ORION 2.0: A Power-Area Simulator for Interconnection Networks

Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi

***Abstract*—As industry moves towards multicore chips, networks-on-chip (NoCs) are emerging as the scalable fabric for interconnecting the cores. With power now the first-order design constraint, early-stage estimation of NoC power has become crucially important. In this work, we present ORION 2.0, an enhanced NoC power and area simulator, which offers significant accuracy improvement relative to its predecessor, ORION 1.0.**

***Index Terms*—Architectural-level modeling, design space exploration, network-on-chip (NoC).**

I. INTRODUCTION

Network power has become increasingly substantial in multicore designs, with the increasing demand for network bandwidth. This requires designers to accurately estimate on-chip network power consumption. Power estimation can be carried out at different levels of abstraction that trade off estimation time versus accuracy, ranging from real-chip power measurements [5], to pre- and post-layout transistor-level simulations [23], to RTL power estimation tools [26] to early-stage architectural power models [4], [9], [18], [19]. Low-level power estimation tools, even RTL power estimation, require complete RTL code to be available, and simulate slowly, on the order of hours, while evaluation of an architectural power model takes on the order of seconds.

Architectural power estimation is important to: 1) verify that power budgets are approximately met by the different parts of the design and the entire design and 2) evaluate the effect of high-level optimizations, which have more significant impact on power than low-level optimizations [9]. Patel *et al.* [16] proposed a power model for interconnection networks based on transistor count. As the model is not instantiated with architectural parameters, it cannot be used to explore tradeoffs in router microarchitecture design. Bona *et al.* [3] gave a methodology for automatically generating the energy models for on-chip communication infrastructure at system level; however, the focus is on bus-based and crossbar-based communication for systems-on-chip (SoC). Bhat *et al.* [2] proposed an architecture-level regression model for different router components based on energy numbers obtained from simulations.

ORION 1.0, a set of architectural power models for on-chip interconnection routers, was proposed in [18] and has been widely used for early-stage NoC power estimation in literature and industry. However, for the Intel 80-core Teraflops chip [10] there is up to 8× difference between ORION 1.0 estimations (per component) and silicon measurements. Also, the estimated total power is about 10× less than actual. Indeed, ORION 1.0 does not include clock and link power models, which are major components of NoC power.

In addition, since architectural design space exploration is typically done for current and future technologies, models must be derivable from standard technology files (e.g., Liberty [23] and LEF [22]), as well as extrapolatable process models such as PTM [24] or ITRS [21], whereas ORION 1.0 collects inputs from *ad hoc* sources to derive its internal power models. Therefore, we have developed ORION 2.0 with two key goals: 1) to update and enhance ORION's power and area estimation accuracy and 2) to encompass ORION 2.0 within a semi-automated flow (i.e., using shell scripting) so that ORION can be continuously maintained and updated using standard technology files and/or extrapolatable process models.

This paper draws on a preliminary account published at DATE-09 [11]. Here, we add the following contributions: 1) discussion of supported router architectures; 2) evaluation of the proposed models against different microarchitectural parameters, and against synthesized router results; 3) revised clock power model; and 4) highlights of potential shortcomings of the proposed models. Contributions of ORION 2.0 beyond the original ORION 1.0 include those given here.

- **New**:
  - — We add flip-flop and clock dynamic and leakage power models.
  - — We add link power models, leveraging accurate models recently developed in [6].
  - — We modify the virtual-channel (VC) allocator microarchitecture in ORION 1.0 to optimize its power consumption. Also, a new VC allocation model, based on the microarchitecture and pipeline proposed in [13], is added in ORION 2.0.
  - — We add arbiter leakage power model using the methodology proposed in [7].
  - — We add accurate area models for all the router building blocks.
- **Improved**: Application-specific technology-level adjustments (use of different $V_{\rm th}$ flavors and transistor widths) are used in ORION 2.0 to improve power estimation for SoC and high-performance applications. ORION 1.0 used a single set of parameters for all designs at a given technology node.
- **Updated**: Transistor sizes and capacitance values are updated in ORION 2.0 with new process technology files.

Most of today's chip prototypes, as well as virtual channel routers, are covered by ORION 2.0 models. In addition, different topologies, e.g., flattened butterfly [12], express virtual channel [15], etc. can be easily explored using ORION 2.0 models. In general, any topology that uses wormhole routers is supported by ORION 2.0 models; more significantly, several router microarchitectures such as token flow control router [14] have been modeled using different subcomponents of ORION 2.0 models. In addition, ORION 2.0 models have been incorporated to network simulators for efficient system-level design space exploration. For example, GARNET [1] incorporates ORION 2.0 power models. Various performance counters keep track of the amount of switching at various components of the network (i.e., for a given application) during simulation, and pass the activity factors to ORION models for power estimation.

