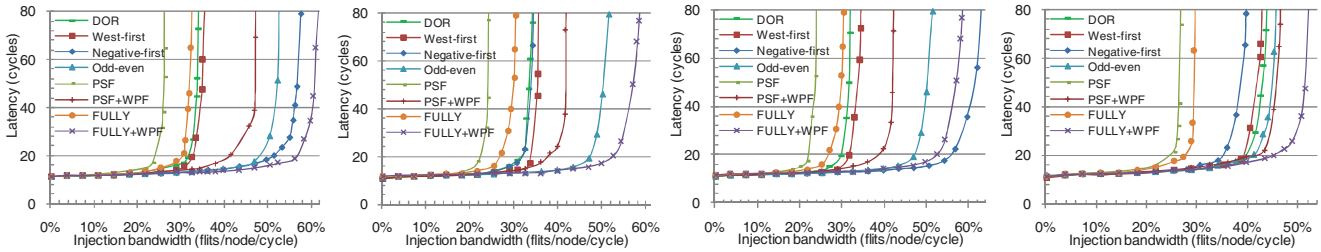
For transpose-1, a node (i, j) sends messages to node $(3 - j, 3 - i)$. Negative-first deteriorates to DOR for this pattern. West-first still offer adaptiveness for 37.5% of the traffic, thus it has better performance than negative-first. Odd-even offers greater adaptiveness than the other two partially adaptive algorithms and has higher performance. FULLY+WPF offers adaptiveness for all traffic, achieving 15.7% higher saturation throughput than odd-even.

Transpose-2 is a favorable pattern for negative-first; a node (i, j) sends messages to node (j, i). Negative-first offers adaptiveness for all traffic in this pattern and has the highest performance. Although FULLY+WPF offers adaptiveness for all packets as well, its performance is limited by the restriction on usage of the escape VCs: only if the output port adheres to DOR, can the escape VC be used. The performance of FULLY+WPF and odd-even with transpose-2 are very close to their performance with transpose-1 since the two transpose patterns are symmetric and these two algorithms offer the same adaptiveness for them.

With hotspot traffic, four nodes are chosen as hot spots and receive an extra 20% traffic in addition to the uniform random traffic. This pattern mimics memory controllers receiving a disproportionate amount of traffic. FULLY+WPF and odd-even algorithms show higher performance than negative-first and west-first ones, because they can offer greater adaptiveness. Due to the more limited adaptiveness offered by odd-even, its performance is worse than FULLY+WPF. DOR has better performance than negative-first and west-first, since DOR more evenly distributes uniform traffic which is used as the background in this pattern.

In summary, with conservative VC re-allocation, the fully adaptive algorithm has the worst performance. Negative-

[4]The saturation point is measured as the injection rate at which the average latency is 3 times the zero load latency.



(a) Bit reverse. (b) Transpose-1. (c) Transpose-2. (d) Hotspot.

Figure 10. Routing algorithm performance for the baseline configuration.

Table 5. Average saturation throughput improvement.

Algorithm	Improvement	Algorithm	Improvement
FULLY	88.9%	Odd-even	16.3%
DOR	64.5%	PSF	130.9%
West-first	58.6%	PSF+WPF	31.3%
Negative-first	26.6%		

first and west-first offer uneven adaptiveness for different patterns. For example, although negative-first achieves high performance for transpose-2, it deteriorates to DOR for transpose-1. As the traffic in NoCs changes throughout runtime and different VNs may run different traffic patterns, these partially adaptive routing algorithms are unsuitable. Odd-even offers limited adaptiveness for all traffic patterns. WPF improves VC utilization for fully adaptive routing algorithms. In a VC-limited environment, routing flexibility must also be considered to fully leverage WPF. PSF+WPF does not provide enough routing flexibility resulting in lower performance than some partially adaptive algorithms. FULLY+WPF provides high VC utilization and routing flexibility, leading to the best performance. Table 5 gives the average saturation throughput improvement of FULLY+WPF over the other algorithms across all four patterns. The 88.9% saturation throughput gap between FULLY+WPF and FULLY reflects the effect of WPF. The gap between FULLY+WPF and PSF+WPF represents of the effect of routing flexibility; sufficient flexibility leads to an average saturation throughput improvement of 31.3%.

