Accurate Machine-Learning-Based On-Chip Router Modeling

Kwangok Jeong, *Student Member, IEEE*, Andrew B. Kahng, *Fellow, IEEE*, Bill Lin, *Senior Member, IEEE*, and Kambiz Samadi, *Student Member, IEEE*

***Abstract*—As industry moves towards multicore chips, networks-on-chip (NoCs) are emerging as the scalable fabric for interconnecting the cores. With power now the first-order design constraint, early-stage estimation of NoC power, performance, and area has become crucially important. In this work, we develop accurate architecture-level on-chip router cost models using machine-learning-based regression techniques. Compared against existing models (e.g., ORION 2.0 and parametric models), our models reduce estimation error by up to 89% on average.**

***Index Terms*—Machine learning, nonparemetric regression, on-chip networks.**

I. Introduction

NETWORKS-ON-CHIPS (NoCs) are emerging as the *de facto* interconnection fabric of choice for both general-purpose chip multiprocessors (CMPs) [10], [15] and application-specific multiprocessor systems-on-chip (MPSoCs) [5], [13]. With increasing core counts, there is a corresponding increase in communication demands in multicore designs to facilitate high core utilization and the corresponding critical need for high-performance NoCs. With many opposing parameters, effective early-stage design exploration is essential to enable the realization of achievable power-delay-area tradeoffs. Existing architectural estimation methods in the literature can be broadly classified as *template-based* approaches: in one way or another, these existing methods assume a specific architecture and underlying circuit implementation in their modeling efforts. The template-based estimation approach is exemplified by the widely-used early-stage NoC power estimation tool ORION [16], and its enhanced successor ORION 2.0 [8].

The template-based approach has two fundamental limitations. First, for the power and area estimations to be accurate, the actual router microarchitecture used for implementation must closely match the microarchitecture assumed. The second limitation of template-based estimation is the difficulty of capturing effects of different application-specific integrated circuit (ASIC) implementation flows and flow options.

Other template-based approaches are based on parametric models [1], [2], [4], [12]. These approaches also assume a specific underlying router microarchitecture, requiring the development of new models for different microarchitectures. For parametric models the modeler needs full comprehension of the underlying router microarchitecture in order to come up with a relevant parametric model. Moreover, it is difficult to capture configuration parameter interactions that may gain significance as the design complexity increases. Finally, most existing parametric modeling approaches fail to consider implementation flow parameters in their models.

To quantify the above limitations, Kahng *et al.* [9] evaluate the accuracy of ORION 2.0 and parametric models against post-layout power results for different router configurations created using Netmaker [20], a public-domain tool that generates fully-synthesizable RTL for parameterized input-buffered VC routers. Even when provided with the actual (TSMC 65 nm) library information, ORION 2.0 and parametric models have significant deviation (40% and 28% on average) from actual power values, respectively. The fact that ORION 2.0 underlying architecture does not completely match the architecture assumed in [9] confirms the point that the accuracy of the template-based models degrades as the underlying architecture or circuit implementation changes. Such inaccuracy in estimation can lead to erroneous NoC design choices.

In this letter, we propose a new modeling paradigm that uses *machine-learning-based nonparametric* regression to develop accurate architecture-level on-chip router power, performance, and area models. The contributions of our work are as follows.

- We propose a *framework* for modeling on-chip router power, performance, and area using machine-learning-based nonparametric regression methods. Our approach aids the development of accurate architecture- and implementation-specific power, performance, and area models.
- Our results show that *nonparametric* regression techniques can capture the impact of both underlying architectural and implementation parameters on on-chip router power, performance and area. In addition, we show that nonparametric techniques can capture the impact of design optimization techniques (e.g., use of multi-V_{th}) on design power, maximum frequency, etc.
- We introduce a reproducible flow to further aid *automatic* generation of accurate architectural estimation models.
- Finally, to accompany this letter, we are releasing our models in the form of C++ functions and scripts to enable further NoC research and design [22].

Manuscript received March 09, 2010; accepted April 16, 2010. Date of publication May 27, 2010; date of current version September 17, 2010. This letter draws on preliminary accounts published at ISOCC-09 [7] and ASPDAC-10 [9]. This manuscript was recommended for publication by S. K. Shukla.

K. Jeong, B. Lin, and K. Samadi are with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0409 USA (e-mail: kjeong@vlsicad.ucsd.edu, billlin@ece.ucsd.edu; ksamadi@ucsd.edu).