The remainder of this paper is organized as follows. Section II describes ORION 2.0 dynamic and leakage power models, while Section III describes our proposed area model. In Section IV, we validate our models and develop closed-form power and area equations using nonlinear regression. Finally, Section V concludes the paper.

Manuscript received January 12, 2010; revised May 21, 2010; accepted November 02, 2010. This work was supported by the Gigascale System Research Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity. A preliminary version of this work appeared in the Proceedings of DATE, pp. 423–428, 2009.

A. B. Kahng is with the Departments of Computer Science and Engineering, and of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0404 USA (e-mail: abk@ucsd.edu).

B. Li was with Princeton University, Princeton, NJ 08544 USA. She is now with Intel Corporation, Hillsboro, OR 97124 USA (e-mail: bin.li@intel.com).

L.-S. Peh is with the Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139-4309 USA (e-mail: peh@csail.mit.edu).

K. Samadi is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0409 USA (e-mail: ksamadi@ucsd.edu).

Digital Object Identifier 10.1109/TVLSI.2010.2091686

1063-8210/$26.00 © 2011 IEEE

II. POWER MODELING

In ORION 2.0, we add: 1) clocking and link power models; 2) flip-flop-based FIFO power models; and 3) arbiter leakage power model to ORION 1.0. For the remaining components, we enhance or update existing ORION 1.0 models.

A. Dynamic Power Modeling

We derive detailed parameterized equations for estimating switching capacitance of: 1) clocking due to routers; 2) flip-flop-based FIFO buffers; 3) allocators and arbiters; and 4) physical links.

1) Clock: Clock distribution and generation comprise a major portion of power consumption in synchronous designs [8], representing up to 33% of power consumption in a high-performance router [10]. We estimate the term $C_{\text{clk}} = C_{\text{sram-fifo}} + C_{\text{flip-flop-fifo}} + C_{\text{pipeline-registers}} + C_{\text{wiring}}$, where $C_{\text{sram-fifo}}$, $C_{\text{flip-flop-fifo}}$, $C_{\text{pipeline-registers}}$, and C_{wiring} are capacitive loads due to SRAM-based FIFO buffers, flip-flop-based FIFO buffers, pipeline registers, and clock distribution wiring, respectively. Given that the load of the clock distribution network heavily depends on its topology, we assume an H-tree distribution style.

SRAM-Based FIFO Buffers: We adapt the original ORION 1.0 model for SRAM buffers for determining the precharge circuitry capacitive load on the clock network. In a SRAM FIFO with flit width fw, the total capacitance due to precharging circuitry, where P_r and P_w are the number of read and write ports, can be estimated as $C_{\text{sram-fifo}} = (P_r + P_w) \cdot fw \cdot C_{chg}$, where C_{chg} is precharging capacitance.

Flip-Flop-Based FIFOs: We assume a simple D-flip-flop (DFF) as the building block to construct the flip-flop-based FIFOs. In a B-entry flip-flop-based FIFO with flit width of fw bits, the capacitive load on the clock can be estimated as $C_{\text{flip-flop-fifo}} = fw \times B \times C_{\text{ff}}$.

Pipeline Registers: We assume DFF as the building block of the pipeline registers. In a router with flit width of fw bits and N_{pipeline} pipeline registers, the capacitive load on the clock due to pipeline registers is $C_{\text{pipeline-registers}} = N_{\text{pipeline}} \times C_{\text{ff}}$, where $N_{\text{pipeline}} = n_{\text{port}} \times fw$ for buffers (i.e., input and output) and crossbar components, $N_{\text{pipeline}} = 2\times(n_{\text{port}} \times n_{\text{vc}})^2$ for VC allocator, and $N_{\text{pipeline}} = n_{\text{port}} \times n_{vc} + n_{port}^2$ for switch allocator. C_{ff} is the flip-flop capacitance. n_{port} and n_{vc} are the number of ports and number of virtual channels, respectively.

