MODELING AND ANALYSIS OF LARGE-SCALE ON-CHIP INTERCONNECTS

A Dissertation

by

ZHUO FENG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2009

Major Subject: Computer Engineering

MODELING AND ANALYSIS OF LARGE-SCALE ON-CHIP INTERCONNECTS

A Dissertation

by

ZHUO FENG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Peng Li
Committee Members,	Jiang Hu
	Jianhua Huang
	Jose Silva-Martinez
Head of Department,	Costas N. Georghiades

December 2009

Major Subject: Computer Engineering

ABSTRACT

Modeling and Analysis of Large-scale On-chip Interconnects. (December 2009)

Zhuo Feng, B.Eng., Xi'an Jiaotong University, China;

M.Eng., National University of Singapore, Singapore

Chair of Advisory Committee: Dr. Peng Li

As IC technologies scale to the nanometer regime, efficient and accurate modeling and analysis of VLSI systems with billions of transistors and interconnects becomes increasingly critical and difficult. VLSI systems impacted by the increasingly high dimensional process-voltage-temperature (PVT) variations demand much more modeling and analysis efforts than ever before, while the analysis of large scale on-chip interconnects that requires solving tens of millions of unknowns imposes great challenges in computer aided design areas. This dissertation presents new methodologies for addressing the above two important challenging issues for large scale on-chip interconnect modeling and analysis:

- In the past, the standard statistical circuit modeling techniques usually employ principal component analysis (PCA) and its variants to reduce the parameter dimensionality. Although widely adopted, these techniques can be very limited since parameter dimension reduction is achieved by merely considering the statistical distributions of the controlling parameters but neglecting the important correspondence between these parameters and the circuit performances (responses) under modeling. This dissertation presents a variety of performance-oriented parameter dimension reduction methods that can lead to more than one order of magnitude parameter reduction for a variety of VLSI circuit modeling and analysis problems.

- The sheer size of present day power/ground distribution networks makes their analysis and verification tasks extremely runtime and memory inefficient, and at the same time, limits the extent to which these networks can be optimized. Given today's commodity graphics processing units (GPUs) that can deliver more than 500 GFlops (Flops: floating point operations per second). computing power and 100GB/s memory bandwidth, which are more than 10X greater than offered by modern day general-purpose quad-core microprocessors, it is very desirable to convert the impressive GPU computing power to usable design automation tools for VLSI verification. In this dissertation, for the first time, we show how to exploit recent massively parallel single-instruction multiple-thread (SIMT) based graphics processing unit (GPU) platforms to tackle power grid analysis with very promising performance. Our GPU based network analyzer is capable of solving tens of millions of power grid nodes in just a few seconds. Additionally, with the above GPU based simulation framework, more challenging three-dimensional full-chip thermal analysis can be solved in a much more efficient way than ever before.

To my parents and my wife, Ting

ACKNOWLEDGMENTS

My sincere gratitude goes to my Ph.D. advisor, Professor Peng Li. Without his guidance and support throughout the long and challenging journey, most ideas presented in this dissertation would not have been accomplished. His diligence, intelligence and wisdom have had great influence on my research development. Research directions, technical criticisms as well as valuable feedback from innumerable office discussions, group meetings, and seminar presentations brought me continuous sources of motivation and inspiration, forming the indispensable parts of my academic life in the past four years.

I would like to thank Dr. Zhuoxiang Ren and Dr. Hui Zheng, who served as my mentors during two summer research internships at Mentor Graphics Inc. and Magama Design Automation in 2007 and 2008. Dr. Ren helped originate the design-specific interconnect corner extraction technique while Dr. Zheng encouraged me to have the first try on the hardware acceleration of the SPICE circuit simulator. These internships have brought deeper understandings of the practical needs in industrial applications. I have to thank Dr. Frank Liu and Dr. Sani Nassif at IBM Austin Research Laboratory, who provided important benchmarks and insights for my power grid research project.

I also owe a lot to Professor Jiang Hu, Professor Jianhua Huang and Professor Jose Silva-Martinez, who have been serving as my Ph.D. committee members and providing important feedback during my Ph.D. study. Without Professor Hu's patient guidance, I would not have understood the fundamental physical design problems and latest research activities, while Professor Huang and Professor Silva brought valuable insight to the statistical VLSI circuit design/modeling problems.

I would also like to thank many other people in the Computer Engineering group

at Texas A&M University, who have contributed to this dissertation by providing technical discussions, testing benchmark circuits, and simulation codes: Guo (Hugo) Yu helped design the first few analog circuit test cases for my statistical circuit modeling projects, and his expertise provides valuable insight on improving the statistical parameter dimension reduction algorithms. Numerous discussions with Xiaoji Ye on model order reduction methods and clock tree simulations were very helpful for developing some of my ongoing research projects. Xiaoji also contributed the quadratic interconnect moment sensitivity derivations/codes for the corner extraction program. Wei Dong has strengthened my understanding of the RF and parallel circuit simulation techniques. Zhiyu (Albert) Zeng is always energetic and passionate on discussing the power grid optimization/analysis problems, and he has spent a very long time on implementing and improving the package-die power grid simulation program. Shiyan Hu gave me the first few lectures on physical design research related algorithms. Zhanyuan Jiang helped me so much seeking job opportunities and provided valuable information during this great economic crisis. Yang Yi helped me understand the fundamental and advanced parasitics extraction techniques and the package level circuit simulation issues.

Without the help of many of my friends, I would not have led such a fruitful and joyful life in Texas A&M University. So a lot of my thanks go to my close friend, Huifeng Li, who has been my six year classmate in both the National University of Singapore and Texas A&M University since 2003. He is also an expert in electromagnetics as well as semiconductor processing, who helped me understand some very complicated research problems. I would also like to thank Yifang Liu, Zheng Wang, Zhongwei Jiang, Hang Su, Qinghe Du, Xiaojian Wu, Yong Zhang, Leyi Yin,..., for the numerous discussions and conversations that constitute indispensable parts of daily activities in the CE group.

Over the past four years, most of my research projects have been generously funded by the Semiconductor Research Corporation (SRC). I really appreciate the opportunity of being a member of the SRC family and the experience of participating the many conferences and funding reviews that make me more visible to the industry and the academic world, knowing a lot of kind and professional people and bringing me numerous ideas that lead to new innovations.

Finally, I appreciate so much the help and support from my parents and my wife, Ting. Without their constant support and encouragement, I could not have accomplished the research projects so well. My six years of study in the US and Singapore has brought hard times to my parents, since the only communications we have are through the internet and phone calls. In the past three years, I got a lot of support and help spiritually, mentally and emotionally from my wife, Ting, who is always standing behind me. She has absolutely been playing the most important and difficult role in the whole family during my Ph.D. study, so my deepest thanks go to her: Wishing us forever love and happiness in the universe.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

TABLE Page

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION AND ORGANIZATION

A. Statistical On-Chip Interconnect Modeling and Analysis Using Parameter Dimension Reductions

As CMOS technologies continuously scale down, predicting the circuit performance variations due to various process variations becomes increasingly critical and difficult [1]. While the growing magnitude of process variations pushes for more complex parametric models that are capable to capture the nonlinear effects of these process variations, the dramatically increasing sources of process variability further impose a formidable high-dimensional parameter space in which a given design have to be verified and optimized. Curse of dimensionality impacts a wide range of CAD problems since the feasibility as well as the efficiency of many CAD algorithms critically depend on the dimension of the parameter space:

- The cost and complexity of many empirical macromodeling techniques (e.g. RSM based performance modeling) grow exponentially in the number of parameters [2, 3]. For example, to extract a model with 20 parameters by a level-two RSM technique will require some fraction of 2^{20} data, which is very expensive. Practically, building RSM models with hundreds or thousands parameters is simply infeasible.

- The same issue appears in a large body of more formal parameterized interconnect reduced order modeling algorithms and variational analysis techniques developed for capturing interconnect variability [4, 5, 6, 7, 8, 9, 10]. The high di-

---

This dissertation follows the styles of *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

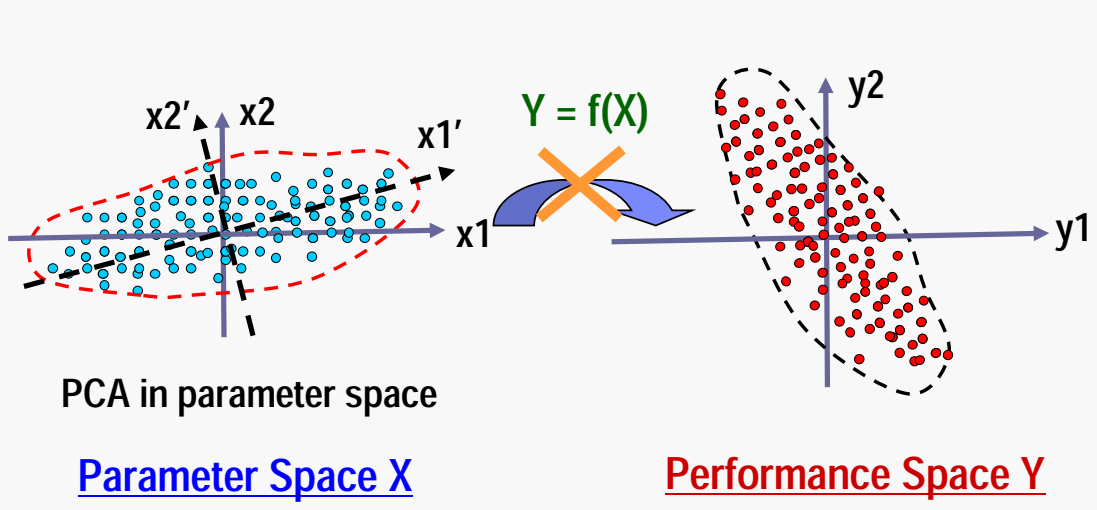

Fig. 1. Limitation of PCA.

mensional parameter space makes such model order reduction extremely difficult and complex. For example, the size of the reduced model generated by algorithm [8] exponentially increases with the number of parameters. Although it has been observed in [6] that the projection subspace used in model reduction does not grow fast in the number of process variations, identifying such low-dimensional subspace, however, requires expensive samplings in high-dimensional parameter space.

In the CAD community, the standard practice employs PCA (principal component analysis) and its variants for parameter reduction [11, 12, 13]. Although widely adopted, these techniques are limited since parameter reduction is achieved by only considering the statistics of the controlling parameters while neglecting the important correspondence between these parameters and circuit performances under modeling (as shown in Fig. 1). Parameter screening has also been applied under the context of response surface modeling [2], however, the technique is empirical in nature as it prunes parameters one at a time based on sensitivity-like measures.

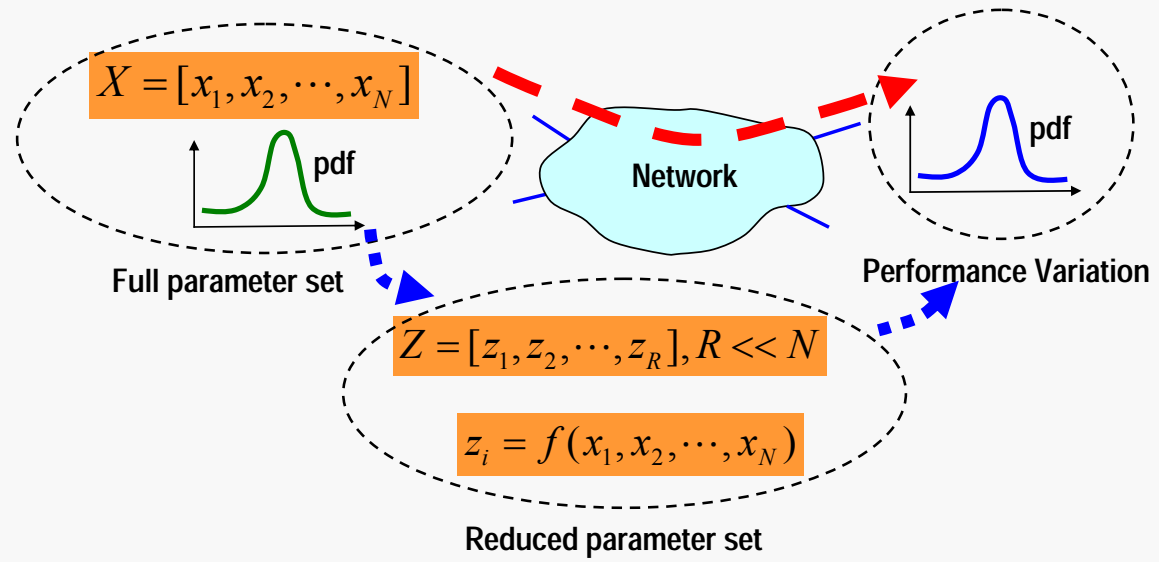

Fig. 2. Performance-oriented parameter reduction.

Given the fact that systematic CAD specific parameter reduction methodologies are lacking, Chapter II of this dissertation presents a performance-oriented parameter dimension reduction framework that can significant reduces the modeling and verification cost (as shown in Fig. 2). More specifically, our contributions include:

1. Introduced the linear reduced rank regression framework for linear interconnect circuit parameter dimension reductions;

2. Proposed nonlinear iteration-based reduced rank regression method to perform parameter dimension reductions for nonlinear transistor circuit components;

3. Proposed nonlinear moment-based parameter reduction method for the given quadratic performance models.

The above linear and nonlinear parameter dimension reduction methods have been utilized in a variety of statistical circuit modeling and analysis applications:

1. Very compact parametric interconnect models can be obtained more efficiently

than ever before, which will significantly reduce the computational cost of the parameterized model order reductions (PMOR);

2. Statistical design-dependent interconnect corner can be extracted in a more efficient way, avoiding excessive model generation cost as well as corner finding efforts;

3. Prior second-order statistical static timing analysis (SSTA) algorithm has been extended to capture 10X more local and global variation sources than before.

B. Hardware Acceleration of Large Scale On-Chip Power Grid Analysis

Power grid analysis requires solving very large scale linear equations with tens of millions of unknowns. In the past decade, on the standard general-purpose CPU platform, a body of power grid analysis methods have been proposed [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25] with various tradeoffs. The previous methods fall into the following categories:

- Direct methods such as LU factorization and Cholesky decompositions [18, 24] produce the most accurate results, at the cost of high runtime and memory consumption. Typically, the runtime and memory of direct methods can increase super-linearly in the problem sizes, which limits the methods to solve up to only a few million unknowns. For instance, the state-of-art direct solver Cholmod takes around 8 Gb memory and 1,000 seconds for solving a nine-million 2D grid problem.

- Iterative methods such as preconditioned conjugate gradient (PCG) [14, 25], multigrid (MG) method [15, 17] and the classic first or second order iterative methods [20, 21] are memory efficient, but may face with slow convergence prob-

lems. For instance, PCG methods usually require fairly good pre-conditioners for fast convergence that is very difficult to obtain, the classic iterative method requires good initial solution guess as well as an accurate estimate of the spatial radius of the grid, while the multigrid methods typically rely on good coarse level grid approximations as well as efficient and effective smoothers to damp out the error components.

- Stochastic method such as random walk [22, 26] can be efficient for solving a very small portion of a grid design. However, the accuracy level is hard to predict and a satisfactory results may require a huge number of random walks that leads to much longer runtime than other methods.

Recently, the emergence of massively parallel single-instruction multiple-data (SIMD), or more precisely, single-instruction multiple-thread (SIMT) [27], based GPU platforms offers a promising opportunity to address the challenges in large scale power grid analysis. Today's commodity GPUs can deliver more than 500 GFlops [1] of theoretical computing power and 100GB/s off-chip memory bandwidth, which are 10X greater than offered by modern day general-purpose quad-core microprocessors [27]. The ongoing GPU performance scaling trend justifies the development of a suitable subset of CAD applications on such platform.

In Chapter III of this dissertation, a GPU accelerated power grid solver is presented for very fast power grid analysis. Our contributions include:

1. Proposed GPU-friendly Jacobi and Gauss Seidel iteration schemes that can achieve a performance of more than 100 Gflops on a 128-core GPU hardware (theoretical peak performance for the hardware is 500 GFlops), which are more than 100X faster than the CPU based computations;

[1] Flops: floating point operations per second.

2. Proposed a GPU-friendly geometrical multigrid (GMD) solver for fast on-chip power grid analysis;

3. Proposed a hybrid multigrid method (HMD) for solving irregular, multi-layer power supply networks which takes the most advantages of both the CPU and GPU computing platforms.

Designed as a general partial differential equation (PDE) solver, our powerful GPU based simulation engines can be also used in other VLSI design automation applications such as clock mesh simulation, power-gated circuit verification, as well as three dimensional full-chip thermal analysis, etc.

CHAPTER II

STATISTICAL CIRCUIT MODELING USING PARAMETER DIMENSION REDUCTION

A. Statistical Parameter Dimension Reduction Methods

In this section, we first review the well-known parameter reduction technique using principal component analysis (PCA). Then we propose the performance-oriented parameter reduction while several important application issues are also discussed. Finally, we show how to implement the performance-based parameter reduction technique in practical circuit modeling problems.

1. PCA-Based Parameter Reduction

Traditional statistical analysis starts by transforming the correlated Gaussian variables to uncorrelated ones using principal component analysis (PCA). Subsequently, the statistical performance distributions can be evaluated accordingly based on the new variables.

a. Procedures of PCA

PCA transforms the data set to new coordinates such that the largest variance of the data is projected onto the first few coordinates (also called the principal components). PCA can be used for dimension reduction for a data set by keeping those characteristics of the data set that contribute most to its variance. By keeping the most dominant principal components, the new data set obtained by PCA usually contains the "most important" aspects of the original data set.

Consider an n-dimensional data set $X \in \mathbb{R}^n$, which has zero mean and multivari-

ate normal distributions. After obtaining the covariance matrix Σ_{xx} of this data set, PCA first computes the eigen-decomposition of Σ_{xx}, which gives

$$\Sigma_{xx} = P\Lambda P^T, \tag{2.1}$$

where Λ is a diagonal matrix containing all the eigenvalues of Σ_{xx}, and P contains all the corresponding orthogonal eigenvectors. By including few eigenvectors (of P) that have the largest eigenvalues into the projection matrix P_r, the new parameter set X_r that has a smaller dimension than the original data set X can be obtained by

$$X_r = P_r^T X. \tag{2.2}$$

b. Limitations of PCA

The objective of statistical circuit analysis is to compute circuit performance variations. Under such context, PCA can be applied to improve the efficiency of circuit analysis by identifying the principle components of process variations that impact circuit performances. However, PCA removes the redundancy in the process variation data set without considering specific design performances. In practice, such design-independent parameter reduction reduces the cost of statistical circuit analysis, but in a limited way. To better understand the limitations of PCA, we show the following two examples.

- *DC Response Variation*: We first consider a resistance-capacitance (RC) circuit with a single voltage source input and no grounded resistors, representing the widely used on-chip RC interconnect model for timing analysis. We assume that the RC circuit is perturbed by the manufacturing fluctuations in the forms of wire width (W), thickness (T) and dielectric layer thickness (H) variations. It is well known that the DC voltage response of such circuit can be trivially

determined by the input voltage excitation regardless of any RC element variations. However, if one blindly applies PCA to reduce the dimension of RC variations for the purpose of modeling the DC circuit performance, one will fail to identify the trivial fact that the dimensionality of the variability of the DC performance is essentially zero.

- *Timing Performance Variation*: We consider the more useful issue of modeling the timing performance variations of the RC circuit. Suppose that variance of wire width (W) is greater than that of dielectric layer thickness (H), then a relevant question to ask is: which variation is more critical (statistically) in terms of the delay variability? Without taking any circuit information into account, PCA may just pick W since it has a larger variance. However, in terms of delay the W variation may not necessarily be a more dominant factor, since the increase in W leads to an increase in wire capacitance but also a decrease in wire resistance so that the delay may not be influenced much.

From the above examples, it can be understood that for different performances of interest (DC response variation or timing variation), PCA will provide the same parameter reduction results since only the statistical properties of the parameter set are considered during the reduction. In order to incorporate the output performance into the parameter reduction procedure and achieve a greater extent of parameter reduction, we propose the performance-oriented parameter reduction method as follows.

2. Performance-Based Parameter Reduction

Unlike the standard principal component analysis (PCA), our performance-based parameter reduction exploits not only the statistical properties of underlying process

parameters, but also the correlation between these parameters and circuit performances of interest, by incorporating and extending the theoretic framework of the linear reduced rank regression (RRR) [28].

a. Procedures of Linear Reduced Rank Regression (RRR)

We denote the original parameter set by $X \in \mathbb{R}^n$ and the dependent output performance set by $Y \in \mathbb{R}^m$. As before, we assume X has zero mean and multivariate normal distributions. It is assumed that the covariance matrix of Y and X is given by Σ_{yx} and the covariance matrix of X is Σ_{xx}. RRR finds the projection matrix by performing the eigen-decomposition of matrix $\Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{yx}^T$, instead of Σ_{xx} as in PCA. A few eigenvectors (from U) of the largest eigenvalues (from Λ) form the projection matrix U_r, which is used to get the reduced parameter set Z by

$$Z = U_r^T X. \tag{2.3}$$

Obviously, in the above dimension reduction procedure, RRR not only considers the statistical properties (Σ_{xx}) of the original parameter data set X, but also considers the correlation (Σ_{yx}) between the output performance Y and the parameter data set. As a result, parameter reduction via RRR is achieved while considering specific design information, which distinguishes itself from PCA-based dimension reduction method. In practice, for a given set of circuit performances, not all the parameter variations or the combinations of thereof will be equally important. Exploiting design information potentially leads to a higher degree of parameter reduction, hence brings additional benefits beside what PCA can offer.

b. Application Issues

After the brief introduction of the performance-based dimension reduction (using RRR), we discuss the following important aspects for practical applications of such techniques.

- *Robust Mathematical Framework*: The previous description of linear RRR provides a basic flavor of our proposed performance-oriented parameter reduction techniques. However, in reality, circuit performances may react nonlinearly to large range process variations. As will be described in this Chapter, we extend linear RRR to a more general nonlinear framework to facilitate more robust circuit modeling.

- *Smart Sample Collection*: To efficiently carry out the parameter dimension reduction under the RRR framework, the correlation between the output performances (Y) and the input parameters (X) must be obtained carefully. A large set of statistical parameter/performance data may be needed to compute the covariance Σ_{yx}. However, collecting such data through brute-force full-blown Monte-Carlo simulations is too expensive, which also defeats the purpose of the parameter reduction. Therefore, smart data collection techniques that avoid direct circuit simulations are necessary to realize efficient data collection.

- *Efficient Model Characterization*: After parameter reduction, parametric modeling can be much more efficiently obtained in the reduced parameter space. As such, parameter reduction can be considered as a pre-processing step of parameterized circuit modeling, though this characterization step is the not the main focus of this work.

- *Retaining Physical Meaning of Parameters* : Parameter reduction maps the

original set of parameters to a new and reduced set. It is important to relate the resultant new (or artificial) parameters to the original physical parameters. In the proposed approach, this is achieved by keeping the linear, or more generally, nonlinear mapping between the original and reduced parameter sets. This allows us to physically interpret circuit models built upon the reduced parameter set. Additionally, it enables the application of parameter reduction under the full-chip modeling context: parameter reduction and parameterized modeling is applied first to individual circuit blocks and then the resultant models are further combined to generate models at the next higher level. The ability of transforming back and forth between the original and reduced parameter sets makes it possible to handle common global parameters shared by multiple circuit blocks.

c. Statistical Circuit Modeling Using Performance-Based Dimension Reduction Methods

Following the ideas we presented previously, we outline how our proposed performance-oriented parameter reduction can be applied under the general context of statistical circuit analysis, as illustrated in Fig. 3.

- *Circuit Partitioning*: Partition the circuit into several building blocks using divide and conquer;
- *Smart Sampling*: Apply the smart sampling technique within each small partition to collect the statistical data for parameter reduction.
- *Dimension Reduction*: Based upon the correlation information from the samples to perform the performance based parameter reduction within each small

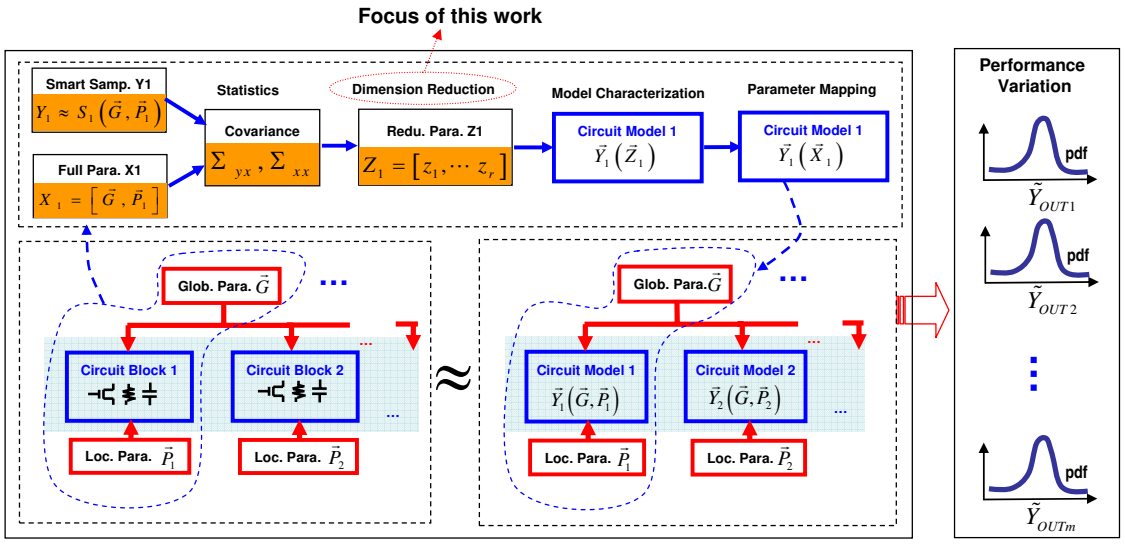

Fig. 3. Performance-based parameter reduction for full chip circuit modeling (considering multiple circuit blocks).

partition; obtain the mappings between the full parameter set (global/local variations) and the reduced parameter set.

- *Model Characterization*: Generate individual parametric circuit models in the reduced parameter space.

- *Parameter Mapping*: Use the mappings obtained in the dimension reduction step to map the reduced parameter set to the full parameter space; Consequently, generate the parametric performance models in the full parameter space.

- *Final Model Evaluation*: Combine all the parametric circuit models and evaluate the statistical variations due to all the global/local variations.

In the above procedures, it is evident that the efficiency of the model characterization step will be significantly improved, since much smaller parameter dimension is considered. Additionally, the mappings between the full and reduced parameter

sets produced by the parameter reduction will allow proper handling of global/local variation parameters impacting multiple circuit blocks. Such mappings also enable us to recursively apply parameter reductions for the full chip modeling. Due to the scope of this work, we will only focus on parameter reduction of partitioned circuit blocks and leave the hierarchical full-chip modeling to the future work.

B. Performance-Based Dimension Reduction via Reduced Rank Regression (RRR)

To achieve more powerful parameter dimension reduction, it is clear that a framework that can take into account the meaningful structural information of a given design and consider the modeling of multiple performances is desired. To facilitate a statistical parameter reduction approach rigorously, we adopt RRR as a suitable modeling tool and extend it for practical circuit modeling needs.

1. Dimension Reduction with Linear Reduced Rank Regression Framework

a. Linear Multivariate Reduced Rank Regression Theory

Regression analysis has been widely used in statistical data analysis. We consider the general multivariate linear model

$$Y = CX + \varepsilon, \tag{2.4}$$

where Y is an $m \times N$ matrix containing N-samples of m dependent variable vectors, X is an $n \times N$ matrix containing N-samples of n predictor variables, C is an $m \times n$ regression coefficient matrix and ε is the zero-mean random errors of the regression. As a standard approach, C can be found by using the least square criterion, which aims to minimize the trace (sum of the diagonal elements) of the covariance matrix,

$\Sigma_{\varepsilon\varepsilon}$ of ε

$$tr(\varepsilon\varepsilon^T) = tr[(Y - CX)(Y - CX)^T]. \tag{2.5}$$

A unique solution for C can be shown to be

$$C = YX^T(XX^T)^{-1}. \tag{2.6}$$

It is easy to show that the minimization of the trace of $\Sigma_{\varepsilon\varepsilon}$ also implies the minimization of the standard deviation error for each dependent variable in the response vector Y. Note that the above linear regression model does not lend itself to parameter reduction. The standard regression model does not exploit any statistical redundancy and correlation between Y in the model. In practical problems, however, it is very likely that significant model redundancy may exist, which manifests in the possibility of constructing a rank-reduced regression matrix $\tilde{C}$.

