ORION 2.0: A Power-Area Simulator for Interconnection Networks

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1109/TVLSI.2010.2091686">http://dx.doi.org/10.1109/TVLSI.2010.2091686</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>Version</td>
<td>Author's final manuscript</td>
</tr>
<tr>
<td>Accessed</td>
<td>Sun Apr 24 10:45:33 EDT 2016</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/67492">http://hdl.handle.net/1721.1/67492</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike 3.0</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/3.0/">http://creativecommons.org/licenses/by-nc-sa/3.0/</a></td>
</tr>
</tbody>
</table>
ORION 2.0: A Power-Area Simulator for Interconnection Networks

Andrew B. Kahng, Fellow, IEEE, Bin Li, Member, IEEE, Li-Shiuan Peh, Member, IEEE, and Kambiz Samadi, Student Member, IEEE,

Abstract—As industry moves towards multi-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. 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 [18].

Index Terms—Network-on-chip, architectural-level modeling, design space exploration.

I. INTRODUCTION

Network power has become increasingly substantial in multi-core 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 [25] to early-stage architectural power models [4], [19], [18], [9]. 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 trade-offs 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 SoC. Bhat et al. [2] proposed an architecture-level regression analysis model for different router components based on energy numbers obtained from simulations using Magma [25] tools.

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 8X difference between ORION 1.0 estimations (per component) and silicon measurements. Also, the estimated total power is about 10X 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], 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. Figure 1 shows the usage model and modeling flow of ORION 2.0 with its main inputs and outputs. Contributions of ORION 2.0 beyond the original ORION 1.0 include:

• New: (1) We add flip-flop and clock dynamic and leakage power models. (2) We add link power models, leveraging accurate models recently developed in [6]. (3) 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. (4) We ad arbitrer leakage power model using the methodology proposed in [7]. (5) We add accurate area models for all the router building blocks. (6) We provide a semi-automated flow for extracting technology parameters from standard technology files, as well as extrapolatable models of process to allow ORION 2.0 to be easily and continuously updated in the future.

• Improved: Application-specific technology-level adjustments (use of different $V_{DD}$ 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.

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 digital circuits [8], representing up to 33% of power consumption in a high-performance router [10]. We estimate the term $C_{clk} = C_{pre} + C_{post} + C_{pipeline} + C_{wiring}$, where $C_{pre} = C_{fifo} + C_{pipe} + C_{reg}$, $C_{post} = C_{fifo} + C_{pipe} + C_{reg}$, and $C_{wiring}$ are capacitive loads due to SRAM-based FIFO buffers, flip-flop-based FIFO buffers, pipeline registers, and crossbar components, 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 an SRAM FIFO with flip width $fw$, the total capacitance due to precharging, with $P_r$ and $P_w$ being the number of read and write ports, can be estimated as $C_{pre} = (P_r + P_w) \times fw \times C_{chg}$, where $C_{chg}$ is precharging capacitance.

Flip-Flop-Based FIFOs. We assume 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 flip width of $fw$ bits, the capacitive load on the clock can be estimated as $C_{flip-flop} = fw \times B \times C_{ff}$.

Pipeline Registers. We assume DFF as the building block of the pipeline registers. In a router with flip width of $fw$ bits and $N_{pipeline}$ pipeline registers, the capacitive load on the clock due to pipeline registers is $N_{pipeline} \times C_{ff}$, where $N_{pipeline} = n_{port} \times fw$, i.e., (input and output) and crossbar components. $C_{pipeline} = 2 \times n_{vc} \times n_{vc}$, for VC allocator, and $N_{pipeline} = n_{vc} + n_{vc} + n_{vc}$, for switch allocator. $C_{ff}$ is the flip-flop capacitance and is extracted from 65nm $HP$ (high-performance) and LP (low-power) libraries. $n_{port}$ and $n_{vc}$ are number of ports and number of virtual channels, respectively.

Wire Load. We assume $n_{inter}$ $H$-tree clock distribution within each individual router block. If the router block dimension is $D$ (typically, tens of microns e.g., $D = 25\mu m$ in the router block of each tile in the Intel 80-core chip), the total wire capacitance of an L-level $H$-tree is $\sum_{i=0}^{D} \frac{2^{i} \times 1}{2^{i}} \times C_{int}$ where each term is (number of segments per level) $\times$ (fraction of $D$ per segment at that level) $\times$ (router dimension $D$) $\times$ (per unit length wire capacitance $C_{int}$). E.g., for a 5-level $H$-tree, the total wire capacitance is $(2^{2} + 2^{3} + 2^{4} + 2^{5} + 2^{6}) \times C_{int}$. In our studies, we use a fixed number of levels (equal to 5) in the $H$-tree, which 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 as the building block of the registers. Register-based FIFOs can be implemented as shift registers or as matrix of flip-flops (FFs).

2.1 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 (FF). We consider both read and write operations. The write operation occurs at the top-most level 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 $E_{write} = H(f_n, f_o) \times E_{ff}^{\text{switch}}$, where $E_{ff}^{\text{switch}}$ is the energy to switch one bit. We simply estimate the average switching activity as $H = \frac{1}{2}$; then, the average write energy is $E_{write} = \frac{1}{2} \times E_{ff}^{\text{switch}}$. 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 $E_{read} = (fw - 1) \times E_{ff}^{\text{switch}}$.

