

Orion: A Power-Performance Simulator for Interconnection Networks

Hang-Sheng Wang

Xinping Zhu

Li-Shiuan Peh

Sharad Malik

Department of Electrical Engineering,
Princeton University, Princeton, NJ08544
{hangshen,xzhu,peh,sharad}@ee.princeton.edu

Abstract

With the prevalence of server blades and systems-on-a-chip (SoCs), interconnection networks are becoming an important part of the microprocessor landscape. However, there is limited tool support available for their design. While performance simulators have been built that enable performance estimation while varying network parameters, these cover only one metric of interest in modern designs. System power consumption is increasingly becoming equally, if not more important than performance. It is now critical to get detailed power-performance tradeoff information early in the microarchitectural design cycle. This is especially so as interconnection networks consume a significant fraction of total system power. It is exactly this gap that the work presented in this paper aims to fill.

We present Orion¹, a power-performance interconnection network simulator that is capable of providing detailed power characteristics, in addition to performance characteristics, to enable rapid power-performance tradeoffs at the architectural-level. This capability is provided within a general framework that builds a simulator starting from a microarchitectural specification of the interconnection network. A key component of this construction is the architectural-level parameterized power models that we have derived as part of this effort. Using component power models and a synthesized efficient power (and performance) simulator, a microarchitect can rapidly explore the design space. As case studies, we demonstrate the use of Orion in determining optimal system parameters, in examining the effect of diverse traffic conditions, as well as evaluating new network microarchitectures. In each of the above, the ability to simultaneously monitor power and performance is key in determining suitable microarchitectures.

¹Orion is one of the most powerful (brightest) network of stars (constellation) in the sky. It also stands for Open Research Infrastructure for Optimizing Networks.

1 Introduction

Microprocessors are becoming increasingly interconnected. Clusters of computers and server blades connected by interconnection networks are widely deployed in server farms today. Single-chip multiprocessor systems are seeing the use of interconnection networks as the only scalable solution to inter-processor communication [16]. Emerging microprocessor systems will have to interface with an interconnection network fabric. In the future, routers and links will be critical components of a microprocessor system, alongside processors and memories.

Researchers and designers have recognized the importance of low-power computing, especially in emerging microprocessor systems such as blades and SoCs. However, prior work has largely focused on the processing and memory elements of microprocessors, neglecting the communication components. These needs have become more critical, as interconnection networks consume a significant fraction of power in many microprocessor systems. In an Alpha 21364 microprocessor [13], the integrated router and links consume 25W of the total 125W.² In a Mellanox server blade, the InfiniBand switch is estimated to dissipate almost 37.5% of the blade's power budget, taking 15W out of the total power budget of 40W, with the processor allocated the same power budget of 15W [12]. With ever-increasing demand for network bandwidth, power efficiency of interconnection networks will be even more important.

Power models and simulators for processors and memories have been proposed [3, 4, 25], enabling a rich body of research into power-efficient mechanisms [2, 18]. There needs to be a similar drive towards better understanding of the power consumed by routers and links, the basic components of an interconnection network. In this paper, we present Orion, a network power-performance simulator where users can plug-and-play router and link components to form myriad network fabrics, run different communica-

²This is an estimate obtained from Alpha designers, derived using average power density from the earlier generation Alpha EV6. It includes power consumed by the router core and links.

tion workloads on the fabric and investigate their impact on overall network power and performance³. To ease future integration with application simulators, our simulator is constructed within the Liberty Simulation Environment (LSE) [21] which is developed with the goal of enabling systematic architectural design space exploration. LSE already has a user base in microarchitects who use it to model and simulate processors and memories. By extending it to incorporate routers and links, we attest to its flexibility. Our goal is to provide a complete platform for exploring interconnected microprocessors, whether single-chip or spanning multiple chips, at the architectural-level.

The potential impact of Orion is twofold. On one hand, we see it providing designers with a framework for rapid exploration of interconnected microprocessor systems. On the other, by providing fast architectural-level power estimation, we see Orion enabling research in power-efficient hardware and compiler techniques for emerging interconnected processors.

In Section 2, we first describe LSE, the simulation infrastructure on which Orion is built upon, and the building blocks we identify for interconnection networks, aided by a hierarchical modeling methodology of on-chip communication architectures [27]. This is followed with a detailed discussion of Orion’s power models for these building blocks in Section 3. In Section 4, we present three case studies exploring different ways of using our simulator: to pinpoint the optimal power-performance design point for a specific microarchitecture; to see the impact application traffic has on network power and performance; and to evaluate a new microarchitectural mechanism. Section 5 then delves into prior related work and Section 6 concludes the paper with a brief discussion of future directions. The Appendix presents the detailed power models, and explains how the power consumed by network operations are derived from these component power models.

2 The dynamic network simulator

2.1 Simulation infrastructure

We adopt LSE [21] as our basic simulation infrastructure. LSE is a fast execution-driven compiled-code modeling and simulation framework. LSE constructs concurrent structural models and retargetable simulators from a unified structural machine description and specification database. It targets fast design space exploration for modern microprocessors, and has been used for processors and memories. In building Orion, we demonstrate its applicability to another domain. Its flexibility ties in with our end-goal of

³Network performance refers to network latency – how quickly data gets shipped through the network, and network throughput – the amount of data per unit time that the network can handle before it saturates.

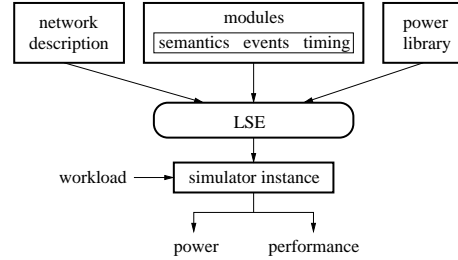


Figure 1. Design, modeling and simulation process under LSE.

providing a complete platform for exploring emerging microprocessor microarchitectures, where processors, memories, routers and links all play critical roles.

In LSE, physical hardware blocks are modeled as logical functional modules that communicate through ports. Data is sent between module ports via message passing. Each module has its own pre-defined parameters and control functions that generates operational and timing behavior for a wide range of application scenarios until pre-compilation stages. For example, a typical buffer module has ports corresponding to read/write ports, and parameters such as buffer size and width.

