Access Regulation to Hot-Modules in Wormhole NoCs

Isask'har Walter[1], Israel Cidon[2], Ran Ginosar[2], Avinoam Kolodny[2]
Electrical Engineering Department, Technion – Israel Institute of Technology, Haifa 32000, Israel
[1]zigi@tx.technion.ac.il , [2]{cidon, ran, kolodny}@ee.technion.ac.il

Abstract

Network on Chip (NoC) may be the primary interconnect mechanism for future Systems-on-Chip (SoC). Real-life SoCs typically include *hot-modules* such as DRAM controller or floating point unit, which are bandwidth limited and in high demand by other units. In this paper we demonstrate that the mere existence of one or more hot-modules in a wormhole-based NoC dramatically reduces network efficiency and causes an unfair allocation of system resources. We demonstrate that a single hot-module destroys the performance of the entire SoC, even if network resources are over-provisioned. In order to resolve the hot-module effect, we introduce a novel low-cost credit based distributed access regulation technique that fairly allocates access rights to the hot-module. Unlike other methods, this technique directly addresses the root cause of network buffer congestion phenomena. Using simulation, we show the effectiveness of the suggested mechanism in various NoC scenarios.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Design

Keywords

Network on-Chip, wormhole, hotspot, resource management, SoC

1. Introduction

Wormhole switching [1] is commonly employed in NoC (e.g. [2]-[5]), due 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 transmitted packets to be segmented and "spread" along the path between the source and destination in a pipeline fashion. The main drawback of wormhole switching is its sensitivity to packet blocking that may quickly consume buffers along the entire path. Therefore, the common design practice of high performance wormhole networks is to allocate ample link capacities to achieve low utilization and to employ multiple virtual channels [6]. In addition, in order to support a mix of data flows with different timing criticality, mechanisms to support Quality-of-Service requirements should be included. For example, In QNoC [3] packet priorities are supported by assigning priorities to virtual channels and defining service levels for messages according to their relative urgency (e.g. interrupts, real time streaming or cache line fills, cache prefetch and large data blocks). The network is equipped with enough resources (capacities, buffers) to deliver adequate throughput at the required latency for each service level.

The above QNoC design methodology [7] works properly as long as all system modules consume messages within their specified capacity. However, at certain times the aggregated traffic demand might exceed a destination module's bandwidth. Similarly, such a module may occasionally operate at a slower than average speed (e.g., a variable speed coder, encoder or storage device) and become congested coincidently or not with an incidental usage peak. We term such a bandwidth-limited high-demanded SoC module a *hot-module* (HM). In such situations, the hot-module is unable to consume incoming packets fast enough. Hot-modules are common in real-life SoCs, as some modules are bandwidth limited and in high demand by other units. Hot-modules may be external (e.g., DRAM [8]-[10]) or internal components (caches, CAMs, specialized arithmetic units, special purpose processors, SRAMs [9]). The identities of the hot-modules are usually known in advance as the critical resources affecting system performance. Moreover, it is likely that such modules remain HMs even in SoCs with multiple use-cases (e.g., external memory bottleneck in [10]).

Congested modules exist in systems with any communication infrastructure (including bus-based communication), but wormhole-based systems are much more sensitive to hot-modules, as the entire network may be affected: The hop-by-hop backpressure, associated with wormhole routing, causes buffers at the router adjacent to the hot-module 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* [11] with the hot-module as its root (Figure 1). Moreover, the domino effect stretches beyond the traffic that is destined to that destination (the saturation tree) as packets that are

Proceedings of the First International Symposium on Networks-on-Chip (NOCS'07)
0-7695-2773-6/07 $20.00 © 2007 IEEE

destined at other modules find no free buffers at certain routers on their route (extending the saturation tree to a larger saturated acyclic graph). The overall NoC system suffers increased delays in packet delivery as well as unfair network utilization (modules near the HM get a larger portion of its resources). This threat is particularly troublesome in wormhole based architectures due to packet "stretching" across multiple hops causing the hot-module effects to extend network-wide instantly. It is very important to note that this phenomenon is *independent of links and router bandwidth*. Such a network freeze may build up even in a system with infinite capacity links because of a single heavily loaded module. Consequently, even largely over-provisioned NoCs suffer from poor performance if potential hot-modules are left unhandled.

We propose a novel one-to-many credit-based access regulation mechanism for solving the NoC buffer overflow problems in QNoC and other wormhole-based systems with predefined HMs. An HM allocation controller is introduced to arbitrate short, high priority credit requests. The controller allows the system architect to regulate hot-module access according to the quality of service requirements of the specific system application. The allocation algorithm employed by the controller is system-specific, since the HM is independent of the network. Credit requests and grants are transmitted as small high-priority signaling packets (grants and requests may be also piggybacked on other messages). In order to eliminate a potential round-trip latency in selected modules, auto-refresh or pre-allocation is used. The access regulation mechanism is implemented in module interfaces and in an appropriate location (e.g., as part of the HM network interface), while NoC routers remain unchanged. The mechanism prevents the accumulation of packets destined at a hot-module within the network buffers. Consequently, other traffic remains unaffected even when the HM load increases significantly.