5.2 Performance of PARSEC benchmarks

Figure 11 shows the speedups relative to PSF for the PARSEC workloads. We divide the 10 applications into 2 classes. For *blackscholes*, *fluidanimate*, *raytrace* and *swaptions*, different routing algorithms have similar performance. The working sets of these applications fit into the caches leading to a lightly loaded network. Their system performance is unaffected by techniques that improve throughput, such as sophisticated routing algorithms. However, the routing algorithm influences the performance of the remaining 6 applications; they exhibit bursty communication and have heavier loads. Their system performance is sensitive to network optimizations; routing algorithms with higher saturation throughputs improve performance. For example, FULLY+WPF has 48.5% and 43.0% speedup over PSF for *facesim* and *streamcluster*. West-first has the best performance for *vips* because most of its bursty communication is eastbound. For *facesim* and *streamcluster*, negative-first offers higher adaptiveness than odd-even for bursty communication, thus achieving higher performance. For all heavy load applications except *vips*, FULLY+WPF has the best performance. Across these applications, FULLY+WPF achieves an average of 21.3% and maximum 37.8% speedup over FULLY. With sufficient flexibility, FULLY+WPF has an average 12.1% speedup over PSF+WPF. The average speedups of FULLY+WPF are 29.3%, 15.0%, 10.1%, 9.9% and 10.4% over PSF, DOR, west-first, negative-first and odd-even, respectively. Our design supports higher saturation throughput; if higher saturation throughput is not required, WPF can benefit the NoC by reducing network resources such as reducing the buffer resources and reducing the channel width.

6 Sensitivity to Network Design

Individual network implementations are likely to vary from our baseline configuration, depending on the needs of the system. We explore variations for further insight. Except for the analyzed parameter, the other parameters are the same as the baseline (Table 3).

Single-flit packet ratios. Single-flit packet (SFP) ratios depend on the running application, the cache hierarchy and the coherence protocol. To test the robustness of WPF, we evaluate 60% and 40% SFP ratios for transpose-1 traffic. As illustrated in Figure 12, DOR, west-first, negative-first and odd-even exhibit nearly identical performance for different SFP ratios. Since they apply aggressive VC re-allocation, their performance is not sensitive to packet length. However, the performance of PSF and FULLY improves as the SFP ratio shrinks. The conservative VC re-allocation used by PSF and FULLY favors long packets since they utilize buffers more efficiently than short ones. As the SFP ratio decreases so does the possibility of applying WPF. Thus, the performance gap between FULLY+WPF and FULLY (or PSF+WPF and PSF) decreases. However, even with a 40% SFP ratio, FULLY+WPF achieves a 53.1% saturation throughput improvement over FULLY. For these different SFP ratios, FULLY+WPF has the best performance.

VC depth. Different NoCs may use different VC depths.

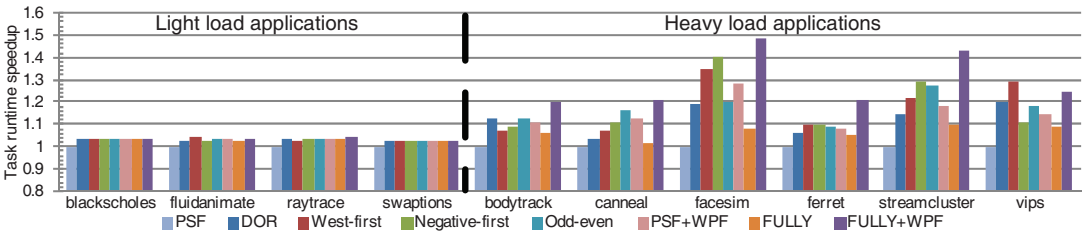
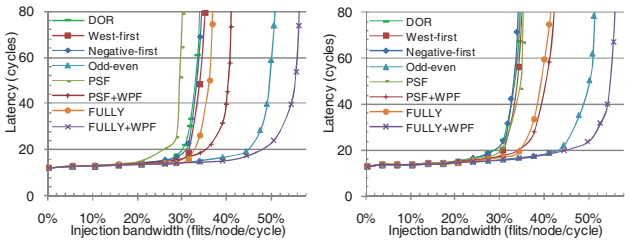
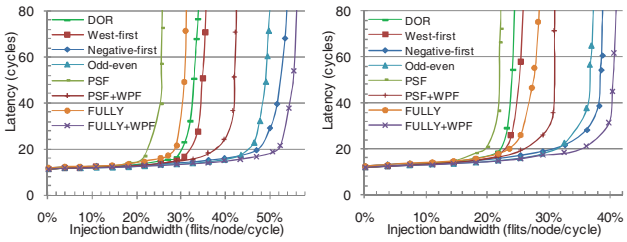


