ORION3.0: A Comprehensive NoC Router Estimation Tool

Andrew B. Kahng, *Fellow, IEEE*, Bill Lin, *Senior Member, IEEE*, and Siddhartha Nath, *Student Member, IEEE*

***Abstract*—Networks-on-Chip (NoCs) are increasingly used in many-core architectures. ORION2.0 (see Ref [1] Kahng etal., Proc. DATE, 2009, pp. 423–428) is a widely adopted NoC power and area estimation tool, but its estimation models can have large errors (up to 185%) versus actual implementations. We present ORION3.0, an open-source tool whose parametric and non-parametric modeling methodologies fundamentally differ from ORION2.0 logic template-based approaches in that the estimation models are derived from actual physical implementation data. When compared with actual implementations, ORION3.0 models achieve average estimation errors of no more than 9.3% across microarchitecture, implementation, and operational parameters as well as multiple router RTL generators. A comprehensive suite of these methodologies has been implemented in ORION3.0 (see Ref [2] Available online: http://vlsicad.ucsd.edu/ORION3/.**

***Index Terms*—Metamodeling, networks-on-chip (NoCs), regression.**

I. Introduction

NETWORKS-ON-CHIP (NoCs) have proven to be highly scalable, low-latency interconnection fabrics in the era of many-core architectures. Because of their growing importance, NoCs must be optimized for latency and power [3]. To facilitate early design-space exploration, accurate NoC power and area estimation tools are required.

We describe ORION3.0, a comprehensive NoC router estimation tool that embodies both parametric and nonparametric models. We include new models of router component blocks using parametric [4] and nonparametric modeling [5] methodologies that fundamentally differ from ORION2.0 [1] in that the estimation models are derived from post-place-and-route (P&R) data that correspond to a given RTL generator and target cell library. Within this paradigm, we describe two approaches that are implemented in ORION3.0.

Manuscript received December 19, 2014; accepted January 29, 2015. Date of publication February 10, 2015; date of current version May 26, 2015. This work was supported in part by the NSF under Grant SHF-1116667, by the MARCO GSRC Focus Center, and by the SRC. Recommended for publication by T.-W. Kuo.

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

B. Lin is with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: billlin@eng.ucsd.edu).

S. Nath is with the Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: sinath@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.2015.2402197

The first approach is based on parametric modeling. Our work in [4] makes a substantial departure from the ORION2.0 approach in that no logic template is assumed for any router component block. Instead, for each component block in the router RTL, appropriate parametric models are derived from post-synthesis netlists by observing how instance counts change with microarchitectural, implementation, and operational parameters. We call these models *ORION_NEW*. We perform least-squares regression (LSQR) with actual post-P&R power and area data to refine these ORION_NEW models. The resulting parametric models achieve worst-case errors significantly better than those of ORION2.0. This parametric modeling methodology enables a separation of concerns and skillsets: it does not require the architect or developer to understand how the architectural components are implemented on chip. Rather, the methodology relies on a one-time characterization of post-synthesis data to derive parametric models of component blocks, and automatic fitting of these models to post-P&R data using parametric regression.

The second approach is based on nonparametric modeling. Estimation models are again derived from post-P&R power and area data that correspond to a given RTL generator and target cell library. The nonparametric modeling approach can automatically derive accurate estimation models based on a sample set of post-P&R results. ORION3.0 extends ideas from [6][7] by incorporating four metamodeling techniques for automatic model generation: radial basis functions (RBF), kriging (KG), multivariate adaptive regression splines (MARS) with linear and cubic splines, and support vector machine (SVM) regression. The nonparametric modeling approach does not require the architect or developer to understand how architectural components are physically implemented.

Often, architects may perform design-space exploration to understand the impact of instance counts, area, and power for a future technology in which technology libraries are not available and nonparametric models hence cannot be derived. Parametric models are useful at this context to provide ballpark insights on a block's size and power. Architects may then provide feedback to library teams to design complex cells or to constrain area, power, timing of the cells in the future technology libraries.

For backward compatibility, ORION3.0 also includes the logic template-based models that comprise ORION2.0. Based on modeling accuracy requirements, and availability of training and testing data for regression, users have the flexibility to choose appropriate modeling methodologies.[1]

[1]For example, when training and testing data for a technology or tool flow is not available, users may use ORION_NEW models with scaled technology parameters.