Suppose that we have a predictor variable vector $X \in \mathbb{R}^n$ and a dependent variable vector $Y \in \mathbb{R}^m$, with each having a zero mean. We denote the covariance matrix of X as $Cov(X) = \Sigma_{xx}$, and the covariance matrix between X and Y as $Cov(Y, X) = \Sigma_{yx} = \Sigma_{xy}^T$. The following theoretical result can be shown [28]:

Theorem 1 *An $m \times r$ matrix A_r and $r \times n$ matrix B_r can be found to minimize the trace*

$$tr\{E[(Y - A_rB_rX)(Y - A_rB_rX)^T]\}, \tag{2.7}$$

where

$$A_r = U, B_r = U^T\Sigma_{yx}\Sigma_{xx}^{-1}, \tag{2.8}$$

and $U = [U_1, ..., U_r]$ contains r normalized eigenvectors corresponding to the r largest eigenvalues ($\lambda_j, j = 1, ..., r$) of the matrix

$$D = \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{yx}^T. \tag{2.9}$$

In the above theory, suppose we are given a matrix $S \in \mathbb{R}^{m\times n}$, which is the first order sensitivity matrix that relates Y and X by $Y = SX$, then we have $\Sigma_{yx} = S\Sigma_{xx}$, which further gives

$$D = S\Sigma_{xx}S^T. \tag{2.10}$$

b. Implication on Parameter Dimension Reduction

It is critical to note that, a successful application of RRR also implies the possibility of parameter reduction. In other words, by the previously described rigorous procedure, the inherent redundancy in the predictor variables can be filtered out statistically. To see this point, we first note that we have computed a rank-r regression model that minimizes the statistical errors in Y in the sense of (2.55)

$$Y = A_r B_r X + \tilde{\varepsilon}, \tag{2.11}$$

where $\tilde{\varepsilon}$ represents the model error. We can construct a new set of variable $Z \in \mathbb{R}^r$ $(r < n)$ as

$$Z = B_r X, \tag{2.12}$$

leading to an optimal regression model

$$Y \approx A_r Z. \tag{2.13}$$

c. Formation of the Reduced Parameter Set

Until now, the original set of parameters in X can be reduced into new variables in Z using a mapping matrix B_r (2.12). From another angle, B_r reveals the importance of each old parameter (in X) with respect to the performance (in Y) in a statistical sense, which can be clearly understood by examining the weighing coefficients. For example, the (i, j) entry of matrix B_r describes the linear contribution of the jth

original parameter x_j to the ith new parameter z_i.

Obviously, this feature of the RRR-based parameter reduction method is more preferable when compared with the traditional parameter screening technique [2]. For instance, we know that the performance of VLSI system can be expressed using response surface models (RSM), and the coefficients of the RSM model actually provide the sensitivities of the underlying parameters. Though people can reasonably reduce the parameter dimension of a specific RSM model by examining the sensitivities of the parameters and removing those parameters with insignificant coefficients, for high-dimensional performance cases, the method becomes difficult to apply.

d. Inverse Mapping: From Z to X

To map the reduced parameter set (Z) back to the full parameter set (X), we can use the pseudo-inverse (Moore-Penrose) [29] of matrix B_r that satisfies

$$X = TZ. \tag{2.14}$$

This mapping is done by computing the singular value decomposition (SVD) of B_r matrix. So if the SVD of B_r matrix is $B_r = U\Sigma V^T$, then the pseudo-inverse is $T = V\Sigma^{-1}U^T$. As will be discussed later (Section a), this inverse mapping is necessary for converting models from the full parameter space to the reduced parameter space.

e. Uncorrelated Reduced Parameter Set

The new parameters in Z are uncorrelated and under certain conditions, they are independent, which lead to the following theorem:

Theorem 2 *For any reduced parameter set $Z = B_rX$, the individual variables $z_1, z_2, ...$ are uncorrelated (under the normal distributions, they are independent). This argu-*

ment is equivalent to that the covariance matrix of Z is diagonal.

Since the covariance matrix of Z is given by:

$$Cov(Z) = E\left(B_r X X^T B_r^T\right) = B_r \Sigma_{xx} B_r^T, \tag{2.15}$$

after substituting $B_r = U^T \Sigma_{yx} \Sigma_{xx}^{-1}$, we have:

$$Cov(Z) = U^T \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xx} \Sigma_{xx}^{-1} \Sigma_{yx}^T U = U^T D U = \Lambda, \tag{2.16}$$

where Λ is a diagonal matrix containing all the eigenvalues of D matrix in (2.9). So the covariance matrix of Z is diagonal thus the new parameters in Z are uncorrelated variables. It is interesting to see that the standard deviations of these new parameters are simply the square roots of the diagonal elements of the matrix Λ.

f. Linear Reduced Rank Regression Algorithm

Algorithm 1 Linear RRR Algorithm

Input: First order sensitivity matrix $\mathbf{S} \in \mathbb{R}^{\mathbf{m}\times\mathbf{n}}$ (satisfies $\mathbf{Y} \approx \mathbf{SX}$, where $\mathbf{Y}$ is the standardized performance vector with $\mu_{\mathbf{Y_i}} = \mathbf{0}, \sigma_{\mathbf{Y_i}} = \mathbf{1}$), parameter dimension $\mathbf{n}$ (of $\mathbf{X}$), the parameter covariance matrix $\mathbf{\Sigma_{xx}}$ and the reduced parameter dimension $\mathbf{r}$ or the pre-defined error tolerance ϵ.
Output: The mapping matrix $\mathbf{B_r}$.

1: Set $\mathbf{D} \leftarrow \mathbf{S\Sigma_{xx}S^T}$;
2: Do eigen-decomposition for $\mathbf{D}$ matrix such that $\mathbf{D} = U\Lambda U^T$ to get all the eigenvalues $\lambda_\mathbf{i}$ (in descending order) and the corresponding eigenvectors $\mathbf{u_i}$;
3: **if** Use the error tolerance ϵ to find the parameter dimension $\mathbf{r}$ **then**
4: Set $\lambda_\mathbf{s} \leftarrow \mathbf{\Sigma_{i=1}^m} \lambda_\mathbf{i}$;
5: Find $\mathbf{r}$ that satisfies $\mathbf{\Sigma_{i=1}^r} \lambda_\mathbf{i} > (\mathbf{1} - \epsilon)\lambda_\mathbf{s} > \mathbf{\Sigma_{i=1}^{r-1}} \lambda_\mathbf{i}$;
6: **else**
7: Use the default $\mathbf{r}$ as the parameter dimension;
8: **end if**
9: Use the $\mathbf{r}$ largest eigenvalues and their corresponding eigenvectors to form a diagonal matrix $\mathbf{\Lambda_r}$ and a matrix $\mathbf{U_r}$;
10: Set $\mathbf{B_r} \leftarrow \mathbf{\Lambda_r^{-1/2} U_r^T S}$;
11: Return the mapping matrix $\mathbf{B_r}$.

We conclude the parameter reduction algorithm using linear RRR in Algorithm 3. The inputs to the algorithm include the first order sensitivity matrix S that linearly relate the standardized performance space Y and the parameter space X and the covariance matrix Σ_{xx} of the full parameter space X. The input may also include the desired dimension of the reduced parameter set r, or the error tolerance ϵ. By following the formulas given in the previous sections, Step 1 of the algorithm computes the the D matrix in (2.9) using a closed form formula. The eigen-decomposition is subsequently performed to find all the eigenvalues and the eigenvectors of the D matrix. After sorting all the eigenvalues in a descending order, if we want to find the reduced parameter set for a predefined error tolerance ϵ, the sum of top r largest eigenvalues $\Sigma_{i=1}^{r}\lambda_i$ are compared with the sum of all eigenvalues λ_s in step 3 and 4, where the dimension of the final reduced parameter set is determined by looking at the the ratio $Rat_r = (\Sigma_{i=1}^{r}\lambda_i)/\lambda_s$: the smallest r that satisfies $Rat_r > 1-\epsilon$ will be chosen as the final dimension of the reduced parameter set. In Step 6 of the algorithm, $\Lambda_r^{-1/2}$ is used to scale the rows of B_r matrix such that the final reduced parameters ($Z = B_r X$) all have the N(0,1) distributions.

We show the comparisons on the key steps of the PCA and the RRR algorithms in Fig. 4. The standard PCA can be applied to reduce the data redundancy in either X or Y, but not the both simultaneously. Under our context of circuit modeling, it is important to note that a reduced rank model such as (2.61) is computed not to simplify a given more complex model [e.g., (2.52)]), instead, it is used as a means to reveal the redundancy in the predictor variables (e.g., process variations) to fulfill the purpose of parameter reduction. In our circuit modeling task, Y does not have to be the circuit performances of interest, more generally it can be chosen to be some other easily computed circuit responses that are closely related to the performances, as described in the following sections.

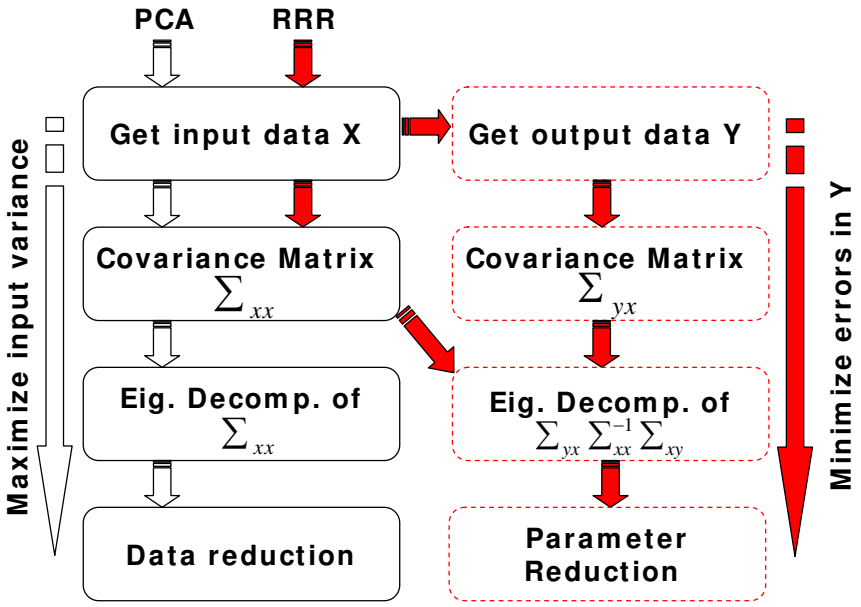

Fig. 4. Comparison of PCA and RRR for circuit modelings.

2. Dimension Reduction with Nonlinear Reduced Rank Regression Framework

For many realistic circuit problems, we have noted that the linear regression models in (2.52) and (2.59) are not completely adequate to capture the noticeable nonlinear relationship between process variables and circuit performances, especially when the range of the process variations is relatively large. To seek a more robust parameter reduction for these cases, we adopt the same notion of reduced rank regression as described in the previous subsection but cast it under a more general quadratic model.

Consider the following quadratic regression model

$$Y = f(X) \approx \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} X \\ X \otimes X \end{bmatrix}, \tag{2.17}$$

where the quadratic terms of X are expressed using the tensor product: $X \otimes X = [x_1^2, x_1x_2, \cdots, x_1x_n, \cdots, x_n^2]^T$, C_1 and C_2 are the first- and second-order coefficient matrices, respectively. To apply the reduced rank approximation, ideally we want to

find some regression matrices $\tilde{A}_{r_1} \in \mathbb{R}^{m\times r}$, $\tilde{A}_{r_2} \in \mathbb{R}^{m\times r^2}$, and $\tilde{B}_r \in \mathbb{R}^{r\times n}$ such that the error of the following reduced-rank regression model can be minimized in a statistical sense

$$Y \approx \begin{bmatrix} \tilde{A}_{r_1} & \tilde{A}_{r_2} \end{bmatrix} \begin{bmatrix} \tilde{B}_r X \\ \left(\tilde{B}_r X\right) \otimes \left(\tilde{B}_r X\right) \end{bmatrix}. \tag{2.18}$$

However, it turns out that an optimal model in the form of (2.18), to the best of our knowledge, cannot be derived explicitly. Thus we propose two feasible ways to obtain a proper reduced parameter set under this nonlinear model.

a. Extended Nonlinear RRR

In this approach, the nonlinear RRR problem is first converted to an equivalent linear RRR problem and then solved for the optimal solution. To include the nonlinear effects of the predictor variables on the response variables, we form an equivalent linear RRR model by including the quadratic terms $X \otimes X$ in the linear RRR model as additional predictor variables.

An augmented predictor variable vector is defined as

$$\tilde{X} = \begin{bmatrix} X \\ X \otimes X \end{bmatrix}. \tag{2.19}$$

We then compute the new covariance matrices $Cov(Y, \tilde{X}) = \Sigma_{Y,\tilde{X}}$ and $Cov(\tilde{X}) = \Sigma_{\tilde{X},\tilde{X}}$ and follow the linear RRR procedure to get a reduced-rank model

$$Y \approx A_r \begin{bmatrix} B_{r_1} & B_{r_2} \end{bmatrix} \begin{bmatrix} X \\ X \otimes X \end{bmatrix}, \tag{2.20}$$

where $A_r \in \mathbb{R}^{m\times r}$, $B_{r_1} \in \mathbb{R}^{r\times n}$ and $B_{r_2} \in \mathbb{R}^{r\times n^2}$. The above model is optimal in a sense similar to (2.55) (the regression model is cast in a quadratic form here). Compared

with the model in (2.18), here we have

$$\begin{aligned} A_r B_{r1} &\approx \tilde{A}_{r1}\tilde{B}_r, \\ A_r B_{r2} &\approx \tilde{A}_{r2}(\tilde{B}_r \otimes \tilde{B}_r). \end{aligned} \tag{2.21}$$

The reduced parameter set $Z \in \mathbb{R}^r$ is expressed in a quadratic form of X

$$Z = B_{r1}X + B_{r2}(X \otimes X). \tag{2.22}$$

It has been observed in our experiments that the above quadratic RRR model can significantly improve the accuracy of the pure linear RRR model, especially for nonlinear digital circuits, where key circuit performances may be very sensitive to underlying process variations. However, one potential drawback of this approach is that the reduced parameter set Z become some nonlinear combinations of X [2]. Sometimes, this will make it difficult to find simple closed-form expressions of the statistical distributions of Z even those of X are known.

b. Iteration-Based Nonlinear RRR

To remedy the drawback of the "extended nonlinear RRR" approach, we propose an alternative iterative procedure to find the approximated reduced set of parameters under a quadratic RRR model. We show the detailed procedure in Algorithm 6.

c. Algorithm Details

The initial step of this algorithm is to apply the traditional linear RRR to find the initial guess for $\tilde{A}_{r1}^{(0)}$ and $\tilde{B}_r^{(0)}$ matrices. Then the kth iteration of this algorithm can be

[2]When the circuit performances have a mild nonlinear dependency on the parametric variations, it has been observed that the nonlinear portion in (2.22) may be safely truncated.

Algorithm 2 Iteration-Based Nonlinear RRR Algorithm

Input: Standardized response vector $\mathbf{Y}$, predictor vector $\mathbf{X}$, the error tolerance $\xi_{\mathbf{0}}$, the dimension of the reduced parameters $\mathbf{r}$ and the maximum number of iterations $\mathbf{N_{max}}$.
Output: $\tilde{\mathbf{A}}_{\mathbf{r1}}$, $\tilde{\mathbf{A}}_{\mathbf{r2}}$ and $\tilde{\mathbf{B}}_{\mathbf{r}}$ matrices in (2.18).

1: Do linear RRR to find the initial $\tilde{\mathbf{A}}_{\mathbf{r1}}^{(\mathbf{0})}$ and $\tilde{\mathbf{B}}_{\mathbf{r}}^{(\mathbf{0})}$; Set $\mathbf{k} \leftarrow \mathbf{1}$;
2: **while** $\left(\xi^{(\mathbf{k})} \geq \xi_{\mathbf{0}}\right) \& \left(\mathbf{k} \leq \mathbf{N_{max}}\right)$ **do**
3: Set $\mathbf{Y}_{\mathbf{quad}}^{(\mathbf{k})} \leftarrow \mathbf{Y} - \tilde{\mathbf{A}}_{\mathbf{r1}}^{(\mathbf{k-1})}\tilde{\mathbf{B}}_{\mathbf{r}}^{(\mathbf{k-1})}\mathbf{X}$;
4: Do nonlinear regression [30] for $\mathbf{Y}_{\mathbf{quad}}^{(\mathbf{k})}$ with respect to the reduced parameter set $\tilde{\mathbf{Z}}^{(\mathbf{k-1})} = \tilde{\mathbf{B}}_{\mathbf{r}}^{(\mathbf{k-1})}\mathbf{X}$, which gives $\tilde{\mathbf{A}}_{\mathbf{r2}}^{(\mathbf{k})}$ that best satisfies: $\mathbf{Y}_{\mathbf{quad}}^{(\mathbf{k})} \approx \tilde{\mathbf{A}}_{\mathbf{r2}}^{(\mathbf{k})}\left(\tilde{\mathbf{Z}}^{(\mathbf{k-1})} \otimes \tilde{\mathbf{Z}}^{(\mathbf{k-1})}\right)$;
5: Set $\hat{\mathbf{Y}}_{\mathbf{quad}}^{(\mathbf{k})} \leftarrow \tilde{\mathbf{A}}_{\mathbf{r2}}^{(\mathbf{k})}\left(\tilde{\mathbf{Z}}^{(\mathbf{k-1})} \otimes \tilde{\mathbf{Z}}^{(\mathbf{k-1})}\right)$;
6: Set $\mathbf{Y}_{\mathbf{lin}}^{(\mathbf{k})} \leftarrow \mathbf{Y} - \hat{\mathbf{Y}}_{\mathbf{quad}}^{(\mathbf{k})}$;
7: Do linear RRR for $\mathbf{Y}_{\mathbf{lin}}^{(\mathbf{k})}$ and $\mathbf{X}^{(\mathbf{k})}$ to get the updated $\tilde{\mathbf{A}}_{\mathbf{r1}}^{(\mathbf{k})}$ and $\tilde{\mathbf{B}}_{\mathbf{r}}^{(\mathbf{k})}$ matrices;
8: Set $\xi^{(\mathbf{k+1})} \leftarrow \left\|\tilde{\mathbf{B}}_{\mathbf{r}}^{(\mathbf{k})} - \tilde{\mathbf{B}}_{\mathbf{r}}^{(\mathbf{k-1})}\right\|$; Set $\mathbf{k} \leftarrow \mathbf{k} + \mathbf{1}$;
9: **end while**
10: Return $\tilde{\mathbf{A}}_{\mathbf{r1}}$, $\tilde{\mathbf{A}}_{\mathbf{r2}}$, $\tilde{\mathbf{B}}_{\mathbf{r}}$ and the number of iterations $\mathbf{k}$.

depicted as follows: a nonlinear regression algorithm [30] is used to find the coefficient matrix $\tilde{A}_{r2}^{(k)}$ for the quadratic portion of $Y_{quad}^{(k)}$ with respect to the quadratic portion of the reduced parameter set $\tilde{Z}^{(k-1)} = \tilde{B}_r^{(k-1)}X$; the updated the linear portion $Y_{lin}^{(k)}$ of Y is given by $Y_{lin}^{(k)} = Y - \tilde{A}_{r2}^{(k)}\left(\tilde{Z}^{(k-1)} \otimes \tilde{Z}^{(k-1)}\right)$; another traditional linear RRR is conducted to obtain the updated $\tilde{A}_{r1}^{(k)}$ and $\tilde{B}_r^{(k)}$ based on the linear portion $Y_{lin}^{(k)}$ and X. Each iteration of the algorithm only requires one time nonlinear regression for the quadratic portions of Y and Z, which will not require a significant optimization cost. In our experiments, it has been observed that an optimal Br matrix can be obtained after only two or three iterations. Thus this approach is rather efficient for coping with the strongly nonlinear effects while maintaining a simple linear mapping between X and Z.

d. Algorithm Complexity

The main computational cost of each iteration loop includes one time nonlinear regression (Step 4) and one time linear RRR (Step 7). The cost of the nonlinear regression depends on the dimension of the reduced parameter set Z, which is typically very small. Consequently, the complexity of the iterative RRR mainly depends on dimension of the reduced parameter set.

We should notice that the main cost of this parameter reduction method is due to the sample data (Y) collection which involves numerous circuit simulations. Once the samples in the parameter space and performance space are collected, the computational cost due to the iteration-based RRR algorithm can be negligible.

C. Variability Modeling of Interconnect Circuits Using Parameter Reduction

1. Statistical Circuit Model Generation with Parameter Reduction

In this section, we first show how our parameter dimension reduction framework is applied to statistical modeling of interconnects using the linear RRR (Section b) or the extended nonlinear RRR (Section a). Next, we show the application of the iteration-based nonlinear RRR (Section b) for the transistor circuits. Finally, some verification methods based on sampling base approaches will be introduced.

a. Parameter Reduction for Interconnects

1) Capturing Interconnect Parametric Variations:

We use the standard modified nodal analysis (MNA) equations to describe an interconnect network

$$\begin{cases} (G + sC)\, x = Bu \\ y = L^T x \end{cases}, \tag{2.23}$$

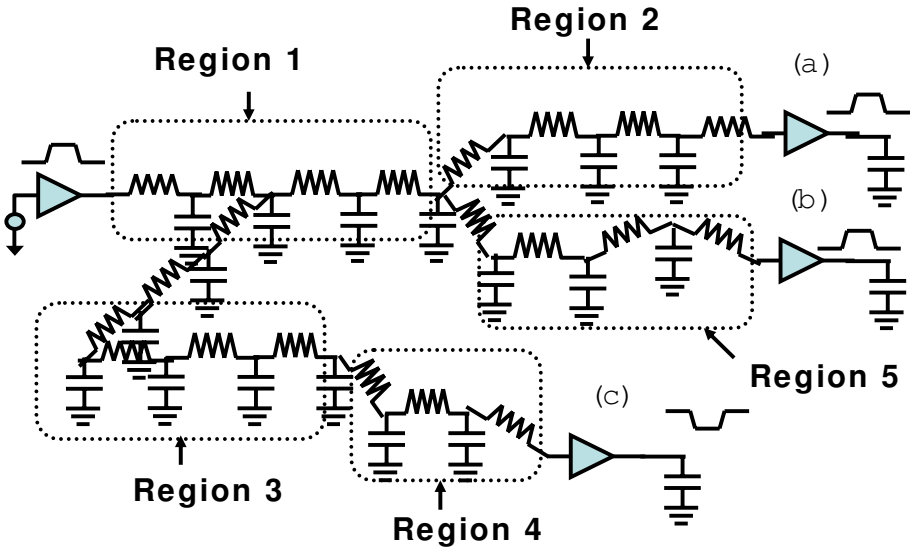

Fig. 5. An RC circuit with parametric variations.

where $u \in R^{n\times 1}$ and $y \in R^{m\times 1}$ represent the inputs and outputs, $x \in R^{N\times 1}$ represents the system unknowns, $G, C \in R^{N\times N}$ are the conductance and capacitance matrices, $B \in R^{N\times n}$ and $L \in R^{N\times m}$ are the input and output matrices, respectively.

In order to possibly capture process variations, without loss of generality, we consider the RC circuit shown in Fig. 5 as an example. The circuit has one nonlinear driver providing the input and three output circuit nodes driving three downstream stages. The circuit is divided into several regions spatially and the local geometrical variations are introduced on a per region basis to capture possible spatial process variations. Variations considered in this dissertation only include various geometrical parameters such as wire width and thickness, dielectric layer thickness, though other types of local or global parameters can be treated in a similar way. Generally, we consider a set of n_p local and global geometrical variation variables: $\vec{p} = [p_1, p_2, \cdots, p_{n_p}]^T$. To capture the influences of $\vec{p}$ on the system equations (2.23), the following expansions

of conductance and capacitance matrices in $\vec{p}$ are used:

$$G = G_0 + \sum_i G_i p_i, \tag{2.24}$$

$$C = C_0 + \sum_i C_i p_i. \tag{2.25}$$

In practice, we may only consider variations in resistances and capacitances and neglect inductance variations, which have been observed to be small.

RRR-Based Interconnect Parameter Reduction

A full account of global and local variations in a large multi-layer interconnect network can lead to the consideration of a large set of geometrical variables (n_p is large). However, if we are only interested in analyzing the circuit performances at the output nodes, the effective parameter dimension of a given network may not be very large since the specific circuit structure can hide certain parametric variations and may even introduce canceling effects between multiple variations. To identify the reduced parameters of interconnect, general linear RRR is applied, since for linear network the nonlinearity is not significant. It is important to understand that linear RRR is only applied to find the reduced parameter set (linear combinations of the original parameters), though the output response of the interconnect circuit may also depend on the nonlinear effects of underlying parameters. The application of the extended nonlinear RRR (a) for linear network has also been described in [31], which is a more general parameter reduction approach but at a higher cost.

In this part, we only consider the linear RRR method proposed in Section b. To apply linear RRR, one would very naturally choose the underlying process variations

($\vec{p}$) as the predictor variables (X). In the proposed approach, linear RRR is only employed as a tool to perform parameter reduction but is not used for performance modeling. Therefore, the dependent variables (Y) may not have to be chosen as certain performance measures such as circuit delays. In practice, this flexibility is particularly useful because in many cases a compact simulation model is often needed but not a performance model.

For interconnect models, we use the standardized transfer function moments (with $\mu = 0, \sigma = 1$) as the dependent variables (Y) based on their strong correlation with timing performance. One important benefit of such choice is that transfer function moments are also easy to compute. We have developed computationally efficient procedures to generate closed-form expressions for transfer function moments and their dependency on the underlying geometrical variations. As such, statistical measures required by RRR (e.g. S and Σ_{xx} in Algorithm 3), can be efficiently obtained in closed-form without resorting Monte-Carlo sampling, leading to high efficiency of the proposed parameter reduction.

We expand a transfer function moment (m_k) at a particular output of interest as

$$m_k = m_{nom,k} + \sum_{i=1}^{n_p} \alpha_{k,i} p_i, \tag{2.26}$$

where $k = 1, \cdots, n_s$ and n_s is the number of moments to be observed. For example, if we want to capture the first three moments for five output nodes, then n_s will be equal to 15. In the above equation, $m_{nom,k}$ is the nominal case moment, and $\alpha_{k,i}$ are the first order coefficients capturing the dependency of m_k on $\vec{p}$. As mentioned before, though the actual transfer function moments not only depend on the first-order terms but also the higher order terms, these first order forms are sufficient for linear RRR to accurately identify the reduced parameters.

The first-order coefficients $\alpha_{k,i}$ can be efficiently computed by reusing the LU factorization of the conductance matrix G [32]. Higher order dependencies of the moments on p_i can also be computed. With all the first-order sensitivities of the transfer function moments obtained above, the covariance matrix between Y and X is given as

$$\Sigma_{YX} = E\left\{SXX^T\right\} = S\Sigma_{XX}, \tag{2.27}$$

where $[S]_{i,j} = \alpha_{i,j}$. By following Algorithm 3, we can obtain the reduced set of parameters $Z = B_r X$ subsequently.

Reduced-Parameter Interconnect Models

To be benefited by parameter reduction in simulation, we need to cast our circuit model such as (2.23) in the reduced parameter set Z. Hence, the dependency of the system matrices on the new parameters should be computed:

$$G = G_0 + \sum_i G_{z_i} z_i; \tag{2.28}$$

$$C = C_0 + \sum_i C_{z_i} z_i. \tag{2.29}$$

Applying the chain rule, we have the first order sensitivities with respect to the new parameters as

$$G_{z_k} = \frac{\partial G}{\partial z_k} = \sum_i \frac{\partial G}{\partial p_i}\frac{\partial p_i}{\partial z_k};\ C_{z_k} = \frac{\partial C}{\partial z_k} = \sum_i \frac{\partial C}{\partial p_i}\frac{\partial p_i}{\partial z_k}. \tag{2.30}$$

To fully compute the above expressions, we still have to find $\frac{\partial p_i}{\partial z_k}$ first, which can be done by expressing X (p_i) in terms of reduced parameter set Z. To this end, T matrix in (2.62), which is the pseudo inverse of B_r, is computed such that $X = TZ$. Then

p_i in X can be written as

$$p_i = \sum_{j=1}^{n_z} t_{ij} z_j, i = 1, ..., n_p. \tag{2.31}$$

So we have $\frac{\partial p_i}{\partial z_k} = t_{ik}$ and the first-order sensitivities with respect to the new parameters z_k as

$$G_{z_k} = \frac{\partial G}{\partial z_k} = \sum_i G_i t_{ik};\ C_{z_k} = \frac{\partial C}{\partial z_k} = \sum_i C_i t_{ik}. \tag{2.32}$$

Upon obtaining the new simulation models in the reduced parameter set z, the immediate benefit of parameter reduction is to conduct Monte Carlo simulation by sampling in the new parameter space, which is much more efficient. We have applied variance reduction techniques such as Latin hypercube sampling (LHS) [33] to reduce the number of random samples needed to estimate performance statistics by working in the reduced parameter space. Due to the application of our RRR based parameter reduction, LHS becomes an effective variance-reduction tool in the low-dimensional parameter space.

