

A System-Level Perspective for Efficient NoC Design

Amit Kumar, Niket Agarwal, Li-Shiuan Peh and Niraj K. Jha
Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544
{amitk, niketa, peh, jha}@princeton.edu

Abstract

With the advent of chip multiprocessors (CMPs) in mainstream systems, the on-chip network that connects different processing cores becomes a critical part of the design. There has been significant work in the recent past on designing these networks for efficiency and scalability. However, most network design evaluations use a stand-alone network simulator which fails to capture the system-level implications of the design. New design innovations, which might yield promising results when evaluated using such stand-alone models, may not look that attractive when evaluated in a full-system simulation framework.

In this work, we present GARNET, a detailed network model incorporated inside a full-system simulator which enables system-level performance and power modeling of network-level techniques. GARNET also facilitates accurate evaluation of techniques that simultaneously leverage the memory hierarchy as well as the interconnection network. We also discuss express virtual channels, a novel flow control technique which improves network energy/delay by creating virtual lanes in the network along which packets can bypass intermediate routers.

1 Introduction

There is an ever-increasing need to design computer hardware or software in the context of the final design. A design innovation might be very attractive intuitively. However, it may be difficult to quantify its impact unless full-system simulation results are presented. A *full-system simulator* is a computer program that can simulate full computer systems at a level of detail such that complete software stacks from real systems can run on the simulator without any modification. Since a simulator is essentially a software program, it can arbitrarily parameterize, control, inspect and analyze the target system it is modeling. A full-system simulator should model the processing system and the memory sub-system in a functionally-accurate manner. Support for timing simulation is a very important characteristic that these simulators must possess. Furthermore, flexibility in modifying the underlying microarchitectural features is also a desired property. With the advent of multi-core systems and the need to simulate systems with large processor counts and complicated memory hierarchy, full-system simulators are increasingly in demand.

With the rapid adoption of many-core chips [5], the on-chip network becomes an integral part of future chip multiprocessor (CMP) systems. Communication affects not only performance, but may also be the most power-hungry part of the system. Thus, interconnection networks can no longer be ignored while performing system characterization. There are three major reasons why full-system simulation infrastruc-

tures should have detailed and accurate network models:

- A system-level optimization might seem very promising when simulated with an approximate network model. However, it may turn out to be not so effective in the actual design. For example, a coherence protocol might be designed under certain assumptions about the network model, which, if not true, might prove costly in the final design.
- There is a need to evaluate network-level techniques from a full-system point of view in order to find out the exact implications of the technique on the whole system. This is important because a network technique, e.g., a new flow control, might show some particular network power and latency characteristics when evaluated on a network-only simulator, but might have totally different system-level implications.
- A lack of full-system simulation infrastructure prevents proper evaluation of techniques that simultaneously use the interconnection network as well as other top-level system components, e.g., caches, etc. For example, a new cache-coherence protocol that leverages some of the network properties, e.g., ordering, flow control, etc., is very difficult to correctly evaluate without a full-system simulation infrastructure which models the network in detail.

Naturally, we wish to simulate systems with high accuracy and reasonable performance. However, 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 has performance implications. However, there are platforms that carefully trade off accuracy and performance using sufficiently abstract important system characteristics while still maintaining reasonable speed of simulation on realistic workloads. One such platform is the GEMS [7] full-system simulation platform.

GEMS limitations: GEMS does a good job of capturing detailed aspects of the processing cores, cache hierarchy, cache coherence and memory controllers. These components are fairly well parameterized and easy to extend. This has enabled the widespread use of GEMS in the computer architecture research community. However, a limitation of GEMS 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. A message traverses the network hop-by-hop towards the destination, stalling when there is contention for

bandwidth. This is an approximate implementation and far removed from what a state-of-art interconnection network (described in Section 2.1) looks like. Also, GEMS does not model a router or a network interface. 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 is not particularly useful.

In the light of the above issues, we have developed a detailed and flexible interconnect model inside GEMS. Our model is called GARNET. GARNET is a detailed timing model of a state-of-art interconnection network modeled in detail up to the microarchitecture level. A classic five-stage pipelined router [2] with virtual channel flow control is implemented. We describe our model in detail in the following section.

