Explicit Modeling of Control and Data for Improved NoC Router Estimation

Andrew B. Kahng[†‡], Bill Lin[†] and Siddhartha Nath[‡]
UC San Diego ECE[†] and CSE[‡] Departments, La Jolla, CA 92093
abk@ucsd.edu, billlin@ece.ucsd.edu, sinath@cs.ucsd.edu

ABSTRACT

Networks-on-Chip (NoCs) are scalable fabrics for interconnection networks used in many-core architectures. ORION2.0 is a widely adopted NoC power and area estimation tool; however, its models for area, power and gate count can have large errors (up to 110% on average) versus actual implementation. In this work, we propose a new methodology that analyzes netlists of NoC routers that have been placed and routed by commercial tools, and then performs explicit modeling of control and data paths followed by regression analysis to create highly accurate gate count, area and power models for NoCs. When compared with actual implementations, our new models have average estimation errors of no more than 9.8% across microarchitecture and implementation parameters. We further describe modeling extensions that enable more detailed flit-level power estimation when integrated with simulation tools such as GARNET.

Categories and Subject Descriptors

C.2 [**Computer-Communications Networks**]: Network Architecture and Design

General Terms

Algorithms, Design, Performance

Keywords

network-on-chip, flit-level power modeling, parametric regression

1. INTRODUCTION

Networks-on-Chip (NoCs) have proven to be a highly scalable and low-latency interconnection fabric in the era of many-core architectures, as evidenced by in commercial chips such as the Intel 80-core [31], IBM Blue Gene [32] and Tilera TILE-Gx [33] processors. Because of their growing importance, NoC implementations must be optimized for latency and power [7, 9, 11, 15, 20]. To aid architects and designers in early design-space exploration, accurate NoC power and area estimators are required. Previous approaches to modeling are of two kinds, (1) based on templates at the architecture level, such as ORION2.0 [3], and (2) based on regression analysis on post-P&R data, such as [2]. ORION2.0 is widely used as a stand-alone tool as well as with full-system NoC simulators such as GARNET [13].

Both template- and regression-based modeling approaches, however, are in need of improvement. ORION2.0 has large estimation errors [2] for two fundamental reasons: (1) models are incomplete because control path resources are not modeled, even though they contribute significantly to power and area, and (2) models are not refined using post-P&R power and area data. Kahng et al. [2] and Jeong et al. [10] proposed non-parametric regression models to overcome the limitations in ORION2.0; [2] further concluded that parametric regression can be very inaccurate. In Figure 1(a), we show power estimation errors at 65*nm* in ORION2.0 and the previous regression approach [2], as a function of the number of virtual channels in the router. The maximum errors are 185% and 75%. Similarly, in Figure 1(b), we show power estimation errors at 90*nm* in ORION2.0 and the previous regression approach [2] when the flit-width is changed.

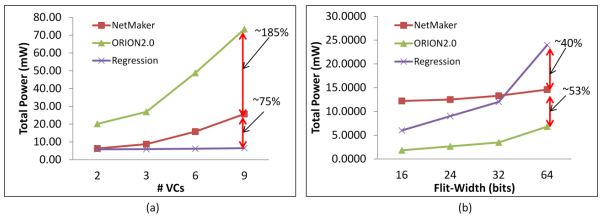

Figure 1: Poor estimations by ORION2.0 [3] and previous regression approach [2]. NetMaker vs. ORION2.0 vs. regression at (a) 65nm. (b) 90nm.

In this work, we propose a new model, ORION_NEW, which improves the ORION2.0 models by explicitly modeling control and data path resources. We perform parametric regression analysis with post-P&R area and power data to refine ORION_NEW models such that the estimates for area and power are highly accurate across multiple router RTLs, microarchitectures and implementation parameters. We demonstrate that accurate parametric models lead to better minimization of error in least-squares regression, and the worst-case errors are significantly better than the worst-case errors of non-parametric regression approaches [2]. We further describe modeling extensions that enable more detailed flit-level power estimation when integrated with simulation tools such as GARNET [13].

Our main contributions are as follows.

1. We explicitly model control and data paths to create ORION_NEW models that are highly accurate and robust across multiple router RTLs, and across microarchitecture and implementation parameters.
2. We demonstrate that parametric regression with accurate models can significantly reduce the worst-case error compared to non-parametric regression approaches for NoC routers.
3. We are the first to propose a detailed, efficient and fine-grained flit-level power estimation model that seamlessly integrates with full-system NoC simulators.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 describes the ORION_NEW model. Section 4 describes our modeling methodology. Section 4.3 presents our new flit-level power estimation model. Section 5 presents experimental results to validate and compare ORION_NEW models with ORION2.0 and the non-parametric regression approach in [2]. Section 6 concludes and outlines future work.

2. RELATED WORK

Previous works have focused primarily on two broad modeling paradigms: (1) architecture-level models using templates for each router component block (input and output buffer, crossbar, switch and VC arbiter) and (2) RTL and gate-level simulation-driven models. For the first approach, Patel et al. [4] propose a transistor count-based analytical model for NoC power. However, their models have large errors because they do not consider any router microarchitecture parameter. ORION [7] and ORION2.0 [3] are architectural models that use microarchitecture and technology parameters for the router component blocks. However, from our experimental studies as well as from [2], ORION2.0 estimates have very large errors.

The other approach is based on pre-layout (RTL or post-synthesis gate-level) [6, 8, 5, 16] or post-layout [12, 1, 17, 14] simulations.

DAC 2012, June 3-7, 2012, San Francisco, California, USA.

