ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration

Andrew B. Kahng[‡†], Bin Li[+], Li-Shiuan Peh[+] and Kambiz Samadi[†]
[‡] CSE and [†] ECE Departments, University of California, San Diego, La Jolla, CA
[+] EE Department, Princeton University, Princeton, NJ
Email: {abk,ksamadi}@ucsd.edu, {binl,peh}@princeton.edu

Abstract

As industry moves towards many-core 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. ORION [29] was amongst the first NoC power models released, and has since been fairly widely used for early-stage power estimation of NoCs. However, when validated against recent NoC prototypes – the Intel 80-core Teraflops chip and the Intel Scalable Communications Core (SCC) chip – we saw significant deviation that can lead to erroneous NoC design choices. This prompted our development of ORION 2.0, an extensive enhancement of the original ORION models which includes completely new subcomponent power models, area models, as well as improved and updated technology models. Validation against the two Intel chips confirms a substantial improvement in accuracy over the original ORION. A case study with these power models plugged within the COSI-OCC NoC design space exploration tool [23] confirms the need for, and value of, accurate early-stage NoC power estimation. To ensure the longevity of ORION 2.0, we will be releasing it wrapped within a semi-automated flow that automatically updates its models as new technology files become available.

1 Introduction

Power has become the most critical design constraint. Increasing power consumption and design complexity have led designers to adopt multi-core designs in chip multiprocessors (CMPs) [16, 26] and multiprocessor systems-on-chip (MPSoCs) [36, 21]. As the demand for bandwidth increases in these systems, NoCs have been introduced to meet the increasing communication demand among cores [8]. However, with increasing demand for network bandwidth, the power that an interconnection network consumes will also be substantial [13]. The International Technology Roadmap for Semiconductors (ITRS) predicts that future generations of high-end VLSI designs will operate in 10-20 GHz range with the communication between cores in Gbit/s [40]. This requires designers to work within a tight power budget. To aid designers in early stages of design, architectural power models were proposed for rough power estimations (see Section 2).

Architectural power estimation is extremely important in order 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 various high-level optimizations, which have been shown to have much more significant impact on power than low-level optimizations. To tackle this problem, ORION, a set of architectural power models for network routers, was proposed in [29] in 2002, and have been fairly widely used for early-stage NoC power estimation in literature and industry. Despite the increase in complexity of today's designs, ORION's original power models have not been updated or enhanced. In a comparison between ORION 1.0 and the Intel 80-core Teraflops chip we notice 6.8*X* difference in reported total power values (see Section 5). This highlights the need for more accurate architectural power models to aid designers in making early-stage NoC design decisions.

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 format [38], LEF [39]), as well as extrapolatable process models (such as, PTM (Predictive Technology Model) [37], or ITRS. ORION 1.0 collects inputs from *ad hoc* sources to drive its internal power models. We see a clear need for a semi-automated flow (i.e., using shell scripting) to extract technology inputs from reliable sources, to ease the updating of models as new technology files become available.

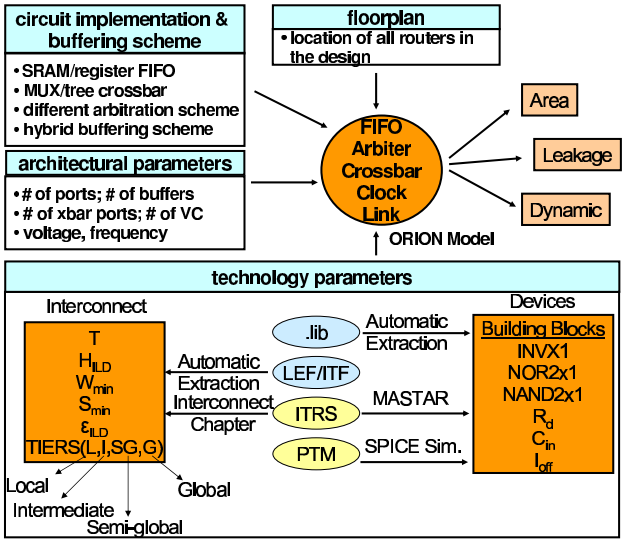

Figure 1: ORION 2.0 modeling methodology.

Table 1: ORION 2.0 contributions vs. ORION 1.0.

Component	App-Specific Sizing	Dynamic Power	Leakage Power	Area
SRAM-FIFO	Improved	Orig	Orig	New
Reg-FIFO	Improved	New	New	New
Crossbar	Improved	Orig	Orig	New
Arbiter	Improved	Orig	New	New
Clock	Improved	New	New	New
Link	Improved	New	New	New

The above two factors prompted ORION 2.0 and its two key goals: (1) To update and enhance ORION's power and area modeling accuracy; and (2) To encompass ORION 2.0 within a semi-automated flow so that ORION can be continuously maintained and updated easily. Figure 1 shows the usage model and modeling flow of ORION 2.0 with its main inputs and outputs.