The contribution of GARNET is three-fold. Firstly, it provides an opportunity for interconnection network evaluations to be done in a full-system simulation fashion. Secondly, it provides a detailed and accurate interconnect model for GEMS, so that it can report more realistic and accurate timing results. Thirdly, it enables evaluation of techniques that use the interconnection network as well as other system-level components simultaneously.

The rest of the paper provides the details of the GARNET model (Section 2) and then describes a novel flow-control technique (Section 3). We finally conclude and present some future directions of research in Section 4.

2 GARNET Network Model

In this section, we first describe a state-of-the-art on-chip network, followed by an overview of GEMS, and finally the detailed description of GARNET.

2.1 State-of-the-art on-chip network

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. Figure 1(a) shows the microarchitecture of a virtual channel (VC) router present at each node in this network – its major components are the input buffers, route computation logic, VC allocator, switch allocator, and crossbar switch. Figure 1(b) shows the router pipeline with each flit going through buffer write (BW) where it gets decoded and buffered according to the packet’s input VC, VC allocation (VA) and switch allocation (SA) where it arbitrates for a free VC and its next output port simultaneously in a speculative manner [8], switch traversal (ST) where it traverses the crossbar, and finally link traversal (LT) where it travels to the next node. This pipeline assumes each packet’s route to be computed one hop in advance using look-ahead routing [4] so as to remove route computation delay from the critical path and enabling flits to compete for VCs immediately after the BW stage.

2.2 GEMS overview

Figure 2 gives an overview of the underlying design of GEMS with GARNET. The major GEMS modules in the system are as follows:

- **Simics:** The Simics [9] module is responsible for the actual functional simulation and various timing modules can be hooked up with it. It approximates a simple

in-order processor with no pipeline stalls. All memory operations, like load, store and instruction fetches, are passed onto the memory system timing module (Ruby). Ruby stalls the requesting Simics’ processor and performs the memory operation. By controlling the timing when Simics advances, Ruby simulates the timing-dependent characteristics of instructions.

- **Opal:** The Opal module models a dynamically-scheduled, superscalar, deeply-pipelined SPARC v9 processor. It taps each instruction from Simics and executes the processor pipeline. When it determines that the time has come for an instruction to retire, it instructs the functional simulator of the corresponding Simics processor to advance one instruction.
- **Ruby:** Ruby is the memory system module that does detailed timing simulation of different memory accesses. It models caches, cache controllers, system interconnect, memory controllers and banks of main memory. It allows a great deal of flexibility in specifying various cache coherence protocols. It uses a queue-driven event model to simulate timing. Components communicate using message buffers of various latencies and bandwidth. After simulating the memory access, it is the job of Ruby to “wake up” the Simics processor that was stalled on that particular memory access. Ruby also has a flexible language in which a cache coherence protocol can be specified.

2.3 GARNET

An interconnect model inside a full-system simulation framework should be flexible as well as model in detail the microarchitectural components. Research works focusing on low-level interconnection network issues require detailed features that are easily extensible. On the other hand, for research that is focused on the processor and memory system, an interconnect model, which abstracts the actual hardware with sufficient accuracy, is acceptable. Such approximate models should, however, include parameters that can be easily tuned. For example, the number of pipeline stages in a router, and the link latency and bandwidth should be configurable. We implemented two models which serve the above requirements. We label the approximate model as the “flexible pipeline” model, since it can mimic an arbitrary router pipeline. The network model that models the network microarchitecture in detail is called the “detailed network” model.