Banerjee et al. [1] report accurate power for a range of routers, but do not present any analytical models for router power. Chan et al. [8] develop cycle-accurate power models with reported average errors up to 20%. Meloni et al. [14] and Lee et al. [16] perform parametric regression analysis on post-layout and RTL simulation results, respectively. Their models, however, are fairly coarse-grained as they cannot explain how power dissipates in each router block with change in load, microarchitecture or implementation parameters. Kahng et al. [2] use non-parametric regression to model NoCs.

Our methodology uses accurate parametric models along with non-negative least-squares regression analysis to provide accurate area and power estimates, with average error of no more than 9.8% across microarchitecture and implementation parameters. Our models calculate area and power on a per-instance basis but avoid the overhead of slow gate-level simulations.

Furthermore, we significantly extend our models to achieve flit-level power estimations. Ye et al. [18] and Penolazzi et al. [17] estimate power dissipation using bit-level model, and Penolazzi et al. [17] propose a static bit-based model to estimate Nostrum NoC power. However, each of these models is tied to a specific router implementation and cannot explain how different bit encodings affect the power consumption in each block within the router. Our flit-level power estimation methodology estimates the power impact for each component block and reports accurate power numbers across different bit encodings in flits.

3. MODEL DESCRIPTION

We now describe the ORION_NEW modeling of each component in a modern on-chip network router. We have developed these models by analyzing post-synthesis and post-P&R netlists of two RTL generators, NetMaker [28] from Cambridge and the Open Source NoC router from Stanford [29]. (Our methodology is described in detail in Section 4.) Figure 2 shows the component blocks in a router, i.e., input buffer, switch and VC (virtual channel) arbiter, crossbar and output buffer [11]. We model instances (or gates) in each component block because our studies show that accurate estimations of area and power are possible only if the instance modeling is accurate. The microarchitecture parameters used are #Ports (P), #VCs (V), #Buffers (B) and Flit-width (F).

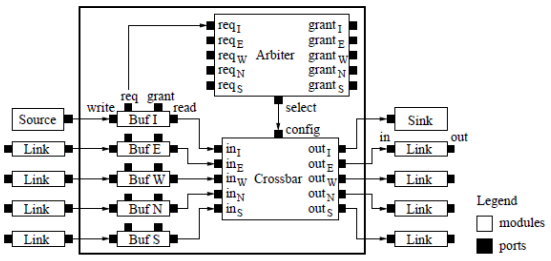

Figure 2: Router architecture [7].

3.1 New Model Elements

The new model explicitly accounts for control and data resources in the router. The new modeling elements are:

1. Control resources such as FIFO select and decode logic signals in the input and output buffers.
2. Tri-state crossbar model.
3. Additional input buffer resources for delay-optimized arbiters [11].
4. Output buffer model to store only head flits.
5. Clock frequency dependent scaling.

3.2 Crossbar (XBAR) Model

This component block is responsible for connecting input ports to output ports so that all flit bits are transferred to output ports [19]. The ORION2.0 models for router crossbar consider two implementations, matrix [11] and multiplexer tree [3]. The multiplexer tree is the smaller of these in terms of instance count and area and is modeled as $P \times P \times F$ multiplexers at each level of the tree. .

Modern router RTLs such as NetMaker and Stanford NoC use a simpler and smaller crossbar implementation where each flit bit is controlled using a tri-state buffer, which can be modeled as a $2:1$ MUX. Hence, the total number of such MUXes required are: $P \times P \times F$. This new model reduces the instance count by a factor of $[2^{log_2 P} - 1]$ when compared to the multiplexer tree implementation.

3.3 Switch and VC Arbiter (SWVC) Model

This block is responsible to generate control signals for the crossbar such that a connection is established between input buffers to output ports [19]. ORION2.0 adds an overhead of 30% to the arbiter by default. Our analysis indicates that this overhead is not needed with frequency ranges $400MHz$-$900MHz$ for process nodes $45nm$ to $130nm$. Beyond this range of frequency a derating factor must be applied, which is discussed in Section 3.7. The ORION_NEW model for switch and virtual channel arbiter is: $9 \times (P \times (P \times (V^2 + 1) + (V^2 - 1))$. The constant factor 9 arises because six 2-input NOR gates, two INVerters and one D-FlipFlop are used to generate one grant signal on each path.

3.4 Input Buffer (InBUF) Model

This block holds the entire incoming payload of flits at the input stage of the router for decode [19]. ORION2.0 models only the buffer instances and does not take into account control signals which are needed at this stage for decode such as FIFO select, buffer enable control signals and logic for housekeeping, such as the number of free buffers available per VC, VC identification tag per buffer, etc. As a result, ORION2.0 underestimates the instances at the input stage of the router.

In our new model, we model control signals and housekeeping logic in addition to the actual FIFO buffers. Modern routers implement the same stage VC and SW allocation to optimize delay [11], leading to doubling of input buffer resources. Hence, the number of FIFO buffers are $2 \times P \times V \times B \times F$. The control signals for decoding the housekeeping logic are modeled as: $180 \times P \times V + 2 \times P^2 \times V \times B + 3 \times P \times V \times B + 5 \times P^2 \times B + P^2 + F \times P + 15 \times P$ (as analyzed from the post-synthesis and post-P&R netlists). Each constant factor in the model denotes the number of instances per path. For example, the 180 factor accounts for instances to generate FIFO select signals and flags for each buffer in the $P \times V$ path. The smaller constant factors $2, 3, 5$ account for instances for local flags in the decode logic. The factor 15 denotes the number of buffers in each FIFO select path of an input port.

