# GARNET: A Detailed On-Chip Network Model inside a Full-System Simulator

Niket Agarwal, Tushar Krishna, Li-Shiuan Peh and Niraj K. Jha
Department of Electrical Engineering
Princeton University, Princeton, NJ, 08544
{niketa, tkrishna, peh, jha}@princeton.edu

## Abstract

*Until very recently, microprocessor designs were computation-centric. On-chip communication was frequently ignored. This was because of fast, single-cycle on-chip communication. The interconnect power was also insignificant compared to the transistor power. With uniprocessor designs providing diminishing returns and the advent of chip multiprocessors (CMPs) in mainstream systems, the on-chip network that connects different processing cores has become a critical part of the design. Transistor miniaturization has led to high global wire delay, and interconnect power comparable to transistor power. CMP design proposals can no longer ignore the interaction between the memory hierarchy and the interconnection network that connects various elements. This necessitates a detailed and accurate interconnection network model within a full-system evaluation framework. Ignoring the interconnect details might lead to inaccurate results when simulating a CMP architecture. It also becomes important to analyze the impact of interconnection network optimization techniques on full system behavior.*

*In this light, we developed a detailed cycle-accurate interconnection network model (GARNET), inside the GEMS full-system simulation framework. GARNET models a classic five-stage pipelined router with virtual channel (VC) flow control. Microarchitectural details, such as flit-level input buffers, routing logic, allocators and the crossbar switch, are modeled. GARNET, along with GEMS, provides a detailed and accurate memory system timing model. To demonstrate the importance and potential impact of GARNET, we evaluate a shared and private L2 CMP with a realistic state-of-the-art interconnection network against the original GEMS* **simple** *network. The objective of the evaluation was to figure out which configuration is better for a particular workload. We show that not modeling the interconnect in detail might lead to an incorrect outcome. We also evaluate Express Virtual Channels (EVCs), an on-chip network flow control proposal, in a full-system fashion. We show that in improving on-chip network latency-throughput, EVCs do lead to better overall system runtime, however, the impact varies widely across applications.*

## 1 Introduction

With continued transistor scaling providing chip designers with billions of transistors, architects have embraced many-core architectures to deal with increasing design complexity and power consumption [13, 14, 29]. With increasing core counts, the on-chip network becomes an integral part of future chip multiprocessor (CMP) systems. Future CMPs, with dozens to hundreds of nodes, will require a scalable and efficient on-chip communication fabric. There are several ways in which on-chip communication can affect higher-level system design. Contention delay in the network, as a result of constrained bandwidth, impacts system message arrivals. In multi-threaded applications, spin locks and other synchronization mechanisms magnify small timing variations into very different execution paths [2]. Network protocols also impact the ordering of messages. A different order of message arrival can impact the memory system behavior substantially. Especially for cache coherence protocols, protocol-level deadlocks are carefully avoided by designing networks that obey specific ordering properties among various protocol messages [8]. The manner in which the ordering is implemented in the network leads to different messages seeing different latencies and again impacts message arrivals. Communication affects not only performance, but can also be a significant consumer of system power [18].

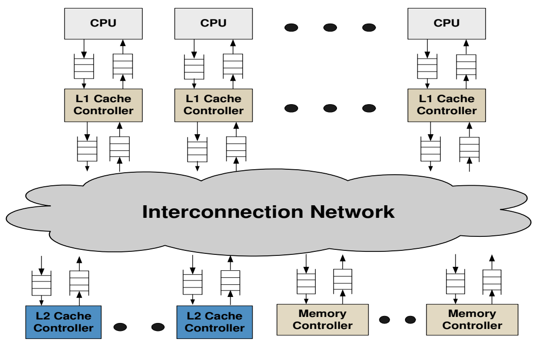


**Figure 1.** Communication in the memory system

Not only do network characteristics impact system-level behavior, the memory system also impacts network design to a huge extent. Co-designing the interconnect and the memory system provides the network with realistic traffic patterns and leads to better finetuning of network characteristics. System-level knowledge can highlight which metric (delay/throughput/power) is more important. The interconnect also needs to be aware of the specific ordering requirements of higher levels of design. Figure 1[1] shows how various components of a CMP system are coupled together. The inter-

[1]Note that the CMP system shown assumes a shared L2.

connection network is the communication backbone of the memory system. Thus, interconnection network details can no longer be ignored during memory system design.

To study the combined effect of system and interconnect design, we require a simulation infrastructure that models these aspects to a sufficient degree of detail. In most cases, it is difficult to implement a detailed and accurate model that is fast enough to run realistic workloads. Adding detailed features increases the simulation overhead and slows it down. However, there are some platforms that carefully trade off accuracy and performance to sufficiently abstract important system characteristics while still having reasonable speed of simulation on realistic workloads. One such platform is the GEMS [20] full-system simulation platform. It does a good job in capturing the detailed aspects of the processing cores, cache hierarchy, cache coherence, and memory controllers. This has led to widespread use of GEMS in the computer architecture research community. There has been a huge body of work that has used GEMS for validating research ideas. One limitation of GEMS, however, is its approximate interconnect model. The interconnection substrate in GEMS serves as a communication fabric between various cache and memory controllers. The model is basically a set of links and nodes that can be configured for various topologies with each link having a particular latency and bandwidth. For a message to traverse the network, it goes hop by hop towards the destination, stalling when there is contention for link bandwidth. This is an approximate implementation and far removed from what a state-of-art interconnection network [10,14] looks like. GEMS does not model a detailed router or a network interface. By not modeling a detailed router microarchitecture, GEMS ignores buffer contention, switch and virtual channel (VC) arbitration, realistic link contention and pipeline bubbles. The GEMS interconnect model also assumes perfect hardware multicast support in the routers. In on-chip network designs, supporting fast and low power hardware multicast is a challenge [15]. These and other limitations in the interconnect model can significantly affect the results reported by the current GEMS implementation. Also, for researchers focusing on low-level interconnection network issues, GEMS has not been adopted, with network researchers typically relying on traffic trace-driven simulation with synthetic or actual traces. In a trace-driven approach, the functional simulation is not impacted by the timing simulator and hence the timing-dependent effects are not captured. This is because the trace is generated *a priori* on a fixed system and the timing variation caused in the network does not affect the message trace. For example, if a network optimization speeds up a message, that can lead to faster computation which results in faster injection of the next message, thereby increasing injection rates. Trace-driven techniques also do not capture program variability [2] that a full-system evaluation can.

