Comprehensive Modeling Methodologies for NoC Router Estimation

Andrew B. Kahng Bill Lin Siddhartha Nath

Abstract

Networks-on-Chip (NoCs) are increasingly used in many-core architectures. ORION2.0 [18] is a widely adopted NoC power and area estimation tool that is based on circuit-level templates, which use specific logic structures to model implementation of different router components. ORION2.0 estimation models can have large errors (up to 185%) versus actual implementation, often due to a mismatch between the actual router RTL and the templates assumed, as well as the effects of optimization tools in modern design flows. In this work, we propose comprehensive parametric and non-parametric modeling methodologies that fundamentally differ from logic template based approaches in that the estimation models are derived from actual physical implementation data. Specifically, we propose a new parametric modeling methodology as well as improvements to previous work on non-parametric modeling.

Our work on parametric modeling proposes (1) ORION_NEW models, developed using a new methodology that does not assume any logic templates for router components, (2) the use of parametric regression to fit the new models to post-layout power and area values, and (3) modeling extensions that enable more detailed flit-level power estimation when integrated with simulation tools such as *GARNET*. The ORION_NEW methodology 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 NoC models.

Our work on non-parametric modeling expands on our previous work [17] and demonstrates the use of four widely used non-parametric modeling techniques: Radial Basis Functions, Kriging, Multivariate Adaptive Regression Splines, and Support Vector Machine Regression. The estimation models are also derived from actual physical implementation data. We observe that non-parametric modeling techniques have low overhead and complexity of modeling, and are highly accurate.

When compared with actual implementations, our modeling methodologies achieve average estimation errors of no more than 9.8% across microarchitecture, implementation, and operational parameters as well as multiple router RTL generators. These methodologies are being implemented in an ORION3.0 distribution [19], [66].

I. Introduction

Networks-on-Chip (NoCs) have proven to be highly scalable and low-latency interconnection fabrics in the era of many-core architectures, as evidenced by commercial chips such as the Intel 80-core [63], IBM Blue Gene [64] and Tilera TILE-Gx [65] processors. Because of their growing importance, NoC implementations must be optimized for latency and power [41], [5], [31], [28]. To facilitate early design-space exploration, accurate NoC power and area estimators are required.

One widely adopted approach for NoC power and area estimation, as embodied in ORION2.0 [18], is to develop a logic structure template for each NoC router component, namely, for the input and output buffers, crossbar, and switch and virtual channel (VC) arbiters. Despite significant enhancements to improve leakage and clock power modeling, ORION2.0 still produces large estimation errors versus actual implementation. This is because there is often a mismatch between the actual RTL and the templates assumed. Also, typical design flows involve sophisticated design steps that have complex interactions among them, making their effects difficult to characterize. For example, the crossbar is assumed to have a multiplexer tree structure, and the switch and VC arbiter is assumed to be a set of NOR and INV gates. However, after logic synthesis and technology mapping, the actual structure can be quite different. For example, AOI instances may be used instead of NOR and INV gates for the arbiter, and tri-state buffers may be used to implement the crossbar. Logic synthesis may also add buffers for performance optimizations. These synthesis and other design flow transformations are hard to predict.

An alternative approach is to use parametric regression with high-level analytical models. Although regression analysis is performed using actual physical implementation data, previous evaluations of this approach [17] show that high-level parametric regression also leads to large estimation errors because important architecture details are often missing in these analytical models.

To illustrate the degree of inaccuracy, we show in Figure 1(a) power estimation errors at 65nm for both ORION2.0 and the parametric regression approach based on high-level analytical models, as a function of the number of virtual channels in the router. The maximum errors are 185% and 75%, respectively. Similarly, in Figure 1(b), we show power estimation errors at 90nm when the flit-width is changed.

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

B. Lin is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407. E-mail: billlin@eng.ucsd.edu.

S. Nath is with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0404. E-mail: sinath@cs.ucsd.edu.

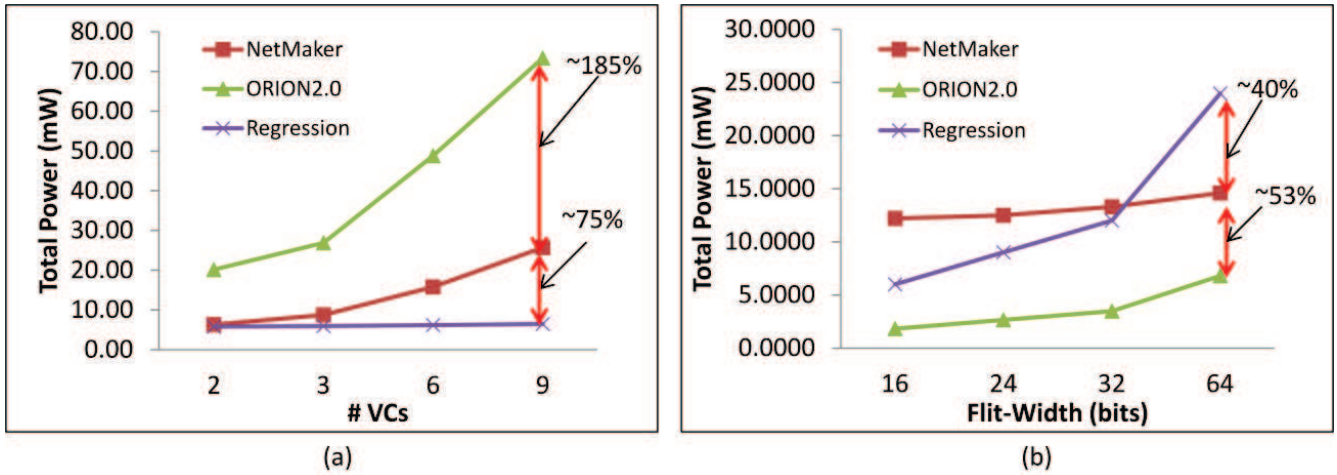

Fig. 1. Poor estimations by ORION2.0 [18] and the previous parametric regression approach [17]. *Netmaker* vs. ORION2.0 vs. regression at (a) 65nm and (b) 90nm.

In our present work, we propose comprehensive parametric and non-parametric modeling methodologies that fundamentally differ from approaches like ORION2.0 in that the estimation models are derived from actual post place-and-route (P&R) data that corresponds to the actual RTL generator and target cell library. Following this paradigm, we describe two approaches.

The first is based on parametric modeling. Our work here is a substantial departure from the ORION2.0 approach because 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 the 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 as well as previous non-parametric regression approaches [17]. We describe modeling extensions that enable more detailed flit-level power estimation when integrated with simulation tools such as *GARNET* [1]. Our parametric modeling methodology does not require the architect or developer to understand how the architectural components are implemented. 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 non-parametric modeling, which was first described in [17]. In this approach, estimation models are also derived from actual post P&R layout power and area data that correspond to the actual RTL generator and target cell library. As described in [17], the non-parametric modeling approach is powerful in that it can automatically derive accurate estimation models based on a sample set of post P&R results. In [17], Multivariate Adaptive Regression Splines (MARS) linear splines were used to derive the estimation models. In this work, we extend the ideas of [17] by incorporating four other metamodeling techniques for automatic model generation: Radial Basis Functions (RBF), Kriging (KG), Multivariate Adaptive Regression Splines with cubic splines, and Support Vector Machine (SVM) regression. This non-parametric modeling approach also does not require the architect or developer to understand how the architectural components are implemented.

Our main contributions are as follows:

1) We describe a new parametric modeling methodology that derives accurate parametric models from actual post-synthesis netlists generated from actual router RTLs by observing how instance counts change with microarchitectural, implementation, and operational parameters. The post-synthesis netlists accurately capture contributions from both the control and data paths. The derived models are highly accurate and robust across multiple router RTLs, and across microarchitecture, implementation, and operational parameters.
2) We demonstrate that parametric regression with accurate models can significantly reduce the worst-case error compared to previous template-based approaches as well as non-parametric regression approaches for NoC routers.
3) We demonstrate that popular non-parametric metamodeling techniques, namely, Radial Basis Functions, Kriging, Multivariate Adaptive Regression Splines, and Support Vector Machine regression, can be highly accurate (worst-case errors less than 20%) in NoC power and area estimations.
4) 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 II reviews related work. Section III describes the ORION_NEW model. Section IV describes the ORION_NEW modeling methodology, and Section IV-C presents our new flit-level power estimation model. Section V provides background on the four metamodeling techniques, and Section VI describes our modeling methodology using these techniques. Section VII presents experimental results to validate our modeling approaches: Section VII-A compares ORION_NEW models with ORION2.0 and the non-parametric regression approach in [17], while Section VII-B presents and compares results of the four metamodeling techniques. Section VIII concludes and outlines future

work.

II. Related Work

Previous works have focused primarily on two broad modeling paradigms: (1) architecture-level models using templates for each router component block (crossbar, switch and VC arbiter, and input and output buffers), and (2) RTL and gate-level simulation-driven models. For the first approach, Patel et al. [30] propose a transistor count-based analytical model for NoC power. Large errors can result because router microarchitectural parameters are not considered. ORION [41] and ORION2.0 [18] use microarchitecture and technology parameters for the router component blocks. From our studies as well as from [17], ORION2.0 estimates have very large errors because the models assume logic template structures for router microarchitecture blocks.

The second approach is based on pre-layout (RTL or post-synthesis gate-level) [29], [4], [9], [23] or post-layout [3], [2], [32], [27] simulations. Banerjee et al. [2] report accurate power for a range of routers, but do not present any analytical models for router power. Chan et al. [4] develop cycle-accurate power models with reported average errors up to 20%. Meloni et al. [27] and Lee et al. [23] perform parametric regression analysis on post-layout and RTL simulation results, respectively. Their models are fairly coarse-grained, e.g., they cannot explain sensitivity of power dissipation in each router block to change in load, microarchitecture or implementation parameters.

In the area of flit-level power estimation, Ye et al. [43] and Penolazzi et al. [32] estimate power dissipation using a bit-level model, and Penolazzi et al. [32] 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.

Non-parametric regression (metamodeling) techniques have been previously used in the field of VLSI and computer architecture. Ipek et al. [13] use Artificial Neural Networks (ANNs) to predict performance of core and memory in the chip-multiprocessor design space. Kriging (KG) has been widely used for spatial correlation modeling of IR drop and on-chip temperature [21], as well as estimation of spatial structure of variability sources, and for variability characterization [34]. Lee et al. [25] use KG to automatically search for an optimal microarchitectural design space to estimate tradeoffs in processor design. Lee et al. [22] use KG to estimate processor power. Multivariate Adaptive Regression Splines (MARS) have been used to model NoC power and area [17], [14]. Support Vector Machine (SVM) regression has been used in performance modeling of analog circuits [20]. SVM classification has been used for fast simulation of rare circuit events to test SRAM [38]. Radial Basis Functions (RBF) have been used to predict crosstalk in interconnects [12]. We choose to explore RBF, KG, MARS and SVM metamodeling techniques in our work because (1) these methods have been used successfully for estimation purposes in VLSI circuit design, and (2) previous reviews have determined that they are more accurate and robust than other available non-parametric modeling techniques [15], [45].

Figure 2 shows the different techniques used to model NoC routers. Our work is highlighted in the orange ovals.