The rest of this paper is organized as follows: In Section 2, the negative effects of HMs in wormhole-based NoCs are discussed. Related work is surveyed in Section 3. In Section 4, a specific credit allocation technique is proposed to allow fair sharing of the hot resource and to mitigate effects on non-HM traffic (traffic not destined at the HM), and Section 5 presents simulations of the suggested mechanism.

2. Hot-module effects

The NoC buffer congestion due to HMs has several negative effects on system performance. The hot-module *access latency* is increased, as packets destined at it contend for the limited HM bandwidth. Unfortunately, additional significant fairness problem arises. Typically, different source modules are at different NoC distances from the HM (as illustrated in Figure 2a). Since a packet has to win local output port arbitration in each router along its path, the HM bandwidth is not fairly shared. Namely, the sharing of the HM capacity is dictated by multiple local decisions made by the network components, and not by system requirements.

More precisely, NoC modules close to the HM enjoy a much larger share of the HM bandwidth than distant ones. This is caused by the fact that NoC routers typically employ a locally fair, round-robin arbitration between packets (or flits) of similar priority waiting at different input ports and contending for the same output port. Therefore, when its inputs are saturated, each router that is part of the HM saturation tree equally divides the bandwidth available at its upstream port among its input ports. Consequently, HM throughput at a source drops exponentially as a function of the number of hops between the source and the HM. When the HM demand is close to its capacity, location and distance diversity also lead to significant differences in access latency. Packets sent by distant sources are more likely to be blocked by other HM-traffic (i.e., traffic destined at a HM) in comparison to packets that travel only short distances. Therefore, modules that are located relatively far from the hot-module experience extremely long access times when HM load mounts. These issues (HM saturation throughput and HM

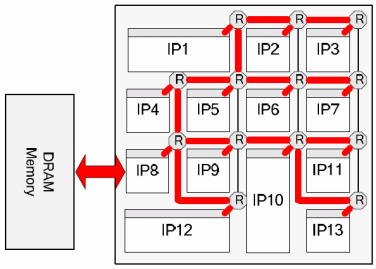
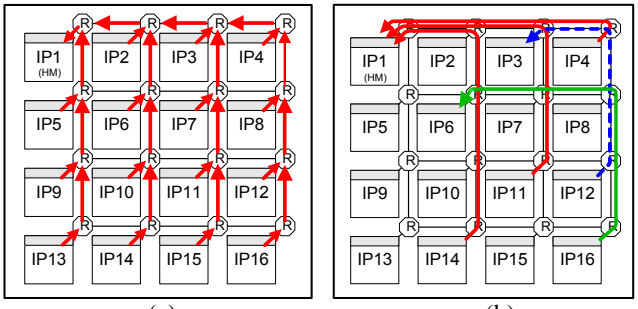


Figure 1: External DRAM as a SOC's hot-module, which causes a saturation tree in the NoC (highlighted links)



(a) (b)

Figure 2: Hot-module effects in a 4×4 YX routed NoC

(a) Source Unfairness**:** on its way to the hot-module (IP1), packets generated by module 12 have to win 6 arbitrations, while module 5 packets have to win only 2.

(b) HM-traffic obstructing non-HM traffic: flow 4→1 slows-down (or blocks) flow 12→3 (which shares a link), and in turn may affect flow 16→6, which is destined at an idle module and does not share any link or router output port with HM-traffic.

access latency) will be referred to as the *source fairness* problem.

Furthermore, performance degradation due to HM load is not restricted to the HM-traffic itself. In typical NoCs, HM and non HM-traffic compete for the same network buffer space and router ports. Therefore, HMs that slowly consume incoming data hinder the delivery of non-HM packets (Figure 2b), as blocked HM packets wait inside the network occupying expensive buffers. As a result, packets destined to lightly loaded modules are also being stalled by the network, suffering delays and fairness problems similar to HM packets (Section 5).

The above discussion applies to any network in the presence of congested end-points. However, left unhandled, hot-module effects in a wormhole network are more severe than in a store-and-forward network, as packets are blocked across multiple routers and buffering space is limited.

It is important to note that these delay and fairness effects are symptoms of HMs presence and not of an inadequately provisioned NoC. In fact, a wormhole NoC would suffer from the presence of a HM even with links and routers of infinite capacity.

