A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator

Nan Jiang
Stanford University
qtedq@cva.stanford.edu

Daniel U. Becker
Stanford University
dub@cva.stanford.edu

George Michelogiannakis
Lawrence Berkeley National Lab
mihelog@lbl.gov

James Balfour
Google Inc.
jbalfour@google.com

Brian Towles
D.E. Shaw
btowles@deshaw.com

John Kim
KAIST
jjk12@kaist.edu

William J. Dally
NVIDIA Research/Stanford University
dally@cva.stanford.edu

Abstract—**Network-on-Chips (NoCs) are becoming integral parts of modern microprocessors as the number of cores and modules integrated on a single chip continues to increase. Research and development of future NoC technology relies on accurate modeling and simulations to evaluate the performance impact and analyze the cost of novel NoC architectures. In this work, we present BookSim, a cycle-accurate simulator for NoCs. The simulator is designed for simulation flexibility and accurate modeling of network components. It features a modular design and offers a large set of configurable network parameters in terms of topology, routing algorithm, flow control, and router microarchitecture, including buffer management and allocation schemes. BookSim furthermore emphasizes detailed implementations of network components that accurately model the behavior of actual hardware. We have validated the accuracy of the simulator against RTL implementations of NoC routers.**

I. Introduction

The interconnection network is a critical part of any modern computer system, including large-scale multiprocessors in a supercomputer, single-chip many-core processors, or a system-on-a-chip (SoC) for mobile devices. As the number of components in a system continues to increase, the interconnection network will have a more significant impact on the overall system performance and cost. Thus, interconnection networks need to be properly modeled and evaluated to understand their performance characteristics, bottlenecks, and the impact on the overall system.

In this work, we describe the ***BookSim*** network simulator—a detailed, cycle-accurate simulator for Network-on-Chips (NoCs) that can also be used to model interconnection networks for a variety of other systems. The original simulator (BookSim1) was released as part of an interconnection network textbook by Dally and Towles [1] and was used to generate the performance graphs in the textbook. BookSim1 was a generic network simulator that did not specifically target the on-chip environment. As a result, it has been widely used for research in many network contexts, including networks found in large-scale supercomputers and many-core processors. BookSim has been used to study many different aspects of network design, including topology [2], [3], routing [4], flow control [5], [6], router microarchitecture [7], quality-of-service [8], as well as new technologies such as nanophotonics [9], [10]. BookSim can also be incorporated into other system simulators to model the network; for example, GPGPU-sim [11], a many-core accelerator simulator for evaluating GPGPU workloads, leverages BookSim to model the on-chip communication.

From a system simulation perspective, BookSim provides the flexibility that is needed in a high-level simulator. All major network components are parameterized to allow rapid sweeps of the network design space. From a network design perspective, BookSim provides detailed modeling of all key components of a network router. The simulator is designed to be modular and to facilitate modifications and the addition of new network features. BookSim is also structured to reflect actual network design—for example, communication between neighboring routers needs to occur through a "channel," rather than a global variable or a data structure. This approach forces simulator users to think about the network as a physical entity before implementing new features in the simulator.

Despite the flexibility of the original BookSim, it did not support some of the more advanced features and topologies proposed in the context of on-chip networks. As a result, an updated simulator—BookSim2—was developed that includes many changes that better reflect the state-of-the-art in on-chip network research. In particular, BookSim2 features more detailed modeling of the router microarchitecture, models inter-router channel delay, and provides support for additional traffic models. To validate the accuracy of the simulator, we have compared its simulation results to an RTL network-on-chip router [7], [12], [13]. The comparison shows that the latency-throughput characteristics of BookSim2 closely match those of the RTL model. We also show how BookSim2 can be used to gain insights about other important aspects of network design; for example, it can be used to investigate the effect of age-based priority on the performance of separable allocators or the performance impact of different router pipeline optimization techniques.

In summary, this papers makes the following contributions:

- We describe the BookSim network simulator that provides a large degree of flexibility and modeling fidelity for the evaluation of novel network designs.
- We describe additional changes introduced in the recent BookSim2 release, including detailed router microarchitecture for on-chip network evaluation and analysis. To the best of our knowledge, this is one of the first works that validate the results of a network simulator against an actual RTL implementation of a router.
- Using the detailed network simulator, we show the impact of accurately modeling the router pipeline on network performance. We also study the impact of age-based arbitration and show its positive and negative effects on the network's saturation throughput.

The remainder of the paper is organized as follows: Section II provides background on interconnection networks and other network simulators. Section III describes the details of the original BookSim1 simulator. New features added to BookSim2 are described in Section IV. Section V presents several case studies conducted using BookSim and validates the accuracy of the simulator against an RTL implementation of a network-on-chip router. We conclude the paper in Section VI.

II. Background

A. Interconnection Networks

Any interconnection network can be characterized by the following basic properties: topology, routing, flow control, and router microarchitecture [1].

Topology: Defines how the channels, routers, and endpoints are interconnected.

Routing: Determines which path a packet takes from its source to its destination.

Flow Control: Determines how shared resources, such as buffers and channel bandwidth, are utilized when contention occurs.

Router Microarchitecture: Defines the internal organization of routers, including the buffers, crossbar, allocators, etc.

Each of these properties affects the performance of the network.

A common metric used to measure network performance is *saturation throughput*, defined as the network throughput at which the first channel saturates [1]. *Zero-load latency* is another key performance metric that represents a lower bound on the network latency. These metrics can be estimated analytically or obtained from the latency-throughput curves produced by simulations, as shown in Figure 1.

Each of the network's properties establishes a bound on its performance: Given the network channel bandwidth, the

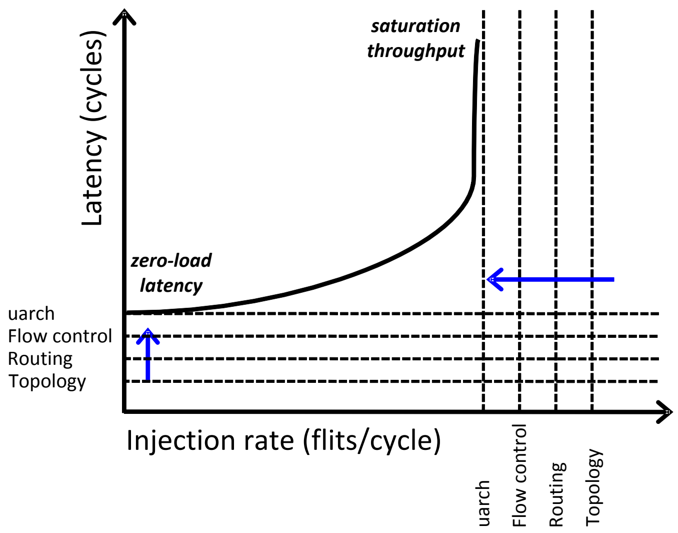


Fig. 1. Latency-throughput curve used to measure performance of interconnection networks.

topology imposes an initial throughput and latency bound, governed by the amount of bisection bandwidth in the network and the network diameter. For each traffic pattern, the routing algorithm maps the communication onto the network topology and further limits how closely the topology bounds can be approached in practice. The routing algorithm also offers different trade-offs for achievable performance; e.g., non-minimal routing can increase the zero-load latency while—if done properly—increasing the throughput on adversarial traffic patterns. Finally, flow control and router microarchitecture introduce router efficiency and contention effects that can further limit the network performance.