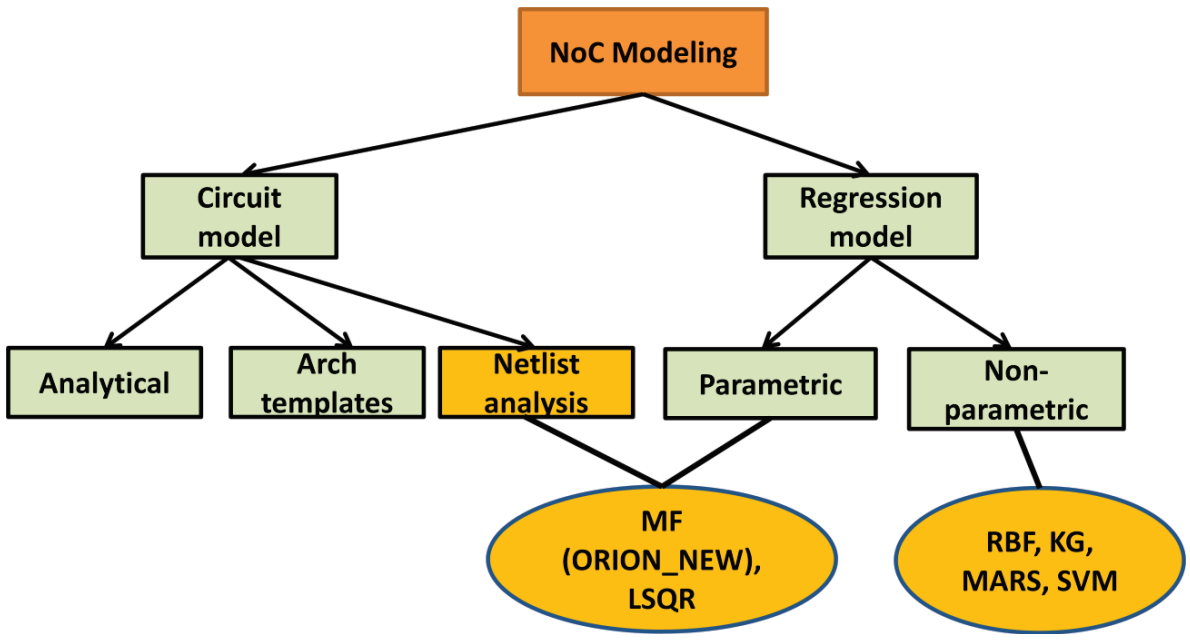

Fig. 2. NoC modeling classification and our work. We use *manual fitting* (MF) on post-synthesis netlists to generate ORION_NEW models. We use least-squares (LSQR) regression fit of ORION_NEW models with post-layout area and power.

III. ORION_NEW – Improvements to Architectural Model

As shown in Figures 1(a) and (b) of Section I, ORION2.0 [18] and the previous parametric regression techniques [17] have large errors compared to implementation. This is because NoC architecture template-based models are incomplete. They do not consider the impact of frequency scaling, and do not consider more optimized implementations of blocks such as the crossbar.

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* [55] from Cambridge and the *Stanford NoC* router [56]. (Our methodology is described in detail in Section IV.) Figure 3 shows the component blocks of a router, i.e., crossbar, switch and VC arbiter, and input and output buffers [31]. 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).

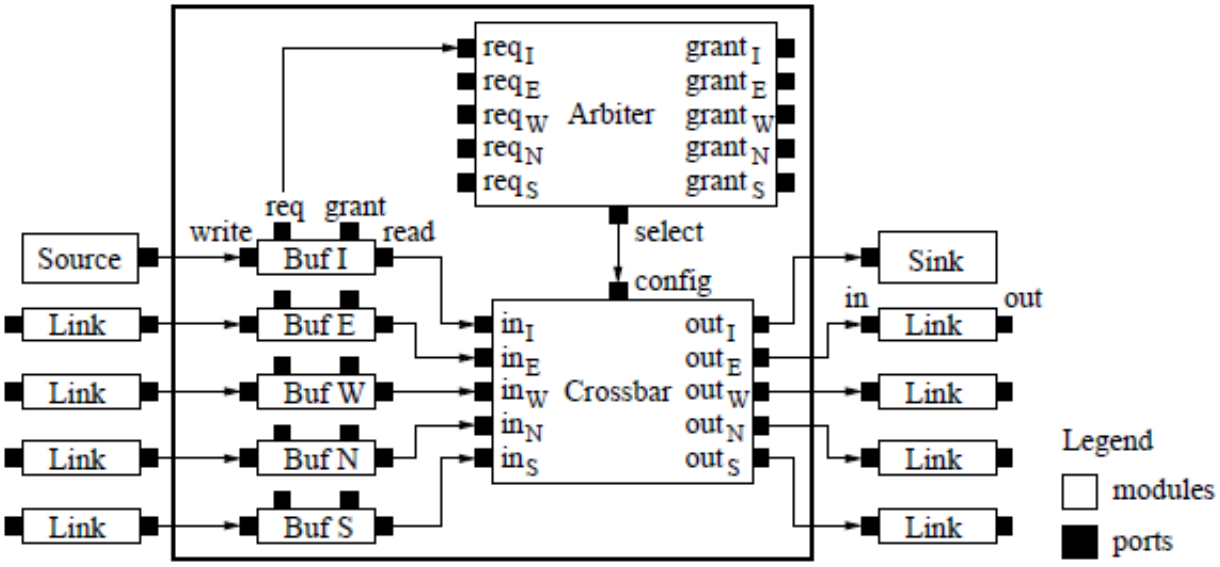

Fig. 3. Router architecture [41].

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

1) tri-state crossbar model;
2) control resources such as FIFO select and decode logic signals in the input and output buffers;
3) additional input buffer resources for delay-optimized arbiters [31];
4) output buffer model to store only head flits;
5) clock and control logic resources; and
6) clock frequency-based derating of total number of instances in the router.

A. Crossbar (XBAR) Model

The crossbar (XBAR) is responsible for connecting input ports to output ports so that all flit bits are transferred to output ports [7]. ORION2.0 crossbar models consider two implementations, matrix [31] and multiplexer tree [18]. The multiplexer tree is the smaller of these in terms of instance counts 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 is: $P \times P \times F$. The new model reduces the instance count by a factor of $(P-1)$ when compared to the multiplexer tree implementation.

B. Switch and VC Arbiter (SWVC) Model

The switch and VC arbiter (SWVC) generates control signals for the crossbar such that a connection is established between input buffers to output ports [7]. ORION2.0 adds an overhead of 30% to the arbiter by default. Our analysis indicates that this overhead is not needed with frequencies in the range 400MHz-900MHz for process nodes 45nm to 130nm. Beyond this range of frequency, a derating factor must be applied, which is discussed in Section III-F. The ORION_NEW model for switch and VC 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. The new model removes the overhead factor of 30%.

C. Input Buffer (InBUF) Model

The input buffer (InBUF) holds the entire incoming payload of flits at the input stage of the router for decode [7]. ORION2.0 models only the buffer instances and does not take into account control signals which are needed at this stage for decode; these include 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 ORION_NEW, 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 [31], leading to doubling of input buffer resources. Hence, in the new model the number of FIFO buffers is $2 \times P \times V \times B \times F$. 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 and 5 account for instances that realize local flags in the decode logic. The factor 15 corresponds to the number of buffers in each FIFO select path of an input port.

D. Output Buffer (OutBUF) Model

The output buffer (OutBUF) holds the head flits between the switch and the channel for a switch with output speedup [7]. ORION2.0 models output buffers in exactly the same way as it models 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 store enough flits to match speeds 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 using $P \times V \times B \times F$ 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 counts grow in the new model 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.

E. Clock and Control Logic (CLKCTRL) Model

The clock and control logic (CLKCTRL) models clock buffers and control logic routing resources as clock frequency scales. ORION2.0 does not model impact on these resources because of frequency scaling. ORION_NEW models these resources as 2% of the sum of instances in the SWVC, InBUF and OutBUF component blocks.

F. Frequency Derating Model

As frequency changes, timing constraints change. To meet setup time at higher frequencies, buffers are inserted which leads to an overall increase in instance counts 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. We find that the number of instances in the crossbar does not vary significantly with frequency because there are no critical paths; we thus ignore the effects of frequency on the crossbar.

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

IV. ORION_NEW Methodology

In this section, we first describe manual and regression-based methodologies that we use to develop the ORION_NEW models; we then describe how we estimate power and area using these approaches. We extend our methodology to flit-level power estimation in Section IV-C. Key elements of our modeling methodology (see details in Table I) include the following.

- Multiple parametrized NoC RTL generators: *Netmaker* [55] from Cambridge and the *Stanford NoC* [56].
- Range of values of microarchitecture parameters (#Ports (P), #VCs (V), #Buffers (B) and Flit-width (F)) and implementation parameters (clock frequency and technology node).
- Operational parameters for power calculation: toggle rate[1] (*TR*) and static probability of 1's in the input (*SP*).
- Multiple commercial tools: Synopsys *Design Compiler* (DC) [46] and Cadence *RTL Compiler* (RC) [52], 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) [53] 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) [47] to run power analysis based on the post-P&R netlist, SPEF [54] and SDC [51].
- *MATLAB* [57] function *lsqnonneg* for regression analysis.

Figure 4 shows the flow we use to develop ORION_NEW models for each component block of the router. In Table II, we summarize the ORION_NEW modeling of instance counts of each component block.

TABLE I
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 $45nm$ = OpenPDK45 from NCSU/OSU $\{45nm, 65nm\}$ = TSMC GS, GP resp. $\{90nm, 130nm\}$ = TSMC G, GHT resp.
Op	Toggle Rate Static Prob of 1's	$TR = \{0.2, 0.4, 0.6, 0.8\}$ $SP = \{0, 0.25, 0.5, 0.75, 1.0\}$
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
P&R	Cadence *SOCE*	*set_default_switching_activity -input_activity TR* *propagate_activity* remaining flow is default
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*

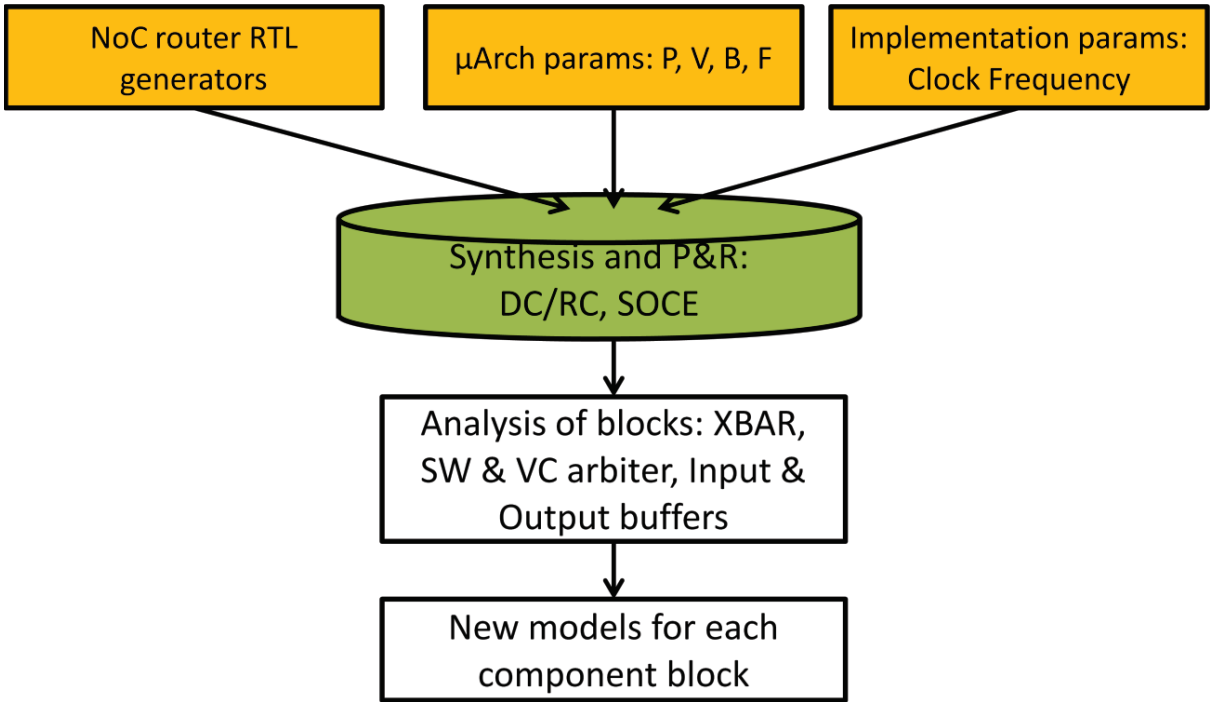

Fig. 4. High-level flow used to develop the ORION_NEW models.

There are two ways to estimate NoC area and power using the ORION_NEW models, as shown in Figure 5. The manual approach is described in Section IV-A, and the regression analysis approach is described in Section IV-B. The benefits of both are described below.

- Both the approaches achieve minimum estimation error when the router RTLs are modular, so that the 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 (G, LP) $\times$ (HVT, NVT, LVT) $\times$ (bc, wc) to correctly select a technology library file. The regression analysis approach, on the other hand, is agnostic of implementation details and only depends on the set of training data.
- The manual approach leads to faster estimation since it only involves technology library lookups and plugging-in of library values into the ORION_NEW models. In contrast, the regression analysis approach requires synthesis and P&R to be performed on the router RTL to generate data points for the training set. On an Intel Core *i*3 2.4GHz processor,

[1]Toggle rate is defined as the average number of times a signal transitions from high to low (or low to high) per clock cycle. We assume the toggle rate of the clock signal to be 1.

TABLE II
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	$0.02\times(SWVC+InBUF+OutBUF)$

the runtime of the manual approach when used with ORION2.0 code is less than 10ms, whereas the regression analysis approach takes 64 SP&R runs to generate the data points for training.[2]

- 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 large 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.

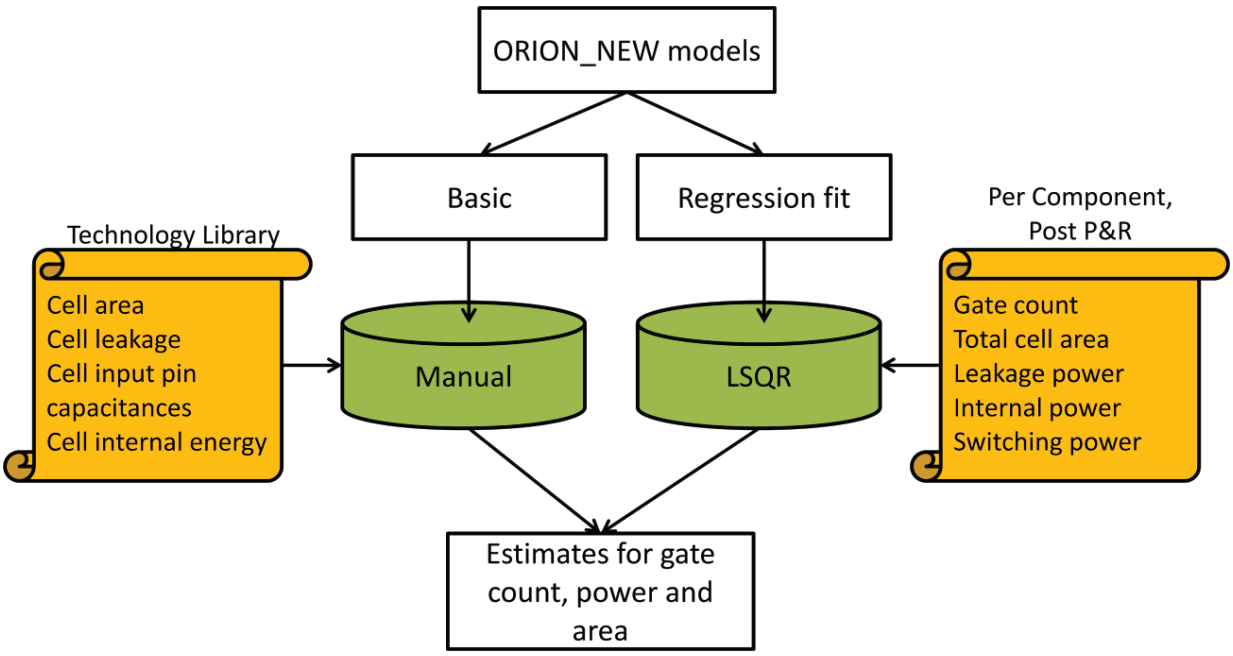

Fig. 5. High-level view of power and area estimation methodology using manual and regression analysis (LSQR) approaches.

A. Manual Approach to Estimate NoC Power and Area

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

- microarchitecture parameters $\{P, V, B$ and $F\}$ and implementation parameter (clock frequency and technology node);
- cell areas, leakage, internal energy and load capacitance; and
- toggle rate.

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

TABLE III
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)	$\left(\frac{Area_{AOI}+Area_{INV}}{2}\right)\times(CLKCTRL)_{insts}$

[2]In our simulations we use 64 data points for training and 192 data points to test the model.

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. Internal and switching power together constitute dynamic power, which varies with operating voltage, capacitive load and frequency of operation. Internal power is the power dissipated inside a cell and consists of short-circuit power and switching power of internal nodes; switching power is the power consumed when a load capacitance on a net is charged and discharged.

In ORION_NEW, toggle rate (*TR*) is equal to the average toggle rate of all input signals in the crossbar, switch and VC arbiter, and control logic in input and output buffers. We assume that buffer cells toggle at 25% of the input toggle rate, 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.

$$P_{leak\_XBAR} = MUX_{leak} \times XBAR_{insts} \tag{1}$$

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

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

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

Internal power calculation. For internal power, table lookups in technology library files return the internal energy of a standard cell given its load capacitance[3] and input slew value of $\approx 5 \times FO4\ delay$.[4] Internal energy is the minimum of the rise and fall energies. Equations (5)-(8) are used to calculate internal power of each component block.

$$P_{int\_XBAR} = MUX_{int} \times TR \times XBAR_{insts} \tag{5}$$

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

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

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

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 of the net. The wire capacitance is approximately calculated as a constant factor times the total pin capacitances. This constant factor is set to 1.4 at 65nm and is assumed to decrease by 14% with each successive process node shrink. Equations (9)-(12) are used to calculate switching power of each component block.

$$P_{sw\_XBAR} = XBAR_{load} \times TR \times XBAR_{insts} \tag{9}$$

$$P_{sw\_SWVC} = SWVC_{load} \times TR \times SWVC_{insts} \tag{10}$$

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

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

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 toggle rate at inputs.
2) Use models in Table II to calculate the instance count of each component block of the router.
3) Use models in Table III 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 power, total internal power and total switching power.

[3]Load capacitance of a cell depends on its fanout and the cell(s) it drives. We use a fanout of 1. The cells driven depend on the component of the router as shown in Table III. For example, one DFF drives another DFF and one AOI in the input and output buffers. So, the load capacitance of the DFF is the sum of input pin capacitances of one DFF and one AOI.

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

B. 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 results. Options are set in synthesis tools 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* [57] function *lsqnonneg* for this purpose, and tool options as given in Table I.

Flow Details. LSQR is applied to fit a model of post-P&R instance counts for each router component block. Our training set has 64 data points. Parametric LSQR is setup as

$$a_1 \cdot Insts_{mod\ <component>} + a_0 = Insts_{tool\ <component>} \tag{13}$$

where $Insts^R_{mod\ <component>}$ 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

$$b_1 \cdot Insts^R_{mod\ <component>} + b_0 = Area_{tool\ <component>} \tag{14}$$

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

$$\begin{aligned} &\{c_5,\ d_5,\ e_5\} \cdot Insts^R_{mod\ XBAR} + \{c_4,\ d_4,\ e_4\} \cdot Insts^R_{mod\ SWVC} + \\ &\{c_3,\ d_3,\ e_3\} \cdot Insts^R_{mod\ InBUF} + \{c_2,\ d_2,\ e_2\} \cdot Insts^R_{mod\ OutBUF} + \\ &\{c_1,\ d_1,\ e_1\} \cdot Insts_{mod\ CLKCTRL} = \{P_{leak},\ P_{int},\ P_{sw}\}_{tool} \end{aligned} \tag{15}$$

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 can result in large errors, e.g., 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 model development flow.[5]

C. 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 two encodings with two consecutive 8-bit flits, where every flit has exactly four bits as 1. In the first encoding, the two consecutive flits are $8b'11110000$ and $8b'11110000$. In the second encoding, the two consecutive flits are $8b'11110000$ and $8b'00001111$. In the first encoding, there are no toggles per consecutive flits, whereas in the second encoding there are eight toggles per consecutive flits. Clearly, the second encoding will lead to higher dynamic power than the first one. To model this effect, we use the flow shown in Figure 6. Before using a testbench, the netlists must pass an equivalence check using tools such as Synopsys *Formality* [48]. We inject different bit encodings in the input during simulation over 10000 cycles using *GARNET* [1], and the resultant VCD (Value Change Dump) is validated using a waveform analyzer such as Synopsys *DVE* [49]. A satisfactory VCD is used as input to Synopsys *PrimeTime-PX* [47] 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* to obtain very accurate estimates.

V. Non-parametric regression for estimation

Non-parametric regression techniques provide another approach to estimate NoC power and area [17]. Such techniques can considerably reduce modeling efforts because they require only the microarchitectural, implementation, and operational parameters as input variables. The models determine the interactions between these input variables and how they affect the output (or response). This alleviates the effort needed to model a NoC, as details of architecture-level implementations are avoided. At the same time, non-parametric regression approaches are scalable across multiple router RTLs. We now give a brief background on four widely used non-parametric regression or metamodeling techniques.

Metamodeling techniques can be broadly classified [11] into linear regression-based methods, interpolation-based methods, neural network and kernel-based smoothing methods, and additive tree-based methods. Popular techniques for estimation

[5]Use of hierarchical synthesis in general leads to lower instance counts, 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.

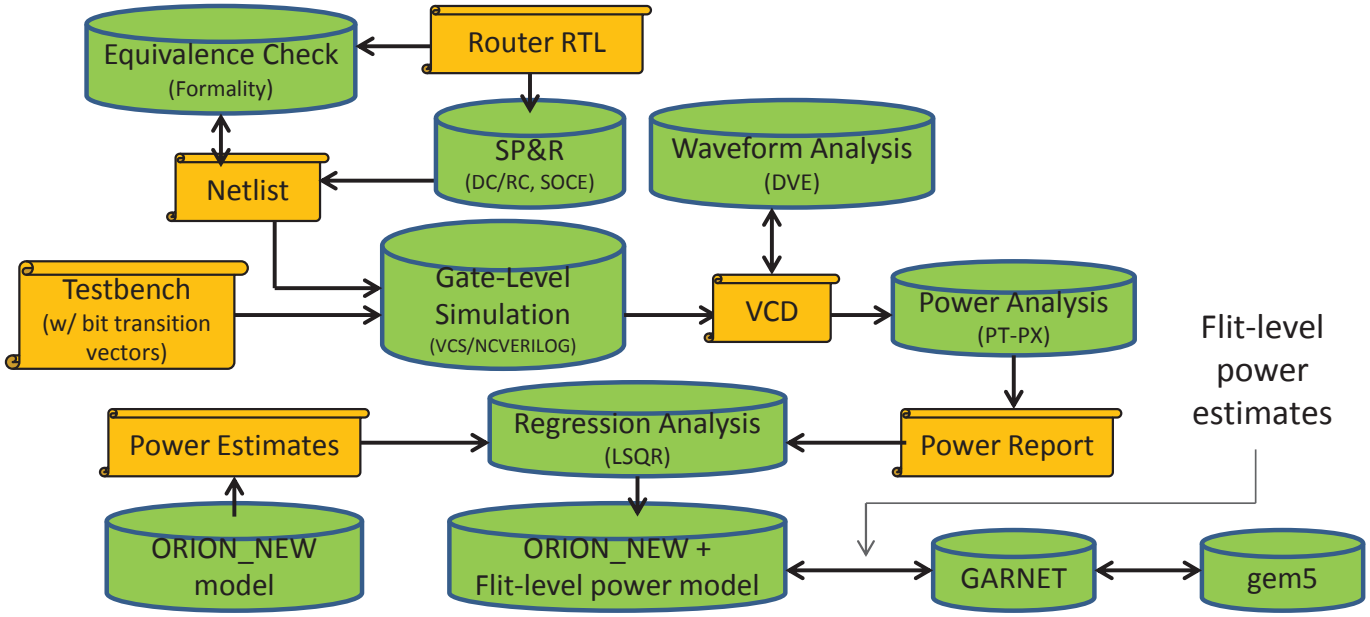

Fig. 6. Methodology to enhance ORION_NEW dynamic power models with flit-level power estimation.

purposes are Artificial Neural Networks (ANN), k-Nearest Neighbors (kNN), b-Splines, Radial Basis Functions (RBF), Kriging (KG), Polynomial Regression (PR), Support Vector Machine Regression (SVM) and Multivariate Adaptive Regression Splines (MARS) [45], [15], [44]. Recent [45] as well as previous studies [15] show that RBF, KG, and MARS models have higher estimation accuracy than others. Jin et al. [15] show that RBF excels in accuracy and robustness for *sparse* as well as *dense* training data points whereas MARS and KG have high accuracy for dense training sets.[6]

A. Radial Basis Functions (RBF)

Radial Basis Functions are hidden unit transfer functions in a linear neural network. The model uses a series of basis functions which are symmetric and centered at each training data point. Their response decreases monotonically and smoothly with distance from a central point [33]. The distance between two points is determined by the difference of coordinates and a set of scaling parameters. RBFs form a bridge between modern kernel methods and older fitting technologies [11]. The RBF model is represented as

$$\hat{y}(x) = \mu + \sum_{i=1}^{N} a_i \phi\left(\frac{\|x - x_i\|}{r_i^2}\right) \tag{16}$$

where where $\hat{y}(x)$ is the predicted response, N is the number of training data points, r_i are the scaling parameters, x_i is the center, μ is either a polynomial model or a constant value, ϕ is the basis function, and a_i are the coefficients of the basis functions calculated by solving a linear system. Popular choices for basis functions are shown in Table IV.

TABLE IV
RBF: BASIS FUNCTIONS. (c = *center*, r = *scale parameter*)

Name	Function	Equation
GAUSS	$\phi(x,c,r)$	$e^{\left(\frac{-(x-c)^2}{r^2}\right)}$
CUBIC	$\phi(x,c)$	$(x-c)^3$
CAUCHY	$\phi(x,c)$	$\frac{1}{1+(x-c)}$
MULTIQUADRIC	$\phi(x,c,r)$	$\frac{\sqrt{(x-c)^2+r^2}}{r}$

The RBF model works in two phases – (1) the *forward selection* phase, which starts with an empty subset, adds one basis function at a time, and minimizes the sum of squared error until a stopping criterion is reached, and (2) the *backward selection* phase, which starts with a full subset, removes one basis function at a time, and minimizes the sum of squared error. A generalized cross validation (GCV) mechanism is used to make the model generalizable so that it minimizes estimation errors on future input.

In our work, we use an academic RBF *MATLAB* toolbox, *rbf2* [60]. We use global ridge regression with Gaussian and multiquadric basis functions with GCV for generalization. Gaussian basis functions are chosen when the training set is sparse.

[6]We define a training set as *sparse* if it contains at most 20% of the total data points. We define a training set as *dense* if it contains at least 30% of the total data points.

The scaling parameter (r) also depends on the size of the training set. When the training set is dense, r is chosen to be less than 1, whereas when the training set is sparse, $0.5 \leq r \leq 2$.

B. Kriging (KG)

Kriging is a special kind of interpolation function that uses correlation between neighboring points in the training data points to estimate the response at some other arbitrary point. Kriging models a deterministic response as a realization of a stochastic process[7]

$$\hat{y}(x) = f(\beta, x) + \varepsilon(x) \tag{17}$$

where $f(\beta, x)$ expresses the deterministic part of the response, and $\varepsilon(x)$ is a random function which expresses the deviation of $\hat{y}(x)$ from the actual response y. $f(\beta, x)$ is a linear combination of k known functions and is a realization of a regression model given by

$$f(\beta, x) = \beta_1 f_1(x) + \beta_2 f_2(x) + \cdots + \beta_k f_k(x) \tag{18}$$

where the coefficients β_i, $i = 1, .., k$ are regression parameters or weights.

Once the Kriging model is built, it is used to predict response at arbitrary points which are not in the training data points. In general, a linear predictor is used [24], i.e.,

$$\hat{y}(x) = f(x)^T \beta^* + r(x)^T \gamma^* \tag{19}$$

where β^* and γ^* are weights determined from the training data points, $f(x)^T$ is the regression model, and $r(x)^T$ is the correlation model. Linear forms of the regression model are widely used [35] and can be polynomials of order 0 (constant), 1 (linear) or 2 (quadratic). Widely used correlation models are one-dimensional [24] and take the form

$$R(\theta, w, x) = \prod_{j=1}^{N} R_j(\theta, d_j) \tag{20}$$

where N is the size of the set of training data points, and $d_j = w_j - x_j$ is the deviation between the neighboring points. Some common choices of $R_j(\theta, d_j)$ are shown in Table V. In all these models the correlation decreases as $|d_j|$ decreases, and larger values of θ lead to faster decrease.

TABLE V
KRIGING: CORRELATION MODELS

Function Name	$R_j(\theta, d_j)$	
EXP	$e^{-\theta\|d_j\|}$	
GAUSS	$e^{-\theta d_j^2}$	
LIN	$\max\{0,\ 1-\theta\|d_j\|\}$	
SPHERICAL	$1-1.5\xi_j+0.5\xi_j$,	$\xi_j = \min\{1,\ \theta\|d_j\|\}$
CUBIC	$1-3\xi_j^2+2\xi_j^3$,	$\xi_j = \min\{1,\ \theta\|d_j\|\}$
SPLINE	$1-15\xi_j^2+30\xi_j^3$,	if $0 \leq \xi_j \leq 0.2$
	$1.25(1-\xi_j)^3$,	if $0.2 < \xi_j < 1$
	0,	if $\xi_j \geq 1$

In our work, we use the KG *DACE* [24] toolbox for *MATLAB* [57]. We choose first-order polynomials for the regression model when the training data points are sparse. Higher-order polynomials do not result in a good fit in such cases. When the training set is dense, we use second-order polynomials for regression. We have tried both EXP and GAUSS correlation models, as shown in Table V, and have found that EXP gives a fit with smaller *Root Mean Square Error* (RMSE), *Magnitude of Mean Error* (MME) as well as *Maximum Error*. Area and power are not smooth functions for higher-order differentials [24] and exhibit a linear behavior near the origin. The EXP models also have this behavior, whereas GAUSS is parabolic near the origin. We make an initial guess on θ to be 40 and then allow the model to change it during the fitting process.

C. Multivariate Adaptive Regression Splines (MARS)

Multivariate Adaptive Regression Splines is a flexible regression modeling technique for high-dimensional input data, that is, for a large number of input variables [11]. The model is a product of spline basis functions, and is constructed using an iterative forward and backward approach. The number of basis functions, product degree, and knot[8] locations are determined automatically using the training data points during iterative steps. The MARS model is represented as a sum of basis functions

$$\hat{y}(x) = c_0 + \sum_{i=1}^{I} c_i f_i(x) \tag{21}$$

[7]Least squares regression (LSQR) fit assumes the residue to have Gaussian distribution with zero mean and constant variance, whereas Kriging models the residue as a stochastic process [21]. This allows Kriging to reduce the estimation error better than LSQR.

[8]Knot is the value of an input parameter that causes a piecewise line segment to changes its slope.

where I is the number of basis functions besides the constant basis function $f_0(x) = 1$, c_0 is the coefficient of the constant basis function, and c_i are the coefficients of each of the I basis functions. These basis functions are of the form

$$f_i(x) = \prod_{j=1}^{J_i} \left[b_{ji} \left(x_{v(j,i)} - t_{ji} \right) \right]_+ \tag{22}$$

where J is the number of input variables, J_i is the number of interactions between variables in the i^{th} basis function, $b_{ji} = \pm 1$, $x_{v(j,i)}$ is the v^{th} input variable, $1 \leq v(j,i) \leq J$, t_{ji} is the knot location on each of the corresponding variables, and the subscript "+" denotes the positive part of a truncated power function.

MARS works in two passes. The *forward pass* starts with a knot, then basis functions are repeatedly added in pairs to the model until the number of added basis functions reaches I, which is an input to the model. The forward pass usually results in an overfit, so it is followed by a *backward pass* that prunes the model with better ability to generalize using a GCV mechanism [8], given by

$$GCV(M) = \frac{1}{N} \frac{\sum_{p=1}^{N} [y_p - \hat{y}(x_p)]^2}{\left[1 - \frac{C(M)}{N}\right]^2} \tag{23}$$

where N is the number of training data points, y_p is the actual response, $\hat{y}(x_p)$ is the predicted response for the input x_p, M is the number of non-constant basis functions, and $C(M)$ is the complexity penalty to avoid overfitting and is typically the number of parameters being fitted. The GCV criterion is a mean squared residual fit to the training data times a penalty to account for the increased variance associated with increasing model complexity.

In our work, we use an academic MARS *MATLAB* toolbox, *ARESLab* [58].[9] We use both cubic as well as linear splines to fit the training data points. As indicated in Table VI below, for all sizes of the training data set, we set the maximum number of basis functions to 50 and the maximum degree of interactions between input variables to three. There are four variables (*P, V, B, F*) in the input, and the maximum degree of interactions is typically chosen to be one less than the number of input variables [14].

