Improved A Priori Interconnect Predictions and Technology Extrapolation in the GTX System

Yu Cao,[1] Chenming Hu,[1] Xuejue Huang,[1] Andrew B. Kahng,[2] Igor L. Markov,[3]
Michael Oliver, Dirk Stroobandt,[4] and Dennis Sylvester[3]

CS Department, UCLA, USA; [1]EECS Department, UC Berkeley, USA; [2]CS Department, UCSD, USA;
[3]EECS Department, U. Mich., Ann Arbor, USA; [4]ELIS Department, Ghent University, Belgium

{oliver}@cs.ucla.edu; {ycao, hu, xuejue}@eecs.berkeley.edu; abk@ucsd.edu; dstr@elis.rug.ac.be; {imarkov,dennis}@eecs.umich.edu

Abstract ― *A priori interconnect prediction and technology extrapolation are closely intertwined. Interconnect predictions are at the core of technology extrapolation models of achievable system power, area density and speed. Technology extrapolation, in turn, informs a priori interconnect prediction via models of interconnect technology and interconnect optimizations. In this paper, we address the linkage between a priori interconnect prediction and technology extrapolation in two ways. First, we describe how rapid changes in technology, as well as rapid evolution of prediction methods, require a dynamic and flexible framework for technology extrapolation. We then develop a new tool, the GSRC Technology Extrapolation System (GTX), which allows capture of such knowledge and rapid development of new studies. Second, we identify several "non-traditional" facets of interconnect prediction and quantify their impact on key technology extrapolations. In particular, we explore the effects of interconnect design optimizations such as shield insertion, repeater sizing and repeater staggering, as well as modeling choices for RLC interconnects.*

Keywords ― *A priori interconnect prediction, technology extrapolation, VLSI, system performance models, interconnect delay, crosstalk noise, inductance.*

I. Introduction

Technology extrapolation ― the calibration and prediction of achievable design in future technology generations ― drives the evolution of VLSI system architectures, design methodologies, and design tools. To fully explore future possibilities, technology extrapolation systems must contain models for a priori performance prediction, i.e., without exact knowledge about the final system. Therefore, a priori interconnect predictions (of wire length distribution, interconnect embedding, and interconnect performance) are at the core of technology extrapolation models of achievable system power, area density and speed. Technology extrapolation, in turn, informs a priori interconnect prediction via models of interconnect technology and interconnect optimizations. In this paper, we address the linkage between a priori interconnect prediction and technology extrapolation.

1.1 A Priori Interconnect Prediction

Currently, a priori interconnect prediction techniques are focused on estimating average wire lengths, or wire length distributions, of a design in advance of the actual placement and routing. These estimates can be useful in floorplanning and design exploration, for example. A priori predictions must be very fast and reasonably accurate to drive floorplan changes, circuit optimizations, and other aspects of the design process.

A primary tenet of interconnect estimation is *Rent's Rule* [1,2] which predicts a power law relationship between the number of gates in a module of a partitioned circuit and the number of terminals needed for communication between the module and the remaining of the circuit. Wire length estimates based on Rent's rule and a hierarchical placement model were introduced by Donath [3] in 1979 and several improvements were published by Stroobandt [4,5] almost 20 years later. Around the same time, Davis [6] independently found a similar result for a flat placement model and this model has been frequently used by other authors. Since the introduction of the System-Level Interconnect Prediction Workshop (SLIP) in 1999 [7], the progress in the field of interconnect prediction has been tremendous [2,8,9]. Still, the field of a priori interconnect prediction has not matured enough to make its results easily applicable. It is difficult to keep up with the rapid evolutions in this field. Also, current techniques do not provide the accuracy necessary to apply them with confidence. In order to improve accuracy, more detailed knowledge about the entire system design is needed. The latter problem can be solved by coupling a priori prediction techniques with technology extrapolation systems. If such a technology extrapolation system can be made flexible enough to be easily adaptable to new a priori prediction results, much progress could be made in both a priori interconnect prediction and technology extrapolation at the same time. This is the goal of this paper.

1.2 Technology Extrapolation

Leading edge VLSI system design aggressively exploits new process technologies, circuit techniques, design methodologies and design tools. It is thus difficult to predict the envelope of *achievable design* ― e.g., with respect to performance, power, area, manufacturing cost, etc. ― for a given behavior or function, in a given (future) process technology. On the other hand, such *technology extrapolation* activity directly influences the evolution of future VLSI system architectures, design methodologies, and design tools, as well as broader investment strategy in the semiconductor and electronics sectors.

Highly influential technology extrapolation systems, developed 5-10 years ago, are due to Bakoglu and Meindl (SUSPENS) [10], Sai-Halasz [11], and Hewlett-Packard Laboratories (AIM) [12]. More recent second-generation systems include GENESYS [13], RIPE [14,15,16], and BACPAC [17], along with Roadmap-related efforts [18,19]. Typically, each system provides a plausible performance prediction model and estimates of die size and power dissipation, based on a small set of descriptors spanning device/interconnect technology through system architecture. The most critical aspect of performance prediction is the idealized *critical path* in the system of interest. For example, a model on-

chip critical path might be described as "12 fanout-4 gates driving average-length local interconnects, plus an optimally buffered corner-to-corner 2 μm-wide global wire". A key component of the model is the a priori interconnection length model, as can be seen from the example above. The validity and accuracy of the a priori interconnect prediction models used are critical to the conclusions drawn from the extrapolation system. Therefore, it is important in technology interpolation systems to see and to understand the underlying models and to be able to easily adjust them.

In Section 2.1, we observe that (i) current technology extrapolation systems are often incomparable, (ii) they are "hard-coded" (hence it is difficult to assess their quality and to explore changes through modeling choices), and (iii) their development has entailed a near-total duplication of effort. These observations limit their application on system level technology extrapolation, including interconnect prediction, and motivate efforts toward an entirely new level of technology extrapolation capability. Our GSRC Technology Extrapolation (GTX) system has been developed with the goals of *flexibility*, *quality* and *prevention of redundant effort* in mind. The GTX system addresses these goals by providing an open, portable *framework* for specification and comparison of alternative modeling choices. A fundamental design decision in GTX is to separate model specifications from the derivation engine. GTX adopts a paradigm wherein *parameters* and *rules* allow users to flexibly capture an essentially unbounded space of attributes and relationships that are germane to VLSI technology and design. GTX, as well as documentation for numerous studies, is downloadable at http://vlsicad.ucsd.edu/GTX/.

1.3 Contributions of This Work

The main contributions of this work are:

- Adding awareness of circuit optimization degrees of freedom into the predictions of interconnect performance used in technology extrapolation.
- Adding more detailed interconnect modeling into interconnect predictions.
- Identifying sensitivities in extrapolation models, showing where the most stringent accuracy requirements are with respect to how interconnect properties and optimizations are modeled.
- Describing GTX - a new, portable, and general framework for reusable technology extrapolation effort.