3. Related work

The negative effects of hot-modules were partially explored in off-chip interconnection networks (e.g. [11]-[17]). In that literature there is no clear distinction between the issue of HM and the congestion of a network port. Typically, suggested solutions attempt to prevent regular traffic from being affected by the traffic of a saturation tree, either by not allowing one to form or by allocating hot traffic exclusive network resources. Unfortunately, such solutions do not bring a fair allocation of the hot resource. In addition, these solutions address multi-computer networks, in which the design considerations are significantly different from those of NoCs. For example, some works modify the network routers in order to throttle packet injection at high loads (e.g., [13], [14]), discard packets (e.g., [15]), deflect packets away from loaded locations (e.g. [16], [17]), use separate buffers for traffic destined at a hot-module (e.g. [12]), or simply use a large number of virtual channels. However, when directly applied to NoCs, such modifications considerably increase NoC router gate count and memory size, resulting in excessive area and power consumption and reduced speed. For example, NoCs typically employ static shortest path routing based on a simple routing function, because of on-chip cost and performance considerations. Note that most of the previous techniques can either slightly postpone the effect of HM as they only increase the number of buffers used by non-HM traffic (by adding buffers or routing non-HM traffic away) or even increase the effect by throttling non-HM packets.

Recently, two papers have studied related but different congestion problems in NoCs. Ref. [18] addresses the classic *flow control* problem, regulating the communication between a source-destination pair. The authors combine software and hardware mechanisms to adjust the length of a period ("send window") in which the source is allowed to inject packets towards a destination. Consequently, no sense of fair sharing of the hot-module is provided. In addition, this scheme only responds after a saturation tree begins to form. In [19], an input regulation scheme is described where each router predicts the availability of buffers in its input ports, according to data collected from its neighbors. When a source observes that its adjacent router is expected to run out of buffers, it delays generation of new packets. However, that technique does not prevent hot-traffic from monopolizing multiple virtual channels and thus might prevent injection of other packets towards idling destinations. Moreover, as other classic end-to-end flow and admission control methods [20], that method does not address the hot-module allocation fairness problem, since routers and sources only have local knowledge regarding the hot-module demand.

The proposed HM access regulation mechanism is considerably different from traditional end-to-end flow control mechanisms. Flow control is conducted on a per source-destination pair basis (e.g. TCP, send window in [18], static window in [21]), and prevents overflow in the destination buffers pre-allocated for this source (e.g. [22], [23]). Flow control does not directly address the hogging of network resources and does not address the problem of fair allocation of scarce resources. In addition, all existing schemes require at least one destination buffer per potential source, which is inappropriate in on-chip NoCs.

4. HM access regulation

In order to reduce the dramatic effects hot-modules have on a wormhole-based NoC (Section 5), a credit-based access regulation mechanism is suggested: each source owns a quota that limits the number of flits it can send towards a HM. When a source quota is exhausted, it can resume transmission only after being granted an additional credit. Consequently, packets that cannot be consumed by the HM do not wait inside the network, a saturation tree can not form and traffic not destined at the HM remains unaffected during congested periods.

Two types of control messages regulate the access to a HM: if a source has insufficient credit to start delivery of a data packet to a HM, it sends a *credit request* packet to a HM *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 therefore cannot be held back in the NoC by data packets. In this work, we assume

that the NoC is equipped with a prioritized virtual channel mechanism, such as the one described for QNoC in [3], guaranteeing fast access of control messages regardless of data traffic loads.
As control packets are a few flits long and a single control packet credits a sizable chunk of data, control traffic is a small percentage of the HM-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 during light load periods, source quota is slowly self refreshing. As a result, credit request and reply are not time critical events.

4.1 Control Messages

Using a credit request message, a source describes the data packet(s) it wishes to send to the hot-module and asks for credit to do so. In addition to the *Destination ID* field of a regular 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 HM allocation 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 allocation controller sort the requests in the best servicing order, postponing less urgent requests to be serviced last. If requests can be ignored unless they are served by a certain time (e.g., speculative cache fetches), an expiration field may be used.
Figure 3a illustrates an example of a credit request packet in which each field fits a flit (more fields per flit are of-course possible). Figure 3b 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 hot-modules. 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, an allocation controller

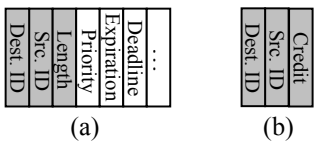


Figure 3: Credit request (a) and reply (b) messages.
The request message may include optional fields that describe the matching data packet.

may reply with a larger number in order to credit modules ahead of time during light load periods. The allocation controller 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.