In the light of the above issues, we have developed GARNET, which is a detailed timing model of a state-of-the-art interconnection network, modeled in detail up to the micro-architecture level. A classical five-stage pipelined router [10] with virtual channel flow control is implemented. Such a router is typically used for high-bandwidth on-chip networks [14]. We describe our model in detail in Section 2. It should be noted that the original interconnect model in GEMS is better than other simulators which only model no-load network latency. We will discuss various state-of-the-art simulators in Section 6.

To demonstrate the strong correlation of the interconnection fabric and the memory system, we evaluate a shared and private L2 CMP with a realistic state-of-the-art interconnection network against the original GEMS *simple* network. We wish to evaluate which configuration (shared/private) performs better for a particular benchmark. To correctly study the effect of the network, we kept all other system parameters the same. We show that not modeling the interconnect in detail might lead to an incorrect outcome and a wrong system design choice.

To demonstrate the need and value of full-system evaluations on network architectures, we also evaluate Express Virtual Channels (EVCs) [19], a novel network flow control proposal, in a full-system simulation fashion. The original EVC paper had presented results with synthetic network traffic. Although it had provided results on scientific benchmarks, it was done in a trace-driven fashion with network-only simulation. Thus, it had failed to capture the system-level implications of the proposed technique. The trace-driven evaluation approach adopted by the authors showed the network latency/throughput/power benefits of EVCs but could not predict the full-system impact of the novel network design. We integrate EVCs into GARNET and evaluate it with scientific benchmarks on a 64-core CMP system against an interconnect with conventional virtual channel flow control [9]. Our evaluation shows the impact EVCs have in improving system performance.

We believe that the contribution of GARNET is three-fold:

- It enables system-level optimization techniques to be evaluated with a state-of-the-art interconnection network and obtain correct results.
- It enables the evaluation of novel network proposals in a full-system fashion to find out the exact implications of the technique on the entire system as opposed to only the network.
- It enables the implementation and evaluation of techniques that simultaneously use the interconnection network as well as other top-level system components, like caches, memory controller, *etc*. Such techniques are difficult to evaluate faithfully without a full-system simulator that models the interconnection network as well as other components in detail.

**Availability:** GARNET was distributed as part of the GEMS official release (version 2.1) in February 2008 and is available at http://www.cs.wisc.edu/gems/. A network-only version of GARNET can be found at http://www.princeton.edu/∼niketa/garnet, to allow network researchers to fully debug their network-on-chip (NoC) designs prior to full-system evaluations. GARNET is starting to gather a user base, and has already been cited by published articles [6, 15]. GARNET has also been plugged into an industrial simulation infrastructure [21] and is being used by industrial research groups at Intel Corp for their evaluations.

The rest of the paper is organized as follows: Section 2 describes the GARNET model in detail. Section 3 provides validation for the GARNET model. Section 4 evaluates a shared and a private CMP configuration with a state-of-the art interconnection network and compares it to the original GEMS' *simple* network model. Section 5 describes EVCs and presents full-system results. Section 6 touches upon the related work and Section 7 concludes.

## 2 GARNET

We next present details of GARNET.

### 2.1 Base GARNET model design

**State-of-the-art on-chip interconnect:** Modern state-of-the-art on-chip network designs use a modular packet-switched fabric in which network channels are shared over multiple packet flows. We model a classic five-stage state-of-the-art virtual channel router. The router can have any number of input and output ports depending on the topology and configuration. The major components, which constitute a router, are the input buffers, route computation logic, VC allocator, switch allocator and crossbar switch. Since on-chip designs need to adhere to tight power budgets and low router footprints, we model flit-level[2] buffering rather than packet-level buffering. The high bandwidth requirements of cache-coherent CMPs also demands sophisticated network designs such as credit-based VC flow control at every router [14]. A five-stage router pipeline was selected to adhere to high clock frequency network designs. Every VC has its own private buffer. The routing is dimension-ordered. Since research in providing hardware multicast support is still in progress and state-of-the art on-chip networks do not have such support, we do not model it inside the routers.

A head flit, on arriving at an input port, first gets decoded and gets buffered according to its input VC in the buffer write (BW) pipeline stage. In the same cycle, a request is sent to the route computation (RC) unit simultaneously, and the output port for this packet is calculated. The header then arbitrates for a VC corresponding to its output port in the VC allocation (VA) stage. Upon successful allocation of an output VC, it proceeds to the switch allocation (SA) stage where it arbitrates for the switch input and output ports. On winning the switch, the flit moves to the switch traversal (ST) stage, where it traverses the crossbar. This is followed by link traversal (LT) to travel to the next node. Body and tail flits follow a similar pipeline except that they do not go through RC and VA stages, instead inheriting the VC allocated by the head flit. The tail flit on leaving the router, deallocates the VC reserved by the packet.

**Router microarchitectural components:** Keeping in mind on-chip area and energy considerations, single-ported buffers and a single shared port into the crossbar from each input were designed. Separable VC and switch allocators, as proposed in [25], were modeled. This was done because these designs are fast and of low complexity, while still providing reasonable throughput, making them suitable for the high clock frequencies and tight area budgets of on-chip networks. The individual allocators are round-robin in nature.

**Interactions between memory system and GARNET:** As shown in Figure 1, the interconnection network acts as the communication backbone for the entire memory system on a CMP. The various L1 and L2 cache controllers and memory controllers communicate with each other using the interconnection network. Note that we are talking about a shared L2 system here [16]. The network interface acts as the interface between various modules and the network. On a load/store, the processor looks in the L1 cache. On a L1 cache miss, the L1 cache controller places the request in the request buffer. The network interface takes the message and breaks it into network-level units (flits) and routes it to the appropriate destinations which might be a set of L1 controllers as well as L2 controllers. The destination network interfaces combine the flits into the original request and pass it on to the controllers. The responses use the network in a similar manner for communication. Some of these messages might be on the critical path of memory accesses. A poor network design can degrade the performance of the memory system and also the overall system performance. Thus, it is very important to architect the interconnection network efficiently.