With respect to interconnect prediction, we conduct a number of studies, within the GTX framework, on the *optimized global interconnect* portion of on-chip critical paths. Specifically, we assess the impact on critical path models of several potentially important, yet previously unmodeled, *optimization degrees of freedom* and *design constraints*, including:

- ♦ Adding extracted inductance estimates (and analytic RLC line delay estimates) to the interconnect model.
- ♦ Modern repeater optimizations, such as detailed repeater size and interconnect width optimizations [20].
- ♦ *Engineering considerations*, e.g. repeater area/size bounds, deliberate backing off of optimal values to the "knee of the curve," and limiting the number of allowed wire widths.
- ♦ *Switch factor* based bounds on delay uncertainty due to crosstalk from neighboring wires.
- ♦ Using *real-world design technology* in the global interconnect models, e.g., repeater staggering and shielding techniques.

Our work attempts to dispel some of the "vagueness" of current performance predictions that arises from the gaps noted above. We do not make any value judgments with respect to existing models; rather, we simply build a comprehensive modeling environment with GTX that allows us to identify the issues that *must* be considered by current and future performance predictions.

The remainder of our paper is organized as follows. Section 2 reviews relevant previous work in VLSI technology extrapolation and reveals large sensitivities to particular input parameters and to modeling choices. The observations lead to general goals for technology extrapolation systems. The remaining of the section describes the architecture and implementation of our GTX technology extrapolation system. In Section 3, we link a priori interconnect models to technology extrapolation and assess the impact of several interconnect design optimizations as well as modeling choices for RLC interconnects. We study (i) inductance on critical paths in terms of shielding, driver sizing, and slew rates (and their impact on coupling noise), (ii) the cost-performance tradeoffs inherent in signal shielding; and (iii) a comprehensive study on wire sizing and repeater optimization. This analysis attempts to give a realistic depiction of what an optimal repeater topology should look like in terms of repeater sizing, wire widths, pitch allocation, etc.

II. GTX: A New Technology Extrapolation Framework

2.1 VLSI Technology Extrapolation

A number of previous systems attempt to forecast and estimate the performance of microprocessors. Four systems - SUSPENS, GENESYS, RIPE and BACPAC – are especially noteworthy.

SUSPENS [10] is the forerunner for most technology extrapolation systems. It predicts the clock frequency, chip area and power dissipation. SUSPENS ignores on-chip cache and memory structure, as well as details of multi-layer interconnect structure and clock distribution. SUSPENS is also oblivious to such DSM effects as scaling and noise.

GENESYS [13] offers both a GUI for MS Windows (95, 98, NT) and a command-line interface (it has no Web interface). The GENESYS output file is divided into four main sections: device/material, circuit, interconnect, and system. System-level outputs include throughput, maximum clock frequency, CPI, and delay times for random logic and interconnect.

RIPE [14] explores the effect of interconnect design and technology tradeoffs on IC performance. Default input data are extracted from the NTRS roadmap. Memory and the multilayer interconnect structure are taken into account. Recent additions include estimations of linewidth variability, yield, signal integrity, and electromigration [15], as well as RLC interconnect models [16]. The user cannot add new parameters and rules.

BACPAC [17] is based on a system-level performance model that consists of smaller-scale analytical models. The innovations of BACPAC include attention to power dissipation, on-chip memory, process variation, and other effects. BACPAC is applicable to both ASICs and microprocessors. It attempts to enhance the accessibility of technology extrapolation via a Web-based interface. However, the derivation flow is mostly fixed, and users cannot add new parameters and rules.

With respect to the previous systems for technology extrapolation, we make the following observations.

1. Different systems may predict the same "parameter" (e.g., microprocessor clock frequency), yet be incomparable due to differing sets of inputs and assumptions, as well as lack of documentation and visibility into internal calculations.
2. Each system typically offers exactly one "inference chain" for any given output of interest (e.g., cycle time). Furthermore, this inference chain can involve a large spectrum of modeling choices. The *quality* of such modeling choices cannot be assessed since the system is "hard-coded", and no exploration of modeling sensitivity or robustness is possible.
3. The hard-coded nature of previous systems also means that they are inflexible: the user cannot define studies of other system parameters, and interaction with the system is limited
4. Finally, development of previous systems has entailed near-total duplication of effort since each system attempts to bind the same envelope of achievable design in gathering, interpreting, and systematizing data and models. Redundant efforts are made even though no single entity - EDA vendor, system house, or academic group - can achieve "best-possible modeling" of all aspects of technology and design.

The second point above leads to a simple and motivating experiment that reveals how existing technology extrapolations have large *sensitivities* to particular input parameters and to modeling choices. We incorporate the *cycle-time models* from SUSPENS [10] (with extensions of Takahashi et al. [21]), BACPAC [17], and Fisher et al. [19] within GTX and reproduce published results with each model. As will be explained in Section 2.3, our implementations are tuned to ensure maximal interchangeability of GTX rules for each model, allowing extensive evaluation of various model sensitivities.

Our experiments address two basic types of sensitivity: *parameter sensitivity* and *model (or rule) sensitivity*. The former describes the influence of changes in the primary input parameters to the model, while the latter describes the influence of changes in the estimation model itself. (We do not aim to make value judgments about or compare the models; rather, our goal is to show the value of being able to try variant estimation methods.) We perform the following experiments:

1. For the same primary inputs, compare the results for different models (model sensitivity).
2. For each model, vary the input parameters by ± 10% and note the difference in the resulting clock frequency (parameter sensitivity).
3. For each rule out of one rule chain (model), replace one rule by a rule from another model that computes the same parameter and record the change in clock frequency (model sensitivity).

Table 1. Logic stage delay t_l, global delay t_g, and overall clock frequency f_c for interconnect models.

model	t_l (ps)	t_g (ps)	f_c (MHz)
BACPAC	893	115	745
Fisher	1162	204	659
SUSPENS	665	–	1505

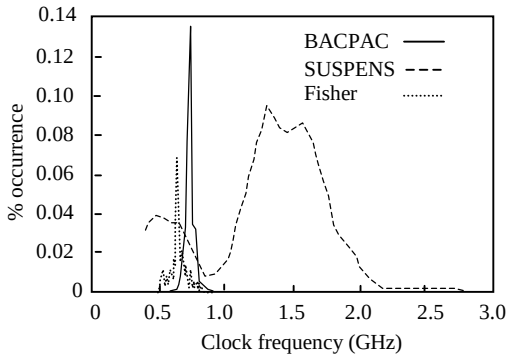

Figure 1. Parameter sensitivity: BACPAC, Fisher, and SUSPENS.

For all experiments and models, we use a common primary input (PI) parameter base derived for 0.25μm technology and mainly following the default parameter values of BACPAC (additional PIs for other models are tuned to these parameter values). Despite the common parameter base, our initial model sensitivity assessment of the SUSPENS, BACPAC, and Fisher models shows very different values for respective predictions of logic stage delay (t_l), global delay (t_g), and overall clock frequency (f_c) (see Table 1).[1] A more detailed type of model sensitivity analysis tests the sensitivity of a given model to "hybridization" with other models. In other words, we take the rule chain for a single model and replace exactly one rule by an equivalent rule (or set of rules) from another model. Details of such experiments are in [22].