Equally important, the reduction of parameter dimension is also a key to enable parameterized model order reduction techniques to compute compact simulation models while considering the impact of process variations [8, 9, 10]. It should be understood that the efficiency and the cost of these algorithms critically depend on the parameter dimension, as discussed before. By performing parameter dimension reduction, we are able to compute highly efficient reduced order models while capturing a large set of (original) process variables. This leads to compact reduced-parameter-order models.

b. Parameter Reduction for Transistor Circuits

Unlike the parameter reduction for the linear networks where the first-order sensitivities of the transfer function moments are extracted to form a linear RRR formulation, parameter reduction of transistor circuits is more complicated. It requires numerous circuit simulations in the parameter space X. The simulation cost incurred in the data collection can be justified under a hierarchical modeling approach, where small portions of the circuit are reduced first and then merged to form a larger portion of the circuit in a bottom-up fashion (as shown in Fig. 3). For this type of circuits, the response vector Y usually strongly depends on the nonlinear portion of the underlying parameters, thus the simple extended nonlinear RRR scheme (Section a) may cause inaccuracy in presence of large variation sources. To improve the modeling accuracy, the iteration-based nonlinear RRR (Section b) is applied, though it may cause a bit higher computation cost.

The typical parameter reduction procedure for the transistor circuits includes simulation data collection, data analysis and model verification. For the step of data collection, we have to determine the circuit performance to be observed. For instance, if we are interested in the circuit delays on some output nodes of a combinational logic circuit, we can first generate hundreds or thousands of samples of input parameters (corresponds to the predictor X) and then do simulations to collect the final delay values as the performance (corresponds to the response Y). Once all the input and response data are collected, we standardize all the response samples in Y and conduct the iteration-based nonlinear RRR to find the transform matrix B_r using Algorithm 6, which may involve only several iterations.

2. Verification of Reduced-Parameter Models

Once the parameter reduction has been carried out using the technique described in previous sections, parameterized circuit models can be generated based upon the reduced parameter set using a technique such as response surface modeling (RSM) or parameterized model order reduction. It is expected that the cost of the parameterized circuit modeling will be significantly reduced if the dimension of the new parameter space is noticeably compressed. Due to the scope limitation of the present work, here we are only concerned with parameter dimension reduction with the understanding that the subsequent parameterized model generation can be conducted by using many existing techniques.

To verify the accuracy of the proposed parameter reduction technique, we show the procedure for re-sampling with the reduced-parameter set. Suppose that we have obtained B_r in (2.12), or B_{r1} and B_{r2} in (2.22), then we can verify the accuracy of such parameter reduction by applying Monte-Carlo simulation of the original circuit model and sampling in the new parameter set Z. Though there are several ways to conduct such sampling scheme, we prefer to use the following two simple methods for the purpose of demonstration.

a. Direct Re-sampling

To examine the model accuracy on the PDF or CDF distributions after parameter reduction, we can do re-sampling in the new parameter space Z. For interconnect and typical RC circuits, by computing Z in (2.12), we can obtain quite satisfactory results (as shown in the result section). For nonlinear circuits, we either keep the complete nonlinear mapping in (2.22) or use the iterative nonlinear RRR to find a linear mapping, where the latter choice is preferred. Either for the interconnect circuit

modeling using linear RRR (Section b) or for the transistor circuit modeling using the iteration-based nonlinear RRR (Section a), the new parameters in $Z = [z_1, z_2, ...]^T$ are uncorrelated variables (Theorem 2). Since Algorithm 3 gives the reduced parameters with N(0,1) distributions, new samples in the reduced parameter space can be easily generated. Subsequently, we can accurately predict the probability distributions of the circuit response. This sampling scheme is proved to realize the variance reduction by various experiments, therefore much fewer samplings are needed for generating the accurate PDF and CDF compared with the samplings in the full parameter model. Our experiments indicate at least $5X$ reductions on the number of samples for an accurate estimate of the probability distributions.

b. Indirect Re-sampling

To verify the model accuracy for specific input parameters with the full parameter model, we can use an indirect approach for both the linear RRR-based model and iteration-based nonlinear model, which is described as follows:

1. Generate a set of random samples of X and use the linear/quadratic mapping of (2.12) or (2.22) to obtain the new parameter set Z;

2. Find the inverse mapping matrix T (2.62) that transforms the samples of Z to the new samples of X (only the linear portion of X in Z is kept);

3. Re-simulate the circuit using these new samples of X.

The third step of the indirect re-sampling method can be explained more comprehensively as follows. The linear portion of (2.22) represents a set of under-determined linear equations, in which the dimension of Z is much less than the dimension of X due to the parameter reduction. Fortunately, we know that a successful parameter

reduction implies that statistically not all the original parameters in X are important but only a few combinations of them (Z) are. This intuition allows us to use the standard methods for under-determined systems such as pseudo-inverse to express X in term of Z. For the iteration-based nonlinear RRR case, the indirect re-sampling can be significantly simplified since we can directly use the quadratic reduced rank regression model in (2.18) to predict the response, which is accurate enough for typical variation ranges.

It shall be noted that the direct re-sampling approach described above can be applied to the reduced-parameter models to speedup statistical circuit analysis. Under a full system analysis context, reduced-parameter models for various building blocks are extracted first, and then the statistical performance distributions of the complete system are computed by sampling in the compressed reduced parameter set resulted from parameter dimension reduction.

3. Numerical Results

We demonstrate the applications of the proposed techniques on several interconnect circuit examples and one digital circuit. This section includes the following four subsections.

A) *Parameter reduction via linear reduced rank regression.* The linear RRR-based algorithm is applied for parameter reduction of various interconnect circuits considering spatial correlation of the intra-die variations.

B) *Parameter reduction via nonlinear reduced rank regression.* The iterative RRR algorithm is applied for the parameter reduction of combinational logic circuit for independent Gaussian variation sources; The dimension reduction results are compared with the results given by PCA-based method.

C) *Combining parameter dimension reduction with model order reduction.* The multiparameter model order reduction is incorporated with the linear RRR-based parameter dimension reduction for interconnect circuits.

D) *Formation of the reduced parameter space.* The compositions of the reduced parameters for the interconnect/digital circuits are demonstrated; The reduced parameter sets produced by linear RRR and iteration-based nonlinear RRR algorithms are compared.

For the interconnect cases, we assume all the random interconnect geometrical variations are Gaussian. The accuracy of our reduced-parameter models as well as the reduced-parameter-order models are verified by examining 50% V_{dd} delays and frequency domain responses. For the combinational logic circuit, we assume all the threshold voltages of the transistors to be Gaussian and the accuracy of our reduced-parameter model is verified by examining the 50% V_{dd} delays at the specific output nodes. We also show the accuracy of two reduced-parameter models on a per sample basis.

a. Parameter Reduction via Linear RRR

1) Two Coupled Lines

First, we consider two coupled long RC lines as shown in Fig. 6. The wire width W and thickness T of each line are both 1 μm, and the dielectric layer thickness H is 0.5 μm. The spacing S between two lines is 0.8μm. To realistically relate the RC parameters with the geometrical parameters, which are subject to process variation, capacitance values are calculated using the closed-form formulas based on the geometrical values [34], while the unit length resistance is calculated using the cross section area and the conductor resistivity. To better investigate the efficiency and

accuracy of the parameter reduction algorithm, in this circuit example, we consider the following assumptions :

Assumption 1—Correlated Variation Sources: The two lines are divided into ten regions and three geometrical variations including W, T and H are considered. We include a dependent variable, wire spacing (S) which is subject to the wire width W. Therefore there are a total of 30 variation variables. We also consider the intra-die correlations in this case: we assume the spatial correlation of parameters p_i in region i and p_j in region j is determined according to their locations by $corr(p_i, p_j) = e^{-|i-j|/l_p}$, where l_p is the correlation length of parameter p. So a larger correlation length l_p indicates a stronger parameter correlation. We consider two sets of correlation coefficients for the intra-die variations: $corr.(1) = [l_w,\ l_t,\ l_h]$ represents a moderate parameter correlation between different regions while $corr.(2) = 10 \times [l_w,\ l_t,\ l_h]$ represents a much stronger spatial correlation. The purpose of this experiment is to examine how the number of the reduced parameters and the model accuracy varies under different spatial correlation setups. It can be expected that a stronger spatial correlation would result in a better accuracy when keeping the same number of the reduced parameters.

Assumption 2—Gaussian Parameters: Considering the fact that the second-order sensitivities of capacitance and resistance with respect to the geometry are quite small compared with the first-order terms, we can safely express the R and C values in the first-order forms of these geometrical parameters. The 3 σ geometrical variation varies from 15% to 30%. We apply the linear RRR-based parameter reduction algorithm in Section b to generated three parameter-reduced models with one, two and three parameters, respectively. Therefore, the maximum parameter reduction achieved is 20X for this example.

Accuracy 1—Delay Distributions: For both spatial correlations, we compare the

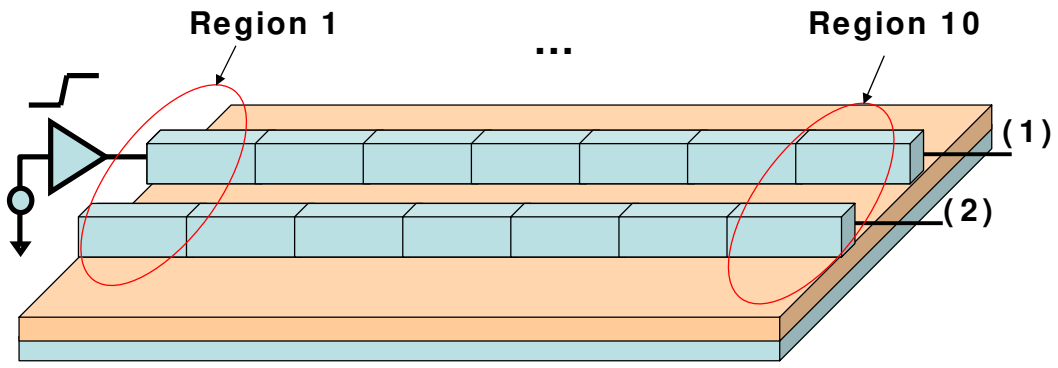

Fig. 6. Two coupled lines.

original model with the 3-parameter models by examining the delays at terminal (1) when a random ramp input is applied, as shown in Fig. 6. For the 3-parameter model, we demonstrate the reduction on the number of random samples required to compute the delay distribution when the 3 σ variations of all parameters are set to be 30%. To improve the sampling efficiency, LHS [33] is used to generate samples for all experiments.

First, we perform Monte Carlo simulations by using the LHS in the full 30-parameter model to get the delay distribution. It is observed that a minimum of $5,000$ samples are required in order to get a stable delay distribution. If sampling in the 2-parameter model using LHS, it is observed that 500 samples are enough to provide an accurate estimation of the same distribution. We also find that the accuracy is better when there is a stronger spatial correlation. We compare the PDF results under different spatial correlations in Fig. 7. As observed, the accuracies shown in the figure are pretty good.

Accuracy 2—Comparisons on Different Parameter Reduced Models: More experiments are conducted on the three reduced-parameter models with one, two and three

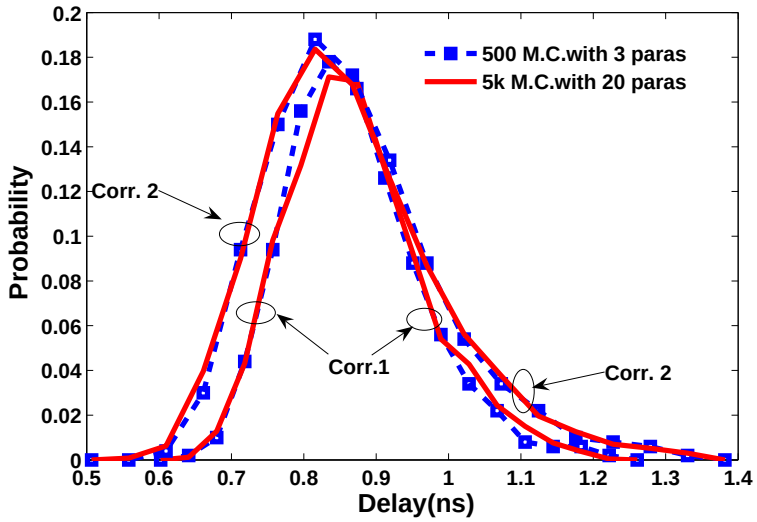

Fig. 7. Comparison of the full and the reduced-parameter models on the delay PDF using different intra-die correlation coefficients.

parameters, respectively in Table I. We use $4,000$ Monte Carlo samples in the full 30-parameter model to get stable estimation of the delay distribution, and compute the mean as well as the standard deviation (std.) as reference values. Then, we verify the accuracy (relative error in mean/std.) of these reduced-parameter models by generating 500 LHS samples for the parameter reduced models. To investigate the model accuracy for different inputs, we set random inputs with Gaussian distributed ramp values. As seen, the parameter reduced models can provide quite accurate estimations on the mean and standard deviation values while the three-parameter model is offering an excellent accuracy. From Table I we can also see that with the increase of the spatial correlations, the number of reduced parameters can be potentially reduced, which means that the true dimensionality of the original parameters depends on both the design structure and the parameter spatial correlations. As a comparison, the parameter dimension found by PCA is only based upon the spatial correlations.

Table I. Comparison of delays (random ramp inputs)

Variations of Parameters			4K LHS (corr. 1)		500 LHS Rel. Err. in % (corr. 1)		
σ_W	σ_H	σ_T	Mean	Std.	1 para.	2 paras.	3 paras.
5%	10%	10%	888.2 ps	77.3 ps	0.36/3.55	0.17/1.6	0.08/3.0.9
10%	5%	10%	899.3 ps	98.1 ps	1.20/6.11	0.87/4.72	0.70/3.04
10%	10%	5%	889.2 ps	70.5 ps	0.60/5.07	0.41/4.01	0.27/1.11
Variations of Parameters			4K LHS (corr. 2)		500 LHS Rel. Err. in % (corr. 2)		
σ_W	σ_H	σ_T	Mean	Std.	1 para.	2 paras.	3 paras.
5%	10%	10%	889.4 ps	101.5 ps	0.04/3.09	0.03/4.64	0.04/0.37
10%	5%	10%	892.2 ps	127.3 ps	0.29/5.73	0.50/2.35	0.47/0.39
10%	10%	5%	888.9 ps	93.9 ps	0.08/3.01	0.00/1.77	0.01/1.56

2) Interconnect with Multiple Outputs

In this example, we consider an interconnect extracted from a realistic circuit ISCAS85 c499 [35] with 12 sink nodes. The circuit is divided into five regions according to the physical layout. Wire width and thickness variations ($3\sigma = 30\%$) within each region are assumed to be Gaussian. So there are totally 10 geometrical parameters in this circuit.

Three reduced parameters are obtained by linear RRR (Section a) to capture the performance variations of the 12 sink nodes. Fig. 8 shows the comparisons of the transient waveforms of 10 random simulations using the full parameter model and the parameter reduced model. As shown in the figure, there is no significant difference between the results of the above two models (for all 12 sinks).

For this 12-output circuit, only 3X parameter reduction is achieved, while 10X reduction is realized for the two-output circuit in the previous example. This can be

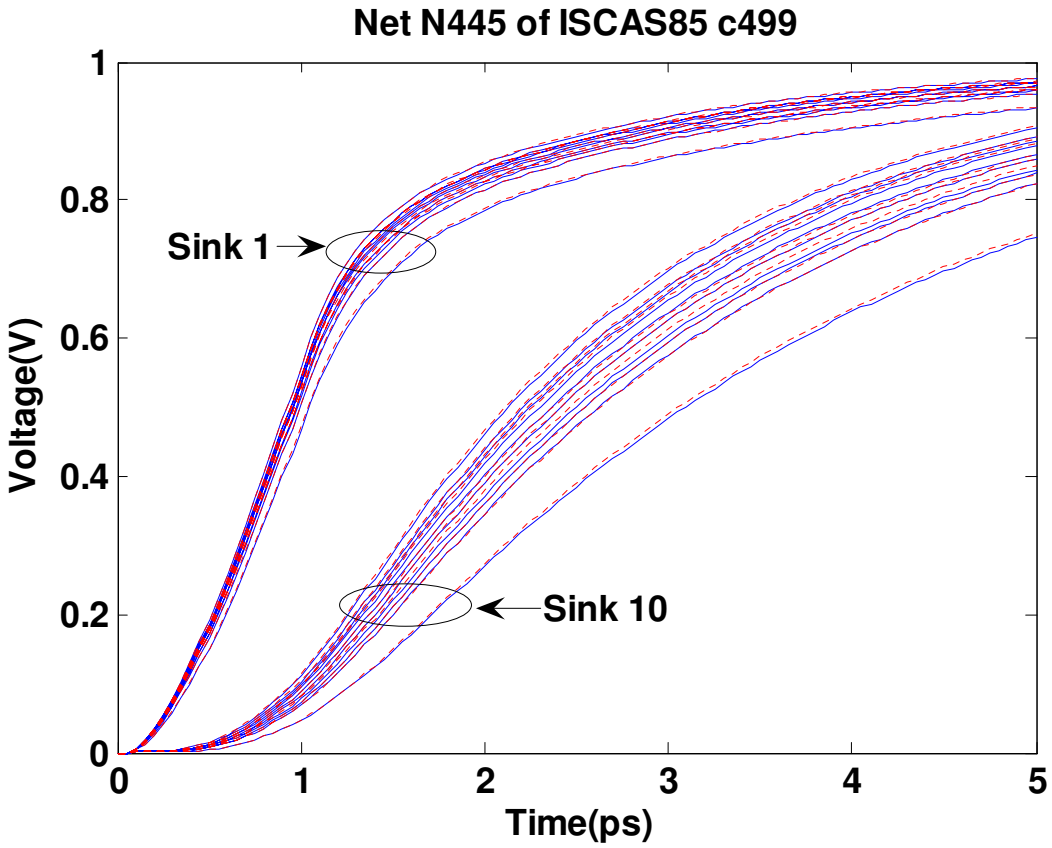

Fig. 8. Waveforms impacted by geometrical variations: Net445 with 12 sinks (10-parameter model vs. 3-parameter model).

easily understood by looking into Equation (2.27): as the number of outputs increases, the rank of the sensitivity matrix S (2.27) of transfer function moments increases, which lead to the increase of the resultant reduced parameter dimension.

b. Parameter Reduction via Nonlinear RRR

In this section we demonstrate the dimension reduction via the nonlinear reduced rank regression for the timing simulation of combinational logic circuits. The ISCAS85 circuit c17 is investigated where the threshold voltages V_{th} of all transistors are considered as independent Gaussian random variables with 3σ variation of 15%.

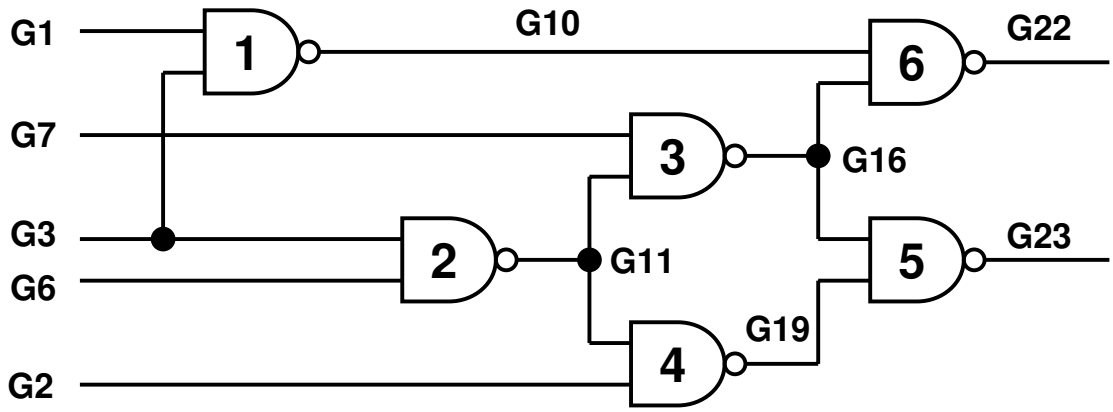

Fig. 9. Topology of ISCAS85 benchmark circuit C17.

The schematic of the circuit is shown in Fig. 9. All of the NAND gates are the same and each of them consists of four transistors. We number the NAND-gates from 1 to 6, while the 24 transistor threshold parameters in the complete circuit are numbered as follows: In gate 1, the parameters $1-4$ correspond to the transistors $M1-M4$; In gate 2, the parameters $5-8$ correspond to the transistors $M5-M8$, ..., etc. The above setup brings totally 24 transistor threshold parameters in the complete circuit.

We apply two rising inputs to $G2$ and $G3$ while randomly varying the threshold voltages of all transistors. The delays and slews at the outputs of gate 5 and gate 6 are set as the observations while the 24 threshold voltage variations are set as the input random variables. We conduct $1,000$ Monte Carlo transient simulations using HSPICE and the results are collected for the linear RRR dimension reduction algorithm (Section b) as well as the iteration-based RRR algorithms (Section b). We found the latter algorithm only need three iterations to find the optimal B_r matrix. The application of our parameter dimension reduction technique leads to a significantly reduced new parameter set with only three parameters. On the other hand, PCA can not do any reduction since all the variation sources are uncorrelated

with the same standard deviations.

We use the indirect re-sampling in section 2 to verify the accuracies of the two parameter reduced models. The relative errors of the two models are plotted in Fig.10 and Fig.11. We can observe that the iteration-based nonlinear RRR is able to find a more optimal B_r matrix than the simple linear RRR approach.

c. Combining Parameter Dimension Reduction with Model Order Reduction

In the previous examples, we have demonstrated the accuracy of the reduced-parameter interconnect models as well as the improved efficiency brought by these models in sampling-based circuit analysis. To tackle the statistical analysis complexity brought by the high parameter dimension and the large design size simultaneously, we combine parameter reduction and model order reduction techniques to compute compact reduced-parameter-order models. It should be noted that the cost of most parameterized interconnect model order reduction algorithms grow exponentially in the number of the parameters, thus a significant reduction in the parameter space will lead to highly efficient parameterized models as shown by the following circuit examples. It is obvious that the reduction of the parameter set will also help control the cost of the parameterized modeling for transistor circuits although this issue is not studied due to the limitation of the scope of this work.

1) Two Coupled RC Lines

For the two coupled RC line circuit (204 circuit unknowns) in Fig. 6, we split the lines into five regions and setup the experiment following the similar procedure depicted in Section a. We first apply the linear RRR algorithm to reduce the parameter dimension from 15 to one and then use the parameterized model order reduction algorithm in [9] to compute a passive one-parameter 12th-order reduced model. Six transfer function moments of nodes (1) and (2) are selected as the dependent variables in

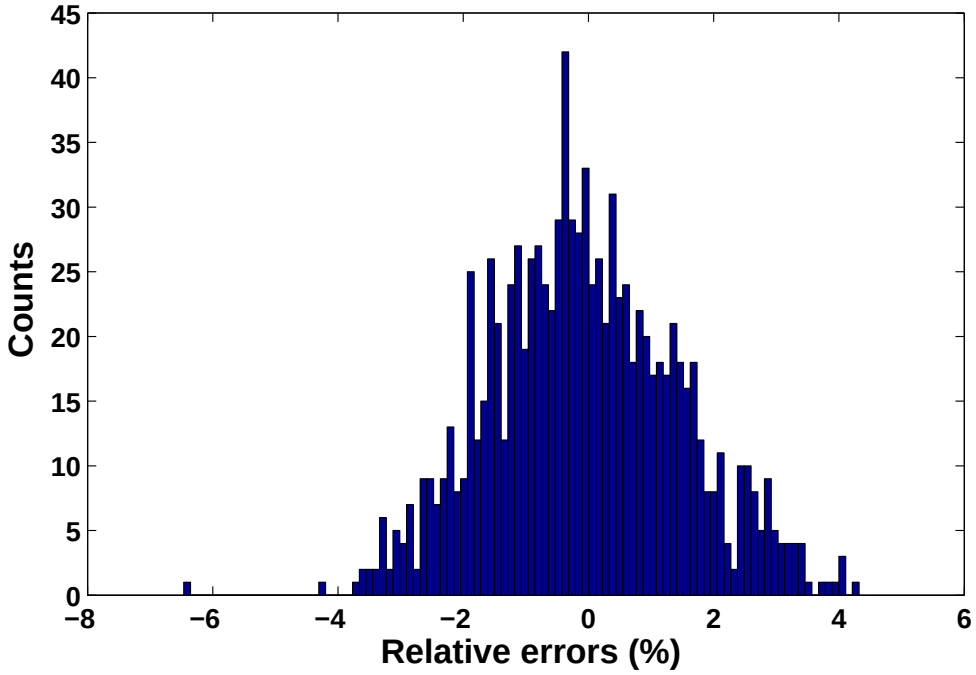

Fig. 10. Relative circuit delay errors of 1,000 simulations using the linear RRR parameter reduction algorithm.

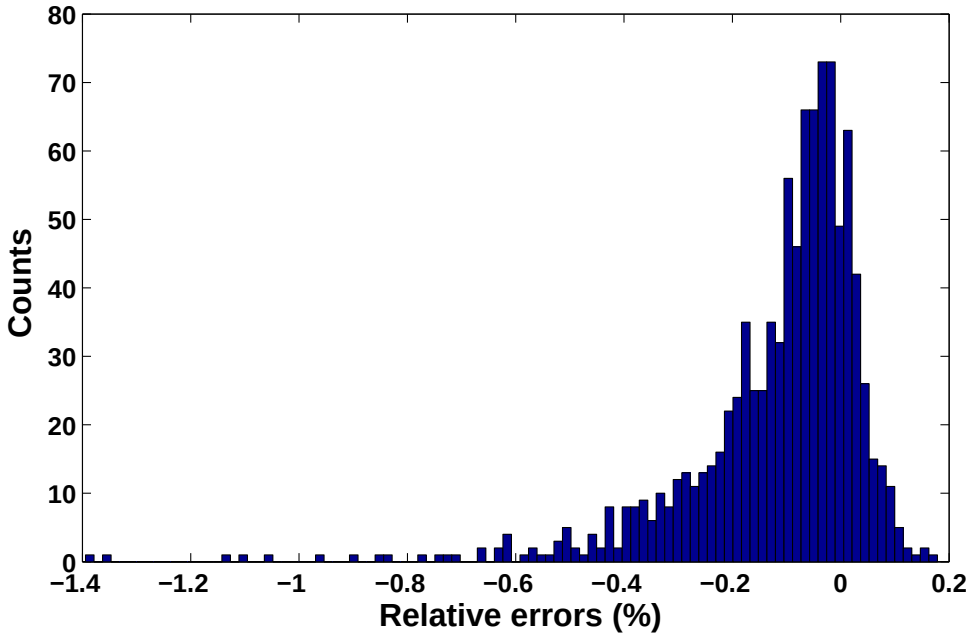

Fig. 11. Relative circuit delay errors of 1,000 simulations using the iteration-based RRR parameter reduction algorithm.

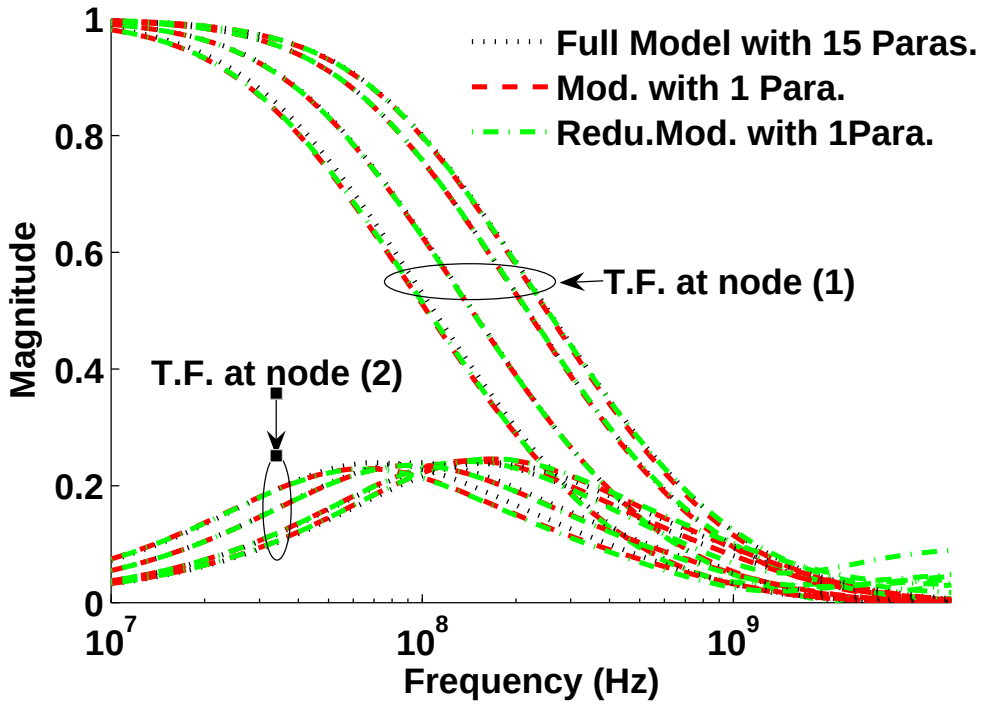

Fig. 12. Transfer function comparison between the full, reduced-parameter, and reduced-parameter-order models.

the RRR procedure. Since the model order reduction algorithm performs moment-matching with respect to the process variable, a direct inclusion of 15 parameters will lead to an explosion in model size. This difficulty is completely avoided by performing a reduction in the parameter space first.

