Packet Chaining: Efficient Single-Cycle Allocation for On-Chip Networks

George Michelogiannakis, Nan Jiang, Daniel U. Becker and William J. Dally
Electrical Engineering, Stanford University, Stanford, CA 94305
E-mail: {mihelog,njiang37,dub,dally}@stanford.edu

Abstract—This paper introduces *packet chaining*, a simple and effective method to increase allocator matching efficiency and hence network performance, particularly suited to networks with short packets and short cycle times. Packet chaining operates by chaining packets destined to the same output together, to reuse the switch connection of a departing packet. This allows an allocator to build up an efficient matching over a number of cycles, like incremental allocation, but not limited by packet length. For a 64-node 2D mesh at maximum injection rate and with single-flit packets, packet chaining increases network throughput by 15% compared to a conventional single-iteration separable iSLIP allocator, outperforms a wavefront allocator, and gives comparable throughput with an augmenting paths allocator. Packet chaining achieves this performance with a cycle time comparable to a single-iteration separable allocator. Packet chaining also reduces average network latency by 22.5% compared to iSLIP. Finally, packet chaining increases IPC up to 46% (16% average) for application benchmarks because short packets are critical in a typical cache-coherent CMP. These are considerable improvements given the maturity of network-on-chip routers and allocators.

Index Terms—On-chip interconnection networks, Interconnection architectures

1 Introduction

The performance of a network-on-chip (NoC) is sensitive to the matching efficiency of the allocators used to allocate switch ports and virtual channels (VCs) in the routers. Allocators can easily be in the critical path of routers and therefore present a trade-off between matching efficiency and cycle time [2].

Separable allocators are a popular choice for modern NoCs because of their short timing path. However, in a single iteration, input or output arbiters may make unfavorable decisions because they are arbitrating independently of each other. In the extreme case of single-flit packets, the allocator starts from scratch each cycle and in a single iteration is not able to compute an efficient matching. While multiple iterations or more complex allocators, such as wavefront or augmenting paths, would improve matching efficiency, they are typically not feasible within the available timing or cost budget, especially for high-radix routers [2], [4].

To provide the efficiency of multi-iteration allocation without extending cycle time, past work has proposed incremental allocation [9]. In this scheme, allocation extends over many cycles and new requests can be injected in any cycle. The results of each iteration generate grants. However, incremental allocation provides no benefit to single-flit packets and little benefit to short packets, which dominate traffic in a typical cache-coherent chip multiprocessor (CMP). To exemplify the point, 53% of the packets in the application benchmarks we simulated in this paper were single-flit and received no benefit from incremental allocation.

In this paper, we introduce *packet chaining* which operates by chaining packets destined to the same output together, to reuse the switch connection of a departing packet. Even with uniform random traffic, a significant number of packets request the same output at each router by following the routing algorithm. This allows an allocator to build up an efficient matching over a number of cycles, like incremental allocation [9], but not limited by packet length. Packet chaining is implemented with an extra allocator, in parallel with the switch allocator. Therefore, packet chaining incurs only a marginal increase in allocation time, and a small cost overhead. Finally, to provide limited fairness we add starvation control which releases a connection after it has been held for a predetermined number of cycles. This way, connections may not be re-used indefinitely, and therefore other packets will not starve. Packet chaining increases allocation efficiency of an iterative allocator to be comparable or higher than more expensive allocators. Packet chaining provides minimal benefits to non-iterative allocators.

Packet chaining extends pseudo-circuits [1] by chaining packets from any input and VC instead of solely the same input VC, as well as maintaining connections when there are conflicting requests, to increase allocation efficiency under load.

• *Manuscript submitted: 25-Apr-2011. Manuscript accepted: 01-Jun-2011. Final manuscript received: 03-Jun-2011.*

2 Detailed Description

2.1 Packet Chaining

Packet chaining in effect *chains* packets together by holding any finishing connections that can be used by waiting packets, even if they are from different inputs or VCs than the packet currently using the connection, so they look like one longer packet to the switch allocator. The switch allocator starts, not from scratch, but from this initial state of chained connections.