- **Flexible pipeline model:** This network model has virtual channel flow control. It is modeled at the flit level. The number of pipeline stages is configurable. Link contention, buffer backpressure due to finite buffering, and virtual channel arbitration are also modeled. Features, such as internal crossbar microarchitecture and speedup, switch allocation and credits, are not modeled. Most of the interconnection network features are parameterized and can be quickly changed to evaluate the desired network configuration.
- **Detailed network model:** The detailed model includes detailed aspects of a router pipeline and has a classic five-stage pipeline [2]. All detailed router features are modeled and are event-driven in nature. Network performance characteristics, such as link activity, buffer usage, switch activity, average network delay, etc., can be

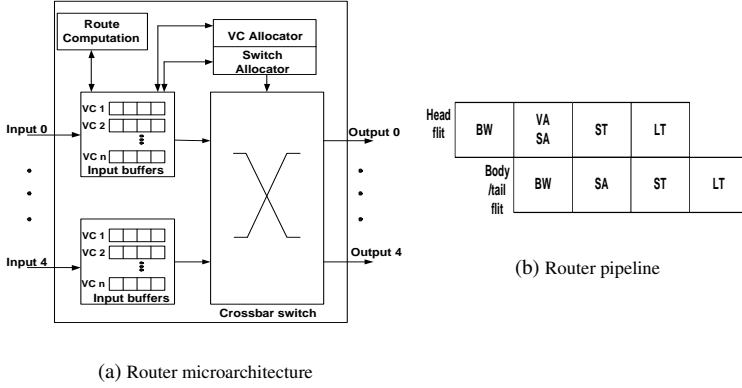


Figure 1. State-of-the-art network

evaluated. These detailed features can be easily used to evaluate network performance and power.

The GARNET model is highly parameterizable and flexible. Various network properties, such as number of VCs, buffer size, number of pipeline stages, flit size, etc., can be passed as parameters. GARNET extracts its topology specification scheme from the GEMS original network model. The topology is specified using a flexible user-defined interface. Any arbitrary network can be created using a configuration file. The topology is specified by a set of links between routers whose latencies can be individually specified. Routing is assumed to be dimension-ordered, but can be easily extended to more sophisticated routing schemes. GARNET models virtual networks for different classes of messages and point-to-point ordering can be enabled or disabled on a per-virtual-network basis. The design provides the flexibility and ease to simulate almost any kind of interconnection network.

With GARNET completely integrated into GEMS, we now have a full-system simulator that can model in detail the processor, memory and interconnection system of a CMP. GEMS already has a power model (Wattch [1]) for the processing elements. We have integrated the interconnection network power model (Orion [10]) into GARNET. Thus, GEMS with GARNET can now report full-system power and performance results.

3 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 largely dominated by contention at intermediate routers, resulting in a high router-to-link energy/delay ratio. In other words, the performance and energy gaps between current state-of-the-art networks and the ideal interconnect, in which all nodes are connected by pair-wise dedicated wires, is quite large. Figure 3(a) shows an example of a typical packet route in this baseline, state-of-the-art network. A packet traveling from node 01 to node 56 must go through the router pipeline at all intermediate nodes along its route.

In this section, we discuss our work on *express virtual channels (EVCs)* [6], a novel flow control and router microarchitecture design which significantly reduces router energy/delay overhead by creating virtual express lanes in the network and allowing packets to bypass intermediate routers when traveling along these lanes. Intuitively, this is achieved

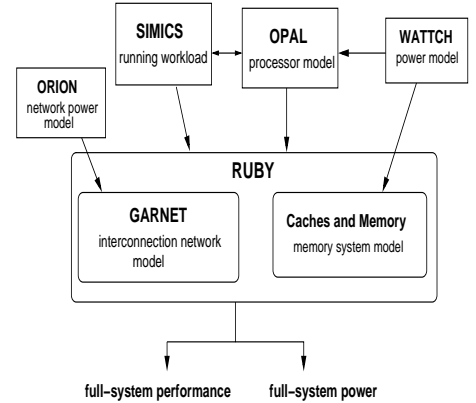


Figure 2. GEMS (with GARNET incorporated) overview

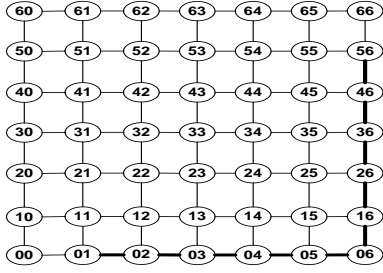
by statically designating a set of EVCs at each router that always connect nodes A and B that are k hops away, and prioritizing these EVCs over normal virtual channels (NVCs) [3] at the intermediate nodes. We present two variants of EVCs – static and dynamic.