D. Support Vector Machine Regression (SVM)

Regression based on Support Vector Machine is an adaptation of the popular SVM classification model [11], [39], [10]. The objective is to minimize a loss function such as, Huber, ε-insensitive, or quadratic. The SVM model can be represented as [11]

$$\hat{y}(x) = c_0 + \sum_{i=1}^{M} c_i h_i(x) \tag{24}$$

where c_0 and c_i are coefficients to be determined by minimizing the loss function, h_i are the basis functions, and M is the maximum number of basis functions added by the kernel method. We use the Huber loss function [10]

$$L_{Huber}(\hat{y}(x) - y) = \begin{cases} \frac{1}{2}(\hat{y}(x) - y)^2 & \text{if } |\hat{y}(x) - y| \leq \mu \\ \mu|\hat{y}(x) - y| - \frac{\mu^2}{2} & \text{otherwise} \end{cases} \tag{25}$$

where μ is a pre-chosen error bound. To determine the coefficients of the basis functions, the following function is minimized

$$min. \sum_{i=1}^{M} L_{Huber}(\hat{y}(x) - y) + \frac{C}{2}\|c_i\| \tag{26}$$

where C is a pre-chosen cost penalty.

In our work, we use the *LIBSVM* [61] toolbox for *MATLAB*. We use the Huber loss function as the optimal regression function to minimize the mean squared error of the model. We set the cost parameter as the maximum difference of the responses in the training data points, and we use RBF as the kernel method to generate basis functions as shown in Table VI.

VI. METAMODELING METHODOLOGY

We now describe how we use the metamodeling techniques described in Section V to estimate NoC area and power. We first perform synthesis using Synopsys *Design Compiler* [46], followed by place and route using Cadence *SOC Encounter* [50], of the *Netmaker* [55] router RTL. Next, we generate area and power reports of these designs to use as training and test data points. Figure 7 shows our synthesis and P&R flow, Table I lists the architecture, implementation, and operational parameters, and Table VI lists the tool options used in our experiments. We generate 256 data points for our experiments. We use two sampling methodologies to generate the training sets – a modified Latin Hypercube Sampling (LHS) [15], and a restricted sampling methodology which samples only values from the lower ranges of the parameters (*B, V, P, F*). Our LHS methodology (for 64 data points of four variables) is as follows.

[9]We compared results of *ARESLAB* with a commercial MARS tool from Salford Systems [59] and found that estimation errors of both these tools are similar.

1) Generate 64 normalized LHS samples over four parameters using the *MATLAB* command *lhsdesign(64, 4)*.
2) Maximize the minimum distance between samples by using the *MATLAB* command *bsxfun* [62].
3) Map the samples generated in the previous step to our ranges of *B, V, P* and *F* parameters by selecting the value which is closest to the sample.
4) Adjust the frequency of the values to make the samples uniformly distributed across our range of values for *B, V, P* and *F* so that each of them occurs the same number of times in the training set.

The restricted sampling methodology does not include higher values of the microarchitectural parameters in the training set. More precisely, the resulting training sets omit all values of $\{B{=}7\}$, or of $\{P = 9\}$, or of $\{V = 7\}$, or of $\{F = 64\}$.

Unlike previous approaches using MARS [17], we model leakage, internal, and switching power separately. This results in more accurate fit of the training data because each of these components of power does not change in the same fashion with microarchitectural, implementation, and operational parameters. Figure 8 shows the flow of our methodology.

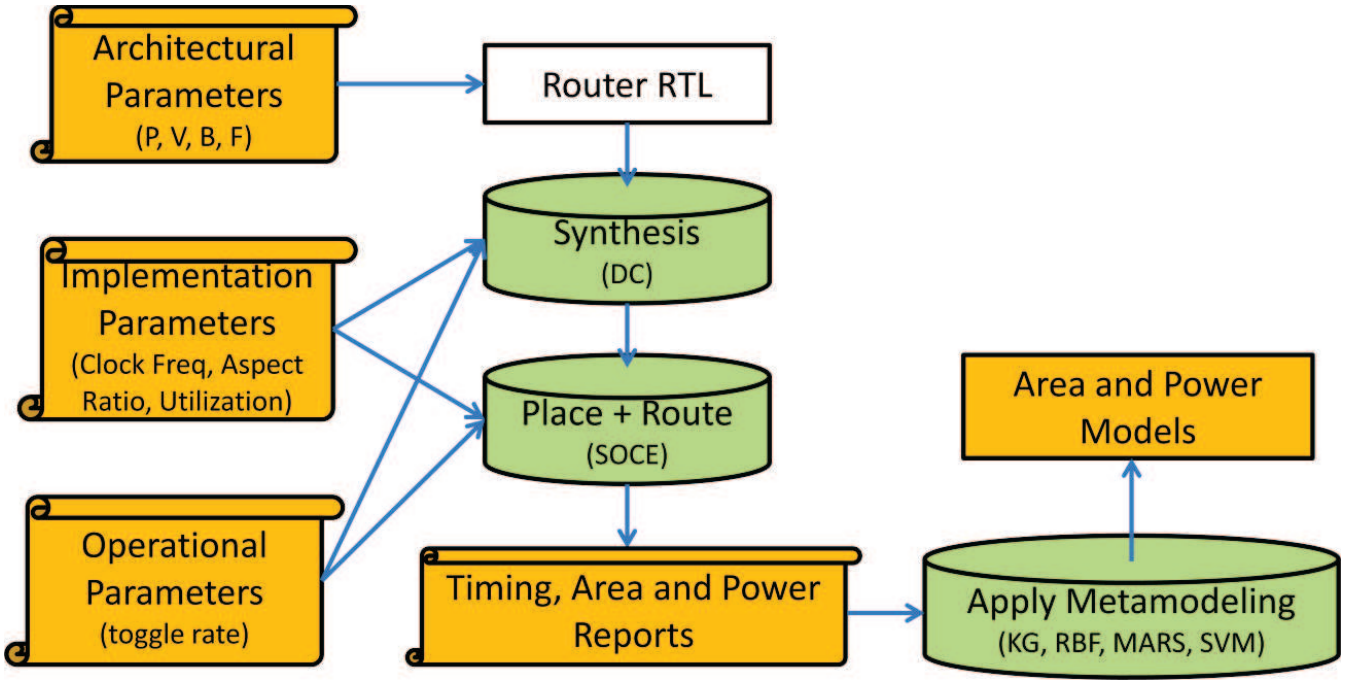

Fig. 7. Implementation flow to generate training and test data points.

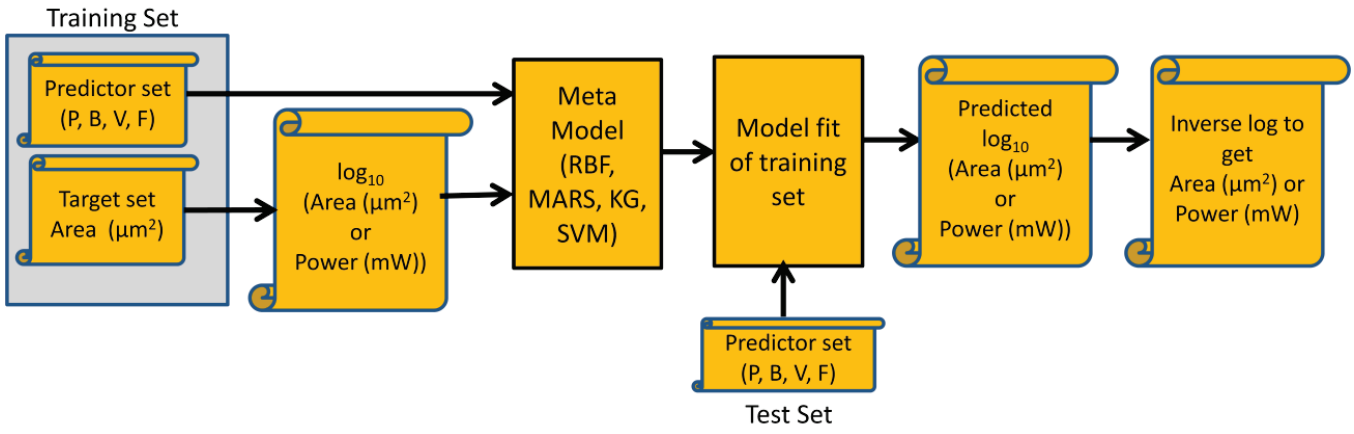

Fig. 8. Area and power modeling and prediction flow.

We assess the goodness of fit of RBF, KG, MARS and SVM models using three metrics.

- *Magnitude of Mean Error* (MME): This is the mean of the magnitude of errors at each predicted output using the test data points.
- *Root Mean Square Error* (RMSE): This is the square root of the mean of the sum of squared error for the predicted outputs using the test data points.

TABLE VI
METAMODELING METHODOLOGY: TOOLS AND PARAMETERS

Stage	Tool	Values
LHS	*MATLAB*	lhsdesign bsxfun
KG	*DACE*	Reg model = {Order 1 and 2 Poly} Corr model = {EXP, GAUSS}, θ = {40}
RBF	*RBF2*	Type = {Ridge Regression} Func. Type = {GAUSS, MULTIQUADRIC} $2 \leq r \leq 0.5$
MARS	*ARESLAB*	Max Basis Funcs = {50} Max Interactions = {3} Spline Type = {linear, cubic}
SVM	*LIBSVM* v3.12	Type = {nu-SVR} Kernel = {RBF}

- *Maximum Error* (MAXE): This is the maximum of the magnitude of errors at each predicted output using the test data points. We include this metric to give a sense of the worst-case error, which is of practical concern for hardware designers and computer architects.

VII. VALIDATION AND RESULTS

In this section, we discuss our validation and results of (1) ORION_NEW, (2) ORION_NEW with parametric regression, and (3) metamodeling techniques. We calculate the magnitude of percentage estimation error as $Error\% = ABS((TOOL - MODEL) / MODEL$

A. Results of ORION_NEW

We set up experiments as described in Table I of Section IV. We use parameters and tools for our experiments as listed in Table I. We discuss the results in two parts, (1) ORION2.0 versus ORION_NEW estimation of area and power, and (2) impact of our regression analysis approach versus the approach used in prior work of [17]. Comparisons are made with respect to post-P&R instance counts, power and area outcomes, and both router RTL generators, *Netmaker* [55] and *Stanford NoC* [56].

1) ORION2.0 versus ORION_NEW Comparisons. Since the instance counts per component are 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 9(a), 10(a), and 11(a) show the large errors in ORION2.0 for crossbar, output buffer and input buffer respectively, and Figures 9(b), 10(b), and 11(b) show the significant reduction in estimation errors 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 for ORION2.0.

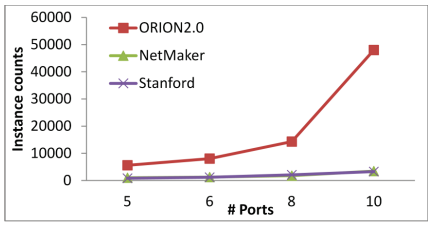

Fig. 9. (a) XBAR with #ports: ORION2.0 vs. implementation. (b) XBAR with #ports: ORION_NEW vs. implementation.

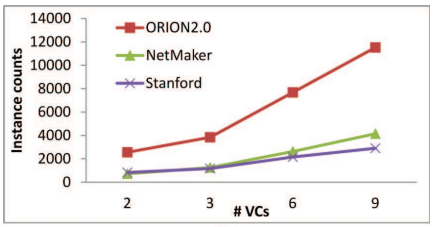

Fig. 10. (c) Output buffer with #VCs: ORION2.0 vs. implementation. (d) Output buffer with #VCs: ORION_NEW vs. implementation.