B. Related Work

Most full-system simulators provide a way to model network traffic; however, network models used in this context often emphasize simplicity and speed over simulation fidelity, e.g. by not accurately modeling contention in the network and using a fixed latency value regardless of network conditions. While this limited fidelity is appropriate for some areas of study, it is insufficient for network research. Various network simulators have been developed in order to address this problem. Garnet [14] is a detailed network simulator that was incorporated into the GEMS (now GEM5) full-system simulator [15] and is also available as a standalone network simulator. Noxim [16] is a network-on-chip simulator implemented in SystemC. SICOSYS [17] is a network simulator specifically targeted at multiprocessor systems. While, many of these previously proposed simulators share similar characteristics, they typically provide only limited flexibility such as limited choice of topologies or limited configurability of individual network components. In contrast, BookSim supports a wide variety of parameterized topologies, routing functions, traffic loads and router components. In addition, we show how our

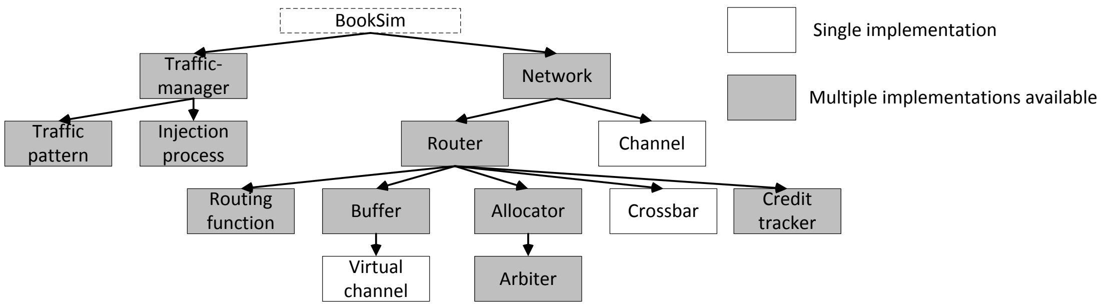


Fig. 2. Module hierarchy of the simulator.

detailed network simulator very accurately matches the results of RTL model of a canonical NoC router.

One important aspect of any simulator is the simulation speed. Compared to some of the other simulators, BookSim can be slower; for example, Garnet [14] is implemented as an event-driven simulator and can be faster than BookSim for low to medium network load. However, compared to a multiprocessor system simulator or a full-system simulator, network simulation is often several orders of magnitude faster and is not necessarily the bottleneck.

III. BookSim1

A. Simulator Overview

BookSim's flexibility comes from its highly modular implementation. The simulator is composed of a hierarchy of modules that implements different functionalities of the network and simulation environment. A hierarchical view of the major simulator modules is shown in Figure 2. Each of these modules has a well-defined interface that facilitates replacement and customization of module implementations without affecting other parts of the simulated system.

A functional top-level block diagram of BookSim is shown in Figure 3. The top level modules of the simulator are the *trafficmanager* and the *network*. The *trafficmanager* is the wrapper around the network being evaluated and models the source and destination endpoints. It injects packets into the network according to the user-specified configuration, including the traffic pattern, packet size, injection rate, etc. To properly model network behavior beyond the point of saturation in open-loop simulations, an infinite source queue[1] is implemented at the injection nodes to ensure that latency measurements properly account for source queuing delay and head-of-line blocking effects. The *trafficmanager* is also responsible for ejecting packets from the destination endpoints, collecting appropriate statistics, and terminating the simulation.

The *network* top level module comprises a collection of *routers* and *channels*, with the topology defining how these modules are interconnected. All communication between neighboring routers occurs through explicit send and receive

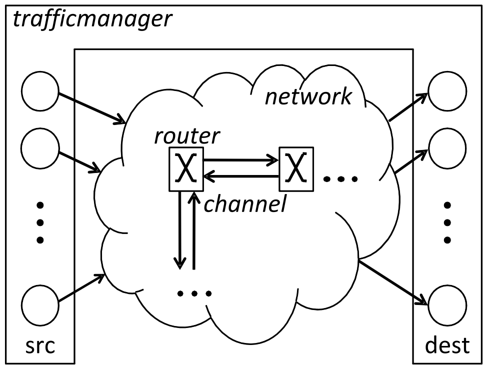


Fig. 3. Top-level block diagram of the simulator.

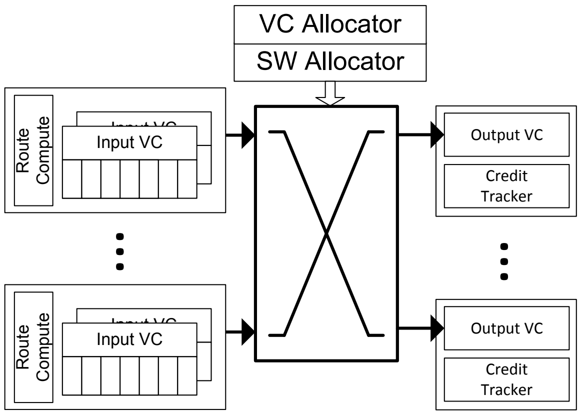
operations across connecting channels, rather than by updating global variables or data structures. The simulator assumes that credit-based flow control is used for buffer management between adjacent routers and uses a separate, dedicated channel to communicate credit information; i.e., each network channel is accompanied by a *credit* channel in the opposite direction.

At the lowest level, BookSim simulates the network on the granularity of flits and clock cycles. A packet consists of one or more flits or *flow control digits*, the smallest unit of channel and buffer allocation. The width of all channels and router pipelines always corresponds to the width of a single flit. During a clock cycle, a network channel can read a single flit from its input and also write a single flit to its output. Studies that require varying the channel width of the network can be conducted by varying the number of flits per packet. For example, when simulating two networks with 64-bit and 32-bit wide channels, transmission of a 512-bit packet requires eight flits on the former network and 16 flits on the latter.