Wire Load: We assume a buffered H-tree clock distribution within each individual router block. If the router block dimension is D (typically, tens of micrometers e.g., $D = 25$ μm in the router block of each tile in the Intel 80-core chip), the total wire capacitance of an L-level H-tree is $\sum_{i=0}^{L-1} 2^i \times D/2^{\lfloor i/2 \rfloor + 1} \times C_{\text{int}}$ where each term is (number of segments per level) × (fraction of D per segment at that level) × (router dimension D) × (per unit length wire capacitance C_{int}), e.g., for a five-level H-tree, the total wire capacitance is $(1 \times D/2 + 2 \times D/2 + 4 \times D/4 + 8 \times D/4 + 16 \times D/8) \times C_{\text{int}}$. In our studies, we use a fixed number of levels (equal to 5) in the H-tree; this both overestimates clock tree wiring cost (since an H-tree is more expensive than skew-bounded Steiner constructions) and underestimates [since some router configurations have significantly more than 32 leaves (sinks)]. However, since the flip-flops in a router have strong spatial clustering (e.g., in FIFOs), we have opted to use the fixed number of levels. The small value of D lessens the impact of this modeling error.

2) Flip-Flop-Based FIFO Buffers: FIFO buffers can be implemented as either SRAM or registers. The ORION 1.0 model supports only the use of SRAM-based FIFOs. We use flip-flops (FFs) as the building block of the registers. Register-based FIFOs can be implemented as shift-registers or as matrix of FFs.

a) Shift-register-based FIFOs: For a B-entry FIFO buffer, the shift-register based FIFO can be implemented as a series of B flip-flops. We consider both read and write operations. The write operation occurs at the tail of the shift register. Assuming the new flit is f_n and the old flit is f_o, the number of switched flip-flops is the Hamming distance between them. Therefore, the write energy is $E_{\text{write}} = H(f_n, f_o) \times E_{\text{switch}}^{ff}$, where E_{switch}^{ff} is the energy to switch one bit. We simply estimate the average switching activity as $\bar{H} = fw/2$; then, the average write energy is $\bar{E}_{\text{write}} = \bar{H} \times E_{\text{switch}}^{ff}$. The read operation has two steps: 1) reading the head flit into the crossbar which does not consume any energy in the buffer and 2) shifting all the subsequent flits one position toward the header. Hence, the average read energy is $\bar{E}_{\text{read}} = (fw - 1) \times \bar{E}_{\text{write}}$.

b) Matrix of FF FIFOs: A better approach to implement flip-flop-based FIFOs may be to use a matrix of flip-flops with write and read pointers as is done in SRAM-based FIFOs to avoid read and write energy consumption at every cycle due to shifts. To implement this, we add a control circuitry to an existing matrix of flip-flops to handle the operation of write/read pointers. The *write pointer* points to the head of the queue, and the *read pointer* points to the tail of the queue. The pointer advances one position for each write or read operation. To model power, we can synthesize the RTL of the above implementation and obtain corresponding power numbers with respect to different buffer size and flit width values.[1] To develop a closed-form power model, linear regression can be used to derive the power of the control unit as a function of buffer size and flit width. In this implementation, read energy is only due to pointer shifts, $\bar{E}_{\text{read}} = \bar{E}_{\text{pointer}}$, whereas write energy is due to pointer shifts and bit switches, $\bar{E}_{\text{write}} = \bar{H} \times \bar{E}_{\text{switch}}^{\text{ff}} + \bar{E}_{\text{pointer}}$, where $\bar{E}_{\text{pointer}}$ is the average energy to advance one position for read or write pointers.

3) Allocators and Arbiters: We modified the separable VC allocator microarchitecture in ORION 1.0 to optimize its power consumption. Instead of two stages of arbiters, we have a single stage of $n_{\text{port}} \times n_{\text{vc}}$ arbiters, each governing one specific output VC, where n_{port} and n_{vc} are the number of router ports and virtual channels, respectively. Instead of sending requests to all output VCs of the desired output port, an input VC first checks the availability of output VCs, and then sends a request for any one available output VC. The arbiters will resolve conflicts where multiple input VCs request the same output VC. This design has lower matching probability but does away with an entire stage of arbiters, significantly saving power. We also added a new VC allocator model in ORION 2.0 which models VC allocation as VC "selection" instead, as was first proposed in [13]. Here, a VC is selected from a queue of free VCs, *after* switch allocation. Thus, the complexity (delay, area, and power) of VC allocation no longer scales horribly with large numbers of VCs.