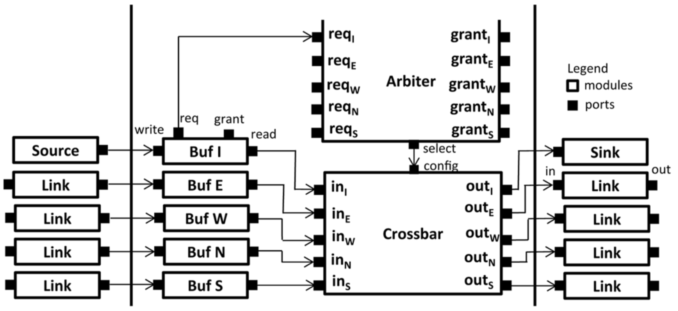
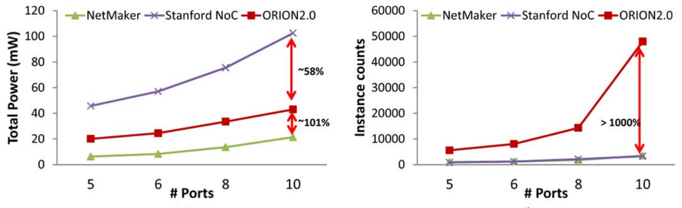


Fig. 1. Router architecture [3].



(a) (b)

Fig. 2. Poor estimations by ORION2.0 [1]. (a) Power and (b) instance counts of *Netmaker* and *Stanford NoC* versus ORION2.0 at 65 nm as a function of #ports.

Our main contributions are as follows:

1) We describe a new parametric modeling methodology that derives accurate parametric models from post-synthesis netlists by observing how instance counts change with microarchitectural, implementation, and operational parameters. Use of post-synthesis netlists accurately captures contributions from both control and data paths.
2) We demonstrate that nonparametric regression techniques (RBF, KG, MARS and SVM) can yield highly accurate (worst-case error $\leq 15\%$) NoC power and area estimates.
3) ORION3.0 is available on the web for download. Over 380 downloads have been made from industry and academia since availability commenced in February 2013.

The remainder of this letter is organized as follows. Section II presents ORION_NEW models and our parametric modeling methodology. Section III provides a description of our nonparametric modeling methodology. Section IV describes the ORION3.0 distribution itself, including software architecture and extensibility with user-defined models, as well as training and testing datasets. Section V concludes this work.

II. Parametric Modeling

Fig. 1 shows an example of a modern on-chip network router with input and output buffers, switch and virtual channel arbiter, and crossbar. ORION2.0 uses logic template models for these router blocks. However, these models can be inaccurate because of mismatches between the actual RTL and the templates assumed. Moreover, typical design flows involve sophisticated design steps that have complex interactions among them, making their effects difficult to characterize. Fig. 2(a) and (b) show power and instance-count estimation errors at 65 nm for ORION2.0 relative to two router RTL generators (*Netmaker* [8] from Cambridge and the *Stanford NoC* router [9]), as a function of the number of input ports in the router. The maximum errors are greater than 100% and 1000%, respectively. For these two RTL generators, [4] reports significant improvements in power, area, and instance-count estimation using ORION_NEW models.

A. Model Enhancements

The ORION_NEW router block models in ORION3.0 model the number of instances (or gates) in each router block; our studies show this to be required for accurate estimations of area and power. The microarchitecture parameters used are #Ports (P), #VCs (V), #Buffers (B), and Flit-width (F). The constant factors in instance-count models of the router blocks are derived by linear regression with post-synthesis netlists.

Crossbar (XBAR) Model. ORION_NEW models comprehend modern router RTL implementations which use smaller crossbars instead of traditional matrix [3] and multiplexer tree [1] implementation options in ORION2.0. Hence, the *total* number of such MUXes required is P^2F.

Switch and VC Arbiter (SWVC) Model. ORION_NEW removes the default overhead factor of 30% used by ORION2.0 based on our analyses. Instance-count in SWVC is modeled as $9(P^2V^2 + P^2 + PV - P)$. The constant factor 9 arises because six 2-input NOR gates, two INVerters, and one D-FlipFlop are used to generate each grant signal.

Input Buffer (InBUF) Model. ORION_NEW models take into account control signals and housekeeping logic which are required to decode flits and manage VCs. ORION2.0 models lack these components and are hence inaccurate. In ORION_NEW, FIFO buffers are modeled as $2PVBF$, and control signals and housekeeping logic are modeled as $180PV + 2P^2VB + 3PVB + 5P^2B + P^2 + PF + 15P$.

