

Curing Hotspots in Wormhole NoCs

Isask'har Walter¹, Israel Cidon², Ran Ginosar², Avinoam Kolodny²

Electrical Engineering Department, Technion, Haifa, Israel

¹zigi@tx.technion.ac.il ²{cidon, ran, kolodny}@ee.technion.ac.il

ABSTRACT

Network on Chip (NoC) has been proposed for future SoC interconnect. Hotspots are SoC modules which occasionally receive traffic that exceeds the rate at which they can absorb data. Hotspots are common in real-life SoCs, such as external DRAM or internal components (caches, CAMs, special purpose processors) that are bandwidth limited and in high demand by other units. In this paper we demonstrate that hotspot modules on wormhole-based NoCs dramatically reduce network efficiency and unfairly allocate system resources. A single hotspot may ruin the performance of the entire NoC. In order to resolve these problems, we introduce a novel low-cost end-to-end credit based resource allocation technique that regulates access to the hotspot module. Using simulation, we show the effectiveness of the suggested mechanism.

Categories and Subject Descriptors

System-Level Design and Co-Design: Network-on-Chip (NoC)

General Terms

Algorithms, Performance, Design

Keywords

Network on-Chip, wormhole, flow-control, hotspot, SoC

1. INTRODUCTION

Wormhole [1] switching is commonly employed in NoC (e.g. [2], [3], [4], [5]), thanks to its small buffer requirements and low latencies at light load. Each packet is divided into small fixed size parts called flits, which are transmitted to the next hop without waiting for the entire packet to be received. This causes the transmitted packet to be “spread” along the path between the source and destination nodes in a pipeline fashion. The main drawback of wormhole interconnect is the high sensitivity to packet blocking, calling for high performance networks to operate at relatively low utilization and to include multiple virtual channels [6].

Bandwidth can be allocated to NoC links for providing adequate performance [7], but speeding up the operation of IP modules is impossible or has an unacceptable cost. Furthermore, at certain times the aggregated traffic demand might exceed the destination module’s bandwidth capacity. A bandwidth-limited SoC module working close to its capacity is termed a *hotspot*. In this situation, when the module is unable to consume incoming packets fast enough, the entire network may be affected. Hop-by-hop backpressure causes buffers at the router adjacent to the hotspot to

be filled up and become stalled blocking new arrivals to this router. This creates a domino effect, by which the delivery of packets to ports of more distant routers is slowed down, forming a *saturation tree* [8] with the hotspot module as its root, as illustrated in Figure 1. The NoC suffers increased delays in packet delivery and unfair network utilization (modules near the hotspot get larger portion of its capacity).

While this effect may also exist in packet based store-and-forward networks, the threat is particularly troublesome in wormhole based architectures due to packet “stretching” across several hops. As a result, hotspot effects may extend network wide instantly. It is important to note that this hotspot phenomenon is independent of links bandwidth, as a saturation tree may build up in a system with infinite capacity links and a single heavily loaded module. Consequently, even carefully designed, largely over-provisioned NoCs may suffer of poor performance if potential hotspots are left unhandled.

We propose a novel credit-based resource allocation mechanism for solving hotspot congestion problems in wormhole-based NoCs. A hotspot allocation controller is introduced to arbitrate short, high priority credit requests. The controller regulates hotspot access according to the quality of service requirements of the specific system. Credit requests and grants are transmitted as high-priority packets (grants may be piggybacked on other messages). Auto-refresh or pre-allocation can be used for selected modules to eliminate light load latencies. One side of the mechanism is implemented in modules’ interfaces and the other in a strategic location, while NoC routers remain unchanged. This prevents the accumulation of packets destined at a hotspot module in the network. Consequently, other traffic remains unaffected even when the hotspot load increases significantly.