4) Physical Links: The dynamic power of links is primarily due to charging and discharging of capacitive loads (wire and input capacitance of next-stage repeater). In this work, we use a hybrid buffering solution that minimizes a linear combination of delay and power. We exhaustively evaluate a given objective function for a given number and size of repeaters, while searching for the optimal (i.e., and number, size) values. Dynamic power is given by $P_{\text{link}} = \alpha \cdot C_l \cdot V_{\text{dd}}^2 \cdot f_{\text{clk}}$, and $C_l = C_{\text{in}} + C_{\text{gnd}} + C_{\text{cc}}$, where P_{link}, α, C_l, V_{dd} and f_{clk} denote the link dynamic power, activity factor, load capacitance, supply voltage, and frequency, respectively. The load capacitance is the sum of the input capacitance of the next repeater (C_{in}), and the ground (C_{gnd}) and coupling (C_{cc}) capacitances of the wire driven. Here, link power refers to links incident to the router (i.e., connecting ports of the given router to ports of adjacent routers). We count only the input link power, so that when composing router power models for an entire NoC, there is no double-counting.

---

[1] We have developed a simple RTL code of a flip-flop-based FIFO implementation, and have added it to the latest release of ORION 2.0 [25].

B. Leakage Power Modeling

As technology scales to deep submicrometer processes, leakage power becomes increasingly important as compared to dynamic power. There is thus a growing need to characterize and optimize network leakage power as well. Chen *et al.* [7] proposed an architectural methodology for estimating leakage power. However, [7] only considered subthreshold leakage, whereas, from 65 nm and beyond, gate leakage gains importance and becomes a significant portion of the leakage power. We follow the same methodology proposed in [7] with the addition of gate leakage in our leakage analysis.

To derive an architectural leakage model, we can separate the technology-independent variables such as transistor width from those that depend on the specific process technology, $I_{\text{leak}}(i,s) = W(i,s) \cdot (I'_{\text{sub}}(i,s) + I'_{\text{gate}}(i,s))$, where I_{leak} is total leakage current. I'_{sub} and I'_{gate} are subthreshold and gate leakage currents per unit transistor width for a specific technology, respectively. $W(i,s)$ refers to the effective transistor width of component i at state s. We measure I'_{sub} and I'_{gate} for a variety of circuit components, input states, operating conditions (i.e., voltage and temperature), and different V_{th} flavors [i.e., high V_{th} (HVT), normal V_{th} (NVT), and low V_{th} (LVT)]. We compose the architectural leakage power model in a bottom-up fashion for each building block [7].

1) Arbiter Leakage Power: In ORION 2.0, we add arbiter leakage power and support matrix and round robin arbiters. Given a matrix arbiter with R requesters, its priorities may be represented by an $R \times R$ matrix, with a 1 in row i and column j if requester i has higher priority than requester j, and 0 if otherwise. Let req_i be the ith request, gnt_n the nth grant, and m_{ij} the element in the ith row and jth column in the matrix. Hence, the grant logic can be denoted as $gnt_n = req_n \times \prod_{i<n}(\overline{req_i} + \bar{m}_{\text{in}}) \times \prod_{i>n}(\overline{req_i} + \bar{m}_{ni})$. Then, we decompose the grant logic into elementary building blocks including NOR, INV, and D-flip-flops, and compute the leakage current for the entire arbiter as $I_{\text{leak}}(\text{arbiter}) = I_{\text{leak}}(\text{NOR2}) \cdot ((2R-1)R) + I_{\text{leak}}(\text{INV}) \cdot R + I_{\text{leak}}(\text{DFF}) \cdot R(R-1)/2$.[2] The previous equation can be readily obtained from the gate-level netlist of a given arbiter, if available. Hence, arbiter power can be computed as $P_{\text{leak}}(\text{arbiter}) = I_{\text{leak}}(arbiter) \cdot V_{\text{dd}}$, where V_{dd} is the supply voltage. Similarly, for a round-robin arbiter we break its corresponding grant logic into elementary building blocks (i.e., NOR and INV), and use D-FFs to store the priority bits.

2) Physical-Link Leakage Modeling: The leakage power of links is due to repeaters inserted in them. In repeaters, leakage occurs in both output states. NMOS devices leak when the output is high, while PMOS devices leak when the output is low. This is applicable for buffers also because the second stage devices are the primary contributors due to their large sizes. Leakage power has two main components: 1) subthreshold leakage and 2) gate-tunneling current. Both components depend linearly on device size and are modeled using linear regression with the values from SPICE simulations.