3.1 Static EVCs

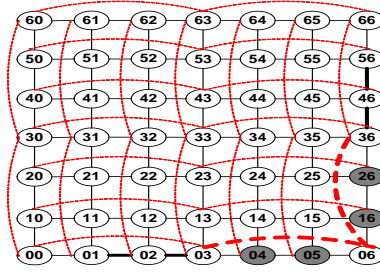
Each node in a static EVC network is distinguished as either an EVC source or sink node, or a bypass node. A node is an EVC source or sink along a specific dimension if an EVC (of length k -hops) along that dimension originates or terminates at that node. Bypass nodes, on the other hand, do not act as EVC sources or sinks and are the ones that are virtually bypassed by packets traveling on EVCs. For example, in Figure 3(b), which shows a 7×7 2D mesh network with three-hop static EVCs ($k = 3$), node 00 is an EVC source or sink for both the x and y dimensions, whereas node 13 is an EVC source or sink node along the x dimension, and node 04 is an EVC source or sink node along the y dimension. Nodes 01 and 02 are examples of bypass nodes along the x dimension; nodes 10 and 20 are examples of bypass nodes along the y dimension. The dotted lines depict EVCs, where EVCs are *not* additional physical channels, but VCs that share existing physical links. The entire set of VCs is divided into two types:

- *NVCs*: these are VCs which are allocated just like in traditional VC flow control [3] and are responsible for carrying a packet through a single hop.
- *k-hop EVCs*: these are VCs which carry the packet through k consecutive hops (where k is the fixed length of the EVC and is uniform throughout the network).

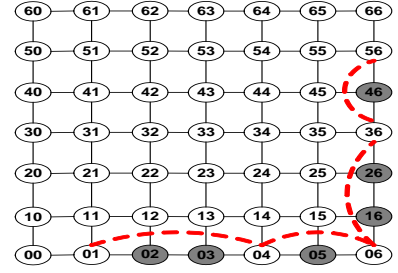
Bypass nodes only support the allocation of NVCs, not EVCs, with EVCs bypassing their router pipelines. Therefore, packets can acquire EVCs along a particular dimension only at EVC source/sink nodes. When a packet traveling on an EVC reaches a bypass node, it bypasses the entire router pipeline, skipping VC allocation as it continues on the same EVC it currently holds. It does not need to go through switch allocation as EVCs are prioritized over NVCs and are thus able to gain automatic passage through the switch without any contention. In other words, a packet traveling on a k -hop EVC traverses the next $k - 1$ nodes without having to go through the router pipeline. Thus, a packet tries to traverse as many EVCs as possible along its route from the source to destination. NVCs are only used to reach an EVC source/sink in order to hop onto an EVC or when the hop-count in a dimension is less than k , the EVC length. Figure 3(b) depicts the



(a) Baseline state-of-the-art network



(b) Three-hop static EVC network



(c) Dynamic EVC network with $l_{max} = 3$

Figure 3. Example packet route in the baseline and an EVC network with shaded nodes depicting the ones which are bypassed (solid lines are NVCs, dotted ones are EVCs)

VCs acquired by a packet traveling from node 01 to node 56 using deterministic XY routing. Here, the packet first travels on two NVCs to reach an EVC source/sink node 03, followed by EVC traversals which allow it to skip the router pipeline at nodes 04, 05, 16 and 26, and finally NVC traversals to reach its destination node 56.

3.2 Impact on router overhead

Bypassing nodes using EVCs helps significantly reduce the router overhead in packet-switched designs. Figure 4(a) shows the *express* router pipeline which a flit goes through whenever it bypasses a node using EVCs. As the flit does not need to go through BW, VA and SA stages, it can head directly to ST, followed by LT, to the next node at the end of which the flit gets latched. The crossbar switch can be aggressively tailored for EVCs to further shorten the *express* pipeline by removing the ST stage and allowing EVC flits to bypass the crossbar as well (shown in Figure 4(b)). As can be seen, the pipeline is now reduced to just LT: approaching that of the ideal interconnect. Note that bypassing routers using EVCs happens *non-speculatively* at all levels of network loading, unlike prior techniques, like bypassing or speculation, which are effective only under low network load.