3.5 Output Buffer (OutBUF) Model

This block holds the head flits between the switch and the channel for a switch with output speedup [19]. ORION2.0 models the output buffers in exactly the same way as input buffers; this is inaccurate for modern routers that use hybrid output buffers, and leads to an overestimate of the instance count. The output buffers need to only store enough flits to match the speed between the switch and the channel. At the output, these buffers are used to stage the flits between the switch and channel when channel and switch speeds mismatch. Instead of $P \times V \times B \times F$ used in ORION2.0, output buffers are proportional to $P \times V$. There are several control signals per port and VC associated with each buffer, which makes the overall instance count grow as $P \times (80 \times V + 25)$. The constant factor 80 accounts for the instances used to generate flow control credit signals for each VC, while the constant factor 25 accounts for buffers and flags.

3.6 Clock and Control Logic (CLKCTRL) Model

ORION2.0 does not accurately model clock buffers and control logic routing resources as clock frequency scales. ORION_NEW models these resources as 2% of the sum of instances in the SWVC, InBUF and OutBUF component blocks.

3.7 Frequency Derating Model

As frequency changes, timing constraints change. To meet setup time at higher frequencies, buffers are inserted leading to an overall increase in instance count in the design. ORION2.0 scaling is agnostic to implementation parameters such as clock frequency. This causes large errors in area and instance counts at higher frequencies for component blocks such as SWVC, InBUF and OutBUF where there are several logic signals which consume routing resources. The

number of instances in the crossbar does not vary much with frequency because there are no critical paths. So, we can ignore the effects of frequency on the crossbar.

To derate for frequency, we find the frequency below which the instance counts change by less than 1%. In 65*nm* technology, this is 400MHz for both NetMaker and Stanford NoC routers. We derate instance counts based on this frequency as: $\Delta Instance = \Delta Frequency \times ConstantFactor$. The constant factor is dependent on the amount of control logic versus FIFO for each component block. In SWVC and InBUF, the $control/FIFO \approx 1$, so the constant factor value is 1. In OutBUF, $control/FIFO \approx 0.16$, and a fitted constant factor of 0.03 is used to account for setup buffers.

4. ORION_NEW METHODOLOGY

In this section, we describe how we estimate power and area using the two approaches described in Sections 4.1 and 4.2. We extend our methodology to flit-level power estimation in Section 4.3. We use:

- Multiple parametrized NoC RTL generators, NetMaker [28] from Cambridge University and the Open Source NoC from Stanford [29] to make the ORION_NEW models robust.
- Range of values of microarchitecture parameters, #Ports (P), #VCs (V), #Buffers (B) and Flit-width (F) and implementation parameters such as clock frequency and technology node.
- Operational parameters for power calculation: switching activity (TR) and static probability of 1's in the input (SP).
- Multiple commercial tools, Synopsys DesignCompiler (DC) [22] and Cadence RTL Compiler (RC) [21], with options to preserve module hierarchy after synthesis because we analyze each router component block. We compare instance counts, area and power reported by each tool to ensure that for a given RTL these results do not vary by more than 10%.
- Cadence SOC Encounter (SOCE) [21] with die utilization of 0.75 and die aspect ratio of 1.0 to place and route the synthesized router netlist.
- Synopsys PrimeTime-PX (PT-PX) [23] to run power analysis of the post-P&R netlist, SPEF [26] and SDC [27].
- MATLAB [30] function *lsqnonneg* for regression analysis.

Table 1 summarizes these details.

Table 1: ORION_NEW Methodology: Tools and Parameters

Stage	Tool	Options
RTL	NetMaker Stanford NoC	ISLAY config default
μarch	Ports; VCs; BUFs; Flit-Width	P = {5, 6, 8, 10}; V = {2, 3, 6, 9} B = {8, 10, 15, 22}; F = {16, 24, 32, 64}
Impl	Clock Freq Tech Nodes	*Freq* = {400, 700, 1200, 2000} MHz Switching Activity (TR) = {0.2, 0.4, 0.6, 0.8} Static Prob of 1's (SP) = {0, 0.25, 0.5, 0.75, 1.0} 45*nm* = OpenPDK45 from NCSU/OSU 65*nm*, 90*nm*, 130*nm* = TSMC GP, G, GHP resp.
Syn	Synopsys DC (v2009.06-SP2) Cadence RC (vEDI09.12)	*compile_ultra -exact_map* *-no_autoungroup -no_boundary_optimization* *report_area -hierarchy; report_power -hierarchy* default synthesis flow
Power	Synopsys PT-PX (v2009.06-SP2)	*set power_enable_analysis true* *set power_analysis_mode averaged* *set_switching_activity -toggle_count TR* *-static_probability SP -type inputs* *read_sdc router.sdc; read_parasitics router.spef*
Regression	MATLAB	*lsqnonneg*

Figure 3 shows the flow we use to develop ORION_NEW models for each component block of the router. In Table 2, we summarize the ORION_NEW instance count model of each component block.

Table 2: ORION_NEW model for Instances

Component	Equation
XBAR	P^2F
SWVC	$9(P^2V^2+P^2+PV-P)$
InBUF	$180PV+2PVBF+2P^2VB+3PVB+5P^2B+P^2+PF+15P$
OutBUF	$25P+80PV$
CLKCTRL	$0.02\times(SWVC+InBUF+OutBUF)$

There are two ways to estimate NoC area and power using the ORION_NEW models as shown in Figure 4. The manual approach is described in Section 4.1, and the regression analysis approach is described in Section 4.2. The benefits of each are described below.