Parameter sensitivity studies that vary *single* PI parameter values in each model's evaluation, changing each PI value by ± 10%, are also detailed in [22]. More extensive studies *simultaneously* change more than one parameter value, again by ± 10%. Since this produces three values for each parameter and since there are between 15 (SUSPENS) and 46 (BACPAC) primary inputs, it is not possible to consider all possibilities, and we therefore only vary smaller parameter subsets (up to 7 parameters at the same time). Figure 1 plots the relative occurrence of clock frequency values in small intervals that result from the sweeping. If we say that a more "robust" (to changes of its input parameters) model is one with a narrower and higher peak, then BACPAC would seem to be the most robust, and SUSPENS the least robust.[2]

2.2 Goals of a Technology Extrapolation Framework

The above observations motivate three key goals as we seek a new level of technology extrapolation capability.

[1] SUSPENS does not have a model for global delay on chip. We believe this was compensated by taking into account more stages but we chose to use a number of stages equal to that of the other models to maintain interchangeability. While the very high 1.5 GHz frequency predicted by SUSPENS is largely due to the lack of a global interconnect model, the logic stage delay is still significantly different from the other models.

[2] In general, we find BACPAC to be much less sensitive to either hybridization with other models, or variation of input parameter values. This does not necessarily imply that BACPAC is a better model (e.g. if a model predicts a clock frequency of 700 MHz independent of any input parameter value, then this is "robust" but not practical or correct). Note also that sweeping over more than 7 parameters at once will widen the peaks shown in the plot.

Flexibility. To experimentally determine model sensitivity and robustness, users must have the ability to (i) (interactively) edit available inference chains and collections of "rules," (ii) define new parameters and rules, and (iii) request specific types of studies, such as parameter optimization or trade studies. Support for interaction (GUI, session management, etc.) is an implicit requirement.

Quality. We seek adoptability in the sense of having an easy learning curve and providing much "value" in the form of high-quality embedded data, embedded models, and user interface. We aim for a system that can be continuously improved to have "best-possible models" across the entire scope of technology extrapolation. Since no single group can achieve this alone, we require an open-source mechanism that is conducive to distributed ownership and maintenance.

Prevention of redundant effort. To avoid redundant effort, we seek a "permanent repository of first choice" for rules and data (calibration points) related to technology extrapolation. Beyond the open distribution mechanism noted above, *adoptability* (by academics open to collaboration, or by companies with proprietary data and firewalls) and *maintainability* become key concerns. A lower bound for adoptability is a platform-independent implementation that subsumes the functionality of all previous "hard-coded" systems. This recognizes the proprietary nature of user data and offers usability behind firewalls, with frequent releases to update the state of model/data collection.

2.3 The GSRC Technology Extrapolation (GTX) System

GTX establishes a clear separation between *knowledge* and *implementation* (Figure 2). Knowledge is represented independently from its implementation in a serializable public-domain format. It contains data (*parameters*), the models (*rules*) that can operate on them and studies (*rule chains*), a collection of rules to obtain a particular result. The implementation then consists only of a *derivation engine* and a graphical user interface (*GUI*). The engine can load *modules* of parameters, rules and a rule chain and automatically operate on them. The result of the operation is new data. *Known studies* are supplied in pre-packaged rule chains; additional modules can be written and shared by users.

2.3.1 Parameters, Rules and Rule Chains

As previously mentioned, the values of interest are encapsulated in *parameters*, and potential inferences between them in *rules*. Each rule accepts as inputs a fixed collection of parameters, and its evaluation computes a single output parameter. The collection of available rules and parameters is naturally viewed as a *bipartite digraph* in which an edge extends from a rule to a parameter if the parameter is the output of the rule, or from a parameter to a rule if the parameter is an input to the rule.

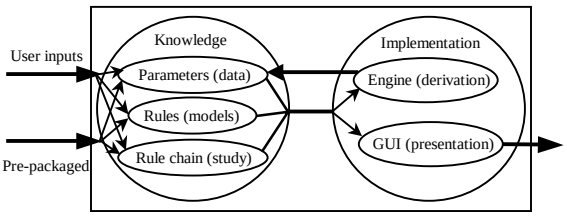

Figure 2. Schematic view of GTX framework.

Two or more rules may compute the same output (i.e., alternative models of the same value), and the above digraph may contain cycles. However, any particular calculations must avoid such irregularities to prevent value conflicts and infinite loops. This is supported through the notion of a *rule chain* - an acyclic subgraph of the graph of available rules and parameters such that no two rules compute the same output.

Parameters

Parameters are the common base on which rules of different types operate. The main attributes of a parameter are its name, data type and its units. In order to obtain the goal of high reuse-ability of rules and parameters, the parameter names have to be carefully chosen so that they are easy to understand. Also, we must ensure that no physical attribute receives two different names in GTX and that no GTX parameter name is used for two different physical attributes. Therefore, we have devised strict rules for the parameter names [22]. The grammar for parameters is specified at our website [23]. Following is a very simple example representing the chip edge length.

```
#parameter dl_chip
#type double
#units {m}
#default
  1e-2
#description
  chip edge length
#endparameter
```

Namespaces

To provide a better indication of the source of rules and parameters, we have implemented *namespaces* in which a rule or parameter may be found. For example, rules derived from BACPAC are in the BACPAC namespace. "Meaningful" parameters should be in the *global namespace*; parameters that are used only for the purpose of specific rules should be in the namespace of those rules. A rule can "see" parameters in its own namespace or in the global namespace. The :: operator is used to indicate the namespace, similar to C++.

Rules

GTX supports the following types of rules:

ASCII rules provide a closed-form expression language that allows calculating the output from the input using common mathematical functions or operations, interpolation or table lookup, and if-then-else. There is no program flow and therefore no iteration per se, but *vector* operations are provided that allow common computations such as sums. In a new addition to the "ASCII rules" language, it is now possible for rules to call other rules as *functions*. This means that one or more of the input parameters of the called rule may be substituted by an expression in the calling rule. The specific motivation for this extension is the ability to take, e.g., a rule that computes a parameter based on calendar year, and use it in another rule, but *shifted* by some number of years.

External executable rules cause the engine to invoke a specified executable file (e.g., a PERL script), passing the input values on the command line or through a file. The external executable saves its output into a temporary file to be read by the engine.

External executable rules allow the inclusion of executables for which source code is not available or for computations that cannot be expressed in ASCII rules.

Code rules are hard-coded into the engine itself and require recompilation of the engine code. Therefore, they are appropriate only when execution speed is an issue.

These types provide a reasonable expressive power and facilitate easy updates to GTX with new models. The following is an example of an ASCII rule computing the chip edge length from the chip area. The `#output` and `#inputs` sections declare the types and units of output and input parameters. The formula in the `#body` section specifies the evaluation of the rule.

```
#namespace BACPAC
#rule dl_chip
#description
rule from BACPAC for the chip edge length
#output
   double {m} dl_chip;// chip edge length
#inputs
   double {m^2} dA_chip; // chip area
#body
   sqrt(dA_chip)
#reference
   BACPAC
#endrule
```

Rule chains

The GTX user indicates to the engine which of the currently available rules should be evaluated, by providing a simple list of those rules. The order in which rules are executed forms the *rule chain*, and is decided by the engine based on the relations between the rule inputs and outputs. If we had a rule "BACPAC::dA_chip" that computes the chip area, e.g., as a function of number and size of the gates, then the chip edge length could be computed by executing the following rule chain

```
BACPAC::dA_chip
BACPAC::dl_chip
```

2.3.2 Engine Structure and Operation

For each parameter, the engine maintains zero, one or more values. Values can be set by default, loaded from files, entered by the user or computed. Multiple values can be computed by *sweeping*, i.e., evaluating rules over multiple combinations of input parameters. When instructed to evaluate a rule chain, the engine clears values that can be computed by rules of the chain. For each combination of values of primary inputs of the chain, the engine evaluates rules in topological order and adds their output values to respective collections of values, unless some constraints fail. A faster algorithm is possible to produce all derivable *sets of values*, but with our simple algorithm the inputs of any particular value can be recovered (e.g., for minimization along a rule chain).

2.3.3 Graphical User Interface

The GUI is implemented with the cross-platform toolkit wxWindows; we have run it successfully on Windows 95/98, Windows NT, Solaris and Linux. At any given time, the user may view (i) current parameters, (ii) current rules, (iii) current rule chain, (iv) values of parameters in the current chain, or (v) the graph of rules and parameters. When a particular parameter or rule is selected, its details are shown and can be edited. The chain view shows all rules in the chain and helps the user to add new rules to the chain. The values view shows both inputs to and outputs of the current chain. The inputs may be edited. This view permits invoking the chain and observing the output, sweeping over multiple input values, observing the trace of such a sweep (including optimization) and plotting (Figure 3). The graph view facilitates understanding what parameters are inputs (or outputs) to what rules, or optionally what parameters are used to compute what other parameters. In addition to the five views, the GUI handles extensive file I/O and interactive addition of new parameters and rules.

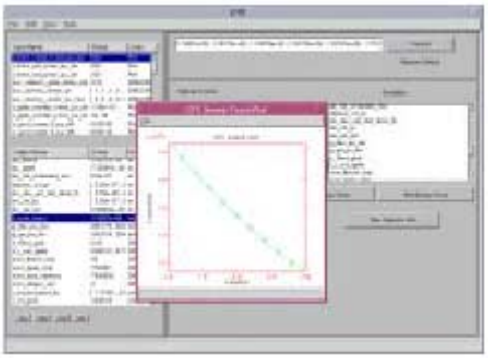

Figure 3. Screen shot of GTX GUI.

III. New Dimensions of A Priori Interconnect Prediction

The very nature of technology extrapolation requires the use of a priori prediction techniques based on simple models with a limited number of parameters, preferably represented by simple ASCII rules. One of the key aspects of interconnect-centric design is the interconnection length. The early technology extrapolation systems used the simple a priori interconnection length estimation technique of Donath [3] to estimate average wire lengths. More recent systems included Davis' interconnection length model [6] and progress in a priori interconnect prediction has accelerated with the introduction of the System-Level Interconnect Prediction Workshop (SLIP) in 1999 [7]. For a technology extrapolation system to remain up-to-date, it is necessary to adapt to such evolutions easily. The flexibility of GTX allows such adaptations (simply by rewriting a few ASCII rules without having to recompile) and ensures that the effects of the latest improvements in wire length estimation can be investigated. In particular, we have found that GTX is useful as a development tool for adding new rules that model a very particular part of the design behavior, as an emulation tool for existing estimator tools, as a comparison tool between different estimation methods and as an evaluation tool for those methods.

To better explain the capabilities of the GTX system, the remainder of this paper is dedicated to the combination of a priori interconnect prediction models and models of the process and/or the design technology. Both types of models are easily implemented in GTX rules which allows interesting studies of the impact (quantitatively and qualitatively) of potential design or process improvements. In the next two subsections, examples of such studies - (1) with respect to inductance and RLC interconnect analysis, and (2) with respect to interconnect design optimizations - are presented. As can be seen from the results, these represent two important new dimensions in achieving accurate a priori interconnect performance predictions. In general, these studies are aimed at achieving reus-

able, transparent, well-engineered prediction models for optimized interconnects and on-chip critical paths. The default technology used in our studies (exceptions will be noted) is a 0.18μm CMOS process with a supply voltage of 1.8V. V_{th} is 0.3V, and the I_{dsat} values for NMOS/PMOS are 700/350 μA/μm. The critical global interconnect we assume is a 1.5cm top-level copper line with thickness of 1.3μm and $\varepsilon_r = 4.0$.

3.1 Inductance and RLC Interconnect Analysis

The effect of inductance on the wire delay is well demonstrated in [10]. Interconnects in deep-submicron designs operating at high frequencies, whose inductive impedance cannot be neglected, must be modeled using RLC segment models. When the ratio of inductive impedance to resistance exceeds a certain threshold in an interconnect line, a non-monotone voltage response (i.e., oscillation before settling to a steady state value) results. This makes threshold delay calculation much more difficult than in the RC line case. In such regimes, Elmore and other RC line models cannot accurately estimate signal delay.

Inaccuracies in delay estimation are not only harmful to technology projections, but can also damage performance-driven routing methods that try to optimize interconnect segment length, width, spacing, and repeater/buffer sizing, etc. based on analytic delay formulas. Our study quantifies the impact of using analytic threshold delay formulas derived from RLC line models as opposed to RC line models.

3.1.1 RLC Delay Modeling

Inductance has a larger impact on inductive noise peak and indirectly affects the capacitive coupling noise peak because the slew times at all the nodes of the wire are faster when the line is modeled as RLC. Inductance is calculated based on expressions from [24,25] and the partial inductance concept [26]. We focus on analytical RLC interconnect delay models because their continuous, closed-form nature is well suited to modern iterative-improvement interconnect design methodologies and global optimization techniques. Gate delay is computed separately using a Thevenin model with voltage source and source resistance corresponding to the driver, and the load is modeled with a capacitance.

The two-pole delay model we use in this study was originally presented in [27] and is briefly described here. The transfer function for the two-pole model is given by

$$H(s) = \frac{1}{1 + b_1 s + b_2 s^2} \tag{1}$$

The coefficients in this transfer function are given by:

$$b_1 = R_S C + R_S C_L + \frac{RC}{2} + RC_L$$
$$b_2 = R_S \frac{C^2}{6} + \frac{R_S RCC_L}{2} + \frac{RC^2}{24} + \frac{R^2 CC_L}{6} + \frac{LC}{2} + LC_L \tag{2}$$

where R_S is source resistance, C_L is load capacitance at the end of the line, and R, C, L are the total electrical characteristics of the line. When the input at the source is modeled as a step input the output response is computed separately for the underdamped and overdamped cases.

We have implemented three different interconnect models and compared with SPICE results. Figure 4 shows the results with

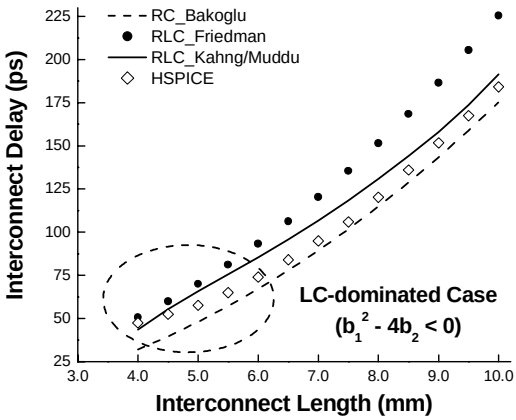

Figure 4. Comparison of RC/RLC delay models

varying line lengths. Line width is fixed at 1μm. The driver size and receiver size are fixed at $(W_p, W_n)=(54,18)$μm. For long wire lengths or for narrow line widths, the line tends to be more resistive (RC dominant), and Bakoglu's RC model produces results closely matching with SPICE. However, when delay is more LC dominated (i.e., large inductance value), the RC model underestimates delay by more than 10%. Friedman's model [28] matches well with SPICE for LC-dominated cases but overestimates delay by up to 30% in RC-dominated cases. Finally, the two-pole model in [27] described above matches SPICE for both RC and RLC cases within 10% error. (Given its acceptable accuracy, we use the two-pole model for subsequent studies below.) Note that with increasing line length, the 2-pole model changes from the complex pole case (overdamped or LC-dominated) to the real pole case (underdamped or RC-dominated). The condition to determine the case is from $b_1^2 - 4b_2 > 0$ (real poles) or ≤ 0 (complex or double poles).

We also study the reduction of threshold delay by controlling overshoot/undershoot of the voltage response. Typically, circuit design guidelines will define the amount of overshoot and undershoot allowed in a response. These can be translated into a condition between the first and second moments of the interconnect transfer function, which are in turn functions of driver and interconnect parameters. As shown in Table 2, undershoot conditions in 0.18μm technology can be easily avoided with proper repeater sizing and by providing reasonable signal return paths.

Table 2. Undershoot voltage normalized to Vdd with varying drive strengths and return path distances; width=2μm

Repeater Size (Wn/Ln)	Return path distance (μm)			
	25	50	100	150
200	0.0004	0.008	0.021	0.029
300	0.0098	0.035	0.061	0.074
400	0.0262	0.062	0.093	0.108
500	0.0420	0.083	0.116	0.132

3.1.2 Shielding Topologies

Shielding is an important technique that designers can leverage to maximize interconnect performance at the cost of increased routing area [29]. By inserting ground and V_{dd} shield wires, current return paths can be clearly defined and loop inductance can be reduced

compared to cases without explicit shielding. The extreme case of shielding is described in [30] where every signal wire has a ground and V_{dd} wire as its two nearest neighbors. In this study, we seek to minimize the cost of a design while achieving good performance. The width of the shield wires (W_{shield}) and signal wires (W_{sig}), the spacing between signal wires (S_{sig}), and the spacing from signal to neighboring shield wires (S_{shield}) are all parameters in this study. We examine the following three scenarios:

- No shielding (NS) – all current returns through a regular power grid. Wiring pitch is equal to ($W_{sig} + S_{sig}$).
- Single shielding (1S) – each signal wire has one shield wire as a nearest neighbor, while the other neighbor is another signal wire. If signal wires are denoted by S and shield (ground) wires by G, the order is G-S-S-G-S-S-G-S-S-G. Wiring pitch is ($2S_{shield} + S_{sig} + 2W_{sig} + W_{shield}$)/2.
- Double shielding (2S) – signal and shield wires alternate. This case is identical to the dense wiring fabric in [30]. Wiring pitch is ($W_{sig} + W_{shield} + 2S_{shield}$).

The cost function is defined as the product of wiring pitch, repeater sizing factor, and the number of repeaters inserted in the path. We attempt to minimize this cost function based on the following constraints:

1. Maximum delay is set at 1 ns and calculated according to each of the three delay models we have implemented.
2. Peak noise is fixed at 20% of V_{dd} and calculated based on the exponential model in [31].
3. Delay uncertainty is constrained and defined to be the difference between the RC (2-pole) and RLC delays.
4. The maximum allowable input slew time is 0.5 ns.

Using these constraints, we can examine the impact of shielding topology on circuit performance via coupling capacitance (constraints 2,4) and inductance (constraints 1–3). Recall that switch factors account for the capacitive Miller effect – the impact of neighboring wires switching in the same (opposite) directions can be modeled by lumping their coupling capacitances to ground and multiplying by some switch factor. Switch factors are 1, 2, and 3 in this study [32].

We sweep repeater size, number of repeaters, W_{sig}, and S_{sig} to find the minimal layout cost while meeting the above constraints. We also set W_{shield} to $2W_{sig}$ and S_{shield} equal to S_{sig} to reduce the total number of variables. Results are presented in Table 3, which shows the achievable cost (in arbitrary units) with varying switch factors and delay models. The 2S case can yield the minimal cost when a high switching factor is used. This is true in both RC models – these two models show very similar results from the optimization runs. The RLC model gives the overall best-cost results. Also, the slew time constraint can be more easily met if inductive effects are accounted for. The third constraint described above turns out to be a limiting factor for many input combinations – we find that RLC delay uncertainty is within bounds for smaller repeater sizes and for the 1S and 2S cases where inductance is small due to nearby current return paths.

3.2 Design Optimization Studies

In this subsection, we introduce a number of techniques to optimize the use of repeaters in critical paths. Models are developed and used to account for many effects that are currently dealt with in an *ad hoc* manner.

Table 3. Cost function comparison for varying switch factors, delay models, and shielding scenarios.

Model	Shielding	SF = 1	SF = 2	SF = 3
RC, 1 pole	NS	3.45	5.75	8.75
	1S	5.55	7.4	9.25
	2S	7.65	7.65	7.65
RC, 2 pole	NS	3.45	6.25	9.0
	1S	5.55	7.4	9.25
	2S	7.65	7.65	7.65
RLC	NS	2.85	4.6	6.75
	1S	5.1	6	7.4
	2S	7.05	7.05	7.05

3.2.1 Wire Sizing

We next turn to the impact of wire sizing on important design metrics such as delay, noise, and cost. We begin with an expression for optimal wire width as a function of line length, l, from [20]:

$$W_{opt}(l) = \sqrt{\frac{R_{sh}(C_f l + 2C_L)}{2R_D C_a}} \quad (3)$$

Here C_a and C_f denote the area and fringing capacitances per unit length, R_{sh} is the sheet resistance, R_D is the driver resistance, and C_L is the load capacitance at the end of the line.[3] We first examine the impact of line spacing on optimal wire width by changing spacing from 0.5 to 2 μm – Figure 5 plots the optimal line width + spacing for a 1.5 mm line, versus spacing alone on the x-axis. This plot shows an inflection point for switch factors 3 and 2, which corresponds to the optimal *pitch*, not just the optimal line width.

