Novel Flow Control for Fully Adaptive Routing in Cache-coherent NoCs

Sheng Ma, Zhiying Wang, *Member, IEEE,* Natalie Enright Jerger, *Senior Member, IEEE,* Li Shen and Nong Xiao, *Member, IEEE*

Abstract—Routing algorithms for cache-coherent NoCs only have limited VCs at their disposal, which poses challenges to the design of routing algorithms. Existing fully adaptive routing algorithms apply conservative VC re-allocation: only empty VCs can be re-allocated, which limits performance. We propose two novel flow control designs. First, whole packet forwarding (WPF) re-allocates a non-empty VC if the VC has enough free buffers for an entire packet. WPF does not induce deadlock if the routing algorithm is deadlock-free using conservative VC re-allocation. It is an important extension to several deadlock avoidance theories. Second, we extend Duato's theory [11] to apply aggressive VC re-allocation on escape VCs without deadlock. Finally, we propose a design which maintains maximal routing flexibility with low hardware cost. For synthetic traffic, our design performs averagely 88.9% better than existing fully adaptive routing. Our design is superior to partially adaptive and deterministic routing.

Index Terms—Networks-on-chip, Cache Coherence, Fully Adaptive Routing, Deadlock Avoidance Theory, VC Re-allocation.

1 Introduction

The performance of Networks-on-chip (NoCs) is sensitive to the routing algorithm, as it defines the network latency and saturation throughput [8], [29]. Many novel NoC routing algorithms have been proposed to deliver high performance [17], [20], [22], [25], [27], [37], [43]. In addition to performance considerations, the routing algorithm has correctness implications; any routing algorithm must be deadlock free, at both the network- and protocol-level. The guarantee of network-level deadlock freedom is generally based on deadlock avoidance theories. There are many theories for fully adaptive [11], [12], [16], [26], [36], [38], [39], [41] and partially adaptive routing design [4], [7], [17], [18]. Although most theories were originally proposed for off-chip networks, they are widely used in today's NoCs [17], [20], [22], [25], [27], [37], [43].

However, NoC packets are quite different than off-chip network packets. Abundant wires lead to wider flits which decreases the flit count per packet; short packets dominate NoC traffic. In contrast, the wires in off-chip networks are limited by pin counts; the flit width of a typical off-chip router is 32 bits (e.g., Alpha 21364 router [31]), while the typical NoC flit width is between 128 [19] and 256 bits [10]. With such wide flits, coherence messages with an address and control information but no data are encoded as single-flit packets. Fig. 1 shows that averagely 78.7% of packets are single-flit ones for PARSEC workloads [3]; other packets are 5 flits long with a 64B cache line.

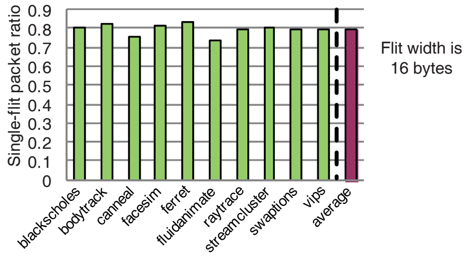


Fig. 1. Single-flit packet ratio for the PARSEC benchmarks.

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

Industrial products	
Alpha 21364 [31]	1 PN (7 VNs)
TILE64 [42]	5 PNs (1 VN/PN)
TRIPS [19]	2 PNs (4 VNs for OCN, 1 VN for OPN)
Cache coherence protocols in GEMS simulator [30]	
MESI directory	5 VNs
MOESI directory	4 VNs
MOESI token	4 VNs

Another noteworthy difference is that the buffers in NoCs are more precious than in off-chip networks due to the tight area and power budgets [15], [21], thus NoCs generally use flit-based wormhole flow control [6]. Although buffers are limited, several physical or virtual networks are needed for delivering different messages to avoid protocol-level deadlock. TABLE 1 lists the number of physical and virtual networks in some off-chip and on-chip networks. We also show the virtual network counts for some coherence protocols in GEMS simulator [30]. Typically, four or five virtual networks are needed to avoid protocol-level deadlock. Considering the expense of NoC buffers, each virtual network will have only a small number of VCs [5]. For example, TILE64 [42] and TRIPS [19] have one VC per virtual network. Thus, a NoC routing algorithm is generally running with limited VCs.

In VC-limited NoCs with short packets dominating traffic, the design of fully adaptive routing faces new challenges. In a wormhole network, fully adaptive routing algorithms based on existing theories require

• *Sheng Ma, Zhiying Wang, Li Shen and Nong Xiao are with the State Key Laboratory of High Performance Computing, National University of Defense Technology, China. Sheng Ma is also with the National Key Laboratory for Paralleling and Distributed Processing, NUDT.*
E-mail: {masheng, zywang, lishen, nongxiao}@nudt.edu.cn
This research was mostly done when Sheng Ma was a visiting international student at the University of Toronto.
• *Natalie Enright Jerger is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada. E-mail: enright@eecg.toronto.edu*

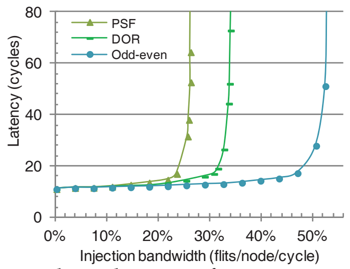


Fig. 2. Routing algorithms performance with bit reverse traffic. (PSF: fully adaptive routing, DOR: deterministic routing; Odd-even: partially adaptive routing.)

conservative VC re-allocation: only empty VCs can be re-allocated [11], [12], [16], [26], [36], [38], [39], [41]. This scheme prevents network-level deadlock, but it is very restrictive with many short packets [17]. Fig. 2 shows the performance of three algorithms[1]. Despite its flexibility and load-balancing capability, the fully adaptive algorithm is inferior to the deterministic and partially adaptive ones, since the latter algorithms apply aggressive VC re-allocation. It is imperative to enhance the design of fully adaptive routing.

This paper focuses on improving fully adaptive routing algorithms by designing novel flow control. We propose two mechanisms. First, whole packet forwarding (WPF) allows a non-empty VC which has enough buffers for an entire packet to be re-allocated. WPF can be viewed as applying packet-based flow control in a wormhole network. We prove that a fully adaptive routing algorithm using WPF is deadlock-free if this algorithm is deadlock-free with conservative VC re-allocation. WPF is an important extension to previous deadlock avoidance theories. Second, we extend Duato's theory [11] to apply aggressive VC re-allocation on escape VCs without deadlock. Part of this research was presented at HPCA 2012 [28].

The novel flow control enables the design of a fully adaptive routing algorithm with high VC utilization and maximal routing flexibility. Compared with existing fully adaptive routing algorithms, our design provides an average 88.9% gain for synthetic traffic. It is also superior to partially adaptive and deterministic algorithms. We make the following contributions:

- Proposes WPF to improve the performance of fully adaptive routing algorithms.
- Demonstrates WPF is an important extension to existing deadlock avoidance theories.
- Extends Duato's theory to apply aggressive VC re-allocation on escape VCs without deadlock.
- Proposes an efficient design which maintains high routing flexibility with low overhead.