A new waiting packet is suitable for chaining if (a) it has been routed to the same output as the tail flit of the departing packet, (b) there is a free output VC it is eligible to use, and (c) there is at least one credit for that output VC. The chained packet need not be at its start. Partially transmitted packets can be chained — in this case the only eligible output VC is the one to which the packet is already assigned, which is stored in control state logic of input VCs.

Fig. 1 illustrates an example allocation with packet chaining for a 6×6 router. Each input VC contains a single one-flit packet which is labeled by its destination output port. In the same example, iSLIP without packet chaining would restart the allocation process every cycle and in a single iteration would not able to compute an efficient matching. iSLIP separable allocators [8] use round-robin arbiters and update the priorities of each arbiter when that arbiter generates a winning grant. In Fig. 1, an X denotes a connection during the previous cycle and a dot (•) denotes

Posted to the IEEE & CSDL on 6/9/2011
DOI 10.1109/L-CA.2011.15 1556-6056/11/$26.00 © 2011 Published by the IEEE Computer Society

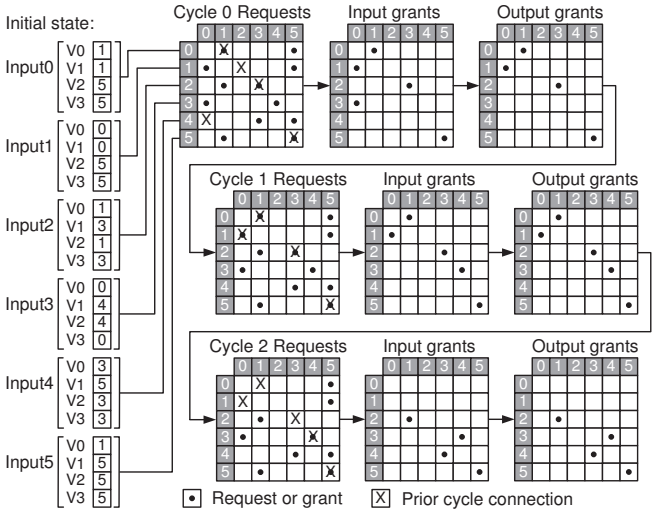

Fig. 1. Example allocation of iSLIP with packet chaining.

a request or a grant. In cycle 0, the allocator starts with five connections inherited from cycle -1. Three of these connections are reused by new packets requesting the same output as denoted by a square in the cycle 0 request matrix with both an X and a •. The other two connections, denoted by an X without a •, are terminated. The three chained packets eliminate competing switch requests. Only one of the new requests is granted by the output arbiter, giving four packets transmitted in cycle 0.

As shown in the second row of Fig. 1, all four connections from cycle 0 are chained in cycle 1. The allocator makes one additional grant giving a total of five packets transmitted in cycle 1. In cycle 2, only two of the five connections are chained and the allocator makes two additional grants, resulting in four packets being transmitted.

Other than output 2 receiving no requests, there are only two other idle output cycles, compared to five for allocation without chaining in the same scenario. Of these two idle cycles, only the first –on output 4 in cycle 0– is avoidable. A better allocator could have assigned VC 1 or 2 from input 3 to this channel. The allocator matchings in cycles 1 and 2 are both maximum. However, packet chaining does not always result in maximum or maximal matchings. For example, an additional allocator iteration would add a grant from input 3 to output 4 in cycle 0.

Connections are released if they cannot be used productively either because the output VC has no more credits or the input VC becomes empty [7]. To accommodate higher-priority traffic, a connection is released by a higher-priority request for the connected output. Furthermore, we provide limited fairness by implementing starvation control, which is also applicable to incremental allocation. If a connection has been held for more than a maximum number of cycles, the connection is released (potentially mid-packet). Connections that will reach the starvation threshold at the next cycle are not eligible for chaining. Thus, these ports can be reassigned to waiting packets by the switch allocator. Finally, packet chaining does not cause out-of-order delivery of packets or flits if they would otherwise be ordered.