There is ongoing debate whether future CMPs will employ shared or private last-level on-chip caches [31]. While shared on-chip caches increase the overall on-chip cache capacity (by avoiding replication), it increases the L2 access time on an L1 cache miss. It is not entirely clear now as to what kind of cache organizations would be employed in future CMPs. For a shared-cache CMP architecture, as shown in Figure 1, the on-chip network is used for communication between various L1's and L2 banks and also between the L2 banks and on-chip memory controllers. In contrast, for a private cache CMP architecture, the on-chip network comes into play only for the interaction between private L2's and on-chip memory controllers. The more the communication on the on-chip network, the more it becomes critical to model it in detail. This does not, however, mean that for a private-cache CMP architecture, the on-chip network design details are not important. The interconnect is typically architected to tolerate average network traffic. Thus, during moderate to high loads, the on-chip network can experience high latencies. Similarly, if a configuration has a low bandwidth requirement in the average case, the network chosen will be of low bandwidth and will congest much earlier. Thus, modeling the details of the on-chip interconnect is important to know the exact effect of the network on the entire system.

**Point-to-point ordering:** Some coherence protocols require the network to support point-to-point ordering. This implies that if two messages are sent from a particular source node to the same destination node one after the other, the order in which they are received at the destination has to be the same in which they were sent. To provide this support, GARNET models features wherein the VC and switch allocators support system-level ordering. In the VC allocation phase, if there are contending requests from VCs to the same output port for an output VC, then requests for the packet which arrived at the router first have to be serviced first. During switch allocation, if there are contending requests for the same output port from multiple VCs from the same input port, then the request for the packet that arrived at the router first will be serviced before others. This and deterministic routing guarantee that packets with the same source and destination will not be re-ordered in the network.

**Network power:** GARNET has the *Orion* power models [30] incorporated. Various performance counters that are required for power calculations are recorded during the simulation. The performance counters keep track of the amount of *switching* at various components of the network. GARNET models per-component events (*e.g.,* when a read occurs in a router

[2]A flit is the smallest unit of flow control. A packet consists of a head flit, followed by body flits, and ends with a tail flit.

buffer), but not bit-wise switching activity. An average bit-switching activity (0.5) is assumed per component activity. The variable, SWITCHING_FACTOR, inside Orion can be used to vary this factor. The various statistics collected per router are the number of reads and writes to router buffers, the total activity at the local and global (VC and switch) arbiters and the total number of crossbar traversals. The total activity at each network link is also recorded. The total power consumption of the network is equal to the total energy consumed times the clock frequency. The total energy consumption in the network is the sum of energy consumption of all routers and links. The energy consumed inside a router, as shown in Equation 1, is the summation of the energy consumed by all components inside the router. Now, the energy of each component is the summation of the dynamic and leakage energy. The dynamic energy is defined as E = $0.5\alpha$C$V^2$, where $\alpha$ is the switching activity, C is the capacitance and $V$ is the supply voltage. The capacitance is a function of various physical (transistor width, wire length, *etc.*) and architectural (buffer size, number of ports, flit size, *etc.*) parameters. The physical parameters can be specified in the Orion configuration file and the architectural parameters are extracted from GARNET. GARNET feeds in the per-component activity to Orion. With all the above parameters available, the total network dynamic power is calculated. The total leakage power consumed in the network is the sum total of the leakage power of router buffers, crossbar, arbiters and links. Since GEMS does not simulate actual data values, we do not feed them into Orion. Orion uses the leakage power models described in [7] for its calculations.

$$\begin{aligned} E_{router} &= E_{buffer\_write} + E_{buffer\_read} \\ &\quad + E_{vc\_arb} + E_{sw\_arb} + E_{xb} \end{aligned} \tag{1}$$

**Table 1.** GARNET usage parameters in GEMS

| | |
|---|---|
| g_GARNET_NETWORK | True: use GARNET<br>False: use *simple* network |
| g_DETAIL_NETWORK | True: use *detailed* model<br>False: use *flexible* model |
| g_NETWORK_TESTING | True: network-only mode<br>False: full-system mode |
| g_NUM_PIPE_STAGES | Number of pipeline stages in the router (for flexible-pipeline model) |
| g_VCS_PER_CLASS | Number of virtual channels per message class |
| g_BUFFER_SIZE | Number of flit buffers per VC |
| g_FLIT_SIZE | Number of bytes per flit |

## 2.2 GARNET configuration and statistics

We next present details about the configuration and statistics in GARNET.

**Configuration:** GARNET can be easily configured to model a desired interconnect. The various input parameters of GARNET are shown in Table 1. The various configurable elements of GARNET are as follows:

1. **Network topology:** In GARNET, the network topology is configurable. GEMS allows the number of processors, L2 banks, memory banks, *etc.,* to be specified. GARNET models network interfaces (NICs) that interface the various cache and memory controllers with the network. Each NIC is connected to a network router. The topology of the interconnect is specified by a set of links between the network routers in a configuration file. The configuration file can be used to specify complex network configurations like fat tree [10], flattened butterfly [17], *etc.* Since the topology in GARNET is a point-to-point specification of links, irregular topologies can be evaluated as well.

2. **Interconnect bandwidth:** Interconnect bandwidth can be configured through flit size[3] that specifies the link bandwidth – number of bytes per cycle per network link. Links with lower bandwidth than this (for instance, some off-chip links) can be modeled by specifying a longer latency across them in the topology configuration file.

3. **Router parameters:** Since the topology of a GARNET network is configurable, routers in the network can have an arbitrary number of input and output ports. The number of VCs per port and the number of buffers per VC are parameterized and can be specified.

4. **Routing algorithm:** Each router in GARNET has a routing table that is populated at configuration time. The routing table, as shown in Figure 2, specifies the output port to which a particular packet needs to be forwarded to reach a particular destination. GARNET reads the topology configuration file and populates the routing tables by calculating a minimal path between the nodes. There can be more than one minimal path that can exist from a source to a destination and GARNET puts all such entries in the routing table. The routing table also has a field, *link_weight*, that can be specified in the topology configuration file on a link-by-link basis. During the RC phase inside the routers, these weights are looked up and compared. The output link with the minimal weight is selected for routing. This allows various routing algorithms to be modeled in GARNET. Figure 3 demonstrates how X-Y routing can be modeled by assigning all X direction links to have a lower weight than the Y direction links. This leads to routing conflicts being broken in favor of X-direction links. Other routing protocols, such as X-Y-Z routing, left-first, south-last, *etc.,* can be similarly specified using the configuration file. In it's current distribution, GARNET relies on the routing protocol to avoid deadlocks in the network. Thus, the assignment of link weights should be done such that it leads to a deadlock-free routing scheme. Since GARNET adopts a table-based approach to routing, adaptive routing can be easily implemented by choosing routes based on dynamic variables, such as link activity, buffer utilization, *etc.,* and not based on link weights. A deadlock recovery/avoidance scheme would also have to be implemented in that case. Table-based routing also allows the

