Transaction-Aware Network-on-Chip Resource Reservation

Zheng Li*, *Student Member, IEEE,* Changyun Zhu†, *Student Member, IEEE,*
Li Shang‡, *Member, IEEE,* Robert P. Dick◇, *Member, IEEE,* Yihe Sun*, *Member, IEEE*

* IME, Tsinghua University † ECE, Queen's University
‡ ECE, University of Colorado at Boulder ◇ EECS, Northwestern University

Abstract—Packet-switched interconnect fabric, widely viewed as the de facto on-chip data communication standard in the many-core era, offers high throughput and excellent scalability. However, these benefits come at the price of router latency due to run-time multi-hop data buffering and resource arbitration, which account for the majority of on-chip transaction latency. In this work, we address the latency issue of on-chip network design and propose dynamic in-network resource reservation techniques that are guided by high-level data transaction information. This idea is motivated by the need to preserve existing abstraction and general-purpose network performance while optimizing for latency-critical events. Experiments with multithreaded benchmarks demonstrate that the proposed techniques reduce on-chip data access latency and demonstrate the importance of considering transaction-level information in network design.

Index Terms—Processor Architectures, Interconnection Architectures, Multiprocessor Systems.

1. Introduction and Motivation

Performance and scalability are primary concerns during on-chip interconnect design for many-core chip-multiprocessors. Compared with circuit-switched interconnects, such as shared bus and point-to-point links, packet-switched fabric offers more efficient resource sharing, higher throughput, and better scalability [16], [2]. While delivering excellent throughput, on-chip networks introduce unnecessary and non-deterministic latency due to multi-hop data buffering and resource arbitration.

This article addresses the latency issue of on-chip network design. This work is motivated by recent research on high-throughput low-latency interconnect fabric design [13], [10]. Most recent work views an on-chip network as a physical interconnect medium and does not explicitly consider the common-case behavior of higher-level protocols and data transactions. In contrast, we argue that knowledge of high-level run-time transactions be used to optimize network performance. We observe the following common traffic patterns for on-chip transactions in many-core chip-multiprocessors.

1) *Heterogeneous performance requirements:* Not every network packet is time critical. For example, read and write requests need to be transferred on time, but evictions and writeback messages do not. Speeding up all the network packets in a throughput-limited network is suboptimal, since non-critical packets might delay critical ones.

2) *Predictable network resource usage:* Run-time data usage typically shows strong temporal and spatial locality, i.e., a processor core repeatedly accesses data from a particular L2 cache. As a consequence, the use of the underlying on-chip network also exhibits locality, e.g., the same routing path is repeatedly used before being discarded.

These observations indicate that high-level data transactions provide valuable information, e.g., distinguishing time-critical traffic from non-critical traffic. Furthermore, we argue that it is possible to optimize network design for commonly-occurring time-critical transactions while maintaining network generality and abstraction. To investigate the performance benefit of leveraging high-level transaction information in on-chip network design and optimization, we develop two techniques, inter-transaction and intra-transaction reservation, which utilize run-time transaction information. These techniques selectively speed up time-critical network traffic, e.g., cache protocol messages and missed data word packets, by prioritizing and pre-planning network resource usage.

2. Transaction Aware Resource Reservation

In this section, we characterize on-chip transaction latency and then present dynamic in-network resource reservation techniques to minimize the latency of commonly-occurring time-critical transactions.

2.A. Transaction Latency Analysis

Consider a tiled-based chip-multiprocessor design. Each tile has a processor core, a private L1 cache, a directory, and an L2 cache. Tiles are connected using canonical input-buffer virtual-channel routers. Each on-chip router consists of five pipeline stages, including flit buffering and routing (RC), virtual channel allocation (VA), switch arbitration (SA), switch traversal (ST), and link traversal (LT) [5]. Only ST and LT stages are associated with intrinsic communication delay, other stages are used for resource arbitration.