Figure 11. System speedup for PARSEC benchmarks.



(a) 60% SFP ratio (b) 40% SFP ratio.

Figure 12. The performance with different SFP ratios.



(a) 3-flit deep VCs. (b) 2-flit deep VCs.

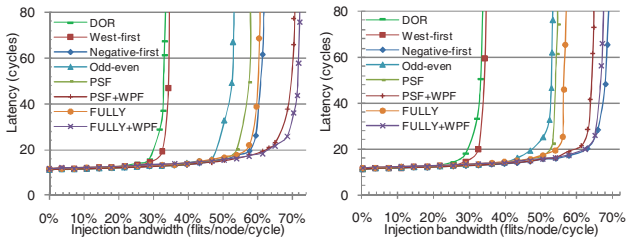
Figure 13. The performance with different VC depths.

To test the flexibility of WPF, we further evaluate 3- and 2-flit deep VCs. Comparing 4 flits/VC (Figure 10(a)) and 3 flits/VC (Figure 13(a)), the performance of DOR and west-first remain almost constant, while FULLY and PSF exhibit minor performance degradation. DOR and west-first offer no or very limited adaptiveness which is a major factor in their performance. Thus, reducing the VC depth from 4 to 3 has little effect. The bottleneck of FULLY and PSF is conservative VC re-allocation. Considering the majority of single-flit packets, reducing the VC depth from 4 to 3 only affects performance slightly. However, the performance of FULLY+WPF, PSF+WPF, odd-even and negative-first declines with shallower VCs since the VC depth is a bottleneck for them. Shallow VCs increase the number of hops that a blocked packet spans, which increases the effect of chained blocking [46].

Comparing 3 and 2 flits/VC, performance drops for all algorithms. FULLY has better performance than DOR and west-first with 2 flits/VC. As VC depth decreases, the difference between aggressive and conservative VC re-allocation declines. FULLY has a relative performance improvement. However, even with only 2 flits/VC, WPF still optimizes the performance since short packets dominate traffic. In Figure 13(b), FULLY+WPF has a 46.2% saturation throughput improvement over FULLY. Applying WPF on fully adaptive routing algorithms leads to superior performance even with half of the buffer resources; enabling the design of a very low-cost NoC. With 2 flits/VC (Figure 13(b)), FULLY+WPF achieves a saturation throughput of 40.3%, while the saturation throughput of FULLY is 32.3% with 4 flits/VC (Figure 10(a)). The same is true for PSF+WPF with 2 flits/VC and PSF with 4 flits/VC.

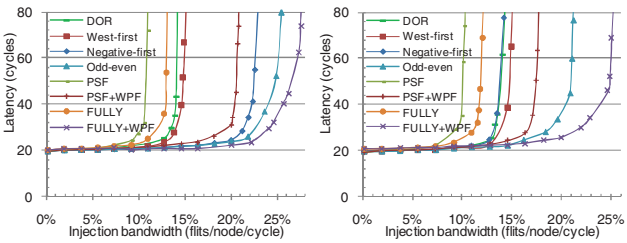
VC count. As semiconductor scaling continues, a VN may be configured with more VCs. Coherence protocols may be optimized to reduce the number of required VNs allowing more VCs per VN. Comparing Figure 14(a) (4 VCs/VN) and Figure 10(a) (2 VCs/VN), the performance of DOR, west-first and odd-even is almost the same. These algorithms offer limited adaptiveness; although additional experimental results show increasing the VC count from 1 to 2 improves performance, increasing the VC count from 2 to 4 cannot reduce the congestion for physical paths and does not further improve performance. Negative-first has a modest performance improvement. In contrast, PSF, FULLY, PSF+WPF and FULLY+WPF all have significant improvement; more VCs mitigate the negative effects of conservative VC re-allocation. The performance difference between PSF and FULLY (or PSF+WPF and FULLY+WPF) decreases with more VCs; more VCs reduce the possibility of using escape VCs in PSF which limits the packets that lose adaptivity.