III. AREA MODELING

As area is an important economic concern in integrated circuit (IC) design, it needs to be estimated early in the design flow to enable design space exploration. We use a recent model proposed by [20] and the analysis in [17] to estimate the areas of transistors and gates such as inverters, NAND, and NOR gates.

A. Router Area

To estimate the router area, we basically compute the area of each of the building blocks and sum them up with an addition of 10% (rule of thumb) to account for global whitespace. For each building block, we first identify the implementation style of the block and then decompose the block into its basic logical elements (i.e., gate-level netlist). For example, for SRAM-based FIFOs, we can compute word line length using $L_{\text{word-line}} = fw \cdot (w_{\text{cell}} + 2(P_r + P_w)d_w)$, and bit line length using $L_{\text{bit-line}} = B \cdot (h_{\text{cell}} + (P_r + P_w)d_w)$, where $fw, B, w_{\text{cell}}, h_{\text{cell}}, d_w$, P_r, and P_w are flit width in bits, buffer size in flits, memory cell width, memory cell height, wire spacing, number of read ports, and number of write ports, respectively. Hence, the total area for a B-entry buffer is calculated as $\text{Area}_{\text{fifo}} = L_{\text{word-line}} \cdot L_{\text{bit-line}}$. For other router components, namely, crossbar and arbiter, we similarly decompose them into their circuit building blocks (i.e., gate-level netlist). Then, using the gate area model, we estimate the area of individual circuit component and compute the area of the entire block.

B. Link Area

The area occupied by links is due to wires and repeaters. We use the above-described gate area model to estimate the area of repeaters. The area of global wiring can be calculated as $\text{Area}_{\text{link}} = \text{fw} \times (w_w + s_w) + s_w$, where $\text{Area}_{\text{link}}$ denotes the wire area, fw is the flit width in bits, and w_w and s_w are the wire width and spacing computed from the width and spacing of the layer (global or intermediate) on which the wire is routed and from the design style.

IV. MODEL EVALUATION

Here, we provide further insight into our models with respect to: 1) different microarchitectural parameters; 2) different technology nodes and transistor types; 3) synthesis of router RTLs; and 4) two recent NoC prototypes. ORION 2.0 models can be broadly classified as *template-based*, that is, derived from a mix of circuit templates, e.g., matrix crossbar or SRAM-based FIFO. The following subsections give several "sanity" checks for these models.

A. Microarchitectural Parameters

Here, we investigate the impact of different microarchitectural parameters on router power and area. We demonstrate that our models behave as expected with respect to each parameter. Router microarchitectural components include: 1) buffers; 2) crossbar; 3) VC allocator; 4) switch allocator; 5) clock; and 6) link. The microarchitectural parameters for each router are: 1) buffer size per VC per port; 2) flit width; 3) number of virtual channels; and 4) number of ports.[3] For all of the experiments, we use a supply voltage of 1.1 V, switching activity of 0.3, and a clock frequency of 3 GHz in 65-nm technology. In each experiment, we only vary one microarchitectural parameter of interest and keep the others fixed. Nominal values for buffer size, flit width, number of virtual channels, and number of ports are 4 flits, 32 bits, 1 (i.e., wormhole configuration), and 5, respectively.

1) Buffer: Buffer power and area are affected by buffer size, flit width, number of VCs, and number of ports. When we vary buffer size, we expect buffer dynamic and leakage power to increase linearly, respectively. This is because buffer size linearly increases precharge capacitance load and the number of bitcell transistors. When we vary flit width, we expect buffer dynamic and leakage power to increase linearly, since flit width linearly increases the precharge and bitline capacitances as well as the number of bitcell transistors, respectively.

---

[2] For a given elementary building block, X, $I_{\text{leak}}(X)$ is calculated using $W(X)$, $I'_{\text{sub}}(X)$, and $I'_{gate}(X)$.

[3] We assume that the crossbar has the same number of ports as the router, so the number of router ports equals the number of crossbar ports.

On the other hand, as we increase the number of virtual channels, buffer dynamic power will not change since the number of flits arriving at each input port is the same. However, we expect buffer leakage power to increase linearly. This is because, in VC routers, there are n_{vc} queues in each input port, where n_{vc} is the number of virtual channels. If we increase number of ports, we expect buffer dynamic and leakage power to increase linearly. This is because the addition of a new port will add a new buffer set, i.e., with the same buffer size and flit width.