We organize the paper as follows. Sec. 2 reviews related work. Sec. 3 gives the research motivation. Sec. 4 proposes the novel flow control and routing design. Sec. 5 analyzes the results. Sec. 6 concludes. The supplementary material further proves the logical equivalence of a routing algorithm with conservative VC re-allocation and this algorithm with WPF. It also provides more experimental analysis.

1. See Sec. 5 for experimental configuration and description.

2 Background

Here, we discuss related work in deadlock avoidance theories and designs for fully adaptive routing.

2.1 Deadlock Avoidance Theories

Since NoCs typically use wormhole flow control [6], [8], [29], [33], we focus on theories for wormhole networks. Dally and Seitz proposed a seminal deadlock avoidance theory [7] to design deterministic and partially adaptive routing. Duato introduced the routing sub-function, and gave an efficient design methodology [11], [12]. The message flow model [26] and channel waiting graph [36] were used to analyze deadlock. Taktak *et al.* [38] and Verbeek and Schmaltz [39] leveraged the decision procedure to establish deadlock-freedom. This methodology can be also applied to our designs. Verbeek and Schmaltz [40], [41] gave the first static necessary and sufficient condition for deadlock-free routing. These theories [11], [12], [16], [26], [36], [38], [39], [41] can design fully adaptive routing.

The theories of fully adaptive routing require only empty VCs to be re-allocated. This requirement guarantees that all blocked packets can reach VC heads to access 'deadlock-free' paths. Yet, it limits performance. Some deadlock-recovery designs [1] or theories [13] remove this limitation. They allow the formation of deadlock, and then invoke some recovery mechanisms. In contrast, we extend deadlock avoidance theories to prohibit the formation of deadlock.

There are several partially adaptive routing algorithms based on turn models [4], [17], [18]. They allow aggressive VC re-allocation: a VC can be re-allocated once the tail flit of last packet arrives [7], [9]. However, partially adaptive routing algorithms suffer from limited adaptivity: packets cannot use all minimal paths between the source and destination.

2.2 Fully Adaptive Routing Algorithms

Duato's theory [11] is widely used to design fully adaptive routing. This theory classifies VCs into escape and adaptive ones. Escape VCs apply more restrictive algorithms, typically dimension order routing (DOR), to form deadlock-free paths. An escape VC can only be used when the port adheres to DOR.

Many algorithms based on Duato's theory [20], [27], [31], [43] select the physical port first. Once selecting a port, packets can only request VCs of this chosen port. This requirement imposes a limitation: if a packet enters an escape VC, it can only use escape VCs until delivered. Otherwise, escape VCs may be involved in deadlock. In VC-limited NoCs, this limitation easily induces adaptivity loss. However, Duato's theory supports the design of algorithms which can use adaptive VCs after using an escape VC if packets can always request escape VCs [11]. Based on these observations, we propose a design which maintains high routing flexibility with low hardware cost.

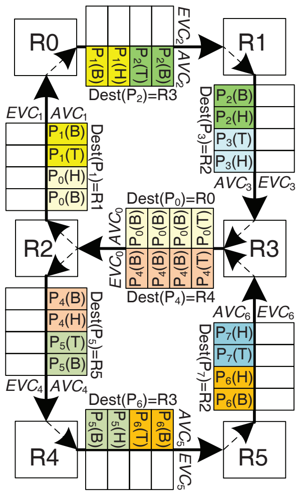


Fig. 3. Deadlock in a fully adaptive algorithm violating conservative VC re-allocation. (AVC: adaptive VC; EVC: escape VC; P_i(H), P_i(B) and P_i(T): the head, body and tail flit of Packet P_i; Dest(P_i): the destination of Packet P_i.)

3 Motivation

In this section, we analyze the requirements and limitations of fully adaptive routing algorithms.

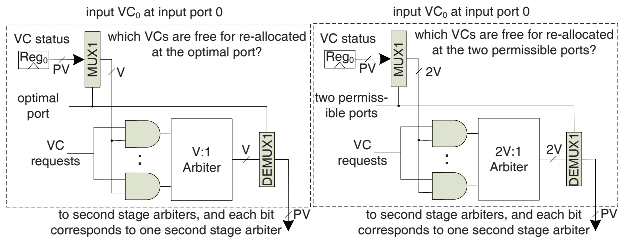
3.1 VC Re-allocation

Fully adaptive routing limits only empty VCs can be re-allocated. This requirement is reasonable since VCs may form cycles in fully adaptive routing. For example, Duato's theory arbitrarily uses adaptive VCs [11]; if multiple packets reside in one VC, a deadlock configuration appears, as shown in Fig. 3. Here each VN has two VCs: an adaptive VC (AVC) and an escape VC (EVC). Configuring more VCs cannot eliminate this deadlock scenario since cycles exist among AVCs.

Fig. 3 involves eight packets: P_0-P_7. P_0's head flit is behind P_1's tail flit 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 VC heads, they cannot move as both AVC_0 and EVC_0 are occupied. This deadlock scenario is due to that some head flits are not at VC heads, resulting in some packets being unable to access the 'deadlock-free' path. Also, the tail flits of these packets reside in other VCs, prohibiting other packets from reaching VC heads. These following packets then cyclically block aforementioned packets. For example, P_0's tail flit resides in AVC_0, blocking P_3 from using this VC, which cyclically blocks P_0. If the packet length is greater than the VC depth, putting multiple packets in one VC may induce deadlock. Yet, we notice that moving entire short packets into non-empty VCs will not prevent following packets from reaching VC heads. This is an opportunity for optimization, especially with many short NoC packets.

3.2 Routing Flexibility

This section discusses routing flexibility. Many fully adaptive algorithms based on Duato's theory consist of the routing function and selection strategy [20], [27], [31], [43]; they are called *port-selection-first* since



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

Fig. 4. First stage arbiter. (Each VN has V VCs and P ports.)

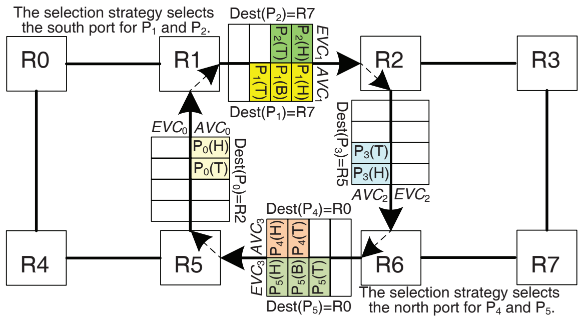


Fig. 5. A deadlock configuration if packets in EVCs can request AVCs in port-selection-first algorithms.

once the selection strategy chooses a port, the packet only requests VCs of this particular port. Assuming a separable VC allocator with two stages of arbiters [2], [32], [34], a port-selection-first algorithm only needs V:1 arbiters in the first stage as shown in Fig. 4a.