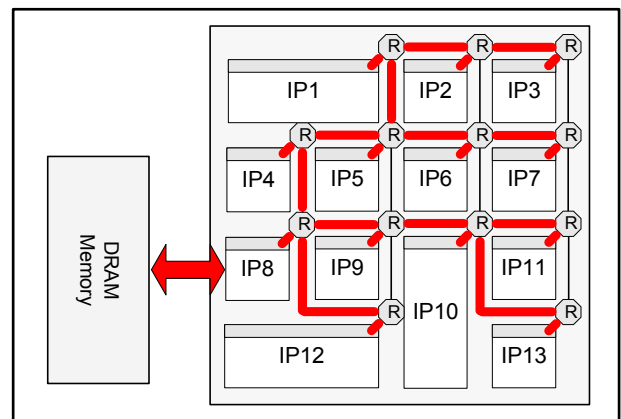


Figure 1: SoC hotspot at external DRAM interface and the resulting NoC saturation tree (highlighted links)

The rest of this paper is organized as follows: Related work is surveyed in section 2. In section 3 we discuss the negative effects of hotspots in wormhole-based NoCs. In section 4, an end-to-end allocation technique is proposed to allow fair sharing of the hotspot resource and to mitigate effects on non-hotspot traffic (traffic not destined at the hotspot module), and section 5 presents simulation of the suggested mechanism.

2. RELATED WORK

The problem of hotspot contention has been thoroughly explored in off-chip interconnection networks (e.g. [8], [9]). Many papers analyzed hotspot effects on non-hotspot traffic and suggested various schemes for improving system performance. However, most previous works have addressed multi-computer networks, in which the design considerations are significantly different from those of NoCs. For example, most works modify the network routers in order to throttle packet injection at high loads (e.g., [10], [11]), discard packets (e.g., [12]), deflect packets away of loaded locations (e.g. [13], [14]), use separate buffers for traffic destined at a hotspot module (e.g. [9]), or simply use a large number of virtual channels. However, when applied to NoCs, such modifications considerably increase NoC router gate count, resulting in excessive area and power consumption and reduced speed. Typically, NoCs must employ static shortest path routing based on a simple routing function, because of on-chip cost and performance considerations. Moreover, those works do not address the hotspot allocation fairness problem. If we assume a router-based solution to this problem (such as fair queueing mechanism per source-destination pair [15]) or complicated admission control [16], the NoC costs become infeasible.

The proposed hotspot resource allocation mechanism is considerably different from traditional end-to-end flow-control mechanisms. Flow-control is conducted on a per-source basis (e.g. TCP, static window in [17]), and prevents overflow in the destination buffers pre-allocated for this source (e.g. [18]). Flow control does not treat the hogging of network resources and does not address the problem of fair allocation of scarce hotspot resources. In addition, existing schemes require at least one destination buffer per potential source, which is inappropriate in on-chip NoCs. Our scheme, using credit request messages, takes care of all latter problems.

3. HOTSPOT EFFECTS

The formation of a saturation tree due to hotspot congestion has several negative effects on system performance. *Hotspot access latency* is increased, as packets destined at the hotspot contend for buffer space and bandwidth. However, this effect is unavoidable when demand approaches available hotspot module capacity. Unfortunately, additional significant problems arise: Typically, different source modules are at different distances from the hotspot module (as illustrated in Figure 2 using a common NoC topology). Since a packet has to win local output port arbitration in each router along its path, the hotspot module is not fairly shared. In particular, modules close to the hotspot enjoy a larger share of saturation bandwidth than distant ones. Location and distance diversity also lead to differences in access latency, as

distant modules experience very long access times. This will be referred to as the *source fairness* problem.

Performance degradation due to hotspot load is not restricted to hotspot (HS) traffic (i.e., traffic destined at the hotspot). In typical NoCs, HS and non-HS traffic compete for the same network resources, i.e. buffer space, link bandwidth and router ports. Therefore, hotspots may hinder the delivery of non-HS packets (Figure 3). The above discussion may apply to any network. However, left unhandled, hotspot effects in a wormhole network are more severe than in a store-and-forward one, as packets are blocked across multiple routers and buffering space is limited.

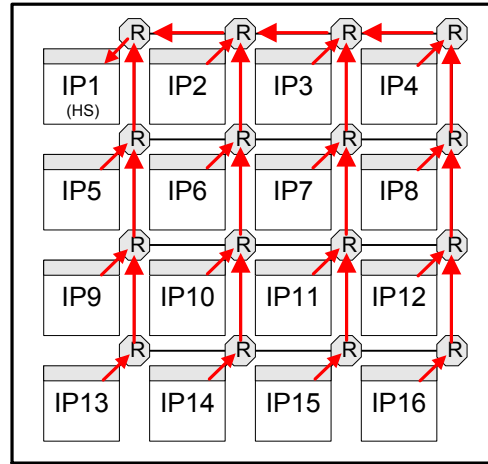


Figure 2: Source Unfairness in a 4x4 YX routed NoC
On its way to the hotspot module (#1), packets generated by module 16 have to win 6 arbitrations, while module 5 packets have to win only 2.