Figure 14(b) shows the performance of transpose-2, which is a favorable pattern for negative-first. FULLY+WPF achieves almost the same performance as negative-first; with more VCs, the effect of restricting the use of escape VCs in FULLY+WPF declines. With more VCs, the gap between FULLY and FULLY+WPF (or PSF and PSF+WPF) diminishes. More VCs improves the possibility of a packet being forwarded into an empty VC, thus improving the performance of FULLY (or PSF). Furthermore, using WPF to forward a new packet into a non-empty VC may result in head-of-line congestion [8] and degrade the performance of FULLY+WPF (or PSF+WPF). Nevertheless, FULLY+WPF still shows an average 19.8% saturation throughput improvement over FULLY for these two patterns with 4 VCs; providing high VC utilization strongly outweighs the negative effect of HoL blocking in a VC-limited environment. Yet, additional experimental results



(a) Bit reverse. (b) Transpose-2.

Figure 14. The performance with 4 VCs/VN.



(a) Bit reverse. (b) Transpose-1.

Figure 15. The performance for an 8×8 mesh network.

show that WPF results in minor performance degradation with 8 VCs/VN for some traffic patterns. Similarly, aggressive VC re-allocation slightly degrades the performance of deterministic and partially adaptive algorithms compred to conservative VC re-allocation with 8 VCs/VN for some patterns. With abundant VCs, the HoL blocking outweighs the positive effect of high VC utilization. Similar to VC depth, with only 2 VCs, FULLY+WPF offers near or even better performance (Figures 10(a) and 10(b)) than FULLY with 4 VCs (Figure 14) across these two patterns; WPF provides similar or higher performance with half as many VCs.

Network size. Figure 15 explores the scalability of our design for an 8×8 mesh. The trends across different algorithms are the same as with the 4×4 mesh (Figure 10). Communication is determined by the traffic pattern not the network size. Since larger networks lead to higher average hop counts [31], a larger network puts more pressure on VCs than a smaller one. WPF achieves greater performance improvement in a larger network. The average saturation throughput improvement for these two patterns of FULLY+WPF over FULLY is 108.2%, while in a 4×4 mesh, it is 93.1%. As packets must travel more hops in a larger network, the possibility of entering an escape VC increases. For PSF and PSF+WPF, once the packet enters an escape VC, it loses adaptivity in subsequent hops. Therefore the performance gap between FULLY+WPF and PSF+WPF (or FULLY and PSF) increases with a larger network; providing routing flexibility becomes more important with a larger network.

In summary, although the effect of WPF decreases with smaller SFP ratios, shallower VC depths or more VCs, applying WPF with fully adaptive routing still improves performance significantly. The effect of WPF as well as routing flexibility increases with a larger network. Applying WPF to fully adaptive routing algorithms provides similar or even better performance with half of the buffer resources or VCs as a network that does not employ WPF.