We compare the frequency responses of the full model and the one-parameter 12th-order model on circuit samples generated by perturbing all the 15 geometrical parameters by $\pm 10\%$ and $\pm 20\%$, respectively. In Fig. 12, four samples of the frequency responses at nodes (1) and (2) are obtained based on three models: 15-parameter full-order model, one-parameter full-order model and one-parameter 12th-order reduced model, are plotted. We also plot the transfer functions of three circuit nodes located in different regions (as shown in Fig. 13). Not surprisingly, the accu-

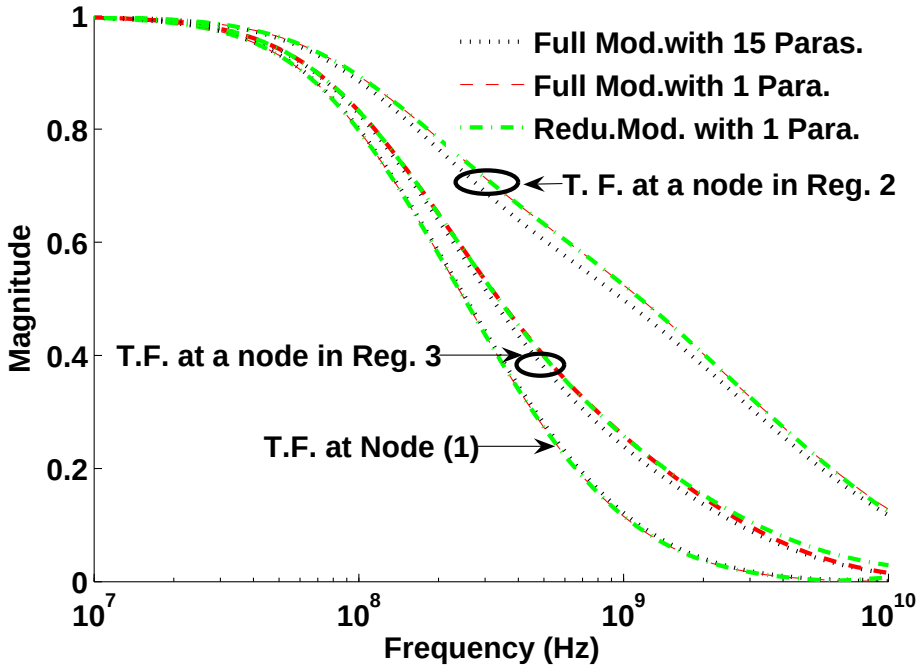

Fig. 13. Accuracy of the reduced models in different regions.

racy of the reduced models becomes worse at the node (region 2) that is far away from the observation nodes (nodes (1) and (2) in region 5) used in the RRR procedure.

2) An RLC Line

We apply the same reduction procedure to an RLC line. The line is 4 mm long and it contains 120 resistors, inductors and capacitors. We divide the line into ten regions and each region has three geometrical variations with the nominal values as: wire width $W = 1.2\ \mu m$, wire thickness $T = 1\mu m$, and dielectric layer thickness $H = 1\ \mu m$. Again we apply the linear RRR algorithm to reduce the number of the variation parameters from 30 to 1, resulting a 30x reduction. Then a one-parameter reduced order model is computed, which has a size of 16. We introduce $\pm 25\%$ variations on all 30 geometrical parameters to generate a set of circuit samples.

In Fig. 14, two circuit samples are selected for the full model, the one-parameter

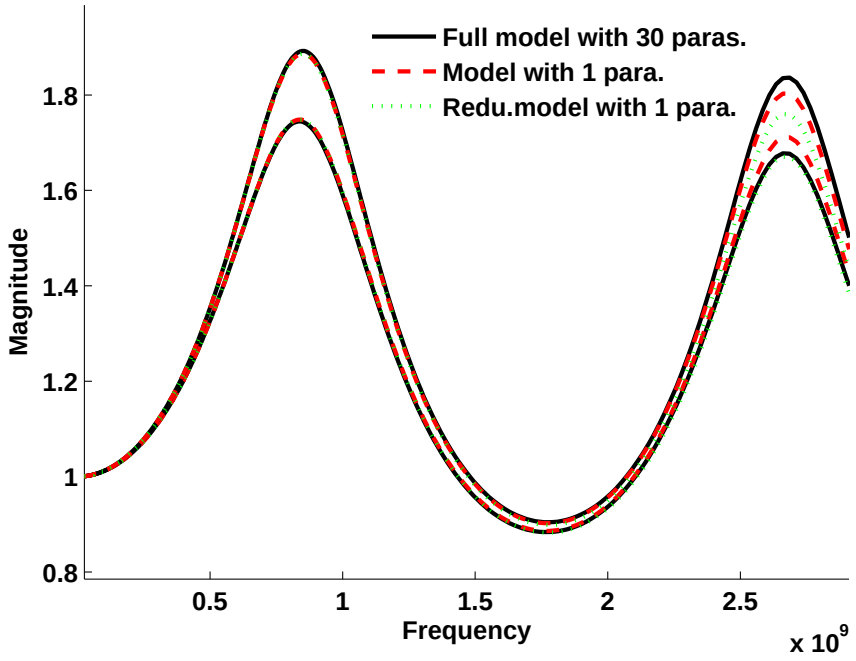

Fig. 14. Frequency responses of various models for the RLC line.

full model and the one-parameter 16th-order reduced model in terms of the frequency response at the outputs. These results show indistinguishable curves for the lower frequency band but larger errors in the higher frequency region. This down-gradation in accuracy is due to the incapability of the first three moments. Setting higher order transfer function moments into the regression model will improve the accuracy.

d. Formation of the Reduced Parameter Set

As mentioned in Section b, the matrix elements in the B_r matrix (2.12) reflect the composition of each new parameter for the given performances, we thereby show a clearer picture of each x_j's statistical importance in the new parameter space. We plot the compositions of the first three new parameters (z_1, z_2 and z_3) of Section a in Fig. 15. We designate the variation sources from each of the ten regions using the

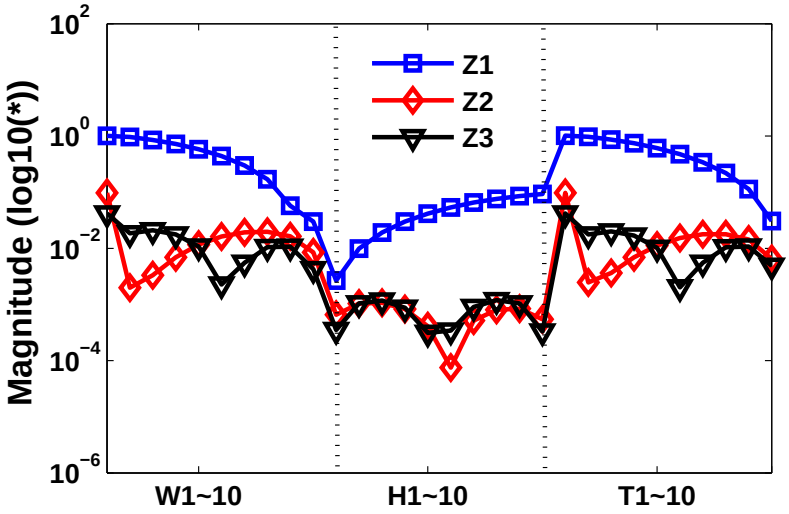

Fig. 15. Composition of the new reduced parameters for the two coupled lines.

corresponding region numbers. It is evident that the wire width and thickness variations, especially those in the first few regions, contribute most to the new parameters, if we only keep the observations at the two output nodes. These results can be well explained by circuit intuition. However, our approach provides a statistical approach to reveal the importance of the variation sources quantitatively.

In Fig. 16 we show the compositions of the first new parameters obtained from the linear RRR and iteration-based nonlinear RRR. As observed, the resultant new parameter set given by the linear RRR method is quite different from the set given by the iteration-based method. This example also indicates the necessity of using this iteration-based algorithm for some strongly nonlinear circuits.

4. Summary

The performance-oriented statistical parameter reduction framework allows us to analyze interconnect variations by reducing the cost of sampling-based simulation and generating very compact parameterized interconnect models with only a few com-

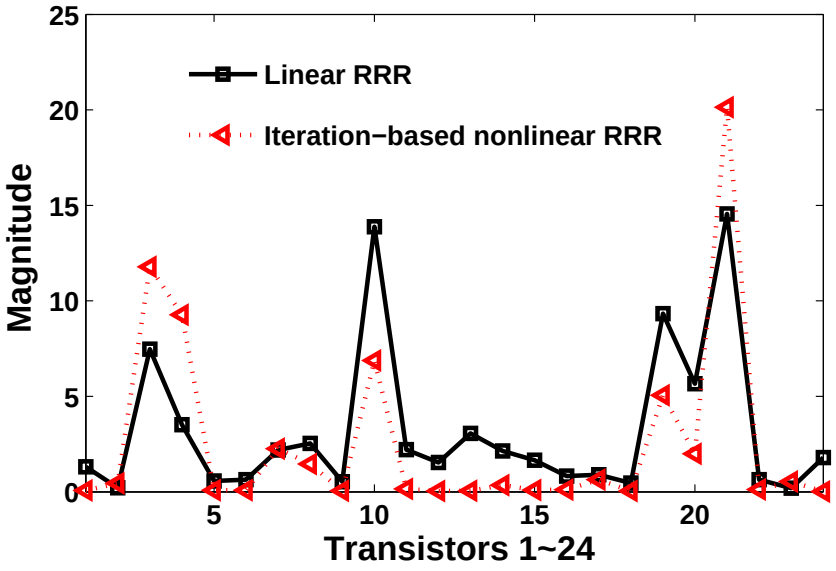

Fig. 16. Composition of the new reduced parameters for ISCAS85 C17.

pressed parameters. Furthermore, it is shown that the same parameter reduction approach can be applied to combinational logic circuits. The presented framework provides a much needed tool to control the explosion of statistical circuit analysis considering high-dimensional inter/intra-die variations.

D. SICE: Statistical Interconnect Corner Extraction Using Parameter Reduction

While traditional worst-case corner analysis is often too pessimistic for nanometer designs, full-blown statistical circuit analysis requires significant modeling infrastructures. In this section, a design-dependent Statistical Interconnect Corner Extraction (SICE) methodology is proposed [36]. SICE achieves a good trade-off between complexity and pessimism by extracting more than one process corners in a statistical sense, which are also design dependent. Our new approach removes the pessimism incurred in prior work while being computationally efficient. The efficiency of SICE comes from the use of parameter dimension reduction techniques and an effective

parametric timing metric. The statistical corners are further compacted by an iterative output clustering method. Numerical results show that SICE achieves up to $260X$ speedups over the Monte Carlo method.

1. Background

Interconnect delay variations are becoming increasingly important to capture in design due to continuous technology scaling [1]. Ideally, a full blown statistical design methodology that can take into consideration various systematic and random process variations is desirable. However, the adoption of such statistical design methodology in practice requires significant investment in modeling infrastructure. On the other hand, corner based methods are simple to apply and do not require significant change of existing design flows. However, they have two well known limitations: pessimism and inconsistence with performance corners. The latter is particularly true for interconnects, where extreme process corners do not necessarily correspond to performance corners. Hence, a systematic corner based methodology that reduces pessimism and correlates well with performance variations is highly desirable. Traditional process corner analysis (PRCA) has been widely used in industry for estimating the interconnect timing variations due to its simplicity (a few process corners are evaluated at $C_{max}, C_{min}, RC_{max}$ and RC_{min} corners), however, with the issues we discussed above.

In this paper, an efficient approach, SICE, is proposed for extracting the interconnect design-dependent parameter/performance corners that can be used for variation-aware timing analysis. The main contribution of this work includes:

- We apply the parameter dimension reduction method [31] to reduce the inter/intra-die variations, thus to alleviate the design-dependent corner finding cost.
- We adopt an efficient timing metric (D2M metric [37]) for performance corner

finding, which is very efficient to compute and able to capture the fidelity/trend of the performance variations.

- We propose an iterative sink nodes clustering algorithm to reduce the number of design-dependent corners.

We show that finding the design-dependent interconnect process corners that correspond to the best/worst performance corners can be very simple and efficient (without a single circuit simulation). The parameter dimension reduction technique can significantly improve the overall extraction efficiency.

2. Preliminaries

a. Process Variation Model

For interconnect circuits, the global variables (inter-die variation sources) typically refer to the process parameters that impact the whole chip, such as the dielectric thickness (H_i) and dielectric constants (ϵ_i) for metal layer i. The process parameters such as the metal width (W_i) and metal thickness (T_i) for metal layer i are usually modeled as the local variables (intra-die variation sources), since these process parameters are usually impacting much smaller areas with various spatial correlation properties [1]. In this work we model the statistical distributions of the process parameters as multivariate normal distributions. Similar to the prior work [8], we use G_0 and C_0 to represent the nominal system matrices while G_i and C_i denote the sensitivity matrices $w.r.t$ the underlying parameter p_i for an parametric interconnect network.

b. Parametric Interconnect Circuit Modeling

Similar to the prior work [8], we use the standard modified nodal analysis (MNA) equations to describe an interconnect network. Consider a set of n_p local and global geometrical variation variables: $\vec{p} = [p_1, p_2, \cdots, p_{n_p}]^T$ that impact the system equations

$$\begin{cases} [G(\vec{p}) + sC(\vec{p})]\, x = Bu \\ y = L^T x \end{cases}, \tag{2.33}$$

where $u \in R^n$ and $y \in R^m$ represent the inputs and outputs, while $x \in R^N$ represents the system unknowns. The parametric conductance and capacitance matrices are defined as:

$$G(\vec{p}) = G_0 + \sum_i^{n_p} G_i p_i;\ C(\vec{p}) = C_0 + \sum_i^{n_p} C_i p_i, \tag{2.34}$$

where G_0 and C_0 represent the nominal system matrices while G_i and C_i denote the sensitivity matrices $w.r.t$ the underlying parameter p_i. $B \in R^{N\times n}$ and $L \in R^{N\times m}$ are the input and output matrices, respectively.

The nominal q_{th} $(q = 0, 1, ...)$ order transfer function moment of the above system is defined as:

$$m_q = (-G_0^{-1}C_0)^q G_0^{-1} B. \tag{2.35}$$

The parametric forms (in terms of the parameter set $\vec{p}$) of the above transfer functions have been derived in [32], where it is shown that the first and second-order parameter coefficients of $\vec{p}$ of the transfer function moments can be very efficiently computed by reusing the LU factors of G_0 in (2.34).

3. SICE Overview

The main steps of our design-dependent interconnect corner extraction algorithm, SICE, are briefly depicted as follows (Fig. 17):

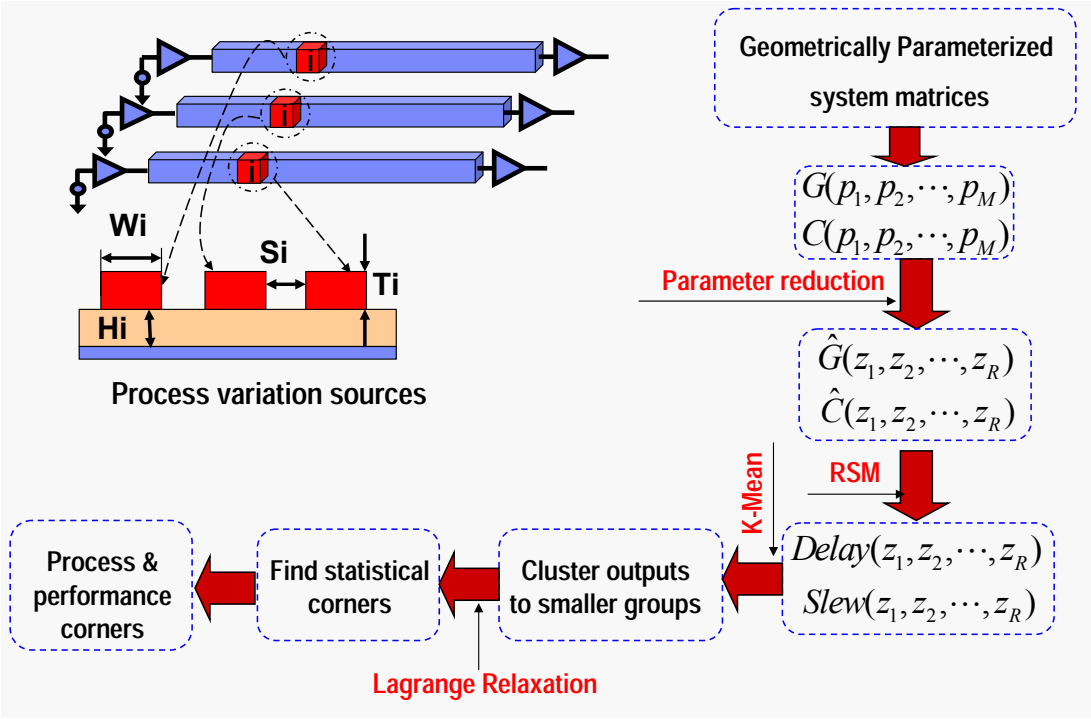

Fig. 17. Overall flow of SICE.

- *Step 1*: Compute the sensitivity matrices (2.34) according to the input R, C sensitivity netlist that can be generated by commercial tools such as [38];
- *Steps 2*: Perform parameter dimension reduction technique [31] to reduce the parameter dimensionality;
- *Step 3*: Build quadratic timing models (response surface models) for the targeted sink nodes in the reduced parameter space ($\vec{z}$);
- *Step 4*: Find design-dependent process corners in the reduced parameter ($\vec{z}$) space iteratively for a desired confidence level.

The outputs of SICE can be of the following three types:

- *Process corner netlists*: SICE provides the process corners for obtaining performance corners. The process corners tell how to perturb the global variables (

such as H_i) and the local variables (such as W_i and T_i) within a specific grid, such that the performance corners can be reached.

- *Performance corner results*: Once the process corners are found, SICE can simulate these corners to obtain their corresponding performance corners.

- *R, C corner netlists*: SICE also provides the R, C netlists that will produce the performance corners. Such R, C netlists can be further reduced [39] and combined with driver models [40, 41, 42] for the stage delay corner characterizations.

4. Parameter Dimension Reduction

Algorithm 3 Interconnect Parameter Reduction

Input: The nominal system matrices $(\mathbf{G_0}, \mathbf{C_0})$, the sensitivity matrices $(\mathbf{G_i}, \mathbf{C_i})$ and the covariance matrix $\mathbf{\Sigma_{\tilde{p}\tilde{p}}}$ of the original parameter set $\mathbf{\tilde{p}}$, error tolerance ϵ.
Output: The sensitivity matrices $(\mathbf{G_{z_i}}, \mathbf{C_{z_i}})$ of the reduced parameter set $\mathbf{\tilde{z}}$, the dimension reduction mapping matrix $\mathbf{B_r}$ and the inverse mapping matrix $\mathbf{T_r}$.

1: Compute the transfer function moment sensitivity matrix $\mathbf{S}$ using the formulas in [32];
2: Set $\mathbf{D} \leftarrow \mathbf{S\Sigma_{\tilde{p}\tilde{p}}S^T}$;
3: Do eigen-decomposition for $\mathbf{D}$ matrix such that $\mathbf{D} = U\Lambda U^T$ to get all the eigenvalues $\lambda_\mathbf{i}$ (in descending order) and the corresponding eigenvectors $\mathbf{u_i}$;
4: Use the $\mathbf{n_z}$ largest eigenvalues and their corresponding eigenvectors to form a diagonal matrix $\mathbf{\Lambda_r}$ and a matrix $\mathbf{U_r}$;
5: Set $\mathbf{B_r} \leftarrow \mathbf{\Lambda_r^{-1/2} U_r^T S}$;
6: Set $\mathbf{T_r} \leftarrow$ pseudo inverse of $\mathbf{B_r}$;
7: For $\mathbf{k = 1}$ **to** $\mathbf{n_z}$:
8: set $\mathbf{G_{z_k}} = \sum_{\mathbf{i=1}}^{\mathbf{n_p}} \mathbf{G_i T_r(i,k)}$; $\mathbf{C_{z_k}} = \sum_{\mathbf{i=1}}^{\mathbf{n_p}} \mathbf{C_i T_r(i,k)}$;
9: Return $\mathbf{G_{z_k}}$, $\mathbf{C_{z_k}}$, $\mathbf{B_r}$ and $\mathbf{T_r}$.

For interconnect circuits, it has been shown that linear reduced rank regression (RRR) method can achieve more reductions of interconnect parameters than principal component analysis (PCA) [31], since RRR is a design-dependent methodology

while PCA merely relies on the parameter data. To perform RRR based parameter dimension reduction, the covariance matrix $\Sigma_{\vec{p}\vec{p}}$ of the local/global process parameters can be constructed based on the correlation models provided by foundries. In this work, we use the distance-based correlation formula $Cor\,(i,j) = e^{-\frac{Dist(i,j)}{CorLength}}$ to model the spatial correlation of the intra-die variation, where $Dist\,(i,j)$ denotes the distance between grids i and j, while $CorLength$ represents the correlation length of the parameter. An error tolerance ϵ is used to truncate the reduced parameter set by keeping only the top few dominant reduced parameters in $\vec{z}$. For R, C interconnect circuit, only the first order sensitivities of the transfer function moments $w.r.t\ \vec{p}$ are needed in parameter reduction, which can be computed efficiently [32] by reusing the LU factorization of the nominal conductance matrix G_0 of (2.34). Assume there are n_m sink node moments to be considered as targeted output in parameter reduction, then we have to compute the sensitivity matrix $S \in R^{n_m \times n_p}$ [32] and express the transfer function moments ($\vec{m} \in R^{n_m}$) of the sink nodes as:

$$\vec{m}(\vec{p}) \approx \vec{m}_0 + S\vec{p}. \tag{2.36}$$

Algorithm 3 is applied to reduce the interconnect parameters, which transforms the original sensitivity matrices G_i and C_i in (2.34) into the sensitivities of the reduced parameters, G_{z_i} and C_{z_i}, and yields an alternative parametric system:

$$\begin{cases} (G(\vec{z}) + sC(\vec{z}))\, x = Bu \\ y = L^T x \end{cases}, \tag{2.37}$$

where

$$G(\vec{z}) = G_0 + \sum_i^{n_z} G_{z_i} z_i;\ C(\vec{z}) = C_0 + \sum_i^{n_z} C_{z_i} z_i. \tag{2.38}$$

Compared with (2.33), the above system (2.37) has much fewer parameters ($n_z << n_p$). The algorithm also generates the parameter reduction mapping matrix B_r which

maps the original parameter set $\vec{p}$ to the reduced parameter set $\vec{z}$ by $\vec{z} = B_r\vec{p}$. Essentially, in Algorithm 3, the correlations between parameter variations and the resulting performance variations are utilized to identify a reduced set of parameters that are statistically critical to the performances. This leads to a design dependent parameter reduction [31]. A unique feature of the reduced parameter set $\vec{z}$ is that they are uncorrelated normal variables with $N(0,1)$ distributions [28]. The inverse mapping matrix T_r which is the pseudo inverse of B_r, maps $\vec{z}$ to $\vec{p}$ by $\vec{p} = T_r\vec{z}$. As we will describe later, these reduced parameters ($\vec{z}$) can significantly simplify the parametric timing model generation, design-dependent interconnect corner extraction and the process corner clustering procedures. Additionally, by using the mapping matrix T_r, we are able to map the process corners in $\vec{z}$ to the original process corners in $\vec{p}$.

5. Parametric Timing Model

After we obtained the reduced parameter set $\vec{z}$, the interconnect timing model can be parameterized in $\vec{z}$ via statistical modeling techniques such as design of experiment (DOE) [43] or Latin Hypercube Samplings (LHS) [33].

Quadratic interconnect timing model is essential for capturing the nonlinear performance variations due to the underlying process parameters, which typically requires $O(n_p^2)$ data samples to generate. However, existing interconnect simulation methods are usually impractical to utilize due to the high simulation cost. On the other hand, it is not necessary to build an absolutely accurate performance model, since our interests lies in finding the process corners that correspond to the performance corners. In [44, 45], the authors successfully use the Elmore delay (ED) to find accurate process corners (corresponding to the performance corners) for long interconnects, since ED can very well capture the variation trend/fidelity of the true performance. However, ED may not be a good metric for estimating the variational

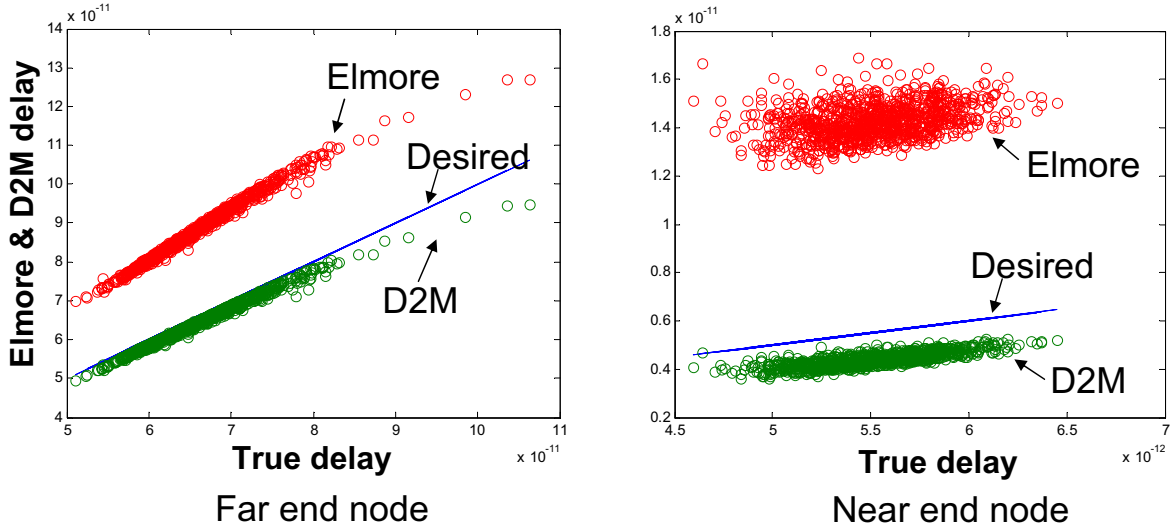

Fig. 18. Comparison of two interconnect timing metrics.

trend/fidelity of near-end nodes. As an example, we use an interconnect designed on 65nm technology, where the dielectric thickness, the metal width and thickness variations are considered. The R, C sensitivities due to these parameters are calculated using the closed form formulas [34] and the R, C elements are divided into a few grids for intra-die correlation modeling purpose. Random simulation results in Fig. 2 show the scatter plots of the true delay (TD) w.r.t the other two delay metrics (Elmore and D2M). As observed, both timing metrics correlate well with the true delay values for the far-end node, while for the near-end node, D2M delays exhibit much better correlation with the true delays (as shown in Fig. 18). In this work, we balance the efficiency and accuracy by using the D2M metric [37] for timing model generation. The D2M delay is given as:

$$Delay = \frac{m_1^2}{\sqrt{m_2}} \ln 2, \tag{2.39}$$

where m_1 and m_2 are the first and second transfer function moments, respectively. To simplify the following design-dependent process corner finding step, we generate

quadratic timing model by sampling in the reduced parameter space $\vec{z}$ and computing the standardized D2M delay samples $Y = (Y_t - \bar{Y}_t)/\sigma_{Y_t}$, where Y_t represent the D2M delay values. The second-order parametric transfer function moments [32] have been used for computing the D2M delays to further improve the efficiency.