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

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2010.2051413

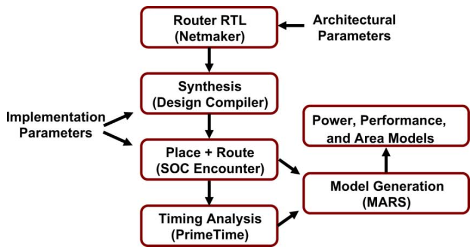
Router RTL (Netmaker)
Architectural Parameters
Synthesis (Design Compiler)
Implementation Parameters
Place + Route (SOC Encounter)
Timing Analysis (PrimeTime)
Power, Performance, and Area Models
Model Generation (MARS)

Fig. 1. Implementation flow.

II. Implementation Flow and Scope of Study

A. Implementation Flow and Tools

Fig. 1 shows our physical implementation flow, which includes the traditional synthesis, placement and routing (SP&R) steps plus static timing analysis (STA) and model generation, scripted for "push-button" use. At each step we require that the design meets timing requirements before it can pass on to the next step.

In our flow, we first synthesize corresponding RTL codes of the on-chip virtual channel (VC) router with worst-case timing libraries. To mimic a typical industrial timing-driven flow, we impose the target frequency as the primary constraint, with area and power minimization being optimization objectives.

Using synthesized netlists, we implement the designs through place and route (P&R) steps using different row utilization and aspect ratio values, at the floorplan stage. In addition, we use multi-V_{th} flow in which cells are chosen from a selection of high V_{th} (HVT), normal V_{th} (NVT), and low V_{th} (LVT) to obtain a larger leakage vs. frequency tradeoff envelope. After routing, we obtain power and area values used in model generation. In addition, we perform static timing analysis to obtain the longest (critical) path delay of each design, used in performance model generation.

We use *Netmaker v0.82* to generate a library of synthesizable on-chip router RTL netlists. We perform our experiments using multi-V_{th} libraries in TSMC 65 GP technology. We use *Synopsys Design Compiler v2009.06-SP2* [18] to synthesize the RTL netlists, *Cadence SOC Encounter v7.1* [22] to execute the P&R flow, and *Synopsys PrimeTime v2007.12-SP3* [21] for static timing analysis. Finally, *MARS3.0* [19] is used to implement nonparametric modeling techniques.

B. Scope of Study

We focus on the microarchitectural and implementation parameters that are of interest at the system level, and that significantly affect the quality of results. In this letter, we use a baseline virtual channel (VC) router in which VC allocation and switch allocation are performed sequentially in one clock cycle. In a VC router the microarchitectural parameters are: 1) flit-width, fw; 2) number of virtual channels, n_{vc}; 3) number of input and output ports, n_{port}; and 4) buffer size, l_{buf}. Table I shows the microarchitectural parameters and the values they take on in our study.

Our implementation parameters include 1) clock frequency, f_{clk}; 2) aspect ratio, ar; 3) row utilization, $util$; and 4) type of V_{th} flavors (V_{th}). Target clock frequencies for the router design are 200, 400, and 700 MHz. We use three different aspect ratio values, 1.0, 2.25, and 3.75, and three row utilizations, 50%, 75%, and 90%. We also use single-V_{th} and triple-V_{th} library flavors.[1] Depending on the specific design, there could be other relevant architectural parameters that need to be considered. In addition, depending on the specific design constraints there could be extra implementation optimization steps required, e.g., clock gating, power gating, transistor stacking, etc., which then have to be included in the model development for enhanced accuracy. However, the parameters that we have considered in our model development represent the main parameters that are also easily available at the early stages of the design [9].

TABLE I
List of Microarchitectural Parameters Used in Our Studies

Parameter	Values
fw	{16, 24, 32}-bits
n_{vc}	{2, 3, 5}
n_{port}	{3, 5, 7}
l_{buf}	{3, 5, 8}-flit buffers

III. Modeling Methodology

We develop architecture-level on-chip router power, performance, and area models using a new paradigm in which we use machine-learning-based techniques for model generation process. A baseline model generation flow is as follows.