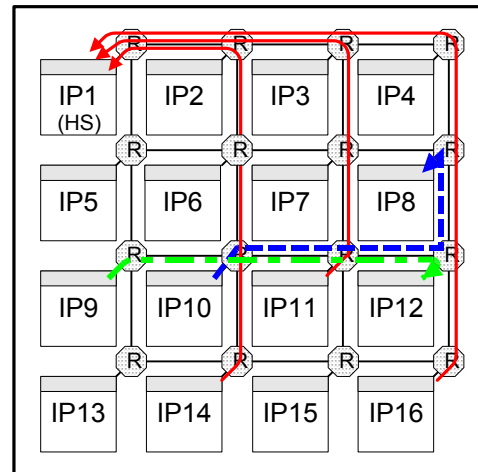


Figure 3: Hotspot traffic obstructing non-HS traffic.
Flow 16→1 may slow-down (or block) flow 10→8 (which shares a link), and in turn may affect flow 9→12.

4. HOTSPOT RESOURCE ALLOCATION

In order to reduce the dramatic effects of hotspots in a wormhole-based NoC (section 5), a credit-based resource allocation mechanism is suggested: Each source gets a quota that limits the number of flits it can send towards a hotspot module. When a source quota is exhausted, it can resume transmission only after being granted additional credit. Consequently, a saturation tree can not form and traffic not destined at the hotspot module remains unaffected during congested periods.

Two types of control messages regulate the access to a hotspot: If a source has insufficient credit to start delivery of a data packet to a hotspot module, it sends a *credit request* packet to a hotspot allocation controller, describing the requested transaction. When appropriate, the controller sends back credit using a *credit reply* packet. Due to their significance and short length, credit request and reply messages are given a high priority level and cannot be held back in the network by data packets. In this work, we assume a prioritized virtual channel mechanism such as the one described in [6], to deliver control messages, guaranteeing fast hotspot controller access regardless of data traffic loads.

Since control packets are a few flits long and a single control packet credits a large chunk of data over the same path, it is clear that the control traffic is a small percentage of the HS traffic. Therefore, the buffers of the prioritized virtual channel are kept at low utilization, resulting in minimal network queuing time. In order to overcome credit request and reply latency in light load periods, source quota can be slowly self refreshing.

4.1 Control Messages

Using a credit request message, a source describes the data packet(s) it wishes to send to the hotspot module and asks for credit to do so. In addition to the *Destination ID* field of a regular data packet, a request packet contains two mandatory fields: *Source ID* and *Length*. The former states the requesting module identity and the latter describes the size (in flits) of the data packet to be delivered. The system designer may choose to include additional information which would enable the hotspot controller to decide upon the best service order. This information can be embedded in optional fields of the request packet. An example of such a field is a priority value, which indicates the "urgency" of the data packet, relative to requests that are sent by other sources of the same kind. A deadline field that indicates the requested completion time can help the hotspot controller sort the requests in the best servicing order, postponing less urgent requests to be serviced later. If requests can be ignored unless they are served by a certain time, an expiration field may be used.

Figure 4(a) illustrates an example of a credit request packet in which each field fits a flit (more fields per flit are of-course possible). Figure 4(b) illustrates a credit reply packet. The destination ID field is used to route the packet back to the requester. The source ID enables the requester to identify the controller sending the reply and is necessary in a system with multiple hotspots. The *Credit* field states the number of credits granted in the reply packets. Generally, this number is equal to the length field in the matching request packet. However, a hotspot controller may choose to reply with a larger number in order to credit modules ahead of time during light load periods. The hotspot module may also reply with less credit than

requested. In this case, a source may choose to send part of the data packet, thus freeing up local buffer space, or send another pending but shorter packet instead. Other optional fields may include a specific time or earliest time of transmission so credit can be given and scheduled among multiple sources ahead of time.