A limitation of these algorithms is that once a packet enters an escape VC, it must continue to use escape VCs; the packet loses adaptivity. Violating this limitation induces deadlock as shown in Fig. 5. Assuming P_1 and P_2 select the south port, they can only request AVC_2 since EVC_2 can only be requested when the port adheres to DOR. Similarly, P_4 and P_5 select the north port; they only request AVC_0. No packet can move. Thus, requiring that if a packet enters an escape VC, it can only use escape VCs subsequently is necessary for port-selection-first algorithms.

Duato's theory supports algorithms to use adaptive VCs after using escape VCs, if they guarantee packets are always able to request escape VCs. To achieve this target, a packet could request all permissible output VCs, since at least one port must adhere to DOR and the packet can use the escape VC of this port [11]. However, this naive design has additional overhead. As shown in Fig. 4b, the VC allocator uses $2V$:1 arbiters to cover the at most two permissible ports for minimal routing. We propose a low-cost design to maintain high routing flexibility.

4 Flow Control and Routing Designs

First, we present whole packet forwarding. Then, we extend Duato's theory to apply aggressive VC re-allocation on escape VCs. Next, we give a low-cost design to maintain routing flexibility. Finally, we describe the implementation and overhead.

4.1 Whole Packet Forwarding

We propose whole packet forwarding (WPF). Suppose a packet P_k with $length(P_k)$ resides in VC_i, and VC_j

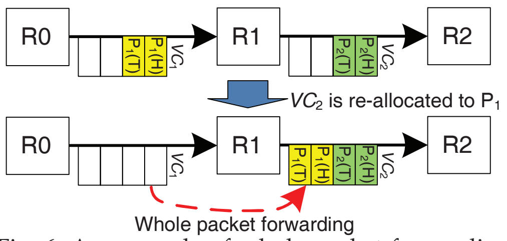


Fig. 6. An example of whole packet forwarding.

is downstream of VC_i. Assume the routing algorithm allows 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 last allocated packet is sent out, i.e., it is empty [9]. For WPF, VC_j can be re-allocated if it has received the tail flit of the last allocated packet, and its free buffer count is no less than $length(P_k)$.

Fig. 6 shows a WPF example. The algorithm allows P_1 to use VC_2. VC_2 received P_2's tail flit and has two free slots; this space is enough for P_1's two flits. Thus, WPF re-allocates VC_2 to P_1. WPF works similarly to packet-based flow control such as store-and-forward (SAF) [14] and virtual cut-through (VCT) [23]. VCT enables the design of high frequency routers [35]. Yet, our design uses wormhole with empty VCs, which does not require the empty VC to be large enough for entire packets; this reduces the buffering requirement compared with VCT, and provides more flexibility.

Our contention is that if the routing algorithm with conservative VC re-allocation is deadlock-free, then applying WPF will not induce deadlock. Intuitively, this contention is true, since short packets will be either at VC heads, or behind some packets whose head flits are at VC heads. Thus, some packets can access the 'deadlock-free' path. Yet, the 'deadlock-free' path is coupled to special theories. We provide a general proof. We label the algorithm with conservative VC re-allocation as Alg; $Alg + WPF$ is Alg with WPF to forward entire packets into non-empty VCs. The proof needs a simple assumption about the router.

***Assumption* 1.** Assume a packet can use multiple VCs. If any permissible VC is *eventually* available, the packet must have some possibility to request this VC.

This assumption provides *weak fairness* among permissible VCs. It can be fulfilled in the following manner. The router continually monitors the status of all downstream VCs. If one permissible VC is available, the packet requests it immediately. If the algorithm first selects a port and limits the packet to request VCs of the selected port, the selection strategy should guarantee that any permissible port has some possibility to be selected. As shown in Sec. 4.4, common NoC routers adhere to Assumption 1.

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

Description: By contradiction. We prove that if Alg+WPF has a deadlock configuration, then Alg has a deadlock configuration as well. Using $Config_0$ in Fig. 7 as an example, we remove packets whose head flits are not at VC heads, and get a new configuration

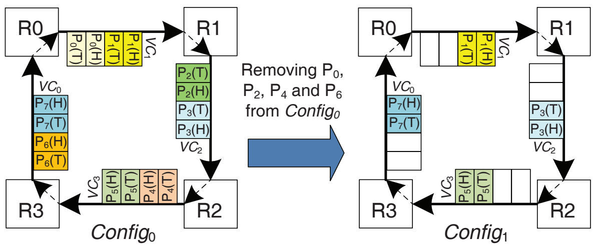


Fig. 7. The construction of a new configuration.

$Config_1$. We prove that Alg can achieve $Config_1$, and $Config_1$ is a deadlock configuration. However, Alg is 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 there is a deadlock configuration for Alg in 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's head flit is not at the VC head, then this VC was allocated to P_i using WPF; all flits of P_i must reside in one VC. We remove P_i and label these removed packets as P_{subset_0}. We label the new configuration as $Config_1$, and the set of packets in $Config_1$ as P_{subset_1}.

Step 2: We prove that when the network is routed by Alg, all packets in P_{subset_1} can move into their current VCs in $Config_1$. For each packet P_j in P_{subset_1}, we consider each hop hop_k of P_j when it is routed by Alg+WPF. We assume that P_j's head flit moves into VC_k during hop_k. There are two situations.

2.1) VC_k is empty when P_j reaches it; VC_k is allocated to P_j with conservative VC re-allocation. Thus, when routed by Alg, P_j can use VC_k.

2.2) VC_k is not empty when P_j reaches it; VC_k is allocated to P_j with WPF. The algorithm allows P_j to use VC_k. Yet, when routed by Alg, P_j cannot move into VC_k until it is empty. Since Alg is deadlock-free, the packet currently in VC_k must move out in a limited time. Then VC_k is available for conservative VC re-allocation. Based on Assumption 1, P_j has some possibility to request VC_k. Then, P_j can move into VC_k. Thus, when routed by Alg, P_j can use VC_k.

Considering 2.1) and 2.2) together, and taking into account that the head flits of all packets in $Config_1$ are at VC heads, if a VC is used by P_j during any hop when routed by Alg+WPF, this VC can be also used by P_j when 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 set P_{subset_0}, all flits of P_i reside in one VC but P_i's head is not at the VC head. Thus, removing P_i does not create empty VCs. Alg uses conservative VC re-allocation which only re-allocates empty VCs; all packets in P_{subset_1} still wait for VCs held by other packets in P_{subset_1}. $Config_1$ is a deadlock configuration for Alg. But Alg is deadlock-free, so there is no deadlock configuration. Thus, Alg+WPF is deadlock-free as well. □

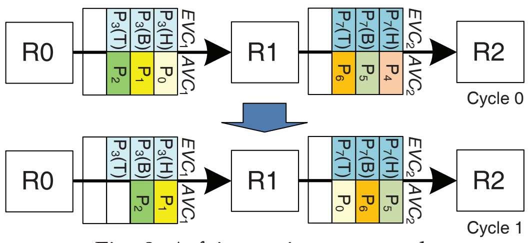


Fig. 8. A fairness issue example.