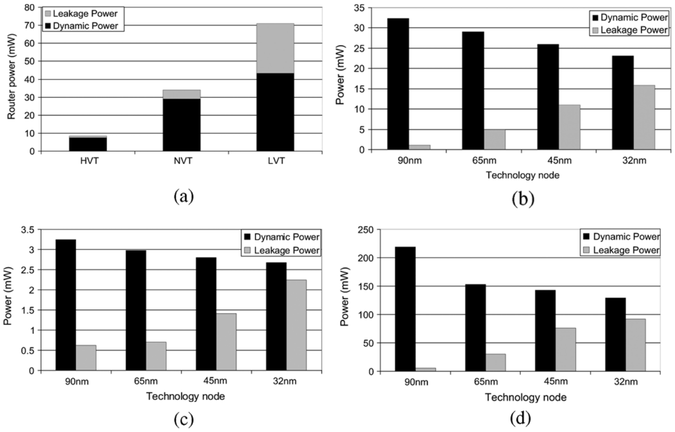
Buffer area also follows power trends as expected. As buffer size increases, we expect buffer area to increase linearly. This is because as buffer size increases by one unit, the number of flits per buffer also increases by one unit. In addition, buffer area changes linearly with flit width because flit width linearly increases the number of bitcells in each FIFO entry.

2) Crossbar: Crossbar power and area are affected by the number of router ports. If we increase number of ports, we expect dynamic and leakage power to increase quadratically. This is because a $N \times N$ crossbar allows arbitrary one-to-one connections between N input ports and N output port. Similarly for area, if we increase number of ports, we expect crossbar area to increase quadratically.

3) VC and Switch Allocator: If we increase the number of virtual channels, dynamic and leakage power are expected to increase linearly and quadratically, respectively. This is because the number of arbiters increases linearly with number of virtual channels. In addition, for each arbiter the request width increases linearly with number of virtual channels. Hence, leakage power increases quadratically with the number of virtual channels. Since the utilization rate of each arbiter is assumed to be inversely proportional to the number of virtual channels, dynamic power is expected to change linearly with the number of virtual channels.[4] In our experiments, we have assumed a two-stage separable VC allocator. For switch (SW) allocator, if we increase the number of VCs, dynamic power and leakage power are expected to increase linearly because in the SW allocator the request width of each arbiter increases linearly with respect to the number of virtual channels.

Also, if we increase the number of ports, we expect VC allocator dynamic and leakage power to increase quadratically. This is because the request width for each arbiter in the second stage of arbitration increases linearly with respect to the number of ports, and the number of such arbiters is proportional to the number of ports. Hence, VC allocator power consumption is quadratically dependent on the number of ports. Similarly, VC allocator area is expected to increase quadratically with the number of virtual channels and the number of ports. On the other hand, if we increase the number of ports, we expect dynamic and leakage power to increase quadratically. This is because the request width for each arbiter in the second stage of arbitration increases linearly with respect to the number of ports, and the total number of arbiters is proportional to the number of ports. Similarly, the SW allocator area changes linearly and quadratically, respectively, with the number of virtual channels and number of ports.[5]

In addition to the above "sanity" checks, we evaluate our leakage power model by verifying that the leakage power density (defined as total leakage power/total gate-width) remains the same as we change each of the microarchitectural parameters. We observe that leakage power density for buffer, crossbar, and arbiter is 0.0003 mW/μm.



(a) (b) (c) (d)

Fig. 1. Power consumption versus technology parameters. (a) Power consumption versus transistor type. (b) Router power versus technology node with NVT transistors. (c) Router power versus technology node with HVT transistors. (d) Router power versus technology node with LVT transistors.

B. Technology Parameters

In ORION 2.0, we include transistor sizes and capacitance values for three combinations of V_{th} and transistor width: 1) large transistor size with low V_{th} (LVT) for high-performance designs; 2) nominal transistor size with nominal V_{th} (NVT) for general-purpose designs; and 3) small transistor size with high V_{th} (HVT) for low-power designs. When transistor type changes from HVT to NVT to LVT, dynamic power is expected to increase due to the increase in transistor width (i.e., assuming a fixed technology), and leakage power is expected to increase due to the increase in transistor width and decrease of threshold voltage, as confirmed in Fig. 1(a). For the experiments in Fig. 1, we use a router with five ports, two virtual channels, buffer size of 4, and 32-bit flit width; for HVT, NVT, and LVT, we use (0.8 V, 0.2 GHz), (1.0 V, 1 GHz), and (1.1 V, 3 GHz), respectively.