The integration of power models is based on the event subsystem of LSE that facilitates the collection of execution statistics. Users define events associated with each module. Power models in the power simulation library are hooked to these events so when an event occurs during the execution, it triggers the specific power model, which calculates and accumulates the energy consumed. Figure 1 shows the process of building a performance and power simulator in LSE.

2.2 Building blocks of an interconnection network

Just as a processor can be viewed as an assembly of individual components such as register files, ALUs, caches, etc, an interconnection network can be abstracted as composing of message generating, transporting and consuming agents, such as sources, router and link components and sinks respectively. A similar analogy can be drawn between instructions that flow through processor components and messages that stream through network components.

A key aspect of our simulation environment is a careful selection of the building blocks (modules) of an interconnection network. This process is guided by a hierarchical modeling methodology of on-chip communication architecture [27]. We identify the basic components such as message sources and sinks, router buffers, crossbars, arbiters and links. Conceptually, we classify these modules into two classes. The message transporting class of modules do not

store or modify messages when delivering them. Links and crossbars fall within this class. Another class of modules, the message processing class, generates, stores or modifies message content. Message sources and sinks, buffers and arbiters fall within this second category.

All the modules mentioned above support different types of operational and timing behavior depending on the dynamic configuration specified at the design stage. This generalizes the modules so they can be utilized across different application scenarios. For instance, our modeling enables a buffer module to represent buffers ranging from a simple FIFO queue to complex dynamic prioritized multi-queue buffers.

Through careful parameterization, we have been able to construct a fairly wide range of interconnection networks. Our experience shows that our relatively small library of modules is able to represent an extensive range of architecture choices, from statically-scheduled on-chip networks to complex commercial InfiniBand switches. A small parameterized set of reusable modules is also critical for easy model maintenance. For instance, in Orion, wormhole [6] and virtual-channel [5] networks share exactly the same modules but with differently configured functional and timing behavior.

3 Power modeling

We derive architectural-level parameterized power models for several of the major building blocks identified in Section 2, namely FIFO buffers, crossbars and arbiters. These components occupy about 90% of the area of the Alpha 21364 router [1], and are sufficient for building a diverse range of router microarchitectures.

Dynamic power, the primary source of power consumed in CMOS circuits, is formulated as $P = E f_{clk}$, where energy $E = \frac{1}{2} \alpha C V_{dd}^2$, with f_{clk} the clock frequency, α the switching activity, C the switch capacitance, and V_{dd} the supply voltage. We derive detailed parameterized equations for estimating switch capacitance C of each component of an interconnection network, and track the switching activity α of these components through network simulation.

3.1 Component power modeling

Table 1 gives the terminology we use throughout the paper. We use Cacti [23] to compute the actual C_g , C_d and C_w values. Transistor sizes can be user-input parameters, or automatically determined by Orion with a set of default values from Cacti [23] and applied with scaling factors from Wattch [3]. Sizes of driver transistors, e.g. crossbar input drivers, are computed according to their load capacitance. E_x denotes the energy dissipation per switch of component

Table 1. Terminology

$C_g(T)$	gate capacitance of transistor/gate T
$C_d(T)$	diffusion capacitance of transistor/gate T
$C_a(T)$	$C_g(T) + C_d(T)$
$C_w(L)$	capacitance of metal wire of length L
E_x	$\frac{1}{2} C_x V_{dd}^2$ or $C_x V_{dd}^2$ depending on how to count switches, provided C_x is defined

x , and is implicitly defined when component capacitance C_x is defined.

For each component, we first describe its canonical structure in terms of architectural and technological parameters. We then proceed through detailed analysis to derive parameterized capacitance equations, taking into account both gate and wire capacitances. We then combine capacitance equations and switching activity estimation to derive energy consumption per component operation.

To illustrate our power modeling methodology, we discuss our modeling of FIFO buffers in routers. Buffers are typically implemented as SRAM arrays. We thus adapt architectural-level SRAM array power models that have been proposed for modeling caches and register files [9, 28], incorporating several features specific to router microarchitectures. For instance, a buffer with a dedicated port to the switch does not require tri-state output drivers.

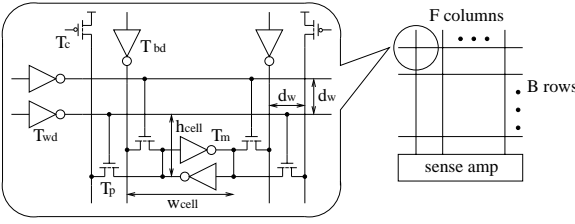
Table 2 shows our power model for FIFO buffers – its canonical structure, architectural and technological parameters, parameterized capacitance equations, and derived energy equations for its operations. Capacitance equations are derived based on circuit structure. Take wordline capacitance C_{wl} as an example, each wordline is connected with F memory cells through 2 pass transistors per memory cell, so C_{wl} is the sum of gate capacitance of these pass transistors $2FC_g(T_p)$, driver capacitance $C_a(T_{wd})$, and wire capacitance $C_w(L_{wl})$.

Power models for crossbars and arbiters are derived in a similar fashion, and are explained in detail in [22]. They are also listed in the Appendix.

3.2 Discussion

Architectural-level modeling. The goal of power modeling in Orion is to derive architectural-level parameterized power models that can provide reasonably accurate power estimates. While it would have been easier to estimate power based on simple rules of thumb such as transistor count and area, our power models are based on detailed estimation of gate and wire capacitance and switching activities. First, estimation based on transistor count and area can only be useful for estimating average power and are thus unable to reflect the impact of varying activity in a power

Table 2. Model for FIFO buffers

Canonical structure	
A FIFO buffer with 1 read port and 1 write port	
	
Architectural parameters	
B	buffer size in flits
F	flit size in bits
P_r	number of buffer read ports
P_w	number of buffer write ports
Technological parameters	
h_{cell}	memory cell height
w_{cell}	memory cell width
d_w	wire spacing
Capacitance equations	
wordline length	$L_{wl} = F(w_{cell} + 2(P_r + P_w)d_w)$
bitline length	$L_{bl} = B(h_{cell} + (P_r + P_w)d_w)$
wordline cap.	$C_{wl} = 2FC_g(T_p) + C_a(T_{wd}) + C_w(L_{wl})^*$
read bitline cap.	$C_{br} = BC_d(T_p) + C_d(T_c) + C_w(L_{bl})$
write bitline cap.	$C_{bw} = BC_d(T_p) + C_a(T_{bd}) + C_w(L_{bl})$
precharge cap.	$C_{chg} = C_g(T_c)$
memory cell cap.	$C_{cell} = 2(P_r + P_w)C_d(T_p) + 2C_a(T_m)$
sense amp energy	E_{amp} from empirical model [28]
Operation energy equations	
δ_{bw}	number of switching write bitlines
δ_{bc}	number of switching memory cells
read energy	$E_{read} = E_{wl} + F(E_{br} + 2E_{chg} + E_{amp})$
write energy	$E_{wrt} = E_{wl} + \delta_{bw}E_{bw} + \delta_{bc}E_{cell}$

(*) T_p is the pass transistor connecting bitlines and memory cells, T_{wd} is the wordline driver, T_{bd} is the write bitline driver, T_c is the read bitline precharge transistor, T_m is the memory cell inverter.

simulator. Second, information such as transistor count and area is typically not available at the time of architectural exploration.