Output Buffer (OutBUF) Model. ORION_NEW models take into account hybrid output buffer implementations in modern router RTLs as well as control signals per port and VC associated with each buffer. Output buffers are thus modeled differently from input buffers, with OutBUF given as $80PV + 25P$.

Clock and Control Logic (CLKCTRL) Model. Unlike ORION2.0, ORION_NEW models the clock buffers and routing resources; these are modeled as 2% of the sum of instances in the SWVC, InBUF and OutBUF component blocks.

Frequency Derating Model. ORION2.0 models ignore implementation parameters such as clock frequency, which results in large estimation errors at high frequencies. We first find the frequency below which instance counts change by less than 1%. We derate instance counts by a multiplier $\Delta Instance$ that is based on this frequency as $\Delta Instance = \Delta Frequency \times ConstantFactor$.

B. Modeling Methodology

Fig. 3 shows our flow to derive new parametric models, ORION_NEW, from post-synthesis netlists by linear regression[2] and a subsequent refinement process that automatically fits the models to post-P&R area, power, and instance-count data. We use the *Netmaker* and the *Stanford NoC* router RTL generators, and a range of values of microarchitecture parameters (P, V, B and F) and implementation parameters

[2]We determine the parametric form of each router block and the constant factors by observing how instance count changes as P, V, B, F, clock frequency, and technology node vary.

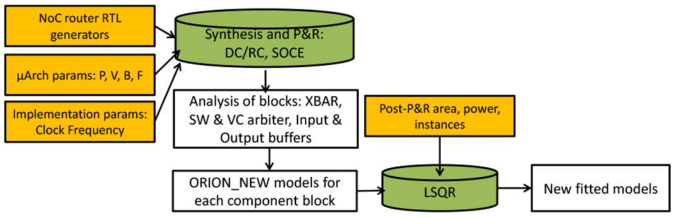

NoC router RTL generators
µArch params: P, V, B, F
Implementation params: Clock Frequency
Synthesis and P&R: DC/RC, SOCE
Analysis of blocks: XBAR, SW & VC arbiter, Input & Output buffers
Post-P&R area, power, instances
ORION_NEW models for each component block
LSQR
New fitted models

Fig. 3. Development of ORION_NEW and fitted models using post-P&R data.

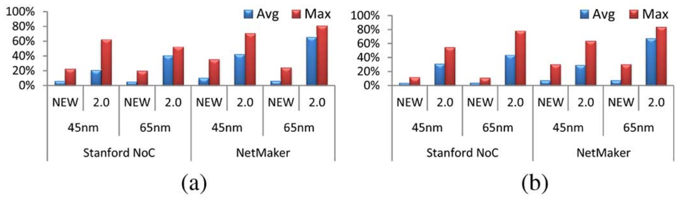


Fig. 4. Regression fit versus ORION2.0. (a) area and (b) power estimation errors.

(clock frequency and technology node) to configure the router. We synthesize the router RTLs using Synopsys *Design Compiler vI-2013.12-SP2* (DC) [10] and Cadence *RTL Compiler vEDI13.1* (RC) [11], with options to preserve module hierarchy to enable us to analyze each router component block. We derive ORION_NEW models from analysis of post-synthesis netlists of the component blocks.

To refine these models, we generate post-P&R power and area data. We place and route the synthesized netlists using Cadence *SOC Encounter vEDI13.1* (SOCE) with die utilization of 0.75 and die aspect ratio of 1.0. To account for process variation, we perform multimode multicorner P&R by defining two scenarios for both setup and hold analyses. The *nominal* scenario uses {ss, 0.85 V, 125C}[3] and {ff, 1.05 V, 125C} for setup and hold corners respectively, and the *overdrive* scenario uses {ss, 1.10 V, 125C} and {ff, 1.30 V, 125C} for setup and hold corners, respectively. With each scenario, we use foundry libraries for 65 and 45 nm. To run power analysis based on the post-P&R netlist, SPEF [10] and SDC [12], we use Synopsys *PrimeTime-PX vH-2013.06-SP3-6* (PT-PX) [10]. To account for the effects of input toggle rates, we use the *set_switching_activity* command to annotate switching activity factor values from 0.05 to 1.0 in steps of 0.05 and the percentage of time the signal is at logic 1 from 0.1 to 1.0 in steps of 0.1.[4] Finally, we use the *MATLAB vR2012b* function *lsqnonneg* to fit the models to post-P&R data. We normalize all the input parameters using z-scoring, and randomly select 35% (respectively, 15%) of the data points to train (respectively, validate) the model. We then use the remaining 50% of the data points to test the model and report average, maximum estimation errors.