6. Design-Dependent Corners

We describe in detail the procedures of finding the design-dependent (application-specific) corners for interconnect circuits. This section first introduces how to find such corners for a single sink node based upon its quadratic timing model that is obtained in advance (Section 5). Then we propose an iterative clustering algorithm that clusters the sink nodes into a smaller number of groups, such that only a few representative process corners are needed to predict the true performance corners of all these sinks.

a. Design-Dependent Corner Analysis

Assume the the quadratic timing models for all sink nodes (say n_s sink nodes) are generated in the reduced parameter space $\vec{z}$. The timing model for sink node k is given as follows:

$$y_k(\vec{z}) = \vec{z}^T A_k \vec{z} + B_k^T \vec{z} + C_k, \tag{2.40}$$

where A_k, B_k and C_k are the second-order, first order and constant coefficients. We follow the corner extraction methodology proposed in [46] to find n_s pairs of best/worst design-dependent process corners for n_s sink nodes, respectively. The corner extraction for sink node k can be formulated as the following optimization problem:

$$\max / \min \left\{ y_k(\vec{z}) = \vec{z}^T A_k \vec{z} + B_k^T \vec{z} + C_k \right\}, s.t. \, \|\vec{z}\| = \alpha. \tag{2.41}$$

where α is used to define the confidence region of the parameter space. As discussed in Section 4, all the reduced parameters in $\vec{z}$ are uncorrelated normal variables with $N(0,1)$ distributions. Therefore, the concept of ellipsoid confidence region [47] of $\vec{p}$ now becomes a hypersphere confidence region of $\vec{z}$. The confidence levels of the corners found using Lagrange Relaxations can be adjusted by setting different α values. More specifically, since the reduced variables in $\vec{z}$ are independent, then the probability density function (pdf) of $\vec{z}$ becomes:

$$pdf\ (\vec{z}) = (2\pi)^{-\frac{n}{2}}\, e^{-\frac{1}{2}\vec{z}^T\vec{z}} = (2\pi)^{-\frac{n}{2}}\, e^{-\frac{1}{2}\|\vec{z}\|^2}. \tag{2.42}$$

Consequently, $pdf\ (\vec{z})$ is constant when $\|\vec{z}\|^2$ is constant, and the probability $P\{\vec{z} \mid \vec{z}^T\vec{z} \leq \alpha^2\}$ can be used to determine the confidence region in the performance space. Since $\|\vec{z}\|^2 = \sum_{k=1}^{n_z} z_k^2$ has a chi-square distribution with degree n_z, we can therefore compute α^2 for a desired $P\{\vec{z} \mid \vec{z}^T\vec{z} \leq \alpha^2\}$ by evaluating the inverse of the cumulative distribution function (cdf) of $\|\vec{z}\|^2$.

b. Iterative Sink Node Clustering

It is worthwhile to emphasize that (2.41) can be solved in the reduced parameter space $\vec{z}$, which typically has a much lower dimensionality than $\vec{p}$. The corner finding efficiency can therefore be significantly improved than ever before. Assume we have computed n_s pairs of best/worst design-dependent corners for all the n_s sink nodes already, but these $2n_s$ corners can be difficult to utilize in practical applications due to the high complexity. To reduce the number of corners, a clustering algorithm has been proposed [46], where clustering is performed on the performance sample data. Unfortunately, reducing the number of corners using this method may produce more conservative corners. In this work, we propose to do clustering in the reduced parameter space $\vec{z}$, by clustering the $2n_s$ best/worst parameter corners. For each sink

node (say node k), we introduce a new corner vector $\vec{z}_k$ to include its best/worst parameter corners as:

$$\vec{z}_k = \begin{bmatrix} \vec{z}_{bst,k} \\ \vec{z}_{wst,k} \end{bmatrix} \tag{2.43}$$

Then we adopt the K-mean algorithm to cluster $\vec{z}_k$ ($k = 1, ..., n_s$) into fewer groups. It can be shown that the reduced parameter size (n_z) obtained by RRR is smaller than the number of observation points (n_s) [28]. In fact, for most interconnect circuits, n_z is much smaller than n_s. Therefore, clustering using $\vec{z}_k$ instead of the performance samples may greatly reduce the clustering effort. Since K-mean clustering results are very sensitive to the number of clusters, so we propose an iterative clustering method. The algorithm uses an initial guess on the number of clusters (we choose $n_s - 1$ as the initial number). Then the optimization problems similar to (2.41) can be solved to find the representative corners for the clusters where A_k and B_k for a single node are replaced by the representative coefficients A'_i and B'_i (for cluster i):

$$A'_i = \sum_{k \in Cls_i} w_k A_k, B'_i = \sum_{k \in Cls_i} w_k B_k. \tag{2.44}$$

Then we follow (2.41) to find the best/worst corners for each of these clusters. For accuracy purpose, these new parameter corners are substituted into the timing models for all sink nodes (2.40), to compute their performance corners. If the errors of the performance corners are within the tolerance, the number of clusters is reduced and another K-mean clustering is performed. By repeating the above procedures several times, we can determine the minimum number of clusters, which can produce accurate performance corners. Finally, the representative parameter corners are obtained based on the final clusters.

7. Algorithm Complexity

The algorithm complexity of each step in SICE can be analyzed as follows:

- *1. Parameter Reduction:* The computation of the linear sensitivities (S matrix in Algorithm 3) of a few transfer function moments $w.r.t$ to the original parameters $\vec{p}$ (for R, C circuit, only the first moment is adequate) is trivial, where only one time LU matrix factorization of G_0 and $\mathrm{O}(n_p)$ times reuse of the LU factors are needed;

- *2. Timing Models Generation:* We compute the quadratic forms of the transfer function moments by reusing the LU of G_0, to obtain the D2M delay samples and build the timing models (2.40) for all sink nodes. Thus the computational cost of this step mainly attributes to $\mathrm{O}(n_z^2)$ times reuse of the LU factors of G_0;

- *3. Design-dependent corner extraction:* A few runs of the optimizations (2.41) and K-mean clusterings in the reduced parameter space are required. The cost is negligible for the circuits that have few sink nodes, when compared with the matrix factorization cost.

Consequently, the main cost of SICE algorithm is the one-time matrix factorization of G_0, and $\mathrm{O}(n_p + n_z^2)$ times reuse of the LU factorizations.

8. Numerical Results

SICE is implemented in C++ and executed on a Pentium-4 3GHz machine running Linux system. Since We first compare SICE with the traditional process corner analysis (PRCA) method. Next, we present the confidence-region aware corners given by SICE. Finally, we compare the parameter reduction results obtained by RRR and PCA on various circuit examples, where the runtime are also compared against the

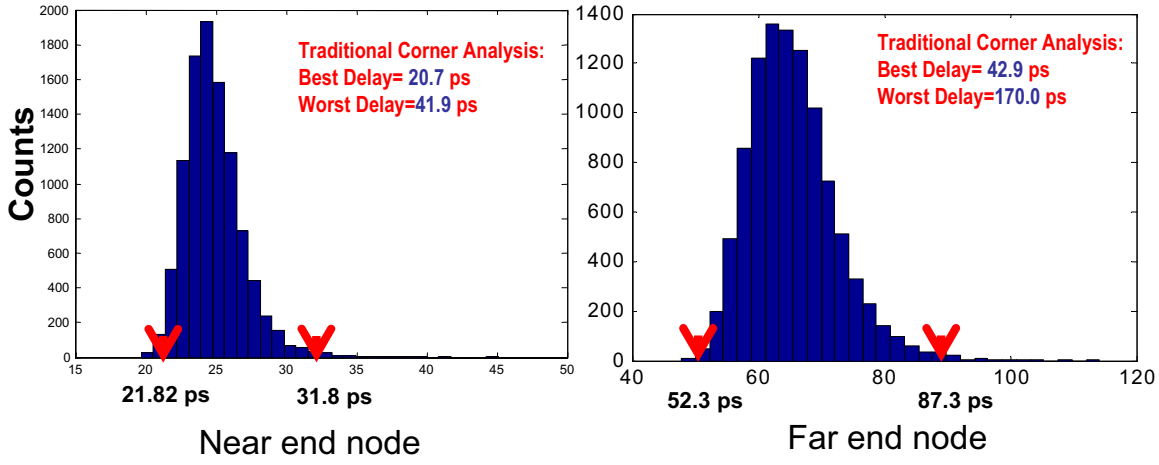

Fig. 19. SICE vs. 10k Monte Carlo (std= 10%).

Monte Carlo results. All circuits examples in this section are designed in 65nm technologies. We consider the inter-die variation (dielectric thickness H) as well as the intra-die variations (T and W). The R, C sensitivities $w.r.t$ the geometric parameters $\vec{p}$ are calculated using closed forms [34]. All parameters in $\vec{p}$ are set to have $std = 10\%$ variations by default.

a. SICE vs Process Corner Analysis (PRCA)

We compare the design-dependent corner extraction results of SICE with the traditional process corner analysis results on a ten-grid R, C interconnect circuit with 12 sinks as targeted outputs. SICE produces two reduced parameters and finds two clusters of sink nodes for corner extraction. The 99% performance corners are compared with the PRCA results in Fig. 19 and Fig. 20, where the PDF of 10K Monte Carlo simulations are also shown. We can find that in both cases, SICE predicts pretty accurate corners even for very large variations, while PRCA usually gives very conservative ones (especially the worst case delays).

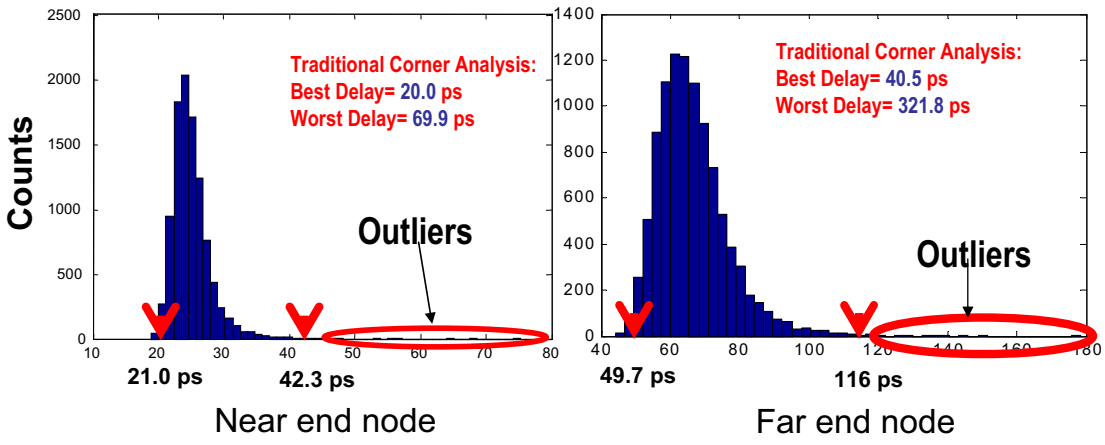

Fig. 20. SICE vs. 10k Monte Carlo (std= 15%).

b. Corners by Different Timing Models

As described in Section 5, the D2M delay is more accurate than the Elmore delay (E. D.), thus should provide more accurate design-dependent corners. We consider a ten-grid R, C clock tree circuit with three sink nodes (sinks 1-3) as targeted outputs. Two sets of correlation lengths ($Cor_a = 5 \times Cor_b$) are considered for modeling the intra-die variations. The 99% performance corners obtained by SICE using three timing models are shown in Table II. The maximum and minimum delay values of 1K Monte Carlo simulations are also attached, to illustrate the realistic performance corners. It is worth noting that SICE may produce pessimistic worst case corners if the performance variation is large (1a-3a), which is due to the insufficiency of the quadratic timing models used for corner finding.

c. Confidence-Region Aware Corners

We illustrate the confidence-region aware SICE corners for a four-grid R, C clock circuit with four sink nodes as targeted outputs in Fig. 21. SICE produces two reduced parameters and finds two clusters for corner extraction. The 99%, 95%

Table II. Corner accuracy using the D2M timing model under different correlation models

Sink	Accurate Model		D2M Timing Model				1K Monte Carlo	
Index	Best	Worst	Best	Rel. Err.	Worst	Rel. Err.	Min	Max
S1a	36.3	88.5	36.3	0.13%	98.5	11.35%	36.9	85.7
S2a	8.7	16.0	8.6	-1.35%	16.9	5.69%	8.1	13.1
S3a	49.3	111.7	49.2	-0.31%	121.0	8.28%	44.22	125.4
S1b	38.1	78.0	38.0	0.3%	78.2	0.3%	37.1	72.2
S2b	7.65	14.8	7.6	-0.6%	15.0	0.13%	8.1	13.2
S3b	46.2	102.3	45.5	-1.5%	101.9	0.39%	46.1	95.8

Table III. Parameter reduction comparisons (RRR/PCA)

Circuits setups					SICE Results	
CKT	Size	N_{Sinks}	N_{Grids}	n_p	n_z (RRR/PCA)	$N_{Corners}$
C1a	200	12	6	13	3/9	4
C1b	200	12	10	21	4/17	6
C2a	55	6	4	9	2/7	4
C2b	55	6	6	13	3/11	6
C3a	10	2	2	5	2/5	2
C3b	10	2	4	9	3/9	4

Table IV. Runtime comparisons (SICE/1K Monte Carlo)

Circuits setups					SICE time T_{SICE} (ms)				T_{MC} (ms)	Sp.
CKT	Size	N_{Sinks}.	N_{Grids}	n_p	T_{PR}	T_{TM}	T_{CC}	T_{SC}	T_{MC}	T_{MC}/T_{SICE}
C1a	200	12	6	13	2.9	3.6	4.1	36.1	9106	195X
C1b	200	12	10	21	4.6	5.4	5.2	54.7	9380	134X
C2a	55	6	4	9	0.6	0.8	1.5	12.8	2857	182X
C2b	55	6	6	13	0.6	1.4	2.1	18.3	2934	131X
C3a	10	2	2	5	0.1	0.26	0.7	1.2	588	260X
C3b	10	2	4	9	0.3	0.54	0.7	2.5	848	210X

and 90% performance corners are compared against the 5K Monte Carlo simulation results. As observed, these confidence-region aware corners are reasonably accurate for the non-normal performance distributions.

d. Parameter Reduction and Runtime

We demonstrate three interconnect circuits in six cases, where each of the circuit uses two correlation models ($Cor_a = 2 \times Cor_b$). For each circuit case, we compare the parameter reduction results (N_z) using RRR and PCA under the same error tolerance levels. From Table III we find RRR based parameter reduction method always achieves $2 - 3X$ more reductions than PCA.

SICE runtime (T_{SICE}) is also compared against 1K Monte Carlo (MC) runtime in Table IV, which includes four parts: the parameter reduction time (T_{PR}), the parametric timing model generation time (T_{TM}), the design-dependent corner finding time (T_{CC}) and the performance corner simulation time (T_{SC}). We use a linear circuit simulator for MC and the performance corner simulations. In each transient simulation, the conductance matrix will be factorized once and the LU factors will be

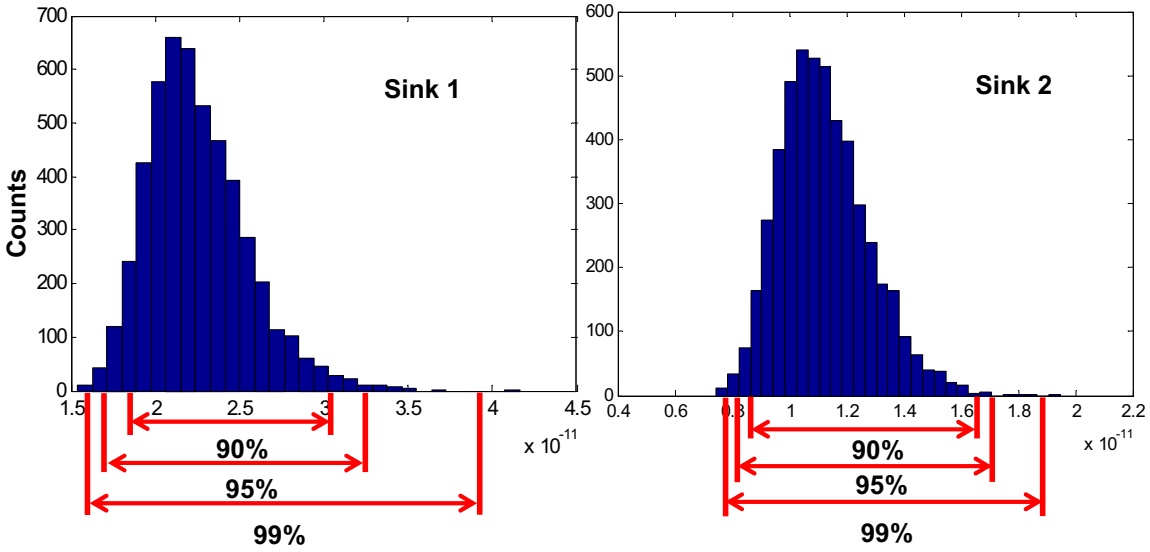

Fig. 21. Confidence aware timing corners vs 5k Monte Carlo simulations (std= 10%).

reused afterwards (100 time steps are used for all cases). As observed in all cases, the performance corner simulations of SICE take 1/2 to 4/5 SICE runtime. The overall performance of SICE is $131X$ to $260X$ faster than the MC simulations.

9. Summary

A design-dependent Statistical Interconnect Corner Extraction (SICE) methodology is proposed for efficient and accurate interconnect performance corner extraction under process variations. SICE removes the pessimism produced by the previous process corner based analysis methods and achieves much higher efficiency than the traditional statistical methods. Statistical parameter dimension reduction technique helps to dramatically reduce the dimensionality of the complex variation sources without loss of accuracy. An effective timing metric for parametric timing model generation as well as an iterative sink nodes clustering method are proposed to facilitate the design-dependent performance corner finding algorithm. Experiments on various

interconnect circuits are demonstrated for validation.

E. Second-Order Statistical Static Timing Analysis Using Parameter Reduction

While first-order SSTA techniques enjoy good runtime efficiency desired for tackling large industrial designs, more accurate second-order SSTA techniques have been proposed to improve the analysis accuracy, but at the cost of high computational complexity. Although many sources of variations may impact the circuit performance, considering a large number of inter-die and intra-die variations in traditional SSTA analysis is very challenging. In this work, we address the analysis complexity brought by high parameter dimensionality in statistical static timing analysis and propose an accurate yet fast second-order SSTA algorithm based upon novel on-the-fly parameter dimension reduction techniques. By developing reduced-rank regression based and the moment-based parameter reduction algorithms within block-based SSTA flow, we demonstrate that accurate second-order SSTA analysis can be extended to a much higher parameter dimensionality than what is possible before. Our experimental results have shown that the proposed parameter reductions can achieve up to $10X$ parameter dimension reduction and lead to significantly improved second-order SSTA analysis under a large set of process variations.

1. Background

As a very popular research topic in the past few years, various Statistical Static Timing Analysis (SSTA) algorithms [48, 49, 50, 51, 52, 53, 54, 55, 56] have been proposed to predict the impacts of process variations on circuit performance. It has tremendous advantages over the traditional corner based timing analysis, in which the number of corners may increase exponentially with the number of process variations. In com-

parison, SSTA computes the statistical variations of timing performance to provide more accurate and realistic estimates, while maintaining relatively low analysis complexity. Several linear SSTA algorithms have been proposed to achieve high runtime efficiency, which is desirable for large industrial designs [48, 49, 51]. However, due to the linear approximations of variational device models as well as the atomic *max* operations, such efficient algorithms may not be accurate in timing analysis. To improve the accuracy, on the other hand, second-order SSTA techniques [54, 56, 55] are very attractive in terms of robustness and accuracy despite of their higher computational cost. Among the above high order SSTA algorithms, the ones proposed in [54, 55] can give very accurate results, but may be limited to a small number of parameters due to the dramatic growing of the analysis cost. The other algorithm [56] adopts a simple and efficient linear approximation for the *max* operation, which makes it suitable for dealing with a large number of parameters. Unfortunately, as reported in their paper, this algorithm may be less accurate in some cases [57].

In reality, the performances of modern designs may be severely influenced by a large set of independent or correlated inter-/intra-die variations. This fact creates an immediate need to consider these effects efficiently and accurately in circuit analysis, such as in SSTA. Hence, it is very appealing to develop SSTA techniques that are capable of handling a large set of random parameters, and at the same time, being efficient. In this work, we propose to extend the applicability of the existing second-order SSTA techniques to a higher dimensional parameter space. To this end, the key issue is to control the complexity introduced by multiple parameters properly under the context of SSTA.

In this work, we adopt two techniques, a linear reduced rank regression (RRR) approach [28] and a moment-based dimension reduction (MOM) technique [58, 59], respectively. The first method is suitable for parameter reduction of moderate nonlin-

ear performance variations and usually requires minimum reduction cost. The latter is specifically proposed for more robust and accurate parameter reduction of those performance models with stronger nonlinearity but at a higher cost. The key and common distinguishing factor of these two methods is that they allows more powerful parameter reduction while considering the interdependency between parameters and the corresponding performances. As demonstrated recently in [31], the application of RRR leads to significant parameter reduction of variational interconnect problems. This dissertation extends the work in [60] to develop SSTA-specific RRR based and moment-based parameter dimension reduction techniques to facilitate fast and accurate second-order parameterized SSTA. The efficiency of the proposed SSTA approach stems from the on-the-fly performance-based parameter dimension reduction techniques. The latter helps maintain a low effective number of process parameters that need to be considered by exploiting statistical parameter redundancy imposed by design structure.

a. Process Variation Model

Let us consider a set of $X_g \in \mathbb{R}^{n_g}$ global (die-to-die) process variations and a set of $X_l \in \mathbb{R}^{n_l}$ local (within-die) process variations. In this work, we use the process variation model in Fig. 22 to model the impacts of these global and local variations on circuit timing performance [49]. The global variations impact the complete die while each local variation is only impacting a local region on the die.

Although the number of the global variation sources may be small, the number of local variations can be quite large for modeling intra-die spatial variations. Hence, it is prohibitive to simultaneously consider all the intra-die variations in the traditional SSTA algorithms. As shown in Fig. 22, even though a local variable may only impact device parameters in a small local region, its impacts can propagate to other parts

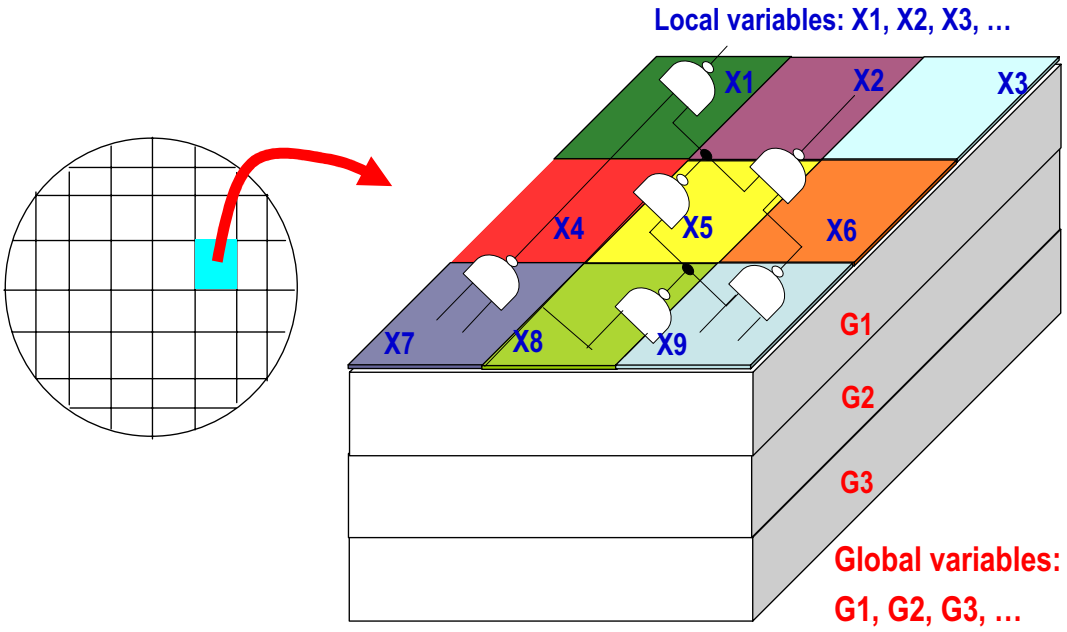

Fig. 22. Process variation models.

of the circuit through the fanout signal paths, some of which may be reconvergent fanouts. Because of reconvergent fanouts, handling a large set of local variables becomes difficult even under the case where local variables are independent from each other. In the rest of this section, we assume all the local and global variations are multivariate Gaussian variables with zero means.

b. Block-Based SSTA

Block-based SSTA algorithms walk through the whole circuit by a breadth-first search scheme and propagate the probability density functions (PDFs) of signal arrival times from source nodes to sink nodes of a timing graph. First order block-based SSTA algorithms are quite attractive to large industrial designs due to the high runtime efficiency.

However, first order algorithms expand the arrival times in terms of the process

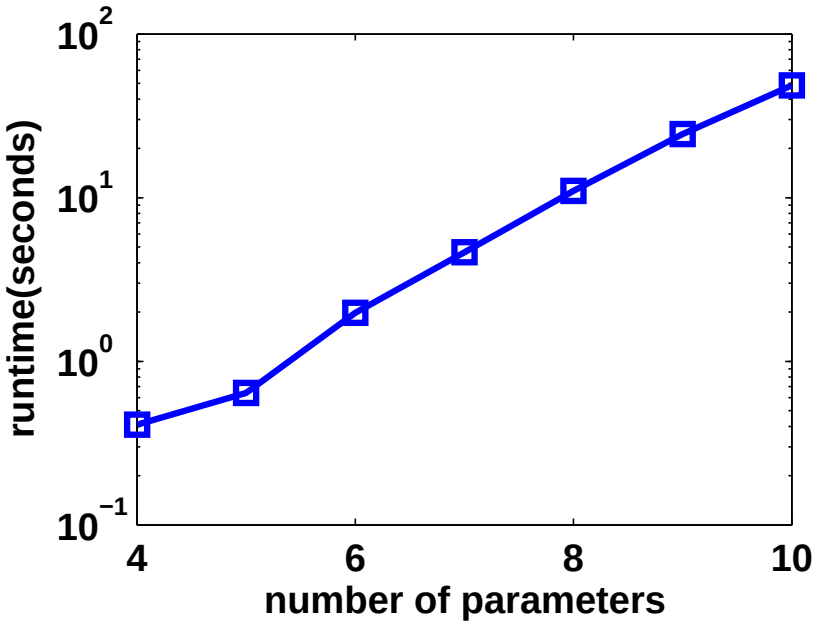

Fig. 23. Moment matching based quadratic SSTA runtime vs. the number of parameters.

variations using the first order canonical forms, therefore they may fail to accurately predict delay distributions due to the high order nonlinearities of *max* operations as well as the gate and interconnect delays. Consequently, several quadratic SSTA algorithms [56, 55, 54] have been proposed to address the above insufficiency. A very efficient *max* operation for computing second-order forms of signal arrival times and delays using fast linear approximations was proposed in [56]. This approach can handle a large number of variations but is limited in accuracy. A moment matching based algorithm was proposed in [55] for highly accurate *max* operations but at the cost of relatively high complexity. Unfortunately, the cost of the moment matching based approach may grow quickly as the number of variaitons increases (Fig. 23). The quadratic SSTA via moment matching is briefly reviewed in the following.

Suppose two signal arrival times y_i ($i = 1$ *and* 2) have the following quadratic

forms:

$$y_i = X^T A_i X + B_i^T X + C_i. \tag{2.45}$$

The sum y_3 of y_1 and y_2 is simply given by:

$$y_3 = X^T A_3 X + B_3^T X + C_3, \tag{2.46}$$

where

$$A_3 = A_1 + A_2, B_3 = B_1 + B_2 \; and \; C_3 = C_1 + C_2. \tag{2.47}$$

Unlike the *sum* operations, y_4, the *max* of y_1 and y_2 is much more difficult and expensive to compute. The algorithm [55] proposes to approximate the PDF of y_4 using a quadratic model,

$$y_4 = X^T A_4 X + B_4^T X + C_4, \tag{2.48}$$

and the coefficients A_4, B_4 and C_4 are computed by matching the first few moments of the PDF functions of y_4. Though this method can compute very accurate PDFs of the *max* operations, the numerical convolutions and integrations in each *max* operation make the total computational cost much higher than the first order SSTA algorithms.

To achieve the best possible statistical static timing analysis considering a large number of process variations, it is desired to develop new SSTA algorithms that can accurately capture the impacts of multiple process variations but at the same time being runtime efficient. Consequently, incorporating parameter dimension reduction techniques into the SSTA algorithms is very desired.

c. PCA-Based Parameter Reduction Before SSTA Starts

Before the standard SSTA algorithm starts, the spatially correlated process variations (Gaussian variables such as $V_{th}, L_{eff}, ...$) are transformed into the uncorrelated ones

using PCA. We show how PCA reduce the dimension of a parameter set before SSTA starts.

Assume $X \in \mathbb{R}^n$ is an n-dimensional data set, which has zero mean and multivariate normal distributions. PCA first computes the eigen-decomposition of the covariance matrix Σ_{xx} of X as follows:

$$\Sigma_{xx} = U \Lambda U^T, \tag{2.49}$$

where Λ is a diagonal matrix containing all the eigenvalues of Σ_{xx}, and U contains all the corresponding orthogonal eigenvectors. By including few eigenvectors (in U) that have the largest eigenvalues into the projection matrix U_r, the new parameter set X_r that has a smaller dimension than the original data set X can be given by

$$X_r = U_r^T X. \tag{2.50}$$

Subsequently, the *sum* and *max* operations of SSTA can be performed upon these uncorrelated variables when computing the parameterized signal arrival times for each node of a timing graph. The objective of statistical static timing analysis is to compute the variations of signal arrival times for all the nodes on a timing graph. Under such context, PCA can improve the efficiency of circuit analysis by identifying the principle components of process variations that impact timing performances. However, PCA only removes the redundancy in the process variation data set without considering circuit structural information (such as the timing graph properties). In practice, such design-independent parameter reduction reduces the cost of the SSTA algorithms, but in a limited way.

d. On-the-Fly Performance-Based Parameter Reduction During SSTA

Basic Idea

In this work, we propose performance-based parameter reduction methods that can be combined with the standard PCA method to produce much more effective dimension reduction during the SSTA flow. In addition to the dimension reduction realized by PCA before SSTA starts, our methods propose to reduce the parameter dimension during the SSTA procedure, by considering the correlation information of the performance space (such as signal arrival times) and its corresponding parameter space at each stage of the SSTA algorithm.

1) Mathematic Methods

By adopting the dimension reductions based on the linear reduced rank regression (RRR) or moment-based (MOM) methods as a systematic tool, we can identify the redundancy (reduced parameters) in the process variables. Considering the process variation model in Subsection a, we show a general mathematical framework under which a large set of local process variations can be reduced for the purpose of timing performance modeling without any other assumptions regarding the statistical property of the random process variables.