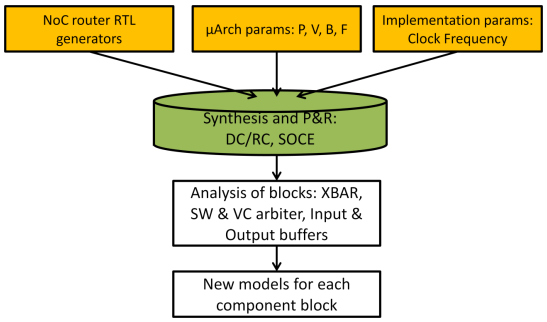

Figure 3: High-level flow used to arrive at ORION_NEW models.

- Both the approaches have minimum estimation error when the router RTLs are modular so that instance count and area numbers per component block can be calculated.
- The manual approach requires knowledge of process node and finer implementation details such as (HP, LSTP, LOP) × (HVT, NVT, LVT) × (bc, wc) to correctly select a technology library file. The regression analysis approach, on the other hand, is agnostic of implementation details. It only depends on a training set of data. More data points help the tool to minimize the sum of square error.
- The manual approach leads to faster estimation since it only involves technology library look-ups and plugging-in of library values into the ORION_NEW model. In contrast, the regression analysis approach requires synthesis and P&R to be performed on the router RTL for at least six data points. On an Intel Core $i3$ 2.4GHz processor, the runtime of the manual approach when used with ORION2.0 code is less than 10*ms*, whereas the regression analysis approach takes about 140*ms*, when 64 test data points are used.
- It is extremely difficult to capture fine-grained implementation details in ORION_NEW models, e.g., area and power contribution of wires after routing, and change in coupling capacitance and power after metal fill. These missing details cause estimation errors versus actual implementation when the manual approach is used. In order to reduce errors with respect to implementation, the regression analysis approach with post-P&R area and power is preferred.

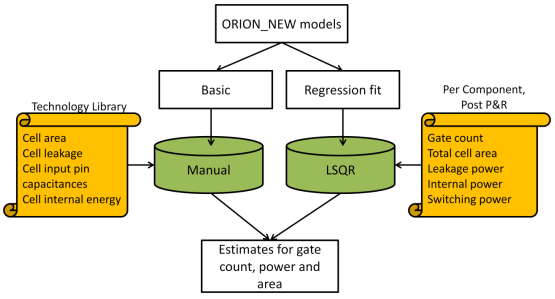

Figure 4: High-level view of power and area estimation methodology using Manual and Regression Analysis (LSQR) approaches.

4.1 Manual Approach to Estimate NoC Power and Area

This approach uses ORION_NEW models along with the technology library file of the process node in which the router is going to be fabricated. The key ingredients of this approach are:

- Microarchitecture parameters {P, V, B and F} and implementation parameter (clock frequency).
- Cell areas, leakage, internal energy and load capacitance.
- Switching activity.

ORION_NEW simplifies design of a NoC, using only a few standard cells. Instance count for each component block for a given set of router microarchitecture parameters is calculated from Table 2. Cell area is obtained from technology files. The area calculation, along with TSMC standard-cell names in parentheses, is shown in Table 3.

Table 3: Area Models using Instance count

Component	Logic (TSMC Cell Name)	Area
XBAR	MUX2 (MUX2D0)	$Area_{MUX} \times XBAR_{insts}$
SWVC	6 NOR2, 2 INV, 1 DFF (NR2D1, INVD1, DFQD1)	$\left(\frac{6Area_{NOR}+2Area_{INV}+Area_{DFF}}{9}\right) \times SWVC_{insts}$
InBUF + OutBUF	1 AOI, 1 DFF (AOI22D1, DFQD1)	$\left(\frac{Area_{AOI}+Area_{DFF}}{2}\right) \times (In+Out)BUF_{insts}$
CLKCTRL	1 INV, 1 AOI (INVD1, AOI22D1)	$\left(\frac{Area_{AOI}+Area_{INV}}{2}\right) \times (CLKCTRL)_{insts}$

Power has three components, that is, leakage, internal and switching. Leakage power is static power when the cell is not transitioning between logic states. It is dependent on current state of the input pins of the cell as well as process corner, voltage and temperature. Switching and internal power together constitute dynamic power, which varies with operating voltage, capacitive load and frequency of operation. Switching power is the power consumed when a load capacitance on a net is charged and discharged; internal power is the power dissipated inside a cell and consists of short-circuit power and switching power of internal nodes.

In ORION_NEW, toggle rate (*TR*) is equal to the input switching activity for all nets in the crossbar, arbiters and buffer control logic. We assume that buffer cells toggle at 25% of the input switching activity, since multiple VCs do not require buffer contents to change in every cycle.

Leakage power calculation: For leakage power, the model uses the weighted average of the state-dependent leakage of the cells. Equations (1)-(4) are used to calculate the leakage power of each component block.

$$P_{leak\_XBAR} = MUX_{leak} \times XBAR_{insts} \quad (1)$$

$$P_{leak\_SWVC} = \left(\frac{6NOR_{leak} + 2INV_{leak} + DFF_{leak}}{9}\right) \times SWVC_{insts} \quad (2)$$

$$P_{leak\_BUF} = \left(\frac{AOI_{leak} + DFF_{leak}}{2}\right) \times (In+Out)BUF_{insts} \quad (3)$$

$$P_{leak\_CLKCTRL} = \left(\frac{AOI_{leak} + INV_{leak}}{2}\right) \times (CLKCTRL)_{insts} \quad (4)$$

Internal power calculation: For internal power, table look-ups in technology library files return the internal energy of given standard cells with load capacitance of fanout pins and slew value of $\approx 5 \times FO4\ delay$.[1] Internal energy for a pin is the minimum of the rise and fall energies. Equations (5)-(8) are used to calculate internal power of each component block.