In this paper, we substantially improve the original ORION 1.0, to allow it to accurately estimate power for designs beyond the 65nm technology node. ORION 2.0 surgically tackles various deficiencies of ORION 1.0 highlighted through our validation with two Intel chips (Section 5) and our close interactions with both the chip designers and developers of ORION 1.0. Table 1 summarizes the contributions of ORION 2.0 beyond the original ORION 1.0, highlighting the extensive modifications:

New:

- Flip-flop and clock power models (both leakage and dynamic) are added. Flip-flop power models will enable the faithful modeling of flip-flop-based FIFOs in addition to the SRAM-based implementation in ORION 1.0. Clock power is a major component of overall chip power especially in high-performance applications [13], but was omitted in ORION 1.0.

- Link power models are added, leveraging accurate models recently developed in [6] as links are a major contributor to NoC power. Prior existing work on link power and delay modeling [11, 12] focus on minimum-delay buffer insertion, whereas we adopt a hybrid solution which minimizes

a weighted product of delay and power. ORION 1.0 did not have a parameterized link model.

- Virtual-channel allocator microarchitecture in ORION 1.0 is modified to optimize its power consumption. Also, a new VC allocation model, based on the microarchitecture and pipeline proposed in [17], is added in ORION 2.0.
- Arbiter leakage power, previously not covered in ORION 1.0, is now modeled.
- An accurate area model is added, allowing for detailed router floorplanning which enhances the accuracy of early-stage power estimation.
- An automatic flow for extracting technology parameters from standard technology files (e.g., Liberty format [38], LEF [39]), as well as extrapolatable models of process (e.g., PTM [37], ITRS [40]) is added to allow ORION 2.0 to be easily and continuously updated in the future.

Improved:

- Application-specific technology-level adjustments (used different V_{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 – industry SPICE models and ITF (Interconnect Technology Format) – instead of ad hoc scaling factors as in ORION 1.0.

Our power model is validated against the Intel 80-core Teraflops chip and the Intel Scalable Communications Core, and is within 7% and 11% of the corresponding total power values. We also integrated ORION 2.0 models in COSI-OCC communication synthesis infrastructure and found that accurate models substantially affect the NoC system-level design exploration.

The remainder of this paper is organized as follows. In Section 2 we contrast against prior related work. Section 3 describes ORION 2.0 dynamic and leakage power models, while Section 4 describes our proposed area model. In Section 5 we validate our models against the Intel 80-core chip [13] and the Intel SCC chip [14], and show the impact of the new models on achievable NoC configurations. Finally, Section 6 concludes the paper.

2 Related Work

Power modeling can be carried out at different levels of fidelity, trading off modeling time with accuracy, ranging from real-chip power measurements [15], to pre and post-layout transistor-level simulations [34], to RTL power estimation tools [32] to early-stage architectural power models [5, 30, 29, 10]. 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 an architectural power model takes on the order of seconds. Circuit-level power estimation tools, though providing excellent accuracy, has even longer simulation times, and require even more substantive development effort. These shortcomings have prompted a plethora of early-stage architectural power models and simulators, such as the widely-used Wattch [5] and SimplePower [30] for uniprocessor power modeling. These models allowed computer architects and designers to factor in power when making early-stage design decisions.

Power models at different abstraction levels have also been proposed for a variety of network fabrics in the past. There have been several RTL-level NoC power models proposed [2, 1]. In [2], a RTL level power model for NoCs was developed by first extracting the SPICE level netlist from the layout and then integrating the characterized values into the VHDL based RTL design. In [1], an accurate power characterization of a range of NoC routers was performed through RTL synthesis and place and route using standard ASIC tool flow. Both studies inherit limitations of RTL-level power simulation: simulation time is slow, and requires detailed RTL modeling, making them unsuitable for early-stage NoC design space explorations. Besides, power models cannot be targeted for future technology nodes.

At the architecture level, Patel *et al.* [20] first proposed a power model for interconnection networks, deriving its power estimates based on transistor count. As the model is not instantiated with architectural parameters, it cannot be used to explore tradeoffs in router microarchitecture design. ORION [29], an early-stage architectural power model for NoCs, was originally proposed and released in 2002, and has since been fairly widely used in academia and incorporated into industry toolchains (Intel, AMD, IBM, Freescale). Bona *et al.* [3] also presented a methodology for automatically generating the energy models for on-chip communication infrastructure at system level. But the focus is on bus based and crossbar based communication for SoC. Bhat *et al.* [4] proposed an architecture level regression analysis model for different router components based on energy numbers obtained from simulations using MAGMA tools. Besides the downsides of RTL-based models, the router architecture varieties and process technologies modeled were limited compared to ORION 1.0.

3 Power Modeling

In this section, we derive architectural-level parameterized power models for major router building blocks including FIFO buffers, crossbar switches, and arbiters. In addition, we model the power due to clocking of router blocks and links between routers. These power components form up to 94% of total power consumed in a high-performance router used in the Intel 80-core Teraflop chip [13] in the 65nm technology node. The communication power is significant at 28% of each processing tile's total power. As shown in Figure 2, clocking power, 33%, is the largest component of router power, with the FIFO buffers the second largest component at 22%. Power due to physical links, crossbar switch, and arbiter come next at 17%, 15% and 7%, respectively.[1] In ORION 2.0 we add (1) clocking and link power models, (2) flip-flop-based FIFO power models, and (3) arbiter leakage model. For the rest of the components we enhance/update current ORION models. In this section we first describe our dynamic power modeling and then present our leakage power modeling with specific analysis of arbiter leakage power model.