7 Further Discussion and Future Work

Packet length and VC depth. Packet lengths for cache coherence traffic typically have a bimodal distribution. However, optimizations such as cache line compression [11, 25] create packet distributions that are not bimodal; the packet length may be distributed between a single flit and the maximum flits per packet supported by the architecture. To apply WPF on such NoCs, more downstream VC status registers are needed for the first-stage arbiters shown in Figure 9. An important consideration is how many different packet lengths to apply WPF to. The longest packet length that can use WPF is one flit shorter than the VC depth. Designers can ignore long packets, since they have few opportunities to apply WPF. This tradeoff depends on the packet length distribution, VC depth, hardware overhead and the expected performance gain. Delving into this tradeoff is left for future work. In this paper, we assume the VC depth is shorter than the maximum packet length. If the VC depth exceeds the maximum packet length, conservative VC re-allocation results in poorer VC utilization for wormhole than VCT, and applying WPF in this case behaves the same as VCT. The contribution of WPF is that it allows multiple packets to reside in one VC while allowing the VC depth to be shorter than the maximum packet length, thus giving designers more flexibility.

DAMQ and hybrid flow controls. Previous research proposed the dynamically allocated multi-queue (DAMQ) designs for both off-chip [45] and on-chip networks [38, 50] to improve the VC utilization. Even with DAMQ, allowing multiple packets to reside in one VC may lead to a deadlock configuration similar to Figure 3 for fully adaptive routing algorithms in a wormhole network. WPF is complimentary to DAMQ as it ensures deadlock-freedom and improves the design flexibility. WPF can viewed as a hybrid mechanism combining wormhole and VCT. There are some previous hybrid flow controls [42, 44, 30]. Hybrid switching [42] and buffered wormhole [44] remove a blocked wormhole packet to release held physical channels by utilizing either the processing node memory [42] or a central buffer [44]. Layered switching divides the long wormhole packets into several groups and tries to keep the switch allocation grants for a whole group [30]. The purpose of WPF is quite different; we focus on improving the performance for fully adaptive routing algorithms in wormhole networks.

8 Conclusion

Whole packet forwarding is a novel VC re-allocation scheme for fully adaptive routing algorithms in wormhole networks. This scheme allows multiple packets to reside in one VC concurrently; it greatly improves VC utilization in VC-limited networks where short packets dominate traffic. We prove that WPF does not lead to deadlock if the algorithm is deadlock-free with conservative VC re-allocation. Thus, WPF is an important extension to existing deadlock-avoidance theories. We further propose a novel fully adaptive routing algorithm that exploits WPF and provides routing flexibility with modest hardware overhead. Compared with conservative VC re-allocation, WPF improves the saturation throughput by 88.9% on average in synthetic traffic patterns and achieves up to 37.8% (21.3% average) full-system speedup for network-intensive PARSEC benchmarks, and offers similar or even better performance with half of the buffer resources or VCs.

Acknowledgments

We thank the anonymous reviewers for their helpful suggestions and members of Prof. Enright Jerger's group for feedback. We also thank Daniel Becker of Stanford for his explanation on the open-source router implementation. This work is supported by the University of Toronto, NSERC of Canada, the Connaught Fund, 863 Program of China (2012AA010302), NSFC (61070037, 61025009, 60903039, 61103016), China Edu. Fund. (20094307120012), Hunan Prov. Innov. Fund. For PostGrad. (CX2010B032).

References

[1] K. Anjan and T. Pinkston. An efficient, fully adaptive deadlock recovery scheme: Disha. In *ISCA 1995*.

[2] D. Becker and W. Dally. Allocator implementations for network-on-chip routers. In *SC 2009*.

[3] C. Bienia *et al*. The parsec benchmark suite: characterization and architectural implications. In *PACT 2008*.

[4] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Trans. Parallel Distrib. Syst.*, 11(7):729 –738, July 2000.

[5] W. Dally. Virtual-channel flow control. *IEEE Trans. Parallel Distrib. Syst.*, 3(2):194 –205, Mar 1992.

[6] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comp.*, C-36(5):547 –553, May 1987.

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

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

[9] W. Dally and C. Seitz. The Torus routing chip. *Distributed Computing*, 1:187–196, 1986.

[10] R. Das *et al*. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *HPCA 2009*.

[11] R. Das *et al*. Performance and power optimization through data compression in network-on-chip architectures. In *HPCA 2008*.

[12] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 4(12):1320 –1331, December 1993.

[13] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 6(10):1055 –1067, 1995.

[14] J. Duato. A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks. *IEEE Trans. Parallel Distrib. Syst.*, 7(8):841 –854, August 1996.