2.2 Matrix of FFs 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 each 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. 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, $E_{read} = E_{pointer}$, whereas write energy is due to pointer shifts and bit switches, $E_{write} = H \times E_{ff}^{\text{switch}} + E_{pointer}$, where $E_{pointer}$ 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 $n_{port} \times n_{vc}$ arbiters, each governing one specific output VC. Since $n_{port}$ 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, 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
Hence, the grant logic can be denoted as otherwise. Let \( req \) be the element in the \( i \)-th row and column \( j \) if requester \( R \) has the number of router ports equals the number of crossbar ports. The area of global wiring can be calculated as 

\[
\text{Area}_{\text{global}} = f_w \times (w_{\text{cell}} + s_{\text{cell}} + s_w),
\]

where \( f_w \) is 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 

\[
\text{Area}_{\text{buffer}} = f_w \times (w_{\text{cell}} + s_{\text{cell}} + s_w).
\]

The following subsections give several ‘anity’ checks for these models.

### A. Microarchitectural Parameters

In this subsection, 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 parameters include (1) buffers, (2) crossbar, (3) virtual channel allocator, (4) switch allocator, (5) clock, and (6) link. The microarchitectural parameters for each router are: (1) buffer size per VC per port, (2) buffer size in flits, (3) number of virtual channels, and (4) number of ports. For all the experiments, we use a supply voltage of 1.4V, switching activity of 0.5, and a clock frequency of 3GHz in 65nm technology. In this 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, 2, 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. This component 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.

As area is an important economic concern in IC (integrated circuit) 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.

### III. AREA MODELING

#### 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 technology-independent elements (i.e., gate-level netlist). For example, for SRAM-based FIFOs we can compute word line length 

\[
L_{\text{word-line}} = f_w \times (w_{\text{cell}} + s_{\text{cell}} + s_w) + f_d \times (w_{\text{cell}} + s_{\text{cell}} + s_w),
\]

where \( f_w \), \( f_d \), \( w_{\text{cell}} \), \( s_{\text{cell}} \), \( s_w \), \( P_{\text{cell}} \), and \( P_d \) 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 

\[
\text{Area}_{\text{buffer}} = f_w \times (w_{\text{cell}} + s_{\text{cell}} + s_w).
\]

For all the experiments, we use a supply voltage of 1.4V, switching activity of 0.5, and a clock frequency of 3GHz in 65nm technology. In this 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, 2, 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. This component also because the second stage devices are the primary contributors due to their large sizes.
On the other hand, as we increase the number of virtual channels, buffer dynamic power will not change since the number of flts 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 the number of ports, we expect buffer dynamic and leakage power to increase linearly. This is because addition of a new port will add a new buffer set, i.e., with the same buffer size and fit width.

Buffer area also follows power trends as expected. As buffer size increases, we expect buffer area to increase linearly. This is because we increase buffer size linearly. Similarly, 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. Hence, arbitration power increases linearly 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. In our experiments, we have assumed a two-stage separable VC allocator. For switch (SW) allocator, if we increase number of virtual channels, dynamic power and leakage power are expected to increase linearly because in SW allocator the request width of each arbiter increases linearly with respect to number of virtual channels.

Also, if we increase 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 number of ports, and also the number of such arbiters is proportional to the number of ports. Hence, VC allocator power consumption is dependent on the number of ports. Similarly, VC allocator area is expected to increase quadratically with number of virtual channels and number of ports. On the other hand, if we increase 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 number of ports, and in addition the total number of arbiters is proportional to the number of ports. Similarly, SW allocator area changes linearly and quadratically, respectively, with number of virtual channels and number of ports.

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 area) remains the same as we change any of the microarchitectural parameters for different components. We observe that leakage power density for buffer, crossbar, and arbiter is 0.0003 mW/\( \mu m \).

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), (2) nominal transistor size with nominal \( V_{th} \) (NVT), 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 increase in transistor width, and decrease of threshold voltage, as confirmed in Figure 2. For the experiment in Figure 2, we use a router with 5 ports, 2 virtual channels, buffer size of 4, and 32-bit flit width; for HVT, NVT, and LVT we use (0.8V,0.2GHz), (1.0V,1GHz), and (1.1V,3GHz), 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 Figures 3, 4 and 5.\(^5\) We use similar microarchitectural and transistor type values, but vary technology node from 90nm down to 32nm.

C. Router RTL Synthesis

In this subsection 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 [26]. We pick a baseline virtual channel (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 65nm GP 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, gate sizing, etc.) to satisfy the power, performance constraints. The detailed impacts of such transformations are difficult to capture at early stages of the design where not all implementation information is available. However, Figures 6-13 show that ORION 2.0 models’ trends (cf. previous subsections) match those of synthesized routers. In our comparisons, we use a supply voltage of 0.9V.

\(^4\)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. VC allocator dynamic power is linearly dependent on number of virtual channels (i.e., \( \frac{1}{n_{vc}} \times n_{vc} \times n_{vc} = n_{vc} \)).

\(^5\)In addition, we observe that the clock and link power and area models follow the expected trends.

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

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 tying it to substantially improving accuracy and tying it to ORION 2.0 making a significant impact on future NoC research and design.

VI. ACKNOWLEDGMENTS

The authors acknowledge the support of the Gigascale System Research Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

REFERENCES


TABLE I: Intel 80-core router configuration.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Frequency</th>
<th>Transistor type</th>
<th>number of ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2V</td>
<td>5.1GHz</td>
<td>LVT</td>
<td>5</td>
</tr>
<tr>
<td>number of VC</td>
<td>input buffer</td>
<td>output buffer</td>
<td>flit width</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>0</td>
<td>39</td>
</tr>
</tbody>
</table>

TABLE II: Intel SCC router configuration.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Frequency</th>
<th>Transistor type</th>
<th>number of ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8V</td>
<td>2.5GHz</td>
<td>HVT</td>
<td>5</td>
</tr>
<tr>
<td>number of VC</td>
<td>input buffer</td>
<td>output buffer</td>
<td>flit width</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>0</td>
<td>39</td>
</tr>
</tbody>
</table>

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