C. *Results*

Fig. 4(a) and (b), respectively, compare power and area estimation errors of ORION2.0 to those of ORION_NEW at 45 and 65 nm. Both maximum and average errors are substantially improved. The ORION_NEW estimates are very close to actual implementation (average error of 9.3% in estimating *Netmaker* power at 45 nm) and are robust across multiple microarchitecture, implementation parameters, and router RTLs. Compared to [4], we improve average error in area (respectively, power) estimation from 9.8% to 9.3% (respectively, 10.2% to 6.1%).

[3]We represent process, voltage, and temperature corners as three tuples—{process, voltage, temperature}. Our corners consist of two process conditions (*ss*, *ff*), four voltages (0.85, 1.05, 1.10, and 1.30 V) and one temperature (125C).

[4]Specific NoC routing algorithms affect the switching activity factors at the inputs of the router. Therefore, we do not model different routing algorithms as they are subsumed by the input switching activity factors.

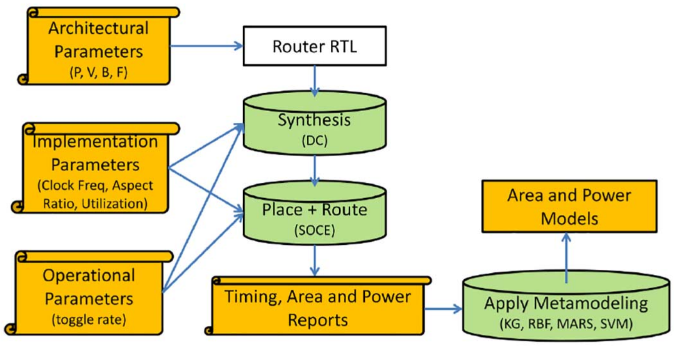

Architectural Parameters (P, V, B, F)
Router RTL
Implementation Parameters (Clock Freq, Aspect Ratio, Utilization)
Synthesis (DC)
Place + Route (SOCE)
Operational Parameters (toggle rate)
Timing, Area and Power Reports
Apply Metamodeling (KG, RBF, MARS, SVM)
Area and Power Models

Fig. 5. Development of nonparametric regression models using post-P&R data.

III. Non-Parametric Modeling

Non-parametric regression techniques provide another approach to estimate NoC power and area [6][7]. The models determine the interactions between all input variables and how they affect the output (or response). This alleviates the effort needed to model architecture-level implementations of NoCs. At the same time, nonparametric regression approaches are scalable across multiple router RTLs, technology libraries and commercial tool flows. In ORION3.0, we implement four popular nonparametric regression or *metamodeling* techniques—RBF, KG, MARS, and SVM. Detailed descriptions of these techniques are in [13].

A. *Modeling Methodology*

We derive NoC area and power models by performing nonparametric fit of post-P&R data using 65 and 45 nm technology libraries. Fig. 5 shows our flow to derive nonparametric regression models. We apply the methodology described in Section II-B to run synthesis and multimode multicorner P&R, perform power simulations by annotating activity factors, and generate datasets for modeling.

B. *Results*

We use 256 data points of post-P&R power and area values using 45 nm and 65 nm technology libraries to generate training and test data points. The input variables to all the models are P, V, B, and F and the responses are post-P&R power and area. We use two training set sizes—"sparse and restricted" with 50 data points that omit higher values of the microarchitectural parameters,[5] and "sparse only" with 64 data points that are sampled using Latin Hypercube Sampling [14]. The "sparse and restricted" set allows us to assess how well the models generalize

[5]More precisely, the resulting training sets omit all values of $\{B = 7\}$, or of $\{P = 9\}$, or of $\{V = 7\}$, or of $\{F = 64\}$.

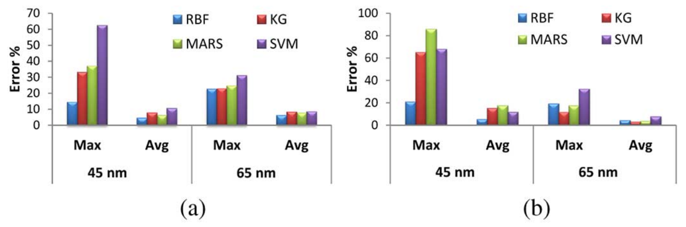


Fig. 6. Estimation errors in (a) area and (b) power with "sparse and restricted" training sets (i.e., 50 data points for training).