[15] J. Duato and T. Pinkston. A general theory for deadlock-free adaptive routing using a mixed set of resources. *IEEE Trans. Parallel Distrib. Syst.*, 12(12):1219 –1235, dec 2001.

[16] N. Enright Jerger and L. Peh. *On-Chip Networks*. Morgan and Claypool Publishers, 2009.

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

[18] E. Fleury and P. Fraigniaud. A general theory for deadlock avoidance in wormhole-routed networks. *IEEE Trans. Parallel Distrib. Syst.*, 9:626–638, July 1998.

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

[20] C. Glass and L. Ni. The turn model for adaptive routing. In *ISCA 1992*.

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

[22] P. Gratz *et al*. On-chip interconnection networks of the TRIPS chip. *Micro, IEEE*, 27(5):41 –50, Sept.-Oct. 2007.

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

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

[25] Y. Jin *et al*. Adaptive data compression for high-performance low-power on-chip networks. In *MICRO 2008*.

[26] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 1979.

[27] J. Kim *et al*. A low latency router supporting adaptivity for on-chip interconnects. In *DAC 2005*.

[28] M. Li *et al*. DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *DAC 2006*.

[29] X. Lin *et al*. The message flow model for routing in wormhole-routed networks. *IEEE Trans. Parallel Distrib. Syst.*, 6(7):755 –760, Jul 1995.

[30] Z. Lu *et al*. Layered switching for networks on chip. In *DAC 2007*.

[31] S. Ma *et al*. DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip. In *ISCA 2011*.

[32] P. S. Magnusson *et al*. Simics: A full system simulation platform. *Computer*, 35:50–58, February 2002.

[33] R. Marculescu *et al*. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *IEEE Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 28:3–21, January 2009.

[34] M. M. K. Martin *et al*. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput. Archit. News*, 33:92–99, November 2005.

[35] S. Mukherjee *et al*. The Alpha 21364 network architecture. In *Hot Interconnects 2001*.

[36] R. Mullins *et al*. Low-latency virtual-channel routers for on-chip networks. In *ISCA 2004*.

[37] N. Neelakantam *et al*. FeS2: A full-system execution-driven simulator for x86. In *Poster presented at ASPLOS 2008*.

[38] C. Nicopoulos *et al*. ViChar: A dynamic virtual channel regulator for network-on-chip routers. In *MICRO 2006*.

[39] U. Y. Ogras *et al*. Key research problems in noc design: a holistic perspective. In *CODES+ISSS 2005*.

[40] L.-S. Peh and W. Dally. A delay model and speculative architecture for pipelined routers. In *HPCA 2001*.

[41] L. Schwiebert and D. N. Jayasimha. A necessary and sufficient condition for deadlock-free wormhole routing. *J. Parallel Distrib. Comput.*, 32:103–117, January 1996.

[42] K. Shin and S. Daniel. Analysis and implementation of hybrid switching. In *ISCA 1995*.

[43] A. Singh *et al*. GOAL: a load-balanced adaptive routing algorithm for torus networks. In *ISCA 2003*.

[44] C. B. Stunkel *et al*. The SP2 high-performance switch. *IBM Syst. J.*, 34:185–204, April 1995.

[45] Y. Tamir and G. Frazier. High-performance multiqueue buffers for VLSI communication switches. In *ISCA 1988*.

[46] A. Vaidya *et al*. Impact of virtual channels and adaptive routing on application performance. *IEEE Trans. Parallel Distrib. Syst.*, 12(2):223 –237, Feb 2001.

[47] F. Verbeek and J. Schmaltz. A comment on "a necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks". *IEEE Trans. Parallel Distrib. Syst.*, 22(10):1775 –1776, oct. 2011

[48] F. Verbeek and J. Schmaltz. On necessary and sufficient conditions for deadlock-free routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 22(12):2022 –2032, dec. 2011.

[49] D. Wentzlaff *et al*. On-chip interconnection architecture of the TILE processor. *Micro, IEEE*, 27(5):15 –31, Sept.-Oct. 2007.

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