We implement packet chaining on top of a combined [7] switch/VC iSLIP allocator that reserves output VCs only for packets that win switch allocation. This leaves more output VCs free compared to performing VC allocation in advance, therefore giving more flexibility to packet chaining to find free output VCs.

2.2 Chaining Variations

We consider three variations for the VCs considered for chaining:

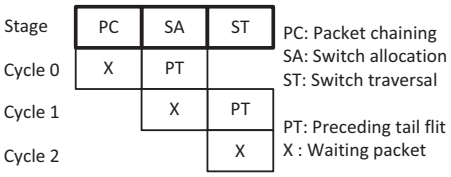

Fig. 2. Packet chaining pipeline.

- *Same input VC*: The simplest scheme is to consider only the same input VC as the previous packet that used the connection.
- *Same input, any VC*: This scheme considers all eligible VCs of the same input.
- *Any input, any VC*: This scheme considers eligible packets in any input and any VC.

Restricting packet chaining to consider a single input significantly simplifies the packet chaining logic. A full PC allocator is required only if considering all inputs and VCs.

2.3 Packet Chaining Pipeline

Packet chaining adds an extra PC (packet chaining) allocation stage to a conventional two-stage VC router in parallel with switch allocation as illustrated in Fig. 2. Newly arriving packets skip this new stage starting directly in SA (switch allocation). Hence, adding the PC stage does not increase router latency. Waiting packets, however, are chained during the PC stage. The figure shows a waiting packet X that shares its output with PT (a preceding tail flit). Packet X is granted the connection from the PC allocator during cycle 0, while PT is in the SA stage. If PT does not hold a connection and fails switch allocation during cycle 0, PC allocation is cancelled and both packets remain in the same pipeline stage. In this case, PT receives a switch grant and traverses the switch during cycle 1, while the head flit of X advances to the SA stage but does not participate in switch allocation. In our latency-optimized two-stage pipeline, chained packets may not skip SA even if no flit is in ST, because that would require a separate VC allocator and would complicate timing with the input channels, buffers and routing logic.

Because packet chaining operates in its own pipeline stage, it is guaranteed to chain an eligible packet and remove competing packets from consideration during the SA stage. Biasing the switch allocator to favor maintaining the connection does not achieve the same end because the competing packets are not removed from consideration, and thus may impact switch allocator decisions negatively.

Packet chaining eligibility is determined at the beginning of the cycle. Packets considered for chaining do not participate in switch allocation because their output is connected. However, conflicts may still arise between the switch and PC allocators because each input has multiple VCs. If the two allocators grant the same input, the PC allocator's decision is disregarded. In addition, the eligibility of an input VC may depend on switch allocator decisions, which are unknown during PC allocation. For example, an input VC may contain a packet eligible for chaining, but the input port which contains that VC may be part of another connection which has to be released first. Similarly, a tail flit without a connection will only provide a chaining opportunity if it receives a switch grant, thus forming a connection. Any packet that will become eligible for chaining only by a favorable switch allocator decision generates a lower-priority request to the PC allocator. For input VCs participating in switch allocation, PC allocator requests are generated based on the flits *behind* the buffer head. Those PC requests are also marked as lower-priority.

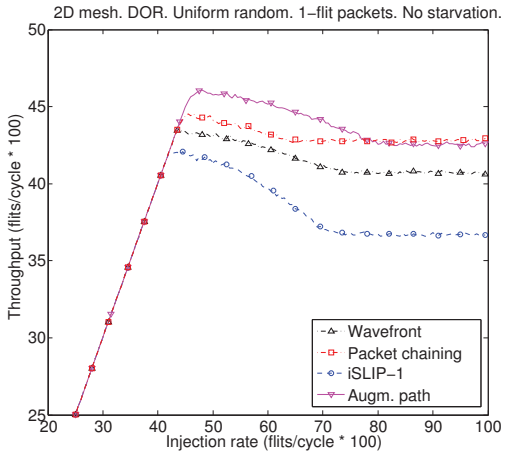

Fig. 3. An illustration of network instability.

3 Methodology

Evaluation is performed with a cycle-accurate network simulator. We use a 2D mesh arranged in an 8×8 grid. Routers use the pipeline of Section 2.3. Each router is connected to one network terminal, and all channels have a single cycle of latency. We use deterministic dimension-order routing (DOR) because it is a simple and popular choice. We use 4 VCs, with 8 buffer slots statically assigned to each. For this configuration, packet chaining is optimal when considering all VCs of the same input as departing tail flits because with DOR packets are unlikely to switch dimensions. Therefore, the packet chaining results we present assume this scheme. For less predictable routing algorithms, considering all inputs and VCs may be appropriate.

We use uniform random, random permutation, shuffle, bit complement and tornado traffic patterns. Injection rate is in flits. Incremental allocation [9] is used for networks without packet chaining. We also present results for a typical cache-coherent CMP with 64 superscalar and out-of-order RISC CPUs. Each core has two threads. We use a custom execution-driven simulator. Network datapath width is 64 bits. Therefore, short packets are single-flit while packets carrying our 32-byte cache lines have five flits. A starvation threshold of eight cycles is used. L1 caches are 8KB, four-way set-associative, have a single cycle of latency and are private to the cores. L2 caches are shared, non-inclusive, four-way set-associative, have 32KBs per core, and have five cycles of latency. There is one directory and one L2 cache slice at each core, and one memory controller at every network quadrant. We assume cores clocked at four times the network frequency. Results for IPC correspond to those for speedup. On training input datasets IPC results matched measured speedup.

4 Evaluation

4.1 Throughput and Latency

As illustrated in Fig. 3, compared to iSLIP-1 (incremental allocation without packet chaining), packet chaining increases throughput under maximum injection rate by 15% with single-flit packets, by reducing tree saturation [6]. This is important for systems because without elaborate throttling, it is very difficult to consistently operate a network at the point of saturation. In addition, a throttling mechanism might be overly conservative thus reducing available throughput. With packet chaining, throughput drops only marginally (2.5%) past saturation. Packet chaining offers 6% more throughput at maximum injection rate compared to wavefront, and comparable throughput (1% more) to an augmenting paths allocator. Packet chaining achieves these gains without

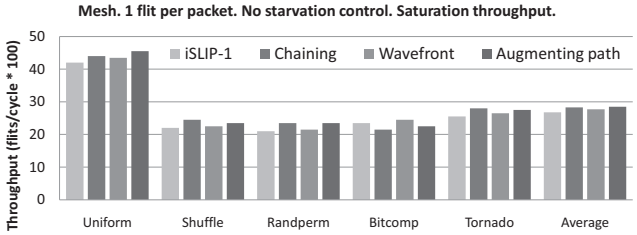

Fig. 4. Allocator comparison by traffic pattern.

the extra cost and complexity of wavefront and augmenting paths. Augmenting paths allocators are especially complex because they locate all paths from unmatched inputs to unmatched outputs in the directed bipartite allocation graph [4]. Packet chaining is comparable or marginally better than augmenting paths because augmenting paths may make suboptimal decisions at the network level and does not regard fairness. Packet chaining does not reduce instability from globally unfair allocation.

Fig. 4 compares saturation throughput by traffic pattern. Compared to iSLIP-1, packet chaining offers up to 12% (5.5% average) higher throughput. Compared to wavefront, throughput increases up to 9.5% (2.5% average). By average, packet chaining is comparable to an augmenting paths allocator. Performance differences become smaller with non-uniform random traffic patterns because they use only a subset of router inputs and outputs, making allocation easier. Packet chaining provides lower throughput for *bitcomp* (bit-complement) because bitcomp creates some continuous flows of traffic which starve other flows. By using a starvation threshold of four cycles with bitcomp, packet chaining offers comparable (2% higher) saturation throughput to iSLIP-1.

Moreover, packet chaining provides a 22.5% lower average latency compared to iSLIP-1 — computed as an average from low to maximum injection rates. The number of cycles eligible head flits spend blocked waiting for a switch allocator grant or the connection to their desired output to be released is reduced by 13% for single-flit packets and 7.5% for eight-flit packets. Furthermore, in the application benchmark simulations of Section 4.4, maximum packet latency was reduced by an average of 20% while average latency was reduced by 7%. Latency reductions are due to more efficient allocation, because flits are more probable to advance if their output is free.

4.2 Packet Length

Packet chaining gains decrease as packet length increases, because long packets increase allocator efficiency using incremental allocation. This is illustrated in Fig. 5. Throughput gains drop to 2% (comparable throughput) for eight-flit or longer packets, compared to iSLIP-1. Packet chaining provides a marginal (1.5%) throughput gain even for sixteen-flit packets, because some traffic patterns benefit in throughput from the lack of starvation control. In those cases, starvation control increases fairness and provides performance no lower than iSLIP-1. Traffic patterns which create both short and long packets (bimodal), provide larger throughput gains for packet chaining than patterns with only long packets. For instance, when assuming a request-reply protocol with single-flit short and five-flit long packets, packet chaining provides a 4% increase for uniform random traffic. These gains are compared to 2.5% for five-flit packets and 5% for single-flit packets.

Even though performance gains drop, packet chaining still provides comparable or marginally increased throughput than more complex allocators. With eight-flit packets, packet chaining is comparable to wavefront (outperforms it by 2%), and augmenting paths (1.5%), by average across traffic patterns. For uniform

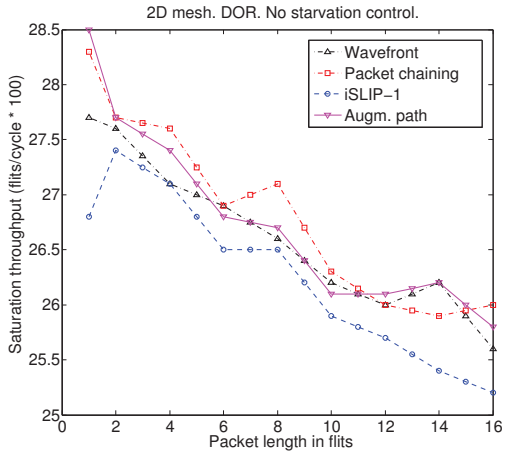

Fig. 5. Throughput by average across traffic patterns.

random traffic, packet chaining is comparable to augmenting paths and provides a 2.5% higher throughput than wavefront.

Throughput drops for all test cases with the increase of packet length. This is due to diminishing effects with our constant buffer size which cause increased blocking in VCs. With packet chaining, long packets can be divided into shorter ones to avoid this effect, without loss of allocation efficiency.

4.3 Starvation and Priorities

Starvation control has a small effect on throughput. A starvation threshold of eight cycles is comparable for single-flit (1.5% higher throughput) and for eight-flit packets (equal throughput) to disabling starvation control. A starvation threshold lower than the packet length reduces throughput gains because starvation control releases connections mid-packet, so packets wait for another switch allocator grant while having reserved an output VC. In the cases where performance drops with starvation control, packet chaining never performs worse than iSLIP-1.

Disabling priority-handling in the PC allocator reduces throughput by 6.5% for uniform random traffic and 4.5% by average across traffic patterns, with single-flit packets. That is because PC allocator requests which are more likely to be cancelled due to unfavorable switch allocator decisions are no longer lower priority.

4.4 Application Performance

TABLE 1 presents results for five PARSEC [3] benchmarks and FFT. In these simulations, 53% of the packets were single-flit. Except for increased throughput, the primary factor for the IPC increase is reducing packet latency. Applications with an increased network load receive higher benefits from packet chaining. However, applications in CMPs make light use of the network if their working sets fit in the L1 caches, and thus are not affected by techniques improving throughput, such as packet chaining. Therefore, application gains depend on the working set's size and the cache hierarchy, which affect the amount of network traffic. Packet chaining makes the option of making a network cheaper and thus more throughput-limited more attractive by increasing throughput and reducing latency past very low loads.

4.5 Packet Chaining Cost

The vast majority of the cost for packet chaining is that of the PC allocator. Implementations of combined allocators with priorities and the associated control path consume 7% [5] or 2.5% [7] of router power, and 7% [7] of router area. This cost is amortized when considering it in the network or system level. Also, by

TABLE 1
Packet chaining versus iSLIP-1 using benchmarks.

Benchmark	IPC increase	Benchmark	IPC increase
Blackscholes	46%	Canneal	1%
Dedup	6%	FFT	9%
Fluidanimate	3%	Swaptions	29%
Average	16%		

average, the switch allocator activity factor will be reduced, reducing its dynamic power. Finally, in implementations where the switch allocator is not in the critical path, packet chaining does not increase cycle time. Even if the switch allocator is in the critical path, it is lengthened only by the simple peripheral logic required for packet chaining, such as determining input VC eligibility and masking out PC allocator decisions due to conflicts.

5 Conclusions

Packet chaining extends the benefits of incremental allocation to packets of any length. As we have shown, packet chaining improves the IPC of application benchmarks on a typical cache-coherent CMP by up to 46% (16% average) because short messages dominate CMP traffic. On synthetic traffic patterns at maximum injection rate and single-flit packets, packet chaining increases throughput by 15% and decreases latency by 22.5% compared to an iSLIP-1 allocator with comparable cycle time. In fact, packet chaining outperforms a wavefront allocator which is typically not feasible within the cycle time of a router [2], [4]. It also gives performance that is comparable to an ideal (but not realizable) augmenting paths allocator. This is true for both short and long packets. Considering the mature state of router and allocator design, these performance results represent a considerable improvement.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant CCF-0702341, in part by the National Security Agency under Contract H98230-08-C-0272-P007 and in part by the Robert Bosch, Prof. Michael Farmwald and Prof. Michael J. Flynn Stanford Graduate Fellowships.

References

[1] M. Ahn and E. J. Kim. Pseudo-circuit: Accelerating communication for on-chip interconnection networks. In *MICRO '43: Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010.

[2] D. U. Becker and W. J. Dally. Allocator implementations for network-on-chip routers. In *SC'09: Proceedings of the 2009 ACM/IEEE Conference on Supercomputing*, pages 1–12, 2009.

[3] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[4] R. R. Hoare, Z. Ding, and A. K. Jones. A Near-optimal Real-time Hardware Scheduler for Large Cardinality Crossbar Switches. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC'06)*, 2006.

[5] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro*, 27(5):51–61, 2007.

[6] C. P. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Trans. Comput.*, pages 1091–1098, December 1983.

[7] A. Kumar, P. Kundu, A. Singh, L.-S. Peh, and N. Jhay. A 4.6tbits/s 3.6ghz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *Proceedings of the 25th International Conference on Computer Design*, pages 63 –70, 2007.

[8] N. McKeown. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transanctions on Networking*, 7:188–201, 1999.

[9] S. S. Mukherjee, F. Silla, P. Bannon, J. Emer, S. Lang, and D. Webb. A comparative study of arbitration algorithms for the alpha 21364 pipelined router. *SIGARCH Comput. Archit. News*, 30:223–234, 2002.