The on-chip network supports various transactions, such as read, write, clean eviction, and dirty data writeback. Among those, read and write transactions are time critical. More specifically, read and write miss request messages, critical words, and write acknowledgment messages are time critical. Non-critical messages include write invalidation (if relaxed consistency), eviction, and write back messages. Fig. 1 shows a categorization of the network traffic of six SPLASH2 and ALP-Bench multithreaded benchmarks. Among these benchmarks, the percentage of time-critical packets varies from 10.3% to

This work was supported in part by the NSERC Discovery Grant #388694-01 and in part by the National Science Foundation under awards CNS-0347941, CNS-0702761, and CNS-0720691.
Manuscript submitted: 26-Mar-2008. Manuscript accepted: 02-May-2008. Final manuscript received: 28-May-2008

Posted to the IEEE & CSDL on 6/12/2008
DOI 10.1109/L-CA.2008.9 1556-6056/08/$25.00 © 2008 Published by the IEEE Computer Society

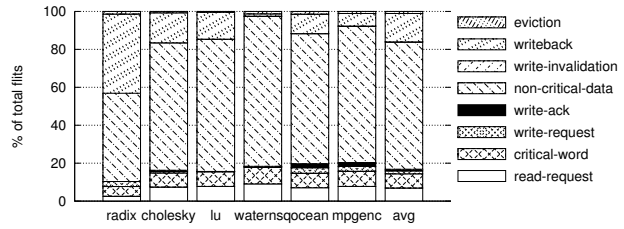


Fig. 1. Network traffic breakdown.

20.3%. On average, only 17.8% of traffic is time critical. It is important to accelerate the time-critical traffic. On the other hand, speeding up non-critical packets may not be beneficial, especially when this has the side effect of slowing down time-critical ones.

Next, we quantify the latency of on-chip data transactions using read transactions as an example. A read transaction starts from an L1 cache miss request, which is forwarded to the corresponding home directory node. If a clean copy of the data exists on chip, the home directory node forwards the request to the corresponding L2 cache node, which then forwards the data to the requesting node. In the event of a cache miss, the request is forwarded to the off-chip memory. A well-designed on-chip cache architecture usually leads to a high L2 cache clean hit rate. The latency of an on-chip read transaction thus results from network propagation delay and L2 cache access latency. This latency can be broken into the following components.

1) $T_{request}$: Network latency to transfer the request message to the corresponding L2 cache node via the home directory node. Request messages are small, typically one flit long.

$$T_{request} = \sum_{\rightarrow directory \rightarrow L2} (N + contention) \quad (1)$$

where N is the number of pipeline stages per router and $contention$ characterizes the average extra resource contention clock cycles per router.

2) T_{cache}: L2 data access latency, which is typically in the range of eight to ten clock cycles using current technology [1].

3) T_{data}: Network latency to transfer the data back to the requesting node. A data packet is large, typically consisting of a whole cache line. Therefore, packet serialization latency should be considered.

$$T_{data} = \sum_{\rightarrow processor} (N + contention) + Flits_{per\ packet} \quad (2)$$

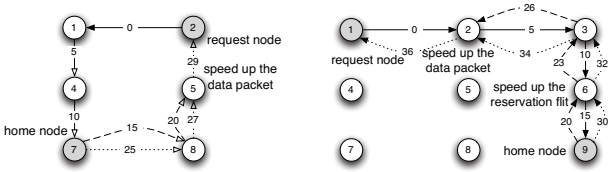
2.B. Transaction-aware Resource Reservation

This section presents run-time network resource reservation and re-planning techniques to accelerate time-critical network traffic. Resource reservation techniques have been proposed in the past [5], [10]. However, past work ignores transaction-level information and treats all on-chip traffic equally, which may potentially slow down time-critical traffic.

2.B.1) Techniques: Run-time data usage shows strong temporal and spatial locality, which has direct impact on the use of physical on-chip network resources. When a processor core repeatedly accesses a particular on-chip cache block, this pattern is reflected in the network resources connecting the processor and cache. It is therefore possible to allow consecutive transactions to reserve and use the corresponding network resources without requesting these resources repeatedly via run-time arbitration. As described in the previous section, on-chip network latency includes $T_{request}$ and T_{data}. To minimize $T_{request}$, when an L1 miss repeatedly requests the same directory and accesses the data from a particular L2 cache, the network resources along the corresponding routing path can then be reserved. Consecutive L1 cache miss requests to the same directory and L2 cache are forwarded along the reserved routing path without the need of resource arbitration. To minimize T_{data}, as soon as a request reaches the destination L2 cache, the return path is reserved in parallel with fetching data from the L2 cache. In addition, if the request message shares the same path with the returning data (in the opposite direction), the request message can be used to reserve the return path. On the other hand, since reserving resources for one transaction may potentially increase the latency of others, it is important to only use resource reservation for time-critical transactions. In this work, we target read and write transactions (assuming L1 write allocation policy).