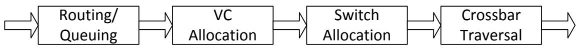
B. Router Pipeline Model

The primary router model in BookSim is the input-queued virtual channel router. The major router components and the canonical four-stage pipeline for packet header flits are shown in Figure 4. This roughly corresponds to the modular organization and program flow of the simulator. Arbitrary delays can be assigned to each pipeline stage, and the entire router can be configured to mimic the behavior of a single cycle router.

[1]The simulator models the behavior of an infinite queue by allowing each injection node to operate independently. Since only the head of the source queue is injected into the network, the injection of each source is allowed to *lag* behind and enable an implementation with a finite-size queue [1].



(a) Component diagram



(b) Pipeline stages

Fig. 4. BookSim's input-queued router model.

BookSim offers multiple configuration options to overcome the throughput limitations that head-of-line blocking imposes on input-queued routers. One such option allows the speed of the router's internal pipeline to be increased relative to the network channels. With a router speedup of 1.5, the router's pipeline runs three times for every two channel/trafficmanager cycles, directly increasing the router's throughput relative to the rest of the network. This is useful when the focus of the simulation is not on the router microarchitecture, but on other network properties (e.g., the routing algorithm). For the router to behave correctly with speedup, additional buffering is added at the router outputs.

The configuration of the router's crossbar can also be adjusted to allow for non-square crossbar configurations. With a crossbar input speedup of two, each router input port has access to a pair of inputs into the crossbar, with input Virtual Channels (VCs) statically assigned to each individual input. In conjunction with a sufficiently large number of VCs, additional crossbar inputs can thus reduce the effect of head-of-line blocking. For a k-port router, an input speedup of k (with k virtual channels) effectively mimics the behavior of an output queued router.

BookSim furthermore supports atomic VC allocation that eliminates head-of-line blocking. In regular VC allocation, a VC becomes available for allocation when the tail flit of the packet currently holding the VC departs the router. With atomic VC allocation, a VC cannot be reallocated until the credit for the previous tail flit is received by the router. This ensures that each VC only holds a single packet at any given time and thereby avoids dependencies between successive packets in the network. While regular VC allocation can provide higher throughput with a limited number of VCs, atomic VC allocation can provide better performance with large numbers of VCs due to the elimination of head-of-line blocking.

C. *Simulator Configuration*

The parameters used in a simulation are specified in a configuration file; parameters that are not explicitly specified use default values. A set of exemplary parameters is listed in Table I. At the top level, a network in BookSim is defined by the topology and its associated parameters. During network initialization, basic network characteristics are automatically derived based on the topology; this includes the network size, number of routers and channels, the number of ports for each router, and the connectivity of the routers and channels. All routers are configured identically based on flow control, routing, and microarchitecture parameters. For common network topologies, route computation at each router is done algorithmically by a routing function based on the network connectivity. The routing computation has sufficient access to information about the router's buffer/credit occupancy to facilitate the implementation of adaptive routing algorithms.

IV. BookSim2

The current version of the BookSim simulator—BookSim2—improves upon various aspects BookSim1 while incorporating new features for simulating NoCs, which we describe in the following sections. The simulator maintains its cycle-accurate nature, and a greater emphasis is placed on the detailed modeling of network components based on realistic hardware implementations. A large part of the revision effort has focused on improving the model of the Input-Queued (IQ) router microarchitecture, with the aim of facilitating more accurate simulation in a NoC context. While the basic structure of the IQ router remains the same as shown earlier in Figure 4a, features and optimizations have been added to reflect current trends in NoC research. Other modifications to the simulator include increased flexibility for traffic generation and integration with other traffic sources.One addition to BookSim2 that significantly affected results compared to BookSim1 is accurate modeling of channel latencies. Channel latency affects many aspects of network performance, including the utilization of buffers, credit round-trip latency, as well as the propagation delay for congestion information in adaptive routing algorithms. Therefore, depending on the length of the channels and the signaling assumptions, the channel latency can have a significant impact on performance.

A. *Router Microarchitecture Modeling*

Many of the updates to the IQ router model were directly inspired by the development of an Open-Source NoC Router RTL [7], [12], [13]. While BookSim1 aimed to avoid software constructs that would be impractical to implement in hardware, it does so primarily at the network level, e.g. by avoiding the use of global state shared among multiple network components. In contrast, its fidelity at the microarchitecture level, particularly in the router pipeline, is somewhat limited. The router pipeline is implemented by tagging each input VC with a pipeline state and a timestamp that indicates when the VC is ready for the next step of execution. In each clock cycle, all

TABLE I
SOME REPRESENTATIVE PARAMETERS IN BOOKSIM SIMULATOR. THESE PARAMETERS ARE VALID IN BOTH 1.0 AND 2.0 RELEASE.

BookSim parameters	Description
Topology	
topology	Specifies the network connectivity based on well known topologies (e.g., mesh, butterfly)
k, n	Specify the network size and configuration, for the selected topology.
Routing	
routing_function	Determines the routing algorithm for the selected topology (e.g., dimension order)
Flow Control	
num_vcs	Number of virtual channels per physical channel
vc_buf_size	Input buffer size of each virtual channel
wait_for_tail_credit	Enable atomic VC allocation
Router Microarchitecture	
vc_allocator, sw_allocator	The type of allocator used for switch and virtual channel allocation
credit_delay, routing_delay, vc_alloc_delay, etc.	Latency parameters for the router pipeline
input_speedup, output_speedup, internal_speedup	Speedup in the crossbar and router pipeline compared to network channels.
Traffic	
traffic_pattern	Synthetic traffic pattern used in the simulation (e.g., uniform, transpose, etc.)
injection_rate	Average injection rate for each node

pipeline stages are executed sequentially. Since the execution of each pipeline stage can update the internal state of the router directly, this can lead to unintended propagation of information between pipeline stages in cases where multiple stages should be executed concurrently. As an example, in the case of speculative switch allocation, the VC and switch allocation stage are executed independently and in parallel; however, the simulator actually executes both of them successively and updates the router's internal state after each stage. As a result, the switch allocator is aware of whether or not VC allocation succeeded or not, even though the results of the latter would not be available until the end of the cycle in a real router, and therefore unrealistically avoids cases of misspeculation in which a crossbar time slot is assigned speculatively but no VC is secured.