Also, for a given transistor type, dynamic power is expected to reduce as technology advances due to smaller area, and leakage power is expected to increase due to leakier devices as confirmed in Fig. 1(b)–1(d).[6] We use similar microarchitectural and transistor type values, but vary technology node from 90 nm down to 32 nm.

C. Router RTL Synthesis

Here, we further validate the trend of our models by comparing them against those of router RTL synthesis. We use *Netmaker*, which is a library of fully synthesizable parameterized network-on-chip (NoC) implementations [27]. We pick a baseline VC router in which VC allocation and switch allocation are performed sequentially in one clock cycle.

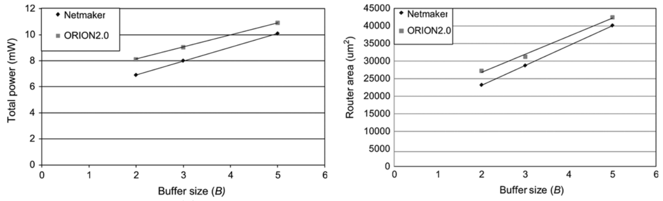
Using automation scripts, we vary the above parameters and generate corresponding RTL for each combination of parameters. We then synthesize the RTL codes using TSMC 65 nm general-purpose library. The difference between ORION 2.0 and the synthesized router results are due to the fact that ORION 2.0 does not capture the effects of the implementation flows. Modern IC implementation flows incorporate powerful logic synthesis and physical synthesis transformation (i.e., logic restructuring and gate sizing) to satisfy the power and performance constraints. The detailed impacts of such transformations are difficult to capture at early stages of the design, where not all of the implementation information is available. However, Fig. 2 show that ORION 2.0

---

[4] Note that VC allocator dynamic power is equal to arbiter utilization rate multiplied by the product of per-arbiter dynamic power and the total number of arbiters. Hence, VC allocator dynamic power is linearly dependent on number of virtual channels (i.e., $1/n_{\mathrm{vc}} \times n_{\mathrm{vc}} \times n_{\mathrm{vc}} = n_{\mathrm{vc}}$).

[5] In addition, we observe that the clock and link power and area models follow the expected trends.

[6] Our estimations for 45- and 32-nm technologies are derived using scaling factors from ITRS [21]; hence, they may not accurately represent production processes.



(a) (b)

Fig. 2. ORION 2.0 versus RTL synthesis comparison. (a) Router total power versus buffer size. (b) Router area versus buffer size

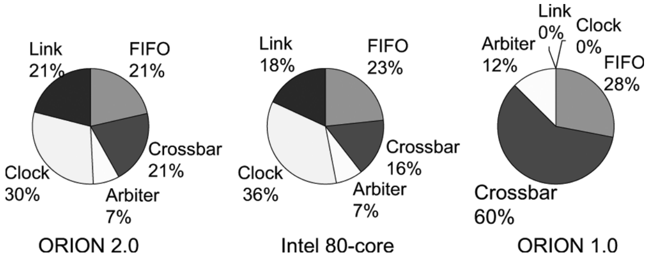


Fig. 3. Power breakdown of the Intel 80-core chip versus estimations from ORION 1.0 and ORION 2.0 models.

models' trends (see the previous subsections) match those of synthesized routers.

D. Real Router Designs

Finally, we also validate our models by comparing them with post-layout and pre-layout simulations of recent NoC prototypes: 1) Intel 80-core Teraflops chip, targeted for high-performance chip multiprocessors (CMPs), and 2) Intel Scalable Communications Core (SCC) chip, targeted for ultralow-power multiprocessor systems-on-chip (MPSoCs). As noted in the Introduction, there is up to an 8× difference between ORION 1.0 estimations (per component) and Intel 80-core chip silicon measurements. Also, the estimated total power is about 10× less than the actual. Again, ORION 1.0 does not include clock and link power models. Fig. 3 shows the percentage of each of power component in an Intel 80-core chip and the same statistic from ORION 1.0 and ORION 2.0 models. We observe that ORION 2.0 more accurately represents the impact of each individual component.[7]

We use switching activity of 0.15 for both test cases. The estimated total power consumption, using ORION 2.0 models, is within -7% and 11% of the Intel 80-core post-layout and Intel SCC pre-layout power estimations, respectively. In addition, the estimated total area, using ORION 2.0 models, is within -23.5% and 25.3% of the Intel 80-core and Intel SCC, respectively.