This proof only needs a simple assumption. WPF can be used with many fully adaptive routing algorithms if they are deadlock-free with conservative VC re-allocation. It is an important extension to existing theories [11], [12], [16], [26], [36], [38], [39], [41]. In the supplementary material, we further prove that if Alg+WPF is deadlock-free, Alg is also deadlock-free.

4.2 Aggressive VC Re-allocation for Escape VCs

We apply WPF to fully adaptive algorithms designed based on Duato's theory [11]. Yet, using WPF directly may bring fairness issues for long packets, as shown in Fig. 8. Both three-flit packet P_3 and single-flit packet P_0 are waiting to move forward. The free buffers in EVC_2 and AVC_2 allow P_0 to move. P_3 must wait for EVC_2 or AVC_2 to have 3 free slots, or for them to be empty. If P_4 moves forward in the next cycle, a similar situation happens again: the free buffers in EVC_2 or AVC_2 only allow P_1 to move. Under the extreme case, P_3 will be permanently waiting if single-flit packets are continuously injected into AVC_1[2].

A design to address this fairness issue is to deploy a time counter and a priority allocator. Once the counter crosses the threshold value for the blocked packet P_3, a high priority is assigned to P_3's VC allocation so as to prevent single-flit packets from occupying the buffers in EVC_2 and AVC_2. This design involves additional costs including the time counter and priority allocator. Also, an appropriate threshold value needs to be configured to provide the desired fairness.

Instead of using this complex design, we leverage a more elegant observation. We notice the escape VCs in Duato's theory can apply aggressive VC re-allocation without deadlock. With aggressive re-allocation, P_3 and P_0 can equally use EVC_2; P_3 cannot be blocked forever. Under the worst case with a stream of continuous short packets using AVC_2, P_3 may cannot acquire AVC_2; this may cause long packets to have less adaptivity than short ones. Yet, two factors mitigate the possible negative effect. First, network streams generally consist of both short and long packets. Second, our routing design in Sec. 4.3 can request adaptive VCs after using escape VCs, thus this limitation only causes minor adaptivity loss. Also, our design has less overhead than the complex design, and aggressive VC re-allocation further improves the VC utilization and performance. This design is coupled to Duato's theory; we reiterate some definitions from Duato's paper [11], [12] to make this paper self-contained.

2. This extreme case never appears in our simulation as short and long packets are randomly injected.

***Definition* 1. [Network path]** A network path consists of a set of VCs, VC_0, VC_1, ... , VC_i. Along this path, packets can be sent from VC_0 to VC_i.

***Definition* 2. [Connected routing function]** A routing function is connected iff it can always establish a path for every packet from its current VC to its destination.

***Definition* 3. [Routing sub-function]** A routing sub-function is based on a routing function. The input VCs of the routing sub-function are the same as the routing function. The output VCs supplied by the routing sub-function are a subset of the routing function.

***Definition* 4. [Direct dependency]** If a packet can use VC_j immediately after using VC_i, there is a direct dependency from VC_i to VC_j.

***Definition* 5. [Indirect dependency]** There is an indirect dependency between two escape VCs, EVC_i and EVC_j, iff there is a network path consisting of EVC_i, AVC_0, ... , $AVC_{k(k\geqslant 0)}$, EVC_j. In other words, EVC_i and EVC_j are the first and last VCs in this path, and all intermediate VCs are adaptive VCs.

***Definition* 6. [Extended VC dependency graph]** The extended VC dependency graph D_E is defined for escape VCs. The vertices of D_E are escape VCs. The arcs of D_E are the pairs of escape VCs (EVC_i, EVC_j) such that there is either a direct dependency or an indirect dependency from EVC_i to EVC_j.

Duato's necessary and sufficient theory [12] defines the direct cross and indirect cross dependency. Our research is mainly based on Duato's necessary theory [11], thus we omit these two types of dependency.

Duato's Necessary Theory. An adaptive routing function is deadlock-free if there exists a routing sub-function that is connected and has no cycles in its extended VC dependency graph [11].

The routing sub-function is defined on escape VCs. As the extended VC dependency graph is acyclic, we can assign an order among escape VCs so that if there is a direct or an indirect dependency from EVC_i to EVC_j, then $EVC_i{>}EVC_j$ [11]. Moreover, since the routing sub-function is connected, any packet can move to its destination by using escape VCs [11].

Theorem 1 applies WPF to both adaptive and escape VCs. This section further applies aggressive VC re-allocation to escape VCs. We label the *deadlock-free* algorithm based on Duato's theory with conservative VC re-allocation as Alg; Alg+WPF+Agg is Alg with WPF for adaptive VCs and aggressive VC re-allocation for escape VCs. We prove Theorem 2.

***Theorem* 2**: If Alg is a deadlock-free routing algorithm designed based on Duato's theory [11], then Alg+WPF+Agg is also deadlock-free.

Description: The structure of our proof is similar to the proof of Theorem 2 in Duato's paper [11]. The difference is that we prove that even with WPF on adaptive VCs and aggressive VC re-allocation on escape VCs, there is still a movable packet in a hypothetical deadlock configuration. Two key points are: the routing sub-function on escape VCs is connected, and there is an order among escape VCs.

Proof: Suppose Alg+WPF+Agg has a deadlock configuration ($Config_0$), in which no packet head has already reached the destination. There are two cases.

1: All escape VCs are empty; $Config_0$ only consists of adaptive VCs. Let AVC_i be an adaptive VC in $Config_0$, and P_i is the packet whose flit is at AVC_i's head. There are two situations.

1.1) The flit at AVC_i's head is P_i's head flit. Since the routing sub-function on escape VCs is connected, there is an escape VC EVC_i that P_i can use. As all escape VCs are empty, P_i can move into EVC_i.

1.2) The flit at AVC_i's head is not P_i's head flit; P_i spans multiple VCs. Since WPF moves entire packets, P_i is not forwarded by WPF. P_i's head flit must reside at another adaptive VC's head. Similar to 1.1), P_i can move into an empty escape VC as well.

2: $Config_0$ involves escape VCs. There is an order among escape VCs; let EVC_i be the non-empty escape VC in $Config_0$ such that all escape VCs with an order less than EVC_i are empty. Let P_i be the packet whose flit is at EVC_i's head. There are two situations.

2.1) The flit at EVC_i's head is P_i's head flit. Since the routing sub-function is connected, there is an escape VC EVC_j that P_i can use. It implies there is a direct dependency from EVC_i to EVC_j; thus, the order of EVC_j is less than EVC_i. EVC_j is empty; P_i can move into EVC_j.

2.2) The flit at EVC_i's head is not P_i's head flit. Aggressive VC re-allocation may make P_i's head flit reside in another escape VC (EVC_j). In such a case, there is a direct or an indirect dependency from EVC_i to EVC_j. The order of EVC_j is less than EVC_i. EVC_j should be empty; P_i's head flit cannot reside in EVC_j.

As P_i spans multiple VCs, it was not forwarded by WPF. P_i's head flit must be at the head of an adaptive VC. Since the routing sub-function is connected, there is an escape VC EVC_k that P_i can use, which implies there is an indirect dependency from EVC_i to EVC_k. Thus, EVC_k is empty. P_i can move into EVC_k.