In order to address these issues, the router model in BookSim2 was modified to use a two-phase protocol for updating all router state. In the evaluation phase, which loosely corresponds to combinational logic in a hardware implementation, only read accesses to the routers' internal state are permitted. The result of the evaluation phase is a set of state updates, each of which is tagged with the time at which it takes effect. Once the evaluation phase is completed for all pipeline stages of all routers, the simulator enters the update phase, in which the routers' internal state is modified to reflect any such updates that are due in the current cycle. This evaluate-update protocol enforces clean clock cycle boundaries and avoids cases in which unintentional serialization is introduced where a parallel implementation was intended.

B. Pipeline Optimizations

Packet latency often has a direct impact on overall performance. In network topologies with a large network diameter, such as the mesh topology, router latency represents a significant fraction of the overall network latency. Many NoC research efforts have thus focused on optimizing the router pipeline to reduce latency. While the original BookSim can vary router latency by changing the delay associated with each pipeline stage, this does not represent an accurate model of realistic hardware implementations. Using zero-latency pipeline stages also does not capture the performance impacts of the various pipeline optimizations. In BookSim2, many pipeline optimization techniques for IQ routers have been implemented in full detail, providing the desired improvement in router latency without degrading the accuracy of the simulation results.

Look-ahead routing is commonly used to eliminate the delay associated with route computation from the router's critical path. In networks with a well defined topology, routing information for the current router can be precomputed at the upstream router. Newly arrived packets are immediately eligible for VC allocation, and look-ahead routing for the next hop can be performed in parallel to allocation. For most network configurations, route computation does not have side effects on other packets in the router. As a result, we can emulate the effect of look-ahead routing by setting the delay of the routing stage to zero without compromising accuracy in many scenarios. However, this approach can lead to inaccurate results when adaptive routing is used to make routing decisions based on the current network state: In such cases, a realistic model must account for the fact that propagating congestion information to the upstream router incurs delay, and that adaptive routing decisions must therefore be made based on a slightly outdated view of the network state.

The delay of the VC allocation pipeline stage can effectively be hidden by employing speculative switch allocation [18]. In routers that utilize this technique, a packet requesting VC allocation also sends a request to the switch allocator, *speculating* that a VC will be granted. If both types of allocation succeed, the packet effectively performs VC and switch allocation in parallel, shortening the packet's effective pipeline traversal delay. In case VC allocation is not successful, any speculatively assigned crossbar time slot goes unused.

Using speculative switch allocation is especially useful for reducing router latency at low network loads, where very little contention occurs in both allocators and any speculative request is likely to succeed.

As crossbar times slots assigned to a speculative request go unused if VC allocation fails, it is necessary to prioritize non-speculative requests—which will always utilize assigned crossbar time slots—during switch allocation in order to avoid performance degradation. Prior research efforts have proposed several methods to guarantee that non-speculative requests take priority over speculative ones [7], [18]. BookSim2 supports three different mechanisms: A simple method is to statically assign higher priority to non-speculative requests and to use BookSim's priority-aware allocators for switch allocation. With a winner-takes-all priority allocator, the highest-priority request always wins, eliminating all speculative requests that conflict with non-speculative requests. BookSim also supports the canonical implementation described in [18], which uses separate allocators for speculative and non-speculative requests; the resulting grants from the two allocators are merged at the end of allocation, and any speculative grants that conflict with non-speculative grants are discarded. Alternately, speculative grants can be masked pessimistically based on non-speculative *requests*, which are available earlier in the clock cycle [7]; this method can be used to reduce the switch allocator's critical path delay without significantly degrading speculation efficiency.

The VC allocation pipeline stage can also be removed entirely by employing *combined VC and switch allocation* [19]. To do so, head flits participate in switch allocation, and any winning head flits are assigned a free output VC if one is available. Similarly to speculative switch allocation, such requests must be handled with lower priority than requests from body and tail flits in order to avoid performance degradation due to unused crossbar time slots. In total, BookSim2 features four different methods of performing VC and switch allocation in the same cycle. In combination with look-ahead routing, this can reduce the router pipeline delay from four to two cycles.

BookSim2 retains the original simulator's ability to configure arbitrary delays for each pipeline stage. A single-cycle router can be created by setting the switch traversal delay to zero in combination with the optimizations described above. However, this is not feasible to implement in hardware, as it requires that switch allocation and switch traversal execute sequentially in a single cycle, creating a long critical path. Thus, for realistic simulations, BookSim2 has a minimum router latency of 2 cycles.

C. Allocators and Prioritization

BookSim supports a wide variety of allocator implementations for both VC and switch allocation, including iSLIP [20], lonely output, parallel iterative matching [21], maximum matching, and wavefront allocators [22]. Each of these designs represents a different trade-off between implementation complexity and quality of allocation. For NoC simulations, customizable separable allocators have been added to the collection. Contemporary NoC designs commonly employ such allocators as their principle of operation—decomposing allocation into multiple stages of arbitration—facilitates simple hardware implementations with comparatively low delay; however, this is achieved at the expense of reduced allocation quality [1].

In general, separable allocators are implemented using two stages of arbitration. A first stage of arbiters—one per input—ensures that at most a single output is granted to each input, while a second stage ensures that each output is granted to at most one input. These two stages can be completed in either order, corresponding to input- and output-first allocation. BookSim2 allows such allocators to be implemented from multiple different types of arbiters; available designs include matrix arbiters, round-robin arbiters, weighted-round robin arbiters, as well as tree arbiters constructed from any of the aforementioned elementary arbiter types. All arbiter designs are priority-aware and thus support the construction of priority-aware allocators.

By default, BookSim2 also supports a variety of packet prioritization schemes. Specifically, the user can either manually specify a static priority for each traffic class, or priorities can be dynamically assigned to packets based on various network factors. For example, age-based priority dynamically calculates each packet's priority value based on the number of cycles that have elapsed since the packet was generated [23]. When a packet is at the head of a VC, its output request is tagged with its current priority before being sent to the allocators. To avoid priority inversion, BookSim2 also supports a simple priority donation scheme in the input VCs, in which the head-of-line flit in each buffer is assigned the highest priority of any packet currently stored that buffer.

D. Dynamic Input Buffer Management

Prior research has shown that input buffers dominate the area and power of NoC routers [24], [25]. Efficient utilization of the buffer resources is thus critical in minimizing the total cost of the network for a given set of performance constraints. Most NoC routers that employ VCs implement simple static buffer management schemes where each VC is assigned a fixed number of buffer slots. The amount of buffer storage for each VC is chosen to be sufficiently large to cover the credit round-trip latency, allowing the VC to sustain full throughput on the physical channels without incurring credit stalls. However, depending on the number of VCs and their assignment, this can cause a large fraction of the available buffer space to remain empty even under heavy load and thus lead to significant under-utilization of a costly resource.