$$P_{int\_XBAR} = MUX_{int} \times TR \times XBAR_{insts} \quad (5)$$

$$P_{int\_SWVC} = (6NOR_{int} + 2INV_{int} + DFF_{int}) \times TR \times SWVC_{insts} \quad (6)$$

$$P_{int\_BUF} = (AOI_{int} + 0.25DFF_{int}) \times TR \times (In+Out)BUF_{insts} \quad (7)$$

$$P_{int\_CLKCTRL} = (AOI_{int} + INV_{int}) \times TR \times (CLKCTRL)_{insts} \quad (8)$$

Switching power calculation: For switching power, the load capacitance is calculated as the sum of the input capacitances of pins that are driven by a net and the wire capacitance on the net. The wire capacitance is approximately calculated as a constant factor times the total pin capacitances. This constant factor is 1.4 at 65*nm* and is assumed to decrease by 14% with for each successive process node shrink. Equations (9)-(12) are used to calculate switching power of each component block.

$$P_{sw\_XBAR} = XBAR_{load} \times TR \times XBAR_{insts} \quad (9)$$

$$P_{sw\_SWVC} = SWVC_{load} \times TR \times SWVC_{insts} \quad (10)$$

$$P_{sw\_BUF} = (In+Out)BUF_{load} \times TR \times (In+Out)BUF_{insts} \quad (11)$$

$$P_{sw\_CLKCTRL} = (CLKCTRL)_{load} \times TR \times (CLKCTRL)_{insts} \quad (12)$$

[1]The $FO4$ delay is the delay of a minimum-sized INV and is a standard proxy for switching speed in a given process technology. The resulting slew time values are 80 – 100ps for 45nm and 65nm technologies.

Flow details: The steps below describe how total area and power are estimated using the ORION_NEW models and equations above.

1. Choose microarchitecture parameters (P,V,B,F), clock frequency and average switching activity at inputs.
2. Use models in Table 2 to calculate the instance count of each component block of the router.
3. Use models in Table 3 to calculate the area of each router component block. Total area is calculated as the sum of areas of all blocks.
4. Obtain state-dependent leakage of cells from technology library files. Use Equations (1)-(4) to calculate leakage power of each component block. Total router leakage power is calculated as the sum of leakage power of all component blocks.
5. Obtain internal energy of cells from technology library files. Use Equations (5)-(8) to calculate internal power of each component block. Total internal power is calculated as the sum of internal power of all component blocks.
6. Obtain input pin capacitances of cells from technology library files. Use Equations (9)-(12) to calculate switching power of each component block. Total switching power is calculated as the sum of switching power of all component blocks.
7. The total power dissipated by the router is calculated as the sum of total leakage, total internal and total switching power.

4.2 Regression Analysis Approach to Estimate NoC Power and Area

As another approach to estimation of router area and power, we use parametric regression to fit parameters for cell area, leakage, internal energy and load capacitance into ORION_NEW models. This approach requires instance counts, area, and total leakage, internal and switching power of each component block of the router from post-P&R tools. Options are set in synthesis to preserve module hierarchy and names. Constrained least-squares regression (LSQR) is used to enforce non-negativity of coefficients (cell area, leakage, internal energy, load capacitance). We use the MATLAB [30] function *lsqnonneg* for this purpose, and tool options as given in Table 1.

Flow Details: LSQR is applied to fit a model of post-P&R instance count for each router component block. At least six data points are needed in the training set because there are four microarchitecture parameters and two implementation parameters (clock frequency and toggle rate). Our parametric LSQR setup is as follows.

$$a_1 \cdot Insts_{model\ <component>} + a_0 = Insts_{tool\ <component>} \quad (13)$$

$Insts^R_{model\ <component>}$ is the refined instance count of each component block after LSQR. The refined instance count is used to fit models of post-P&R area and power as follows:

$$b_1 \cdot Insts^R_{model\ <component>} + b_0 = Area_{tool\ <component>} \quad (14)$$

In Equation (14), b_1 is the fitting coefficient for cell area and the coefficient b_0 accounts for the routing overhead.

We model leakage, internal and switching power as:

$$\{c_5, d_5, e_5\} \cdot Insts^R_{model\ XBAR} + \{c_4, d_4, e_4\} \cdot Insts^R_{model\ SWVC} + \{c_3, d_3, e_3\} \cdot Insts^R_{model\ InBUF} + \{c_2, d_2, e_2\} \cdot Insts^R_{model\ OutBUF} + \{c_1, d_1, e_1\} \cdot Insts_{model\ CLKCTRL} = \{P_{leak\ tool}, P_{int\ tool}, P_{sw\ tool}\} \quad (15)$$

where coefficients $\{c_5, \cdots, c_0\}$ are used to fit cell leakage power, and similarly $\{d_5, \cdots, d_0\}$ and $\{e_5, \cdots, e_0\}$ are respectively used to fit internal energy and load capacitance.

It is possible to skip the instance count refinement step (Equation (14)) and directly perform LSQR for area and leakage, internal and switching power using the above equations. We observe that average error can change by 3% in either direction by omitting the instance count refinement step. Note that it is necessary to perform per-component LSQR; if LSQR is performed for the entire router's area or power, large errors result because multiple components have the same parametric combination of (P,V,B,F). Failing to separate these contributors to area or power results in large errors: at 65nm, we have experimentally observed worst-case errors of 296%

for power and 557% for area. Thus, it is important to preserve module hierarchy during synthesis in the flow.[2]

4.3 Extension to Flit-Level Power Modeling

The dynamic power models used in ORION2.0 and ORION_NEW do not consider bit encodings in a flit, which can lead to significant errors in dynamic power estimation. As an example, consider an 8-bit flit with four bits as 1. This flit can either be $8b'11110000$ or $8b'10101010$. In the first encoding, there is only one toggle per flit, whereas in the second encoding there are seven toggles per flit. Clearly, the second flit will lead to higher dynamic power than the first one. To model this effect, we devise a flow as shown in Figure 5. Before using a testbench, the netlists must pass an equivalence check using tools such as Synopsys Formality [24]. We inject different bit encodings in the input during simulation over 10000 cycles and the resultant VCD (Value Change Dump) is validated using a waveform analyzer such as Synopsys DVE [25]. A satisfactory VCD is used as input to Synopsys PrimeTime-PX [23] to obtain power values. Regression analysis is performed using the tool-reported power values with the ORION_NEW estimates to obtain an enhanced ORION_NEW model for flit-level power estimation. These models may be invoked by NoC full-system simulators such as GARNET [13] to obtain very accurate estimates.