In all cases, a packet can move. There is no deadlock configuration for $Alg + WPF + Agg$. □

4.3 Maintain Routing Flexibility

This section proposes a low-cost design to maintain high routing flexibility. This design is based on Duato's theory [11]. The design should allow the use of adaptive VCs after using escape VCs. Otherwise, once a packet enters escape VCs, it will lose adaptivity. The design must guarantee that a packet can request an escape VC at any time [11]. Once this condition is satisfied, packets can always find a path which is not involved into cyclic dependencies, since the extended dependency graph of escape VCs is acyclic [11].

In port-selection-first algorithms, the only time a packet cannot use escape VCs is when the selected port violates DOR. We make a simple modification by violating the selection in this case; the packet requests the escape VC of the non-selected permissible port in addition to adaptive VCs of the selected one. Using P_1 in Fig. 5 as an example, if the south port is selected, our design allows P_1 to request the escape VC of the east port. If there is only one permissible port, this port must adhere to DOR, and the packet can request its escape VC. This design guarantees that a packet can always request an escape VC. It allows a packet to move back into adaptive VCs after using an escape VC. Also, 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.

4.4 Router Microarchitecture

The pipeline of a canonical NoC router consists of routing computation (RC), VC allocation (VA), switch allocation (SA) and switch traversal (ST) [9], [14], [32], [34]. We apply several optimizations. The speculative allocation parallelizes SA with VA at low loads [34]. Look-ahead routing calculates at most two permissible ports one hop ahead and the selection strategy chooses an optimal one [20], [24], [27]. The router delay is two cycles plus one cycle for link traversal.

This baseline router adheres to Assumption 1 in Sec. 4.1. First, a common selection strategy guarantees that any permissible port can be selected. For example, Sec. 5 uses a strategy which prefers the port with more free buffers. This strategy will not prohibit any permissible port from being selected since once the permissible ports have equal free buffers, it randomly selects one from them. Second, the router continually monitors the status of all downstream VCs using credits [9]. If one permissible VC of the selected port is eventually available, the packet can find out this situation, and requests this VC immediately. Therefore, a NoC router generally provides *weak fairness* among permissible VCs, even when selecting the port first.

Our design only requires simple modifications to the VC allocator. Any type of allocator can be used; we assume a low-cost separable allocator which consists of two stages of arbiters [2], [32], [34]. We modify the first stage arbiters to apply WPF on adaptive VCs and aggressive VC re-allocation on escape VCs. WPF requires calculating whether the free buffers are enough for an entire packet, which has some overhead. Yet, considering cache coherence packets exhibit a bimodal distribution and the majority is single-flit packets, we apply WPF only on single-flit packets.

Fig. 9 shows the proposed VC allocator. Reg_0 and Reg_1 record whether a downstream VC can be re-allocated. If a downstream VC is an adaptive VC, the corresponding bit in Reg_0 records whether it can be re-allocated with conservative re-allocation. If a downstream VC is an escape VC, the corresponding bit in Reg_0 records whether it can be re-allocated with aggressive re-allocation. The criterion is that the VC

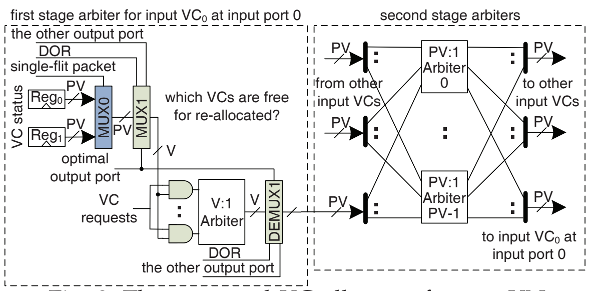


Fig. 9. The proposed VC allocator for one VN.

holds the tail flit of its last allocated packet and still has free buffers. Reg_1 records whether a VC can be re-allocated with WPF for single-flit packets. Its criterion is the same as applying aggressive VC re-allocation on single-flit packets. Thus, we use Reg_0 or Reg_1 based on the incoming packet type. If it is a single-flit packet, we apply WPF, sending Reg_1 to MUX1. Otherwise, Reg_0 is sent to MUX1. Updates to Reg_0 and Reg_1 are off the critical path as the router monitors the status of downstream VCs. The increased delay is an additional 2-input multiplexer: MUX0.

To maintain routing flexibility, we modify MUX1 and DEMUX1, as shown in Fig. 9. MUX1 needs two additional signals: *DOR* and *the other output port*. The *DOR* signal indicates if the selected port obeys DOR. *The other output port* signal records the non-selected permissible port. The routing logic produces these signals. If *DOR* is '0', the selected port violates DOR. Then, the status of the escape VC for *the other output port* rather than the selected one is sent to the V:1 arbiter. This is accomplished with a 2-input multiplexer whose control signal is *DOR*. DEMUX1 also needs these signals. If *DOR* 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*. This is accomplished with a 2-input demultiplexer. The increased delay is an additional 2-input multiplexer and demultiplexer.

We implement the three VC allocators (Figs 4 and 9) in RTL Verilog for an open-source NoC router [2] and synthesize in Design Compiler with a TSMC 65nm standard cell library. The designs operate at 500 MHz under normal conditions (1.2V, 25°C). We use round-robin arbiters [9]. This router has 5 ports (P=5) and 4 VNs; each VN has 2 VCs (V=2). TABLE 2 gives 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 the area by 0.2%. An allocator's power consumption is largely decided by the arbiter size [2], [43]; given the small arbiters in our design, there should be negligible power overhead compared with the port-selection-first design.

5 Evaluation

We modify Booksim [9] for evaluation. Fully adaptive routing includes the high flexibility design (FULLY)

TABLE 2. The critical path delay and area results.

Design	**Delay** (ns)	**Area** (μm^2)
Port-selection-first (Fig. 4a)	1.78	49437.4
Naive design (Fig. 4b)	1.92	56045.4
Proposed design (Fig. 9)	1.79	49512.6

in Sec. 4.3 and the port-selection-first design (PSF). FULLY and PSF are evaluated with conservative re-allocation (named as FULLY and PSF), and with WPF on adaptive VCs and aggressive re-allocation on escape VCs (named as FULLY+WA and PSF+WA). The deterministic routing is DOR. West-first, negative-first and odd-even represent partially adaptive routing. We use a local selection strategy for all adaptive routing; when there are two permissible ports, it chooses the one with more free buffers. If the ports have equal free buffers, it randomly chooses one of them.

Our evaluation for synthetic traffic uses one VN since each VN is independent. We use 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 single-flit packet ratio is 80%. The simulator is warmed up for 10,000 cycles and then the performance is measured over another 100,000 cycles. We also perform sensitivity studies on several configuration parameters and evaluate full-system performance using PARSEC applications [3]. These results are shown in the supplementary material.