Dynamic buffer management schemes—commonly referred to as buffer sharing—represent one approach for improving buffer utilization by allowing multiple VCs to share a common pool of buffer resources. BookSim2 adds support for such schemes, as well as hybrid schemes where only part of the buffer capacity is available for sharing. While a variety of different hardware implementations have been proposed in prior work [26], [27], the implementation differences between them typically do not manifest in different network behavior and are thus transparent to the simulator; as such, BookSim2 can accurately model most such implementations.

BookSim2 furthermore supports several schemes for regulating buffer sharing in an effort to avoid performance pathologies that can result from allowing buffer space to be shared freely [6]. With credit-based flow control, such schemes can be implemented by restricting credit availability at the routers' output ports. To this end, BookSim2 adds a modular facility for implementing sharing policies to the credit tracking mechanism, allowing the maximum share of credits available to each VC to be assigned either statically or dynamically based on network conditions. Several exemplary dynamic sharing policies are included with BookSim2; for example, one simple policy monitors the number of active VCs at each output and divides the credit pool evenly among them. A more advanced sharing policy tracks the level of the congestion for each active output VCs and allocates a larger share of buffer space to less congested VCs with the goal of avoiding inefficient buffer occupancy by blocked flits [6].

E. Flexible Synthetic Traffic Simulation

In the early stages of network evaluation, it is faster and often more insightful to test designs using synthetic traffic patterns. Synthetic traffic can systematically stress specific parts of the network and thus help exercise particular corner cases. BookSim1 supported a variety of well-known synthetic traffic patterns, as well as a facility for randomly generating permutation patterns. In BookSim2, we have expanded the flexibility of synthetic traffic simulations by allowing arbitrary combinations of synthetic traffic patterns. A combined traffic pattern is created by specifying a list of individual synthetic traffic patterns, each with separately configurable injection rates and packet sizes. At runtime, packets are injected into the network by randomly choosing one of the specified subpatterns for each packet based on the relative injection rates. This feature allows BookSim2 to simulate an arbitrarily diverse set of synthetic traffic patterns and can help reveal interactions between different patterns. The traffic composition of the combined pattern is only visible at injection time; in particular, routers cannot distinguish between the individual components that make up a combined pattern.

The simulator also supports flexible configuration of traffic classes: each class can specify its individual packet size, injection rate, injection model, priority scheme, and traffic pattern (including the combined traffic pattern described earlier). Packets from different traffic classes are generated and queued independently at the traffic sources. By default, packets from different classes arbitrate for network injection in a round-robin fashion. Traffic class IDs are carried in the packet header, enabling different classes to be handled differently in the routers. This enables the user to test networks that offer differentiated services based on traffic classes.

Finally, BookSim2 also supports simulations using multiple subnetworks: Instead of assigning packets from different traffic classes to different ranges of VCs within a single network, this allows traffic classes to be transported on separate physical networks, eliminating any interaction or contention. This approach has been shown to be attractive for many NoC use cases both in research and in the industry [28], [29]. In the simulator, subnetworks can be declared independently with different network parameters such as topology and buffer sizes. The subnetworks only share the injection and ejection modules, and they otherwise operate completely independently. By default, the simulator assigns each packet to a subnetwork either randomly or based on its traffic class; however, other specialized subnetwork assignment schemes can be easily implemented by modifying the injection modules.

F. Alternative Traffic Models

In addition to traditional open-loop synthetic traffic simulations, the traffic manager in BookSim2 has been extended to support closed-loop modeling with synthetic traffic, as well as generating NoC traffic by interfacing with a full-system simulator or by replaying traffic traces.

BookSim2 introduces support for conducting *closed-loop* experiments with synthetic traffic by modeling request-reply traffic. Closed-loop evaluations can be more representative of system performance, as the injection rate of packets is influenced by the network load. The injection modules use finite queuing and limit the maximum number of outstanding requests; when the maximum number of outstanding requests is reached, the injection module stalls and no additional requests are injected until one or more requests complete. This mechanism can be used to approximate the behavior of a processor-memory network in which processors have a finite number of Miss Status Handling Registers (MSHRs); once all of a processor's MSHRs are in use, it stalls on the next memory request until one of the outstanding requests completes. Differences and similarities between open-loop and closed-loop network evaluation with BookSim have been studied earlier in [30].

In order to facilitate simulations of realistic on-chip workloads, BookSim2 supports both closed-loop and open-loop full-system simulations by either interfacing with a full-system simulator or playing back previously generated traces, respectively. An earlier version of BookSim was used to model the on-chip network for the GPGPU-sim simulator [11]. Additionally, BookSim2 offers an interface that can be integrated with the GEM5 [15] full-system simulator, replacing Garnet [14] or the simple network model included with GEM5.

The injection module can also be programmed to inject traffic based on traces that were generated offline by full-system simulation or other means. At the cost of reduced fidelity, this allows for much higher simulation speed compared to using full-system simulation. BookSim2 supports traces in a simple ad-hoc format that lists each packet injection and ejection along with its source and destination node. Additionally, it supports pseudo-closed-loop simulation using dependency-annotated traces; this functionality leverages the *Netrace* library developed at UT Austin [31] and represents a middle ground in terms of fidelity and performance between traditional trace-based simulation and full-system simulation.

V. Evaluation

In this section, we provide several demonstrative results obtained using the BookSim2 network simulator. We first show that results generated by BookSim2 accurately match the behavior of RTL implementations of NoC routers (Section V-A). We then show how the router pipeline optimizations that were described in Section IV impact network performance (Section V-B). Finally, we present a case study that evaluates the impact of age-based priority on the throughput of a mesh network using separable allocators (Section V-C).