4.2 Implementation

The source control logic is embedded in the network interfaces that connect cores to NoC infrastructure: sources capable of communicating with potential HMs are equipped with logic that stores current quota, generates quota requests and handles incoming quota replies. In order to keep track of the available credit, the source interface includes a *credit status table* (CST), with an entry for each potential HM. If all potential HMs are known during design time, the entries can be pre-coded in hardware. Otherwise, these entries 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 HMs as soon as link-level protocol 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 hot-module 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.
The access to potential HMs is regulated using 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 the hot-module's network interface, as an independent module, or as a *separate* central unit serving multiple hot-modules. In this work, we assume that the scheduler is embedded within potential hot-module interfaces.
The implementation of the allocation controller unit (Figure 4) includes a *pending requests table* (PRT), with an entry for each source module. The entry fields are selected during design time, according to the fields of request packets and the specific system needs. For example, a simple system may only need the source ID and length fields, while other designs may also describe request type, priority, deadline and expiration values. When receiving a credit request packet, the scheduler control logic decodes the request and logs it in its PRT (Figure 5). In addition the allocation controller may be provided with the status of the HM, its current speed and its current queued tasks. The

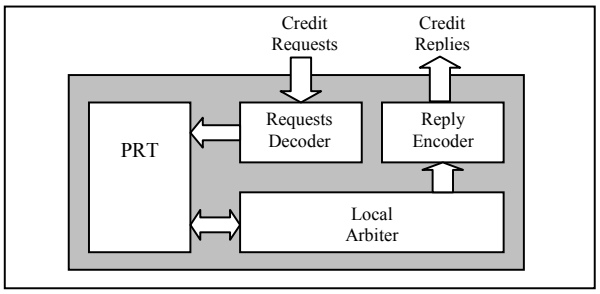


Figure 4: Hot-Module Allocation Controller.

```
Upon receiving request for K credits from module i
   If HM idle
      Send k credits to module i
   Else
      Log request in PRT

Upon finishing servicing a packet
   /*use local arbiter and PRT to choose
     next module to be served*/
   i←local_arbiter(PRT)
   /*extract requested credit from PRT
   k←PRT(i)
Send k credits to module i
Remove request from PRT
```

Figure 5: Operation of a simple HM allocation controller.

local arbiter examines the PRT as well as the HM status and chooses a module, subject to QoS, fairness definitions and the HM status and encodes a credit reply packet carrying a calculated amount of credit and sends it to a selected source.

The scheduling algorithm, which is crucial to the success of the suggested technique, allows the system architect to adequately share the hot resource among requesting modules during high load periods according to the system's needs. In order to keep the cost of the HM allocation controller hardware minimal, in this work we assume that a simple, round robin local arbiter is implemented and demonstrate its effectiveness. The design of more complex controllers and schedulers is left for future work.

It should be noted that the access regulation mechanism is optional and transparent to the HM and source modules. In particular, the system architect may allow some modules to access potential HMs without requesting credit at all, if their traffic should not be delayed by the controller under no circumstances.

5. Performance evaluation

In this section, the performance of the suggested HM access regulation mechanism is examined by means of simulation. Results are compared to a "standard" wormhole based NoC with no such mechanism. Two scenarios are used: a "Classic" hotspot traffic pattern and a real-life MPEG decoder SoC. The presented results exemplify the severity of the HM effects (system performance degradation and the source fairness problem) and quantify the extent to which the allocation scheme solves them.

The term "end-to-end latency" in this paper refers to the time elapsed since the packet is created at the source until its last flit is consumed by the destination. Therefore, the measured latency accounts for source queuing, network blocking, virtual channel multiplexing, link bandwidth limitations, and overhead of the access regulation protocol. The results are generated using the OPNET based simulator [24], modeling a wormhole network at the flit level. The model includes all network layer components, including wormhole flow control, virtual channels, routing, finite router buffers and link capacities.

5.1 Hotspot traffic

Traditionally, congestion alleviation techniques in off-chip networks are evaluated using an "all-to-one" traffic pattern. Although not typical for SoCs, this synthetic scenario is analyzed here in order to clearly demonstrate the effects of hot-module congestion and resource arbitration.

Each set of results has been obtained for fixed non HM-traffic, which serves as background communication, and for varying HM load in a system similar to the one illustrated in Figure 2.

The following evaluation model is used:

1. A QNoC system consists of 16 modules, arranged in a 4×4 grid with a single HM, placed at the upper-leftmost corner. Fixed, symmetric XY routing [3] is employed.
2. All network links and modules (except HM) have identical capacities (10 Gbit/sec).
3. The HM has 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-HM flows exist in the system and have identical characteristics. Similarly, all possible HM 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 ports in a round-robin manner.
9. The allocation controller is implemented as part of the network interface of the HM and employs round-robin arbitration among pending requests.

5.1.1 System Performance

Figure 6a shows mean end-to-end delay in the system, with and without the allocation protocol. It is clear that the access regulation mechanism considerably reduces the average access latency. Figure 6b breaks the results down,

separating HM-traffic from non HM-traffic. Due to the bandwidth consumed by the control packets, the mean delay of the HM-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 almost unaffected by the mounting HM load. This improvement is caused by the fact that HM-traffic no longer occupies expensive network buffers, and non-HM packets can use them effectively to reach their destinations. The small increase in HM-traffic delay due to the load of control messages can be further circumvented. The designer can prevent control messages from consuming the limited HM bandwidth by placing the controller in a location such that the control path does not conflict with the HM data path.