Model hierarchy and reusability. To maximize reuse of our power models, so users can extend them to new microarchitectures easily, we construct our models in a hierarchical fashion, *i.e.* a model can have another as its component, so complex models can be built in a divide-and-

conquer fashion. By reusing old models and hierarchically building new models, we can save significant modeling effort.

For instance, in our modeling of central buffers, we heavily leveraged existing component power models rather than starting from scratch. Central buffers are implemented as pipelined shared memories [10], essentially regular SRAM banks connected by pipeline registers, with two crossbars facilitating the pipelined data I/O. We reused our FIFO buffer model for the SRAM banks, and the flip-flop sub-component models from our arbiter model for the pipeline registers. The two crossbars are modeled with our crossbar power model. Through a well-defined interface to propagate data and collect power statistics, the top-level central buffer model interacts with these lower-level models to estimate the power consumption.

Validation. We are in the process of validating our power models against measured power numbers of existing routers, and against low-level power estimation tools. Preliminary validation with router designers' guesstimates found the power estimates derived by Orion for two commercial routers – the Alpha 21364 router [13] and the IBM InfiniBand 8-port 12X switch [8] to be within ballpark [22]. Due to the sensitivity of the data, we have yet to obtain rigorous power estimates or measurements of the two routers, and are thus unable to provide precise error margins. This is the focus of our current efforts, and we are actively exploring more detailed validation of both chip-to-chip and on-chip networks through close collaboration with various design groups.

Link power modeling. A large variety of link architectures has been proposed for interconnection networks, both for chip-to-chip and on-chip signaling. Across different link architectures, power characteristics differ greatly. For this paper, we choose to plug in actual power numbers of specific links obtained from published datasheets. It is clearly preferable to have parameterized link power models, just as we have parameterized router power models, so architects can perform architectural-level tradeoffs for links as well. We are currently working with chip-to-chip and on-chip link designers to develop such parameterized link power models.

Release of power models. We will be distributing our power models (coded in C) as part of Orion's release. This will allow our power models to be used independently from the simulator, either as a separate power analysis tool, or as a plug-in to other network simulators.

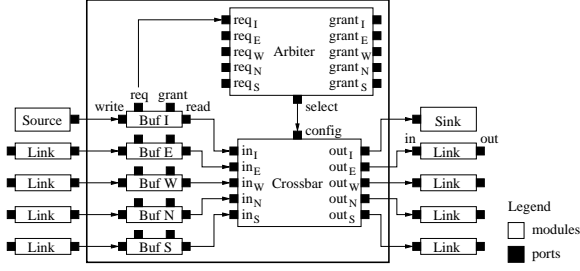


Figure 2. A simple wormhole router as modeled in Orion.

3.3 Walkthrough example of a simple wormhole router

We now show how a simple wormhole router is modeled in Orion, and walk a head flit⁴ through the router, showing how its power consumption is estimated. We assume the router has 5 input/output ports, with 4 flit buffers per input port and each flit 32 bits wide; a 5×5 crossbar and a 4:1 arbiter per output port⁵. We also assume source routing for simplicity.

Figure 2 sketches the module representation of a wormhole router and its neighboring links in Orion. The source module injects a head flit into the *write* port of the input buffer module *Buf I*. The buffer module writes the flit into the tail of the FIFO buffer and emits a *buffer write* event, which triggers the buffer power model to compute buffer write energy E_{wrt} .

When the flit emerges at the head of the FIFO buffer, it is checked via the *read* port of the buffer module, its route read, and a request sent to the req_I port of the arbiter module for the desired output port, say the north output port. The arbiter module performs the required arbitration and emits an *arbitration* event, which signals the arbiter power model to compute arbitration energy E_{arb} .

Assuming the request is granted, the arbitration result is sent to the *config* port of the crossbar module. A grant signal is also sent to the *grant* port of the buffer module *Buf I*, leading to the *read* port of the buffer module activated. The flit is then read, emitting a *buffer read* event, which causes the buffer power model to compute buffer read energy E_{read} .

The flit next traverses the crossbar, from input port in_I , to the north output port out_N . The crossbar module emits a *crossbar traversal* event and the crossbar power model computes traversal energy E_{xb} .

Finally, the flit leaves the router, enters the *in* port of the

link module, traverses the link and leaves through the *out* port of the link module. The link module emits a *link traversal* event, which calls the link power model to compute link traversal energy E_{link} .

The total energy this head flit has consumed at this node and its outgoing link is thus:

$$E_{flit} = E_{wrt} + E_{arb} + E_{read} + E_{xb} + E_{link}$$

4 Case studies

We foresee three potential ways of using Orion for rapid exploration of network microarchitectures, as shown in Figure 3. Subsequently, we will discuss three case studies demonstrating instances of these three usage categories and show how the architectural-level exploration of network power-performance tradeoffs enabled by Orion provided valuable insights.

First, the architect may wish to trade-off two configurations of a microarchitecture, exploring their effect on network power and performance. This involves selecting the modules forming that microarchitecture, and setting different module parameters for the two configurations. Given a representative communication workload of the targeted application, the user can feed the workload and configurations into two different instances of Orion, and obtain their power and performance numbers.

Second, an architect may wish to explore the impact of two application traffic patterns on a specific network microarchitecture, to see if the network can handle diverse communication patterns. This can be part of a study to design a network suitable for various communication patterns representative of an application suite. It can also be used to provide feedback to the compiler or application programmer on how to better place the program and data to avoid communication patterns that adversely tax network power and performance.