Unlike speculative techniques, EVCs also lead to a significant reduction in network energy consumption. While traveling on an EVC, a packet skips the router pipeline at intermediate nodes, without the need for getting buffered or having to arbitrate for a VC or the switch port. This in effect saves buffer write energy, buffer read energy, VC arbitration energy and switch arbitration energy, thereby significantly reducing router energy and approaching ideal energy. The aggressive express pipeline removes crossbar traversal energy as well, though link energy increases slightly because of higher load.

Using virtual express lanes, which effectively act as dedicated wires between pairs of nodes, EVCs are also able to create partial communication flows in the network, thereby improving resource utilization and reducing contention at individual routers. Thus, packets spend less time waiting for resources at each router which lowers the average queuing delay, allowing the network to push through more packets before saturation and hence approach ideal throughput.

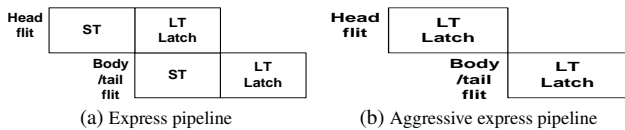


Figure 4. EVC router pipelines

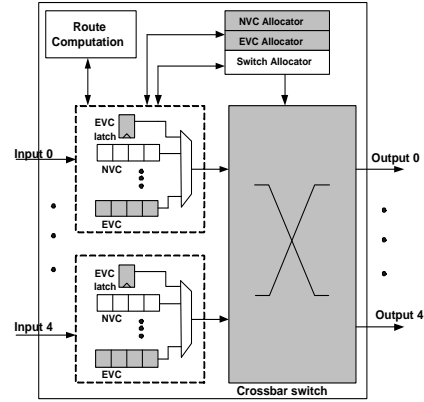


Figure 5. EVC router microarchitecture

3.3 Dynamic EVCs

A static EVC design can lead to asymmetry in the network due to classification of nodes as source/sink and bypass nodes and lack of flexibility due to fixed-length EVCs. Dynamic EVCs overcome these limitations by: (a) making every node in the network a source/sink, thereby leading to a symmetric design with fairness among nodes, and (b) allowing EVCs of varying lengths (from two hops upto a maximum of l_{max} hops) to originate from a node, which improves adaptivity by allowing packets to pick EVCs of appropriate lengths to match their route the best. Unlike static EVCs, which partition all VCs between two bins of NVCs and uniform-length EVCs, dynamic EVCs divide the VCs at each router port into a total of l_{max} bins with one bin for NVCs and $l_{max} - 1$ bins for EVCs of lengths from two through l_{max} hops. Figure 3(c) shows the VCs acquired by a packet traveling from node 01 to node 56 using XY routing in a dynamic EVC network with $l_{max} = 3$, where all nodes are sources and sinks of two- and three-hop EVCs (only the used EVCs are shown for clarity). As can be seen, the packet is able to bypass more nodes along its path by using a combination of EVCs which best match its route.

3.4 EVC router microarchitecture

Figure 5 shows the microarchitecture of a router in a dynamic EVC design. The differences from a generic router are shaded. An EVC latch is used at each input port to hold flits arriving on an EVC. As mentioned before, EVC flits do not need to get buffered at the router while bypassing. As mentioned earlier, the entire set of VCs at each port in a dynamic EVC design is partitioned into NVCs and EVCs of lengths from two through l_{max} hops. In order to prevent head-of-

line blocking, two separate sets of VC allocators are used at every node: one which allocates EVCs and the other NVCs. Depending on the output port and number of hops left in the packet's next dimension, the packet either places a request to allocate an NVC (if the number of straight hops left in the next dimension is less than two which is the smallest EVC length) or an appropriate EVC based on the number of straight hops left in the next dimension. For the non-aggressive express pipeline in Figure 4(a), no modifications need to be made to the crossbar switch design, since the EVC latch in the input port is multiplexed with the local NVC input buffers, sharing a single input port to the crossbar. However, to achieve the aggressive express pipeline in Figure 4(b), where EVC flits (which always travel straight and cannot turn when using EVCs) bypass the switch altogether, additional links are needed around the crossbar only along straight directions, with these links multiplexed with the normal crossbar output. The EVC latch, in this case, needs to be physically located near the center of the router and acts as a staging latch, breaking the flit's path through the network and effectively bypassing the entire datapath of the router.