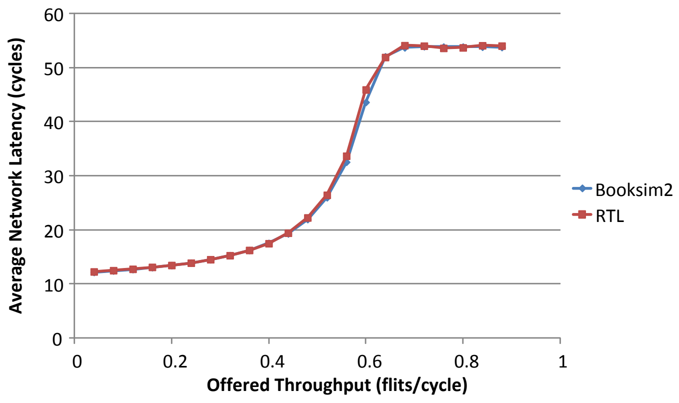
A. Validation with RTL Router Implementation

The design of BookSim2 places significant emphasis on hardware realistic implementation of an IQ NoC router. We can demonstrate this by comparing the results generated from the software simulator with RTL simulations using the same set of network configurations. The network we simulated is a 3×3 mesh network with a single VC and 16-flit input buffers. Both switch and VC allocation use separable input-first allocators with round-robin arbiters. Speculative switch allocation is disabled. The networks are running uniform random traffic patterns with a packet size of four flits. Network latency and throughput statistics are collected over a period of 100K cycles per simulation. We are limited to simulating small NoCs due to the time and resource constraints of the RTL simulation; however, the results are applicable to larger network configurations as well.

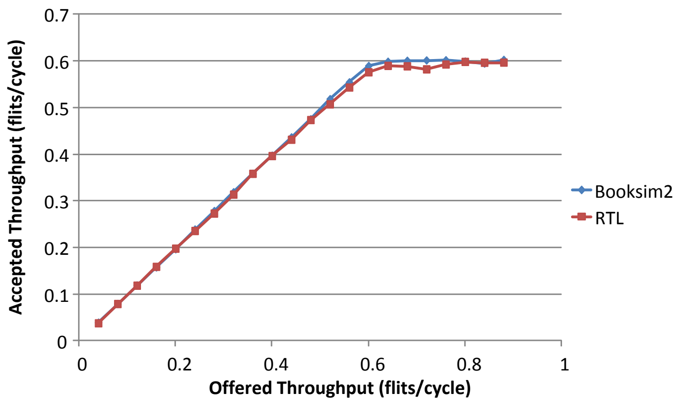
Figure 5 shows the accepted throughput and in-network latency of the two simulated networks as we varied the injection rate. Overall, the simulations using BookSim and the RTL exhibit nearly identical performance. The network latency in Figure 5a measures only the delay inside the network; i.e., it does not include source queuing delay. The results show a maximum difference of 5% in network latency measurements. The latency plateau at 60% load shows when both networks have reached saturation, when nearly all buffers in the network have become saturated and network queuing delay becomes constant. When networks are below saturation, by definition the accepted throughput is equal to the injection rate as shown by the linear region of Figure 5b. After the onset of saturation, the accepted throughput plateaus as the network reaches its maximum capacity. The result shows a maximum difference of 3% in the accepted throughput of the two simulated networks. Network throughput is largely determined by the matching efficiency of the allocators, showing that the allocator implementation in BookSim and in the RTL exhibit nearly identical behavior.

B. Impact of Router Pipeline Optimization

The router pipeline optimizations included in BookSim2 not only reduce the router traversal latency, but can also have an impact on the network's saturation throughput. This impact could not be observed in the original BookSim because



(a) Network Latency



(b) Minimum Accepted Throughput

Fig. 5. Network latency and throughput comparison of a 3×3 mesh network using the software simulation and RTL simulation.

its router latency reduction is simulated by simply merging pipeline stages without any change to the underlying router logic.

Figure 6 compares the the performance of 2-cycle routers implemented in the two versions of BookSim and a baseline network using 3-cycle routers. For BookSim1, we set the pipeline delay of routing computation and VC allocation to zero. A packet arriving at a router performs routing computation, VC allocation, and switch allocation sequentially in the same cycle before traversing the switch in the second cycle. For BookSim2, one network uses speculative switch allocation and the other network uses the combined allocation technique described in the previous chapter. For the speculative network, separate allocators are used to process speculative and non-speculative requests. After allocation, the resulting speculative grants are masked by non-speculative grants. All networks are 64-node meshes with two VCs, each with a 16-flit input buffer. Network channel traversal takes a single cycle. We apply a uniform random traffic pattern with a packet size of 4 flits.

At low network load, the latencies for the three networks that use 2-cycle routers are identical and 20% lower than the latency of the baseline. This is a direct result of the reduced router latency, which particularly affects high-diameter network topologies like the mesh. For the speculative net-

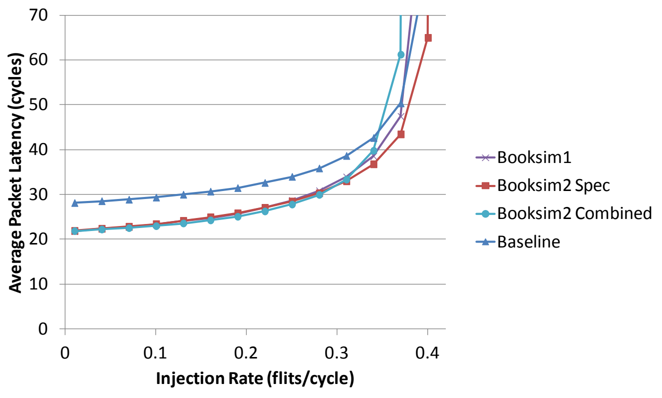


Fig. 6. Latency-throughput of mesh networks using 2-cycle routers in BookSim1 and BookSim2 compared to a baseline network with 3-cycle routers. The network saturation throughput varies depending on the pipeline optimization method used.

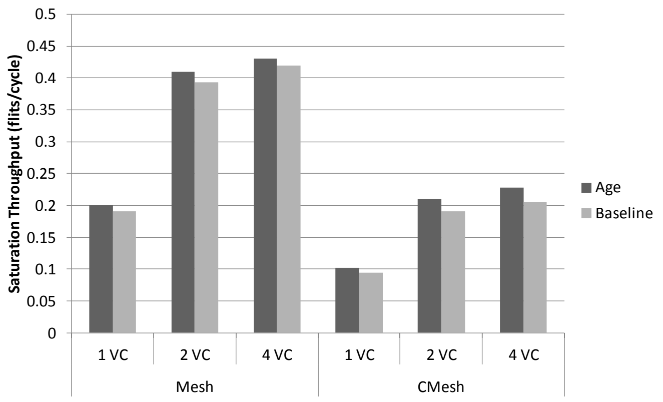


Fig. 7. Effect of age-based priority on the saturation throughput of 64-node mesh and concentrated mesh networks for different numbers of VCs with DOR.

work, very little allocator contention exists at low loads, and consequently speculative requests are nearly always granted. As network load increases, the latency of the speculative network remains consistently lower than the non-speculative baseline and on par with the latencies of the combined and the BookSim1 networks. This indicates that speculation remains effective even for higher network loads. Because speculative switch requests have a lower priority than non-speculative requests, networks with speculative switch allocation should never perform worse than the baseline.