[3]GARNET assumes the flit size to be equal to phit (physical unit) size. If the phit size is different than a flit size, the on-chip link latencies can be readily modified to incorporate the delay a flit would take to traverse an on-chip link.

| | |
|---|---|
| Destination 1 | <outport $A_1$, weight $A_1$>, <outport $B_1$, weight $B_1$>, .. |
| Destination 2 | <outport $A_2$, weight $A_2$>, <outport $B_2$, weight $B_2$>, .. |
| ●<br>● | |
| Destination N | <outport $A_N$, weight $A_N$>, <outport $B_N$, weight $B_N$>, .. |

**Figure 2.** Routing table inside a GARNET router

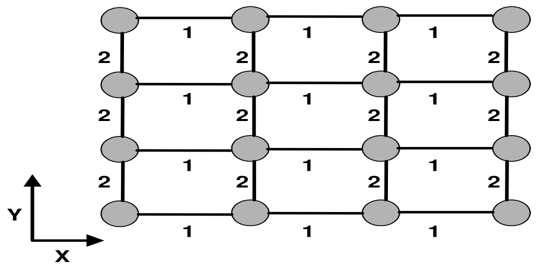


**Figure 3.** Link weight assignment for X-Y routing

opportunity to employ fancier routing schemes, such as region-based routing [12] and other non-minimal routing algorithms. The table population algorithm would have to be modified for that, which can be done easily. The power consumption of the routing tables are, however, not currently modeled in GARNET, since it is not required for simpler routing schemes, such as X-Y routing. This can, however, be easily modeled similar to router buffers.

5. **Configuring the router pipeline:** GEMS uses a queue-driven event model to simulate timing. Various components communicate using message buffers of varying latency and bandwidth, and the component at the receiving end of the buffer is scheduled to wake up when the next message is available to be read from the buffer. The simulation proceeds by invoking the wakeup method for the next scheduled event on the event queue. Since GARNET is a part of GEMS, it is also implemented in a similar event-driven fashion. The dependencies between various modules are as follows. The NIC puts flits into a queue and schedules the network link after a specific number of cycles. The network link wakes up and picks up the specific flit and schedules the input port of the attached router next. The input port wakes up and picks up the flit from the network link and writes into a particular VC buffer. The subsequent router pipeline stages (VA, SA and ST) follow in a similar event-driven fashion. The event-driven nature of the implementation allows the GARNET pipeline to be readily changed to a different one with a bit of programming effort. Different stages can be merged with each other to shorten the pipeline to one with fewer stages. Additional stages can similarly be added by simply following the event-driven approach. The modular design also allows the allocators to be easily modified. The VC and switch allocators currently implemented are separable [25] with the local and global arbitration being round-robin in nature. If a different allocation scheme is desired (*e.g.,* matrix arbiter), only the arbitration code would need to be modified. As an illustration of the ease with which GARNET can be modified, it took us just one week to code up and evaluate EVCs in the GEMS+GARNET infrastructure.

   **Flexible pipeline model:** Some system designers may wish to vary the NoC pipeline length to explore the sensitivity of full-system performance to per-hop router delay, while still faithfully modeling link, switch and buffer constraints. GARNET includes a "flexible-pipeline" model to enable such evaluations. This model implements an output-queued router that adds a variable number of cycles after a flit reaches an output of a router. This models the router pipeline delay. A head flit, on arriving at a router, moves to a destined output port and output VC, which is decided beforehand. The output port and VC are arbitrated for one hop in advance. This is because, on arrival at the router, the flit has to be guaranteed a buffer. The flit gets buffered at the output queue for the number of pipeline stages specified. In the meanwhile, the head flit sends a VC arbitration request to the downstream router, which calculates the output port for the flit and looks for a free VC. On successfully getting a free VC, the router signals the upstream router that VC arbitration for the flit was completed. The head flit now arbitrates for the output physical link and moves out of the router. Body and tail flits follow the same course except that they do not arbitrate for a VC. Tail flits also release the VC after they exit the router. A flit, prior to its departure from a router, checks for free buffers in the downstream router. Thus, finite buffering is also modeled. This implementation models link contention, VA, router pipeline delay and finite buffering. What it idealizes is the flow control, specifically how the router crossbar switch is allocated.

6. **Network-only simulation:** Some network studies require an interconnect model to be evaluated with synthetic traffic types (uniform random, tornado, bit complement, *etc.*) as inputs. Such studies are very common in the interconnection network research community. Synthetic traffic stresses various network resources and provides an estimate of the network's performance/power under various scenarios. Keeping this in mind, GARNET has been designed to run in a network-only mode also. Different synthetic traffic types can be fed to the network and various network performance/power statistics extracted.

**Statistics:** The various statistics that GARNET outputs are total number of packets/flits injected into the network, link utilization (average and per-link), average load on a VC, average network latency (inside the network as well as queuing at the interfaces) and network power numbers (dynamic and leakage). Apart from this, various counters can be easily added to modules and statistics displayed.

## 3 GARNET Validation

We validated the GARNET model by running network-only simulations with synthetic traffic (uniform random and tornado) and comparing the results against previously published ones [19, 22]. We also simulated other synthetic

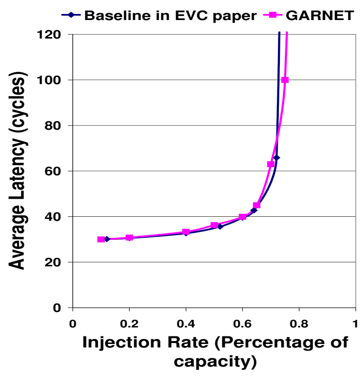


**Figure 4.** GARNET under uniform-random traffic

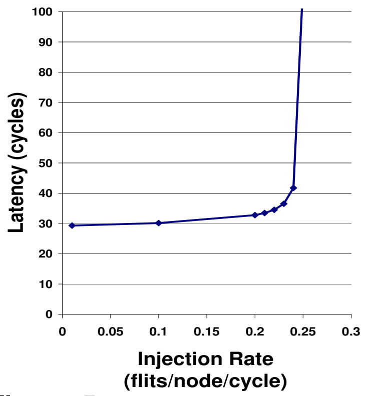


**Figure 5.** GARNET under tornado traffic

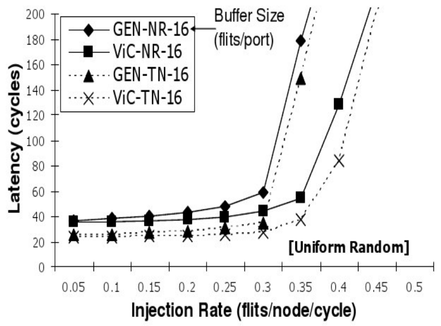


**Figure 6.** ViChar [22] baseline under tornado (TN) and normal random (NR) traffic

traffic patterns and validated them against the PoPNet [1] network simulator, establishing that the latency-throughput curves match.