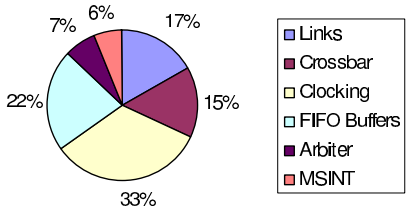

Figure 2: Router power breakdown at 4 GHz, 1.2 V, and 110°C [13].

3.1 Dynamic Power Modeling

Dynamic power consumption in CMOS circuits is formulated as $P = E \cdot f_{clk}$, where energy $E = \frac{1}{2}\alpha C V_{dd}^2$, with f_{clk} the clock frequency, α the switching activity, C the switched capacitance, and V_{dd} the supply voltage. We derive detailed parameterized equations for estimating switching capacitance of (1) register-based FIFO buffers, (2) clocking due to routers, and (3) physical links.

3.1.1 Clock

Clock distribution and generation comprise a major portion of power consumption in synchronous designs [9], representing up to 33% of power consumption in a high-performance router [13].

We estimate the term C_{clk}, as shown in Equation 1. Throughout our modeling approach it is assumed that all components are built using static CMOS gates. Given that the load of the clock distribution network heavily depends on its topology, we assume an H-tree distribution style

$$C_{clk} = C_{sram-fifo} + C_{pipeline-registers} + C_{register-fifo} + C_{wiring} \quad (1)$$

where $C_{sram-fifo}$, $C_{pipeline-registers}$, $C_{register-fifo}$, and C_{wiring} are capacitive loads due to memory structures, pipeline registers, FIFO registers, and clock distribution wiring respectively.

Memory structures. We adapt the original ORION model for SRAM buffers for determining the precharge circuitry capacitive load on the clock network. The pre-charging circuit is just the pre-charging transistor, T_c, which commonly is just a single PMOS.

[1] Power breakdown numbers include both dynamic and leakage power values.

Hence, its capacitance, C_{chg}, is due to its gate and drain end capacitances, $C_g(T_c)$ and $C_d(T_c)$ respectively as shown in Equation 2. In an SRAM FIFO with B buffers and flit size F, the total capacitance due to pre-charging circuitry can be derived using Equation 3, with P_r and P_w being the number of read and write ports, respectively.

$$C_{chg} = C_g(T_c) + C_d(T_c) \quad (2)$$

$$C_{sram-fifo} = (P_r + P_w) \cdot F \cdot B \cdot C_{chg} \quad (3)$$

Pipeline registers. Typical interconnection network routers have different pipeline stages. To advance, each flit must proceed through the steps of: (1) routing computation, (2) virtual-channel (VC) allocation, (3) switch allocation, and (4) switch traversal.[2] We assume DFF as the building block of the pipeline registers. In a router with flit size of F bits and $N_{pipeline}$ pipeline stages, the capacitive load on the clock due to pipeline registers is

$$C_{pipeline-registers} = N_{pipeline} \cdot F \cdot C_{ff} \quad (4)$$

where C_{ff} is the flip-flop capacitance and is extracted from 65nm *HP* (high-performance) and *LP* (low-power) libraries.

Register-based FIFOs. FIFO buffers can be implemented as a series of flip-flops. We assume simple DFF to construct the FIFO. In a B-entry register-based FIFO with flit size of F bits, the capacitive load on the clock can be computed as:

$$C_{register-fifo} = F \cdot B \cdot C_{ff} \quad (5)$$

For the registers we assume D flip-flop (DFF) is used as the building block. We obtain the capacitance value across different drive strengths from TSMC 65nm *G* and *LP* standard cell library data sheets.[3] Architectural parameters change the effective loading of each gate in the design. Hence, to use the appropriate drive strength for the registers we use their load capacitance and timing requirements. In this work, we assume minimum-size DFFs are used in all the registers.

Wiring load. For a 5-level H-tree clock distribution, total wire capacitance is given in Equation 6, where C_{int} is the per-unit-length wire capacitance and D is the chip dimension.

$$C_{wiring} = (\frac{16}{2}D + \frac{1\times 8}{2}D + \frac{2\times 4}{2}D + \frac{4\times 2}{2}D + \frac{8\times 1}{2}D) \cdot C_{int} \quad (6)$$

3.1.2 Register-Based FIFO Buffers

FIFO buffers consume up to 22% of the total router power in [13]. As mentioned earlier, FIFO buffers can be implemented as either SRAM or shift registers. The ORION 1.0 model supports only the use of SRAM-based FIFOs. We use flip-flops as the building block of the shift registers [28]. Hence, a B-entry FIFO buffer can be implemented as a series of B flip-flops (FF).

Write operation. 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:

$$E_{write} = H(f_n, f_o) E^{ff}_{switch} \quad (7)$$

where E^{ff}_{switch} is the energy to switch one bit. To simplify the analysis, let $\overline{H}$ denote the average switching activity; then, the average write energy is:

$$\overline{E}_{write} = \overline{H} \cdot E^{ff}_{switch} \quad (8)$$

Read operation. The read operation has two steps:

1. The flit stored at the header of the buffer is read into the crossbar. Since the header of the buffer is directly connected to the input port of the crossbar, this step does not consume any energy in the buffer.
2. Subsequent flits in the buffer are shifted one position towards the header. If the buffer holds n flits before the read operation, n-1 flip-flop writes are performed to shift the data.

Hence, the average read energy is:

$$\overline{E}_{read} = (n-1) \cdot \overline{E}_{write} \quad (9)$$

We obtain the capacitance value across different drive strengths from TSMC 65nm *G* and *LP* standard cell library data sheets.

3.1.3 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 pv arbiters, each governing one specific output VC, where p and v are the number of input physical and virtual channels, respectively. Instead of sending requests to all output VCs of the desired output port, input VCs first check the availability of output VCs, then send a request for any one available output VC. The arbiters will resolve conflicts where multiple input VCs request for 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 [17]. In this approach, the pipeline first goes through switch allocation, before selecting an available VC, effectively simplifying VC allocation to reading from a queue of available VCs. Power consumed thus becomes largely invariant to actual number of VCs, and can be modeled as a constant (derived through RTL synthesis) that will be scaled across process technologies. For example, in a 65nm 5×5 router with two VCs, the separable allocator consumes 8.47mW as opposed to 0.167mW consumed by VC selection model with V_{dd}=0.9V, f_{clk}=5.1GHz, and α=0.1. The VC selection power value is derived from the synthesis of the entire VC allocator using *Synopsys Design Compiler v2007.12-SP5* [33].

3.1.4 Physical Link

The dynamic power of links is primarily due to charging and discharging of capacitive loads (wire and input capacitance of next-stage repeater). Internal power dissipation, arising from charging and discharging of internal capacitances and short-circuit power, is noticeable for repeaters only when the input slew times are extremely large. Link power is a major component of a router total power (i.e., 17% [13]). Previous works such as [12, 11] only use delay as the objective for buffer insertion. This results in large repeaters and significantly increases the power consumption. 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 (number, size) values. Dynamic power is given by the well-known equations:

$$P_{link} = \alpha \cdot C_l \cdot V^2_{dd} \cdot f_{clk} \quad (10)$$

$$C_l = C_{in} + C_{gnd} + C_{cc} \quad (11)$$

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. C_{in} can be reliably obtained from industry Liberty files. The unit C_{gnd} and C_{cc} are also extracted from industry LEF files.

3.2 Leakage Power Modeling

As technology scales to deep sub-micron 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 65nm

[2] The number of pipeline stages can be different for various applications and is a function of clock frequency. We assume this is input by the user.

[3] TSMC represents the high-performance domain with *G* library.

Table 2: I'_{sub} and I'_{gate} (per-micron of gate width) for each fundamental circuit component i at different input state s, 25°C and for high V_{th} flavor.

i	s	I'_{sub} (A)	I'_{gate} (A)
NMOS	0	1.097e-07	4.622e-09
PMOS	1	3.172e-07	3.291e-09
INV	0	1.097e-07	4.622e-09
	1	3.172e-07	3.291e-09
NAND2	00	7.098e-08	3.549e-09
	01	1.134e-07	5.103e-09
	10	1.342e-07	1.194e-08
	11	1.766e-07	1.625e-08
NOR2	00	1.971e-07	6.701e-09
	01	1.034e-07	4.343e-09
	10	1.412e-07	8.048e-09
	11	7.245e-08	6.448e-09

and beyond gate leakage gains importance and becomes a significant portion of the leakage power. This is even more visible for high-performance applications where gate oxides are much thinner (i.e., ~1.5nm in 65nm *HP* library).

We follow the same methodology proposed in [7] with addition of gate leakage in our leakage analysis. We also use different V_{th} flavors to better represent leakage power consumption for different applications (i.e., high-performance vs. low-power).

To derive an architectural leakage model, we can separate the technology-independent variables such as transistor width from those that stay invariant for a specific process technology:

$$I_{leak}(i,s) = W(i,s) \cdot (I'_{sub}(i,s) + I'_{gate}(i,s)) \quad (12)$$

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., HVT, NVT, and LVT). We compose architectural leakage power model in a bottom-up fashion for each building block [7].

3.2.1 Derivation of I_{leak}

For each component i and input state s, we simulate I'_{sub} and I'_{gate} using HSPICE and 65nm foundry SPICE model with corresponding technology parameters. Table 2 lists I'_{sub} and I'_{gate} simulated for each fundamental circuit component i (leakage currents differ at different states due to stacking and body biasing effects). Circuit structures can then be hierarchically composed from these fundamental circuit components.

3.2.2 Arbiter Leakage Modeling

We applied our methodology to the major building blocks of interconnection networks – buffers, crossbar, and arbiter. Here, we walk through our arbiter modeling to demonstrate the methodology. We model three types of arbiters: (1) matrix, (2) round-robin, and (3) queuing. Here, we explain just the matrix arbiter. For an arbiter with R requesters, you can represent its priorities 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 i^{th} request, gnt_n the nth grant, and m_{ij} the i^{th} row and jth column elements in the matrix. Using these variables [29],

$$gnt_n = req_n \times \prod_{i<n} (\overline{req_i} + \overline{m_{in}}) \times \prod_{i>n} (\overline{req_i} + \overline{m_{ni}}) \quad (13)$$

Input state probabilistic analysis. We analyze the probability distribution of each input state of a circuit component by examining how architectural units function. Given the I'_{sub} and I'_{gate}, and the probabilities of each input state $Prob(i,s)$, the leakage current for a building block is:

$$I_{leak}(Block) = \sum_i \sum_s Prob(i,s) \cdot W(i,s) \cdot (I'_{sub}(i,s) + I'_{gate}(i,s)) \quad (14)$$

Input state simulation. Input states can also be tracked through network simulation.

$$I_{leak}(Block,t) = \sum_i W(i,s(t)) \cdot (I'_{sub}(i,s(t)) + I'_{gate}(i,s(t))) \quad (15)$$

where $I_{leak}(Block,t)$ is the leakage current at time t, and $s(t)$ is the state of circuit type i at time t within this circuit block. Finally, we can estimate total leakage current of an arbiter (Equation 16) while its power is leakage current multiplied by supply voltage (Equation 17).

$$I_{leak}(arbiter) = I_{leak}(NOR2) \cdot ((2R-1)R) + \\ + I_{leak}(INV) \cdot R + I_{leak}(DFF) \cdot \frac{R(R-1)}{2} \quad (16)$$

$$P_{leak}(arbiter) = I_{leak}(arbiter) \cdot V_{dd} \quad (17)$$

3.2.3 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. Thus, leakage power can be calculated using [6]:

$$p_s = \frac{p_s^n + p_s^p}{2} \quad (18)$$

$$p_s^n = \kappa_0^n + \kappa_1^n . w_n \quad (19)$$

$$P_s^p = \kappa_0^p + \kappa_1^p . w_p \quad (20)$$

where p_s^n and p_s^p are the leakage power for NMOS and PMOS devices, respectively, and κ_0^n = -6.034, κ_1^n = 26.561, $\kappa_0^p = 1.238$ and $\kappa_1^p = 27.082$ are coefficients determined using linear regression against *65nm LP* library. State-dependent leakage modeling can also be performed using Equations (19) and (20) separately.

3.3 Model Inputs and Technology Capture

ORION 1.0 used a set of fixed transistor sizes and capacitance values across designs with different target frequencies (i.e., high-performance and low-power). In ORION 2.0 we include data for three device types: (1) high-performance (HP), (2) low standby power (LSTP), and (3) low operating power (LOP) as defined in ITRS [40]. For wires we follow the approach in the [6] comprehending key interconnect circuit and layout design styles, including a power-efficient buffering technique that overcomes unrealities of previous delay-driven buffering techniques.

In addition, the most critical gap in existing system-level and NoC optimizations has been the lack of well-defined pathways to capture necessary technology and device parameters from the wide range of available sources. Since exploration of the system-level performance and power envelope is typically done for current and future technologies, the models that drive system-level design must be derivable from standard technology files (e.g., Liberty, ITF, LEF) and process models that can be extrapolated (e.g., PTM, ITRS). Earlier works on NoC design space exploration and synthesis [22, 24] collect inputs from *ad hoc* sources to drive internal models of performance, power and area. In contrast, we have developed a semi-automated flow (i.e., using shell scripting) to extract technology inputs from the above reliable sources. In the absence of such sources, we use predictive models and SPICE simulations to generate appropriate scaling factors.

4 Area Modeling

With the increase in number of cores on a single design, the area occupied by the communication components such as links and routers increases. As area is an important economic incentive in IC (integrated circuit) design, it needs to be estimated early in the design flow to enable design space exploration. In this section we present accurate models for router and link area.

Table 3: Process and technology input parameters used in the gate area model [27].

Parameter	Description
H_{n-diff}	Maximum height of n-diffusion
H_{p-diff}	Maximum height of p-diffusion
$H_{gap-same}$	Minimum gap between diffusions of the same type
$H_{gap-opp}$	Minimum gap between n and p diffusions
$H_{power-rail}$	Height of V_{dd} and V_{ss} rails
W_{poly}	Minimum width of poly
S_{pp}	Minimum poly-to-poly spacing
$W_{contact}$	Contact width
S_{pc}	Minimum poly-to-contact spacing

4.1 Gate Area

We use a recent model by [31] and the analysis in [27] to estimate the areas of transistors and gates such as inverters, NAND, and NOR gates. This is a fast technique to estimate standard cell characteristics before the cells are laid out. Figure 3 shows the layout model that has been used in [31]. Table 3 shows the process- and technology-level input parameters required by this gate area model. When a transistor exceeds a certain maximum value, the transistor assumed to be folded. Given the width of an NMOS, W_n, the number of folded transistors can be calculated as follows:

$$N_{folded-transistor} = \lceil \frac{W_n}{H_{n-diff}} \rceil \quad (21)$$

The equation for total diffusion width of $N_{stacked}$ transistors when they are not folded is given by the following equation:

$$W_{diffusion-area} = 2(W_{contact} + 2S_{pc}) + N_{stacked}W_{poly} + (N_{stacked} - 1)S_{pp} \quad (22)$$

total diffusion width of $N_{stacked}$ transistors when they are folded is given by the following equation:

$$W_{diffusion-area} = N_{folded-transistor}(2(W_{contact} + 2S_{pc}) + N_{stacked}W_{poly} + (N_{stacked} - 1)S_{pp}) \quad (23)$$

and finally the height of a gate is calculated using the following equation:

$$H_{gate} = H_{n-diff} + H_{p-diff} + H_{gap-opp} + 2H_{power-rail} \quad (24)$$

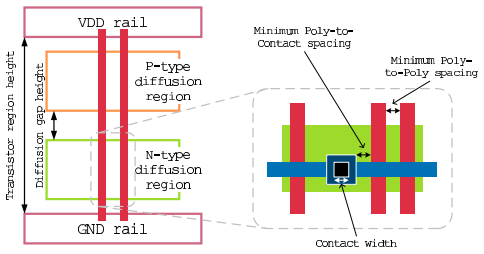

Figure 3: Layout model of gates [31].

4.2 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). Then, we use the gate area model described earlier to estimate the area of the entire block.

FIFO buffers. Designers typically implement buffers as SRAM arrays. Some on-chip networks, such as Raw microprocessor [25], use shift registers due to less demanding buffer space. We model the area of both implementations, but explain only the SRAM-based model here. We use the same SRAM-based FIFO buffer as in [29]. Equations (25) and (26) compute the word line and bit line lengths of the FIFO, respectively.

Table 4: Power (mW) breakdown comparison of ORION 2.0 with post-layout simulation results for the Intel 80-core chip.

Component	%diff (ORION 2.0 vs. Intel 80-core)
Buffer	-14.8
Crossbar	16.9
Arbiter	-9.0
Clock	-20.9
Link	8.8

$$L_{word-line} = F \cdot (w_{cell} + 2(P_r + P_w)d_w) \quad (25)$$

$$L_{bit-line} = B \cdot (h_{cell} + (P_r + P_w)d_w) \quad (26)$$

where F, B, w_{cell}, h_{cell}, d_w, P_r, and P_w are flit size 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 with flit size of F is calculated as follows. In this model, h_{cell} and w_{cell} are computed using the gate area model described earlier.

$$Area_{fifo} = L_{word-line} \cdot L_{bit-line} \quad (27)$$

For other router components, namely, crossbar and arbiter we first decompose them into their circuit building block (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.

4.3 Link Area

The area occupied by links are due to wires and repeaters. We use the described gate area model to estimate the area of repeaters. The area of global wiring can be calculated as

$$Area_{link} = F \times (w_w + s_w) + s_w \quad (28)$$

where $Area_{link}$ denotes the wire area, F is the flit size in bits, and w_w and s_s 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.

5 Validation and Significance Assessment

Table 7 lists all the architectural parameters used in ORION 1.0 and ORION 2.0 models. $N_{pipeline}$, App, and D are three new parameters added to ORION 2.0. We now show the validation of the our ORION 2.0 model against the Intel 80-core Teraflops chip [13] targeted for high-performance CMPs. The router in this design is a $5.1GHz$ 5-port switched router that uses two virtual channels (VC) with 16 buffers within each VC for dead-lock free routing. The flit width is 39 bits. The router uses a 5-stage pipeline with a round robin arbitration scheme. The supply voltage is at 1.2V with the transistor junction temperature at 110°C. The crossbar switch is implemented using the popular multiplexer-tree and the chip is taped out in 65nm technology. Our estimated total power consumption per router is within 7% of the Teraflops post-layout power estimation. Table 6 shows the difference between ORION 1.0 and ORION 2.0 with Intel 80-core Teraflops chip post-layout power and area numbers. We observe that ORION 2.0 has significantly improved the accuracy of the power and area models compared with ORION 1.0.[4] Table 4 gives a power breakdown comparison of the Intel 80-core Teraflops chip with ORION 2.0.[5] We attribute the deviation of ORION 2.0 estimates from the real chip simulations to: (1) difference in library files (i.e., between Intel's and those used in our study), (2) our use of 6T SRAM vs. Intel's use of 8T SRAM buffers, and (3) difference in arbitration scheme.[6]

We also validate our proposed models with the Intel Scalable Communications Core (SCC) chip [14] targeted for ultra-low power Systems-on Chip (SoCs). The router in this design has five ports with an arbitrated 5×5 crossbar and flip-flop-based buffers. The router is unpipelined, and thus has only 1 stage. There are two 1-flit buffers per input port with flit size of 32 bits. There are two input buffers per input port and one output buffer at the output port. The operating frequency is $250MHz$ and the supply

[4] We do not model power management mechanisms in ORION 2.0. Hence, for the Intel 80-core Teraflops chips we assume that all components are on.

[5] Currently we do not have a power breakdown of the Intel SCC chip.

[6] For arbiter power we use the modified separable VC allocator model.

Table 5: Comparison of network power (P), area (A), total number of routers, and hop count, using ORION 1.0 and ORION 2.0 models.

SoC		P (mW)		A (mm^2)		# of routers		max. # of router ports		Max. # of hops	
		ORION1.0	ORION2.0	ORION1.0	ORION2.0	ORION1.0	ORION2.0	ORION1.0	ORION2.0	ORION1.0	ORION2.0
VPROC	65nm	0.857	0.924	2.043	2.329	33	25	8	12	6	5
dVOPD	65nm	0.412	0.486	1.217	1.343	18	16	6	6	11	10

Table 6: Comparison of power (mW) and area (μm^2) models of ORION 1.0 and ORION 2.0 with post-layout simulation results of Intel 80-core and pre-layout simulation results of Intel SCC.

	Intel 80-core		Intel SCC	
	ORION1.0	ORION2.0	ORION1.0	ORION2.0
%diff (total power)	-85.3	-6.5	202.4	11.0
%diff (total area)	-80.9	-23.5	31.9	25.3

Table 7: Architectural parameters in ORION 1.0 and 2.0.

Parameter		Description
ORION 1.0	ORION 2.0	
B	B	No. of buffers
F	F	flit size
P	P	No. of ports
V	V	No. of virtual channels
X	X	No. of crossbar ports
$tech$	$tech$	technology node
f_{clk}	f_{clk}	clock frequency
V_{dd}	V_{dd}	supply voltage
-	$N_{pipeline}$	No. of pipeline stages
-	App	application domain (i.e., HP, LP, etc.)
-	D	chip dimension

voltage is 1.08V with temperature at 25°C. Our estimated power consumption per router is within 20% of the SCC's pre-layout simulation results. Table 6 shows the comparison results. We do not publish the absolute numbers as they are still company-confidential.

Finally, to assess the impact of improved power models on system-level design-space exploration, we integrate our models in COSI-OCC [23]. We use two representative SoC designs as test cases. The first design (VPROC) is a video processor with 42 cores and 128-bit data-width. The second design is based on a dual video object plane decoder (dVOPD), where two video streams are decoded in parallel by utilizing 26 cores and 128-bit data-width. Table 5 shows the comparison of the network power, area, total number of routers, and hop count when ORION 1.0 [29] and ORION 2.0 models are used. The clock frequency used is 2.25 GHz for 65nm technology node. We can observe that with ORION 2.0 models fewer routers with more ports are used. Since ORION 1.0 models were missing a number of important power components (i.e., clock power, link power, etc.) they tend to underestimate the power. Also, we observe that relative power due to an additional port (i.e., buffers and crossbar port) is not as high in ORION 2.0 vs. 1.0. Finally, more accurate ORION 2.0 models lead to a better-performing NoC (i.e., satisfying requirements at a lower hop count).

6 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 developed back in 2002 is inaccurate for current and future technologies and can lead to misleading design targets. We have proposed accurate power and area models for network routers (ORION 2.0) that are easily usable by system-level designers. We have presented a reproducible methodology for extracting inputs to our models from reliable sources. ORION 2.0 is in the form of a library of C files and shell scripts. By maintaining the user interfaces of the original ORION 1.0 while substantially improving its accuracy and fidelity, we see ORION 2.0 making a significant impact on future NoC research and design.

7 Acknowledgments

We would like to thank Dr. Sriram Vangal and Dr. Anthony Chun of Intel Corporation for providing us the Intel 80-core Teraflops and the Intel SCC power estimates, Dr. Hangsheng Wang of Freescale for his technical support of the original ORION code, and Tushar Krishna of Princeton University for assisting with validation. This research is supported by the MARCO Gigascale Systems Research Center.

References

[1] A. Banerjee, R. Mullins and S. Moore, "A Power and Energy Exploration of Network-on-Chip Architectures", *Proc. NoCs*, 2007, pp. 163-172.
[2] N. Banerjee, P. Vellanki and K. S. Chatha, "A Power and Performance Model for Network-on-Chip Architectures", *Proc. DATE*, 2004, pp. 1250-1255.
[3] A. Bona, V. Zaccaria, and R. Zafalon, "System Level Power Modeling and Simulation of High-End Industrial Network-on-Chip", *Proc. DATE*, 2004, pp. 318-323.
[4] S. Bhat, "Energy Models for Network-on-Chip Components", *M.S. Thesis*, Dept. of Mathematics and Computer Science, Royal Institute of Technology, Eindhoven, 2005.
[5] D. Brooks, V. Tiwari and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", *Proc. ISCA*, 2000, pp. 83-94.
[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*, 2008, pp. 258-264.
[7] X. Chen and L.-S. Peh, "Leakage Power Modeling and Optimization in Interconnect Networks", *Proc. ISLPED*, 2003, pp. 90-95.
[8] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *Proc. DAC*, 2001, pp. 684-689.
[9] D. E. Duarte, N. Vijaykrishnan and M. J. Irwin, "A Clock Power Model to Evaluate Impact of Architectural and Technology Optimization", *IEEE TVLSI* 10(6), 2002, pp. 844-855.
[10] N. Eisley and L.-S. Peh, "High-Level Power Analysis for On-Chip Networks," in *Proc. CASES*, 2004, pp. 104–115.
[11] S. Heo and K. Asanovic, "Power-Optimal Pipelining in Deep Submicron Technology", *Proc. ISLPED*, 2004, pp. 218-223.
[12] S. Heo and K. Asanovic, "Replacing Global Wires with an On-Chip Network: A Power Analysis", *Proc. ISLPED*, 2005, pp. 369-374.
[13] Y. Hoskote, S. Vangal, A. Singh, N. Borkar and S. Borkar, "A 5-GHz Mesh Interconnect for a Teraflops Processor", *IEEE MICRO*, 2007, pp. 51-61.
[14] D. A. Ilitzky, J. D. Hoffman, A. Chun and B. P. Esparza, "Architecture of the Scalable Communications Core's Network on Chip", *IEEE MICRO*, 2007, pp. 62-74.
[15] C. Isci and M. Martonosi, "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data", *Proc. MICRO*, 2003, pp. 93-104.
[16] P. Kongetira *et al.*, "Niagara: A 32-Way Multithreaded SPARC Processor," in *IEEE MICRO*, 25(2), 2005, pp.21-29.
[17] A. Kumar, P. Kundu, A. 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", *Proc. ICCD*, 2007, pp. 63-70.
[18] N. Muralimanohar, R. Balasubramonian and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0", *Proc. MICRO*, 2007, pp. 3-14.
[19] K. Niyogi and D. Marculescu, "System-Level Power and Performance Modeling of GALS Point-to-Point Communication Interfaces", *Proc. ISLPED*, 2005, pp. 381-386.
[20] C. S. Patel, S. M. Chai, S. Yalamanchili and D. E. Schimmel, "Power Constrained Design of Multiprocessor Interconnection Networks", *Proc. ICCD*, 1997, pp. 408-416.
[21] D. Pham *et al.*, "The Design and Implementation of a First-Generation Cell Processor", *Proc. ISSCC*, 2005, pp. 184-185.
[22] A. Pinto, A. Bonivento, A. Sangiovanni-Vincentelli, R. Passerone and M. Sgroi, "System Level Design Paradigms: Platform-Based Design and Communication Synthesis", *ACM TODAES*, 11(3), 2006, pp. 537-563.
[23] A. Pinto, L. P. Carloni, A. L. Sangiovanni-Vincentelli, "A Methodology and an Open Software Infrastructure for Constraint-Driven Synthesis of On-Chip Communications", *Technical Report*, UCB/EECS-2007-130, 2007.
[24] V. Soteriou, N. Eisley, H. Wang and L.-S. Peh, "Polaris: A System-Level Roadmap for On-Chip Interconnection Networks", *Proc. ICCD*, 2006, pp. 134-142.
[25] M. B. Taylor *et al.*, "The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs", *IEEE MICRO*, 22(2), 2002, pp. 25-35.
[26] M. B. Taylor *et al.*, "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams", *Proc. ISCA*, 2004, pp. 2-13.
[27] S. Thoziyoor, N. Muralimanohar, J. H. Ahn and N. P. Jouppi, "CACTI 5.1", *Technical Report HPL-2008-20*, HP Laboratories, 2008.
[28] H. Wang, "Power-Efficient Design of On-Chip Interconnection Networks", *Ph.D. Thesis*, Dept. of Electrical Engineering, Princeton Univeristy, 2005.
[29] H. Wang, X. Zhu, L.-S. Peh and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks", *Proc. MICRO*, 2002, pp. 294-395.
[30] W. Ye, N. Vijaykrishnan, M. Kandemir and M. J. Irwin "The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool," in *Proc. DAC*, 2000, pp. 340-345.
[31] H. Yoshida, D. Kaushik and V. Boppana, "Accurate Pre-Layout Estimation of Standard Cell Characteristics", *Proc. DAC*, 2004, pp. 208-211.
[32] http://www.synopsys.com/products/power/power_ds.html
[33] *Synopsys Design Compiler* User Guide, v2007.12.SP-5.
[34] http://www.cadence.com/us/pages/default.aspx
[35] http://www.synopsys.com/products/primetimepx/ .
[36] *ARM* Integrated Multiprocessor Core, 2006 http://www.arm.com/ .
[37] *Predictive Technology Model*, http://www.eas.asu.edu/~ptm/ .
[38] *Liberty File Format*, http://www.synopsys.com/products/libertyccs/libertyccs.html
[39] *LEF/DEF Exchange Format*, http://openeda.si2.org/projects/lefdef .
[40] *International Technology Roadmap for Semiconductors*, http://www.itrs.net