Nominal and optimistic switch factors may have such an inflection point, but they do not fall in the design space of the process technology. Figure 5 uses Equation (3) to calculate optimal line width.

We compare line widths obtained using Equation (3) to the optimal line widths as found by sweeping the line width in GTX, for a range of driver and interconnect topologies. In addition, we incorporate inductance into the delay expressions and again perform exhaustive sweeping to find optimal line widths based on minimizing RLC as well as RC stage delay. As shown in Figure 6, our results demonstrate that (3) matches the GTX results within 10% and often less than 5% error. However, the presence of inductance causes the optimal line width to shrink substantially and (3) therefore overestimates W_{opt} for RLC lines. Also, increasing repeater size leads to a rise in W_{opt} for all models studied – expression (3) shows slightly more error for larger drivers.

3.2.2 Repeater Sizing

The most commonly cited optimal buffer sizing expression is that of Bakoglu [10]:

[3] Fringing capacitance is taken as the difference between the total line capacitance and the parallel-plate capacitance from [36].

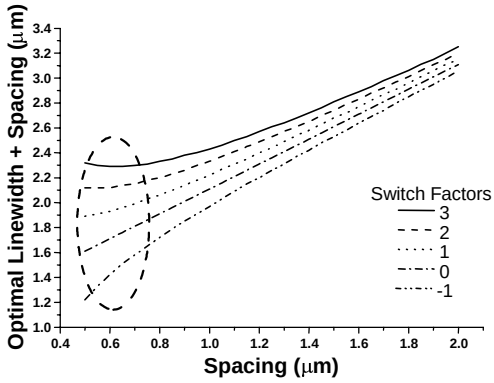

Figure 5. Translation from optimal line width to optimal pitch demonstrates inflection points for certain switch factors.

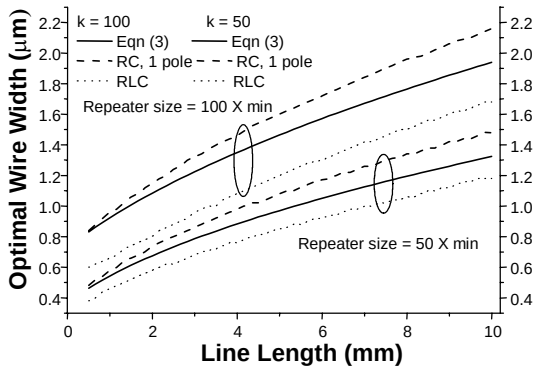

Figure 6. Optimal wire width expression (Eq. 3) exhibits 30% error with respect to RLC model, less error compared to RC.

$$S = \sqrt{\frac{R_D C_{int}}{R_{int} C_{in}}} \quad (4)$$

R_D reflects the minimum-sized driver resistance, C_{in} is the input gate capacitance of a minimum-sized inverter, and R_{int} and C_{int} are respectively the line resistance and capacitance per unit length. Although this expression can give accurate results in some cases when optimizing for delay only, the delay vs. device size relationship lends itself to further optimization due to its insensitivity near the optimal point. Results obtained from Equation (4) are often unrealistically large – typical standard cell libraries may include inverters or buffers up to 54-96X the minimum size (W_n=L_{drawn}) whereas (4) can give results in the range of 400-700X minimum. To compensate for this, an expression was derived in [17] to optimize a weighted delay-area product rather than purely delay – it gave results on the order of 50-60% smaller than (4). Even with this modification, however, so-called "optimal repeater sizes" seem impractical in the face of power and area constraints.

Here and in the remainder of the subsection, we present a more experimental approach to finding optimal repeater size. For various wire geometries, noise conditions, area and placement constraints and delay models, we develop a complete picture of the optimal repeater topology solution. We begin with a simple sweep of the repeater size for a single stage of a chain, and examine both delay and energy-delay product vs. repeater size in Figure 7.

As Figure 7 shows, the optimal buffer sizing as calculated from (4) is 480 times the minimum-sized inverter. From pure delay analysis, GTX optimization results indicate that the ideal buffer size for our standard critical path is ~140-150 times the minimum size. When

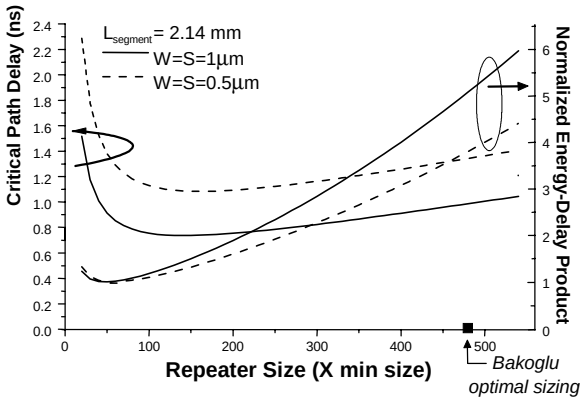

Figure 7. This plot clearly demonstrates the severe oversizing resulting from simple expressions such as Eq. (4).

optimizing the energy-delay product, that value drops all the way to 50-60 times minimum. Any range of weighting functions can be easily incorporated into the rule chains – for instance, (energy-delay)^2 or (energy-delay)^3. Results from such functions are not included here, but will push the optimal size towards the delay-only size of 140-150 times minimum. It is also important to note from Figure 7 that the path delay function around the delay-optimal repeater size is very flat: a buffer which is 43% smaller than optimal yields only a 6.8% delay penalty. Since the energy-delay optimal size is found in the steep part of the delay curve, a truly ideal choice would more closely reflect the knee of the delay curve. In the case of Figure 7, our choice of "optimal repeater size" is in the range of 80-100 times the minimum inverter size.

3.2.3 Repeater Placement Uncertainty

The placement of repeaters in a deep submicron design is non-trivial – many thousands of repeaters must be used to meet timing and noise objectives, and this number will increase with process scaling. As a result, the area consumed by these buffers is substantial and may no longer be ignored during the floorplanning design phase. Particularly in a hierarchical design methodology, such as that proposed in [33], it may not be possible to place repeaters at any given location either inside a pre-designed block or at the top-level of the hierarchy. A potential solution to this problem involves the formation of repeater block regions located around the chip at the floorplanning stage that provide specified areas for repeaters to be placed [34]. However, with such an approach the feasible distances between repeaters are discrete, not continuous.