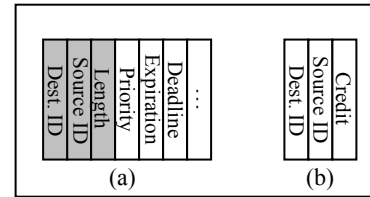


Figure 4: Credit request (a) and reply (b) messages.
The request message may include optional fields that describe the matching data packet.

4.2 Implementation

The source control logic is embedded in the network interfaces that connect cores to the NoC infrastructure: Sources capable of communicating with potential hotspots are equipped with logic that stores current quota, generates quota requests and handles incoming quota replies. In order to keep track of available credit, the source interface includes a *credit status table* (CST), with an entry for each potential hotspot module. If all potential hotspots are known during design time, the entries can be pre-coded in hardware. Otherwise, these numbers can be programmed as part of the configuration process.

The CST is updated by the interface control logic upon receiving credit reply packets and upon injecting a packet: Source module interfaces are modified so that data packets are no longer injected towards potential hotspots as soon as link-level flow-control allows it. Instead, the source control logic looks up the CST using the destination ID. If an entry with a matching module ID does not exist, the destination is not a potential hotspot and the data packet can be injected into the network immediately. Otherwise, the current credit status is retrieved and compared with the size of the data packet. If sufficient quota exists, the packet is injected into the network and its size is subtracted from the corresponding CST entry, reflecting the consumed credit. Otherwise, a request packet is generated applying for the missing credit. Meanwhile, the data packet waits in the source network interface buffers until a credit reply arrives.

Potential hotspot modules are equipped with an allocation controller that receives credit request messages, decides upon service order and sends credit reply packets. This scheduling logic can be implemented as part of a hotspot network interface, as an independent module, or as a unit serving multiple hotspot modules. In this work, we assume that the scheduler is embedded within potential hotspot interfaces.

The implementation of the scheduler unit includes a *pending requests table* (PRT), with an entry for each source module. The entry fields are selected during design time, according to specific system needs. For example, a simple system may only need the source number and length fields, while other advanced designs

may also describe request priority, deadline and expiration values. When receiving a credit request packet, the scheduler control logic decodes the request and logs it in its PRT. In order to select a source to be serviced, a local arbiter examines the PRT and chooses a module, subject to QoS definitions. In order to grant the selected source permission to access the hotspot module, the scheduler control logic encodes a data reply packet carrying adequate credit and sends it to the appropriate source.

5. NUMERICAL RESULTS

In this section, the performance of the suggested hotspot allocation mechanism is examined by means of simulation. Results are compared to a wormhole based NoC with no such hotspot resolution mechanism. The presented results quantify the extent to which the allocation scheme solves hotspot effects (system performance degradation and the source fairness problem).

The following evaluation model is used:

1. The system consists of 16 modules, arranged in 4×4 grid with a single hotspot module, placed at the upper-leftmost corner. Fixed, deadlock-free YX routing is employed.
2. All network links and non-hotspot modules have identical capacity (10 Gbit/sec).
3. The hotspot module has only 1Gbit/sec capacity.
4. Data packets are 200 flits long and are generated by a Poisson process; Flits are 16 bits long.
5. Routers have a 10-flit input queue per port.
6. All possible non-HS flows exist in the system and have identical characteristics. Similarly, all possible HS flows exist and have identical characteristics.
7. A prioritized virtual channel is used to deliver control packets, which are two flits long each.
8. Routers resolve contention for output port in a round-robin manner.
9. The hotspot scheduler, which is implemented as part of the hotspot network interface, employs round-robin arbitration among pending requests.

In general, each set of results has been obtained for a fixed non-HS traffic, which serves as background communication, and for varying hotspot load on a system similar to the one illustrated in Figure 2. The term "end-to-end latency" in this paper refers to the time elapsed since the packet is created in the source until the last flit is consumed by the destination. Therefore, the measured latency accounts for source queuing, network blocking, virtual channel multiplexing, link bandwidth and for the end-to-end protocol overhead.

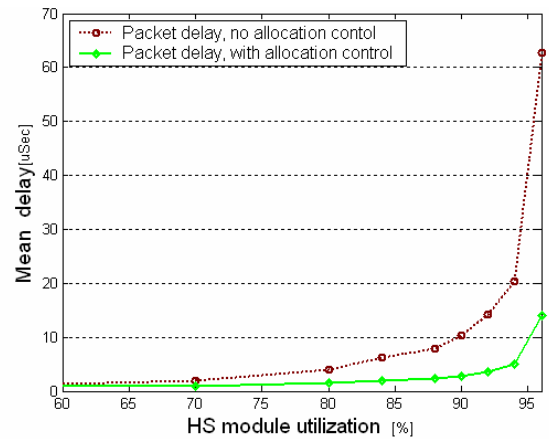
The results were generated using the OPNET based simulator [19], modeling a wormhole network at the flit level. The model accounts for all network layer characteristics, including wormhole flow-control, virtual channels, routing, finite router buffers and link capacities.

5.1 System Performance

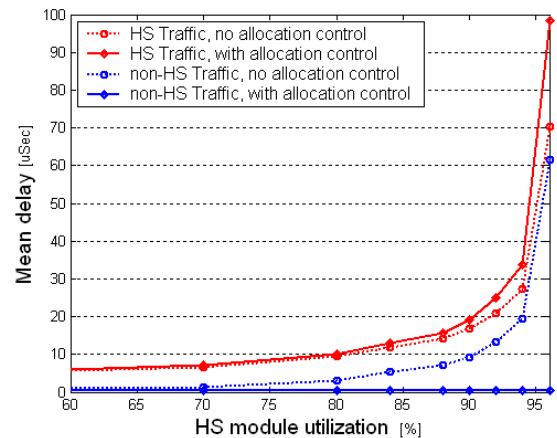
Figure 5(a) shows mean end-to-end delay in the system, with and without the allocation protocol. It is clear that the allocation mechanism considerably reduces the average access latency. Figure 5(b) breaks the results down, separating HS-traffic from

non-HS traffic. Due to the bandwidth consumed by the control packets, the mean delay of the HS traffic is slightly increased when using the proposed mechanism. However, there is a dramatic improvement in the delay of the background traffic, which is now largely unaffected by the mounting hotspot load.

The small increase in HS traffic delay due to the load of control messages can be further circumvented. The designer can prevent control messages from consuming the limited hotspot bandwidth by placing the scheduler in a location such that the control path does not conflict with the hotspot data path. For example, if the hotspot module is a bandwidth limited off-chip DRAM, the hotspot controller can be placed on-chip (e.g., as part of the network interface), meaning that control messages are never transmitted on the slow off-chip bus to DRAM. Note that in the absence of the resource allocation mechanism, the mean hotspot access latency may be misleading at high loads. When congestion is high, the limited hotspot bandwidth is not fairly divided between modules. In particular, some modules systematically experience a delay which is considerably higher than the mean delay. This does not happen in a resource controlled network, as demonstrated below (section 5.2).



(a)



(b)

Figure 5: Mean HS (Hotspot) and non-HS traffic End-to-End Delay vs. Hotspot Load

5.2 Source Fairness

Figure 6(a) demonstrates the severity of the source fairness problem in a wormhole network in steady-state. When the hotspot maximal utilization is approached, the delays vary largely between sources. While modules close to the hotspot experience only a slight increase in their end-to-end delay (e.g. module 1 and 5), the delay seen by distant modules is hundreds of times larger. This unfairness increases as the number of SoC modules grows. Figure 6(b) shows the results of activating the hotspot allocation mechanism in a system with the same loads. The fair arbitration scheme manages to distribute the limited resource almost equally among the system modules.

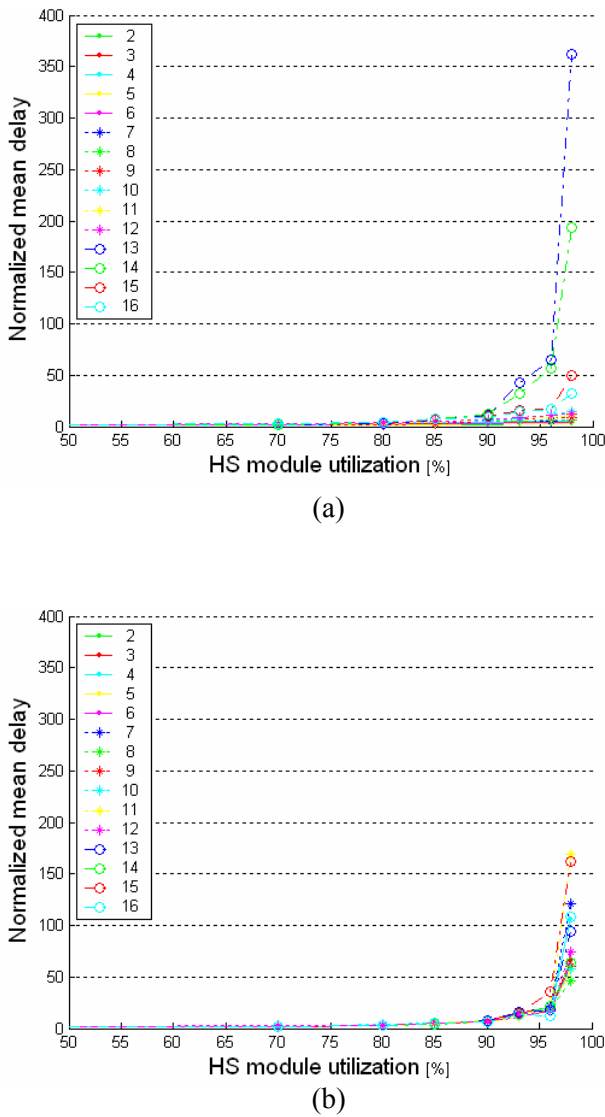


Figure 6: Mean Source End-to-End Delays vs. Hotspot Load. Each curve represents the mean delay experienced by a single source, normalized by the zero-load delay. (a) uncontrolled network; (b) controlled network.

An additional important performance metric under heavy load is the *saturation throughput*: assuming that all sources always have data to send to the hotspot module, saturation throughput is the bandwidth each source achieves. This predicts the system behavior at periods of extreme congestion in which the total load exceeds the hotspot capacity and the system operates beyond saturation point. As generally, NoC routers employ a round-robin based arbitration between ports, each router effectively divides its upstream saturation bandwidth equally among requesting ports. Therefore, in a basic wormhole based network, the further a module is from the hotspot, the less bandwidth it will get. This unfairness also increases with the number of SoC modules.

Figure 7 shows the saturation throughput with and without allocation control mechanism. When no control is applied, module 5 enjoys 25% of the hotspot limited bandwidth, while module 16 gets less than 1% due to the large number of hops on the path to the hotspot. The hotspot allocation mechanism distributes the saturation throughput fairly among source modules, even when the network is extremely loaded. This is attributed to the fact that control messages can bypass the slowly-moving data packets and do not suffer from the source fairness problems.

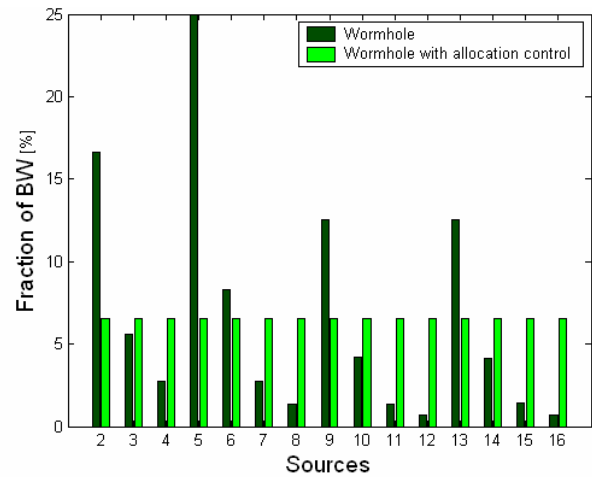


Figure 7: Saturation throughput with and without control When no control is used, distant modules only gets a small share of the bandwidth.

6. SUMMARY

The unique characteristics of wormhole routing make it particularly suitable for high-performance networks-on-chip but also highly vulnerable to destination modules hotspots, which may immediately extend system wide.

In this work, we have examined the effects of hotspot congestion caused by a highly utilized module. Two main problems were identified: A source fairness problem, caused by grid topology and router local arbitration policy; and degradation of the entire system performance, as non-hotspot traffic is obstructed during hotspot congestion. In order to solve both problems, we suggest a novel end-to-end hotspot allocation mechanism. Short control messages, delivered over a prioritized virtual-channel, are used to

arbitrate access to the hotspot module, thus significantly reducing packet blocking probability and guaranteeing fairness. The protocol, which is transparent to the system functional units, is implemented without any modification in the network routers, allowing them to be fast, small and simple. Simple control logic is added to source module interfaces, and potential hotspot modules are enhanced by a simple controller, customized to match system needs.

Simulation results show that the proposed technique successfully solves the source fairness problem and achieves a significant improvement in the delay of crossing traffic. In light of these results, and since SoCs often include a popular, bandwidth limited resource, we believe that hotspot allocation control is an essential supplement for any wormhole-based NoC architecture.

7. REFERENCES

- [1] W.J. Dally and C. Seitz, "The Torus Routing Chip", *Distributed Computing*, vol. 1, no. 3, 1986.
- [2] K. Goossens, J. Dielissen and A. Radulescu, "AETHEReal Network on Chip: Concepts, Architectures, and Implementations", *IEEE Design and Test of Computers*, September/October, 2005.
- [3] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "QNoC: QoS Architecture and Design Process for Network on Chip", *Journal of Systems Architecture*, Volume 50, February 2004.
- [4] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip", *Integration, the VLSI Journal*, Oct. 2004.
- [5] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip", *Circuits and Systems Magazine*, IEEE Volume 4, Issue 2, 2004.
- [6] W. Dally, "Virtual Channels Flow Control", *Proc. ISCA*, May 1990.
- [7] Z. Guz, I. Walter, E. Bolotin, I. Cidon, A. Kolodny and R. Ginosar, "Efficient Link Capacity and QoS Design for Wormhole Network-on-Chip", *DATE* 2006.
- [8] G. F. Pfister and V. A. Norton, "Hot Spot contention and combining in multistage interconnection networks", *IEEE Trans. Comp.*, vol. C-34, no. 10, Oct. 1985.
- [9] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo, "A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks", *High-Performance Computer Architecture (HPCA) 2005 Proceedings*.
- [10] E. Baydal, P. Lopez and J. Duato, "A Congestion Control Mechanism for Wormhole Networks", *Ninth Euromicro Workshop on Parallel and Distributed Processing (PDP '01) Proceedings*.
- [11] A. Smai and L. Thorelli, "Global Reactive Congestion Control in Multicomputer Networks", *In 5th International Conference on High Performance Computing*, 1998.
- [12] W. S. Ho and D. L. Eager, "A Novel Strategy for Controlling Hot-spot Congestion", *Proc. 1989 Int'l Conf. Parallel Processing Proceedings*.
- [13] T. Lang and L. Kurisaki, "Nonuniform Traffic Spots (NUTS) in Multistage Interconnection Networks", *Journal of Parallel and Distributed Computing*, 1990.
- [14] P. Gawghan and S. Yalamanchi, "Adaptive Routing Protocols for Hypercube Interconnection Networks", *IEEE Transactions on Computers*, May 1993.
- [15] S. W. Moon, J. Rexford and K. G. Shin, "Scalable Hardware Priority Queue Architectures for High-Speed Packet Switches", *IEEE Transactions on Computers*, November, 2000.
- [16] K. H. Yum, E. J. Kim, C. R. Das, M. Yousif and J. Duato, "Integrated Admission and Congestion Control for QoS Support in Clusters," *IEEE International Conference on Cluster Computing (CLUSTER'02)*, 2002.
- [17] V. Shurbanov, D. R. Avresky, P. Mehra and W. J. Watson, "Flow Control in ServerNet Clusters", *Euro-Par* 2000.
- [18] A. Radulescu, J. Dielissen, S. G. Pestana, O. Gangwal, E. Rijpkema, P. Wielage and K. Goossens, "An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Programming", *IEEE Transactions on CAD of Integrated Circuits and Systems*, January 2005.
- [19] OPNET Modeler, www.opnet.com