- We begin with a parameterized synthesizable RTL specification for a given router microarchitecture, i.e., a *configurable* router microarchitecture specification, which we use to generate the actual router implementations under different configuration parameters.
- Using a small subset of selected configurations for training, we run through each configuration in this training set through a target ASIC implementation flow, including timing-driven RTL synthesis and timing-driven placement and routing, to obtain a detailed physical layout for each router instance.
- Finally, we apply machine-learning-based nonparametric regression techniques to a training set of power, performance, and area data to derive the corresponding estimation models.

A. Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) is a machine-learning-based regression technique based on multivariate smoothing splines. The general MARS model can be represented as [17]

$$\hat{y} = c_0 + \sum_{i=1}^{I} c_i \prod_{j=1}^{J} b_{ij}(x_{ij}) \tag{1}$$

where $\hat{y}$ is the target variable (e.g., on-chip router power, performance, and area), c_0 is a constant, c_i are fitting coefficients, and $b_{ij}(x_{ij})$ is the truncated basis function[2] with x_{ij} being the microarchitectural/implementation parameter used in the i^{th} term of the j^{th} product. I is the number of basis functions and J limits the order of interactions.

[1]Our models are specific to the 65-nm technology node. To develop models for other technology nodes, we would need to run through our flow with the corresponding library files.

[2]Each basis function can be a constant, a hinge function that is of form $\max(0, c - x)$ or $\max(0, x - c)$, or a product of two or more hinge functions.

Power Model
b_1 = max(0, f_{clk} – 200); b_2 = max(0, n_{port} – 3)×b_1; $\cdots$ b_{34} = max(0, V_{th} – 1)×b_4; b_{35} = max(0, f_{clk} – 200)×b_{17}; $p = 5.021 + 0.009 \times b_1 + 0.003 \times b_2 + 0.002 \times b_3 + \cdots + 0.061 \times b_{33} - 9.946e-5 \times b_{34} - 0.001 \times b_{35}$
Performance Model
b_1 = max(0, f_{clk} – 400); b_2 = max(0, 400 - f_{clk});$\cdots$ b_{34} = max(0, V_{th} – 1)×b_{33}; b_{35} = max(0, V_{th} – 1)×b_{32}; $f_{max} = 1/(3.778 + 0.004 \times b_1 + 0.003 \times b_2 + \cdots + 4.695e-5 \times b_{33} - 4.965e-5 \times b_{34} + 0.0001 \times b_{35})$
Area Model
b_1 = max(0, n_{port} – 5); b_2 = max(0, 5 - n_{port});$\cdots$ b_{33} = max(0, f_{clk} – 400)×b_6; b_{35} = max(0, V_{th} – 1)×b_{33}; $a = 0.020 + 0.008 \times b_1 + 0.005 \times b_2 + 0.010 \times b_3 + \cdots + 9.115e-7 \times b_{31} - 3.226e-6 \times b_{33} - 2.130e-6 \times b_{35}$

Fig. 2. Sample power, performance, and area models of a 65-nm router.

The optimal MARS model is built in two passes. 1) Forward pass: MARS starts with just an intercept, and then repeatedly adds basis functions in pairs to the model. The total number of basis functions is an input to the modeling process. 2) Backward pass: during the forward pass, MARS usually builds an overfit model; to build a model with better generalization ability, the backward pass prunes the model using a generalized cross-validation (GCV) scheme

$$GCV(K) = \frac{1}{n} \frac{\sum_{k=1}^{n} (y_k - \hat{y})^2}{\left[1 - \frac{C(M)}{n}\right]^2} \tag{2}$$

where n is the number of observations in the data set, K is the number of nonconstant terms, and $C(M)$ is a complexity penalty function to avoid overfitting.

B. On-Chip Router Cost Models

We model both dynamic and leakage power components. Dynamic power is due to charging and discharging of switching capacitances c_{sw}, and leakage power is due to subthreshold and gate leakage currents i_{leak}. Thus, our goal is to model dependence of switching capacitance and leakage current on microarchitectural and implementation parameters. To model performance we define maximum *implemented* clock frequency to be the reciprocal of the maximum delay obtained for a given combination of implementation and microarchitectural parameters. Maximum implemented clock frequency primarily depends on the given cycle time constraint, however, it is also affected by the V_{th} flavors of the library used. Similarly, we model the dependence of the sum of standard cell areas on microarchitectural and implementation parameters. Fig. 2 illustrates the form of resulting router power, performance, and area models for our target router in 65 nm.

IV. Experimental Results and Discussion

A. Model Validation

To generate our models, we randomly select 10% of our entire data set as training data; we then test the models on the other 90% of the data. To show that the selection of the training set does not substantially affect model accuracy, we randomly select 10% of the entire data set five times and show the corresponding models' maximum and average error values (Table II). Furthermore, to assess the generality of our models, we validate the models against 72 data points outside the scope of our study (cf. Section II). We observe that our power and area models are within 7.9% (48.4%), i.e., on average (maximum), of the layout data.

TABLE II
Model Stability Versus Random Selection of the Training Set

Experiments	power % diff		performance % diff	
	max	avg	max	avg
Exp 1	42.208	3.251	39.113	3.384
Exp 2	44.732	3.369	42.503	3.893
Exp 3	36.769	3.571	42.927	3.279
Exp 4	39.782	3.665	37.446	3.472
Exp 5	40.092	3.315	43.011	3.523

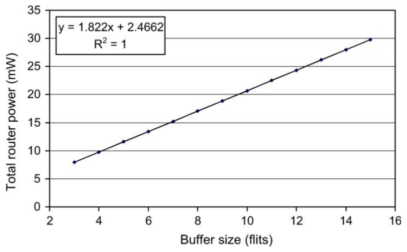

Fig. 3. Total router power versus buffer size.

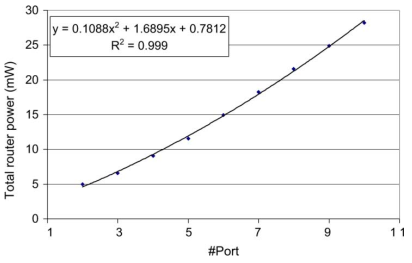

Fig. 4. Total router power versus number of ports.

We also investigate the impact of different microarchitectural and implementation parameters on router power, performance, and area. For the experiments, we use a supply voltage of 0.9 V, switching activity of 0.2, clock frequency of 400 MHz and a single-V_{th} 65 nm timing library. In each experiment, we only vary one microarchitectural parameter of interest and keep the others fixed. Nominal values for buffer size, flit-width, number of virtual channels, and number of ports are 5 flits, 32 bits, 1 virtual channel (i.e., wormhole configuration), and 5, respectively.

Figs. 3 and 4 give 'sanity checks' for our models.[3] Router power is affected by buffer size. When we vary buffer size, we expect router power to increase linearly, as confirmed in Fig. 3. This is because buffer size linearly increases the number of registers required for the additional flits. Another important microarchitectural parameter is the number of router ports. If we increase number of ports, we expect router power to increase quadratically as confirmed in Fig. 4. This is because the baseline VC router uses a multiplexer tree crossbar which enables arbitrary one-to-one connections between N input ports and N output ports [8].

The takeaway from Figs. 3 and 4 is that *automatic* discovery of accurate, and physically and architecturally meaningful, models is possible using nonparametric regression.

In addition, we compare our proposed models against 1) parametric and 2) ORION 2.0 models. To develop models using

[3]Our models similarly follow the expected trend with respect to the flit-width and the number of virtual channels.

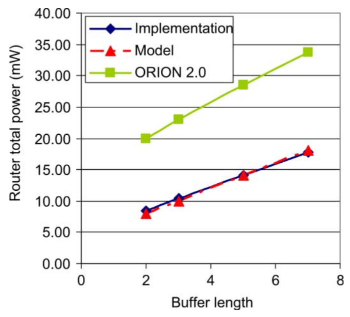

Fig. 5. Router total power versus buffer length.

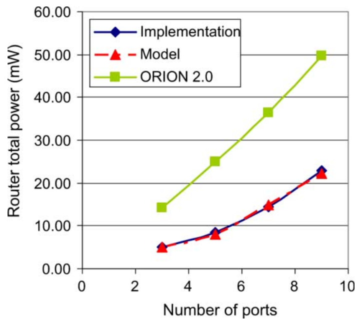

Fig. 6. Router total power versus number of ports.

parametric regression, we use the same data set as used in our model generation process. The drawback of the parametric regression methods (i.e., linear, quadratic regression, etc.) is their limited accuracy since there are no procedures in the modeling methodology to help the modeler understand the interactions between the individual parameters. Hence, to develop parametric models comprehensive understanding of the underlying architecture and circuit implementation is needed. We also compare our models with ORION 2.0 using similar microarchitectural parameters. Figs. 5 and 6 show the significant accuracy improvement of our new model relative to ORION 2.0, when compared against router implementation data. We believe that a key limitation of ORION 2.0 is that it does not capture the effects of the implementation flow. 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 in static circuit templates (e.g., ORION 2.0) as they depend on such implementation parameters as process and library flavor, operating voltage, etc. Our results show that our all models reduce the estimation error by up to 86% and 89% (on average) versus parametric and ORION 2.0 models, respectively [9].

Table III shows relative variable importance in our maximum implemented clock frequency model. *MARS3.0* computes the variable importance based on the reduction in goodness of fit when the variable is removed. The right column in Table III shows the relative importance of variables in terms of percentage of the highest loss in GCV, that is highest reduction of goodness of fit among all variables [3].[4] From Table III, we observe that given clock frequency f_{clk} is the main contributor

[4]Note that the relative variable importance depends on the existing set of implementation and microarchitectural parameters. If we remove any of the parameters, the relative variable importance and the associated rankings could change.

TABLE III
RELATIVE VARIABLE IMPORTANCE FOR MAXIMUM IMPLEMENTED CLOCK FREQUENCY MODELING

Parameter	Variable Importance (%)
f_{clk}	100
V_{th}	38.85
n_{port}	22.78
n_{vc}	5.26
l_{buf}	4.95
$util$	2.09
ar	0.20

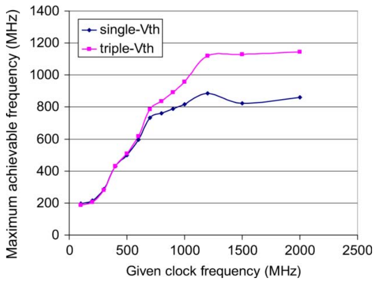

Fig. 7. Maximum implemented clock frequency versus given clock frequency.

to the maximum implemented frequency. Among the microarchitectural parameters, flit-width does not affect maximum implemented frequency because it only changes the bandwidth (i.e., simultaneous number of bits processed by the router). Buffer size and number of virtual channels do not noticeably affect the maximum implemented clock frequency because buffer size only determines the amount of flit storage, and virtual channels only provide parallel, multiplexed-paths to the crossbar switch. Hence, they do not change the critical path and have no noticeable impact on maximum delay. However, number of ports affects the maximum implemented frequency because it changes the crossbar switch interconnect grid, i.e., more ports result in a longer signal path within the crossbar switch.

B. *Significance Assessment*

In this section, we highlight the benefits of our proposed machine-learning-based modeling methodology. First, our methodology considers the interactions between different architectural and implementation parameters and accurately captures their combined effect on the power and performance of the implemented on-chip routers. Our results show a close match (3.9% error on average) between our model estimations and layout data.

Second, our methodology captures the impact of design optimization techniques. For example, it is well known that use of multi-V_{th} libraries can reduce leakage power or increase performance when HVT and LVT libraries are used, respectively. However, it is difficult to capture the impact of optimization techniques on design power and performance. As an example, our methodology enables designers to predict the impact of using triple-V_{th} libraries on design power and performance. Fig. 7 shows the maximum implemented frequency versus the given clock frequency for single-V_{th} and triple-V_{th}. Fig. 8 shows router leakage power versus clock frequency for the above two libraries. From the figures, we estimate how much performance can be improved or leakage power can be reduced when we use triple-V_{th} libraries. Despite up to 60% difference between single-V_{th} and triple-V_{th} leakage power values, given

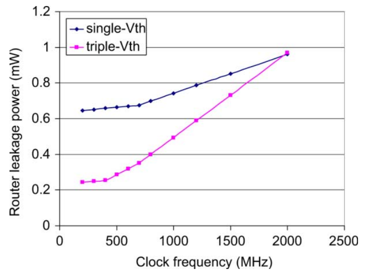

Fig. 8. Router leakage power versus clock frequency.

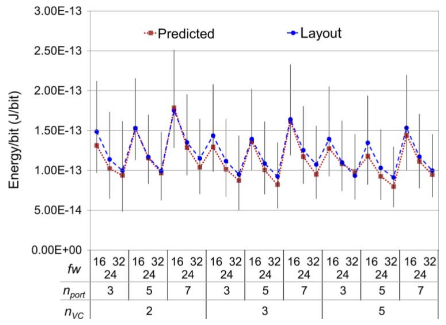

Fig. 9. Router energy-per-bit versus choice of microarchitectural parameters.

the same implementation and microarchitectural parameters, our leakage power model remains within 3.8% (on average) of the layout data. This confirms that nonparametric regression techniques can accurately capture the impact of underlying optimization techniques in the implementation flow.

Third, our proposed models enable designers to optimize over different configurations and solutions using our closed-form power and performance equations. As an example, we have performed a case-study design space exploration in which we show on-chip router energy efficiency with respect to architectural parameters. We assess the energy efficiency of a single router at a fixed activity factor using the energy-per-bit metric, where total number of simultaneous bits to a router is defined as the product of $n_{port} \times n_{vc} \times fw$. In Fig. 9, on the x-axis we show 27 data points corresponding to different combinations of flit-width, number of ports, and number of virtual channels, and on the y-axis we show the associated energy-per-bit values for each combination. In this figure, we show two sets of data: 1) the dotted line which represents our energy efficiency prediction (i.e., using our proposed closed-form models), and 2) the dashed line which represents the same metric derived from layout data. We observe that our predictions closely match the layout data. This confirms that our proposed models can efficiently drive accurate design space explorations. In addition, Fig. 9 shows the energy-per-bit as a function of n_{port}, n_{vc}, and fw. Note that buffer size does not change the total number of simultaneous bits to a router, but impacts the energy. Thus, for each combination of $(n_{vc},\ n_{port},\ fw)$ we have a range of energy values corresponding to a range of different buffer sizes. From Fig. 9 we can conclude that larger flit-width and smaller number of ports increase router energy efficiency. On the other hand, buffer size can be an important knob in controlling the achievable energy-performance envelope, as it also directly impacts the network latency.

V. Conclusion

We have proposed a new modeling paradigm in which accurate on-chip router cost models are derived using machine-learning-based nonparametric regression techniques. Our models provide significant estimation error reductions of 85% and 89%, on average, compared with ORION 2.0 [9] and parametric models, respectively. With this letter, we are releasing our modeling methodology in the form of C++ functions and scripts to aid future NoC research and design.

References

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

[2] A. Bona, V. Zaccaria, and R. Zafalon, "System level power modeling and simulation of high-end industrial network-on-chip," in *Proc. DATE*, 2004, pp. 318–323.

[3] L. C. Briand, B. Freimut, and F. Vollei, "Using multiple adaptive regression splines to support decision making in code inspection," *J. Syst. Software*, vol. 73, no. 2, pp. 205–217, 2004.

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

[5] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. DAC*, 2001, pp. 684–689.

[6] J. H. Friedman, "Multivariate adaptive regression splines," *Ann. Statist.*, vol. 19, no. 1, pp. 1–66, 1991.

[7] K. Jeong, A. B. Kahng, and K. Samadi, "Architectural-level prediction of interconnect wirelength and fanout," in *Proc. ISOCC*, 2009, pp. 53–56.

[8] 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," in *Proc. DATE*, 2009, pp. 423–428.

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

[10] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-Way multithreaded SPARC processor," *IEEE MICRO*, vol. 25, no. 2, pp. 21–29, 2005.

[11] B. Y. Lin and H. Zhang, "Component selection and smoothing in smoothing spline analysis of variance models," *Ann. Statist.*, vol. 34, no. 5, pp. 2272–2297, 2006.

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

[13] G. de Micheli and L. Benini, "Networks on chip: A new paradigm for systems on chip design," in *Proc. DATE*, 2002, pp. 2–6.

[14] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-tile sub-100-W teraFLOPS processsor in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, 2008.

[15] H. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *Proc. MICRO*, 2002, pp. 294–395.

[16] Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," *J. Syst. Software*, vol. 80, pp. 1349–1361, 2007.

[17] Synopsys Design Compiler User's Manual [Online]. Available: http://www.synopsys.com/

[18] MARS User Guide [Online]. Available: http://www.salfordsystems.com/mars.php

[19] Netmaker [Online]. Available: http://www-dyn.cl.cam.ac.uk/~rdm34/wiki

[20] Synopsys PrimeTime User's Manual [Online]. Available: http://www.synopsys.com/

[21] Cadence SOC Encounter User's Manual [Online]. Available: http://www.cadence.com/

[22] UCSD VLSI CAD Laboratory: NOC Modeling Resources [Online]. Available: http://vlsicad.ucsd.edu/NOCModeling