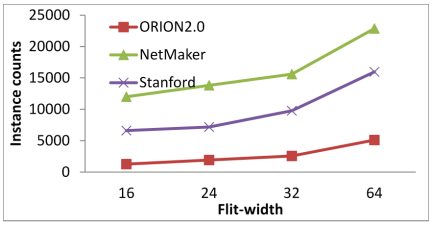

Fig. 11. (e) Input buffer with flit-width: ORION2.0 vs. implementation. (f) Input buffer with flit-width: ORION_NEW vs. implementation.

ORION2.0 modeling of instance counts for a component does not consider implementation parameters such as clock frequency. As a result, the instance counts do not scale when frequency is changed, even though at higher frequencies buffers are typically inserted to meet tight setup time constraints. Our ORION_NEW models apply a frequency derating factor to the instance models for component blocks as described in Section III-F. Figures 12(a) and (b) respectively show results for output and input buffer component blocks; the incorrect estimates by ORION2.0 contrast sharply with the estimates from ORION_NEW, which are very close to actual implementation.

Table 13 summarizes ORION2.0 and ORION_NEW estimation errors with respect to *Netmaker* and *Stanford NoC* post-P&R area. Higher error values are highlighted in red. Figures 14(a) and (b) plot the estimation errors for power and area respectively

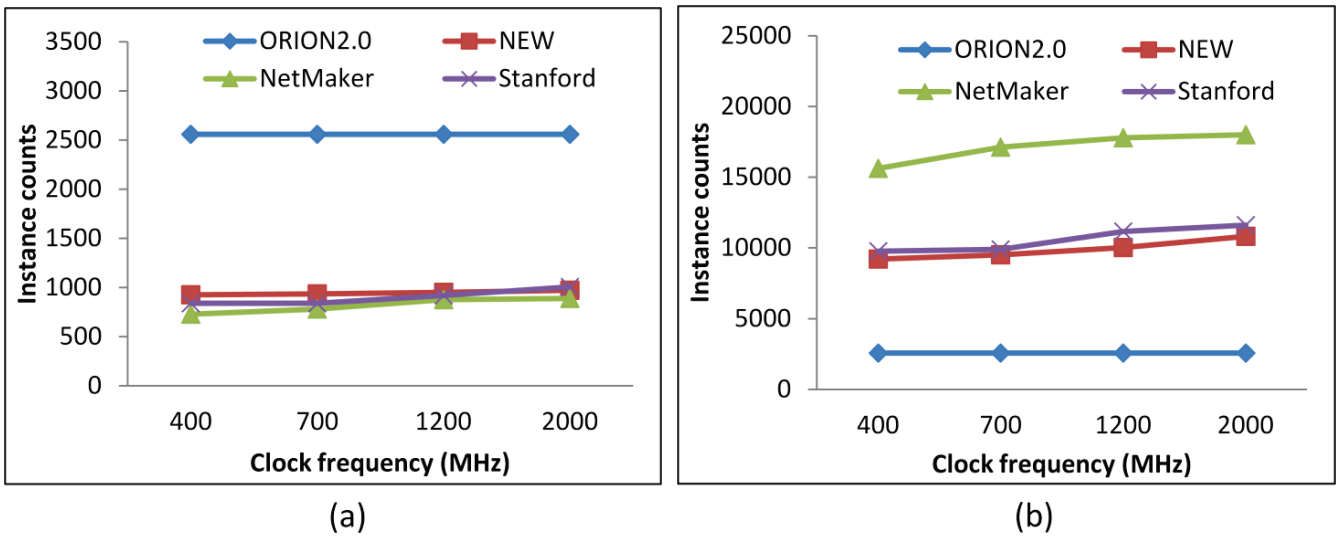

Fig. 12. (a) Output buffer with clock frequency: ORION2.0 vs. ORION_NEW. (b) Input buffer with clock frequency: ORION2.0 vs. ORION_NEW.

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%

Fig. 13. Instance counts and area error comparison of ORION2.0 vs. ORION_NEW.

at 45nm and 65nm technology nodes after applying the regression fitting approach described in Section IV-B. We see that ORION_NEW estimates are very close to actual implementation (average error of 9.8% in estimating *Netmaker* power at 45nm) and are robust across multiple microarchitecture, implementation parameters, and router RTLs.

Next, we analyze the impact of flit-level power modeling as described in Section IV-C. To capture the effect of running simulations with input vectors having different bit encodings (shown in Figure 6), we use options in Synopsys *PrimeTime-PX* [47] to vary toggle rates and bit encodings in the input. We run simulations using four different toggle rates (0.2, 0.4, 0.6, and 0.8) and four different encodings of 1's in 32-bit input flits. We 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. Figure 15 compares dynamic power estimation error of ORION2.0, ORION_NEW, and ORION_NEW with flit-level power models. We observe that by using flit-level power models, average dynamic power estimation error can be within 12%.

2) Impact of our regression analysis approach. In Section IV-B, we describe our parametric regression analysis approach using the ORION_NEW models. As seen from the above results (Section VII-A1), the ORION_NEW models are accurate across microarchitecture, implementation, and operational parameters. With these accurate models, regression analysis can minimize errors and generate accurate fitting coefficients. The previous parametric regression approach [17] reports large errors because underlying ORION2.0 models do not model control path elements. The non-parametric regression approach of [17] using MARS achieves reduced average power modeling errors of 5.82% at 65nm and 5.65% at 90nm, and reduced average area errors of 5.41% at 65nm and 5.01% at 90nm. In our work, we use parametric regression analysis but with accurate ORION_NEW models. Our average errors are similar to [17]; however, our maximum error for power (resp. area) is reduced from 59.41% to 24.42% (resp. from 61.84% to 30.30%) at 65nm. At 90nm the reduction of maximum power (resp. area) error is from 60.15% to 28.04% (resp. from 60.07% to 19.36%). The reduction of maximum estimation error is significant because NoC designers and architects care about worst-case accuracy.

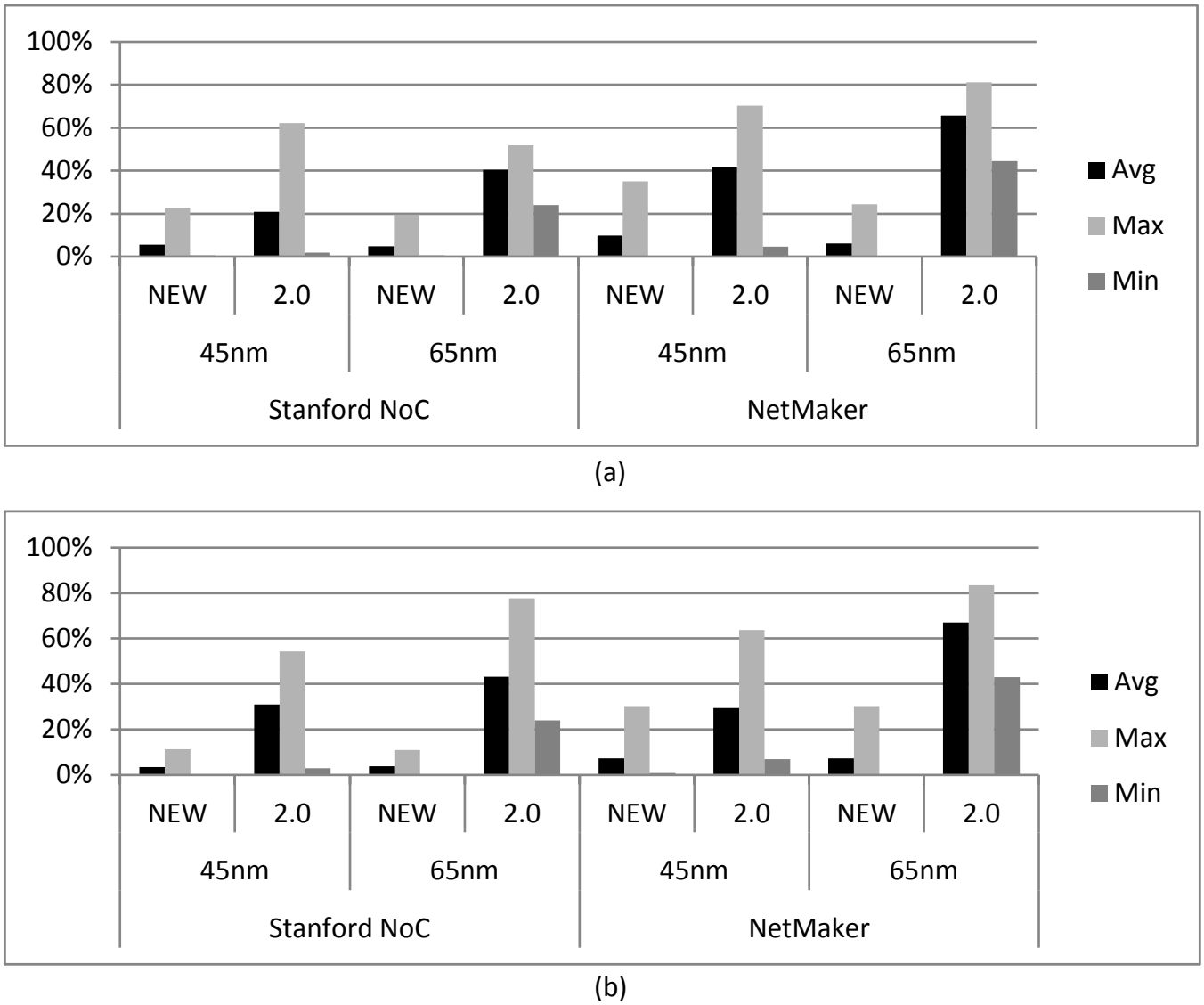

Fig. 14. ORION_NEW with regression fit vs. ORION2.0: (a) power estimation error and (b) area estimation error.

B. Results of Metamodeling Techniques

We set up experiments for each of the metamodeling techniques using the parameters, tools, and methodology as described in Table VI and Section VI. We generate 256 data points of post-P&R power and area values using 45nm and 65nm technology libraries. 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 the sampling methodology described in Section VI to generate training sets of three sizes.

- 50 data points – sparse and restricted,
- 64 data points – sparse, and
- 102 data points – dense.

We use the sparse and restricted training set to test the accuracy of models 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 3s and response estimation takes around 1.88s. We repeat all experiments 10 times for each training set size, and report the averages of all the error values across the 10 trials.

We present the results of metamodeling techniques as (1) comparisons among the techniques used, (2) comparisons against MARS with linear splines [17], and (3) comparisons against parametric regression techniques [16].