2) Implementation Framework

We outline how the theoretical framework of the parameter reductions are applied to the second-order SSTA. For each circuit partition, parameter reduction is conducted once to reduce the number of local process variations and then the second-order SSTA (*sum* and *max* operations) can be performed much more efficiently for the original set of global variations and a reduced set of local variations. The way in which the parameter reduction is combined with SSTA is shown in Fig. 24 and Fig. 25, where parameter reduction is intertwined with each SSTA processing step to dynamically

control the parameter dimension. As described in the following sections, we will demonstrate that by our specific problem formulations, the statistical information inputs to RRR or MOM can be efficiently gathered during a SSTA run, leading to the very much desired design specific parameter dimension reduction capability.

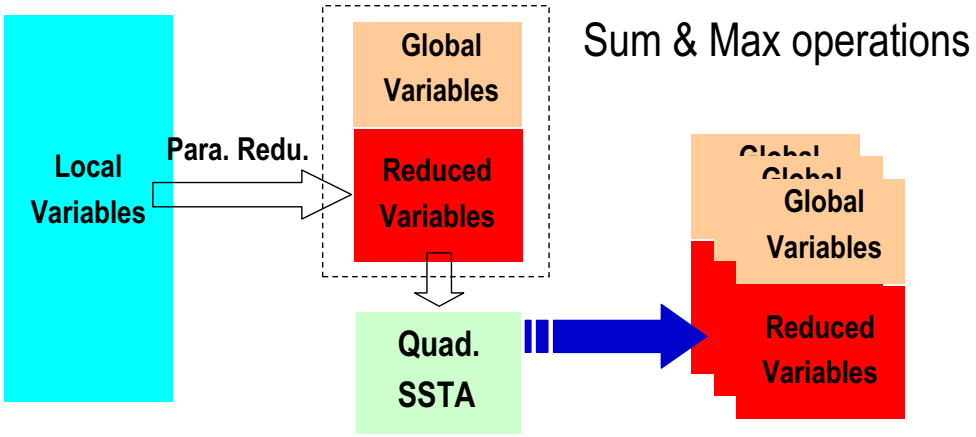

Fig. 24. Second-order SSTA with local variation parameter dimension reductions (conceptual).

3) An Example

The basic idea of our method is demonstrated through an example shown in Fig. 26, where we only consider the spatial correlated intra-die variations (local variables). For simplicity, we only consider the first order parametric form of the signal arrival times, but in reality, our method is not constrained by this assumption. We assume the following standard setup are done before the SSTA starts:

- Use the grid modeling method in [11] to partition the circuit into grids (four grids in this example);
- Apply PCA to map the original correlated parameters ($V_{th}, L_{eff}, etc.$) into the uncorrelated ones (X_1 to X_4) that forms the parameter vector X;
- Extract the delay models based upon the uncorrelated parameter set X;

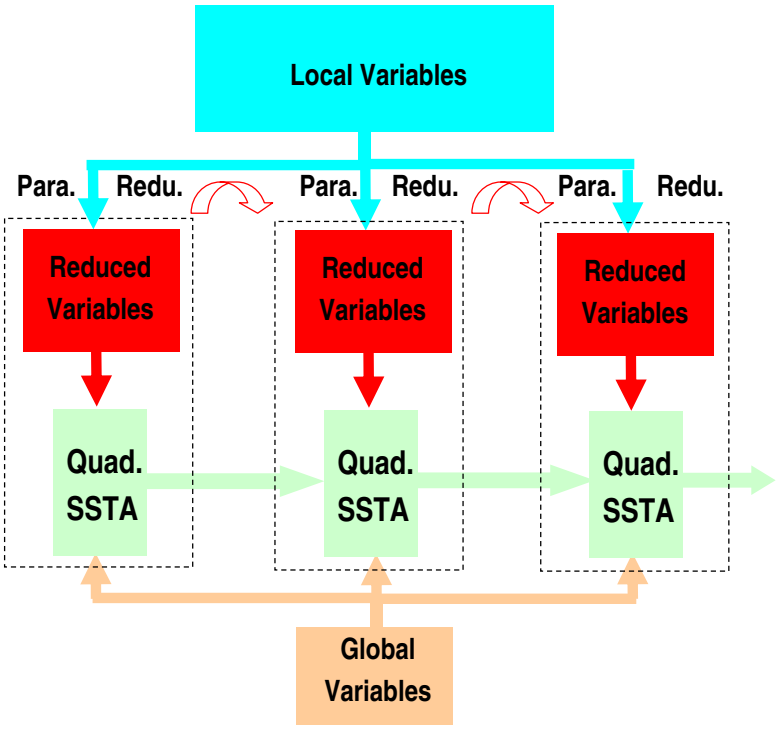

Fig. 25. Second-order SSTA with local variation parameter dimension reductions (actual).

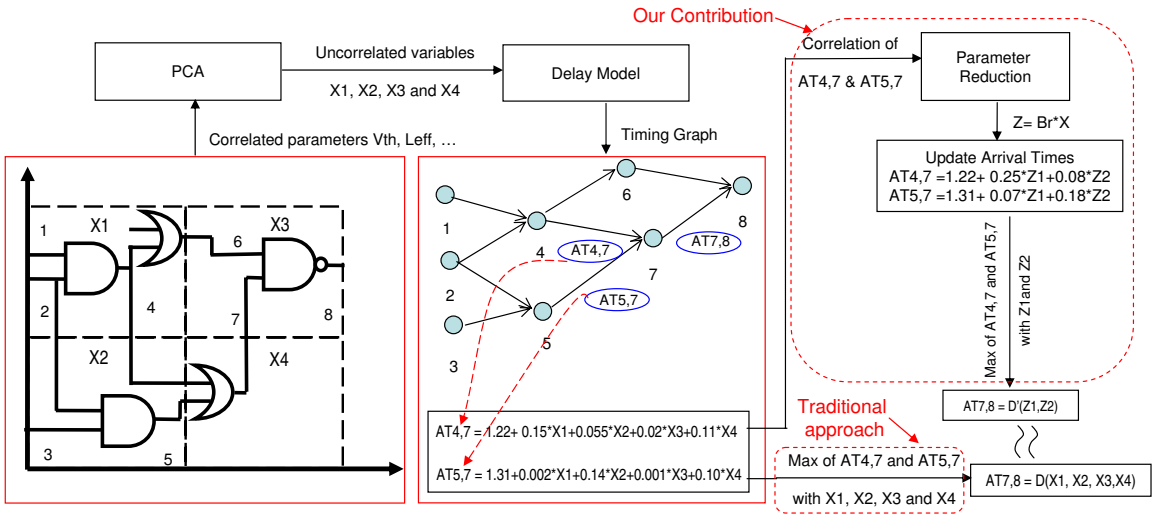

Fig. 26. SSTA flow using parameter dimension reductions.

- Generate the timing graph using the delay models.

In Fig. 26 we can see each of the four circuit grids has one independent Gaussian variable for modeling the intra-die variations. Assume the parameterized signal arrival times $AT_{4,7}, AT_{5,7}$ are given in advance (as shown in the Fig. 26).

Traditional SSTA algorithm computes the signal arrival time $AT_{7,8}$ by performing the *sum* and *max* operations directly on variables X_1 to X_4 (shown on the bottom right of Fig. 26). On the other hand, our method (shown on the top right of Fig. 26) that incorporates the performance-based parameter dimension reduction includes the following extra steps:

- Extract the correlation data of $AT_{4,7}$ and $AT_{5,7}$ with respect to the underlying parameters X_1 to X_4;
- For $AT_{4,7}$ and $AT_{5,7}$, conduct performance-based parameter reduction to obtain the common reduced parameters Z_1 and Z_2 in Z through a mapping $Z = B_r X$;
- Update the parameterized signal arrival times $AT_{4,7}$ and $AT_{5,7}$ in the reduced parameter space;
- Perform *sum* and *max* operations of $AT_{4,7}$ and $AT_{5,7}$ in the reduced parameter space;
- Express $AT_{7,8}$ in Z_1 and Z_2.

Compared with the traditional SSTA, it is obvious that the above method (using parameter reduction) can save a lot of computing efforts for the *max* operations, since much fewer variables are considered.

2. Parameter Reduction Methods for SSTA

In this section, we first introduce the quadratic timing model that is used in the second-order SSTA algorithm. Subsequently, the theoretic background of two parameter dimension reduction techniques, the linear RRR approach [28] and a moment-based dimension reduction technique [58, 59], are introduced respectively. Finally, numerical comparisons on the above two methods are made.

a. Quadratic Timing Model in SSTA

Assume we are considering parameter reduction during SSTA for the m parameterized signal arrival times y_k, where $k = 1, ..., m$. Each of these signal arrival times has the following parametric form:

$$y_k = \bar{y}_k + \underbrace{d_k^T X_g + X_g^T A_k X_g}_{global} + \underbrace{c_k^T X_l + X_l^T B_k X_l}_{local}, \tag{2.51}$$

where $\bar{y}_k$ is the nominal case output value. $d_k \in \mathbb{R}^{n_g}$ and $c_k \in \mathbb{R}^{n_l}$ include the first order coefficients of the global and local variables, while $A_k \in \mathbb{R}^{n_g \times n_g}$ and $B_k \in \mathbb{R}^{n_l \times n_l}$ are the second-order coefficients capturing the dependency of y_k on the global and local variables, respectively. To come up with a common set of reduced parameters for these m signal arrival times, the linear reduced rank regression method and the moment-based method are adopted.

b. Linear RRR with Two Regressors (RRR)

In this subsection, we introduce the linear RRR approach in detail, which is suitable for dimension reduction problem that has moderate nonlinear performance dependence of the variation sources.

1) The Theory

Considering the quadratic model in (2.51), if the second-order terms are not significantly larger than the first order coefficients, it is natural to relate all the m parameterized signal arrival times y_k (in the performance vector Y) with the parameter space (X_l and X_g) by the following multivariate linear regression model [1]

$$Y \approx CX_l + DX_g + \varepsilon, \tag{2.52}$$

where ε is the model error. First order coefficient matrices C and D are defined as

$$C = [c_1, ..., c_k, ..., c_m]^T \in \mathbb{R}^{m \times n_l}, \tag{2.53}$$

$$D = [d_1, ..., d_k, ..., d_m]^T \in \mathbb{R}^{m \times n_g}, \tag{2.54}$$

To effectively reduce the dimension of X_l (the local process variables), a rank-deficient regression coefficient matrix $\tilde{C}$ can be used to approximate C without losing the capability of predicting Y accurately. If such a good rank-deficient matrix $\tilde{C}$ can be successfully found, it implies that as far as Y is concerned, the true dimensionality in X_l is low. Hence, building such a reduced-rank model with respect to X_l serves the need for discovering the redundancy in the full regression model and therefore fulfills performance specific parameter dimension reduction.

The construction of a rank reduced linear regression model can be described as follows. We denote the covariance of Y and X_l by Σ_{YX_l} and the covariance of X_l by $\Sigma_{X_lX_l}$. We also assume no correlation between the global and local variables ($\Sigma_{X_lX_g} = 0$). An optimal reduced rank regression model with two regressors can be shown to be [28]:

[1] In our work, linear RRR is only used to do parameter reduction, however, the SSTA is still conducted under a quadratic framework.

Theorem 3 *An $m \times r$ matrix A_r and $r \times n_l$ matrix B_r can be found to minimize the trace [28]:*

$$tr\{E[(Y - A_r B_r X_l - DX_g)(Y - A_r B_r X_l - DX_g)^T]\}, \tag{2.55}$$

where

$$A_r = U_r, B_r = U_r^T \Sigma_{YX_l} \Sigma_{X_l X_l}^{-1}. \tag{2.56}$$

In the above equations, $U_r = [u_1, ..., u_r]$ contains r normalized eigenvectors corresponding to the r largest eigenvalues of the matrix

$$Q = \Sigma_{YX_l} \Sigma_{X_l X_l}^{-1} \Sigma_{X_l Y}. \tag{2.57}$$

The above model is optimal in the sense of minimization of the variance of the reduced rank regression model errors.

As indicated by above theory, if we are given a matrix C in (2.53), which is the first order sensitivity matrix for the local variables in X_l with respect to the performance vector Y, then we have $\Sigma_{YX_l} \approx C\Sigma_{X_l X_l}$, which further gives:

$$Q \approx C\Sigma_{X_l X_l} C^T. \tag{2.58}$$

2) Implication On Parameter Dimension Reduction

The inherent redundancy in the predictor variables X_l can be filtered out statistically by the above procedure. Suppose a rank-r regression model with two regressors is computed through the above procedure that minimizes the statistical errors in Y:

$$Y \approx A_r B_r X_l + DX_g + \tilde{\varepsilon}, \tag{2.59}$$

where $\tilde{\varepsilon}$ represents the model error. We can construct a new set of variable $Z \in \mathbb{R}^r$

$(r < n_l)$ as

$$Z = B_r X_l. \tag{2.60}$$

Consequently, the quadratic timing model in (2.51) can be rewritten using the reduced parameter set Z instead of X_l as:

$$y_k \approx \bar{y}_k + \underbrace{d_k^T X_g + X_g^T A_k X_g}_{global} + \underbrace{\tilde{c}_k^T Z + Z^T \tilde{B}_k Z}_{local}, \tag{2.61}$$

A reduced rank model such as (2.61) can be used to reveal the redundancy in the predictor variable X_l (e.g. local process variations).

3) Composition of The Reduced Parameter Set

Once the original set of parameters in X_l is reduced into new variables in Z using the mapping matrix B_r (2.60), we can reveal the importance of each old parameter (in X_l) with respect to the performance (in Y) in a statistical sense, by examining the weighing coefficients. For example, the (i, j) entry of matrix B_r describes the linear contribution of the j^{th} original parameter x_j to the i^{th} new parameter z_i.

This exciting feature of the parameter reduction method is more preferable when compared with the traditional parameter screening technique [2]. For the example shown in Fig. 26, we know that the coefficients of the parameterized signal arrival times ($AT_{4,7}$ and $AT_{5,7}$) actually provides the sensitivities of the underlying parameters (X_1 to X_4). Consequently, for a specific signal arrival time $AT_{4,7}$ ($AT_{5,7}$), people can easily reduce the parameter dimension by removing those parameters X_2 and X_3 (X_1 and X_3) with insignificant coefficients. However, once we need to consider $AT_{4,7}$ and $AT_{5,7}$ simultaneously (in *max* operations), the parameter screening becomes difficult to apply.

4) Mapping Z back to X

To map the reduced parameter set (Z) back to the original parameter set (X_l), we

can use the pseudo-inverse (Moore-Penrose) [29] of matrix B_r that satisfies

$$X = T_r Z. \tag{2.62}$$

This mapping is done by computing the singular value decomposition (SVD) of B_r matrix. So if the SVD of B_r matrix is $B_r = U\Sigma V^T$, then the pseudo-inverse is $T_r = V\Sigma^{-1}U^T$.

This inverse mapping is necessary for converting the quadratic timing models (2.51) in the full parameter space to an alternative model in the reduced parameter space. For instance, (2.51) can be rewritten as

$$y_k \approx \bar{y}_k + \underbrace{d_k^T X_g + X_g^T A_k X_g}_{global} + \underbrace{(c_k^T T_r)Z + Z^T (T_r^T B_k T_r)Z}_{local}. \tag{2.63}$$

From (2.61) and (2.63), we have

$$\tilde{c}_k = T_r^T c_k, \tilde{B}_k = T_r^T B_k T_r. \tag{2.64}$$

Algorithm 4 Linear RRR Algorithm in SSTA

Input: First order sensitivity matrix $\mathbf{C} \in \mathbb{R}^{\mathbf{m}\times\mathbf{n}}$, parameter dimension $\mathbf{n_l}$ (of $\mathbf{X_l}$), the parameter covariance matrix $\mathbf{\Sigma_{X_l X_l}}$ and the reduced parameter dimension $\mathbf{r}$.
Output: The mapping matrix $\mathbf{B_r}$.

1: Set $\mathbf{Q} \leftarrow \mathbf{C\Sigma_{xx}C^T}$;
2: Do eigen-decomposition for $\mathbf{Q}$ matrix such that $\mathbf{Q} = U\Lambda U^T$ to get all the eigenvalues $\lambda_\mathbf{i}$ (in descending order) and the corresponding eigenvectors $\mathbf{u_i}$;
3: Use the $\mathbf{r}$ largest eigenvalues and their corresponding eigenvectors to form a diagonal matrix $\mathbf{\Lambda_r}$ and a matrix $\mathbf{U_r}$;
4: Set $\mathbf{B_r} \leftarrow \mathbf{\Lambda_r^{-1/2} U_r^T C}$;
5: Return the mapping matrix $\mathbf{B_r}$.

5) The Algorithm Details

We conclude the parameter reduction algorithm using linear RRR in Algorithm 4.

The inputs to the algorithm include the C matrix in (2.53) and the covariance matrix $\Sigma_{X_l X_l}$ of the full parameter space X_l. The input may also include the desired dimension of the reduced parameter set r.

By following the formulas given in the previous sections, Step 1 of the algorithm computes the Q matrix in (2.57) using a closed form formula. If all the variables in X_l are independent Gaussian with N(0,1) distributions, the covariance matrix $\Sigma_{X_l X_l}$ is simply the identity matrix, which leads to $Q = CC^T$. The eigen-decomposition is subsequently performed to find all the eigenvalues and the eigenvectors of the Q matrix. The r largest eigenvalues and their eigenvectors are used to form Λ_r and U_r. In Step 4 of the algorithm, the $\Lambda_r^{-1/2}$ matrix scales the rows of B_r matrix such that the final reduced parameters ($Z = B_r X$) all have the N(0,1) distributions.

c. Moment-Based Dimension Reduction (MOM)

In the above section, we show that the linear RRR can be used to detect the real parameter dimension by only a few extra computations. However, linear RRR is built upon a linear regression framework, thus it is possible that the above dimension reduction may cause relatively large errors when encountering stronger nonlinearities (e.g. the quadratic timing models or the signal arrival times that have relatively large second-order coefficients). Consequently, we introduce the moment-based dimension reduction method in the following subsections to remedy the insufficiency of the linear RRR method.

1) K-th Moment Dimension Reduction

To capture more challenging nonlinear effects of process variations in the performance space, we apply a recently developed moment-based (MOM) dimension reduction technique [58, 59]. In the following, we use Y to represent the performance space vector and the standardized predictor vector (including local process variations) by

X. The centered kth conditional moment $M^{(k)}(Y|X)$ is defined as:

$$\begin{aligned} M^{(k)}(Y|X) = E\left[(Y - E(Y|X)) \otimes \right. \\ \left. \cdots \otimes (Y - E(Y|X)) \otimes (Y - E(Y|X))^T |X\right] \end{aligned}, \tag{2.65}$$

where $\otimes$ indicates the Kronecker product that appears $k-1$ times in the above definition. Considering conditional moments, the moment-based method aims to find an $r \times n$ matrix B_r, $r < n$, such that the random vector BrX contains all the information about Y which is available from $M^{(1)}(Y|X), M^{(2)}(Y|X), ..., M^{(k)}(Y|X)$. If we use the notation U⊥V|Z to represent that the vectors U and V are independent given any value for the random vector Z, then the following definition for the *central k-th moment dimension reduction subspace (DRS)* is given by:

$$\mathrm{If:}\ \ \mathrm{Y} \perp \left\{\mathrm{M}^{(1)}(\mathrm{Y|X}), \ldots, \mathrm{M}^{(\mathrm{k})}(\mathrm{Y|X})\right\} \big| \, \mathrm{B_r X}, \tag{2.66}$$

then the subspace spanned by the columns of B_r is called a kth moment DRS for the regression of Y on X, which can be denoted by $Sub^{(k)}_{Y|X}$. The central subspace is designed to capture the entire conditional distribution of $Y|X$ and provide an overall picture of the dependence of Y on X, while the central kth moment subspace (CKMS) $Sub^{(k)}_{Y|X}$ is defined to be the intersection over all kth moment DRS. If such a subspace exists, it is the smallest kth moment DRS. The existence of CKMS can be guaranteed under various mild conditions [59]. For most data sets, existence is not a practical issue, then the following relations hold:

$$\mathrm{Sub}^{(1)}_{\mathrm{Y|X}} \subseteq \cdots \subseteq \mathrm{Sub}^{(\mathrm{k})}_{\mathrm{Y|X}}. \tag{2.67}$$

Since the focus in multivariate regression is on the first two moments of the conditional distribution of the response given the predictor, the central subspace $Sub^{(2)}_{Y|X}$ is of particular interest. To obtain the subspace $Sub^{(2)}_{Y|X}$, if we denote by X_s the

standardized X vector, two matrices, $K21c$ and $K22c$ can be obtained:

$$\begin{aligned} K21c &= \left[\; E\left(X_s Y^T\right), \quad E\left(X_s Y^T \otimes Y^T\right) \; \right], \\ K22c &= E\left[X_s X_s^T \otimes [Y^T - E(Y^T)]\right]. \end{aligned} \tag{2.68}$$

It has been shown in [58, 59] that the subspace $Sub(K21c)$ spanned by $K21c$, and the subspace $Sub(K22c)$ spanned by $K22c$ satisfies:

$$Sub(K21c) \subseteq Sub_{Y|X}^{(2)}, \; Sub(K22c) \subseteq Sub_{Y|X}^{(1)}. \tag{2.69}$$

It may be of interest to note that the matrix $K22c$ is the extended version of the *Principal Hessian Direction (PHD)* [61, 62] to multivariate response regression. It has been suggested that in practical applications, $K21c$ can better detect the linear trends or odd functions (cross terms), while $K22c$ is better at revealing symmetric trends or even functions (second-order self terms). In this work, we only use $K21c$ for dimension reduction and neglect $K22c$ to simplify the computation.

2) The Algorithm

From the above theory, it is easy to realize that the moment-based dimension reduction is based on estimating the moments of functions of the response (performance) and predictors. Unfortunately, the original theoretical work in [58, 59] does not assume a known model (mappings between the responses and the predictors) and rely on the sample-based estimators to evaluate $K21c$, which is rather expensive for practical circuit applications.

We have derived the closed-form formulas to compute the central 2nd moment dimension reduction subspace (DRS) based upon a given quadratic response model [2]. That is, if a quadratic model relating circuit performances with the process variables

[2]Higher order DRS with more complex forms can be derived in similar ways.

is given, parameter reduction can be achieved rather efficiently without performing sampling. The detailed algorithm is depicted in Algorithm 5, where we observe that M_1 reflects the information of linear models (find the same DRS as the linear RRR approach) while M_2 takes the second-order information into consideration.

Algorithm 5 K-moment dimension reduction Algorithm

Input: Quadratic forms of the standardized response vector $\mathbf{Y} = [\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_m}]^{\mathbf{T}} \in \mathbb{R}^{\mathbf{m}}$ in terms of the standardized predictor vector (uncorrelated Gaussian) $\mathbf{X} \in \mathbb{R}^{\mathbf{n}}$: $\mathbf{y_i} = \mathbf{X^T B_i X} + \mathbf{c_i^T X} + \mathbf{f_i}$ for $\mathbf{i = 1 : m}$, where $\mathbf{B_i} = [\mathbf{b_{p,q}}]_{\mathbf{n \times n}} \in \mathbb{R}^{\mathbf{n \times n}}$ and $\mathbf{c_i} \in \mathbb{R}^{\mathbf{n}}$;
Output: The dimension reduction matrix $\mathbf{B_r}$.

1: Set: $\mathbf{M_1} = \mathbf{E}\left(\mathbf{XY^T}\right) \leftarrow \left[\begin{array}{cccc} c_1 & c_2 & \cdots & c_m \end{array} \right]$;
2: **for** $\mathbf{i, j = 1 : m}$ **do**
3: Set: $\kappa_{\mathbf{i,j}} \leftarrow \mathbf{B_i c_j + B_j c_i}$;
4: **end for**
5: Set: $\mathbf{M_2} = \mathbf{E}\left(\mathbf{XY^T} \otimes \mathbf{Y^T}\right) \leftarrow \left[\begin{array}{cccc} \kappa_{1,1} & \cdots & \kappa_{m,m} \end{array} \right]$;
6: Set: $\mathbf{K21c} \leftarrow \left[\begin{array}{cc} M_1 & M_2 \end{array} \right]$.
7: Set: $\mathbf{B_r = Ur}$ which include $\mathbf{r}$ left singular vectors $\mathbf{u_1}, ..., \mathbf{u_r}$ of matrix $\mathbf{K21c}$ that corresponds to the $\mathbf{r}$ largest singular values;
8: Return the transform matrix $\mathbf{B_r}$.

3) Combination With Linear RRR

Obviously, the moment-based (MOM) parameter reduction is more computationally expensive when compared with the linear RRR approach, but can provide a better parameter reduction for stronger nonlinear effects. It can be shown later that using this parameter reduction may lead to extra reduced parameters to remedy the insufficiency of the linear RRR method.

In practical application, we suggest to combine this method with the simple linear RRR method to achieve the best accuracy as well as the efficiency. Such combinations can be performed by looking into the relative magnitude (say ϵ) of the second-order coefficient with respect to the corresponding linear coefficients: when the magnitude of ϵ exceeds a predefined threshold, a more accurate moment-based dimension reduction can be applied.

4) Implementation Issues

In realistic digital designs, there may be a lot of signal arrival times to be considered for parameter reduction at the same time, which makes the computation of M_2 matrix too computational expensive, since many cross effects of the outputs need to be evaluated (e.g. $B_i c_j$). Fortunately, we found in our experiments that neglecting the cross-effect evaluations will not impact the accuracy of parameter reduction much. So in practical implementations, we only need to compute the $B_i c_i$ terms for the M_2 matrix in Algorithm 5, which greatly improves the algorithm efficiency.

Not surprisingly, by adopting the above simplification, the moment-based parameter dimension reduction generates at most $2m$ reduced parameters given m observations. On the other hand, the linear RRR based parameter reduction only gives at most m reduced parameters for the m observations.

In this work, in order to reduce the local process variations (X_l) in the SSTA flow, we consider the the quadratic signal arrival times contributed by the local process variations as the performance vector (Y) and the local variables as the predictor variables. The moment-based parameter reduction can be applied to reduce the dimension of the local variables following the flow in Algorithm 5.

d. Comparisons of RRR and Moment-Based Parameter Reductions

In this section, we compare the two parameter reduction techniques, RRR and MOM based parameter reductions, on the $ISCAS85$ $C17$ implemented in TSMC 180nm technology. We use $VDD = 1V$ to test the two parameter reductions for the nonlinear performance space (circuit delays at two output nodes). Each transistor's threshold voltage (V_{th}) is considered as an independent Gaussian/uniform variable. We run 500 spice level Monte Carlo simulations in the full parameter space that includes 24 variables and compare the circuit delay of each simulation with the results obtained

by the two parameter reduction methods.

Since there are only two output nodes in this circuits, using the RRR approach can only reduce the parameters into two, while the moment-based parameter reduction can give at most a four-parameter model (we only consider the $B_i c_i$ for M_2 matrix in Algorithm 5). In the following experiments, we use 300 samples to fit two quadratic timing models for the two outputs, and then apply the moment-based dimension reduction to obtain three reduced parameters.

1) Accuracy with Gaussian Variables

We assume a 30% 3σ variation for each V_{th} variation. The experiment results for Gaussian variables are plotted in Fig. 27, showing the PDFs of the relative delay errors of the parameter reduced models obtained by the RRR and moment-based algorithms. As expected, with one additional reduced parameter, moment-based parameter reduction technique provides more accurate results compared with the RRR based reduction method. As observed, the three-parameter model (by the moment-based algorithm) is more accurate than the two-parameter model (by the RRR based algorithm). In fact, even we keep the same number of reduced parameters, the parameter reduction accuracy of the moment-based method is always better than the accuracy of the linear RRR based method.

The composition of the first reduced parameter set (the mapping coefficients in B_r) obtained by each method is plotted in Fig. 28. Obviously, there are differences between the reduced parameter sets (DRS) found by two dimension reduction methods.

2) Accuracy with Non-Gaussian Variables

The RRR and MOM methods are derived to achieve the optimal parameter reduction accuracy for Gaussian variables. However, the RRR/MOM dimension reduction results for non-Gaussian variables may be less accurate, as demonstrated in the fol-

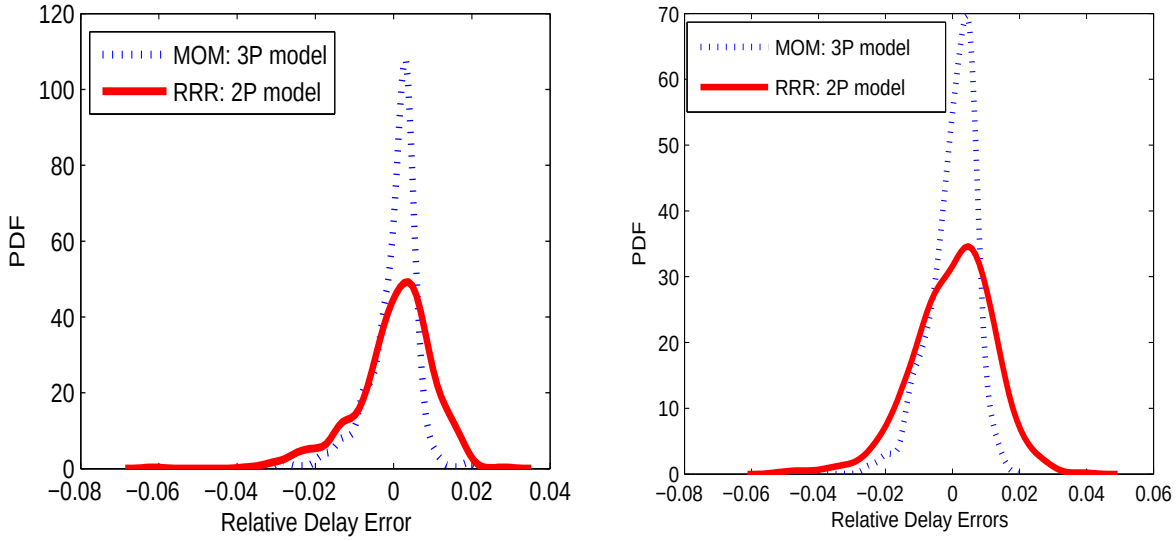

Fig. 27. PDF plots of the relative circuit delay errors by the RRR and moment-based dimension reductions (two output nodes) with independent Gaussian variables.