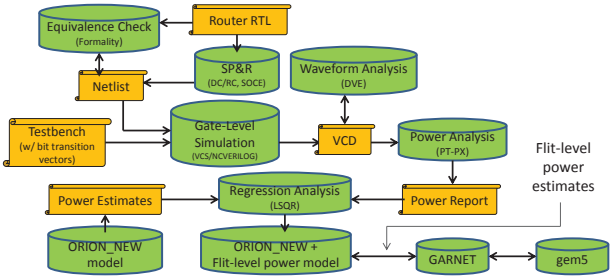

Figure 5: Methodology to enhance ORION_NEW dynamic power models with flit-level power estimation.

5. VALIDATION AND RESULTS

We set up experiments as described in Table 1 of Section 4. We use parameters and tools for our experiments as listed in Table 1. We discuss the results in two parts - (1) ORION2.0 versus ORION_NEW comparisons for area and power, and (2) impact of results with our regression analysis approach versus the approach used in prior work of [2]. We compare the results of our methodology with post-P&R instance count, power and area outcomes for two router RTL generators, Netmaker [28] and Stanford NoC [29].

5.1 ORION2.0 versus ORION_NEW Comparisons

Since the instance count per component is at the core of the ORION_NEW model, we compare ORION2.0 estimates of instance (or gate) counts, as well as the ORION_NEW model estimates with implementation (post-P&R) for each component block. Figures 6(a), 6(c) and 6(e) show the large errors in ORION2.0 in the crossbar, output buffer and input buffer respectively, and Figures 6(b), 6(d) and 6(f) show the significant reduction in estimation error for these components with ORION_NEW models. ORION2.0 and ORION_NEW are plotted in different graphs because of the large errors in instance counts in ORION2.0.

ORION2.0 modeling of instance count for a component does not consider implementation parameters such as clock frequency. As a result, the instance count does not scale when frequency is changed, even though at higher frequencies several buffers are inserted to meet tight setup time constraints. ORION_NEW models apply a frequency derating factor on the instance models for component blocks as described in Section 3.7. Figures 7(a) and 7(b) show the incorrect estimates by ORION2.0; by contrast, the estimates from

[2]Use of hierarchical synthesis in general leads to lower instance count, standard-cell area, and total power as compared with flat synthesis results. This comes at the cost of frequency (timing slack), since flat optimization across module boundaries can sometimes achieve better timing results. For our selection of microarchitecture and implementation parameters, hierarchical synthesis on average has 35% fewer instances, 48.8% less standard-cell area and 49.4% less total power – along with 8% less timing slack – compared with flat synthesis. The runtimes for hierarchical and flat synthesis are within 5% of each other.

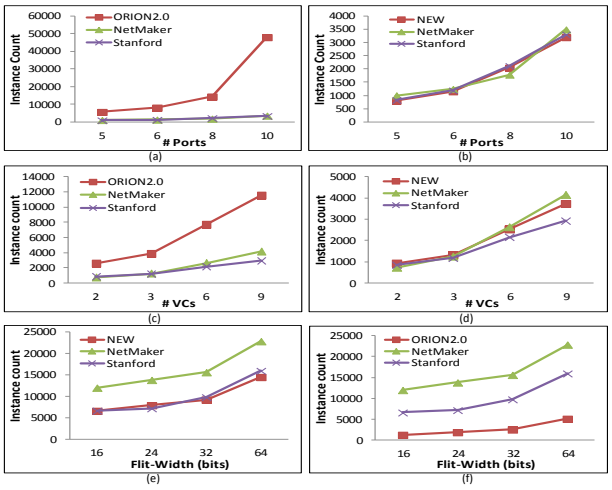

Figure 6: (a) XBAR with #Ports: ORION2.0 vs. Implementation. (b) XBAR with #Ports: ORION_NEW vs. Implementation. (c) Output Buffer with #VCs: ORION2.0 vs. Implementation. (d) Output Buffer with #VCs: ORION_NEW vs. Implementation. (e) Input Buffer with Flit-Width: ORION2.0 vs. Implementation. (f) Input Buffer with Flit-Width: ORION_NEW vs. Implementation.

ORION_NEW are very close to actual implementation for output and input buffer component blocks respectively.

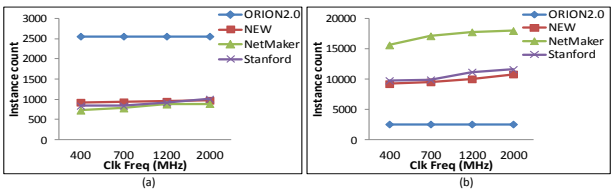

Figure 7: (a) Output buffer with Clock Frequency: ORION2.0 vs. ORION_NEW. (b) Input buffer with Clock Frequency: ORION2.0 vs. ORION_NEW.