Third, a researcher may devise a new microarchitectural technique and wish to explore its impact on interconnection network power and performance, evaluating it against a base microarchitecture. This may involve defining one or several new modules, along with new power models. Usually, a module can be reused, with novel functional and timing characteristics inserted. Component power models can also typically be leveraged and modified for new microarchitectures, as we will demonstrate subsequently.

4.1 Experimental setup

We assume a 16-node network organized as a 4×4 torus, as shown in Figure 4. Each router has five physical bidirectional ports (north, south, east, west, injection/ejection)

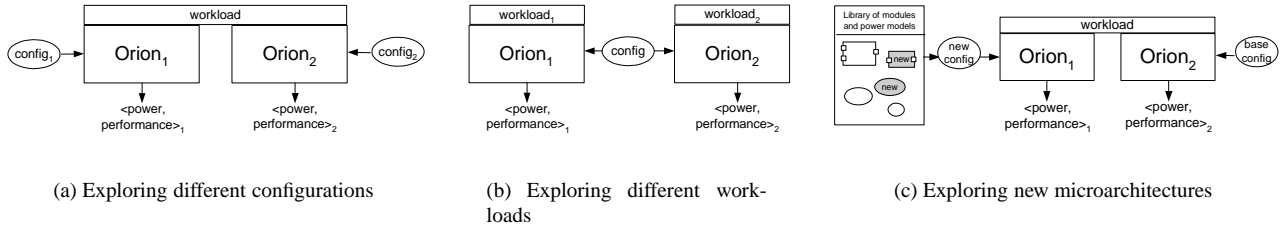


Figure 3. Three potential ways of using Orion

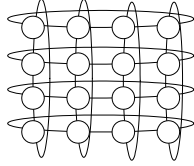


Figure 4. A 4-by-4 torus network.

and propagation delay across data and credit channels is assumed to take a single cycle. Credit-based flow control regulates the use of buffers, *i.e.*, a credit is sent back to the previous router whenever a flit leaves, so a router can maintain a count of the number of available buffers, and no flits are forwarded onto the next hop unless there are buffers to hold it. Since our case studies focus on exploring the impact of network microarchitectures and workloads on performance and power, we choose simple source dimension-ordered routing⁶ where the route is encoded in a packet beforehand at source.

In our experiments, the simulator generates uniformly distributed traffic to random destinations, unless otherwise mentioned. Each simulation is run for a warm-up phase of 1000 cycles with 10,000 packets injected thereafter and the simulation continued at the prescribed packet injection rate till these packets in the sample space have all been received, and their average latency calculated. Latency spans from when the first flit of the packet is created, to when its last flit is ejected at the destination node, including source queuing time and assuming immediate ejection. The saturation throughput of a network is defined as the point at which average packet latency increases to more than twice zero-load latency, *i.e.* the latency experienced by packets when there is no contention in the network. Packets are 5-flit long, consisting of a head flit leading 4 data flits. Routers are pipelined in accordance to the router delay model proposed in [15].

The simulator records energy consumption of each component (input buffer, crossbar, arbiter, link) of a node over the entire simulation excluding the first 1000 cycles. Aver-

⁶Dimension-ordered routing is where a packet always goes along one dimension first, followed by another.

age power is then computed by multiplying the total energy by frequency and then dividing by total simulation cycles.

In our experiments, a typical 4x4 torus network using virtual channels comprises 59 modules. The constructed Orion simulator is 5202KB in size, with a system simulation speed of about 1000 simulation cycles per second on a Pentium III 750MHz machine running Linux.

4.2 Exploring different configurations: wormhole vs. virtual-channel routers

We assume an on-chip network similar to that in [7], *i.e.* a 4x4 torus network on a 12mm x 12mm chip with 256-bit flits, clocked at 2GHz, with $V_{dd} = 1.2V$, in 0.1μm process technology. Link capacitance is 1.08pF/3mm using the same parameters in our router modeling. E_{link} is computed from link capacitance and link switching activities reported by Orion.

We simulate and compare four different router configurations :-

- wormhole router with 64-flit input buffer per port (WH64).
- virtual-channel (VC) router with 2 VCs per port and 8-flit input buffer per VC (VC16).
- virtual-channel router with 8 VCs per port and 8-flit input buffer per VC (VC64).
- virtual-channel router with 8 VCs per port and 16-flit input buffer per VC (VC128).

As prescribed by [15], these virtual-channel routers fit within a 3-stage router pipeline of virtual-channel allocation, switch allocation and crossbar traversal, and the wormhole router has a 2-stage router pipeline of switch arbitration and crossbar traversal. All routers have a 5x5 crossbar.

Figure 5 graphs results obtained from simulating these routers in Orion. Figure 5(a) shows VC16 out-performing WH64, despite having only a quarter of the buffering per input port, saturating at a higher packet injection rate of 0.15 packets/cycle/node. In addition, this performance improvement is not achieved at the expense of higher power consumption, as indicated by Figure 5(b). In fact, VC16

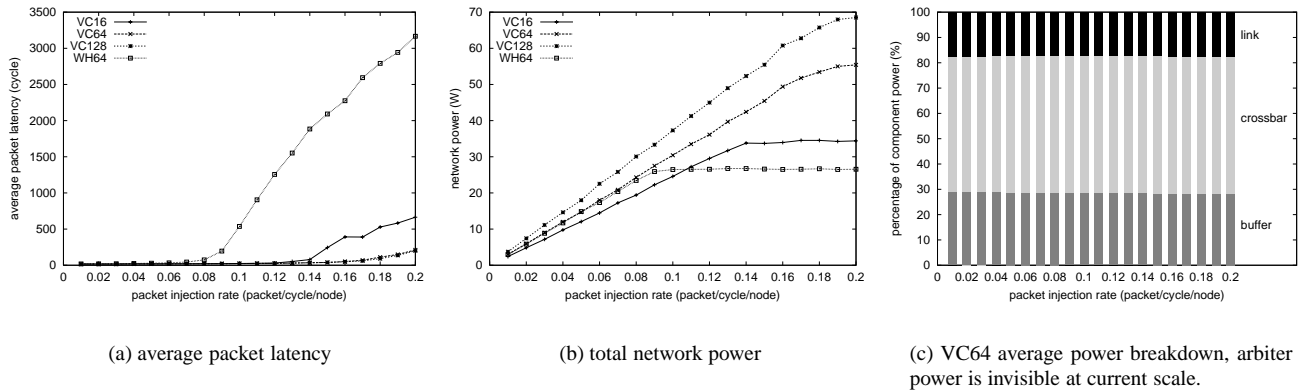


Figure 5. Power-performance of on-chip 4×4 torus networks governed by wormhole and virtual-channel flow control at varying packet injection rates.

dissipates less power than WH64 at the same packet injection rate before the network saturates. Beyond packet injection rate of 0.11 packets/cycle/node, VC16 starts to consume more power than WH64, since it is still able to absorb the higher packet injection rate, so network activity continues to increase. For all configurations, total network power levels off after saturation, since the network cannot handle a higher packet injection rate, so the switching activity of the network remains constant.