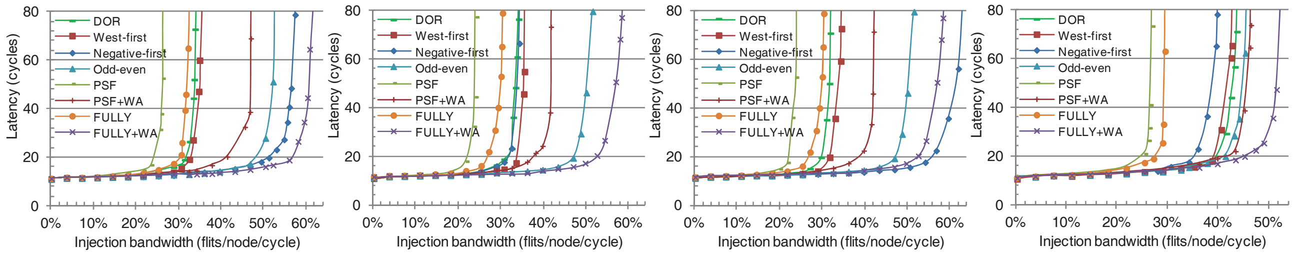
5.1 Performance of Synthetic Workloads

Fig. 10 gives the performance with four synthetic traffic patterns [9]. Across the four patterns, PSF and FULLY perform worst. Although PSF and FULLY offer adaptiveness for all traffic, conservative VC re-allocation significantly limits their performance. In contrast, DOR and partially adaptive routing use aggressive VC re-allocation. PSF is inferior to FULLY. PSF 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 until delivered.

For bit reverse, a node with bit address $\{s_3,s_2,s_1,s_0\}$ sends traffic to node $\{s_0,s_1,s_2,s_3\}$. 62.5% of the traffic is between the north-east and south-west quadrants; negative-first offers adaptiveness for this traffic. 37.5% of the traffic is eastbound; west-first offers adaptiveness for this traffic, and performs worse than negative-first. Although WPF and aggressive VC re-allocation improve the VC utilization for PSF+WA, PSF+WA is inferior to odd-even and negative-first. PSF+WA is limited by poor flexibility. FULLY+WA provides high VC utilization and routing flexibility leading to the highest saturation throughput[3].

Transpose-1 sends message from node (i, j) to node $(3 - j, 3 - i)$. Negative-first deteriorates to DOR for this pattern. West-first still offers adaptiveness for 37.5% of the traffic, thus it is superior to negative-first. Odd-even offers greater adaptiveness than the

3. 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.

Fig. 10. Routing algorithm performance for the baseline configuration.

TABLE 3. 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+WA	31.3%
Negative-first	26.6%		

other two partially adaptive algorithms and achieves higher performance. FULLY+WA offers adaptiveness for all traffic, performing 15.7% better than odd-even.

Transpose-2 is favorable to negative-first; node (i, j) communicates with node (j, i). Negative-first offers adaptiveness for all traffic and performs best. Although FULLY+WA offers adaptiveness for all traffic, it is limited by the restriction on escape VCs: only if the port adheres to DOR, can the escape VC be used. The performance of FULLY+WA and odd-even with transpose-2 is similar to their performance with transpose-1 since the two patterns are symmetric and these designs 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 uniform random traffic. This pattern mimics memory controllers receiving a disproportionate amount of traffic. FULLY+WA and odd-even are worse than negative-first and west-first. Due to the limited adaptiveness offered by odd-even, it is inferior to FULLY+WA. DOR outperforms negative-first and west-first, since DOR more evenly distributes uniform traffic which is the background in this pattern.

TABLE 3 gives average throughput gains of FULLY+WA over other designs. The 88.9% gap between FULLY+WA and FULLY reflects the effect of novel flow control. The gap between FULLY+WA and PSF+WA represents of the effect of routing flexibility; high flexibility brings a gain of 31.3%. The supplementary material provides further insights about performance trends by analyzing the buffer utilization.

5.2 Detailed Analysis of Flow Control

Here, we analyze aggressive VC re-allocation and WPF by measuring the buffer utilization for network VCs. Since escape VCs can only be used when the port adheres to DOR, not all escape VCs are allowable for a particular traffic pattern. To clearly understand the system bottleneck, we leverage a recursive algorithm to calculate the allowable escape VCs (EVCs) for all evaluated patterns. As shown in TABLE 4, some traffic patterns, such as bit reverse, cannot use all EVCs, while other patterns including bit complement and uniform random can use all EVCs. The recursive algorithm is shown in the supplementary material.

Fig. 11 gives the saturation throughput and buffer utilization for fully adaptive designs. 'PSF+WPF' and 'FULLY+WPF' apply WPF on both adaptive and escape VCs. The difference between 'PSF+WA' and 'PSF+WPF' (or 'FULLY+WA' and 'FULLY+WPF') is that 'PSF+WA' (or 'FULLY+WA') applies aggressive VC reallocation on escape VCs. The adaptive VC utilization and escape VC utilization are shown in 'Average AVC utilization' and 'Average EVC utilization' bars. The 'Average allowable EVC utilization' bar gives the average utilization of allowable EVCs.

There are several insights from Fig. 11. First, the performance gain of FULLY+WPF over FULLY (or PSF+WPF over PSF) is due to the improvement of buffer utilization for both adaptive and escape VCs. The gain of FULLY+WA over FULLY+WPF (or PSF+WA over PSF+WPF) is due to the improvement of buffer utilization for escape VCs; aggressive VC reallocation more efficiently utilizes escape VCs than WPF. For example, in bit reverse, the EVC utilization of FULLY+WA and FULLY+WPF is 0.377 and 0.247.

Second, the efficiency of aggressive VC re-allocation depends on routing flexibility. FULLY can use adaptive VCs after using escape VCs; aggressive VC reallocation offers more gains for FULLY+WA than PSF+WA. In bit reverse, FULLY+WA performs 26.6% better than FULLY+WPF; the 52.9% EVC utilization improvement of FULLY+WA over FULLY+WPF brings this gain. In contrast, PSF+WA only achieves 16.3% higher EVC utilization than PSF+WPF, which brings a 9.7% performance gain for PSF+WA.

Third, PSF+WPF requests an escape VC only when the selected port adheres to DOR. Meanwhile, it always applies WPF on adaptive VCs. These two factors induce higher pressure on adaptive VCs than escape ones for PSF+WPF. For example, in transpose-1, the AVC utilization of PSF+WPF is 0.228, while the EVC utilization is 0.158 and the allowable EVC utilization is 0.210. FULLY can always request an escape VC. Its allowable EVC utilization is higher than the AVC utilization for most patterns; FULLY puts more pressure on escape VCs than adaptive ones.