1) Metamodeling accuracy. Figures 16(a) and (b) show the percentage errors observed in estimating standard-cell area at 65nm and 45nm. With a dense training set, RBF, KG and MARS have similar maximum estimation errors of around 20%. SVM, on the other hand, has maximum estimation errors of 37.8% at 45nm and 25% at 65nm. The average estimation errors of all these models are less than 10.7%, with SVM having higher average estimation error than RBF, KG and MARS. With a sparse training set (64 data points), RBF and KG have higher accuracies than MARS and SVM. RBF always performs better than KG, MARS, and SVM with a sparse and restricted training set (50 data points). Figure 16(b) shows that with a sparse and

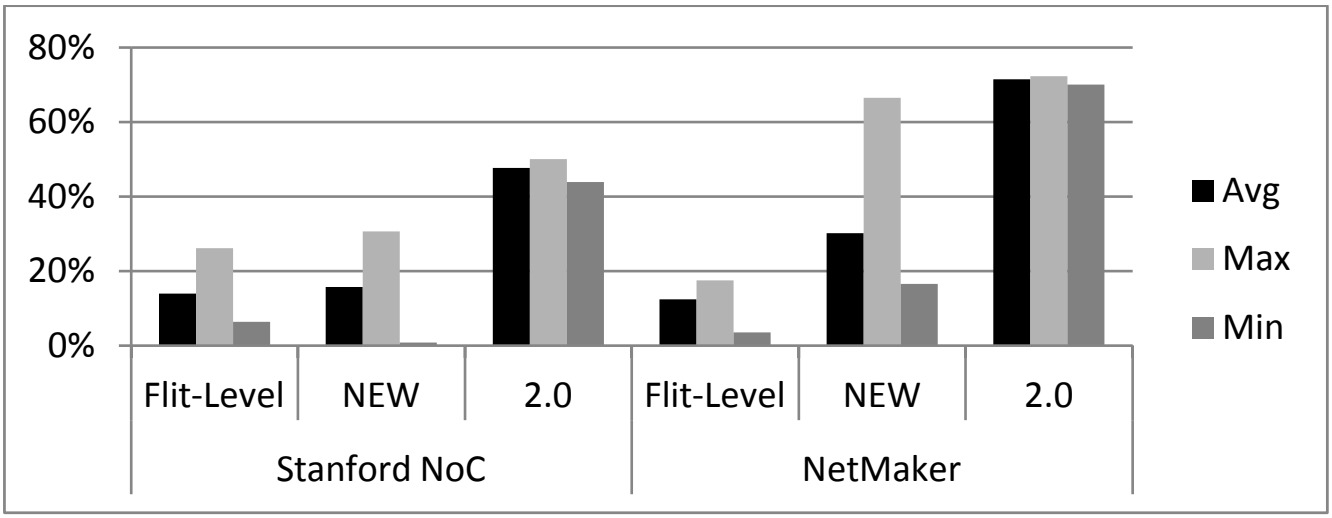

Fig. 15. Comparison of dynamic power estimation error using (1) ORION2.0, (2) ORION_NEW, and (3) ORION_NEW with flit-level power models.

restricted training set the maximum estimation error is less than 12.8% for RBF, whereas the maximum estimation errors are more than 32% for KG, MARS, and SVM. The accuracies of these models in estimating power are similar to their accuracies in estimating area, as shown in Figures 17(a) and (b). With a sparse and restricted training set, RBF can be three times more accurate than KG, SVM and MARS. RBF and KG have similar errors for sparse as well as dense training sets. Across all training set sizes used in our experiments, we observe that area and power estimation errors are the smallest for RBF and are the highest for SVM.

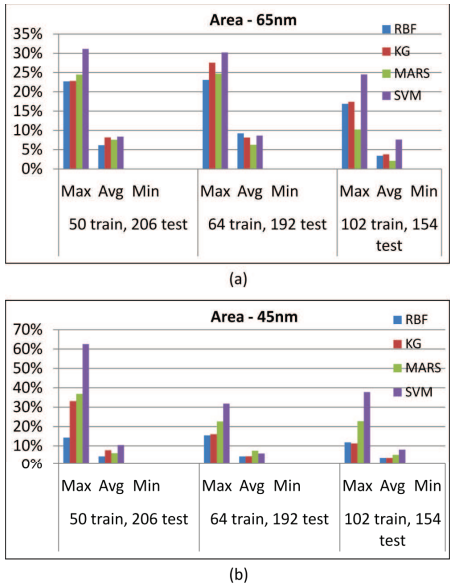

Fig. 16. Area estimation accuracy of metamodeling techniques at (a) 65nm and (b) 45nm.

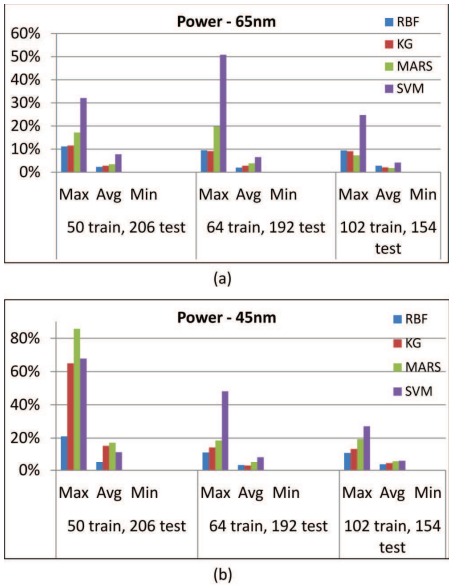

Fig. 17. Power estimation accuracy of metamodeling techniques at (a) 65nm and (b) 45nm.

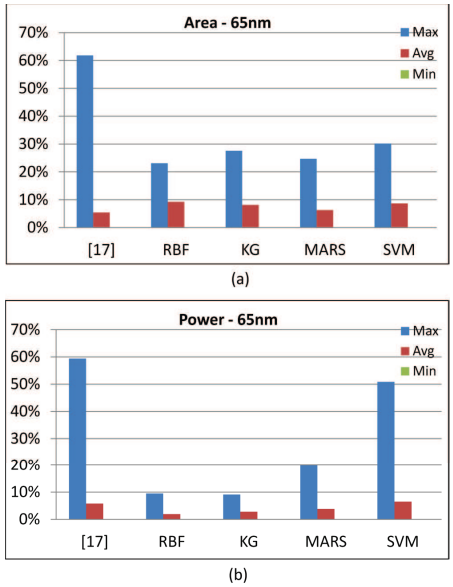

Fig. 18. Comparison with estimation errors of [17] at 65nm: (a) area, and (b) power. Minimum errors are too small to appear in the plot.

2) Comparison of MARS linear and cubic splines. Prior work in [17] uses MARS with linear splines to model NoC area and power, and reports maximum estimation errors of around 60% at 65nm. We use MARS with cubic splines in our experiments. Figures 18(a) and (b) compare area and power at 65nm with our metamodeling techniques. In general, our maximum estimation errors are smaller than those of [17] across all the techniques. In particular, our estimation errors (maximum, average and minimum) for MARS are smaller than in [17]; this is because cubic splines are better than linear splines in minimizing estimation errors. Figures 19(a) and (b) show that cubic splines perform better than linear splines across different technologies and training set sizes. With a sparse training set at 65nm, the maximum area (resp. power) estimation error is 24.8% (resp. 19.7%) with cubic splines, whereas it is 33.6% (resp. 28.3%) with linear splines.

3) Comparison with parametric regression. We use a sparse training set of 64 data points to estimate area and power using the parametric LSQR technique used to fit parametric models of router component blocks in [16]. Figures 20(a) and (b) compare

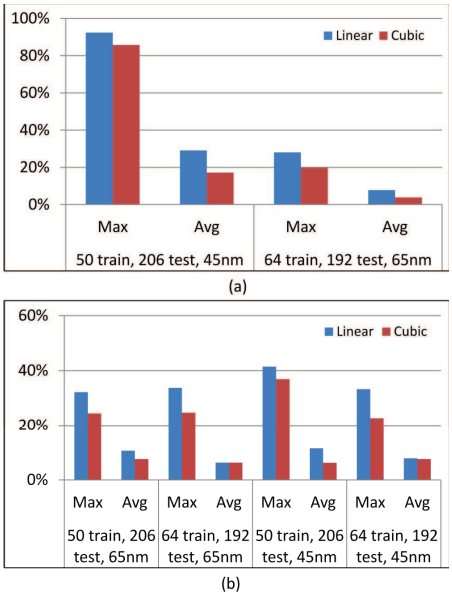

Fig. 19. MARS linear vs. cubic splines: (a) power and (b) area.

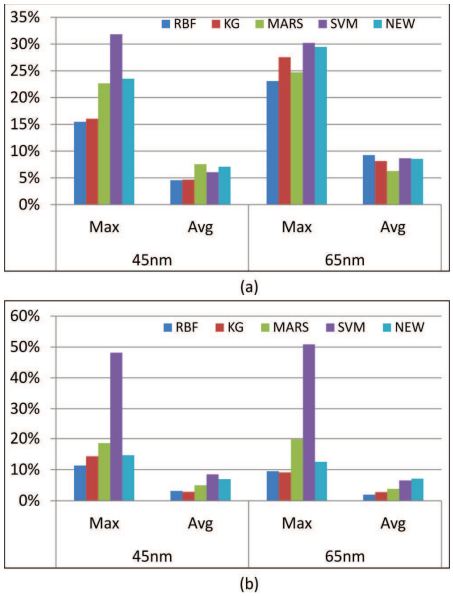

Fig. 20. Comparison of estimation errors of non-parametric vs. parametric regression techniques: (a) area and (b) power.

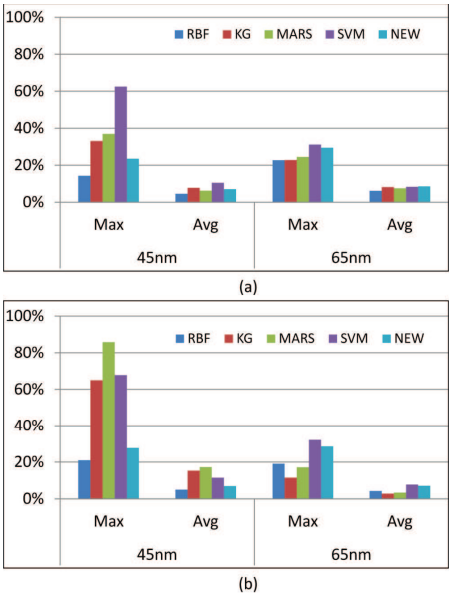

Fig. 21. Comparison of estimation error in flit-level power modeling at 65nm: (a) maximum and (b) average.

the maximum and average estimation errors of the metamodeling techniques with estimation errors of parametric models fitted using LSQR (NEW in the figures). We observe that the maximum area estimation error for NEW is similar to the maximum area estimation error for RBF at 45nm, whereas the average area estimation error for NEW is similar to the average area estimation errors for RBF and KG at both 65nm and 45nm. The maximum estimation error for NEW at 45nm is smaller than the maximum estimation errors of KG, MARS and SVM for both area and power. At 65nm, the maximum power estimation error for NEW is slightly higher than the maximum power estimation errors for RBF, KG and MARS. In general, we observe that the estimation errors for NEW and RBF are smaller than those for KG, MARS, and SVM. The average estimation error for NEW is less than 8% and its maximum estimation error is less than 25% at both 65nm and 45nm.

Figures 21(a) and (b) compare the maximum and average flit-level power estimation errors of metamodeling techniques with ORION_NEW flit-level power models fitted with post-P&R dynamic power using parametric regression (NEW in the figures). We use a sparse and restricted training set of 50 data points for all the modeling techniques in this experiment. We observe that NEW is more accurate than all the metamodeling techniques. This is because flit-level power estimation requires fine-grained modeling of the component blocks in a router. Fine-grained modeling allows accurate estimation of dynamic power dissipation in each component block for different bit encodings in flits. The metamodeling techniques cannot estimate flit-wise power dissipation of each component block because they fit the training data points by treating the router as a black box. Therefore, estimation errors are large when fitting data points with different flit-level bit encodings. We observe that both maximum and average estimation errors for SVM, KG, and MARS are significantly higher ($> 80\%$) than those for NEW (17.6%). RBF is the only technique whose average estimation error is similar to the average estimation error for NEW.