It is interesting to note that VC64 dissipates approximately the same amount of power as WH64 before saturation. Intuitively, since virtual-channel flow control is a more complicated protocol, requiring more complex hardware, we would expect a virtual-channel router to be more of a power hog than a wormhole router. Figure 5(c) explains why the difference is not significant. It shows input buffers and the crossbar switch fabric as the two dominant power consumers of a node (consuming more than 85% of total node power).⁷ While virtual-channel flow control requires more complicated arbitration circuitry, the power consumed by arbiters (less than 1% of node power) is minimal. Hence, the impact on total node power is negligible. Clearly, our experiments show virtual-channel flow control returning better power-performance than wormhole.

Since buffer power is significant, VC128 dissipates higher network power as compared to VC64 or VC16. When this translates to higher network throughput, the increase in power may be acceptable. Clearly, it will not be viable to choose VC128 over VC64 with our configuration, since the additional power dissipation is not matched with an improvement in performance. By providing a platform

⁷In [7], it is estimated that links take up considerably more power than routers. The huge difference between our estimates and theirs is largely because they consider all datapath power as link power, while we consider datapath within the router (crossbar) as router power.

for simulating both power and performance, Orion provides the architect a tool for exploring the optimal configurations of a network microarchitecture.

4.3 Exploring different workloads: broadcast vs. uniform traffic

Just as an application's execution patterns heavily impact the power-performance of a processor, an application's communication patterns also critically affect the power-performance of a network.

We select two different communication patterns :-

- uniform random traffic, *i.e.* each node injects packets to randomly distributed destinations other than itself in the network.
- broadcast traffic, *i.e.* one node injects packets to all the other nodes in the network.

Both communication workloads inject packets at a uniform rate. The total packet injection rate across the entire network is kept the same across the two workloads. For broadcast traffic, the source node at position (1,2) injects at the maximum rate of 0.2 packets per cycle. For uniform random traffic, each node injects at a rate of $\frac{1}{16} \times 0.2$ packets per cycle, so total packet injection rate for the entire network is still 0.2 packets per cycle.

Network topology is fixed across the workloads as a 4×4 torus. We also fix the router microarchitecture – virtual-channel routers with 2 VCs per port and 8 flit buffers per VC. Other aspects of the network microarchitecture remain the same as configured in the previous experiment, *i.e.* an on-chip network.

Figure 6 shows the average power consumed by each node in the 4×4 network as caused by the diverse traffic patterns. Uniform random traffic results in a rather flat power

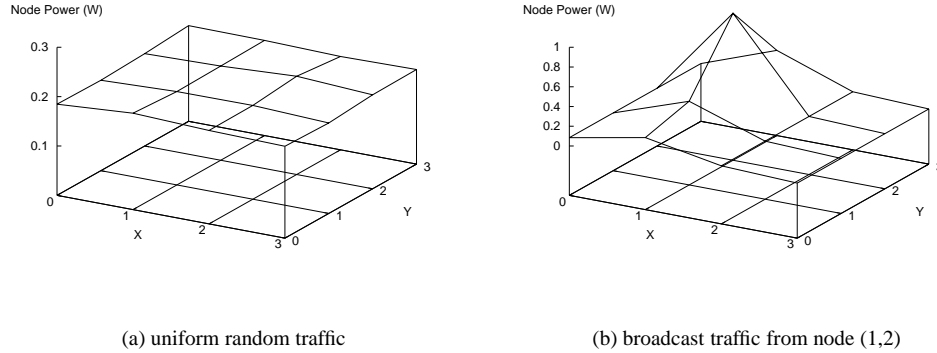


Figure 6. Power spatial distribution for a 4×4 on-chip network loaded with diverse communication traffic. The 16 nodes in the network are labeled in a 2-dimensional Cartesian space with tuples (x, y) .

spatial distribution, *i.e.* all nodes having almost identical power consumption (see Figure 6(a)). This is understandable since uniform random traffic sends the same amount of traffic to each node. Thus, we see the same number of buffer reads/writes, arbitrations, crossbar and link traversals on average. Furthermore, the flat power spatial distribution is a direct result of the uniform torus network topology which does not bias traffic distribution at any particular node.

With broadcast traffic, the source node consumes the most power, as shown in Figure 6(b). Node power goes down quickly as the Manhattan distance from the source node increases. This is because the further the node is from the source node, the lower its workload. To roughly estimate the workload decreasing factor, let L be the workload of the source node, then each of its immediate neighbors has $\frac{L}{4}$ workload on average, and each neighbor 2 hops away has $\frac{L}{8}$ workload on average, so workload diminishes very quickly as distance from the source node increases. Since router and link power depends on flit arrival rate (or workload), node power also diminishes quickly the further a node is from the source.

Furthermore, we can see the effect of the routing policy. In our dimension-ordered routing, we route along the y -axis first. So with node (1,2) broadcasting, nodes (1,1) and (1,3) have higher power consumption than nodes (0,2) and (2,2) since they have more data traffic. Since x -axis is routed last, all nodes with the same x coordinate have identical power consumption as they have the same load (unless they lie along the same x -coordinate as the source node).

This experiment demonstrates how Orion can be used to explore spatial variance in power-performance as a result of drastically different traffic patterns. This can be very useful for the microarchitect in determining a suitable routing policy for application-specific traffic patterns, or to balance

workload among network nodes to avoid power and performance hot-spots. It should be noted that while our experiments use synthetic workloads, as no realistic communication workloads are readily available, Orion can be interfaced with actual communication traces for more realistic results.

4.4 Exploring a new microarchitectural technique: central buffered routers

Central buffered routers (CB), where a shared central buffer forwards flits between input and output ports of a router, have been deployed in IBM SP/2 and InfiniBand routers [19, 8] and are chosen for their potential for higher throughput over input-buffered crossbar-based routers (XB), as they do not experience the head-of-line blocking inherent in XB routers.