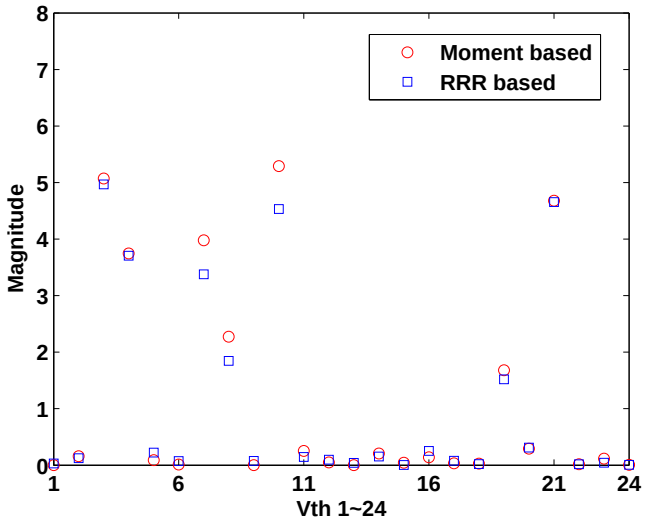

Fig. 28. Compositions of the first reduced parameters (Gaus. Dis.).

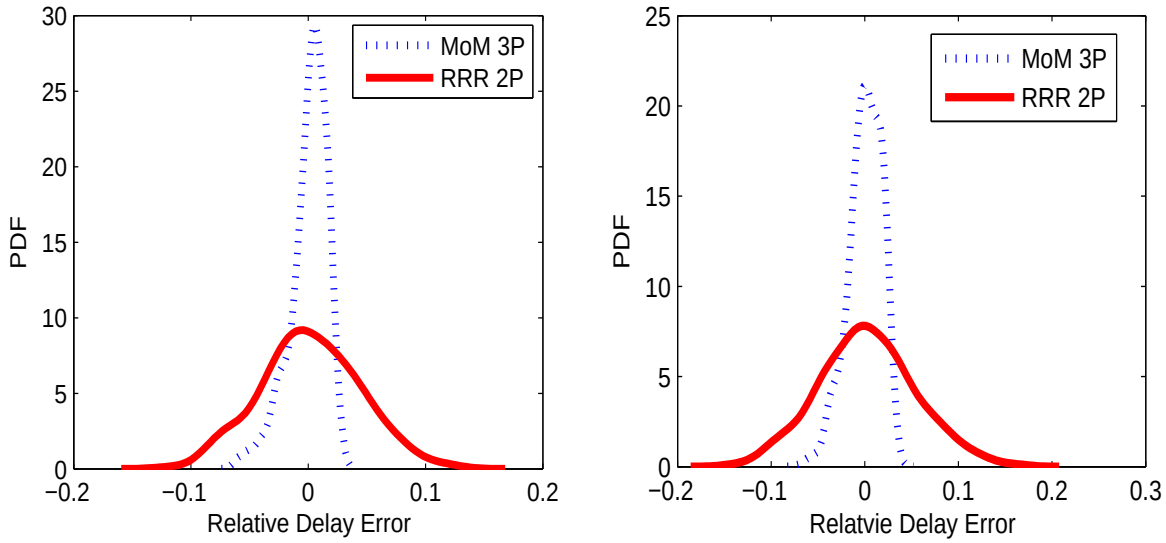

Fig. 29. PDF plots of the relative circuit delay errors by the RRR and moment-based dimension reductions (two output nodes) with independent uniform distributions.

lowing experiments. We consider all the transistor V_{th} variations with uniform distributions and set the variation range from -30% to 30%. After applying the same formulas (Algorithm 4 and 5) and repeating the experiments described in Section d, the results obtained by RRR and MOM are shown in Fig 29. As observed, the relative errors of RRR and MOM with uniform distributions for the two circuit outputs are significantly larger than the errors in the Gaussian case (Section d). Fortunately, we have one possible workaround for handling the non-Gaussian variables that is to transform them to Gaussian ones first and then apply the RRR/MOM formulas. We also want to point out that in many cases the reason that certain process parameters not being Gaussian is that they are not the "root" parameters. For example, resistance and capacitance are not Gaussian, but the gate length, wire width and thickness are Gaussian. So if we are given non-Gaussian parameters, we may convert them to Gaussian by finding the "root" from them.

The composition of the first reduced parameter set obtained by each method is also plotted in Fig. 30, which is quite similar to the composition shown in Fig. 28.

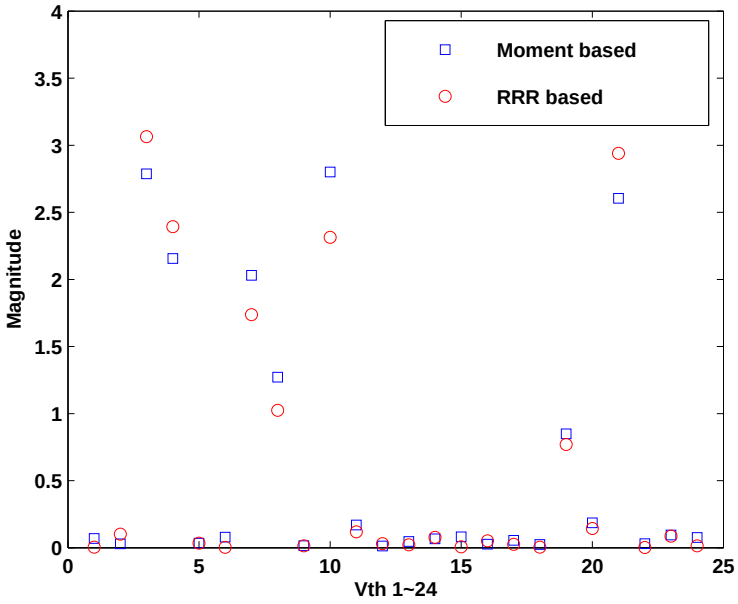

Fig. 30. Compositions of the first reduced parameters (Unif. Dis.).

3. SSTA with Parameter Reduction

In this section, we first introduce the fundamental ideas of embedding parameter reduction within the quadratic SSTA procedure. Next, we describe how to partition the circuit according to specific SSTA and parameter reduction algorithms. The propagation of the reduced parameters in each step of the quadratic SSTA is also demonstrated to provide a complete flow of the algorithm.

a. Overview

When propagating the signal arrival times by following the SSTA algorithm described in Section b, an increasing number of local process variations will need to be con-

sidered. To avoid the explosion of the number of parameters that enter the SSTA, parameter reduction is intervened with SSTA processing steps to compress the local variations. In the following part, we show how to apply the RRR and moment-based parameter reductions in the second-order SSTA algorithm. Suppose there are m signal arrival times to be considered for parameter reductions at the same time.

1) RRR in SSTA

The RRR based parameter reduction involves the computation of the Q matrix (2.57) that requires the covariance matrices relating the response variables and the underlying parameters be computed in advance. Fortunately, the arrival time of each pin is given in the quadratic forms in SSTA. So we can obtain the C matrix in (2.53) easily and follow Algorithm 4 to perform the eigen-decomposition of the Q, whose eigenvectors with the few largest eigenvalues form the B_r matrix (2.60). Subsequently, we can transform the the original local variables into a smaller dimensional parameter set.

2) MOM in SSTA

For the MOM based parameter reduction, the M_1 matrix in Algorithm 5 is same as the C matrix (2.53). To compute M_2, we need to perform the matrix-vector multiplications, say $B_i c_i$, for all the signal arrival times within this circuit partition (we only consider the $B_i c_i$ term for M_2 matrix). So m signal arrival times need additional m times matrix-vector multiplications when compared with the RRR method. Then the B_r matrix can be obtained by computing the SVD of matrix $K21c$.

b. Circuit Partitioning

Assume the global variations are transformed to the uncorrelated variables by the Principle Component Analysis (PCA) before the SSTA algorithm starts. Subsequently, we need to partition the circuit for the parameter reduction step. To apply

the parameter reduction algorithms in SSTA, we first partition the circuits into some building blocks and then do further partitions within each block. Parameter reductions are performed within these small partitions to obtain the reduced parameters, and subsequently, these new parameters can be further reduced by parameter reductions within the larger building blocks. For example, we can partition the circuit by logic levels and within each level, some additional partitions may be performed if necessary.

During the partitioning, we also need to consider the partition size as well as its corresponding local parameter dimension. In general cases, using larger blocks usually results in more local parameters for each block, thus requires more computational cost for the *sum* and *max* operations in the SSTA flow. Therefore, the partition strategy largely depends on the efficiency of the atomic operations as well as the parameter reduction procedures. To understand the impact of circuit partitioning on SSTA using parameter reduction, we consider a circuit with n local variables that is partitioned into k blocks (each block has a similar number of local variables, say $n_l = ceil(n/k)$). We introduce the parameter reduction ratio PRR, which is defined by:

$$PRR = n_r/n_l, \tag{2.70}$$

where n_r represents the number of reduced parameters and n_l denotes the number of the original local variables. Assume that there are totally c (constant number) *max* operations to be performed and the parameter reduction within each circuit block has the same PRR [3], then the total computational cost of all the *max* operations and

[3]This assumption is to ease the computational cost analysis and in reality, this ratio is not exactly the same for each circuit block.

parameter reductions can be approximated as:

$$Cost_{tot} \approx c * Cost_{max}(n_g + n_l * PRR) + k * Cost_{PR}(n_l), \tag{2.71}$$

where $Cost_{tot}$ is the total cost for all the max operations and parameter reductions, $Cost_{max}$ is the cost for each max operation and $Cost_{PR}$ is the cost for each parameter reduction.

We give a very brief description of the circuit partitioning strategy as follows. Considering n parameters, assume the computational cost for each max operation is $O(n^q)$, and for each parameter reduction is $O(n^s)$. If $q >> s$, which is a very typical case especially when using very accurate max operations (e.g. the algorithm in [55]), we should consider very small partition size to make sure the max operations can be performed efficiently on a small set of variables. Consequently, the circuit partition should be performed according to the efficiencies of the atomic operations as well as the parameter reductions for specific SSTA and parameter reduction algorithms.

c. Propagation of Reduced Parameters

After the circuit partitioning step, parameter reduction is conducted hierarchically.

Assume we have computed all the m signal arrival times of partition i by the second-order SSTA, then one of these signal arrival times (say AT_k) is given in the form of (2.51) as:

$$AT_k = \bar{M}_k + \underbrace{d_k^T X_g + X_g^T A_k X_g}_{global} + \underbrace{c_k^T X_{l_i} + X_{l_i}^T B_k X_{l_i}}_{local} \tag{2.72}$$

where the local parameter set $X_{l_i} = \begin{bmatrix} X_{loc\_i} \\ Z_{i-1} \end{bmatrix} \in \mathbb{R}^{n_l}$ includes the local parameters of partition i and the reduced parameter set Z_{i-1}(obtained from the proceeding partition

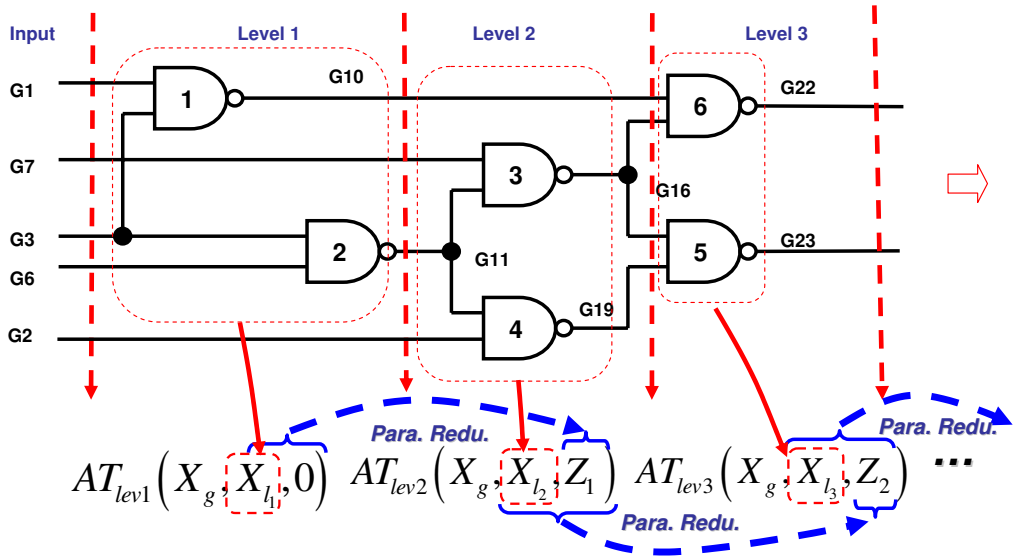

Fig. 31. Parameter propagations in parameter reduced SSTA.

$i-1$ that is driving partition i). In the above way, the reduced parameters can be propagated from the inputs to the outputs. We conclude the flow of Parameter-Reduced SSTA (PR-SSTA) in Algorithm 6.

Algorithm 6 Quadratic SSTA with parameter dimension reduction

1: Partition a large design into a set of building blocks;
2: Do further partitions within each building block if necessary;
3: For partition **i**, reduce the local variables of partition **i** and the reduced parameters from the proceeding partition (that is driving partition **i**) into a few reduced parameters using RRR or moment-based algorithms according to the nonlinearity of the timing model;
4: Perform *max* and *sum* operations for the signal arrival times which are in terms of the global parameters $\mathbf{X_g}$ and the reduced parameters $\mathbf{Z_i}$;
5: Propagate the reduced parameters to its following partitions;
6: **i**++; Go to 3 until all the signal arrival times are computed.

d. Examples of SSTA Using Parameter Reduction

1) A Circuit Example Partitioned By Its Logic Levels

We demonstrate the analysis flow in Fig. 31 for $ISCAS85$ $C17$ circuit where we

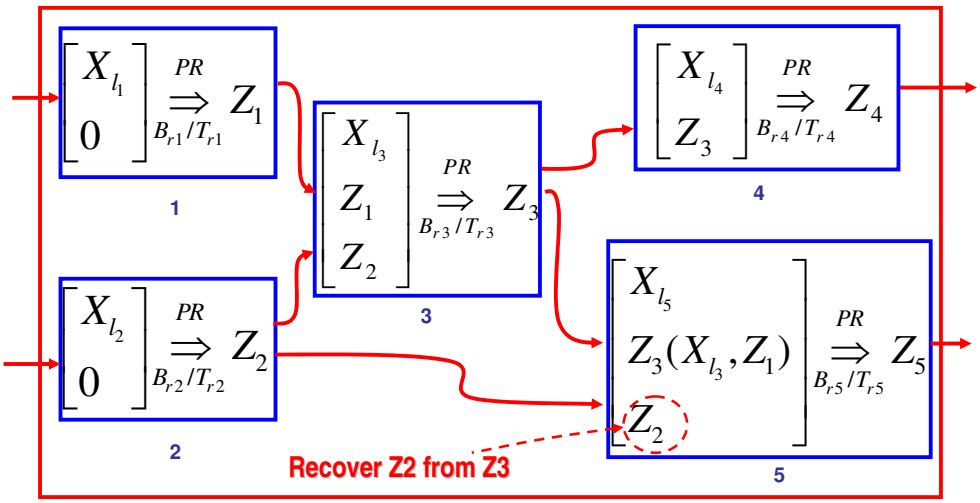

Fig. 32. Parameter propagations in SSTA with reconvergent variables.

assume the circuit is partitioned by its logic levels. In the figure, we denote one of the arrival times, the global and local variables in level i by $AT_{lev,i}$, X_g and X_{l_i} respectively.

If we denote by Z_{i-1} the reduced parameter set obtained in the proceeding logic level $i-1$, the signal arrival time $AT_{lev,i}$ can be written in the quadratic forms of the global parameters (X_g), the local parameters (X_{l_i}) and the reduced parameters from the proceeding logic level (Z_{i-1}). After parameter reduction is performed for partition i to reduce X_{l_i} and Z_{i-1} into a new reduced parameter set Z_i, the *max* and *sum* operations of the signal arrival times in terms of X_g and Z_i are performed, which is much more efficient. As this algorithm continues, the local parameters that enter the SSTA can always be compressed to a few reduced parameters before they are propagated continuously through the whole circuit.

If more partitions need to be considered within each logic level, similar procedures can be followed: parameter reduction is performed for each partition of the level, and then the reduced parameters are propagated to the following partitions, where they

are combined with the local variables that can be further reduced through parameter reductions.

2) A More General Circuit Example

When reconvergent variables exist during SSTA, parameter recovery in Section b is required which maps the reduced parameter set back to the full parameter set. The parameter propagation considering reconvergent variables is shown in Fig. 32, where five circuit partitions are considered. In partition i, the local parameter set X_{l_i} (in partition i) combined with the reduced parameter set Z_{i-1} that feeds into this partition are reduced into a new reduced parameter set Z_i. B_{r_i} matrix maps the original parameter set to the reduced parameter set while T_{r_i} does the inverse mapping. Consequently, to perform parameter reduction in partition 5, the reduced parameter set Z_2 from partition 3 needs to be recovered before using the parameter reduction methods.

e. Computation Complexity

The computational complexity of the parameter reduction algorithm for each partition is determined by the number of signal arrival times of the partition and the number of local parameters (combined with the reduced parameters from the proceeding partitions) to be compressed. For RRR (MOM) based parameter reduction, only the largest few eigenvalues (singular values) and their corresponding eigenvectors (singular vectors) of matrix Q in Algorithm 4 (K_{21c} matrix in Algorithm 5) are needed. The main complexity of the SSTA algorithm is dominated by other parts of the second-order SSTA algorithm (e.g. *max* operations), instead of the parameter reduction procedure. The additional cost due to the parameter reduction algorithm can be almost neglected (as shown in the following experiment section), especially for the quadratic statistic timing analysis where *max* operations contribute most to the

computational costs.

4. Experimental Results

We have implemented our second-order SSTA algorithm by combining the hierarchical parameter dimension reduction algorithm (using RRR or MOM) with the quadratic SSTA algorithm proposed in [55]. Our algorithm is tested on the $ISCAS85$ benchmark circuits. To model the correlation among the intra-die variations, we partition the circuit into several grids (number of grids depends on the circuit size). The correlated global variables are transformed to uncorrelated ones by PCA. All the quadratic timing models are subsequently obtained based on the uncorrelated variables.

We compare our results with the following two types of experiments: (a) $100K$ Monte Carlo simulations with a full set of inter-die and intra-die variations; (b) $100K$ Monte Carlo simulations with only the global variations. The sample size for Monte Carlo simulations is selected to be $100K$ to ensure the accuracy of Monte-Carlo simulations for the high (20 or more) dimensional nonlinear circuit timing analysis problems considered in this work. Various experiments are shown to demonstrate the excellent performance of this new second-order parameter reduction based SSTA algorithm in terms of handling high-dimensional process variations.

a. Impact of Local Process Variations

We first show through three circuit examples ($ISCAS85$ benchmark circuits $C880$, $C1355$ and $C5315$), that neglecting the important local process variations in statistical static timing analysis can lead to significant errors, especially for large circuit designs. We also show that the proposed second-order SSTA with parameter dimension reduction can handle a large set of local variations efficiently and accurately, which is extremely difficult to achieve by existing techniques.

We partition each circuit based on gate locations such that all partitions share the same two global variation variables and are influenced by an individual local variation source. In these experiments, multiple parameter reductions using RRR are sequentially performed during the SSTA analysis as described in Section 3 (the moment-based parameter reduction have quite similar results due to the moderate nonlinearity of the quadratic timing model). At any point of time the effective number of process parameters being kept is controlled to be less than five. As such, via parameter dimension reduction, our new second-order SSTA is able to maintain a very favorable runtime efficiency. This can be well understood by the fact that the runtime complexity of any true quadratic SSTA will grow quickly with the number of parameters.

The circuit delay probability density distributions of the $ISCAS85$ benchmark circuits $C880$, $C1355$ and $C5315$ are compared with the two kinds of MC simulations under the setups (a) and (b). Counting all the parameters in the global and local variation set, the total number of parameters for the above three circuits are 14, 14 and 26 respectively. At the final step of our parameter-reduced SSTA, the complete parameterized circuit timing expressions only include two global variables and two reduced variables, achieving about $10X$ parameter dimension reduction for the local variation sources. Without any significant loss of accuracy, the statistical circuit delay distributions computed by our SSTA with parameter reduction follow closely with the MC simulations, as shown in Fig. 33, Fig. 34 and Fig. 35.

At a glance, the PDF distributions seem to be similar to the ones without considering the local variations, but when we look at the rightmost tails, the curve differences are obvious, which may indicate large differences on the final timing yields. Consequently, neglecting the local variation effects in the timing analysis may lead to quite large errors.

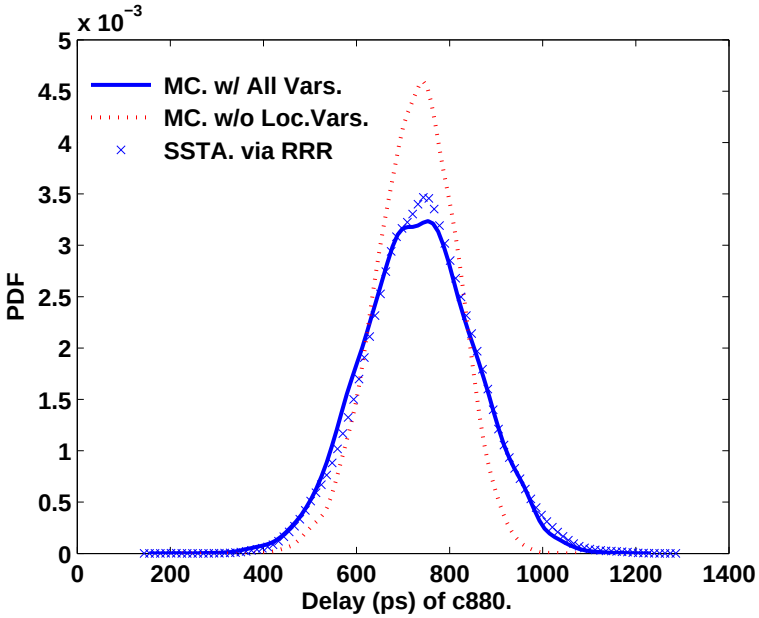

Fig. 33. PDF of ISCAS C880.

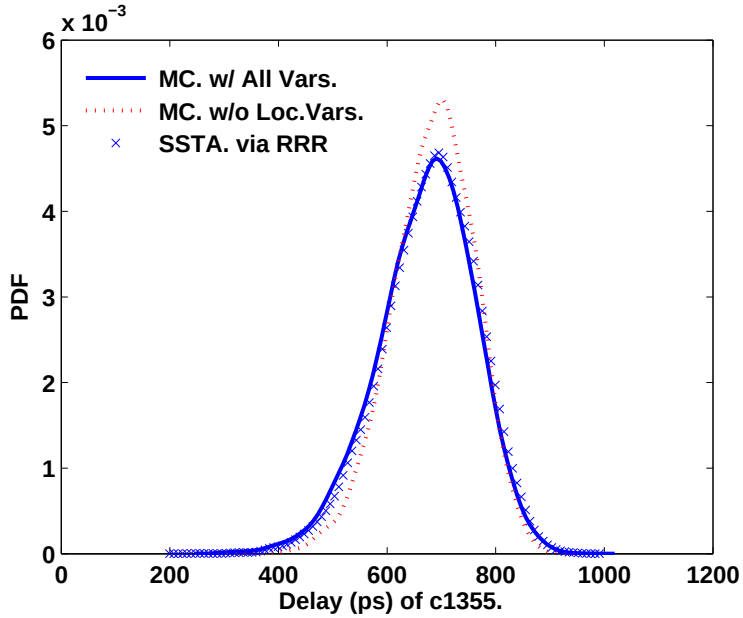

Fig. 34. PDF of ISCAS C1355.

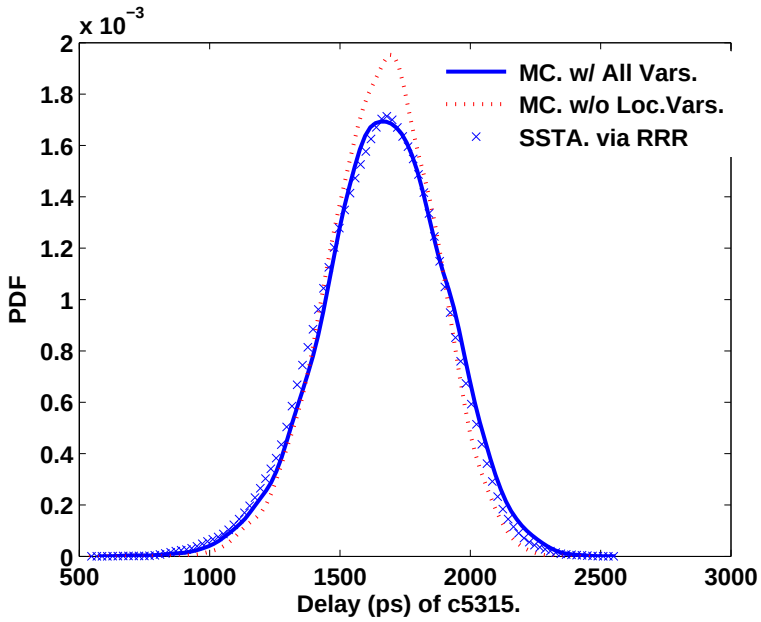

Fig. 35. PDF of ISCAS C5315.

b. Accuracy

In this section we follow similar experiment setup to present the results on several *ISCAS*85 benchmark circuits. We apply the RRR and MOM based parameter reductions for all circuits, respectively[4]. At each step of SSTA, we always keep propagating two reduced parameters (no extra reduced parameters are included in the moment-based approach). Table V shows detailed results of each parameter-reduced SSTA run, where *Total # of parameters* means the total number of the local and global variables considered in the circuit. *μ/σ by MC w/ all var. (ps)* represents the mean delay and the standard deviation given by the $100K$ Monte Carlo simulations considering all variation sources. *μ/σ error of RRR (MOM)-SSTA* represents the relative errors of the mean and standard deviation given by our quadratic SSTA with param-

[4]We have not combined the two approaches, but in real applications, such combinations may lead to better runtime efficiency.

Table V. SSTA results of ISCAS85 benchmark circuits

Circuit	$C7552$	$C6228$	$C5315$	$C3540$
# of Part.	21	12	24	23
Tot. # of para.	$21+2$	$12+2$	$24+2$	$23+2$
μ by MC w/ all var. (ps)	1328	3821	1670	1755
σ by MC w/ all var. (ps)	191.3	557.4	232.4	222.8
μ err. of RRR-SSTA	0.1%	0.89%	1.10%	1.14%
σ err. of RRR-SSTA	4.87%	0.65%	0.23%	0.37%
μ err. of MOM-SSTA	0.07%	0.56%	0.93%	1.05%
σ err. of MOM-SSTA	2.16%	0.41%	0.21%	0.28%
μ err. of MC w/o local var.	0.50%	0.61%	0.75%	1.15%
σ err. of MC w/o local var.	8.75%	2.4%	12.4%	7.8%
T. of MC (s)	582	455	360	243
T. of RRR (s)	0.02	< 0.01	0.02	0.03
T. of MOM (s)	0.05	0.03	0.05	0.08
T. of Quad. SSTA (s)	3.7	4.81	4.0	2.83
Sp. of PR-SSTA	$162X$	$95X$	$90X$	$86X$
Circuit	$C1908$	$C1355$	$C880$	$C499$
# of Part.	20	12	12	5
Tot. # of para.	$20+2$	$12+2$	$12+2$	$5+2$
μ by MC w/ all var. (ps)	1206	671.78	727.32	555.33
σ by MC w/ all var. (ps)	175.8	89.63	99.34	37.29
μ err. of RRR-SSTA	0.52%	1.07%	0.32%	0.56%
σ err. of RRR-SSTA	0.54%	2.07%	0.47%	1.77%
μ err. of MOM-SSTA	0.32%	0.96%	0.25%	0.38%
σ err. of MOM-SSTA	0.46%	1.19%	0.39%	1.03%
μ err. of MC w/o local var.	0.69%	1.89%	0.12%	0.60%
σ err. of MC w/o local var.	7.1%	15.5%	12.5%	17.0%
T. of MC (s)	101	82.5	47.4	29.5
T. of RRR (s)	< 0.01	< 0.01	0	0
T. of MOM (s)	0.02	0.03	< 0.01	< 0.01
T. of Quad. SSTA (s)	0.79	0.49	0.24	0.06
Sp. of PR-SSTA	$128X$	$168X$	$200X$	$492X$

eter dimension reduction while μ/σ *errors of MC w/o local var.* represents the MC simulation without the local variations. Not surprisingly, dropping the underlying local parameters produces quite large errors in the timing results. From Table V we can find that neglecting these local variation sources may introduce up to 1.89% mean difference (C1355) and 17% sigma difference (C499).