VIII. 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. 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. In this work, we propose comprehensive parametric and non-parametric modeling techniques to accurately estimate NoC power and area. Our parametric models, ORION_NEW, explicitly account for control and data path resources. We propose a new methodology that we used to develop the ORION_NEW parametric models from post-synthesis netlists. We further refine these parametric models by performing least-squares regression analysis (LSQR) on post-P&R data. We demonstrate that accurate parametric models and LSQR can reduce the worst-case estimation errors by more than 50% as compared to previous non-parametric regression models for NoC routers [17]. We are also the first to propose detailed flit-level power estimation models that can seamlessly integrate with full-system NoC simulators such as *GARNET*.

For non-parametric regression (or metamodeling), we use four popular techniques – RBF, KG, MARS, and SVM. Our results show that these techniques can be low-overhead and highly accurate in estimating NoC power and area. We describe two methodologies to generate training sets to test the accuracy and robustness of these techniques. Among these four techniques, RBF proved to be the most accurate and robust across technologies and training set sizes. However, these techniques are not accurate for detailed flit-level power modeling because they cannot model flit-level power dissipation in each component block of the router. The ORION_NEW model fitted with post-P&R dynamic power using parametric regression provides more accurate estimates of flit-level dynamic power compared to these metamodeling techniques.

We validate robustness of our modeling methodologies across multiple router RTLs, and across microarchitecture, implementation, and operational parameters. We conclude that our modeling methodologies are highly accurate with average errors $\leq$ 9.8%. Implementations of our modeling methodologies are being made available for download in an ORION3.0 distribution [19], [66]. We plan to extend our work to accurately model link power by incorporating link signaling elements such as differential signaling, scrambling, serdes, equalization and 3D routing.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (NSF) under grant number SHF-1116667. The authors acknowledge the support of the Gigascale Systems Research Center (GSRC), one of the six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

REFERENCES

[1] N. Agarwal, T. Krishna, L.-S. Peh and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator", *Proc. ISPASS*, 2009, pp. 33-42.
[2] A. Banerjee, R. Mullins and S. Moore, "A power and energy exploration of network-on-chip architecture", *Proc. NOCS*, 2007, pp. 163-172.
[3] N. Banerjee, P. Vellanki and K. S. Chatha, "A power and performance model for network-on-chip architectures", *Proc. DATE*, 2004, pp. 1250-1255.
[4] J. Chan and S. Parameswaran, "NoCEE: Energy macro-model extraction methodology for network-on-chip routers", *Proc. ICCAD*, 2005, pp. 254-259.
[5] 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.
[6] W. J. Dally and B. Towles, "Route packets not wires: On-chip interconnection networks", *Proc. DAC*, 2001, pp. 684-689.
[7] W. J. Dally and B. Towles, *Principles and practices of interconnection networks*, Morgan Kaufmann, 2004.
[8] J. H. Friedman, "Multivariate adaptive regression splines", *The Annals of Statistics* 19(1) (1991), pp. 1-141.
[9] G. Guindani, C. Reinbrecht, T. Raupp, N. Calazans and F. G. Moraes, "NoC power estimation at the RTL abstraction level", *Proc. ASVLSI*, 2008, pp. 475-478.
[10] S. R. Gunn, "Support vector machines for classification and regression", *University of Southampton Technical Report*, 1998.
[11] T. Hastie, R. Tibshirani and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, Springer, 2009.
[12] A. A. Ilumoka, "Efficient prediction of crosstalk in VLSI interconnects using neural networks", *Proc. EPEP*, 2000, pp. 87-90.
[13] E. Ipek, S. A. McKee, B. R. de Supinski, M. Schulz and R. Caruana "Efficiently exploring architectural design spaces via predictive modeling", *Proc. ASPLOS*, 2006, pp. 195-206.
[14] 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.
[15] R. Jin, W. Chen and T. W. Simpson, "Comparative studies of metamodeling techniques under multiple modeling criteria", *Trans. Struct. Multidiscip. Optim.* 23 (2001), pp. 1-13.
[16] A. B. Kahng, B. Lin and S. Nath, "Explicit modeling of control and data for improved NoC router estimation" *Proc. DAC*, 2012, pp. 392-397.
[17] A. B. Kahng, B. Lin and K. Samadi, "Improved on-chip router analytical power and area modeling" *Proc. ASPDAC*, 2010, pp. 241-246.
[18] 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.
[19] A. B. Kahng, B. Lin and S. Nath, "Comprehensive modeling methodologies for NoC router estimation", *University of California San Diego Technical Report CS2012-0989*, 2012, pp. 1-14.
[20] T. Kiely and G. Gielen, "Performance modeling of analog integrated circuits using least-squares support vector machines", *Proc. DATE*, 2004, pp. 448-453.
[21] F. Liu, "A general framework for spatial correlation modeling in VLSI design", *Proc. DAC*, 2007, pp. 817-822.
[22] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction", *Proc. ASPLOS*, 2006, pp. 185-194.
[23] 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.
[24] S. N. Lophaven, H. B. Nielsen and J. Sondergaard, "Aspects of the MATLAB toolbox DACE", *Technical University of Denmark Technical Report IMM-REP-2002-13*, 2002.
[25] G. Mariani, A. Brankovic, G. Palermo, J. Jovic, V. Zaccaria and C. Silvano, "A correlation-based design space exploring methodology for multi-processor systems-on-chip", *Proc. DAC*, 2010, pp. 120-125.
[26] G. Mariani, G. Palermo, V. Zaccaria and C. Silvano, "OSCAR: An optimization methodology exploiting spatial correlation in multicore design spaces", *IEEE Trans. CAD* 31(5) (2012), pp. 740-753.
[27] 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.
[28] R. Mullins, A. West and S. Moore, "The design and implementation of a low-latency on-chip network", *Proc. ASPDAC*, 2006, pp. 164-169.
[29] G. Palermo and C. Silvano, "PIRATE: A framework for power/performance exploration of network-on-chip architectures", *Proc. PATMOS*, 2004, pp. 521-531.
[30] C. S. Patel, S. M. Chai, S. Yalamanchili and D. E. Schimmel, "Power constrained design of multiprocessor interconnection networks", *Proc. ICCD*, 1997, pp. 408-416.
[31] L.-S. Peh, "Flow control and micro-architectural mechanisms for extending the performance of interconnection networks" *PhD Thesis*, Stanford University, 2001.
[32] S. Penolazzi and A. Jantsch, "A high level power model for the Nostrum NoC", *Proc. Digital System Design*, 2006, pp. 673-676.
[33] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan and P. K. Tucker, "Surrogate-based analysis and optimization", *Progress in Aerospace Sciences*, 41 (2005), pp. 1-28.
[34] S. Reda and S. R. Nassif, "Accurate spatial estimation and decomposition techniques for variability characterization", *IEEE Trans. Semiconductor Manufacturing* 23(3) (2010), pp. 345-357.
[35] J. Sacks, W. J. Welch, T. J. Mitchell and H. P. Wynn, "Design and analysis of computer experiments", *Trans. Statistical Science* 4(4) (1989), pp. 409-435.
[36] E. S. Siah, M. Sasena, J. L. Volakis, P. Y. Papalambros and R. W. Wiese, "Fast parameter optimization of large-scale electromagnetic objects using DIRECT with Kriging metamodeling", *IEEE. Trans. Microwave Theory and Techniques* 52(1) (2004), pp. 276-285.
[37] T. W. Simpson, J. D. Peplinki, P. N. Koch and J. K. Allen, "Metamodels for computer-based engineering design: Survey and recommendations", *Trans. Engineering with Computers* 17 (2001), pp. 129-150.
[38] A. Singhee and R. A. Rutenbar, "Statistical Blockade: A novel method for very fast Monte Carlo simulation of rare circuit events, and its applications", *Proc. DATE*, 2007, pp. 1379-1384.
[39] A. J. Smola and B. Scholkopf, "A tutorial on support vector regression", *Statistics and Computing*, 14(3) (2004), pp. 199-222.

[40] V. Vapnik, S. Golowich and A. J. Smola, "Support vector method for function approximation, regression estimation, and signal processing", *Advances in Neural Information Processing Systems*, 9 (1997), pp. 281-287.
[41] H.-S. Wang, L.-S. Peh and S. Malik, "Orion: A power-performance simulator for interconnection networks", *Proc. MICRO*, 2002, pp. 294-305.
[42] H. Wang, H. You and X. Jia, "Analysis on the effect of regression and correlation models on the accuracy of Kriging model for IC", *Proc. EDSSC*, 2009, pp. 266-269.
[43] 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.
[44] M. B. Yelten, T. Zhu, S. Koziel, P. D. Franzon and M. B. Steer, "Demystifying surrogate modeling for circuits and systems", *Circuits and Systems*, 12(1) (2012), pp. 45-63.
[45] N. Yosboonruang, A. Na-udom and J. Rungrattanaubol, "A comparison of prediction accuracy of statistical models for computer simulated experiments", *Proc. Intl. Conf. on Statistics and Applied Statistics*, 2010.
[46] Synopsys Design Compiler User Guide. *http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DCUltra/pages/default.aspx*
[47] Synopsys PrimeTime User Guide. *http://www.synopsys.com/Tools/Implementation/SignOff/PrimeTime/pages/default.aspx*
[48] Synopsys Formality User Guide. *http://www.synopsys.com/tools/verification/formalequivalence/pages/formality.aspx*
[49] Synopsys VCS and DVE User Guide. *http://www.synopsys.com/tools/verification/functionalverification/pages/vcs.aspx*
[50] Cadence SOC Encounter User Guide. *http://www.cadence.com/*
[51] SDC User's Guide. *http://www.actel.com/documents/SDC_AN.pdf*
[52] Cadence RTL Compiler User Guide. *http://www.cadence.com/products/ld/rtl_compiler/pages/default.aspx*
[53] Cadence SOC Encounter User Guide. *http://www.cadence.com/*
[54] Standard Parasitic Exchange Format. *http://www.edaboard.com/thread37705.html*
[55] Netmaker. *http://www-dyn.cl.cam.ac.uk/~rdm34/wiki*
[56] Stanford NoC. *https://nocs.stanford.edu/cgi-bin/trac.cgi*
[57] MATLAB. *http://www.mathworks.com/*
[58] ARESLab. *www.cs.rtu.lv/jekabsons/regression.html*
[59] MARS User Guide. *http://www.salfordsystems.com/mars.php*
[60] RBF2 Manual. *http://www.anc.ed.ac.uk/~mjo/rbf.html*
[61] LIBSVM. *http://www.csie.ntu.edu.tw/~cjlin/libsvm*
[62] *http://www.mathworks.com/matlabcentral/newsreader/view_thread/279955*
[63] Intel 80-core Report. *http://techresearch.intel.com/ProjectDetails.aspx?Id=151*
[64] IBM Blue Gene processor. *http://www.research.ibm.com/journal/rd49-23.html*
[65] Tilera TILE-Gx processor. *http://www.tilera.com/products*
[66] ORION3.0. *http://vlsicad.ucsd.edu/ORION3/*