In this experiment we wish to evaluate the power-performance of CB versus the base XB router microarchitecture. For a fair comparison, we define two router configurations of XB and CB routers that take up roughly the same area. As our power models include length estimation of buffer bitlines, wordlines and crossbar input/output lines, router area can be easily estimated assuming a rectangular layout. We estimate router area as the sum of input buffer area and switch fabric area, ignoring arbiter area since arbiters are relatively small. The two configurations are as follows :-

- Central-buffered router with a 4-bank central buffer, each 1 flit wide, 2560 chunks (2560 rows, each row 4-flit wide), 2 read ports, 2 write ports, and a 64-flit input buffer at each port (CB).
- Input-buffered crossbar-based router with 16 virtual channels, 268-flit input buffer per VC and a 5×5 crossbar (XB).

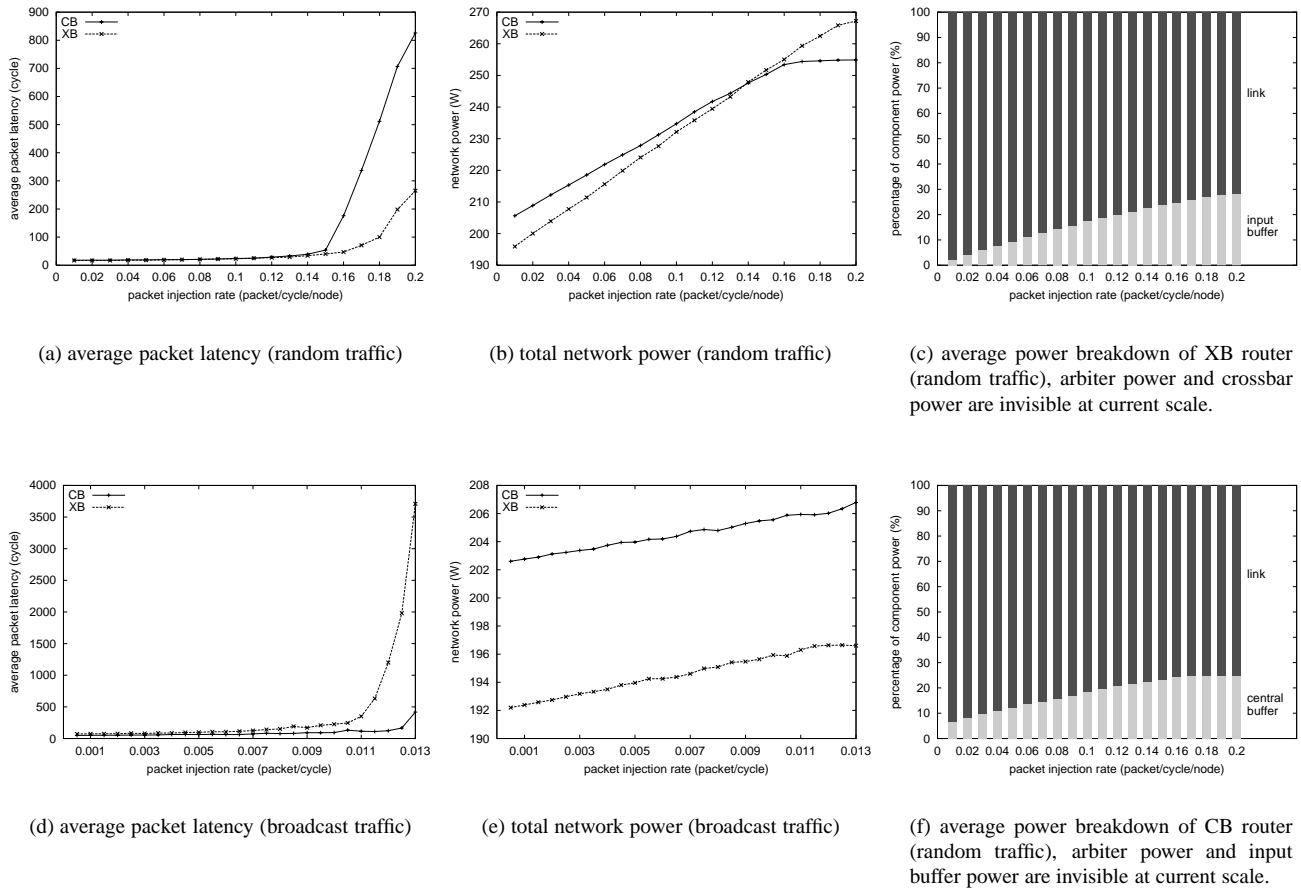


Figure 7. Power-performance of chip-to-chip 4×4 torus networks composed of CB and XB routers at varying packet injection rates.

Unlike in previous experiments, we assume a chip-to-chip 4×4 network here. Flits are 32-bit wide and routers run at 1GHz. So each router port has a 32Gb/s link (assuming no signaling overhead) and we assume a 32Gb/s link that consumes 3W each. This assumption is based on the 3W power consumption of a 30Gb/s IBM InfiniBand 12X link [8]. These chip-to-chip links use differential signaling, and thus consume almost the same power regardless of link activity.

Figure 7 shows the simulation results. We see CB routers having poorer throughput than XB routers with uniform random traffic (Figure 7(a)). This is due to the fewer number of switch fabric ports in the central buffer as compared to the crossbar switch (2 instead of 5). With uniform random traffic, head-of-line blocking in XB routers do not pose problems to overall throughput, since other input ports have sufficient traffic to keep the switch fabric busy. With a nonuniform traffic pattern like broadcast, CB routers perform better, as shown in Figure 7(d), since packets from

the same input port need not line up behind one another if they are destined for different output ports. While virtual-channel flow control alleviates head-of-line blocking, packets of the same VC still needs to wait for packets ahead in the queue.

This performance gain is not without cost though. Figures 7(b) and 7(e) show the higher power consumption of CB routers, which is rather counter-intuitive since the two configurations occupy roughly the same area. Figures 7(f) and 7(c) provide some insight. The dominant power consumer in CB routers is the central buffer while in XB routers, it is the input buffers. Since the switch fabric switches much more frequently than the input buffers (approximately five times more frequently in our experiments), and a central buffer consumes much more energy than a crossbar due to its higher switching capacitance, this leads to the higher power consumption of CB routers.