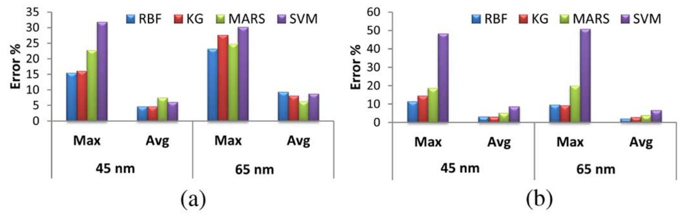


Fig. 7. Estimation errors in (a) area and (b) power with "sparse only" training sets (i.e., 64 data points for training).

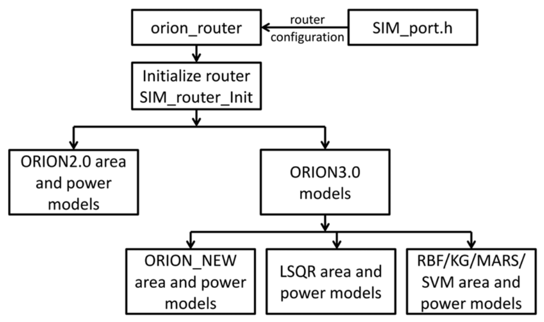

orion_router
router configuration
SIM_port.h
Initialize router SIM_router_Init
ORION2.0 area and power models
ORION3.0 models
ORION_NEW area and power models
LSQR area and power models
RBF/KG/MARS/ SVM area and power models

Fig. 8. Software architecture of ORION3.0.

in estimating area and power for input parameters which are beyond the range of values used for training. In each experiment, model generation takes around 3 seconds and response estimation takes around 1.88 seconds.[6] We apply z-scoring to normalize all the input parameters and randomly select 15% of the remaining data points for validation. We perform five-fold cross-validation [13] and vary the hyperparameters [13] for each technique until our average errors are less than 12%. We repeat these experiments 30 times for each training set size, and report the averages of all the error values across the 30 trials.

Fig. 6(a) and (b) respectively compare area and power estimation errors across all modeling techniques at 45 and 65 nm for the "sparse and restricted" training sets. The average errors are $\leq$ 10% in both area and power. RBF generally performs better than other techniques across technologies in both area and power estimation. RBF can be up to 3$\times$ more accurate than KG, MARS and SVM. Fig. 7(a) and (b) show similar plots for the "sparse only" training sets. Compared to [5], we improve average error in area (respectively, power) estimation from 8.7% to 6.7% (respectively, 8.5% to 3.8%) with the "sparse only" training sets. Again, RBF is more accurate than other techniques. Across all training set sizes used in our experiments, area and power estimation errors are the smallest for RBF and are the largest for SVM.[7]

IV. ORION3.0 Distribution

We now describe ORION3.0 software architecture, extensibility with new models and training/testing datasets, and details of the ORION3.0 software distribution.

A. Software Architecture and Extensibility

ORION3.0 uses a modular software architecture and is written in *C* and *MATLAB vR2012b*. Fig. 8 shows the high-level software architecture, how the router configuration is read by the tool and how models are invoked. ORION3.0 offers command-line options for the user to choose: 1) ORION3.0 versus ORION2.0 models; 2) specific ORION3.0 modeling techniques (*basic, lsqr, rbf, kg, mars, svm*); and 3) training and/or testing datasets when any of {*lsqr, rbf, kg, mars, svm*} is used.

As with ORION2.0, users can configure microarchitecture parameters such as flit-width, #input and output buffers, #virtual channels, #pipeline stages, type of crossbar, #ports, etc. in the *SIM_port.h* file. These parameters are used with either the ORION2.0 models or the ORION3.0 *basic* model; the latter refers to ORION_NEW (cf. Section II), which is used by default when no options are specified by the user. Users may specify implementation and operational parameters such as technology library, size of MUXes in the crossbar, and the input load on the router from the link. We have updated the technology files with accurate leakage and internal power data from a leading foundry's 45GS libraries for all cell types used in modeling, and are in the process of calibrating foundry 28 nm library models. All models report area in μm^2, power in mW, and energy in J.

When choosing any of *lsqr, rbf, kg, mars, svm* methods, a user may optionally provide training and testing data points. ORION3.0 performs basic validation of such user data to ensure that it can be converted to a nonsingular matrix. In the absence of user-provided data, the tool uses default training and testing data points based on the technology configured in the *SIM_port.h* file in the *PARM_TECH_POINT* field. Users may also develop their own regression models in *MATLAB*: the ORION3.0 distribution provides a template shell script that executes the regression model in *MATLAB*, and the shell script is called from *orion_router.c* using the *system* method in *C*.