In terms of throughput, the network using 2-cycle BookSim1 routers has the same saturation throughput as the 3-cycle router. This shows that these two networks are essentially identical, with the exception of artificially lowered network latency. When using speculative 2-cycle routers, the saturation throughput of the network is increased by 3% compared to baseline and BookSim1. This is because the speculative router in essence has two switch allocators operating in parallel. If the grants generated by the two allocators do not conflict, then the effective matching efficiency of the two allocators is higher than that of a single switch allocator. In contrast, the saturation throughput of the combined allocation network is 2% lower than the baseline. This is because packets are never assigned a VC until they can win speculative switch allocation. This effectively reduces the number of requests handled by the non-speculative allocator and thus the number of matchings generated. In addition, a speculative conflict in combined allocation not only cancels the switch grant but also effectively the VC grant. The opportunity cost of a speculative conflict is greater for combined allocation than for speculative switch allocation with separate VC allocation.

This example shows that detailed microarchitectural modeling is important for the purpose of network research. For a high level simulation, it can be sufficient to use the simple and faster BookSim1 approach to router latency reduction. But the detailed model offered by BookSim2 can reveal key insights about the interactions of router components.

C. Case Study: Impact of Age-Based Priority on Network Performance

By using round-robin arbitration, separable allocators can provide strong local fairness between a router's inputs. However, as multiple traffic flows from different sources share the same router input, this is not sufficient to guarantee strong fairness between traffic flows. As a result, adversarial traffic patterns can lead to global unfairness and starvation. Age-based priority [23] represents one way of achieving global fairness in NoCs. It achieves this by prioritizing packets in the network based on the time they were created. During each arbitration step of the separable allocator, each arbiter issues a grant to the request from the oldest packet.

Studies have shown that using age can resolve the global fairness problem of a locally fair allocator [8], [32]. However, using BookSim2, we show that age-based priority also affects a network's saturation throughput under uniform, non-adversarial traffic. These effects are caused by how age-based priorities interact with the separable allocators' matching decisions. Figure 7 shows the effects of age-based priority on the saturation throughput of 64-node mesh and Concentrated Mesh (CMesh) networks using Dimension order routing (DOR) under uniform random traffic. The networks all use separable input-first allocators for both switch and VC allocation. For each configuration, age-based allocation provides a 2-4% increase in network saturation throughput compared to the baseline. Under uniform random traffic, the routers in the center of a mesh network are the most heavily loaded. With DOR, the majority of the network load in these routers is caused by traffic traveling along each dimension. Consequently, the network throughput is limited by the throughput of the east-west and north-south through-traffic in the central routers. The node injection traffic at these routers competes with the critical through-traffic for output bandwidth. Age-based allocators tend to service the older through-traffic more quickly than the locally injected traffic, resulting in higher saturation throughput.

Age-based allocation can also adversely affect performance

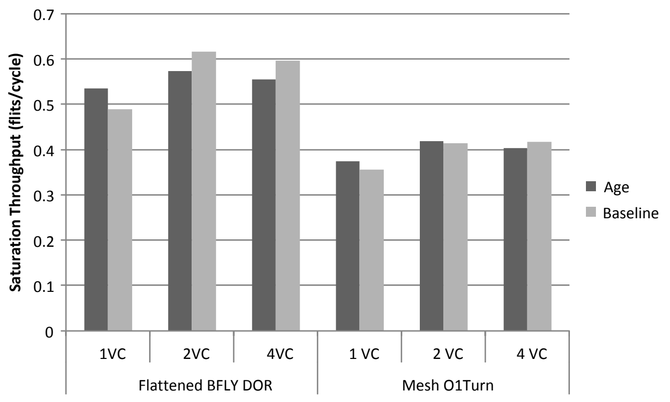


Fig. 8. Effect of age-based priority on the saturation throughput of 64-node Flattened Butterfly and mesh networks.

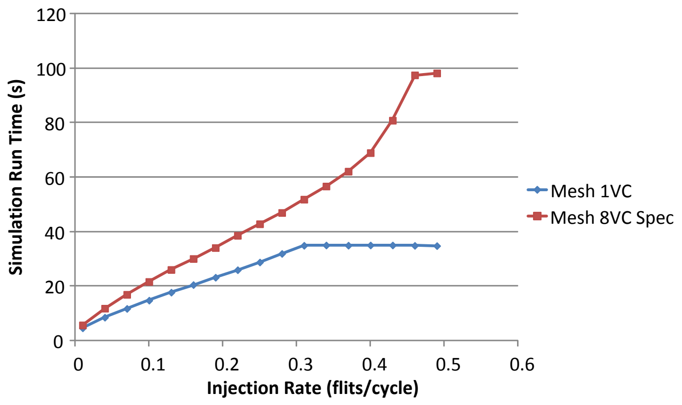


Fig. 9. BookSim2 simulation run time. The speed of the simulator is determined by the complexity of the network configuration and the activity in the network.

for certain network configurations. The constraints imposed by packet priorities can reduce the separable allocators' matching efficiency, and therefore lower the saturation throughput of the network. Figure 8 shows the effect of age-based priority on the saturation throughput of 64-node flattened butterfly and mesh networks under uniform random traffic. In this experiment, the mesh networks use O1TURN routing [33], where the order of dimension traversal for DOR is selected at random for each packet. For simulations with a single VC, both topologies show a higher saturation throughput when using age-based priority. This improvement is due to the prioritization of the through-traffic as described previously. However, increasing the number of VCs causes age-based networks to experience performance degradation compared to the baseline network with no priority.

Matching inefficiencies in separable allocators arise due to the lack of coordination between arbiters within each arbitration stage; for example, multiple arbiters in the input stage of a separable input-first allocator can independently pick the same output port even though other non-conflicting requests exist. As a result, only one port will receive a grant from the output stage, leaving the others idle. For age-based allocators, this problem is further compounded by the fact that any requests that were not granted because of output conflicts will be re-issued in the next cycle with their age increased by one, making these requests more likely to be selected again by the input stage. This creates a positive feedback loop which biases the input arbitration towards selecting requests that are unlikely to succeed in output arbitration, leading to a decrease in matching efficiency.