EVCs require extra wires in the reverse direction for flow-control and starvation signaling. This overhead, however, is only a small fraction of forward wiring – as compared to a 3% overhead for the baseline design (assuming 128-bit wide forward channels), the overhead for an EVC-based design is 5% for static EVCs, 7% for dynamic EVCs with $l_{max} = 2$ and 14% for dynamic EVCs with $l_{max} = 3$.

It can be seen that since EVC connections are virtual as opposed to physical, they are able to connect each node in the network to many other nodes using express paths without incurring the corresponding area/energy overhead in terms of additional router ports and, hence, can be implemented with low hardware overhead using skinny and area- and energy-efficient routers, while requiring only a small overhead in reverse wiring.

3.5 EVC evaluation

We evaluated the performance of EVCs using an in-house commercial cycle-accurate microarchitecture simulator which models all major components of the router pipeline at clock granularity while using Orion [10], an architecture-level network power model, to evaluate network energy.

Figure 6(a) plots flit latencies for uniform random traffic for a 7×7 network as a function of the injected load (assuming two-hop EVCs and the aggressive express pipeline). EVCs significantly outperform the baseline, with static EVCs reducing latency by 29.2% before the baseline saturates. The corresponding reduction for dynamic EVCs is 44.7% along with a significant improvement in throughput, with the network saturating at around 82% capacity. When comparing router energy consumption (Figure 6(b)), at 70% capacity before the baseline saturates, the reduction is 21% and 24.5% for static and dynamic EVCs, respectively, over a power-optimized baseline. The reductions are even higher (34.4% and 52.8% in latency and 23.5% and 38% in router energy for static and dynamic EVCs, respectively) for a larger 10×10 network using three-hop EVCs along with a 23% throughput improvement (approaching 88% capacity) using dynamic EVCs, which highlights the highly scalable nature of this design.

4 Conclusion and Future Work

With on-chip networks becoming a critical component of present and future CMP designs, understanding the system-

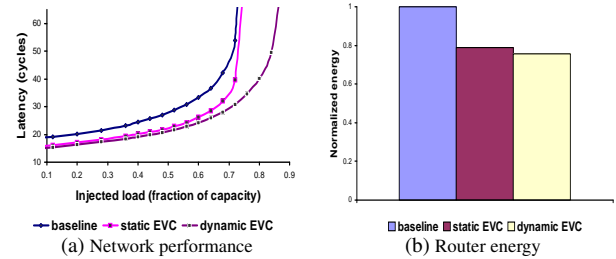


Figure 6. Uniform random traffic results

level implications of network techniques becomes increasingly important. In this work, we presented GARNET, a detailed network model integrated in a full-system simulation environment which allows detailed power/performance evaluations of the complete system. We also presented our work on EVCs, a flow-control technique which helps reduce router overhead in packet-switched network designs. In the future, we plan to do a more detailed system-level evaluation of EVCs using the GARNET framework. We are also in the process of designing a fast and scalable cache coherence protocol for CMP systems which leverages network flow control techniques.

Acknowledgments

This work was supported by NSF under Grant No. CNS-0613074, a grant from Intel Corporation, an Intel PhD Fellowship, MARCO Gigascale Systems Research Center, and Alfred P. Sloan Research Foundation.

References

- [1] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. Int. Symp. Computer Architecture*, pages 83–94, 2000.
- [2] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Pub., San Francisco, CA, 2003.
- [3] W. J. Dally. Virtual-channel flow control. In *Proc. Int. Symp. Computer Architecture*, pages 60–68, May 1990.
- [4] M. Gallet. Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER chip. In *Proc. Hot Interconnects 4*, pages 141–146, Aug. 1996.
- [5] 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.
- [6] 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.
- [7] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Almadeen, K. Moore, M. Hill, and D. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. In *Computer Architecture News*, 2005.
- [8] 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.
- [9] Virutech. Simics full system simulator. <http://www.simics.com/>.
- [10] 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.