### 3.1 Validation of uniform random traffic with EVCs

In [19], results were presented for EVCs for uniform random traffic on a network with the classic five-stage pipeline router. That work used a cycle-accurate in-house interconnect simulator that modeled all major components of the router pipeline, *viz.*, buffer management, routing algorithm, VA, SA, and flow control between routers. GARNET, similarly, models all the above aspects. We evaluated the detailed network configuration in GARNET on a 7×7 mesh with dimension-ordered routing, five ports per router, and six VCs per port with shared 24-flit buffers per port. Traffic was generated in a manner similar to [19]. We obtained the actual latency numbers from the authors of [19] and plotted them against latency values observed in GARNET in Figure 4. As shown in the figure, the plots saturate at similar injection rates. The low-load latencies of GARNET also conform to that of EVC's baseline network.

### 3.2 Validation of tornado traffic with ViChaR

In [22], results were presented for ViChaR for tornado traffic on a baseline network with the classic five-stage pipelined router. That work used an in-house cycle-accurate on-chip network simulator that operated at the granularity of individual architectural components. The simulator modeled pipelined routers and network links very similar to how GARNET models them. We tried to reproduce their results by simulating a similar network configuration on an 8×8 mesh with dimension-ordered routing, five ports per router, and four VCs per port with each VC having 4-flit buffers. The latency plot that we obtained from GARNET is shown in Figure 5. This closely matches results shown in Figure 6 (GEN-TN-16 curve) in [22]. GARNET saturates slightly earlier than ViChaR's baseline network. This could be an artifact of different kinds of switch and VC allocators used in GARNET and ViChaR's baseline router. GARNET uses a round-robin separable allocation scheme. Although ViChaR's baseline router uses separable allocators, the exact arbiters are not mentioned in the paper. The low-load latencies also seem to conform to that of ViChaR's baseline.

### 3.3 Simulation time overhead

In the SIMICS + GEMS setup, the major simulation overhead is the interfacing of additional timing modules (RUBY, OPAL) with the functional module (SIMICS). Additional details in the timing module does not have a huge impact on simulation time. For instance, to simulate the same number of instructions, GEMS + GARNET took about 21% more simulation time than GEMS + *simple* network in our experiments.

## 4 Evaluation of a System Design Decision with and without GARNET

As discussed briefly in Section 2, there is no clear answer yet as to whether future CMP systems would have shared or private last-level caches. We performed an experiment in which we ran a scientific benchmark in full-system fashion and evaluated a shared-cache and a private-cache CMP system separately. We kept all other systems parameters, benchmark data set, processor configuration, coherence protocol, total on-chip cache capacity, on-chip network, memory capacity, *etc.*, the same. The aim of the experiment was to conclude whether a private-cache or a shared-cache configuration was preferable for the chosen benchmark. We performed separate experiments, one with the *simple* pipeline of the original GEMS and one with the detailed pipeline of GARNET.

### 4.1 Target system

We simulated a tiled 16-core CMP system with the parameters shown in Table 2. Each tile consists of a two-issue in-order SPARC processor with 32 KB L1 I&D caches. It also includes a 1 MB private/2 MB shared L2 cache bank. To keep the entire on-chip cache capacity same (32 MB) for both the shared and private configurations, a total of 16 MB on-chip directory caches were used for the private L2 configuration. A DRAM was attached to the CMP via four memory controllers along the edges. The DRAM access latency was modeled as 275 cycles. The on-chip network was chosen to be a 4×4 mesh, consisting of 16-byte links with deterministic dimension-ordered XY routing. We simulated a 4-cycle router pipeline with a 1-cycle link. This is the default

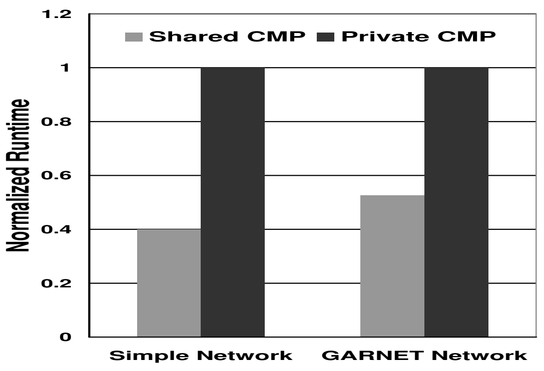


**Figure 7.** *LU* with *simple* and GARNET networks

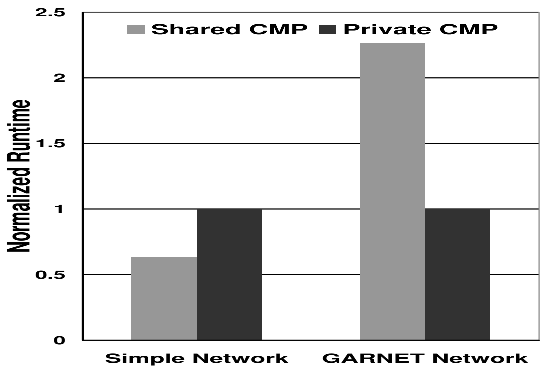


**Figure 8.** *RADIX* with *simple* and GARNET networks

GARNET network configuration. Each input port contains eight virtual channels and four buffers per virtual channel. Since the *simple* network assumes a one-cycle router delay, to mimic the same pipeline, we assume the delay of every on-chip link to be four cycles. The *simple* network assumes packet-level buffering and does not model the router pipeline. It also assumes infinite packet buffering at routers. Hence, the best we could do was to match the per-hop latencies in the *simple* model to that of the GARNET model. To better isolate the contention effects that GARNET captures, and not overwhelm the latency effects of a short *vs.* long pipeline, we selected the same "per-hop" latencies in both the models.