It is also interesting to note the vastly different router vs. link power distribution of chip-to-chip networks and on-

chip networks. In the on-chip network in Figure 5(c), links take up less than 15% of node power. In this chip-to-chip network, they take up more than 70% of node power (see Figure 7(c)). While our chip-to-chip network configuration exaggerates the imbalance, since realistic chip-to-chip networks tend to have a 60-40% link-router distribution [8, 13], the results do highlight the distinct difference between chip-to-chip high-speed links whose power dissipation is traffic-insensitive, and on-chip links whose power consumption depends heavily on traffic. Our results clearly point to a need to address the sizable power consumed by chip-to-chip links that is invariant to network load.

5 Related work

Circuit-level techniques applicable to interconnection networks, such as low-power link architectures [11], have been actively explored. However, architectural-level mechanisms for power-efficient interconnection networks are sorely lacking. Shang, Peh and Jha first proposed a power optimization mechanism for interconnection networks [17], applying dynamic voltage scaling to links in networks. In their study, they had to go through detailed RTL-level power modeling to obtain power estimates. These low-level power estimation tools such as PowerCompiler [20] require complete RTL code to be available, and simulate slowly, on the order of hours, while our architectural-level power simulator takes on the order of minutes. Circuit-level power estimation tools, though providing excellent accuracy, take even longer, and require even more substantive development effort. These shortcomings have motivated architectural-level power simulators for processors and memories such as Wattch [3] that have helped open power analysis to computer architects. It is our hope that Orion will similarly ease future architectural-level research into power-efficient interconnection networks and interconnected microprocessor systems.

Power models have been proposed for a variety of network fabrics in the past. Patel *et al.* first noted the need to consider power constraints in interconnection network design, and proposed a power model of routers and links [14]. There have also been models proposed for other types of networks such as on-chip FPGA networks [26] and IP router fabrics [24]. All these prior studies focused on exploring the power consumption of different network topologies. As their goal was neither in building a tool for enabling fast architectural-level exploration, nor in enabling architectural-level research into power-efficient mechanisms, they adopted models based on transistor count [14], switch width [26] (which can only be used to derive average power estimates), or relied on numbers plugged in from low-level power estimation tools [24]. These information are typically unavailable at the time of

architectural design space exploration. In our approach, we believe that architectural-level parameterized power equations coupled with dynamic simulation of switching activity are key, as they allow researchers and designers to explore the impact of different microarchitectures and workloads on network power and performance.

6 Conclusions

As we enter the era of microprocessor systems with processors coupled together using interconnection networks, we need to expand the tool support we provide to microarchitects for design space exploration. Given the need for determining optimal power-performance design points, it is critical that a power (and performance) simulator be available to microarchitects to evaluate not just the processing and memory elements, but also the communication elements of such emerging microprocessor systems. This paper describes our efforts to address this issue – we describe Orion, a power-performance simulator for interconnection networks that provides a platform for rapid exploration of power-performance tradeoffs in network microarchitecture design. We believe that such simulation support is critical for the development of hardware and software techniques for power optimization of these emerging systems.

Our power modeling is guided by our goal of providing an architectural-level platform for exploring power-performance tradeoffs. We derive detailed architectural-level parameterized power equations that provides reasonable accuracy while allowing fast tradeoffs early in the design. Throughout our modeling, we show how with a careful reuse of sub-component models, we can build power models for diverse interconnection network microarchitectures.

While the core simulation infrastructure is provided by a general component-based simulation framework (LSE), we make unique contributions in the building of the Orion simulator in defining the microarchitectural components needed (the building blocks). We show how a relatively small set of components is sufficient to create a very wide range of network microarchitectures through careful parameterization. Once these components are available, it is possible to “pick, plug and play” any specific network microarchitecture, ultimately synthesizing and building a compiled code simulator.

We present three potential ways of using Orion – to identify the optimal configuration of a microarchitecture based on power and performance, to explore the impact different communication workloads have on a specific microarchitecture, and to evaluate new microarchitectural mechanisms. Three experimental case studies were carried out in these usage categories. In all three, Orion’s ability to dynamically monitor power and performance enabled valuable insights.

Lots of exciting work lies ahead for Orion. We are working on more extensive modules and power models, along with detailed validation. Through releasing the source code of Orion, we hope that the research and design community at large will be able to help us improve Orion with more and better modules and power models, covering a wider range of network microarchitectures. We are currently in the process of tying Orion with a SoC application simulator. Ultimately, we see Orion as a useful cornerstone in a complete platform for rapid architectural-level exploration of interconnected microprocessor systems.

Acknowledgments

We thank the Liberty team at Princeton led by David August for providing extensive assistance in our learning and using of LSE. We also thank Craig Stunkel of IBM T.J. Watson for providing architectural details and power numbers of the central-buffered IBM InfiniBand router, and the Alpha 21364 design team for supplying us with power estimates. In addition, the authors are grateful to Brian Towles of Stanford who helped us better understand the on-chip network described in his paper, and gave us valuable comments on this paper and our power modeling.