V. CONCLUSION

Accurate estimation of power and area of interconnection network routers in early phases of the design process can drive effective NoC design space exploration. ORION 1.0, an existing power model for NoC routers developed back in 2002, is inaccurate for current and future technologies and can lead to misleading design targets. In ORION 2.0, we have proposed more accurate power and area models for NoC routers that are easily usable by system-level designers. We have also developed a reproducible methodology for extracting the inputs to our models from different reliable sources. In addition, we have validated our new models with respect to different microarchitectural and technology parameters, synthesis of router RTLs, and two recent Intel chips. By maintaining the user interfaces of the original ORION 1.0 while substantially improving accuracy and fidelity, we predict that ORION 2.0 will make a significant impact on future NoC research and design.

REFERENCES

[1] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," *Proc. ISPASS*, pp. 33–42, 2009.
[2] S. Bhat, "Energy models for network-on-chip components," M.S. thesis, Dept. of Mathematics and Computer Science, RIT, Eindhoven, The Netherlands, 2005.
[3] A. Bona, V. Zaccaria, and R. Zafalon, "System level power modeling and simulation of high-end industrial network-on-Chip," *Proc. DATE*, pp. 318–323, 2004.
[4] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," *Proc. ISCA*, pp. 83–94, 2000.
[5] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," *Proc. MICRO*, pp. 93–104, 2003.
[6] L. P. Carloni, A. B. Kahng, S. Muddu, A. Pinto, K. Samadi, and P. Sharma, "Interconnect modeling for improved system-level design optimization," *Proc. ASPDAC*, pp. 258–264, 2008.
[7] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnect networks," *Proc. ISLPED*, pp. 90–95, 2003.
[8] D. E. Duarte, N. Vijaykrishnan, and M. J. Irwin, "A clock power model to evaluate impact of architectural and technology optimization," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 6, pp. 844–855, Jun. 2002.
[9] N. Eisley and L.-S. Peh, "High-level power analysis for on-chip networks," *Proc. CASES*, pp. 104–115, 2004.
[10] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE MICRO*, pp. 51–61, 2007.
[11] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," *Proc. DATE*, pp. 423–428, 2009.
[12] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: A cost-efficient topology for high-radix networks," *Proc. ISCA*, pp. 126–137, 2007.
[13] A. Kumar, P. Kundu, A. Singh, L.-S. Peh, and N. K. Jha, "A 4.6 Tbits/s 3.6 GHz single-cycle NoC router with a novel switch allocator in 65 nm CMOS," *Proc. ICCD*, pp. 63–70, 2007.
[14] A. Kumar, L.-S. Peh, and N. K. Jha, "Token flow control," *Proc. MICRO*, pp. 342–353, 2008.
[15] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," *Proc. ISCA*, pp. 150–161, 2007.
[16] C. S. Patel, S. M. Chai, S. Yalamanchili, and D. E. Schimmel, "Power constrained design of multiprocessor interconnection networks," *Proc. ICCD*, pp. 408–416, 1997.
[17] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, CACTI 5.1 HP Laboratories, Tech. Rep. HPL-2008-20, 2008.
[18] H. Wang, X. Zhu, L.-S. Peh, and S. Malik, "ORION: A power-performance simulator for interconnection networks," *Proc. MICRO*, pp. 294–395, 2002.
[19] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and use of simplepower: A cycle-accurate energy estimation tool," *Proc. DAC*, pp. 340–345, 2000.
[20] H. Yoshida, D. Kaushik, and V. Boppana, "Accurate pre-layout estimation of standard cell characteristics," *Proc. DAC*, pp. 208–211, 2004.
[21] International Technology Roadmap for Semiconductors. [Online]. Available: http://public.itrs.net/
[22] LEF/DEF Language. [Online]. Available: http://openeda.si2.org/projects/lefdef
[23] *"Liberty File Format, Liberty NCX User Guide,"* ver. B-2008.06-SP2.
[24] Predictive Technology Model. [Online]. Available: http://www.eas.asu.edu/~ptm/
[25] ORION 2.0. [Online]. Available: http://vlsicad.ucsd.edu/ORION/
[26] Magma BlastPower. [Online]. Available: http://www.magma-da.com/products-solutions/
[27] Netmaker. [Online]. Available: http://www-dyn.cl.cam.ac.uk/~rdm34/wiki/index.php?title=Main_Page

---

[7] We do not have access to the power breakdown for the Intel SCC design.