Component	Avg Error: #Instances		Max Error: #Instances		Avg Error: Total Area		Max Error: Total Area	
	2.0	NEW	2.0	NEW	2.0	NEW	2.0	NEW
XBAR	86.10%	2.10%	93.10%	3.00%	86.20%	0.90%	93.20%	1.80%
SWVC	12.30%	12.30%	35.40%	35.40%	15.90%	20.80%	39.10%	66.80%
InBUF	270.70%	8.00%	417.30%	19.30%	134.40%	6.50%	199.40%	20.20%
OutBUF	69.00%	13.60%	80.60%	27.80%	74.70%	24.80%	86.40%	60.10%
Overall	109.50%	8.80%	156.60%	21.40%	77.80%	13.30%	104.50%	37.20%

Figure 8: Instance and Area error comparison of ORION2.0 vs. ORION_NEW. *Error% = ABS((TOOL - MODEL) / MODEL * 100).*

Table 8 summarizes the error in estimates of ORION2.0 and ORION_NEW when compared with NetMaker and Stanford NoC router post-P&R area. Higher values of error among the two models are highlighted in red. Figures 9(a) and 9(b) plot the estimation errors in power and area respectively at 45*nm* and 65*nm* technology nodes after applying the regression fitting approach described in Section 4.2. We see that ORION_NEW estimates are very close to implementation (average error of 9.8% in estimating NetMaker power at 45*nm*) and are robust across multiple microarchitecture and implementation parameters as well as router RTLs.

Next, we analyze the impact of flit-level power modeling as described in Section 4.3. To capture the effect of running simulations with input vectors having different bit encodings (shown in Figure 5), we use options in Synopsys PrimeTime-PX [23] to vary toggle rate and bit encodings in the input. We run simulations using four different toggle rates $(0.2, 0.4, 0.6, 0.8)$ and four different encodings of 1's in 32-bit input flits, and observe that leakage power is not dependent on bit encodings (changes by less than 2%). However, dynamic power varies by up to 30% (on average) depending on bit encodings in each flit. ORION2.0 models are incomplete because they consider only the flit arrival rates in the dynamic power estimation models. In Figure 10 we compare error in dynamic power estimations in ORION2.0, only ORION_NEW, and ORION_NEW with flit-level power models. We observe that by using flit-level power models, dynamic power estimations can be within 12% on average.

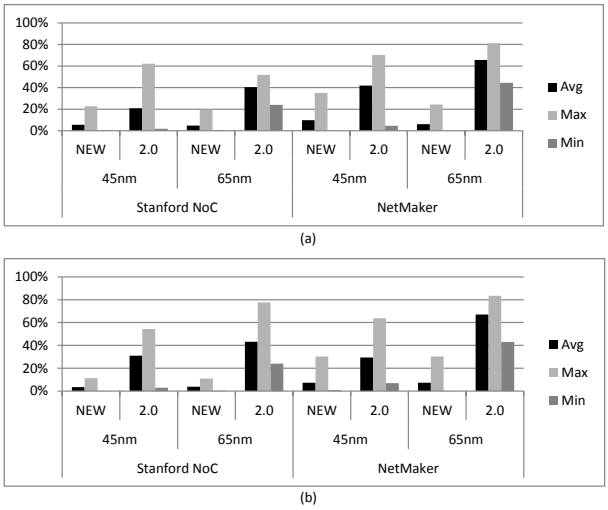

Figure 9: ORION_NEW with regression fit vs. ORION2.0: (a) Power estimation error. (b) Area estimation error.

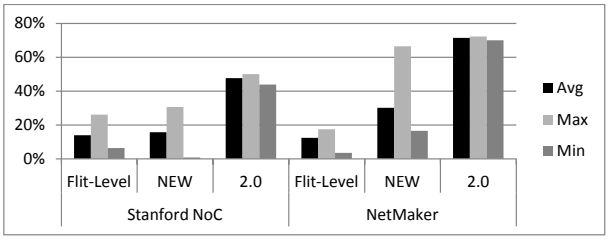

Figure 10: Comparison of dynamic power estimation error using (1) flit-level power model and ORION_NEW, (2) only ORION_NEW, and (3) ORION2.0.

We use these enhanced models (i.e., ORION_NEW with flit-level power models) in the full-system NoC simulator, GARNET [13]. We run simulations with synthetic uniform-random traffic for 10000 cycles and observe the difference in power estimates with the enhanced models and the default ORION2.0 models.

5.2 Impact of our regression analysis approach

In Section 4.2, we describe our parametric regression analysis approach using the ORION_NEW models. As seen from the results in Section 5.1, the ORION_NEW models are accurate across microarchitecture and implementation parameters because they explicitly model control and data path elements. With these accurate models, regression analysis can minimize errors and generate accurate fitting coefficients. The previous parametric regression approach [2] reports large errors because underlying ORION2.0 models do not model control path elements. The non-parametric regression approach of [2] using MARS (Multi-variate Adaptive Regression Splines) achieves reduced average power modeling errors of 5.82% at 65*nm* and 5.65% at 90*nm*, and reduced average area errors of 5.41% at 65*nm* and 5.01% at 90*nm*. In our work, we use parametric regression analysis but with accurate ORION_NEW models. Our average errors are similar to [2]; however, our maximum error for power (resp. area) is reduced by 58.89% (resp. 51%) at 65*nm*. At 90*nm* the reduction of maximum power (resp. area) error is 67.77% (resp. 53.38%). The reduction of maximum estimation error is significant because designers and architects of NoC care about worst-case accuracy.

6. CONCLUSIONS AND FUTURE WORK

Accurate modeling for NoC area and power estimation is critical to successful early design-space exploration in the era of many-core computing. ORION2.0, while very popular, has large errors versus actual implementation because it does not model control path resources. In this work, we propose ORION_NEW models that explicitly account for control and data path resources; we further refine the resulting area and power models by performing parametric regression analysis on post-P&R data. We are also the first to propose a detailed flit-level power estimation model that can seamlessly integrate with full-system NoC simulators such as GARNET. We validate robustness of our models across multiple router RTLs, and across microarchitecture and implementation parameters, and show that the ORION_NEW models are highly accurate with average error $\leq$ 9.8%. We also demonstrate that accurate models and parametric regression can reduce the worst-case estimation errors by more than 50% as compared to previous non-parametric regression models for NoC routers. We plan to extend our work to more accurately model link power by incorporating link signaling elements such as differential signaling, scrambling, serdes, equalization and 3D routing.

7. REFERENCES

[1] A. Banerjee, R. Mullins and S. Moore, "A power and energy exploration of network-on-chip architecture", *Proc. NOCS*, 2007, pp. 163-172.

[2] A. B. Kahng, B. Lin and K. Samadi, "Improved on-chip router analytical power and area modeling" *Proc. ASPDAC*, 2010, pp. 241-246.

[3] 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*, 2009, pp. 423-428.

[4] C. S. Patel, S. M. Chai, S. Yalamanchili and D. E. Schimmel, "Power constrained design of multiprocessor interconnection networks", *Proc. IEEE ICCD*, 1997, pp. 408-416.

[5] G. Guindani, C. Reinbrecht, T. Raupp, N. Calazans and F. G. Moraes, "NoC power estimation at the RTL abstraction level", *Proc. IEEE ASVLSI*, 2008, pp. 475-478.

[6] G. Palermo and C. Silvano, "PIRATE: A framework for power/performance exploration of network-on-chip architectures", *Proc. PATMOS*, 2004, pp. 521-531.

[7] H.-S. Wang, L.-S. Peh and S. Malik, "Orion: A power-performance simulator for interconnection networks", *Proc. MICRO*, 2002, pp. 294-305.

[8] J. Chan and S. Parameswaran, "NoCEE: Energy macro-model extraction methodology for network-on-chip routers", *Proc. IEEE ICCAD*, 2005, pp. 254-259.

[9] K. Chang, J. Shen and T. Chen, "A low-power crossroad switch architecture and its core placement for network-on-chip", *Proc. DATE*, 2005, pp. 375-380.

[10] K. Jeong, A. B. Kahng, B. Lin and K. Samadi, "Accurate machine learning-based on-chip router modeling", *IEEE ESL 2(3)*, 2010, pp. 62-66.

[11] L.-S. Peh, "Flow control and micro-architectural mechanisms for extending the performance of interconnection networks" *PhD Thesis*, Stanford University, 2001.

[12] N. Banerjee, P. Vellanki and K. S. Chatha, "A power and performance model for network-on-chip architectures", *Proc. DATE*, 2004, pp. 1250-1255.

[13] N. Agarwal, T. Krishna, L.-S. Peh and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator", *Proc. IEEE ISPASS*, 2009, pp. 33-42.

[14] P. Meloni, I. Loi, F. Angiolini, S. Carta, M. Barbaro, L. Raffo and L. Benini, "Area and power modeling for network-on-chip with layout awareness", *Proc. IEEE VLSI Design*, 2007, pp. 1-12.

[15] R. Mullins, A. West and S. Moore, "The design and implementation of a low-latency on-chip network", *Proc. ASPDAC*, 2006, pp. 164-169.

[16] S. E. Lee and N. Bagherzadeh, "A high level power model for network-on-chip (NoC) router", *Integration, the VLSI journal* 35(6), 2009, pp. 1-7.

[17] S. Penolazzi and A. Jantsch, "A high level power model for the Nostrum NoC", *Proc. Digital System Design*, 2006, pp. 673-676.

[18] T. T. Ye, G. de Micheli and L. Benini, "Analysis of power consumption on switch fabrics in network routers", *Proc. DAC*, 2002, pp. 524-529.

[19] W. J. Dally and B. Towles, *Principles and practices of interconnection networks*, Morgan Kaufmann, 2004.

[20] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnection networks", *Proc. IEEE ISLPED*, 2003, pp. 90-95.

[21] Cadence Encounter RTL Compiler User Guide. *http://www.cadence.com/products/ld/rtl_compiler/pages/default.aspx*

[22] Synopsys Design Compiler User Guide. *http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DCUltra/pages/default.aspx*

[23] Synopsys PrimeTime User Guide. *http://www.synopsys.com/Tools/Implementation/SignOff/PrimeTime/pages/default.aspx*

[24] Synopsys Formality User Guide. *http://www.synopsys.com/tools/verification/formalequivalence/pages/formality.aspx*

[25] Synopsys VCS and DVE User Guide. *http://www.synopsys.com/tools/verification/functionalverification/pages/vcs.aspx*

[26] Standard Parasitic Exchange Format. *http://www.edaboard.com/thread37705.html*

[27] SDC User's Guide. *http://www.actel.com/documents/SDC_AN.pdf*

[28] Netmaker. *http://www-dyn.cl.cam.ac.uk/~rdm34/wiki*

[29] Stanford NoC. *https://nocs.stanford.edu/cgi-bin/trac.cgi*

[30] MATLAB. *http://www.mathworks.com/*

[31] Intel 80-core Report. *http://techresearch.intel.com/ProjectDetails.aspx?Id=151*

[32] IBM Blue Gene processor. *http://www.research.ibm.com/journal/rd49-23.html*

[33] Tilera TILE-Gx processor. *http://www.tilera.com/products*