- **Inter-transaction Reservation:** To reserve network resources for consecutive transactions, a prediction table is added to each router. This table contains $N \times N$ entries, indicating the priorities of the associated input–output port pair determined based on recent transmissions, where N is the number of input/output ports. A prediction is associated with each input–output pair for the next packet from the same input port. The router updates the prediction table when a new packet arrives, i.e., marked input–output pairs that are partially overlapped with the input and output ports used by the new packet will be reset. Router resource reservation is based on the prediction table. Therefore, the prediction table enables a pre-scheduled communication path for incoming packets following the communication paths as previous packets.
- **Intra-transaction Reservation:** Inter-transaction resource reservation may not always be successful due to dynamic changes in run-time data accesses. In contrast, during each transaction, network resources used by the returning data can always be accurately predicted immediately after the corresponding cache node receives a clean data request. The cache node can then issue a resource reservation flit to reserve the network path for data return, which can be done in parallel with L2 cache access, i.e., T_{cache} is then overlapped with T_{data}. The reservation flit will reserve the necessary network resources along the paths to the requesting processor node like a "circuit switch" network. After the data are fetched from the L2 cache and injected into the network, network resource arbitration is no longer necessary. Two reservation flits may try to reserve the same resource for the same time slot. A first-come first-serve rule is currently implemented. The later attempt fails and is removed from the network. The corresponding data packet must then rely on run-time arbitration starting from the router where the failure occurs.
- **Accelerated Intra-transaction Reservation:** During a successful reservation, a reservation flit traverses the normal router pipeline stages while the following data packet is stepped up. In large-scale networks, the reservation flit might potentially be caught up by the data packet along a long routing path. Fortunately, resource reservation can also be used to accelerate the reservation flit. More specifically, when the request message and the returning data share the same routing path (in opposite directions), request packets are used to reserve network resources for the returning reservation flits, with an accuracy as high as the L2 clean hit rate.

2.B.2) Mechanisms: Once network usage is predicted and network resources are reserved, router pipeline stages, such as resource arbitration and data buffering, can be bypassed. For intra-transaction reservation, reservation flits can accurately



(a) Baseline (b) Accelerated

Fig. 2. Intra-transaction Reservation. Nodes represent routers. Solid lines indicate request packets, dashed lines indicate reservation flits, and dotted lines indicate data packets. Numbers on line indicate the start time of the corresponding transfers.

predict the arrival times of data packets. Therefore, the RC, VA, and SA routing pipeline stages can be bypassed. For request packets in inter-transaction reservation and accelerated reservation flits, the SA stage cannot be eliminated due to potential mis-prediction and resource conflicts. However, they have priorities in SA stage. Fig. 2 illustrates intra-transaction reservation without (a) and with (b) the proposed *accelerated* technique. In this technique, a request packet predicts and reserves a backward path for the reservation flit, which performs accurate resource reservation for the upcoming data packet. Since the accelerated reservation flit and reserved data packet do not need a regular buffer, the change of the path will not result in resource dependence between adjacent routers. Therefore, no deadlock will occur.

3. Hardware Design

This section describes the implementation of transaction-aware resource reservation. In addition to the input–output port prediction table, every input port requires two registers to record the reserved output port and the corresponding accurate time slots for the data packets. The registers are updated by reservation flits. The request packet is sometimes used to accelerate the reservation flit. Therefore, two additional registers per port are needed to record the output port and the approximate time slot for the reservation flit. One flit buffer per input port is added to accommodate the stepped-up flit that failed switch arbitration. Area estimation using models from prior work [3] shows that the proposed techniques increase the router area by about 8%. In addition, the latencies of the switch arbitration and switch traversal stages increase slightly due to priority arbitration and an extra multiplexer stage. However, delay analysis in TRIPS [8] as well as by Peh and Dally [14] show that the virtual channel allocation stage typically bounds clock frequency. We therefore believe that our techniques will have little impact on clock frequency.

4. Experimental Results

To evaluate the proposed techniques, we implemented a cycle-accurate cache-network simulator. The simulator supports k-ary n-cube network topologies consisting of pipeline virtual-channel input-buffer routers and two levels of on-chip cache hierarchy and directories. The experimental setup adopts a hierarchical memory architecture that imitates the server consolidation scenario for many-core chip-multiprocessors [12]. The L1 cache is private, and the L2 cache is only shared among processor cores within the same virtual machine. Network traffic traces are gathered using the M5 full-system simulator [4] running six SPLASH2 [15] and ALPBench [11] multithreaded benchmarks, including mpgenc, radix, lu, cholesky, ocean, and waternsq. The system configuration is summarized in Table 1. In addition, we consider three different workload consolidation

TABLE 1
Configuration

Topology	4-ary & 6-ary 2-mesh
Routing	Dimension-ordered
Arbitration	Basic Separable Allocator
Number of router ports	5
VCs per port	5
Buffers per VC	5
Flit size/channel width	8 Bytes
L1 cache per tile	64KB, 2-way 64-byte line
L2 cache per tile	1MB, 16-way 128-byte line
L2 data array access time	10 cycles
Memory access time	100 cycles
Coherence protocol	MOSI
Processors	Alpha 21264
Clock freq.	2 GHz

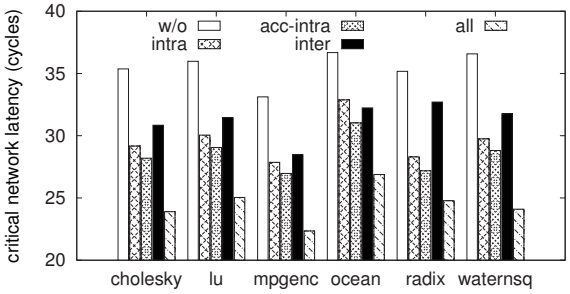


Fig. 3. Network latency reduction of a 4×4 system.

models. *Local:* The threads of the same benchmark are assigned to neighboring processor cores with strong spatial locality. *Random:* The threads of the same benchmark are randomly assigned without spatial locality. *Normal:* A combination of *Local* and *Random* workload assignment policy, which preserves moderate spatial locality.

We first consider a 4×4, 16-node configuration using six single-benchmark workload setups, each of which consists of a 16-thread benchmark. Fig. 3 shows the network latency reduction of the on-chip data transactions using different multithreaded benchmarks with and without the proposed techniques. As shown in this figure, both the proposed intra-transaction and inter-transaction techniques reduce network latency. The "acc-intra" and "intra" columns show network latency reductions with and without the *accelerated* technique enabled in the proposed intra-transaction optimization. When both intra-transaction and inter-transaction techniques are enabled, network latency is reduced. The network read and write access latencies of the 16-thread cholesky, lu, mpgenc, ocean, radix, and waternsq applications are reduced by 32.4%, 30.4%, 32.5%, 26.7%, 29.6%, and 34.1%, respectively, with 31.0% latency reduction on average. The improvement is due to the acceleration of timing-critical packets.

Next, we consider a 6×6, 36-node configuration using nine different workload configurations. Each workload contains one 16-thread application (lu, mpgenc, or waternsq) plus the 4-thread radix, lu, cholesky, mpgenc, and waternsq applications. We apply the three different consolidation models for each workload assignment. Fig. 4 shows the simulation results, which demonstrate that the proposed techniques consistently reduce network latency. Using the 9 different workload configurations, the proposed techniques achieve an average reduction in network latency of 31.5% (29.1% minimum and 33.8% maximum). The latency reduction of the proposed techniques improves when the average on-chip communication distance

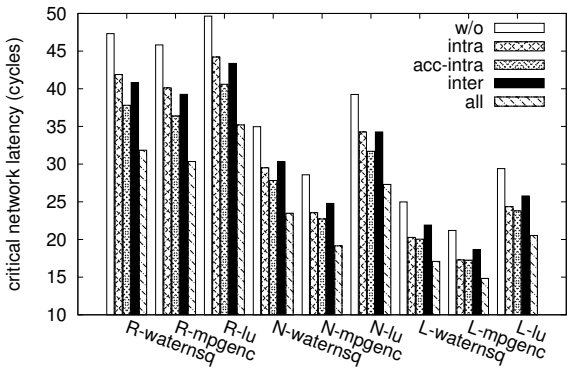


Fig. 4. Network latency reduction of a 6×6 system.

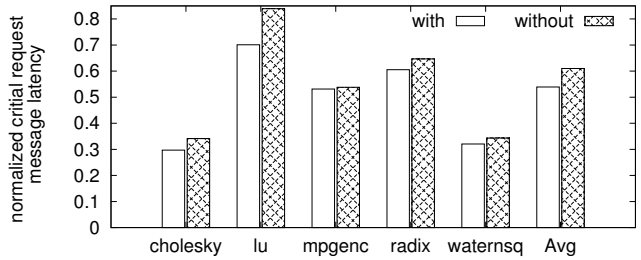


Fig. 5. With and without transaction information guidance.

increases, i.e., from the *Local* model to *Random* model. On average, 86.6% of the data packets were reserved, 74.8% of the predictions are correct, and 59.8% of the reservation flits are accelerated. As a side-effect, the average latency of non-critical packets like eviction and writeback was increased by 7.7%.

To demonstrate the importance of considering transaction-level information to distinguish time-critical traffic from non-critical traffic, we extend the proposed intra-transaction technique to all protocol packets (including both time-critical and non-critical ones) and compare the critical request messages latency. The result is shown in Fig. 5. It shows that when the latency optimization technique is applied to all on-chip traffic, the average latency of time-critical traffic increases by 13.0% on average. Such performance degradation is mainly due to non-critical packets blocking time-critical read and write requests. In addition, the mixture of different types of messages reduces prediction accuracy, hence the effectiveness of resource reservation. This study indicates that accelerating all on-chip traffic is suboptimal. It is valuable to distinguish network packets based on transaction semantics.

5. Related Work

Recent work on on-chip network design proposes alternative topologies and routing algorithms to optimize network latency by minimizing network hop counts. Most past work concentrates on the physical communication medium without considering the common-case behavior of the protocols built upon it, like "Flit-Reservation Flow Control" [5], which is a pure network technique and uses a separate network to transfer control flits. Recently, in-network cache coherence [7] moves part of the directory into routers, which minimizes network hop counts but increases router area significantly and complicates the coherence protocol. Circuit switch coherence [9] proposes a hybrid on-chip network design and a prediction-based coherence protocol, but it requires a separate network to set up the circuit switch path. Ding et al. [6] manage network resources predictively, but high-level compiler and branch predictors are required to perform such prediction. Express virtual channel [10] also focuses on minimizing network per-hop delay, but it is a pure network-level approach. A low-latency single-cycle on-chip network architecture is proposed using network-level techniques [13].

6. Conclusions

In this article, we propose run-time transaction-aware on-chip network resource reservation techniques to minimize transaction latency in many-core chip-multiprocessors. Our study demonstrates the importance of considering high-level run-time transactions during on-chip network design and optimization. Detailed simulation demonstrates that, by leveraging transaction-level information, the proposed inter-transaction and intra-transaction resource reservation techniques identify and accelerate on-chip time-critical transactions.

References

[1] "CACTI: An integrated cache access time, cycle time, area, leakage, and dynamic power model," http://quid.hpl.hp.com:9082/cacti/.

[2] "Tilera TILE64 chip-multiprocessor," http://www.tilera.com.

[3] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Proc. Annual International Conference on Supercomputing*, Jun. 2006, pp. 187–198.

[4] N. L. Binkert *et al.*, "The M5 simulator: Modeling networked systems," *Proc. Int. Symp. Microarchitecture*, vol. 26, no. 4, pp. 52–60, 2006.

[5] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann Pub., 2003.

[6] Z. Ding *et al.*, "Switch design to enable predictive multiplexed switching in multiprocessor networks," in *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium - Papers*, 2005, p. 100.1.

[7] N. Eisley, L.-S. Peh, and L. Shang, "In-network cache coherence," in *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 321–332.

[8] P. Gratz *et al.*, "On-chip interconnection networks of the TRIPS chip," *IEEE Micro*, vol. 27, no. 5, pp. 41–50, Sept.-Oct. 2007.

[9] N. E. Jerger, M. Lipasti, and L.-S. Peh, "Circuit-switched coherence," *IEEE Comput. Archit. Lett.*, vol. 6, no. 1, 2007.

[10] A. Kumar *et al.*, "Express virtual channels: towards the ideal interconnection fabric," in *Proc. Int. Symp. Computer Architecture*. New York, NY, USA: ACM, 2007, pp. 150–161.

[11] M.-L. Li *et al.*, "The ALPbench benchmark suite for complex multimedia applications," in *Int. Symp. Workload Characterization*, Oct. 2005, pp. 34–35.

[12] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," in *Proc. Int. Symp. Computer Architecture*. New York, NY, USA: ACM, 2007, pp. 46–56.

[13] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proc. Int. Symp. Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2004, p. 188.

[14] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. Int. Symp. High-Performance Computer Architecture*, Jan. 2001, pp. 255–266.

[15] "SPLASH2 website," http://www-flash.stanford.edu/apps/SPLASH/.

[16] S. Vangal *et al.*, "An 80-tile 1.28TFLOPS networks-on-chip in 65nm CMOS," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2007, pp. 98–100.