D. Simulator Performance

The detailed modeling provided by BookSim2 does have some negative impact on the performance of the simulator. Figure 9 shows the simulation run time of two different 64-node mesh network configurations under various network loads running for 100K cycles per simulation. One configuration models a network with a single VC while the second configuration has 8 VCs with speculative allocation. For both configurations, the run time initially grows linearly with the injection rate while the networks are not saturated. This shows that the simulation run time is proportional to the amount of activity occurring in the simulator. The number of VCs in a network also has an effect on simulation run time as with more VCs, the simulator has a higher memory footprint and more network activity to process, both of which directly affect simulation time. This causes the 8-VC simulation to have a higher run time slope; at an injection rate of 0.3, the 8-VC configuration has approximately 50% higher run time compared to the 1-VC configuration. From the run time plot, we can also infer when the two networks reach saturation: at that point, the simulation run time becomes constant as the amount of network activity reaches its maximum.

VI. Summary

In this work we have presented BookSim, a flexible and detailed simulator specialized for NoCs. The simulator offers a large degree of network customization and numerous network component designs. The newest version of the simulator also incorporates many state-of-the-art features and optimization for NoCs developed in recent years. The underlying network model of BookSim2 is highly accurate, as we demonstrate by comparing it with an RTL implementation of a NoC router. The latest simulator source code is available online at [34].

Acknowledgment

This research was supported in part by the P. Michael Farmwald, the Prof. Michael J. Flynn, and the Robert Bosch Stanford Graduate Fellowships. This research was furthermore supported by the National Science Foundation under grants CCF-0701341 and CCF-070234, as well as the Stanford Pervasive Parallelism Laboratory. We would like to thank the anonymous reviewers for their helpful suggestions. We would also like to thank the users of BookSim for helping us improve the simulator through comments, suggestions, and bug reports.

References

[1] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.

[2] J. Kim, J. Balfour, and W. J. Dally, "Flattened Butterfly Topology for On-Chip Networks," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, 2007.

[3] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Cost-Efficient Dragonfly Topology for Large-Scale Systems," *IEEE Micro*, vol. 29, no. 1, Jan. 2009.

[4] N. Jiang, J. Kim, and W. J. Dally, "Indirect Adaptive Routing on Large Scale Interconnection Networks," in *Proceedings of the 36th International Symposium on Computer Architecture*, 2009.

[5] G. Michelogiannakis, J. Balfour, and W. J. Dally, "Elastic-Buffer Flow Control for On-Chip Networks," in *Proceedings of the IEEE 15th International Symposium on High Performance Computer Architecture*, 2009.

[6] D. U. Becker, N. Jiang, G. Michelogiannakis, and W. J. Dally, "Adaptive Backpressure: Efficient Buffer Management for On-Chip Networks," in *Proceedings of the 30th IEEE International Conference on Computer Design*, 2012.

[7] D. U. Becker and W. J. Dally, "Allocator Implementations for Network-on-Chip Routers," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2009.

[8] J. W. Lee, M. C. Ng, and K. Asanovic, "Globally-Synchronized Frames for Guaranteed Quality-of-Service in On-Chip Networks," in *Proceedings of the 35th International Symposium on Computer Architecture*, 2008.

[9] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: Illuminating Future Network-on-Chip with Nanophotonics," in *Proceedings of the 36th International Symposium on Computer Architecture*, 2009.

[10] M. J. Cianchetti, J. C. Kerekes, and D. H. Albonesi, "Phastlane: a rapid transit optical routing network," *SIGARCH Computer Architecture News*, vol. 37, no. 3, Jun. 2009.

[11] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proceedings of the IEEE Symposium on Performance Analysis of Systems and Software*, 2009.

[12] Open-Source Network-on-Chip Router Generator. [Online]. Available: http://nocs.stanford.edu/router.html

[13] D. U. Becker, "Efficient Microarchitecture for Network-on-Chip Routers," Ph.D. dissertation, Stanford University, Aug. 2012.

[14] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jah, "GARNET: A detailed on-chip network model inside a full-system simulator," in *Proceedings of the IEEE Symposium on Performance Analysis of Systems and Software*, 2009.

[15] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The GEM5 Simulator," *SIGARCH Computer Architecture News*, vol. 39, no. 2, May 2011.

[16] NOXIM. [Online]. Available: http://noxim.sourceforge.net

[17] V. Puente, J. A. Gregorio, and R. Beivide, "SICOSYS: an integrated framework for studying interconnection network performance in multiprocessor systems," in *Proceedings of the 10th Euromicro conference on Parallel, distributed and network-based processing*, 2002.

[18] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, 2001.

[19] A. Kumar, P. Kundu, A. P. Singh, L.-S. Peh, and N. K. Jha, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *Proceedings of the 25th International Conference on Computer Design*, 2007.

[20] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, Apr. 1999.

[21] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, Nov. 1993.

[22] Y. Tamir and H.-C. Chi, "Symmetric Crossbar Arbiters for VLSI Communication Switches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 1, Jan. 1993.

[23] D. Abts and D. Weisser, "Age-based packet arbitration in large-radix k-ary n-cubes," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, 2007.

[24] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnection networks," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, 2003.

[25] T. T. Ye, G. de Micheli, and L. Benini, "Analysis of power consumption on switch fabrics in network routers," in *Proceedings of the 39th annual Design Automation Conference*, 2002.

[26] J. Liu and J. G. Delgado-Frias, "A DAMQ shared buffer scheme for network-on-chip," in *Proceedings of the Fifth IASTED International Conference on Circuits, Signals and Systems*, 2007.

[27] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, 2006.

[28] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Proceedings of the 20th annual International Conference on Supercomputing*, 2006.

[29] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. Brown III, and A. Agarwal, "On-Chip Interconnection Architecture of the Tile Processor," *IEEE Micro*, vol. 27, no. 5, 2007.

[30] H. Kim, S. Heo, J. Lee, J. Huh, and J. Kim, "On-Chip Network Evaluation Framework," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010.

[31] J. Hestness and S. W. Keckler, "Netrace: Dependency-Tracking Traces for Efficient Network-on-Chip Experimentation," The University of Texas at Austin, Dept. of Computer Science, Tech. Rep., 2011.

[32] M. M. Lee, J. Kim, D. Abts, M. R. Marty, and J. W. Lee, "Probabilistic Distance-Based Arbitration: Providing Equality of Service for Many-Core CMPs," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, 2010.

[33] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi, "Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks," *SIGARCH Computer Architecture News*, vol. 33, no. 2, May 2005.

[34] BookSim 2.0. [Online]. Available: http://nocs.stanford.edu/booksim.html