References

- [1] P. Bannon. Alpha 21364 (EV7). http://www.eecs.umich.edu/vlsi_seminar/f01/slides/bannon.pdf.
- [2] D. Brooks and M. Martonosi. Value-based clock gating and operation packing: Dynamic strategies for improving processor power and performance. *ACM Transactions on Computer Systems*, 18(2), 2000.
- [3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. International Symposium on Computer Architecture*, 2000.
- [4] J. A. Butts and G. S. Sohi. A static power model for architects. In *Proc. International Symposium on Microarchitecture*, 2000.
- [5] W. J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2), 1992.
- [6] W. J. Dally and C. L. Seitz. The torus routing chip. *Journal of Parallel and Distributed Computing*, 1(3), 1986.
- [7] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proc. Design Automation Conference*, 2001.
- [8] IBM. IBM InfiniBand 8-port 12x switch. <http://www-3.ibm.com/chips/products/infiniband/>.
- [9] M. B. Kamble and K. Ghose. Analytical energy dissipation models for low-power caches. In *Proc. International Symposium on Low Power Electronics and Design*, 1997.
- [10] M. Katevenis, P. Vatsolaki, and A. Efthymiou. Pipelined memory shared buffer for VLSI switches. In *Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1995.
- [11] M.-J. E. Lee, W. J. Dally, and P. Chiang. Low-power area-efficient high-speed I/O circuit techniques. *IEEE Journal of Solid-State Circuits*, 35(11), 2000.
- [12] Mellanox Technologies Inc. Mellanox performance, price, power, volume metric (PPPV). <http://www.mellanox.com/products/shared/PPPV.pdf>.
- [13] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb. The Alpha 21364 network architecture. *IEEE Micro*, 22(1), 2002.
- [14] C. S. Patel, S. M. Chai, S. Yalamanchili, and D. E. Schimmel. Power constrained design of multiprocessor interconnection networks. In *Proc. International Conference on Computer Design*, 1997.
- [15] L.-S. Peh and W. J. Dally. A delay model and speculative architecture for pipelined routers. In *Proc. International Symposium on High Performance Computer Architecture*, 2001.
- [16] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Proc. Design Automation Conference*, 2001.
- [17] L. Shang, L.-S. Peh, and N. K. Jha. Power-efficient interconnection networks: Dynamic voltage scaling with links. *Computer Architecture Letters*, 1(2), 2002.
- [18] P. Stanley-Marbell, M. S. Hsiao, and U. Kremer. A hardware architecture for dynamic performance and energy adaptation. In *Proc. Workshop on Power-Aware Computer Systems*, 2002.
- [19] C. B. Stunkel, D. G. Shea, B. Abali, M. G. Atkins, C. A. Bender, D. G. Grice, P. H. Hochschild, D. J. Joseph, B. J. Nathanson, R. A. Swetz, R. F. Stucke, M. Tsao, and P. R. Varker. The SP2 high-performance switch. *IBM Systems Journal*, 34(2), 1995.
- [20] Synopsys Corp. PowerCompiler datasheet. http://www.synopsys.com/products/power/power_bkg.html.
- [21] M. Vachharajani, N. Vachharajani, D. A. Penry, J. A. Blome, and D. I. August. Microarchitectural exploration with Liberty. In *Proc. International Symposium on Microarchitecture*, 2002.
- [22] H.-S. Wang, L.-S. Peh, and S. Malik. A power model for routers: Modeling Alpha 21364 and InfiniBand routers. In *Proc. Hot Interconnects 10*, 2002.
- [23] S. J. Wilton and N. P. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical Report 93/5, DEC Western Research Laboratory, 1994.
- [24] T. T. Ye, L. Benini, and G. D. Micheli. Analysis of power consumption on switch fabrics in network routers. In *Proc. Design Automation Conference*, 2002.
- [25] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. The design and use of SimplePower: A cycle-accurate energy estimation tool. In *Proc. Design Automation Conference*, 2000.
- [26] H. Zhang, M. Wan, V. George, and J. Rabaey. Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs. In *Proc. IEEE Computer Society Workshop on VLSI*, 1999.
- [27] X. Zhu and S. Malik. A hierarchical modeling framework for on-chip communication architectures. In *Proc. International Conference on Computer-Aided Design*, 2002.

[28] V. Zyuban and P. Kogge. The energy complexity of register files. In *Proc. International Symposium on Low Power Electronics and Design*, 1998.

Appendix

Crossbar model. We model two common crossbar implementations – multiplexer tree crossbar and matrix crossbar. Table 3 presents the model for matrix crossbars.

We use average length for control lines and assume they are along the same direction as input lines, thus $C_w(\frac{L_{in}}{2})$ in C_{xb_ctr} . Since data path is much wider than control path in

Table 3. Model for matrix crossbars

Canonical structure and notation	
connector	either a tri-state buffer or a transmission gate
C_{in_cnt}	input node cap. of a connector
C_{out_cnt}	output node cap. of a connector
C_{ctr_cnt}	control node cap. of a connector
Architectural parameters	
I	number of crossbar input ports
O	number of crossbar output ports
W	port width in bits
Technological parameters	
h_t	track height
w_t	track width
Capacitance equations	
input line length	$L_{in} = O \cdot W \cdot w_t$
output line length	$L_{out} = I \cdot W \cdot h_t$
input line cap.	$C_{xb_in} = O \cdot C_{in_cnt} + C_a(T_{id}) + C_w(L_{in})^*$
output line cap.	$C_{xb_out} = I \cdot C_{out_cnt} + C_a(T_{od}) + C_w(L_{out})$
control line cap.	$C_{xb_ctr} = W \cdot C_{ctr_cnt} + C_w(\frac{L_{in}}{2})$
Operation energy equations	
δ_{xi}	number of switching input bits
δ_{xo}	number of switching output bits
traversal energy	$E_{xb} = \delta_{xi}E_{xb_in} + \delta_{xo}E_{xb_out}$

(*) T_{id} is the input driver, T_{od} is the output driver.

Table 4. Model for matrix arbiters

Canonical structure and notation	
C_{FF}	switch capacitance of a flip flop
C_{FC}	clock capacitance of a flip flop
Architectural parameters	
R	number of requesters
Capacitance equations	
request cap.	$C_{req} = C_a(T_I) + (R - 1)C_g(T_{N1}) + C_g(T_{N2})^*$
grant cap.	$C_{gnt} = C_d(T_{N2})$
priority cap.	$C_{pri} = C_{FF} + 2C_g(T_{N1})$
internal cap.	$C_{int} = C_d(T_{N1}) + C_g(T_{N2})$
clock cap.	$C_{clk} = C_{FC}$
Operation energy equations	
δ_{ar}	number of switching request signals
δ_{ap}	number of switching priority bits
δ_{ai}	number of switching internal nodes
arbitration energy	$E_{arb} = \delta_{ar}E_{req} + \delta_{ap}E_{pri} + \delta_{ai}E_{int} + (E_{gnt} + E_{xb_ctr})$

(*) T_{N1} is the first level NOR gate, T_{N2} is the second level NOR gate, T_I is the inverter.

a crossbar and most crossbars are symmetric ($I = O$), this assumption does not affect model accuracy.

Arbiter model. We model three types of arbiters: matrix arbiter, round-robin arbiter and queuing arbiter. Table 4 presents the detailed power model for matrix arbiters.

E_{arb} is derived as follows:

- We treat E_{xb_ctr} as a part of E_{arb} because arbiter grant signals drive crossbar control signals so they have identical switching behavior.
- Since each arbitration grants one and only one request, there is no switching activity factor applied to E_{gnt} and E_{xb_ctr} .

Throughout our power models, the switching activity factors δ_x are monitored and calculated through simulation.