Architectural-Level Prediction of Interconnect Wirelength and Fanout

Kwangok Jeong[‡], Andrew B. Kahng[†‡], and Kambiz Samadi[‡]

[†]CSE and [‡]ECE Departments, UC San Diego, La Jolla, CA
{kjeong,kambiz}@vlsicad.ucsd.edu, abk@cs.ucsd.edu

***Abstract*— This paper proposes accurate architectural-level interconnect wirelength and fan-out models. Existing models are based on Rent's rule and fail to capture the impact of microarchitectural and implementation parameters. Hence, significant deviation is observed when validated against implementation data, i.e., up to 79% (22%) in total wirelength (average fanout). Our proposed models both enable architectural-level prediction of interconnect wirelength and fanout, and show significant accuracy improvement vs. existing models with respect to layout data.**

I. Introduction

Clock frequency, power consumption, and chip size are largely determined by the wiring requirements of a VLSI system. Hence, wirelength estimation has always been instrumental in determining physically achievable design implementations [4]. Likewise, interconnect fanout can be used in interconnect modeling to enable more accurate delay and power calculations.

Based on different design stages and available information at each stage, we categorize wirelength estimation approaches into: (1) analytical, (2) netlist-based, and (3) placement-based [1], [6]. Analytical methods are based on Rent's rule, a famous well-established empirical relationship, and assume that the netlist is not known. However, the input may consist of the major factors, the total number of gates, and average number of pins per gate. In netlist-based methods, the total number of gates, total number of nets, total number of pins, and fanout distributions are easily obtained. Since we know exact values of major parameters, wirelength and fanout values can be predicted more accurately. Such estimates are typified by the "wireload models" used in RTL synthesis optimizations. After the placement stage, all the pins in the netlist are placed and the available routing resources, including number of routing layers, etc. are known and better estimation accuracy is achieved.

In this work, we propose interconnect wirelength and fanout models that have the flexibility of the analytical models with the accuracy of placement-based models. Our contributions are as follows.

- We propose accurate closed-form models to estimate interconnect wirelength and fanout. Our models are derived from implementation data.
- We consider microarchitectural and implementation parameters in our models to allow more accurate estimation of interconnect wirelength and fanout.
- Our models show significant accuracy improvement vs. the existing models with respect to implementation data.

The remainder of this paper is organized as follows. In Section II we review two of the recent works in interconnect wirelength and fanout estimations [3], [9]. Section III describes our implementation flow and testcases, and presents the details of our design of experiments. Section IV describes our modeling methodology and presents our proposed average wirelength and fanout models for one of our testcases. In Section V we validate our proposed models and compare them with the existing models. Finally, Section VI concludes the paper.

II. Existing Models

The underlying assumption for the existing models is that Rent's rule holds throughout an entire system. Rent's rule is a simple power-law relationship between the number of I/O terminals for a logic block, T, and the number of gates contained in that block, N as shown in [8]:

$$T = kN^p \tag{1}$$

where k and p are empirical parameters. Hence, the number of nets of length l, $Net(l)$ may be estimated by [3]

$$Net(l) = q(l)D(l) \tag{2}$$

where $D(l)$ is the number of valid two-pin net placement sites and $q(l)$ is the probability that a placement site is occupied. The occupancy probability can be derived as

$$q(l) = \frac{1}{2l}([1+2l(l-1)]^p + [2l(l-1)+4l]^p - [2l(l-1)]^p - [1+2l(l-1)+4l]^p) \tag{3}$$

and the number of 2-terminal net placement sites within the boundary of a finite, 2-D array of N gates is given in Equation (4).

$$D(l) = \begin{cases} l(l^2-1+6L(L-l))/3 & 1 \le l \le L \\ (2L-l+1)(2L-l)(2L-l-1)/3 & L \le l \le 2L \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

In addition, fanout distribution can also be derived by using Rent's rule recursively in a collection of gates. The following set of equations are due to [9]. In a system with N gates, the overall number of terminals connected through m-point nets is $N \cdot T_{net}(m)$, where $T_{net}(m)$ is the number of terminals shared through an m-point net of each gate in the boundary of the m gates. Therefore total number of m-point nets is simply $N \cdot T_{net}(m)/m$. Hence, total number of m-point nets in the entire system can be described by Equation (5). Subsequently, replacing m with $Fo+1$, where Fo is fanout, gives Equation (6).

$$Net(m) = \frac{kN((m-1)^{p-1} - m^{p-1})}{m} \tag{5}$$

$$Net(Fo) = \frac{kN(Fo^{p-1} - (Fo+1)^{p-1})}{Fo+1} \tag{6}$$

Equation (6) describes the fanout distribution in a logic network of N gates, in which k and p are Rent parameters.