Fourth, the aggressive VC re-allocation in FULLY+WA further increases the pressure on escape VCs;

TABLE 4. Allowable EVC count for synthetic traffic patterns (Total network EVC count = 48).

	bit reverse	transpose-1	transpose-2	hotspot	uniform	shuffle	bit complement	bit rotation
Allowable EVC	36	36	36	48	48	36	48	40

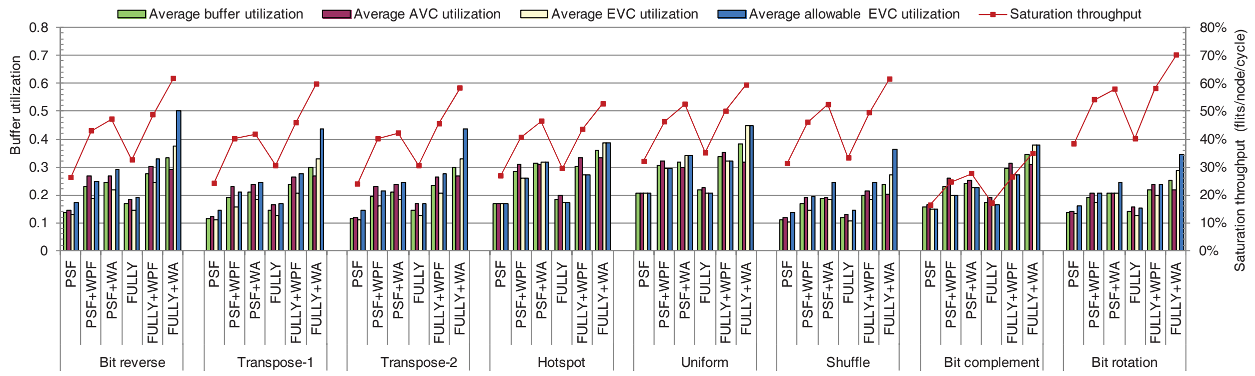


Fig. 11. The buffer utilization and saturation throughput of fully adaptive routing algorithms.

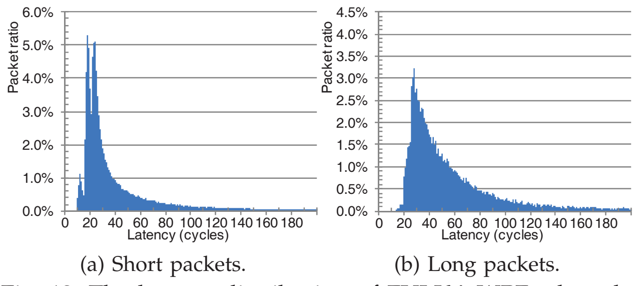
the allowable EVC utilization of most patterns is more than 60% higher than the AVC utilization. FULLY+WA is limited by allowable escape VCs, and mechanisms to improve the capacity of escape VCs, such as dynamically sharing buffers between escape and adaptive VCs will be helpful to FULLY+WA.

5.3 The Effect of Flow Control on Fairness

As shown in Sec. 4.2, aggressive VC re-allocation on escape VCs can address the fairness issue of WPF on long packets. To illustrate this point, Figs 12 and 13 show the latency distribution of short and long packets for FULLY+WPF and FULLY+WA with bit complement traffic; the overall network average latencies are both 52 cycles. Other patterns have similar trends. The short packet latency of FULLY+WPF has three distinct peaks at 11, 17 and 23 cycles, corresponding to packets which take 3, 5 and 7 hops without any queuing delay, respectively. Since each flit of a long packet may face different switch allocation congestion, the long packet latency only exhibits one peak at 27 cycles, corresponding to packets with 7 hops.

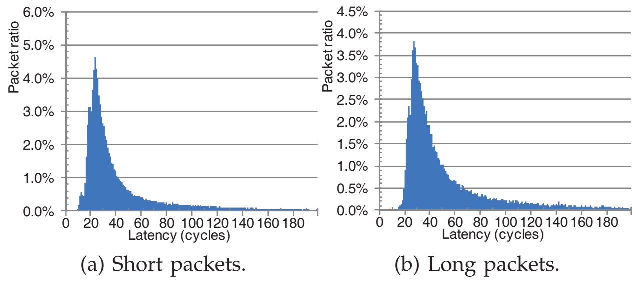
The short and long packet latencies of FULLY+WA both have one peak, at 23 and 27 cycles respectively. To illustrate the difference between FULLY+WA and FULLY+WPF, we subtract Fig. 12 from Fig. 13, as shown in Fig. 14. The ratio of short packets with a latency between 9 and 23 cycles for FULLY+WA is 13.2% less than that for FULLY+WPF. On the other hand, FULLY+WA has 8.8% more long packets with a latency between 19 and 42 cycles than FULLY+WPF. Compared with FULLY+WPF, FULLY+WA accelerates long packets, while sacrificing short packets.

The effect on fairness can be also observed from the average latency of packets injected from different sources, as shown in Fig. 15. In FULLY+WPF, middle nodes have very high latency as their injected long packets are always inferior to short packets from edge nodes. FULLY+WA reduces the peak latency for these middle nodes from 134 cycles to 104 cycles, and increases the latency for the edge nodes, achieving more even latency distribution throughout the network.



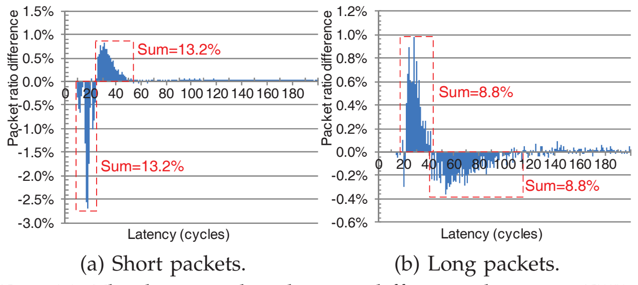
(a) Short packets. (b) Long packets.

Fig. 12. The latency distribution of FULLY+WPF when the average latency is 52 cycles for bit complement traffic.



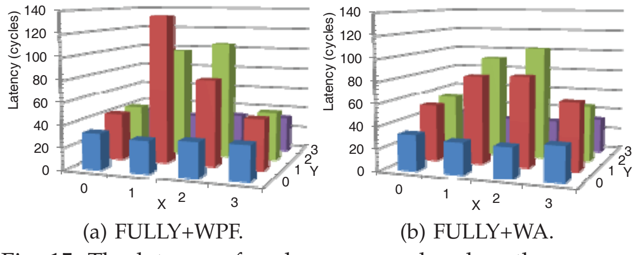
(a) Short packets. (b) Long packets.

Fig. 13. The latency distribution of FULLY+WA when the average latency is 52 cycles for bit complement traffic.



(a) Short packets. (b) Long packets.

Fig. 14. The latency distribution difference between FULLY+WPF and FULLY+WA for bit complement traffic.



(a) FULLY+WPF. (b) FULLY+WA.

Fig. 15. The latency of each source node when the average latency is 52 cycles for bit complement traffic.

The latency shown in Figs 12-15 are snapshots when the network average latencies are 52 cycles. With decreasing injection rate, the latency differences between FULLY+WA and FULLY+WPF become insignificant as the network contention becomes less likely. Conversely, the differences become more pronounced as the network approaches saturation. For example, when the average network latencies of FULLY+WPF and FULLY+WA are both 75 cycles, FULLY+WA has 10.3% more long packets with a latency between 21 and 48 cycles than FULLY+WPF, while FULLY+WPF has 15.7% more short packets with a latency between 11 and 25 cycles than FULLY+WA.

6 Conclusion

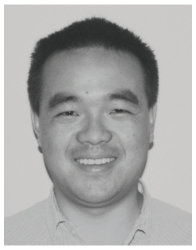
An abundance of short packets and a limited VC budget increase the importance of flow control on the performance of cache-coherent NoCs. We propose two novel flow control designs for fully adaptive routing in wormhole networks. WPF is an important extension to several deadlock avoidance theories. It re-allocates a non-empty VC if the VC has enough buffers for an entire packet. Then, we extend Duato's theory to apply aggressive VC re-allocation on escape VCs. Based on the proposed flow control, we present a low-cost design to maintain high routing flexibility. Our design outperforms several fully adaptive, partially adaptive and deterministic routing algorithms.

Acknowledgments

We thank the anonymous reviewers for their thorough and constructive suggestions. This work is supported by 863 Program of China (2012AA010905), NSFC (61272144, 61070037, 61272143, 61025009), the University of Toronto, NSERC of Canada, Research Project of NUDT (JC13-06-02).

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. Chiu. The odd-even turn model for adaptive routing. *IEEE Trans. Parallel Distrib. Syst.*, 11(7):729 –738, Jul. 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. The torus routing chip. *Distributed Computing*, 1:187–196, 1986.
[7] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Computers*, C-36(5):547 –553, May 1987.
[8] W. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *DAC 2001*.
[9] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
[10] R. Das et al. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *HPCA 2009*.
[11] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 4(12):1320 –1331, Dec. 1993.
[12] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 6(10):1055 –1067, Oct. 1995.
[13] 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.
[14] N. Enright Jerger and L. Peh. *On-Chip Networks*. Morgan and Claypool Publishers, 2009.
[15] C. Fallin, C. Craik, and O. Mutlu. CHIPPER: A low-complexity bufferless deflection router. In *HPCA 2011*.
[16] E. Fleury and P. Fraigniaud. A general theory for deadlock avoidance in wormhole-routed networks. *IEEE Trans. Parallel Distrib. Syst.*, 9:626–638, Jul. 1998.
[17] B. Fu, Y. Han, J. Ma, H. Li, and X. Li. An abacus turn model for time/space-efficient reconfigurable routing. In *ISCA 2011*.
[18] C. Glass and L. Ni. The turn model for adaptive routing. In *ISCA 1992*.
[19] P. Gratz et al. On-chip interconnection networks of the TRIPS chip. *Micro, IEEE*, 27(5):41 –50, Sept.-Oct. 2007.
[20] P. Gratz, B. Grot, and S. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *HPCA 2008*.
[21] M. Hayenga, N. Enright Jerger, and M. Lipasti. SCARAB: A single cycle adaptive routing and bufferless network. In *MICRO 2009*.
[22] J. Hu and R. Marculescu. DyAD - smart routing for networks-on-chip. In *DAC 2004*.
[23] P. Kermani et al. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 1979.
[24] J. Kim et al. A low latency router supporting adaptivity for on-chip interconnects. In *DAC 2005*.
[25] M. Li, Q. Zeng, and W. Jone. DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *DAC 2006*.
[26] X. Lin, P. McKinley, and L. Ni. The message flow model for routing in wormhole-routed networks. *IEEE Trans. Parallel Distrib. Syst.*, 6(7):755 –760, Jul. 1995.
[27] S. Ma, N. Enright Jerger, and Z. Wang. DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip. In *ISCA 2011*.
[28] S. Ma, N. Enright Jerger, and Z. Wang. Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip. In *HPCA*, 2012.
[29] 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, Jan. 2009.
[30] M. Martin et al. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *SIGARCH Comput. Archit. News*, 33:92–99, Nov. 2005.
[31] S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb. The Alpha 21364 network architecture. In *Hot Interconnects 2001*.
[32] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *ISCA 2004*.
[33] U. Y. Ogras, J. Hu, and R. Marculescu. Key research problems in NoC design: a holistic perspective. In *CODES+ISSS 2005*.
[34] L. Peh and W. Dally. A delay model and speculative architecture for pipelined routers. In *HPCA 2001*.
[35] A. Roca, J. Flich, F. Silla, and J. Duato. VCTlite: Towards an efficient implementation of virtual cut-through switching in on-chip networks. In *HiPC 2010*.
[36] L. Schwiebert and D. N. Jayasimha. A necessary and sufficient condition for deadlock-free wormhole routing. *J. Parallel Distrib. Comput.*, 32:103–117, Jan. 1996.
[37] A. Singh et al. GOAL: a load-balanced adaptive routing algorithm for torus networks. In *ISCA 2003*.
[38] S. Taktak, E. Encrenaz, and J. Desbarbieux. A polynomial algorithm to prove deadlock-freeness of wormhole networks. In *PDP 2010*.
[39] F. Verbeek and J. Schmaltz. Automatic verification for deadlock in networks-on-chips with adaptive routing and wormhole switching. In *NoCS 2011*.
[40] 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.
[41] 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.
[42] D. Wentzlaff et al. On-chip interconnection architecture of the Tile processor. *Micro, IEEE*, 27(5):15 –31, Sept.-Oct. 2007.
[43] Y. Xu et al. Simple virtual channel allocation for high throughput and high frequency on-chip routers. In *HPCA 2010*.



Sheng Ma received the B.S. and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT) in 2007 and 2012, respectively. He visited the University of Toronto from Sept. 2010 to Sept. 2012. He is currently an Assistant Professor of the School of Computer, NUDT. His research interests include on-chip networks, SIMD architectures and arithmetic unit designs.

Zhiying Wang received the PhD degree in electrical engineering from the National University of Defense Technology in 1988. He is currently a Professor with School of Computer, NUDT. He has contributed over 10 invited chapters to book volumes, published 240 papers in archival journals and refereed conference proceedings, and delivered over 30 keynotes. His main research fields include computer architecture, computer security, VLSI design, reliable architecture, multicore memory system and asynchronous circuit. He is a member of the IEEE and ACM.

Natalie Enright Jerger received the B.A.Sc. degree from the Department of Electrical and Computer Engineering at Purdue University, and the M.S.E.E. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Wisconsin - Madison. She is currently an Assistant Professor in the Electrical and Computer Engineering Department at the University of Toronto. Her research interests include on-chip networks, many-core architectures and cache coherence protocols. She is a member of the ACM and a senior member of the IEEE.

Li Shen received the B.S., Master, and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT) in 1997, 2000 and 2003, respectively. He is currently an Associative Professor of the School of Computer, NUDT. His research interests include programming model and compiler design, high performance processor architecture, virtualization technologies, and performance evaluation and workload characterization.

Nong Xiao received the BSc and PhD degrees in 1990 and 1996, respectively, from the School of Computer at the National University of Defense Technology (NUDT). He is currently a Professor in the State Key Laboratory of High Performance Computing at the NUDT. His research interests mainly include Storage, Architecture and Network computing. He is a member of the IEEE.