B. Software Distribution

ORION3.0 is downloadable at http://vlsicad.ucsd.edu/ORION3/ [2]. We provide academic *MATLAB* toolboxes for RBF [15], KG [16], MARS [17], and SVM [18] under the same copyright and license agreements as available in their distributions. *Doxygen*-based documentation of all functions and implemented structures is provided in the distribution.

V. Conclusion

Accurate modeling for NoC area and power estimation is critical to successful early design-space exploration in the era

[6]We measure execution times on a six-core Intel Xeon E5-2640 2.5 GHz platform.

[7]We do not apply parametric modeling to certain blocks and nonparametric modeling to other blocks. We model the entire router using nonparametric modeling to generalize these models across router implementations. Further, incorrect parametric models can lead to large estimation errors. Our experimental results show that an architecture-specific model of the crossbar can lead to area and power estimation errors of up to 61% and 53% respectively.

of many-core computing. ORION2.0, while very popular, has large errors versus actual implementation. This is because there is often a mismatch between the actual router RTL and the templates assumed. Also, typical design flows involve sophisticated optimizations that are difficult to characterize. We present ORION3.0, an open-source tool that incorporates comprehensive parametric and nonparametric modeling techniques to accurately estimate NoC power and area. Our ORION_NEW parametric models explicitly account for control and data path resources. We further refine these parametric models by least-squares regression (LSQR) on post-P&R data. ORION3.0 nonparametric models include four popular techniques—RBF, KG, MARS, and SVM. Our studies show that these techniques can be low-overhead and highly accurate in estimating NoC power and area, with RBF being more accurate than the other methods for sparse and restricted training sets. ORION3.0 is now available for web download [2].

Acknowledgment

The authors would to thank J. Fong for developing the front-end interfaces for ORION3.0.

References

[1] 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.
[2] ORION3.0. [Online]. Available: http://vlsicad.ucsd.edu/ORION3/
[3] H.-S. Wang, L.-S. Peh, and S. Malik, “Orion: A power-performance simulator for interconnection networks,” in *Proc. MICRO*, 2002, pp. 294–305.
[4] A. B. Kahng, B. Lin, and S. Nath, “Explicit modeling of control and data for improved noc router estimation,” in *Proc. DAC*, 2012, pp. 392–397.
[5] A. B. Kahng, B. Lin, and S. Nath, “Comprehensive modeling methodologies for noc router estimation,” UCSD CSE Dept., La Jolla, CA, USA, Tech. Rep. CS2012-0989, 2012.
[6] A. B. Kahng, B. Lin, and K. Samadi, “Improved on-chip router analytical power and area modeling,” in *Proc. ASP-DAC*, 2010, pp. 241–246.
[7] K. Jeong, A. B. Kahng, B. Lin, and K. Samadi, “Accurate machine learning-based on-chip router modeling,” *IEEE Embed. Syst. Lett.*, vol. 2, no. 3, pp. 62–66, Sep. 2010.
[8] Netmaker. [Online]. Available: http://www-dyn.cl.cam.ac.uk/~rdm34/wiki
[9] Stanford NoC. [Online]. Available: https://nocs.stanford.edu/cgi-bin/trac.cgi
[10] Synopsys, Inc. [Online]. Available: http://www.synopsys.com/Tools
[11] Cadence Design Systems, Inc. [Online]. Available: http://www.cadence.com/products/
[12] J. Bhasker and R. Chadha*, Static Timing Analysis for Nanometer Designs: A Practical Approach*. New York, NY, USA: Springer, 2009.
[13] T. Hastie, R. Tibshirani, and J. Friedman*, The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin, Germany: Springer-Verlag, 2009.
[14] R. Jin, W. Chen, and T. W. Simpson, “Comparative studies of metamodeling techniques under multiple modeling criteria,” *Trans. Struct. Multidiscip. Optim.*, vol. 23, pp. 1–13, 2011.
[15] RBF2 Manual. [Online]. Available: http://www.anc.ed.ac.uk/~/rbf.html
[16] S. N. Lophaven, H. B. Nielsen, and J. Sondergaard, “Aspects of the MATLAB Toolbox DACE,” Tech. Univ. of Denmark, Lyngby, Denmark, Tech. Rep. IMM-REP-2002-13, 2002.
[17] ARESLab. [Online]. Available: www.cs.rtu.lv/jekabsons/regression.html
[18] LIBSVM. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm