Latency Criticality Aware On-Chip Communication

Zheng Li*, Jie Wu*, Li Shang†, Robert P. Dick‡, and Yihe Sun*
* Tsinghua National Laboratory for Information Science and Technology, Inst. of Microelectronics,
Tsinghua University, Beijing 100084, China {zheng-li, wujie07}@mails.tsinghua.edu.cn sunyh@tsinghua.edu.cn
† Dept. of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309, U.S.A li.shang@colorado.edu
‡ Dept. of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, U.S.A dickrp@eecs.umich.edu

***Abstract*—Packet-switched interconnect fabric is a promising on-chip communication solution for many-core architectures. It offers high throughput and excellent scalability for on-chip data and protocol transactions. The main problem posed by this communication fabric is the potentially-high and nondeterministic network latency caused by router data buffering and resource arbitration. This paper describes a new method to minimize on-chip network latency, which is motivated by the observation that only a small percentage of on-chip data and protocol traffic is latency-critical. Existing work focusing on minimizing average network latency is thus suboptimal. Such techniques expend most of the design, area, and power overhead accelerating latency-noncritical traffic for which there is no corresponding application-level speedup.**

We propose run-time techniques that identify latency-critical traffic by leveraging network data-transaction and protocol information. Latency-critical traffic is permitted to bypass router pipeline stages and latency-noncritical traffic. These techniques are evaluated via a router design that has been implemented using TSMC 65 nm technology. Detailed network latency simulation and hardware characterization demonstrate that, for latency-critical traffic, the proposed solution closely approximates the ideal interconnect even under heavy load while preserving throughput for both latency-critical and noncritical traffic.

1. Introduction and Motivation

In the past few years, packet-switched interconnect fabric has been adopted in multi-core and many-core architectures. Several designs, such as TRIPS [1], TILE64 [2], Larrabee [3], the Cell processor [4], and Intel's 80-core prototype chip [5], have used packet-switched interconnect fabric for global on-chip communication. Leveraging router-by-router data buffering and resource arbitration, packet-switched interconnect fabric is capable of flexibly sharing communication resources among on-chip concurrent traffic flows, providing high throughput and excellent scalability.

A primary performance concern of packet-switched on-chip interconnect fabric is its significant and non-deterministic latency, which has two sources, complex router pipeline and run-time contention among concurrent traffic flows. A modern on-chip network is composed of a number of router modules, which perform functions such as data buffering, resource sharing and arbitration. A canonical input-buffer virtual-channel router [6] consists of five pipeline stages, including buffer write and routing (BW and RC), virtual channel allocation (VA), switch allocation (SA), and switch traversal (ST). In addition, a link traversal (LT) stage is required to relay data between two neighboring routers. Furthermore, on-chip communication resources, i.e., buffers and links, are shared by concurrent traffic flows. Packets belonged to different traffic flows may compete for these resources, thus introducing additional and non-deterministic latency. Our study shows that, in a tiled shared-memory CMP design in which the network is used to relay L2 cache data traffic, network latency can account for 73.3% on-chip data access latency (see Section 2).

Researchers have recently proposed a number of high-throughput low-latency interconnect fabric designs. Kumar et al. proposed Express Virtual Channel, a flow-control method, to statically and dynamically bypass router pipeline stages [7]. Lu, Liu, and Jantsch proposed Layered Switching which combines virtual cut-through and worm whole switching. Grouped flits are created as flow control unit to optimize the utilization of wiring and switching resources [8]. Mullins, West, and Moore proposed a speculative single-cycle router that uses pre-computed arbitration results to save router pipeline stages [9]. Kumar et al. proposed a network-on-chip router using advance bundle signals to set up the network and prioritize arbitration techniques to improve throughput [10]. Jerger, Lipasti, and Peh proposed circuit switch coherence, a hybrid on-chip network design with a prediction-based coherence protocol, enabling significant latency reduction [11]. These techniques are capable of reducing average network latency. Some recent work began to consider coherence protocol information for NoC design. Eisley, Peh, and Shang proposed in-network cache coherence, moving part of the directory into routers, which minimizes network hop counts but increases router area significantly and complicates the coherence protocol [12]. Bolotin et al. proposed an NoC prioritization policy to speed up short cache access requests [13].

In this work, we aim to reduce the latency of on-chip networks without introducing area, throughput, or power overhead. Most past work focuses on optimizing the average network traffic latency and does not explicitly consider and differentiate the timing characteristics of high-level data and protocol transactions. In contrast, we argue that knowledge of high-level network traffic transactions should be used to optimize network performance. Our work is motivated by the following timing characteristics of on-chip network traffic in multi-core/many-core designs.

1) *Heterogeneous latency requirements:* The on-chip network relays cache and coherence protocol traffic in a shard-memory CMP. Not every packet in on-chip network traffic is latency-critical. Protocol requests, acknowledgment packets, and critical word packets in read and write transactions are latency-critical – the processor might be stalled for data or write ownership during data transfer. On the other hand, packets in write-back and eviction transactions, as well as the remainder of transferred cache lines, are less time critical. We also find that latency-critical traffic only accounts for a small

This paper was supported in part by the National Natural Science Foundation of China (NSFC) under grant #60236020 and the Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP) #20050003083, in part by the NSF under awards CCF-0829950, CCF-0702761, CNS-0347941, and CNS-0720691.

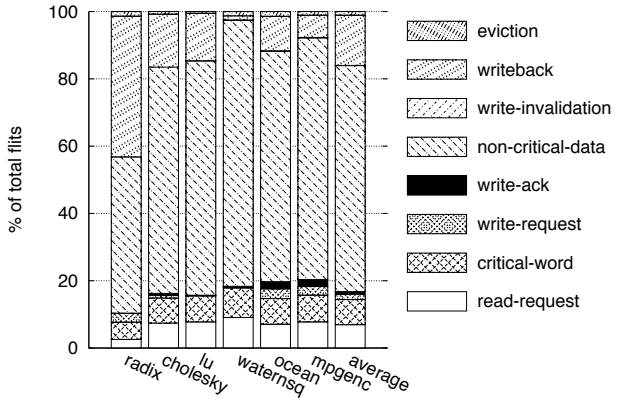


Fig. 1. Network traffic composition.

portion of the overall network traffic (see Section 2).

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 within a small time interval. As a consequence, on-chip network traffic also exhibits locality, e.g., the same routing path is repeatedly used. The temporal and spatial locality of network traffic can be utilized to better pre-plan the network resource usage for latency-critical traffic.

In this work, we leverage the information of high-level network data and protocol transactions to identify latency-critical network traffic. Aggressive prediction techniques are then proposed to speed up latency-critical traffic with little hardware overhead. The proposed techniques, including locality-aware resource planning, latency-critical virtual channels, and prioritized switch arbitration, can reduce the latencies to values comparable to those of ideal interconnect fabric (i.e., intrinsic interconnect delay) even under heavy network traffic workload. In contrast, existing work targeting all network traffic yields suboptimal results – the small portion of latency-critical traffic can be delayed by the remaining noncritical traffic. In addition, speeding up the non-critical traffic may require great effort with little system-level impact.

We have designed a 65 nm on-chip router to evaluate these techniques. Detailed hardware characterization and network-level simulation show that the proposed design achieves network latency within 6.3% (on average) of ideal interconnect, while preserving excellent network throughput for both latency critical and non-critical traffic.

2. Traffic Characterization

This section characterizes on-chip network traffic. We first analyze the timing properties of on-chip data and protocol transactions in order to identify latency-critical transactions. We then evaluate the temporal and spatial locality of latency-critical traffic. This study demonstrates a performance gap between the state-of-the-art solutions and ideal interconnect, and motivates the optimization techniques proposed in Section 3.

2.A. Latency-Critical Traffic Analysis

In a shared-memory multi-core/many-core on-chip system, the network is used for cache data and protocol transactions, including read, write, clean eviction, and dirty data write back. Among these transactions, read and write transactions impact cache miss latency, and thus system-level performance. We therefore identify read and write miss request packets, critical words, and write acknowledgment packets to be *latency-critical*. On the other hand, network traffic transactions, including write invalidation (if relaxed consistency), eviction, the remainder of the cache lines (excluding the critical word), and write back packets are latency-noncritical traffic. Fig. 1 shows a categorization of the network traffic of six SPLASH2 and ALPBench multithreaded benchmarks for the *baseline* network configuration described in Section 4. On average, only 17.8% of the traffic is latency-critical (with a range of 10.3%–20.3%).

2.B. Characterization of Critical Traffic

Next, we investigate the characteristics of latency-critical traffic. Consider a latency-critical cache read. This transaction starts with 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 an L2 cache miss, the request is forwarded to off-chip memory. A well-designed on-chip cache architecture usually has a high L2 cache clean hit rate. The latency of an on-chip read transaction can thus be decomposed into the following components.

1) $T_{request}$: Network latency to transfer the request to the corresponding L2 cache node via the home directory node. Request packets are small, typically one flit long (flit is the basic unit of flow-control. In this paper it is also the basic physical unit, i.e. data transferred per cycle).

$$T_{request} = \sum_{req.\rightarrow air.\rightarrow L2} (d/v + T_{router} + T_{contention}) \quad (1)$$

where d is length of wires per hop, v is the propagation velocity, T_{router} is the delay through a single zero-load router, and $T_{contention}$ characterizes the average extra resource contention delay per router.

2) T_{cache}: L2 access latency, which is typically in the range of eight to ten clock cycles using current technology [14].

3) T_{data}: Network latency to transfer the data, e.g., a cache line, back to the requesting node. A cache line consists of the critical (causing the cache miss) and non-critical words. The arrival of the critical words satisfies time-critical requests.

$$T_{data} = \sum_{L2\rightarrow req.} (d/v + T_{router} + T_{contention}) \quad (2)$$

Using the setup described in Section 4, network latency, $T_{request} + T_{data}$, accounts for 73.3% of on-chip data access latency. Minimizing the latency of critical traffic therefore optimizes system-level performance.

2.C. Towards Ideal Network Latency

For a latency-critical flit, the latency of each router hop can be decomposed as follows.

$$T_{hop} = \underbrace{d/v + T_{crossbar}}_{T_{ideal}} + \underbrace{T_{BW} + T_{VA} + T_{SA} + T_{contention}}_{T_{gap}} \quad (3)$$

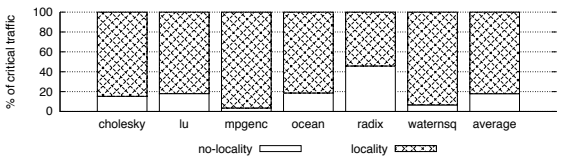


Fig. 2. Locality analysis for critical traffic.

where T_{BW} is the time the flit spends in buffers. T_{VA} and T_{SA} are the times the flit spends in arbitrating buffer and switching resources. $T_{crossbar}$ is the time to actually traverse the router. $T_{ideal} = d/v + T_{crossbar}$ indicates the intrinsic network delay, i.e., the ideal network latency. $T_{gap} = T_{BW} + T_{VA} + T_{SA} + T_{contention}$ indicates the extra router pipeline and resource contention latency in state-of-the-art network design, i.e., the performance gap compared to the ideal network latency.

To optimize the latency-critical traffic, it is critical to minimize T_{gap}. To this end, we make the following observations, which drive the latency criticality aware router design described in the next section.

1) As shown in Fig. 1, latency-critical flits only account for a small portion of the overall network traffic. As a result, $T_{contention}$ is mainly due to run-time resource contention with non-critical traffic. Therefore, to optimize the performance of latency-critical traffic, it is crucial to minimize run-time resource contention with non-critical traffic. Existing work, which seeks to minimize average network latency, fails to consider the performance impact of non-critical traffic on critical traffic, leaving substantial room for improvement.

2) Run-time traffic also exhibits strong temporal and spatial locality, mainly resulting from data access locality in program behavior. We measure the percentage of flits with the same input–output port pairs as the previous flit. Fig. 2 shows that 82.1% critical flits have this property. Deterministic routing further strengthens the traffic locality. Locality can thus be used to preplan the network resource for latency-critical traffic to minimize T_{gap}.

3. Latency Criticality Aware Router Design

This section presents the proposed latency criticality aware router design. As shown in Section 4, the proposed design is capable of bringing the performance of latency-critical traffic near that of ideal interconnect, while maintaining high throughput for both critical and non-critical traffic. In this section, we first describe a locality-aware resource planning technique that leverages the temporal and spatial locality of network traffic and performs resource planning and reservation for latency-critical traffic. We then describe resource prioritization polices appropriate in the absence of traffic locality. This section also discusses the timing characteristics of the proposed 65 nm router design. Detailed hardware characterization is presented in Section 4.

3.A. Locality-Aware Resource Planning

Network traffic during program execution often exhibits temporal and spatial locality. When a processor core repeatedly accesses a particular cache block, the underlying network data transactions also have strong locality. This locality can be used to allow consecutive network data transactions to reserve the corresponding network resources, reusing them many times without repeatedly requesting access via run-time arbitration. More specifically, at each router hop, T_{SA}, T_{VA}, and T_{BW} can be bypassed. Run-time resource contention, $T_{contention}$, due to noncritical traffic can also be avoided.

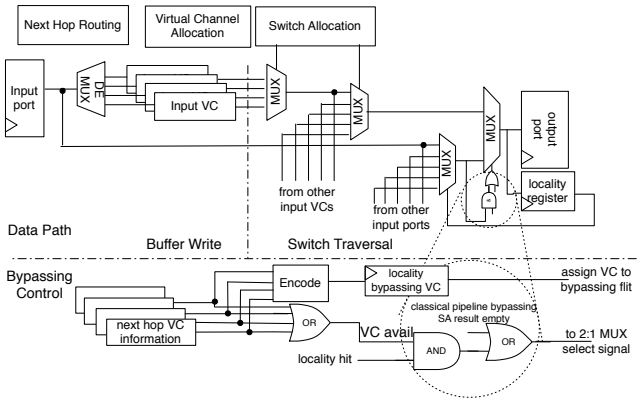


Fig. 3. Router microarchitecture and locality-aware resource planning

Supporting locality-aware resource planning requires minor modification of the control logic of crossbar and buffers. The following sections describe the proposed technique and implementation in detail.

3.A.1) Crossbar-Supported Locality Bypassing: Fig. 3 shows the locality-aware bypassing logic design, which is essentially a crossbar design that selects incoming flits from two sets of inputs: input virtual channels or directly from router input ports (avoiding BW, VA, and SA). This enables latency-critical flits to move directly from router input to output without data buffering. The datapath portion of the proposed crossbar design is similar to the recently-proposed pipeline bypassing router design [7], [10]. These previous designs allow an incoming flit to enter the ST stage speculatively if no other flit resides in the input port. However, this situation will be rare under heavy workload, which is also generally the situation in which network latency is of greatest importance. In contrast, the proposed approach exploits traffic locality, which is common under heavy workload, enabling it to speed up latency-critical traffic.

The proposed locality-aware crossbar design is equipped with a *locality register* at each crossbar output port, which records the input channel information of the latest bypassed latency-critical flit. The locality register is updated once a new critical flit travels through the output port. Its content allows prediction of the outcome of switch allocation for the corresponding output port. A locality hit occurs when two consecutive incoming latency-critical flits arriving at the same input ports target the same output port. The second latency-critical flit can then be directly forwarded from the router input to the crossbar output. In other words, the preceding latency-critical flit reserves the router switching resource for succeeding flits. When a locality hit occurs, the latency-critical flit traverses through the router with only one pipeline stage ST, bypassing the BW, SA, and VA stages and avoiding possible resource contention. Any other flits requesting the same output port during the same clock cycle arbitrate for the crossbar resource in the next clock cycle.

3.A.2) Inter-Router Virtual Channel Management: When a locality miss occurs, the latency-critical flit needs to request a virtual channel for temporary data buffering. Lack of available virtual channels can potentially block following latency-critical packets. To rapidly detect such hazards, each switch output port is equipped with a *locality bypassing VC* register that maintains the available virtual channel in the corresponding input buffer of the successor router, thereby avoiding virtual channel allocation.

Traditional, load balancing based virtual channel allocation is based on the buffer usage among virtual channels, i.e., the virtual channel with the maximum credit count is chosen. This design requires comparator tree logic with significant area and timing overhead. Latency-critical packets are typically short. Therefore, load balancing is not necessary. The logic to generate the *locality bypassing VC* value is simplified: select the first virtual channel with non-zero credit. In addition, a *VC avail* signal based on the availability of virtual channels of the successor router serves as the enable signal for crossbar bypassing. Note that a non-critical packet may also be allocated to the same virtual channel as the *locality bypassing VC*. Since latency-critical flits have higher priority, such double assignment will not cause resource conflict.

3.B. Latency-Critical Virtual Channel

As described in the previous section, a latency-critical flit will be assigned to a virtual channel upon a locality miss. Under heavy workload, virtual channels may be occupied by non-critical traffic, which would delay latency-critical traffic. To avoid this, the proposed router design is equipped with a special *latency-critical virtual channel* for each physical port reserved for latency-critical traffic. During virtual channel allocation, latency-critical packets are allowed to arbitrate for all the available virtual channels of the successor router, while non-critical packets are only allowed to arbitrate regular virtual channels. In on-chip network design, router buffers are mainly used for throughput enhancement. Since latency-critical traffic only takes a small portion of the network traffic, and latency-critical packets are typically short, a small virtual channel is generally sufficient to cover the round trip latency of a latency-critical packet. Therefore, only a small portion of router buffer resources need to be allocated to the latency-critical traffic, resulting in only a small performance impact on non-critical traffic.

3.C. Prioritized Switch Allocation

To minimize $T_{contention}$ due to switch allocation when locality misses occur, a prioritized switch allocation scheme is used. Higher switching resource access priorities are given to latency-critical flits. A separable allocator [6] is used. Switch allocation is partitioned into two tasks: local arbitration across the input ports, and global arbitration across the output ports. A flit request is submitted to separate input and output port arbiters. To reduce $T_{contention}$, critical flits have priority over non-critical flits during each arbitration. Using separate arbiters allows efficient implementation. The prioritized arbiter is implemented as a combination of priority-handling logic and a conventional matrix arbiter, as follows. Assuming n requesters for one resource, each requester has priority $prio(i)$, and asserts its request with $req_i(i)$. The request sent to a conventional arbiter would be $req_o(i)$.

$$prio_{any} = \sum_{i=1}^{n} (prio(i) \cdot req_i(i)) \tag{4}$$

$$req_o(i) = (\overline{prio_{any}} + prio(i)) \cdot req_i(i) \quad \text{for all } i \tag{5}$$

The result $req_o(i)$ is sent to a matrix arbiter [6] that provides a strong fairness guarantee.

3.D. Hardware Critical Path Analysis

We next analyze the critical path of the router hardware implementation. Detailed hardware characterization will be presented in Section 4. The following analysis assumes P physical ports per router and V virtual channels per port.

BW stage: a flit from the input port is stored to a specific virtual channel buffer, which is implemented using circular buffer. The critical path of this pipeline stage consists of a $V:1$ demultiplexer and buffer writing.

VA stage: The header flit of each packet performs virtual channel allocation to request buffer resources from next-hop router. The latency-critical virtual channel can only be allocated to the critical packets. Other virtual channels are available for all traffic. As shown in the previous section, the virtual channel allocator is implemented as a canonical separable allocator. Thus, the only impact on the critical path is the addition of two $P \times V : 1$ arbiters.

SA stage: Switch allocation is also implemented with a separable allocator that prioritizes latency-critical traffic. The critical path of this stage is the combination of prioritized arbiters.

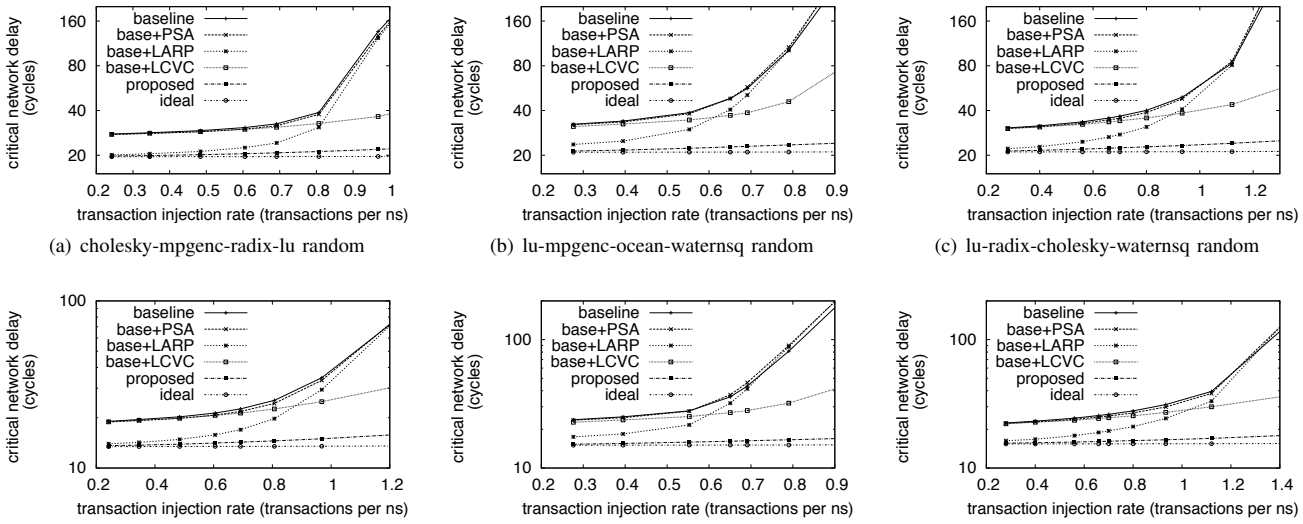
ST stage: Starting from a baseline crossbar design with support for pipeline bypassing, a *locality register* is added to each output port to record the input port information of incoming traffic. A latency-critical flit for which locality-based prediction succeeds will traverse directly from the router input port to the crossbar output. This bypassing is enabled by a new stage of 2:1 multiplexers. The selection signal is determined by *VC avail* and the criticality tag of this flit. A *locality bypassing VC* is also generated to update the flit's destined VC field. The critical path of this stage includes buffer read time, two stages of $V:1$ and $P:1$ multiplexers, and a final 2:1 multiplexer for bypassing selection.

4. Experimental Results

This section evaluates the proposed design using detailed router hardware characterization and network-level simulation.

4.A. Network-Level Results

We implemented a cycle-accurate cache–network simulator. The simulator supports k-ary mesh topologies consisting of pipelined virtual-channel input-buffer routers, two levels of on-chip cache hierarchy, and cache directories. The experimental setup uses a hierarchical memory architecture that imitates the server consolidation scenario for many-core CMPs [15]. Four 16-processor virtual machines are consolidated into one 64-processor CMP. The L2 cache is distributed and shared within the same virtual machine. Since we are considering NoCs for shared memory CMPs, network traffic traces for each virtual machine are gathered using the M5 full-system



(a) cholesky-mpgenc-radix-lu random (b) lu-mpgenc-ocean-waternsq random (c) lu-radix-cholesky-waternsq random

(d) cholesky-mpgenc-radix-lu normal (e) lu-mpgenc-ocean-waternsq normal (f) lu-radix-cholesky-waternsq normal

Fig. 4. Low-latency experiments (critical network latency).

TABLE I: CONFIGURATION

Topology	8-ary 2-mesh
Number of router ports	5 (5 VCs per port)
Buffers per phy. channel	25 (5 per VC)
Flit size/channel width	8 Bytes
L1 cache per tile	64KB, 2-way 64-byte line
L2 cache per tile	256KB, 16-way 128-byte line
L2/memory access time	10/100 cycles
Coherence protocol	MESI
Processors	Alpha 21264

simulator [16] running six 16-thread SPLASH2 [17] and ALPBench [18] multithreaded benchmarks, including mpgenc, radix, lu, cholesky, ocean, and waternsq. The system configuration is summarized in Table I.

We evaluate the network design under two workload models: *random*, in which nodes of the same virtual machine are randomly assigned without spatial locality and *normal*, in which nodes of the same virtual machine are assigned to neighboring processor cores with moderate spatial locality.

Six router configurations are implemented to evaluate the proposed latency-criticality aware router design. The baseline router model, which represents the state-of-the-art design, is a four-stage input virtual channel router with look-ahead routing, speculative switch allocation, and pipeline bypassing. The three techniques discussed in previous section, namely locality-aware resource planning (LARP), latency-critical virtual channel (LCVC), and prioritized switch allocation (PSA), are each individually added to the baseline design to show their effects on latency and throughput. In addition to these designs and the proposed router model, we also estimate the ideal network latency by removing all non-critical packets from the network, thereby preventing interference with critical packets.

During our evaluation, we randomly pick four of the six available benchmarks and consolidate them into one network traffic pattern. We further temporally dilate and contract the traffic to evaluate the performance of the proposed design under different network traffic workload.

Fig. 4 shows the average latency of latency-critical traffic for each of the router configurations. Compared to the baseline state-of-the-art design, the proposed design reduces the critical traffic latency by 36.2% on average (31.1% minimum and 42.9% maximum). Moreover, the proposed design is capable of achieving critical traffic latency within 6.3% of the ideal network. Among the individual proposed techniques, locality bypassing is most effective. The locality bypassing success rate is 85.0% on average. Locality bypassing alone reduces critical traffic latency by 25.5% of the baseline case. Given heavy traffic, i.e., the injection rate with double the zero-load latency, the proposed design can still maintain the latency of critical traffic within 11.9% of that for the ideal network. Under heavy traffic, buffer resources are the resource bottleneck. In these conditions, latency-critical virtual channel management is the most effective single technique. Fig. 5 evaluates the network throughput using different router configurations. The network throughput is defined as the packet injection rate at which the network average latency is double the zero-load latency. This study shows that the proposed design does not decrease network throughput. In addition, since critical packet latency is significantly reduced, the overall network throughput increases by 4.5% compared to the baseline state-of-the-art design (maximum 5.6% and minimum 3.1%). In summary, these studies demonstrate that the proposed design approaches the latency of ideal interconnect for latency-critical traffic while preserving high throughput for both latency-critical and noncritical traffic.

4.B. CIRCUIT-LEVEL RESULTS

The proposed latency criticality aware router is evaluated using a TSMC 65 nm low-power technology under typical condition (1.2 V and 25°C) with nine metal layers. Timing and power results are estimated using Synopsys Design Compiler in topographical mode, which uses a default floorplan to estimate wire delay. Chip floorplan and layout are generated using Synopsys IC Compiler. We compare our design with the

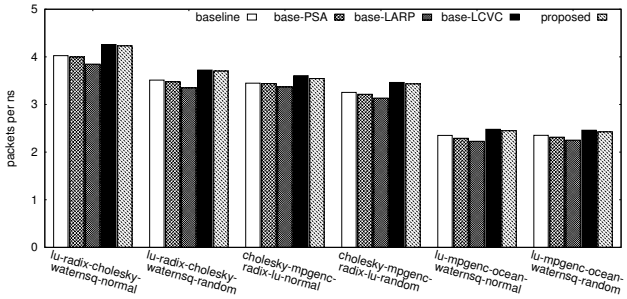


Fig. 5. Throughput of each configuration.

previously-described baseline state-of-the-art router.

Area: Fig. 6 illustrates the floorplan of the proposed router. The input buffer and crossbar dominate the router area, which is estimated to be 0.42 mm². Compared to the baseline design, the proposed design's area overhead is approximately 5% in terms of equivalent gate count.

Timing: Table II shows the synthesis result for each pipeline stage. The switch traversal stage is time-critical; it limits the frequency of the router to 1.34 GHz, which is similar to the 1.41 GHz peak frequency of the baseline design. This slight latency increase is due to the bypassing multiplexer selection logic and increased capacitive load due to the locality registers. In addition, the latencies of VA and SA stages both increase due to latency-critical VC and prioritized SA design. However, these two stages are not the on critical paths in this design.

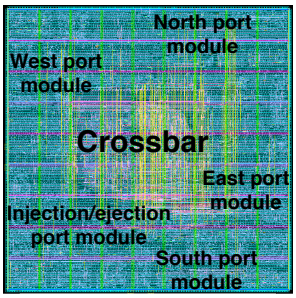


Fig. 6. Router layout (645 µm × 645 µm).

TABLE II: CRITICAL PATH DELAY

Router pipeline stages	Critical path delay baseline (ps)	Critical path delay proposed (ps)
BW	620	626
VA	648	654
SA	627	660
ST	709	743
Frequency (GHz)	1.41	1.34

Power: Power analysis (including dynamic and leakage power) using average-case router traces from network-level simulation shows that the power consumption of the proposed latency criticality aware router is 359.9 mW at peak frequency, which is within 2.3% of the baseline design (351.8 mW).

5. CONCLUSION

In this paper, we have identified a weakness in existing on-chip packet-switched interconnect fabric design techniques. These techniques focus on reducing average packet latency instead of focusing on reducing delays for the latency-critical packets that influence system-level performance. We have determined that it is possible to significantly improve the latencies of these critical packets (by 36.2% on average) while preserving the throughput of non-critical traffic. In order to achieve this goal, we have proposed and evaluated three techniques: locality-aware resource planning, latency-critical virtual channel, and prioritized switch allocation. Combined, these techniques allow critical traffic latencies approaching those of an ideal interconnect fabric (6.3% difference). We have completed a detailed hardware design of the proposed router and found that area, power consumption, and clock frequency differ little from those of a state-of-the art router design. In summary, we have developed and evaluated a low-overhead router design that significantly improves the latency of critical network traffic.

REFERENCES

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

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

[3] L. Seiler, et al., "Larrabee: a many-core x86 architecture for visual computing," *ACM Trans. on Graphics*, vol. 27, no. 3, pp. 6–23, Aug. 2008.

[4] D. Pham, et al., "Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 179–196, Jan. 2006.

[5] S. Vangal, et al., "An 80-tile 1.28 TFLOPS network-on-chip in 65nm CMOS," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2007, pp. 98–589.

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

[7] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: towards the ideal interconnection fabric," in *Proc. Int. Symp. Computer Architecture*, June 2007.

[8] Z. Lu, M. Liu, and A. Jantsch, "Layered switching for networks on chip," in *Proc. Design Automation Conf.*, June 2007, pp. 122–127.

[9] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proc. Int. Symp. Computer Architecture*, 2004, pp. 188–197.

[10] A. Kumar, P. Kundu, A. Singh, L.-S. Peh, and N. Jha, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm cmos," in *Proc. Int. Conf. on Computer Design*, Oct. 2007, pp. 63–70.

[11] N. E. Jerger, M. Lipasti, and L.-S. Peh, "Circuit-switched coherence," *IEEE Computer Architecture Letters*, vol. 6, no. 1, pp. 5–8, 2007.

[12] N. Eisley, L.-S. Peh, and L. Shang, "In-network cache coherence," in *Proc. Int. Symp. Microarchitecture*, Dec. 2006, pp. 321–332.

[13] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny, "The power of priority: NoC based distributed cache coherency," in *Proc. Int. Symp. Networks-on-Chip*, May 2007, pp. 117–126.

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

[15] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," in *Proc. Int. Symp. Computer Architecture*, June 2007, pp. 46–56.

[16] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, 2006.

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

[18] M.-L. Li, R. Sasanka, S. V. Adve, Y.-K. Chen, and E. Debes, "The ALPbench benchmark suite for complex multimedia applications," in *Proc. Int. Symp. Workload Characterization*, Oct. 2005, pp. 34–35.