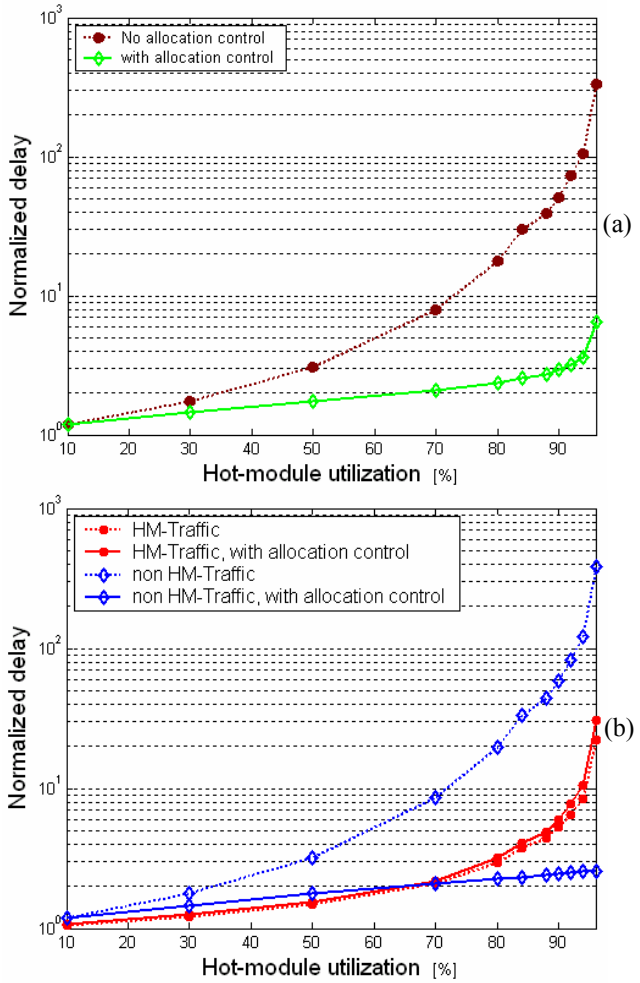


Figure 6: Mean end-to-end delay vs. HM load.

5.1.2 Source fairness

Figure 7 demonstrates the source fairness problem in an uncontrolled wormhole network in steady-state. When the HM maximal utilization is approached, the delays vary largely among sources. While modules close to the HM experience only a slight increase in their end-to-end delay (e.g. module 5), the delay seen by distant modules (e.g. module 16) is considerably larger. This unfairness, caused by the different number of arbitration points along the path (Section 2), increases as the number of SoC modules grows. Figure 7 also shows the results of activating the HM 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 (including ones not shown) as the system approaches its maximal steady state load. As described above, other fairness criteria can be implemented using different HM allocation controller policies.

An additional important performance metric under heavy load is the *saturation throughput*: assuming that all sources have data to send to the hot-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 HM capacity and the system operates at saturation point. As NoC routers employ a round-robin based arbitration among ports, each router effectively divides its upstream saturation bandwidth equally among requesting ports. Therefore, in a basic grid based network, the further a module is from the HM, the less bandwidth it will get. This unfairness also increases with the number of SoC modules.

Figure 8 shows the saturation throughput with and without access regulation mechanism. When no control is applied, module 5 enjoys 25% of the HM limited bandwidth, while module 12 gets less than 1% due to the large number of hops on the path to the HM. This is explained by the network topology (Figure 2a): router R1 (i.e., the router directly connected to IP1) equally divides the HM capacity between its east and south ports. Similarly, router R5 distributes it upstream HM capacity (available at its north port) among its local and south ports, meaning that IP5 enjoys exactly a quarter of the HM bandwidth when all sources are saturated. Following similar analysis, it can be easily shown that IP12 only gets 1/144 of the HM capacity, as its packets contend four times with packets from another input port (routers R12, R8, R4, R1) and two times with packets from two other input ports (routers R3, R2).

The HM access regulation distributes the saturation bandwidth fairly among all source modules, even when the network is extremely loaded. This is attributed to the fact that control messages bypass the slowly-moving data packets and do not suffer from the source fairness problems.

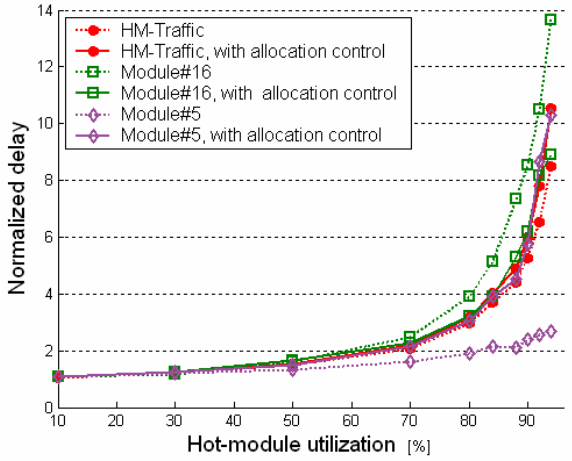


Figure 7: Mean end-to-end delays vs. HM load.
Curves represent mean delay normalized by the zero-load delay.

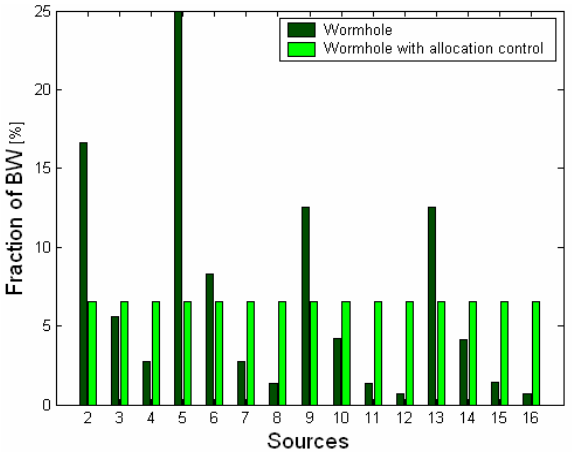


Figure 8: Saturation throughput.

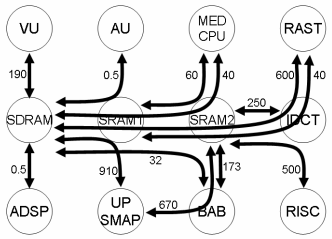


Figure 9: MPEG-4 communication demands.
The amounts specify the average traffic [MB/s]

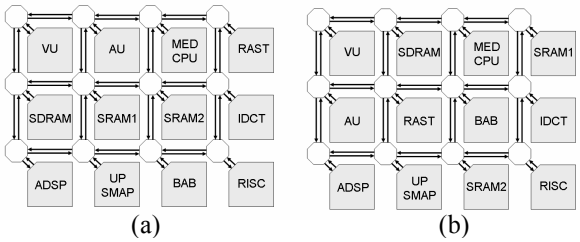


Figure 10: MPEG-4 SoC placement.
Basic (a) vs. optimized (b) placement

5.2 MPEG-4 decoder

In this test case, we use an MPEG-4 video decoder of [9] to evaluate the performance of the access regulation. The SoC is composed of 12 processing elements placed on a 3x4 grid. By analyzing the communication demands (Figure 9), it is clear that two modules are at high demand by multiple sources: The DRAM controller has 7 incoming flows (accounting for 25% of the total traffic), and the SRAM2 module with 4 incoming flows (22% of the total traffic).
In order to evaluate the system's performance, we use a model similar to the one described in Section 5.1, with the MPEG-4 video decoder communication demands [9]. The DRAM controller and SRAM2 modules are equipped with allocation controllers. In order to create varying HM load, the HM bandwidth is adjusted.
Two module mappings are used: In the first mapping (
Figure 10a), the two hot-modules are placed in relative proximity to each other, in a way that causes some of the packets destined at those modules to contend for the same inter-router links. In the second mapping (
Figure 10b), the placement is optimized so that no such sharing takes place but routing paths are kept short. This placement minimizes the effect each HM has on the other.

5.2.1 System Performance

As in the previous example, we first examine the system performance in steady state while increasing the HMs utilization. Figure 11 shows the overall delays in the system using the basic placement, with and without allocation control. When the uncontrolled HMs utilization increases, traffic destined to other modules suffers delays which are more than a hundred times larger than their zero load latency. Activating the allocation controller mechanism frees expensive network buffers, thus allowing the non-HM packets to arrive at their destination with considerably smaller delays. Note that the small increase in the non HM-traffic delay is imminent as it reflects the growing usage of the network resources by HM packets. Figure 12 shows similar effects when the optimized module placement is used. Without control, non HM-traffic suffers of extremely high delays. The latency is reduced by an order of magnitude when the allocation controller is introduced, without any noticeable increase in the latency of the HM-traffic (the two HM-traffic curves in Figure 12b overlap).

5.2.2 Source Fairness

Figure 13 shows the saturation throughput in the MPEG-4 system. As explained above (Section 2), when no control is used, distant modules only get a small fraction of the scarce resource. For example, the share of SDRAM bandwidth that module 00 (VU) gets in the optimized system is more than six times larger than the one of module 20 (ADSP) and of module 21 (UP SMAP).

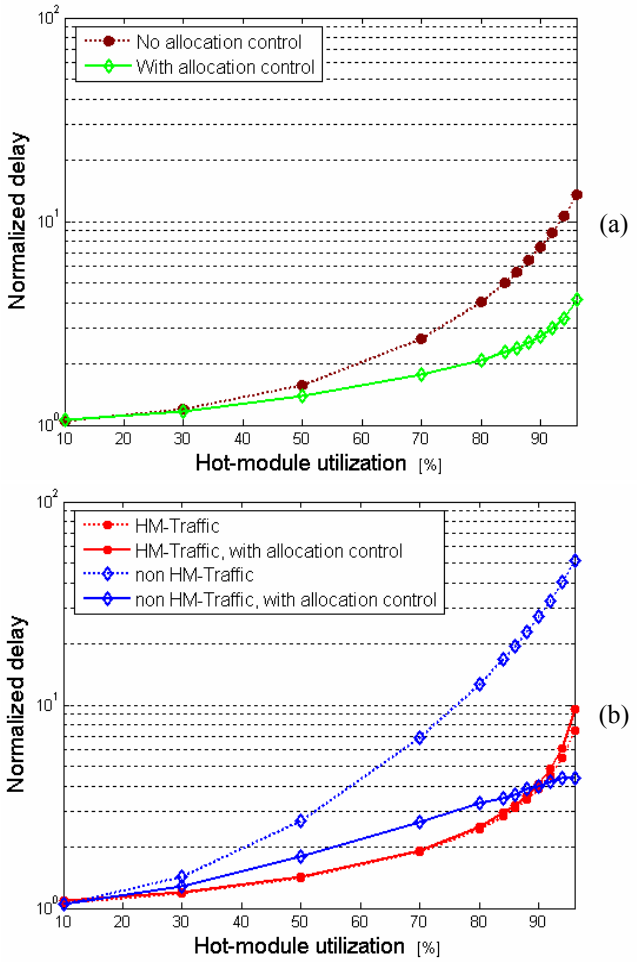


Figure 11: Mean end-to-end delay vs. hot-modules load.
(Basic placement)

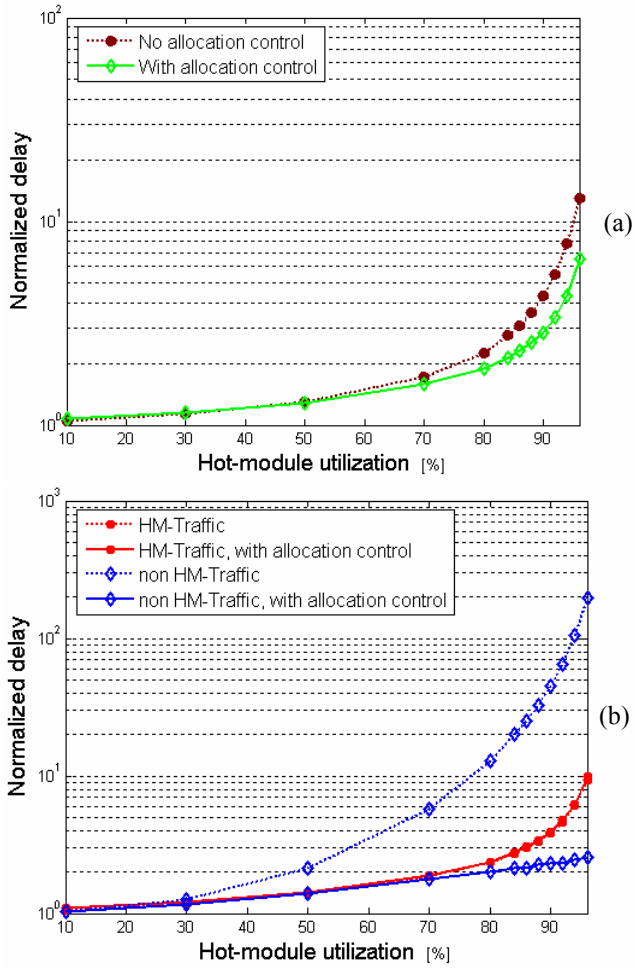


Figure 12: Mean end-to-end delay vs. hot-modules load.
(Optimized placement)

Figure 13a reveals an additional peril in uncontrolled systems: Although HM demand is maximal, the expensive hot-modules are idling 40% of the time. This happens since packets destined at the different hot-modules compete for the same network resources and block each other from making progress in the network. As the hot-modules in the optimized system are placed so that no such sharing happens, the hot-modules are fully utilized. Unfortunately, optimal placement is not always feasible due to layout and timing constraints. HM access is successfully regulated using the control mechanism, and each source gets a fair share of each congested module (Figure 13b). The control packets consume less than 2% of the HM bandwidth.

6. Summary

The unique characteristics of wormhole routing make it particularly suitable for high-performance NoCs. However, it is also highly vulnerable to loaded hot modules. Due to wormhole's backpressure mechanism, NoC buffer depletion effects extend system wide instantaneously.

Two main problems were identified: the source fairness problem, and the degradation of the entire system performance, as non HM-traffic is also blocked during HM congestion. If HMs are left unhandled, system performance is determined by network topology and routers' local arbitration policy, instead of following system optimization goals. The main thrust of this paper is that system behavior should be controlled explicitly by the architect rather than by network side-effects. The network should include mechanisms to facilitate such explicit control. In order to solve both the fairness and the system performance problems, we have presented a low-cost end-to-end access regulation mechanism. Short control messages are used to arbitrate access to the HM, thus significantly reducing packet blocking probability and achieving fairness. The protocol, which is transparent to the system functional units and to the NoC, is implemented without modifying the network routers, allowing them to be simple, and thus fast, small and efficient. The protocol exploits a high-priority service level which is readily available in the NoC for fast signaling. Therefore, there is no overhead at the network

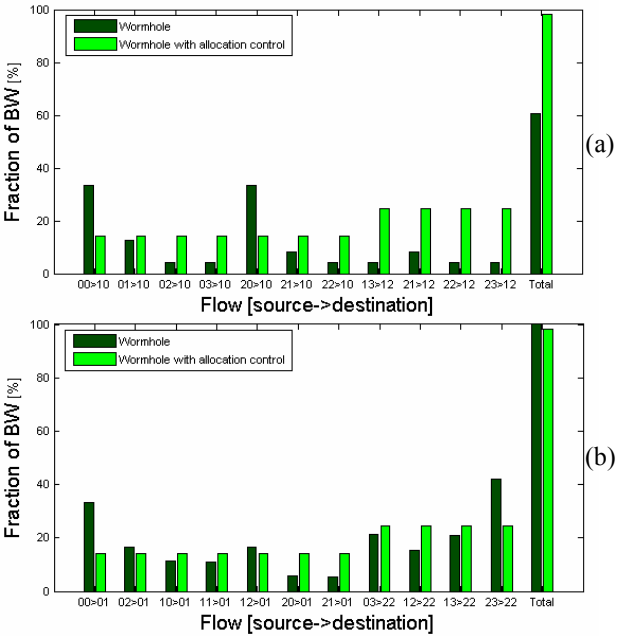


Figure 13: Saturation throughput.
(a) Basic placement (b) Optimized placement

layer. Simple logic is added to the network interfaces of sources, and potential HMs are instrumented with an allocation controller, customized to system needs. We suggest HM access regulation as an essential supplement for any wormhole-based NoC.

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, 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", Proc. Design, Automation and Test in Europe (DATE), 2006.

[8] S. Dutta, R. Jensen, and A. Rieckmann, "Viper: A Multiprocessor SoC for Advanced Set-Top Box and Digital TV Systems", Design & Test of Computers, 2001.

[9] D. Bertozzi, A. Jalabert, S. Murali R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli , "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip", IEEE Transactions on Parallel and Distributed Systems, 2005.

[10] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "A Methodology for Mapping Multiple Use-Cases onto Networks on Chips", Proc. Design, Automation and Test in Europe (DATE) 2006.

[11] 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.

[12] J. Duato, I. Johnson, J. Flich, F. Naven, P. García, and T. Nachiondo, "A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks", Proc. High-Performance Computer Architecture (HPCA), 2005.

[13] E. Baydal, P. Lopez, and J. Duato, "A Congestion Control Mechanism for Wormhole Networks", In Proc. Ninth Euromicro Workshop on Parallel and Distributed (PDP '01), 2001.

[14] A. Smai and L. Thorelli, "Global Reactive Congestion Control in Multicomputer Networks", In Proc. 5th International Conference on High Performance Computing, 1998.

[15] W. S. Ho and D. L. Eager, "A Novel Strategy for Controlling Hot-spot Congestion", In Proc. lnt'l Conf. Parallel Processing, 1989.

[16] T. Lang and L. Kurisaki, "Nonuniform Traffic Spots (NUTS) in Multistage Interconnection Networks", Journal of Parallel and Distributed Computing, 1990.

[17] P. Gawghan and S. Yalamanchi, "Adaptive Routing Protocols for Hypercube Interconnection Networks", IEEE Transactions on Computers, May 1993.

[18] P. Avasare, V. Nollet, J-Y. Mignolet, D. Verkest, and H. Corporaal, "Centralized End-to-End Flow Control in a Best-Effort Network-on-Chip", In Proc. 5th ACM International Conference on Embedded Software (EMSOFT), 2005.

[19] U. Y. Ogras and R. Marculescu, "Prediction-based Flow Control for Network-on-Chip Traffic", In Proc. ACM/IEEE Design Automation Conf., 2006.

[20] K. H. Yum, E. J. Kim, C. R. Das, M. Yousif, and J. Duato, "Integrated Admission and Congestion Control for QoS Support in Clusters", In Proc. IEEE International Conference on Cluster Computing (CLUSTER'02), 2002.

[21] V. Shurbanov, D. R. Avresky, P. Mehra, and W. J. Watson, "Flow Control in ServerNet Clusters", Euro-Par 2000.

[22] 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, 2005.

[23] M. Coenen, S. Murali, A. Radulescu, K. Goossens, and G. De Micheli, "A Buffer-Sizing Algorithm for Networks on Chip using TDMA and Credit-Based End-to-End Flow Control", In Proc. International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2006.

[24] OPNET Modeler, www.opnet.com