Here, we study the impact on critical path delay of this inability to place repeaters at arbitrary locations. As before, we examine a top-level metal 1.5 cm route in the default technology. We define an uncertainty parameter, ε, which can range from 0 (no uncertainty) to 1 (maximum uncertainty). We express the location uncertainty as $(1\pm \varepsilon)\ L_{seg}$ where L_{seg} is the nominal distance between repeaters when there are no placement restrictions. Given these bounds on segment length between consecutive buffers, we examine the worst-case scenario when half of the segments in the critical path have length $(1-\varepsilon)\ L_{seg}$ and the other half are of length $(1+\varepsilon)\ L_{seg}$ while total path length is fixed. Given uniform buffer sizing, half of these segments will be overdriven while the other half are underdriven.

While sweeping ε, we vary the switch factor and plot the path delay and peak noise normalized to the ε = 0 case. Results shown in Figure 8 indicate that the impact of repeater placement uncertainty

is small for total path delay but large for peak noise. This can be understood by realizing that the path delay effectively averages out the resulting fast and slow stages while peak noise is a function of the segment length $(1\pm\varepsilon)\,L_{seg}$ and not the total path length. Since the peak noise results are normalized to the $\varepsilon = 0$ case, the switching factor does not play a major role. With a conservative ε of 0.3, the worst-case peak noise increases by approximately 30%.

3.2.4. Staggered Repeaters

The use of staggered repeaters for global buses was first described in [35]. The layout structure is shown in Figure 9. This approach uses offset buffers in a bus-like structure to minimize the impact of coupling capacitance on delay and crosstalk noise. If repeaters are offset so that each gate is placed in the middle of its neighboring gates' interconnect loads, the effective switching factor is limited to one. This is because potential worst-case simultaneous switching on adjacent wires can be present for only half the victim line's length, and in such conditions the other half of the victim line will consequently experience best case neighboring switching activity.

In our analysis, we examine the potential reduction in delay uncertainty, as well as in peak crosstalk noise, due to staggered repeaters. Figure 10 shows that the noise reductions can easily be greater than 10% of V_{dd} for realistic spacing and switch factors. The delay uncertainty when using non-staggered repeaters can exceed 50% of the nominal delay – but staggering almost completely eliminates this uncertainty that stems from capacitive coupling.

IV. CONCLUSIONS

A priori interconnect prediction is a key enabler for technology extrapolation engines. In addition, the rapid evolution of technology and prediction methods leads to a need for dynamic and flexible prediction frameworks. This paper outlines one such framework that aims at building upon a priori interconnect prediction and comprehensive interconnect models to quantify the available performance envelope for future IC design as well as to investigate the impact of various design optimization strategies in these designs. We have described the architecture and implementation of GTX, the MARCO GSRC Technology Extrapolation system. GTX has the potential to change how we extrapolate the impact of new process and design technology: it can provide a "living roadmap" that incorporates - and serves as a repository for - essentially unlimited forms of domain knowledge. The structure of GTX (with the flexibility of the ASCII rules) is especially useful for including

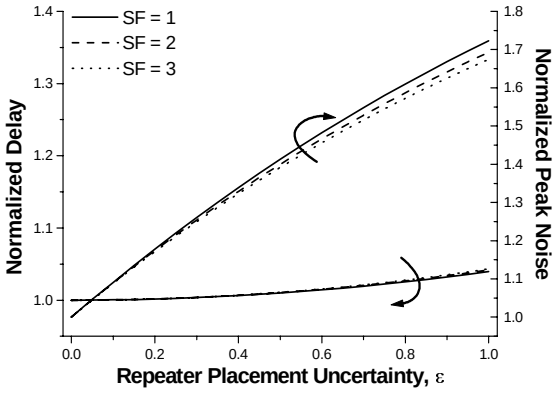

Figure 8. Repeater placement uncertainty has a large impact on noise but little on delay.

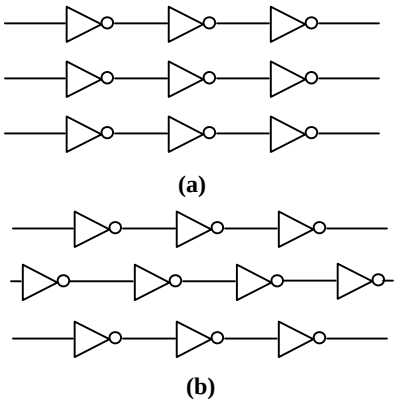

Figure 9. Reduction of worst-case Miller coupling by staggered repeaters. In usual layouts (a), inverters on left and right neighbors are at phase=0 with respect to the inverters on middle line. Staggering (b) places inverters on left and right neighbors at phase=0.5.

a priori interconnect prediction techniques and for keeping the model database up-to-date in this rapidly evolving field.

The combination of a priori interconnect prediction techniques with a highly flexible technology extrapolation system such as GTX (together with the addition of models for process and design technology), enables a quick and easy evaluation of modeling choices through a new set of studies of interconnect properties. Awareness of the latest interconnect optimizations and design degrees of freedom represent "new dimensions" in a priori interconnect prediction. In this paper, we have used GTX to examine the topics of RLC delay modeling, wire shielding, optimal repeater and wire sizing, repeater staggering, and repeater placement uncertainty effects. We demonstrated that when including inductance, errors in estimates of optimal line delay could increase up to 30%, implying that an RLC-based model could be necessary. A closed-form wire sizing expression was evaluated and found to yield good results compared to a 1-pole RC delay model, but more substantial error compared with an RLC model. We also found that conventional models for optimal repeater sizing are insufficient – our examples show significant overestimation of repeater size up to 500%. A more effective sizing criterion would weight energy and delay so that the size closely approximates the knee of Figure 4. We have also modeled the impact of repeater staggering (a layout technique which limits delay uncertainty and peak noise due to capacitive Miller effect).

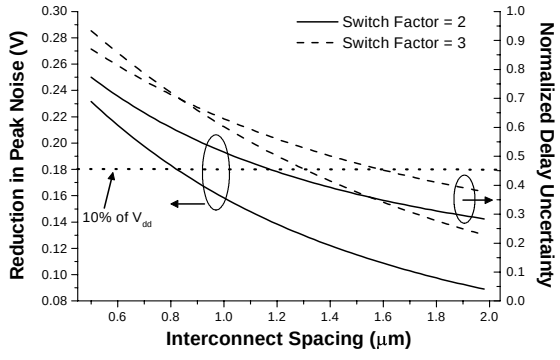

Figure 10. Staggered repeater topology substantially reduces peak noise and delay uncertainty compared to traditional case.

In summary, we have identified the importance of including a wide range of circuit optimization techniques into predictions of interconnect performance in technology extrapolation. GTX strives to bring about this awareness by allowing for more detailed interconnect modeling and optimization degrees of freedom in its analysis. Using GTX, we demonstrated a wide range of modeling and parameter sensitivities in existing extrapolation models – this isolates the most critical modeling needs in extrapolation engines. Bringing these contributions together, we see that while technology extrapolation engines such as GTX rely heavily on a priori interconnect predictions, these estimation methods also require substantial augmentation to develop a truly useful design prediction framework.

REFERENCES

[1] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Trans. on Comput.*, C-20 (1971), pp. 1469-1479.

[2] P. Christie and D. Stroobandt, "The interpretation and application of Rent's rule," *IEEE Trans. on VLSI Systems, special issue on SLIP* 8 (6) (2000), pp. 639-648.

[3] W.E. Donath, "Placement and average interconnection lengths of computer logic," *IEEE Trans. Circ. & Syst.* 26 (1979), pp. 272-277.

[4] D. Stroobandt and J. Van Campenhout, "Estimating interconnection lengths in three-dimensional computer systems," *IEICE Trans. on Inf. & Syst., Special Issue on Synthesis and Verification of Hardware Design*, E80-D (10) (1997), pp. 1024-1031.

[5] D. Stroobandt and J. Van Campenhout, "Accurate interconnection length estimations for predictions early in the design cycle," *VLSI Design, Special Issue on Physical Design in Deep Submicron*, 10 (1) (1999), pp. 1-20.

[6] J.A. Davis, V.K. De and J.D. Meindl, "A stochastic wire-length distribution for gigascale integration – Part I: derivation and validation," *IEEE Transactions on Electron Devices*, 45 (3) (1998), pp. 580-589.

[7] D. Stroobandt and A.B. Kahng, "Workshop Notes of the 1st Intl. Workshop on System-Level Interconnect Prediction," (1999). Notes available at http://www.elis.rug.ac.be/~dstr/SLIP.html

[8] D. Stroobandt, "Recent advances in system-level interconnect prediction," *IEEE CAS. Society Newsletter* 11 (4) (2000), pp. 1;4-20;48.

[9] D. Stroobandt, *A priori wire length estimates for digital design*. Kluwer Academic Publishers, April 2001.

[10] H.B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, 1990.

[11] G.A. Sai-Halasz, "Performance Trends in High-Performance Processors," *Proc. IEEE*, Jan. 1995, pp. 20-36.

[12] P. Raje, "A Framework for Insight into the Impact of Interconnect on 0.35μm VLSI Performance", *Hewlett-Packard J.*, 1995, pp. 1-8.

[13] J.C. Eble, V.K. De, D.S. Wills and J.D. Meindl, "A Generic System Simulator (GENESYS) for ASIC Technology and Architecture Beyond 2001," *Proc. ASIC*, 1996, pp. 193-196.

[14] B.M. Geuskens and K. Rose, *Modeling Microprocessor Performance*, Kluwer Academic Publishers, 1998. Rensselaer Interconnect Performance Estimator website: http://latte.cie.rpi.edu/ripe.html

[15] R. Mangaser, C. Mark and K. Rose, "Interconnect constraints on BEOL manufacturing," *Proc. Advanced Semiconductor Manufacturing Conference*, 1999, pp. 304-308.

[16] C. Mark and K. Rose, "Interconnect strategies for deep submicron CMOS manufacture," *Proc. Advanced Semiconductor Manufacturing Conference*, 2000, pp. 413-418.

[17] D. Sylvester and K. Keutzer, "System-Level Performance Modeling with BACPAC – Berkeley Advanced Chip Performance Calculator,"*Proc.SLIP*,1999,pp.109-114, http://www-device.eecs.berkeley.edu/~dennis/bacpac/

[18] International Technology Roadmap for Semiconductors," December 1999, http://www.itrs.net/

[19] P. D. Fisher and R. Nesbitt, "The Test of Time: Clock-Cycle Estimation and Test Challenges for Future Microprocessors," *IEEE Circuits and Devices Magazine* 14(2) 1998, pp. 37-44.

[20] J. Cong and D.Z. Pan, "Interconnect Estimation and Planning for Deep Submicron Designs," *Proc. Design Automation Conference*, 1999, pp. 507-510.

[21] S. Takahashi, M. Edahiro and Y. Hayashi, "A New LSI Performance Prediction Model for Interconnection Analysis of Future LSIs", *Proc. ASP-Design Automation Conference*, 1998, pp. 51-56.

[22] A E. Caldwell *et al*., "GTX: The MARCO GSRC Technology Extrapolation System," *Proc. Design Automation Conference*, 2000, pp. 693-698.

[23] http://vlsicad.cs.ucla.edu/GSRC/GTX/

[24] L. He, N. Chang, S. Lin, and O.S. Nakagawa, "An Efficient Inductance Modeling for On-Chip Interconnects," *Proc. Custom Integrated Circuits Conference*, 1999, pp. 457-460.

[25] X. Qi *et al*., "On-Chip Inductance Modeling and RLC Extraction of VLSI Interconnects for Circuit Simulation," *Proc. Custom Integrated Circuits Conference*, 2000.

[26] A. E. Ruehli, "Inductance calculations in a complex integrated circuit environment," *IBM J. Res. Dev.*, September 1972, pp. 470-480.

[27] A.B. Kahng and S. Muddu, "An analytical delay model for RLC interconnects," *IEEE Trans. CAD* 16(12) (1997), pp. 1507-1514.

[28] Y.I. Ismail, E. G. Friedman, J.L. Neves, "Equivalent Elmore delay for RLC trees", *IEEE Trans. CAD* 19(1) (2000), pp. 83-97.

[29] Y. Massoud, S. Majors, T. Bustami and J. White, "Layout Techniques for Minimizing On-Chip Interconnect Self-Inductance," *Proc. Design Automation Conference*, 1998, pp. 566-571.

[30] S. P. Khatri, A. Mehrotra, R. K. Brayton, A. Sangiovanni-Vincentelli, and R.H.J.M. Otten, "A Novel VLSI Layout Fabric for Deep Submicron Applications," *Proc. Design Automation Conference*, 1999, pp. 491-496.

[31] K. L. Shepard *et al*., "Design Methodology for the S/390 Parallel Enterprise Server G4 Microprocessors," *IBM J. Res. Dev.*, July-Sept. 1997, pp. 515-554.

[32] A.B. Kahng, S. Muddu and E. Sarto, "On Switch Factor Based Analysis of Coupled RC Interconnects," *Proc. Design Automation Conference*, 2000, pp. 79-84.

[33] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron," *Proc. International Conference on Computer-Aided Design*, 1998, pp. 203-211.

[34] J. Cong, T. Kong and D. Z. Pan, "Buffer Block Planning for Interconnect-Driven Floorplanning," *Proc. International Conference on Computer-Aided Design*, 1999, pp. 358-363.

[35] A.B. Kahng, S. Muddu and E. Sarto, "Tuning Strategies for Global Interconnects in High-Performance Deep Submicron IC's," *VLSI Design* 10(1), 1999, pp. 21-34.

[36] T. Sakurai, "Closed-form Expressions for Interconnect Delay, Crosstalk, and Coupling in VLSIs," *IEEE Transactions on Electron Devices*, pp. 118-124, Jan. 1993.