**Table 2.** Simulation parameters

| | |
|---|---|
| Processors | 16 in-order 2-way SPARC cores |
| L1 Caches | Split I&D, 32 KB 4-way set associative, 2 cycle access time, 64-byte line |
| L2 Caches | 1 MB private/2 MB shared per tile, 10 cycle access time, 64-byte line |
| Directory Caches | 4 MB per memory controller, 16 MB total on chip (for private L2 configuration) |
| Memory | 4 memory controllers, 275-cycle DRAM access + on-chip delay |
| GARNET On-chip Network | 4×4 2D mesh, 16-byte links, 4 cycle router pipeline, 8 virtual channels, 4 buffers per virtual channel, 1 cycle on-chip link latency |
| Simple On-chip Network | 4×4 2D mesh, 16-byte links, 5 cycle per hop on-chip latency |

## 4.2 Workload

We ran the *RADIX* and *LU* application from the SPLASH-2 [28] application suite on the above-mentioned configurations. SPLASH-2 is a suite of scientific multi-threaded applications that has been used in academic evaluations for the past two decades. The benchmarks were warmed up and checkpointed to avoid cold-start effects. We ensured that caches were warm by restoring the cache contents captured as part of our checkpoint creation process. We ran the parallel portion of each benchmark to completion. To address the variability in parallel workloads, we simulated each design point multiple times with small, pseudo-random perturbations of request latencies to cause alternative paths to be taken in each run [2]. We averaged the results of the runs.

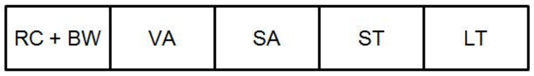
## 4.3 Evaluation results

Figure 7 shows the normalized runtime of the *LU* benchmark with the *simple* GEMS network as well as the GARNET network. Both network results indicate that a shared-cache CMP configuration would perform better for the *LU* benchmark and the particular configuration. However, the magnitude of speedup obtained from the different networks is different. The evaluations with the GARNET network report a 47.3% reduction in overall system runtime as opposed to 60% reported with the *simple* network configuration. Contention modeling in the GARNET network increased the average memory latency of the shared-cache configuration by 18%, whereas increasing the average memory latency of the private-cache configuration by only 9%. This contention latency could not be captured by the *simple* GEMS network. Thus, by having a realistic network model like GARNET, more accurate full-system performance numbers can be obtained.

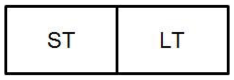
Figure 8 shows the normalized runtime of the *RADIX* benchmark with the *simple* GEMS network as well as the GARNET network. The *simple* network results indicate that a shared-cache CMP configuration performs better for *RADIX* on the specific configurations. However, the GARNET results show that a private-cache CMP architecture far outperforms the shared-cache CMP architecture. Contention modeling in GARNET leads to an increase of 8% in average memory latency for the shared-cache configuration while increasing the average memory latency by only 0.02% for the private-cache configuration. Although there was just an 8% increase in average memory latency for the shared-cache configuration, the full-system runtime increased by more than 3×. The contribution of user instructions to this increase was only 20%, while the OS instructions increased by 3×. As discussed earlier, timing variations in the network can cause parallel programs to take different execution paths, causing differences in the number of instructions and memory accesses. This happens due to different OS scheduling decisions, the different order in which threads attain spin locks, *etc.* This reiterates the importance of an accurate network model, which captures realistic timing characteristics, inside a full-system framework.

## 5 Express Virtual Channels

As explained in Section 2.1, a packet in a state-of-the-art network needs to traverse multiple stages of the router pipeline at each hop along its route. Hence, energy/delay in such networks is dominated largely by contention at intermediate routers, resulting in a high router-to-link energy/delay ratio. EVCs [19] were introduced as a flow-control and router microarchitecture design to reduce the router energy/delay overhead by providing *virtual* express lanes in the network which can be used by packets to bypass intermediate routers along a dimension. The flits traveling on these lanes circumvent buffering (BW) and arbitration (VA and SA) by getting forwarded as soon as they are received at a router, reducing the router pipeline to just two stages (Figure 9). This reduces both the latency (due to bypassing of the router pipeline) and dynamic power (as buffer reads/writes and VC/switch arbitrations are skipped). This technique also leads to a reduction in the total number of buffers required in the network to support the same bandwidth, which in turn reduces the leakage power.



(a) Traditional 5-stage pipeline



(b) EVC pipeline

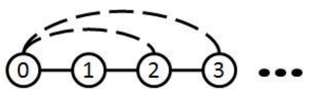
**Figure 9.** Reduction in pipeline stages for EVCs

The express virtual lanes are created by statically designating the VCs at all router ports as either normal virtual channels (NVCs), which carry flits one hop; or $k$-hop EVCs ($k$ can take all values between 2 and some $l_{max}$), which can carry flits $k$ hops at a time, bypassing the $k-1$ routers in between, in that dimension (XY routing is assumed). In dynamic EVCs, discussed in the original paper, each router can act as either a *source/sink* node, which allows flits to get buffered and move through the normal router pipeline, or as a *bypass* node which gives priority to flits on EVCs to pass straight through the switch, without having to be buffered. In the *source/sink* routers, the head flits arbitrate for $k$-hop EVCs or NVCs, depending on their route (*i.e.*, the number of hops remaining in that dimension) and the availability of each type of VCs. The body and tail flits follow on the same VC and release it when the tail flit leaves. Once a $k$-hop EVC is obtained, the intermediate $k-1$ nodes can be bypassed since the EVC flits send lookaheads, one cycle in advance, to set up the switches at the *bypass* nodes (to ensure uncontended access to the output ports). All packets try to use a combination of EVCs to best match their route, such that they are able to bypass most nodes in a dimension. These ideas are illustrated in Figure 10.

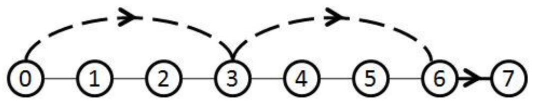
### 5.1 Evaluation of EVCs in GARNET

We evaluated EVCs on GEMS with GARNET as the interconnect model, and EVCs coded into it, to study the system-level impact of this network optimization.