Despite their flexibility, existing analytical wirelength and fanout models are not accurate due to (1) not having accurate information about the netlist, and (2) not taking into account the combining impact of microarchitectural and implementation parameters. Final design outcomes are affected by certain optimization steps that are employed during design implementation, i.e., pre-placement, post-placement, and pre-clock tree synthesis optimization steps, etc. Hence, the choice of implementation parameters can significantly change the quality of results. In this work, we consider the impact of relevant implementation parameters in our models (cf. Section III).

III. Implementation Flow and Design of Experiments

A. Implementation Flow and Tools

Figure 1 shows our implementation flow which includes traditional SP&R flow plus wiring characteristic extraction and model generation steps that we have scripted for "push-button" use in our

experiments. The steps in Figure 1 represent the major physical design steps. At each step we require that the design must meet the timing requirements before it can pass on to the next step.

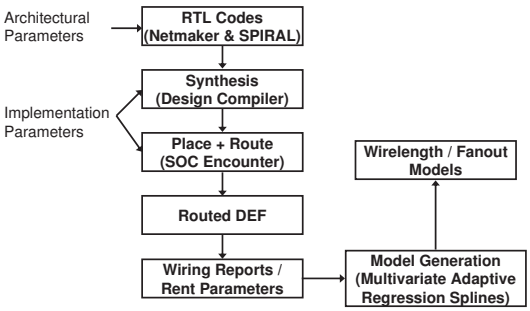

Fig. 1. Implementation flow.

In our flow, we first synthesize corresponding RTL codes of each of our testcases with worst-case timing libraries. During synthesis process we can set certain optimization objectives among which are: (1) target frequency, (2) target area, and (3) target power. Choosing different combinations of these objectives yields widely different quality of results. Hence, to mimic a typical industrial timing-driven flow, where certain performance constraints must be satisfied, we impose the target frequency as the primary objective while area and power minimization are considered as secondary objectives. In addition to optimization parameters, there are some 'inherent noise' in IC implementation tools that may be aggravated with respect to tightness of the timing constraint (i.e., target frequency) [7]. Therefore, we implement the designs at slower frequencies than the maximum achievable.

Using synthesized netlist we implement the designs through place & route (P&R) steps using different row utilization and aspect ratio values, at the floorplan stage. At the end of routing stage we generate the placed and routed DEF, and obtain wiring characteristics, i.e., total wirelength, total number of nets, average fanout, etc. and associate them with the corresponding microarchitectural and implementation parameters. In order to compare our models against the existing models we need to provide Rent parameters. We use an internal Rent parameter evaluation program which calculates Rent parameters of a given circuit design using circuit partitioning method, MLPart [2].

In our experiments we use two different IP cores: (1) on-chip router, and (2) Discrete Fourier Transform (DFT). We use *Netmaker v0.82* and *SPIRAL v1.0* to generate a library of synthesizable router and DFT core RTL netlists, respectively. We perform our experiments using libraries in TSMC 65GP technology. We use *Synopsys Design Compiler* [14] to synthesize the RTL netlists and *Cadence SOC Encounter* [13] to execute the P&R flow. We use different row utilization and aspect ratio values (explained more in Section III-B) and *RentCon*, an internal Rent parameter evaluation program, to extract Rent parameter values. Finally, *MARS3.0* tool suite [12] is used for model generation.

B. Design of Experiments

As with any other model characterization task, it is not trivial to obtain meaningful and sensible results, as large number of variables determines the ultimate outcomes. Hence, we only choose the relevant microarchitectural and implementation parameters that are of interest at the system-level which also significantly affect the quality of results. In our experiments we have two main axes: (1) microarchitectural parameters, and (2) implementation parameters. Below we explain the corresponding microarchitectural parameters for each of our testcases and show the values that they take on in our experiments.

On-Chip Router. For router we pick a baseline virtual channel (VC) router in which VC allocation and switch allocation are performed sequentially in one clock cycle. In a VC router the microarchitectural parameters are:

- fw: flit-width is the width of link or channel between routers and is specified in bits
- n_{vc}: number of virtual channels
- n_{port}: number of input and output ports
- l_{buf}: buffer length which is the number of flit buffers per virtual channel

DFT Core. For DFT we pick a conventional parameterizable DFT core obtained from *SPIRAL* DFT generator [15] with the following relevant microarchitectural parameters:

- n_{DFT}: size of the DFT core
- $width$: specifies the precision of the fixed-point representation of the input, the output, and intermediate data values
- t: twiddle bitwidth which specifies the precision of the fixed-point representation for the precomputed twiddle factors
- n_{fifo}: size of FIFO queues

In addition to the above list, there are three more parameters which control the resource usage of the DFT core: (1) dir which is a boolean parameter that selects whether the core accepts natural-ordered input vectors and outputs bit-reversed-ordered output vectors, or vice versa, (2) $scale$ which is a boolean parameter that determines whether the fixed-point data representation of intermediate values is scaled by a factor 2 to avoid overflow, and (3) d_p which specifies the number of kernel modules instantiated in the DFT implementation (i.e., degree of parallelism). These parameters are to allow the user to customize the tradeoff between performance and resource usage and do not change the functionality of the core. Hence, we do not consider them in our models. However, if their impact on wiring characteristics (i.e., wirelength and fanout) is of interest, then they can be added to the model at the expense of increased setup cost due to larger number of configurations. Table I shows the router and DFT core microarchitectural parameters and the values they take on in our design of experiments.

TABLE I
LIST OF ROUTER AND DFT MICROARCHITECTURAL PARAMETERS USED IN OUR DESIGN OF EXPERIMENT.

Router	
Parameter	Values
fw	{16, 24, 32, 64}-bits
n_{vc}	{2, 3, 5, 7}
n_{port}	3, 5, 7, 9
l_{buf}	{2, 3, 5, 7}-flit buffers
DFT	
Parameter	Values
n	{4, 16, 128}-bits
$width$	{8, 16, 32}
t	2, 4, 6
n_{fifo}	{2, 4, 16}-flit buffers

Our implementation parameters include (1) clock frequency f_{clk}, (2) aspect ratio ar, and (3) row utilization $util$. Target clock frequencies for the router design are 100MHz, 200MHz, and 400MHz, and we use 100MHz, 250MHz, and 450MHz for the DFT core. We also use three different aspect ratio values, 0.5, 1.5, and 2.5, while for row utilization, 50%, 70%, and 92% are used. Using automation scripts we vary the above microarchitectural parameters and generate corresponding RTL netlists for router and DFT cores, respectively. We then implement the RTL netlists using TSMC 65GP library.

IV. MODELING PROBLEM

Approximating a function of several to many variables using only the dependent variable space is a well-known problem with applications in many disciplines. The goal is to model the dependence of

a target variable y on several predictor variables $x_1, \cdots, x_n$ given realizations $\{y_i, x_{1i}, \cdots, x_{ni}\}_1^N$. The system that generates the data is presumed to be described by [5]

$$y = f(x_1, \cdots, x_n) + \epsilon \tag{7}$$

over some domain $(x_1, \cdots, x_n) \in \mathcal{D} \subset \mathcal{R}^n$ containing the data. There are two main regression analysis methods: (1) global parametric, and (2) nonparametric. The former approach has limited flexibility, and can produce accurate approximations only if the assumed underlying function, $\hat{f}$, is close to f. In the latter approach, $\hat{f}$ does not take a predetermined form, but is constructed according to information derived from the data. Multivariate adaptive regression splines (MARS) is a machine learning-based regression technique which is used in our methodology.

Given a design microarchitecture $\mathcal{X}^{\mu arch}$, circuit implementation $\mathcal{C}^{circ}$, we apply MARS to construct interconnect wirelength and fanout models, $WL = \hat{f}(x_1^{\mu arch}, \cdots, x_n^{\mu arch}, c_1^{circ}, \cdots, c_n^{circ})$, and $FO = \hat{g}(x_1^{\mu arch}, \cdots, x_n^{\mu arch}, c_1^{circ}, \cdots, c_n^{circ})$, respectively. Variables $x_1^{\mu arch}, \cdots, x_n^{\mu arch}$, and $c_1^{circ}, \cdots, c_n^{circ}$ denote corresponding router / DFT microarchitectural and implementation parameters, respectively. The general MARS model can be represented as [10]

$$\hat{y} = c_0 + \sum_{i=1}^{I} c_i \prod_{j=1}^{J} b_{ij}(x_{ij}) \tag{8}$$

where $\hat{y}$ is the target variable (i.e., average wirelength and fanout), c_0 is a constant, c_i are fitting coefficients, and $b_{ij}(x_{ij})$ is the truncated basis function with x_{ij} being the microarchitectural / implementation parameter used in the i^{th} term of the j^{th} product. I is the number of basis functions and J limits the order of interactions.

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

$$GCV(I) = \frac{1}{n} \frac{\sum_{k=1}^{n} (y_k - \hat{y})^2}{[1 - \frac{C(M)}{n}]^2} \tag{9}$$

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

Using implementation data and the nonparametric regression technique explained above, we propose accurate architectural-level interconnect wirelength and fanout models. Figures 2 and 3 show our proposed average wirelength and fanout models for DFT core, respectively.

Basis Functions
b_1 = max(0, n_{DFT} - 16); b_2 = max(0, 16 - n_{DFT});
b_4 = max(0, 16 - $width$) $\times b_1$; b_5 = max(0, $util$ - 0.5);
b_6 = max(0, n_{fifo} - 2); b_7 = max(0, $width$ - 16);
⋮
b_{31} = max(0, ar - 1.5) $\times b_7$; b_{35} = max(0, t - 2) $\times b_{31}$;
Average Wirelength Model
WL_{avg} = 22.4886 + 0.056 $\times b_1$ - 0.328 $\times b_2$ + 0.013 $\times b_4$
- 5.891 $\times b_5$ - 0.226 $\times b_6$ - 0.194 $\times b_7$ - 0.271 $\times b_8$
- 0.018 $\times b_9$ + 0.001 $\times b_{11}$ + 0.017 $\times b_{12}$ + 0.0002 $\times b_{13}$
+ 0.001 $\times b_{15}$ + 0.002 $\times b_{16}$ - 9.104e-6 $\times b_{17}$ - 2.176e-5 $\times b_{18}$
- 0.051 $\times b_{19}$ - 0.017 $\times b_{21}$ - 2.228e-5 $\times b_{24}$ + 0.0003 $\times b_{25}$
+ 0.003 $\times b_{27}$ - 0.01284 $\times b_{35}$

Fig. 2. Average wirelength model for DFT core in 65nm.

Basis Functions
b_1 = max(0, n_{DFT} - 16); b_2 = max(0, 16 - n_{DFT});
b_3 = max(0, $width$ - 8); b_4 = max(0, n_{fifo} - 2);
b_5 = max(0, n_{DFT} - 16) $\times b_4$; b_6 = max(0, 16 - n_{DFT}) $\times b_4$;
⋮
b_{30} = max(0, $width$ - 16) $\times b_9$; b_{33} = max(0, 16 - n_{DFT}) $\times b_{18}$;
Average Fanout Model
FO_{avg} = 3.707 + 0.003 $\times b_1$ - 0.034 $\times b_2$ - 0.011 $\times b_3$
- 0.016 $\times b_4$ + 8.602e-5 $\times b_5$ + 0.002 $\times b_6$ + 7.051e-5 $\times b_7$
+ 0.002 $\times b_8$ - 9.943e-5 $\times b_9$ - 0.002 $\times b_{10}$ - 0.084 $\times b_{13}$
+ 7.989e-6 $\times b_{16}$ - 4.533e-6 $\times b_{17}$ + 0.0002 $\times b_{18}$
+ 1.011e-5 $\times b_{21}$ + 0.0005 $\times b_{22}$ + 0.006 $\times b_{23}$ + 0.003 $\times b_{25}$
- 0.0003 $\times b_{27}$ - 1.478e-5 $\times b_{28}$ - 1.492e-5 $\times b_{29}$
- 8.567e-6 $\times b_{30}$ - 1.225e-5 $\times b_{33}$

Fig. 3. Average fanout model for DFT core in 65nm.

V. Experimental Setup and Significance Assessment

We now validate our wirelength and fanout models described in earlier sections. For interconnect wirelength we compare four different models.

1) Our proposed model (Prop.)
2) Christie *et al.* [3] model with N, p, and k modeled as a function of microarchitectural and implementation parameters (Model 2)
3) Modified Christie model with a correction factor (Model 3)
4) Christie model with N, p, and k derived from layout data for each configuration (Model 4)

For interconnect fanout we compare the same four models except that our reference model is the one proposed by Zarkesh-Ha *et al.* [9].

Our proposed model (Prop.) uses machine learning-based regression method, i.e., MARS, to estimate interconnect wirelength and fanout as a function of microarchitectural and implementation parameters. For Model 2 we use the same regression approach to model N, p, and k as a function of microarchitectural and implementation parameters. Then, we use the estimated Rent parameters in the Christie (Zarkesh-Ha) model to obtain wirelength (fanout) values. In the Christie model, unit distance between adjacent placement sites, i.e., $l = 1$, is modeled as $\sqrt{X_{die} Y_{die}/N}$, where X_{die} and Y_{die} are the width and height of the floorplan size of each design, respectively. In Model 3, we first model N, p, and k as a function of microarchitectural parameters only (i.e., with similar modeling approach as in Prop.). Next we apply the estimated Rent parameters in the Christie (Zarkesh-Ha) model and introduce a correction factor α such that *Actual Wirelength* = $\alpha \times Model_{Christie}$ (*Actual Fanout* = $\alpha \times Model_{Zarkesh-Ha}$). Then we model α as a function of implementation parameters only, using the same modeling approach as in our proposed models. In the modified Christie's model (Model 3), we do not include the unit distance model as used in Model 2 because unit distance depends implementation parameters. Finally, Model 4 uses Christie (Zarkesh-Ha) model to estimate wirelength (fanout) using extracted N, p, and k values from implemented designs.

For Models 2, 3, and 4, to extract Rent parameters, we use layout reports to obtain N, and an internal Rent parameter evaluation program *RentCon*, which extracts p and k values from a placed and routed DEF. To compute p and k we use circuit partitioning-based method in which we recursively partition the netlist into smaller partitions using min-cut bisection until the minimum number of cell instances over all partitions reaches 2, and we count each source-sink connection crossing the boundary as one pin. For each level of the recursive bipartitioning, we compute the geometric mean values of cell instances and pins, which represent one data point in the fitted curve of the Rent parameter.[1]

[1]The classic multilevel circuit partitioner MLPart [2] is used to recursively partition the circuit netlist.

TABLE II

COMPARISON OF AVERAGE WIRELENGTH DERIVED FROM OUR PROPOSED (PROP.), MODEL 2, MODEL 3, AND MODEL 4 (CHRISTIE [3]) MODELS WITH RESPECT TO ACTUAL IMPLEMENTATION DATA.

Metric	Discrete Fourier Transform (DFT)				Router			
	Prop.	Model 2	Model 3	Model 4	Prop.	Model 2	Model 3	Model 4
maximum % error	**21.3**	76.4	98.2	79.5	**17.9**	59.4	54.6	59.9
average % error	**3.4**	17.9	22.7	18.1	**2.3**	27.4	16.3	27.2

TABLE III

COMPARISON OF AVERAGE FANOUT DERIVED FROM OUR PROPOSED (PROP.), MODEL 2, MODEL 3, AND MODEL 4 (ZARKESH-HA [9]) MODELS WITH RESPECT TO ACTUAL IMPLEMENTATION DATA.

Metric	Discrete Fourier Transform (DFT)				Router			
	Prop.	Model 2	Model 3	Model 4	Prop.	Model 2	Model 3	Model 4
maximum % error	**5.7**	23.7	18.7	22.7	**1.4**	18.9	22.1	18.2
average % error	**0.8**	10.1	7.3	10.1	**0.2**	5.6	5.7	5.6

To generate our models, we randomly select 10% of our entire data set (i.e., total of 2187 cases for each testcase), and test the models on the other 90% of the data. To show that the selection of the training set insignificantly changes our model accuracy we randomly select 10% of the entire data set five times and show the corresponding maximum and average error values (Table IV).

TABLE IV

IMPACT OF RANDOM SELECTION OF THE TRAINING SET ON MODEL ACCURACY.

Experiments	average wirelength % diff		average fanout % diff	
	max	avg	max	avg
Exp 1	22.9	3.4	4.3	0.7
Exp 2	18.2	3.5	6.0	1.3
Exp 3	23.9	3.4	8.6	0.6
Exp 4	24.2	3.4	5.1	0.7
Exp 5	16.8	3.5	4.8	0.7

Tables II and III show comparisons of our proposed wirelength and fanout models with the above models, respectively. We observe significant accuracy improvement vs. existing models (Model 4) with respect to layout data. Our estimated average wirelength values show an accuracy improvement of up to 14.7% (58.2%), and 24.9% (42%) in average (maximum) errors for DFT and router testcases, respectively. For average fanout, we observe up to 9.3% (17%), and 5.4% (16.8%) in average (maximum) errors for DFT and router testcases, respectively. Finally, Figures 4, 5, 6, and 7 show scatter plots of our average wirelength and fanout estimations against corresponding Christie and Zarkesh-Ha models (i.e., Model 4) with respect to layout data.[2] These plots confirm the accuracy improvement of our proposed models vs. existing models.

From Models 2 and 3 we understand that both microarchitectural and implementation parameters should be considered during model development. In addition, we can confirm that existing Rent's rule-based wirelength and fanout estimation models fail to correctly capture the impact of microarchitectural and implementation parameters which results in unrealistic wiring characteristics estimations.

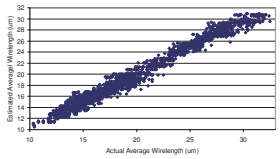

Fig. 4. Our estimated average wirelength against layout data.

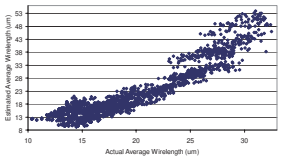

Fig. 5. Christie's average wirelength estimation against layout data.

[2]These plots are for the DFT models, however, router models show similar accuracy.

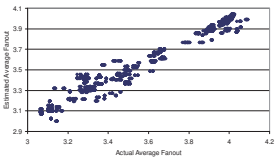

Fig. 6. Our estimated average fanout against layout data.

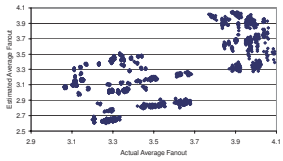

Fig. 7. Zarkesh-Ha's average fanout estimation against layout data.

VI. CONCLUSIONS

In this work, we present accurate architectural-level interconnect wirelength and fanout prediction models. Our models show significant accuracy improvement vs. existing models, i.e., up to 58% and 17% for wirelength and fanout estimations, respectively. Accurate wiring estimations can drive effective early-stage design space exploration. To enable system-level design space exploration, we consider all the relevant microarchitectural parameters of our testcases. In addition, our models consider the combining effect of the architectural and implementation parameters on wiring characteristics.

REFERENCES

[1] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov and A. Zelikovsky, "On Wirelength Estimations for Row-Based Placement", *IEEE Trans. on Computer-Aided Design*, 18(9), 1999, pp. 1265–1278.

[2] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Improved Algorithms for Hypergraph Bipartitioning", *Proc. ASPDAC*, 2000, pp. 661–666.

[3] P. Christie and D. Stroobandt, "The Interpretation and Application of Rent's Rule", *IEEE Trans. on Very Large Scale Integration Systems*, 8(6), 2000, pp. 639–648.

[4] J. A. Davis, V. K. De and J. D. Meindl, "A Stochastic Wire-Length Distribution for Gigascale Integration (GSI) – Part I: Derivation and Validation", *IEEE Trans. on Electron Devices*, 45(3), 1998, pp. 580–589.

[5] J. H. Friedman, "Multivariate Adaptive Regression Splines", *Annals of Statistics*, 19(1), 1991, pp. 1–66.

[6] K. Jeong, A. B. Kahng and H. Yao, "On Modeling and Sensitivity of Via Count in SOC Physical Implementation", *Proc. ISOCC*, 2008, pp. 125–128.

[7] A. B. Kahng and S. Mantik, "Measurement of Inherent Noise in EDA Tool", *Proc. ISQED*, 2002, pp. 206–211.

[8] S. Landman and R. L. Russo, "On a Pin Versus Block Relationship for Partitions of Logic Paths", *IEEE Trans. on Computers*, C-20, 1971, pp. 1469–1479.

[9] P. Zarkesh-Ha, J. A. Davis, W. Loh and J. D. Meindl, "Stochastic Interconnect Network Fan-out Distribution Using Rent's Rule", *Proc. IITC*, 1998, pp. 184–186.

[10] Y. Zhou and H. Leung, "Predicting Object-Oriented Software Maintainability Using Multivariate Adaptive Regression Splines", *Journal of Systems and Software*, 80, 2007, pp. 1349–1361.

[11] *Netmaker*, http://www-dyn.cl.cam.ac.uk/ rdm34/wiki/index.php?title=Main_Page.

[12] *MARS User Guide*, http://www.salfordsystems.com/mars.php.

[13] *Cadence SOC Encounter User's Manual*, http://www.cadence.com/.

[14] *Synopsys Design Compiler User's Manual*, http://www.synopsys.com/.

[15] *SPIRAL*, http://www.spiral.net/hardware/dftgen.html.