**Target system:** We simulated a 64-core CMP with shared L2 distributed in a tiled fashion [16]. Each core consists of a two-



(a) EVCs with $l_{max}$ = 3



(b) Path from 0 to 7 using 3-hop EVCs and an NVC

**Figure 10.** Express virtual channels

issue in-order SPARC processor with 64 KB L1 I&D caches. Each tile also includes a 1 MB L2 bank. A DRAM is attached to the CMP via eight memory controllers placed along the edges. The DRAM access latency was modeled as 275 cycles. The on-chip network was chosen to be an 8×8 mesh consisting of 16-byte links with deterministic XY dimension-ordered routing. Each message class was assumed to contain eight virtual channels with finite buffering.

**Protocols:** We evaluated the MOESI based directory protocol, which is part of the GEMS release, for the network with EVCs, and compared it to the baseline network modeled in GARNET.

**Workloads:** We ran SPLASH-2 applications [28] on the above-mentioned configuration. We ran the parallel portion of each workload to completion for each configuration. All benchmarks were warmed up and checkpointed to avoid cold-start effects. We ensured that caches were warm by restoring the cache contents captured as part of our checkpoint creation process.

**Evaluation results:** The system-level impact of a network optimization like EVCs can vary depending upon the system configuration. For instance, if there is a large number of L2 misses and most requests go off-chip (either because of small on-chip cache or non-local access patterns), consuming hundreds of cycles, saving few tens of cycles on-chip by router bypassing would not give a huge speedup for the whole system. However, if most requests are satisfied on-chip, which only consumes tens of cycles, then reducing the latency of transfer of requests and data on the chip would decrease the overall miss latency as well. Figure 11 shows the comparison of the normalized network latencies of each benchmark against the baseline. We observe an average network latency improvement of 21.5% and, interestingly, this number is almost uniform across all benchmarks. This is because the benchmarks that we ran, did not stress the on-chip network and mostly ran at low loads. Thus, EVCs provide similar performance improvements to all of them over the baseline. We also observe an almost uniform 22.2% reduction in average miss latencies across all benchmarks. This can be understood by studying the cache hit rates for the benchmarks in the baseline network for this particular chip configuration. We observe that, on an average, 86% of the L1 misses are satisfied on-chip, and less than 5% of the total memory accesses go off-chip as L2 misses. This means that on-chip latency dominates the average miss latency value in our case. Hence, the network speedup by EVCs translates to a reduc-

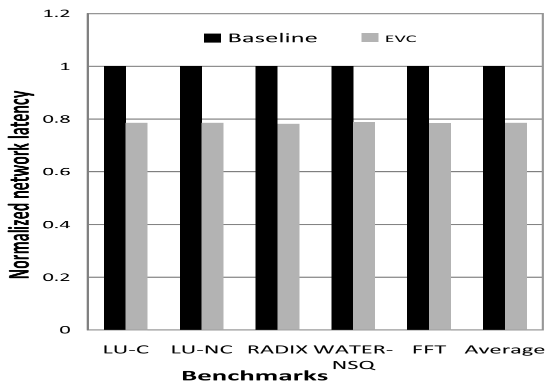


**Figure 11.** Normalized network latency

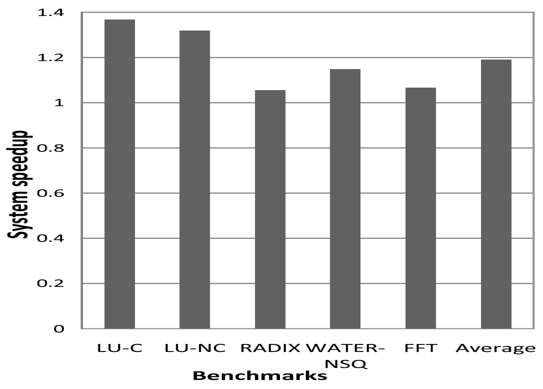


**Figure 12.** System speedup

tion in miss latencies of the same order. However, Figure 12 shows that the overall system speedup is not the same for all benchmarks. For instance, *RADIX* and *FFT* have a smaller overall speedup than the others. This can be attributed to the change in the number of instructions executed by the benchmarks. For *RADIX* and *FFT*, the total number of instructions executed by the program with an EVC network increases over the baseline case. This could be due to a different path taken by the execution because of changes in the timing of delivery of messages, as discussed before. This also changes the cache hit/miss dynamics, and we observe that *RADIX* has a 4% higher L2 miss rate than the baseline case, which would in turn result in more off-chip accesses, thus reducing the impact of reduction of on-chip latency by EVCs. This is an instance of a case where the performance gain by a network optimization might not translate to overall system speedup of the same order.

Evaluating a network optimization like EVCs from a system's perspective can help make design decisions, such as using EVCs to send critical control messages like congestion alerts faster, or to send coherence messages like invalidates, which may be on the critical path, *etc.*, and study the overall impact. GARNET provides a useful platform to study these effects in total, rather than in isolation.

## 6 Related Work

Simulation is one of the most valuable techniques used by computer architects to evaluate design innovations. In the past, SimpleScalar [4] was widely used in the architecture research community to evaluate uniprocessor systems. However, it and other such simulators run only user-mode single-threaded workloads. With the rapid adoption of many-core chips, designers are now interested in simulation infrastructures that can run multithreaded workloads and model multiple processing cores along with the memory system. Multithreaded workloads depend upon many OS services (*e.g.*, I/O, synchronization, thread scheduling, and migration). Full-system simulators that can run realistic parallel workloads are thus essential for evaluating many-core chips. While there exist full-system simulators, like RSIM [23] and SESC [27], that can simulate multiprocessor systems, the interconnection network (specifically the NoC) is not modeled accurately in these frameworks. In the many-core era, the on-chip network is an integral part of the memory system and not modeling its details leads to an approximate model. While PharmSim [5] models the entire system, including the on-chip network, to a certain degree of detail, it is not publicly available for use.

There are various network-only simulators, like NOXIM [24] and SICOSYS [26], that the NoC community uses for experiments, but they cannot be used to perform full-system evaluations. Although SICOSYS has been plugged into RSIM for simulating symmetric multiprocessor systems, the platform is not used for studying CMP systems with an NoC. ASIM [11] is a full-system simulator, used in industrial laboratories, that models the entire processor and memory system. From what we could gather, it models only ring interconnects and thus cannot be used in design explorations that use interconnects other than a ring. There have been recent efforts [3] into looking at FPGA-based simulation infrastructures for many-core systems. These, however, require detailed register-transfer level implementation which is time-consuming. To the best of our knowledge, there exists no publicly available simulation infrastructure that models the entire processing, memory and interconnect system at the level of detail as the proposed GEMS+GARNET toolset.

## 7 Conclusion

With on-chip networks becoming a critical component of present and future CMP designs, understanding the system-level implications of network techniques becomes very important. It also becomes necessary to evaluate CMP design proposals with a realistic interconnect model. In this work, we presented GARNET, a detailed network model, which is incorporated inside a full-system simulator (GEMS). We analyzed the impact of a realistic interconnection network model on system-level design space explorations. We also evaluated EVCs, a network flow control technique, in a full-system fashion and observed overall system performance improvements that cannot be predicted by network-only simulations. Our results indicate that the close-knit interactions between the memory system and the network can no longer be ignored. We thus believe that system designs should also model the on-chip communication fabric, and on-chip network designs should be evaluated in a full-system manner.

## Acknowledgments

The authors would like to thank the GEMS team at University of Wisconsin for helping with the integration of GAR-

NET. This work was supported by NSF (grant no. CNS-0613074), MARCO Gigascale Systems Research Center and SRC (contract no. 2008-HJ-1793).

## References

[1] PoPNet. `http://www.princeton.edu/edu/~lshang/popnet.html`.

[2] A. R. Alameldeen and D. A. Wood. IPC considered harmful for multiprocessor workloads. *IEEE Micro*, 26(4):8–17, 2006.

[3] Arvind, K. Asanovic, D. Chiou, J. C. Hoe, C. Kozyrakis, S.-L. Lu, M. Oskin, D. Patterson, J. Rabaey, and J. Wawrzynek. RAMP: Research accelerator for multiple processors – A community vision for a shared experimental parallel HW/SW platform. Technical Report UCB/CSD-05-1412, EECS, Dept. Univ. of California, Berkeley, Sept. 2005.

[4] T. Austin, E. Larson, and D. Ernst. Simplescalar: An infrastructure for computer system modeling. *Computer*, 35(2):59–67, Feb. 2002.

[5] H. Cain, K. Lepak, B. Schwarz, and M. H. Lipasti. Precise and accurate processor simulation. In *Proc. Wkshp. Computer Architecture Evaluation using Commercial Workloads*, 2002.

[6] M. F. Chang, J. Cong, A. Kaplan, C. Liu, M. Naik, J. Premkumar, G. Reinman, E. Socher, and S.-W. Tam. Power reduction of CMP communication networks via RF-interconnects. In *Proc. Int. Symp. Microarchitecture*, Nov. 2008.

[7] X.-N. Chen and L.-S. Peh. Leakage power modeling and optimization of interconnection networks. In *Proc. Int. Symp. Low Power Electronics and Design*, pages 90–95, Aug. 2003.

[8] D. E. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1998.

[9] W. J. Dally. Virtual channel flow control. *IEEE Trans. Parallel and Distributed Systems*, 3(2):194–205, Mar. 1992.

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

[11] J. Emer, P. Ahuja, E. Borch, A. Klauser, C.-K. Luk, S. Manne, S. S. Mukherjee, H. Patil, S. Wallace, N. Binkert, R. Espasa, and T. Juan. Asim: A performance model framework. *Computer*, 35(2):68–76, 2002.

[12] J. Flich, A. Mejia, P. Lopez, and J. Duato. Region-based routing: An efficient routing mechanism to tackle unreliable hardware in network on chips. In *Proc. Int. Symp. Networks-on-Chip*, pages 183–194, May 2007.

[13] IBM. `http://www-128.ibm.com/developerworks/power/library/pa-expert1.html`.

[14] Intel. From a few cores to many: A tera-scale computing research overview. `http://download.intel.com/research/platform/terascale/terascale_overview_paper.pdf`, 2006.

[15] N. E. Jerger, L.-S. Peh, and M. Lipasti. Virtual circuit tree multicasting: A case for on-chip hardware multicast support. In *Proc. Int. Symp. Computer Architecture*, June 2008.

[16] C. Kim, D. Burger, and S. W. Keckler. An adaptive, non- uniform cache structure for wire- delay dominated on-chip caches. In *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[17] J. Kim, J. Balfour, and W. J. Dally. Flattened butterfly topology for on-chip networks. In *Proc. Int. Symp. Microarchitecture*, Nov. 2007.

[18] J. S. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff. Energy characterization of a tiled architecture processor with on-chip networks. In *Proc. Int. Symp. Low Power Electronics and Design*, pages 424–427, Aug. 2003.

[19] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha. Express virtual channels: Towards the ideal interconnection fabric. In *Proc. Int. Symp. Computer Architecture (and IEEE Micro Top Picks 2008)*, June 2007.

[20] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Alameldeen, K. Moore, M. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *Computer Architecure News*, 2005.

[21] J. Moses, R. Illikkal, R. Iyer, R. Huggahalli, and D. Newell. Aspen: Towards effective simulation of threads and engines in evolving platforms. In *Proc. Int. Symp. Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pages 51–58, Oct. 2004.

[22] C. A. Nicopoulos, D. Park, J. Kim, V. Narayanan, M. S. Yousif, and C. Das. ViChaR: A dynamic virtual channel regulator for network-on-chip routers. In *Proc. Int. Symp. Microarchitecture*, Dec. 2006.

[23] V. S. Pai, P. Ranganathan, and S. V. Adve. RSIM: an execution-driven simulator for ILP-based shared-memory multiprocessors and uniprocessors. In *Proc. Third Wkshp. on Computer Architecture Education*, 1997.

[24] M. Palesi, D. Patti, and F. Fazzino. NOXIM. `http://noxim.sourceforge.net`.

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

[26] V. Puente, J. A. Gregorio, and R. Beivide. SICOSYS: an integrated framework for studying interconnection network performance in multiprocessor systems. In *Proc. Euromicro Wkshp. Parallel, Distributed and Network-based Processing*, pages 15–22, 2002.

[27] J. Renau, B. Fraguela, W. L. J. Tuck, M. Prvulovic, L. Ceze, S. Sarangi, K. S. P. Sack, and P. Montesinos. SESC simulator. `http://sesc.sourceforge.net`, 2005.

[28] SPLASH. `http://www-flash.stanford.edu/apps/SPLASH/`.

[29] Sun. `http://www.sun.com/processors/throughput/`.

[30] H.-S. Wang, X.-P. Zhu, L.-S. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *Proc. Int. Symp. Microarchitecture*, pages 294–305, Nov. 2002.

[31] M. Zhang and K. Asanovic. Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors. In *Proc. Int. Symp. Computer Architecture*, pages 336–345, May 2005.