The runtimes for both the SSTA with parameter reduction and the MC simulations are listed. Since the runtime of the SSTA using the RRR and the MOM algorithms are quite similar, only one of them is listed to show the runtimes and speedups over Monte-Carlo simulations. The runtime of the parameter reduction procedures (RRR/MOM) is only a negligible portion of the total analysis runtime. It shall be emphasized that without effective parameter reductions, existing second-order SSTA algorithms cannot handle such large numbers of parameters due to their super-linear complexity. However, our new algorithms do not suffer from this limitation while maintaining good accuracy.

To demonstrate the impacts of the number of parameters on the runtime, our algorithm is also applied to the first few largest benchmark circuits in Table VI, but using larger circuit partitions. Since the number of parameters that feed our parameter reduced quadratic SSTA is smaller, the runtime for each circuit is reduced.

5. Summary

A hierarchical parameter reduction algorithm for quadratic Statistical Static Timing Analysis is proposed by adopting the RRR and MOM based methodologies. Using these approaches, we are able to accurately capture a large number of intra-die and inter-die variations during SSTA. This is achieved by intervening the RRR/MOM based parameter reductions with the SSTA processing steps such that the effective number of process parameters at any point of time during the SSTA analysis is well

Table VI. SSTA results of benchmark circuits with larger partitions

$Cicrcuit$	$C7552$	$C6228$	$C5315$	$C3540$
Partitions	6	4	8	11
Total # of parameters	$6+2$	$4+2$	$8+2$	$11+2$
μ by MC w/ all	1329	3823	1651	1723
σ by MC w/ all	181.7	551.8	221.6	210.8
μ err. of RRR-SSTA	0.2%	0.40%	1.64%	1.04%
σ err. of RRR-SSTA	2.24%	0.65%	1.29%	0.67%
μ err. of MOM-SSTA	0.11%	0.27%	0.89%	0.65%
σ err. of MOM-SSTA	1.32%	0.55%	0.98%	0.54%
μ err. of MC w/o	0.50%	0.81%	0.4%	0.5%
σ err. of MC w/o	4.0%	2.71%	8.1%	4.5%
T of MC (s)	544	415	362	245
T of RRR (s)	< 0.01	< 0.01	< 0.01	< 0.01
T of MOM (s)	< 0.01	< 0.01	< 0.01	< 0.01
T of Quad. SSTA (s)	2.2	2.83	2.14	1.1
Speedups of PR-SSTA	$247X$	$147X$	$169X$	$223X$

controlled. Our experimental results have demonstrated significantly improved run-time efficiency for accurate second-order SSTA analysis via these parameter dimension reduction techniques.

CHAPTER III

HARDWARE ACCELERATION OF LARGE SCALE ON-CHIP POWER GRID ANALYSIS

The challenging task of analyzing on-chip power (ground) distribution networks with multi-million node complexity and beyond is key to today's large chip designs. For the first time, this work exploit recent massively parallel single-instruction multiple-thread (SIMT) based graphics processing unit (GPU) platforms to tackle power grid analysis with promising performance. Several key enablers including GPU-specific algorithm design, circuit topology transformation, workload partitioning, performance tuning are embodied in the novel GPU-accelerated hybrid multigrid algorithm, GpuHMD, and its implementation. In particular, a proper interplay between algorithm design and SIMT architecture consideration is shown to be essential to achieve good runtime performance. Different from the standard CPU based CAD development, care must be taken to balance between computing and memory access, reduce random memory access patterns and simplify flow control to achieve efficiency on the GPU platform. Extensive experiments on industrial and synthetic benchmarks have shown that for DC power grid analysis using one GPU, the proposed GpuGMD engine can achieve 100X runtime speedup over a state-of-the-art direct solver and be more than 50X faster than the CPU based multigrid implementation. For transient analysis using one GPU, more than 20X speedups is achieved when GpuGMD is compared with the direct method. It is observed that the proposed approach scales favorably with the circuit complexity, at a rate about one second per two million nodes on single GPU card. We also show that utilizing a four-core-four-GPU system, a grid with eight million nodes can be solved within about one second.

A. Background and Overview

We first review the power grid analysis problems and the GPU architecture. Next, an overview of the proposed GpuHMD approach is provided.

1. On-Chip Power Grid Analysis

The power grid analysis covers two main aspects: DC and transient analysis. As for DC analysis, power grid problems are typically formulated into a linear system as [14, 20, 22, 15, 23]:

$$GV = I, \tag{3.1}$$

where G is a symmetric positive definite matrix representing the interconnected resistors, V is the vector including all the node voltages and I is a vector containing all the independent sources. Directly solving such a large system using LU or Cholesky matrix factorizations is typically very expensive and requires huge memory resources [18, 24]. Iterative methods [14, 20, 15] are memory efficient, but may suffer from slow convergence.

Specifically, the point relaxation methods update the node voltage using the neighboring nodes repeatedly until achieving the converged solution:

$$V_x = \sum_{i \neq x} \frac{g_i}{\sum g_i} V_i - \frac{I_x}{\sum g_i}, \tag{3.2}$$

where V_x is the node voltage to be updated and I_x is the current sources flowing out the node, while g_i and V_i are the neighboring conductance and voltages.

Transient analysis requires solving the system at multiple time points. The point relaxation based update can be written as:

$$V_x(t) = \sum_{i \neq x} \frac{g_i}{\sum g_i + \frac{C_x}{h}} V_i(t) - \frac{I_x}{\sum g_i + \frac{C_x}{h}} + \frac{\frac{C_x}{h}}{\sum g_i + \frac{C_x}{h}} V_x(t-h), \tag{3.3}$$

where C_x is the grounded capacitance associated with node V_x and h is the time step.

2. GPU Matrix Solvers?

A basic understanding of the SIMT GPU architecture is instrumental for evaluating the potential in applying GPU matrix solvers to large power grid problems. Consider a recent commodity GPU model, Nvidia G80 series. Each card has 16 streaming multiprocessors (SMs) with each SM containing eight streaming processors (SPs) running at 1.35GHz. An SP operates in single-instruction, multiple-thread (SIMT) fashion and has a 32-bit, single-precision floating-point, multiply-add arithmetic unit [63]. Additionally, an SM has 8192 registers which are dynamically shared by the threads running on it and can access global, shared, and constant memories. The bandwidth of the off-chip memory can be as high as 86GB/s, but the memory bandwidth may reduce significantly under many random memory accesses. The following programming guidelines play very important roles for efficient GPU computing [27]:

1. Low control flow overhead: execute the same computation on many data elements in parallel;

2. High SP floating point arithmetic intensity: perform as many as possible calculations per memory access;

3. Minimum random memory access: pack data for coalesced memory access.

Due to the very nature of the SIMT architecture, it remains as a challenge to implement efficient general-purpose sparse matrix solvers on GPU. In recent such attempts, it is reported that most of runtime is spent on data fetching and writing, but not on data processing [64, 65]. For instance, traditional iterative methods such as conjugate gradient and multigrid [64] involve many sparse matrix-vector computations, leading to rather complex control flows and a large number of random memory

accesses that can result in extremely inefficient GPU implementations. On the other hand, a problem with a structured data and memory access pattern can be processed by GPU rather efficiently. The performance of a dense matrix-matrix multiplication kernel on GPU can reach a performance of over 90 GFLOPS, which is orders of magnitude faster than on CPU [63]. Considering the above facts, it is unlikely to facilitate efficient power grid analysis by building around immature general-purpose GPU matrix solvers or implementing existing CPU-oriented power grid analysis methods [14, 15, 19] on GPU.

3. Our Approach

a. Power Grid Uniformity

To achieve the best analysis efficiency on SIMT platforms, understanding the physical properties of practical power grid designs is critical. It can be expected that if the power grid can be stored and processed like pixel graphics, the GPU SIMT platform can be of a significant advantage over the general purpose CPU platform. Not surprisingly, after examining a set of published industrial power grids [66, 67], we have found that real-life designs have a high degree of global uniformity while exhibiting some local irregularity, as shown in Fig. 36 where the top view of the grid node locations of all metal layers of an industrial power grid design is demonstrated with some of the nodes overlapping. Therefore, to maintain regularity on GPU, it is very natural for us to consider solving an approximate regular power grid that is close to the original grid. However, this brings up the need for developing "regular" numerical methods and correction schemes to guarantee solution accuracy.

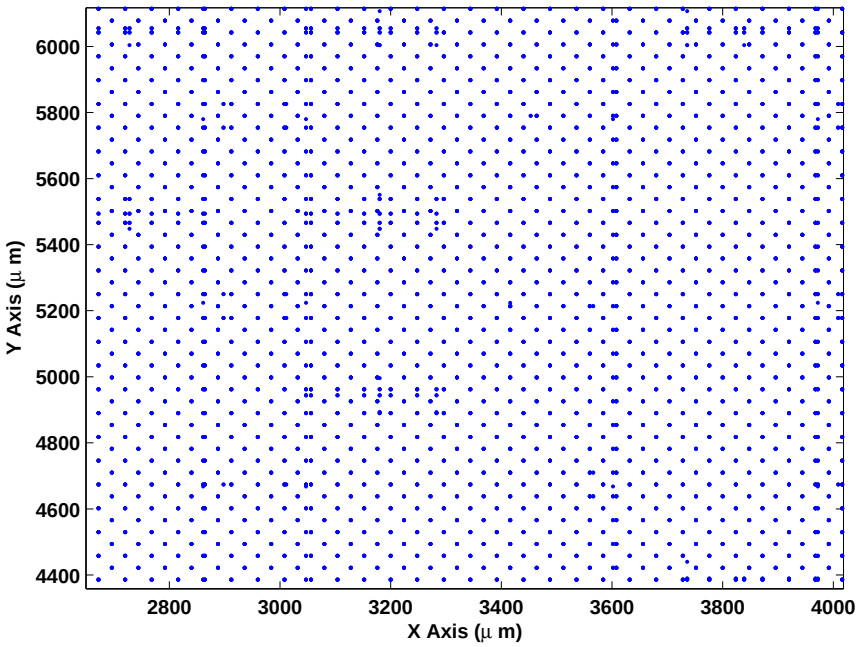

Fig. 36. Node distribution of an industrial power grid design.

b. GPU-Based Geometric Multigrid Method

Multigrid methods are among the fastest numerical algorithms for solving large PDE-like problems [68], where a hierarchy of exact to coarse replicas (e.g. fine vs. coarse grids) of the given linear problem are created. Via iterative updates, the high and low frequency components of the solution error are quickly damped on the fine and coarse grids, respectively, contributing to the efficiency of multigrid. When properly designed, multigrid methods can achieve a linear complexity in the number of unknowns. The hierarchical iterative nature of multigrid is attractive to GPU platforms since the GPU on-chip shared memory is rather limited. Multigrid methods typically fall into two categories, geometric multigrid (GMD) and algebraic multigrid (AMG). AMG may be considered as a robust black-box method and requires an expensive setup phase while GMD may be implemented more efficiently if specific ge-

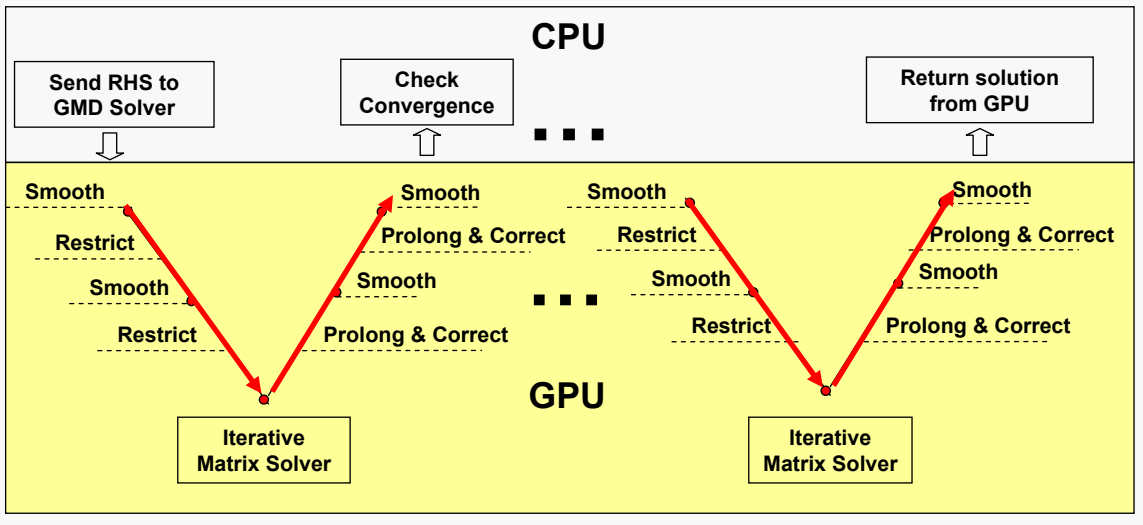

Fig. 37. Acceleration of GMD solve on GPU.

ometric structures of the problem can be exploited. The key operations of a multigrid method include:

1. *Smoothing:* point or block iterative methods (e.g. Gauss-Seidel) applied to damp the solution error on a grid;

2. *Restriction:* mapping from a fine grid to the next coarser grid (applied to map the fine grid residue to the coarse grid);

3. *Prolongation:* mapping from a coarse grid to the next finer grid (applied to map the coarse grid solution to the fine grid);

4. *Correction:* use the mapped coarse grid solution to correct the fine grid solution.

On the k-th level grid with an initial solution of v_k, a typical multigrid cycle $MG(k, v_k)$ has the following steps [68]:

1. Apply pre-smoothing to update the solution;

2. Compute the residue on the k-th grid and map it to the $k+1$-th coarser grid via restriction;

3. Using the mapped residue to solve the $k+1$-th grid directly if the coarsest level is reached, otherwise apply a multigrid cycle $MG(k+1, v_{k+1})$ with a zero initial guess $v_{k+1} = 0$;

4. Map the solution v_{k+1} to the k-th grid via prolongation, and correct the solution v_k by adding v_{k+1};

5. Apply post-smoothing to further improve v_k at the k-th level grid and return the final v_k.

A GPU-specific GMD method is developed in our approach. Starting from a regularized power grid, all the key components of multigrid are realized in a *geometrically regular* fashion across the entire multigrid hierarchy, leading to simple flow controls and highly regular memory access patterns, favoring the GPU implementation.

c. Hybrid Multigrid (HMD) Iterations

The approximate regular power grid is solved efficiently using our custom GMD method on GPU (Fig. 37), where no explicit sparse matrix-vector operations are needed. The work associated with the GMD constitutes the dominant workload of the entire GpuHMD approach. To guarantee the accuracy of the final power grid solution, we further apply HMD iterations between the GPU and host to remove any error that may arise from only solving the approximate regular grid. Denote the true (original) power grid by $Grid_O$ and the regularized grid by $Grid_R$, HMD iterations involve the following main steps (Fig. 38):

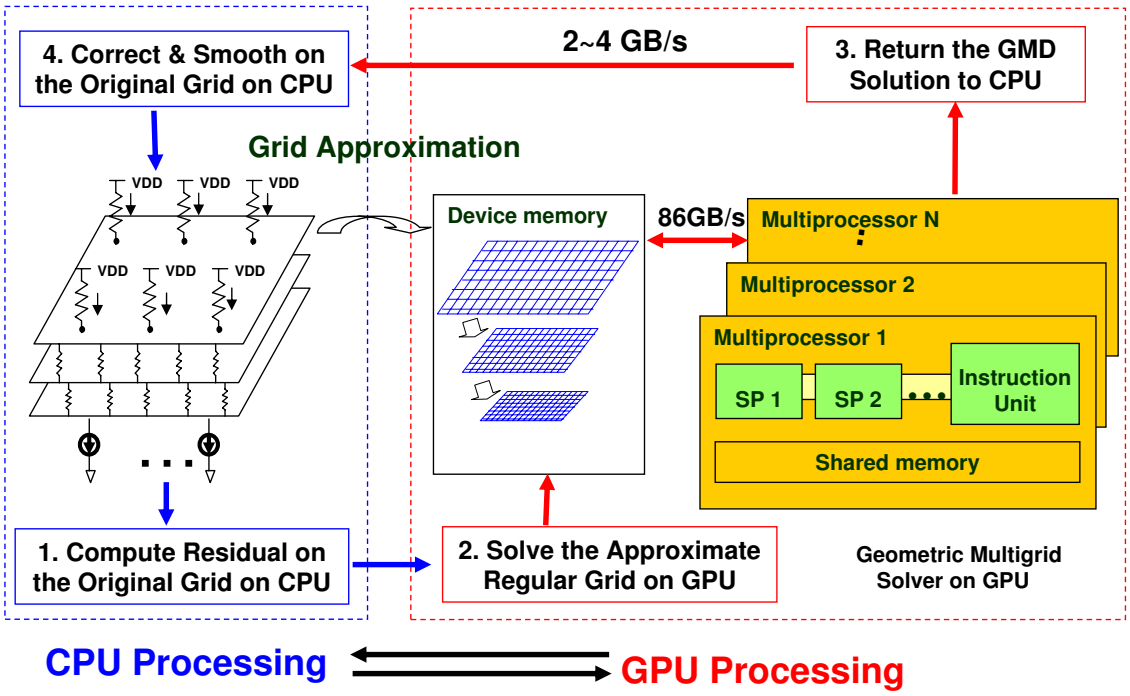

Fig. 38. Overall GpuHMD analysis flow.

1. (CPU:) Compute the residue of the current solution on $Grid_O$ and map the residue to $Grid_R$;

2. (GPU:) Solve the $Grid_R$ problem under the mapped residue using GMD and return the solution to $Grid_O$;

3. (CPU:) Update the $Grid_O$ solution using the GPU result and apply additional smoothing;

4. (CPU:) If the solution error is small enough, exit; otherwise repeat the above steps.

The bulk workload of the entire GpuHMD approach is done on GPU via solving the regular grid (step-2). Only a fraction of the work such as simple residue computation and smooth steps is preformed on the host, where the general-purpose CPU is more

efficient in terms of handling the original (irregular) power grid.

B. Regular Grid Approximation

We discuss several key issues in converting a three-dimensional irregular power grid to a two-dimensional regular approximation that can be processed efficiently on GPU.

1. Mapping to a Regular Grid

The goal is to map the original three dimensional irregular power grid to a two dimensional regular grid such that the electrical property of the original grid can be best preserved. As such, the regular grid can provide very close solution to the true solution, reducing the number of the GPU-CPU hybrid iterations required.

After examining all the IBM power grid benchmarks [66, 67] and several industrial designs, we found that these designs exhibit globally uniform gird structures, except for some local grids that have irregular patterns. If we are able to form an artificial $2D$ regular mesh structure and map all the original multi-layer irregular grid elements (resistors, current and voltage sources) onto this regular grid based on the geometrical and electrical information, the structure of the original power supply network can be well preserved. During the above mapping, we simply neglect the impact of the via resistors which may usually introduce relatively small errors. As demonstrated in Table VII, for the five benchmark power grid designs, the average solution differences of the top metal layer nodes and their corresponding bottom metal layer nodes are relatively small (much smaller than 1 mV). We propose an efficient yet effective mapping procedure that has three subsequent steps:

1. *3D irregular to 2D irregular mapping.* By neglecting via resistances, all the metal layers in the network are overlapped on the same 2D plane, forming a

Table VII. IBM power grid benchmark solution comparisons for the top and bottom layer node solutions (ΔV is the solution ranges, E_{avg} is the average solution difference of the top and bottom layer nodes and $E_{rel}\%$ is the relative solution difference , for the VDD/GND grids respectively).

CKT	N_{node}	N_l	$\Delta V(mv)$	$E_{avg}(mv)$	$E_{rel}\%$
$ibmpg2$	$127,238$	5	337/270	0.87/0.63	0.06/0.20
$ibmpg3$	$851,584$	5	171/151	0.01/0.01	0.01/0.10
$ibmpg4$	$953,583$	6	5.3/2.4	0.02/0.02	0.00/0.78
$ibmpg5$	$1,079,310$	3	48/27	0.05/0.04	0.00/0.15
$ibmpg6$	$1,670,494$	3	154/112	0.1/0.06	0.01/0.10

collapsed 2D irregular grid. Based on analyzing industrial power grid benchmarks, we found that neglecting via resistances typically does not alter the circuit solution in any significant way (Table VII). Nevertheless, the error components introduced by via removal can be quickly damped through our hybrid multigrid iterations.

2. *2D irregular to 2D regular mapping.* By examining the pitches in the collapsed 2D irregular grid, a fixed uniform pitch is chosen for the X and Y directions for the final 2D regular grid, on which all the circuit elements are mapped to.

3. *Circuit element mapping.* After introducing the new regular grid nodes, circuit elements can be mapped onto those nodes. The elements that occupy multiple regular grid nodes have to be decomposed into smaller pieces before mapping, while all the current and voltage sources are simply mapped according to their geometrical locations.

As an example, let us consider a simple example shown in Fig. 39, where a two-metal-layer irregular grid is mapped to a single-layer regular grid. The resultant

conductance values on the regular grid can be calculated as follows:

$$\begin{aligned} &G_1 = 2g_{31} + g_{21}, G_2 = 2g_{31} + g_{22}, G_3 = 2g_{32}, \\ &G_4 = 2g_{32} + g_{23}, G_5 = g_{33} + g_{24}. \end{aligned} \tag{3.4}$$

Note that because of irregularity of the original grid, some of the regular grid nodes may not correspond to any of the original nodes. In this case, small dummy conductances ($G_{min} = 1e-6$) are inserted between such regular grid node and its neighboring nodes. Note also that the uniform pitches of the regular grid may be set to the averaged pitch values in the irregular grid and can be adjusted when appropriate. Smaller uniform pitch values lead to increased regular grid size and improved grid approximation. The possible grid size increase in the regular grid does not significantly impact the overall runtime efficiency of our approach due to the linear complexity of the GPU GMD solver. The improved grid approximation, however, may contribute to faster HMD convergence. As will be demonstrated later, both the accuracy and efficiency of our GpuHMD algorithm are not sensitive to the regular grid size. This is the case even when the regular grid size is varied from 50% to 150% of the original grid size.

2. Table-Based Representation of the Regular Grid

The $2D$ regular grid is represented by several tables, denoted by G_h, G_v, G_z and I_z. The simple representation allows for efficient coalesced memory access to the device memory and is shown to be critical to the GPU implementation. For a regular grid node $N[i, j]$, the following four tables are adopted:

$G_h[i, j]$: Horizontally connected conductance between node $N[i, j]$ and node $N[i + 1, j]$;

$G_v[i, j]$: Vertically connected conductance between node $N[i, j]$ and node $N[i, j+1]$;

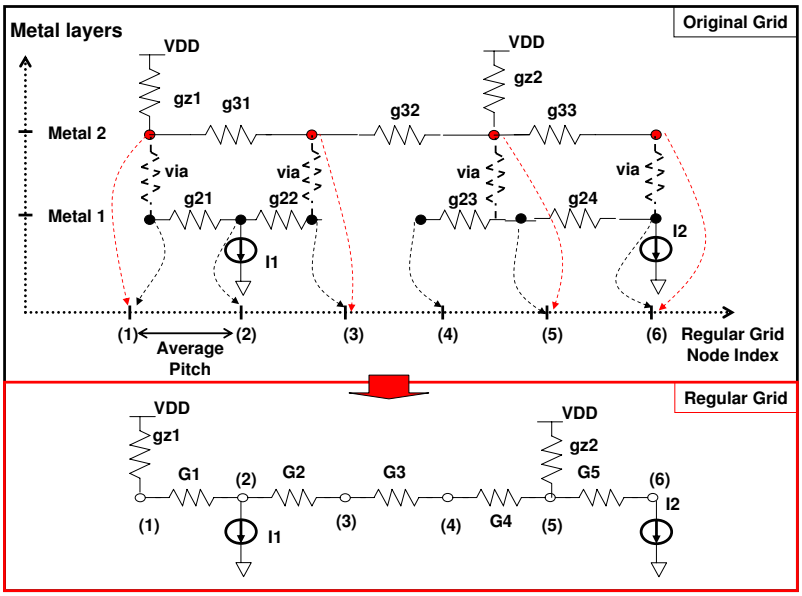

Fig. 39. Cross section view of mapping a two-layer irregular grid to a single-layer regular grid.

$G_z[i,j]$: The conductance that connects node $N[i,j]$ and the voltage sources;

$I_z[i,j]$: The current sources that flows out node $N[i,j]$.

C. Geometric Multigrid on GPU

While the 2D regular grid can be obtained in a relatively straightforward manner, developing an efficient regular grid solver on GPU is non-trivial. Naive implementations for either data transferring or processing can lead to severe performance degradation. The proposed GPU based GMD solver is described by covering the key issues concerning the discussion in Section 2.

1. Coarse Grid Generation and Inter-grid Operators

With the mapped regular 2D grid sitting at the bottom of the multigrid hierarchy, a set of increasingly coarser grids shall be created to occupy the higher levels. In

this case, the regular grid produced by the previous mapping step serves as the finest grid in our GMD method. Ideally, these coarse grids should be created such that the increasingly global behavior of the finest grid is well preserved using a decreasing number of grid nodes. Unlike in CPU based multigrid methods, here, it is critical to carry the regularity of the finest grid throughout the multigrid hierarchy so as to achieve good efficiency on the GPU platform. The goal is achieved from the following view of the I/O characteristics of the power grid.

When creating the next coarser grid, we distinguish two types of wire resistances: resistances connecting a grid node to a VDD source (or VDD pad conductances) vs. those connecting a grid node to one of its four neighboring nodes (or internal resistances) on the regular grid, as shown in Fig. 40. Importantly, the two types of resistances are handled differently. We maintain the same total current I_z that flows out the network and the same total wire conductance (G_z) that connects the grid to ideal voltage sources (e.g. total VDD pad conductance). In this way, the same pull-up and pull-down strengths are kept in the coarser grid of a power distribution network. Denote the voltages of M grid nodes that connect to an ideal voltage source via a wire resistance by V_i for $i = 1, ..., M$, and the N loading current sources by I_j for $j = 1, ..., N$. The following equation holds:

$$\sum_{i=1}^{M} (VDD - V_i)\, G_{z_i} = \sum_{j=1}^{N} I_{z_j}. \tag{3.5}$$

To maintain approximately same node voltages V_i at VDD pad locations in the coarser grid, we ensure that $\sum_{i=1}^{M} G_{z_i}$ and $\sum_{j=1}^{N} I_{z_j}$ are unchanged. Consequently, as shown in Fig. 40, both the VDD pad conductance (G_z) and current loadings (or residues) are summed up when creating the coarser grid problem. Differently, internal conductances are averaged to create a coarser regular grid that approximately

preserves the global behavior of the fine grid.

Use H and h to indicate the fine and coarser grid components, respectively, the coarser grid is created as follows:

$$\begin{aligned} G_h^h[i,j] &= \frac{1}{4} \times \big(G_h^H[2i,2j] + G_h^H[2i+1,2j] + \\ &\qquad G_h^H[2i,2j+1] + G_h^H[2i+1,2j+1]\big), \\ G_v^h[i,j] &= \frac{1}{4} \times \big(G_v^H[2i,2j] + G_v^H[2i+1,2j] + \\ &\qquad G_v^H[2i,2j+1] + G_v^H[2i+1,2j+1]\big), \\ G_z^h[i,j] &= \big(G_z^H[2i,2j] + G_z^H[2i+1,2j] + \\ &\qquad G_z^H[2i,2j+1] + G_z^H[2i+1,2j+1]\big), \end{aligned} \tag{3.6}$$

where i and j denote grid locations, and the numbers of nodes along the horizontal and vertical directions are reduced by a factor of two in the coarser grid. The restriction and prolongation operators are:

$$\begin{aligned} &R^h[i,j] = R^H[2i,2j] + R^H[2i+1,2j] + \\ &R^H[2i,2j+1] + R^H[2i+1,2j+1], \end{aligned} \tag{3.7}$$

$$\begin{aligned} &E^H[2i,2j] = E^H[2i,2j+1] = E^H[2i+1,2j] = \\ &E^H[2i+1,2j+1] = E^h[i,j], \end{aligned} \tag{3.8}$$

where residues and errors (solution corrections) are denoted by R and E, respectively. Apparently, the coarser grid problem is defined completely based on geometry and can be stored in the same regular table-based representation. In our GMD implementation, the coarsest grid is solved via a direct method on the host. To reduce the overhead of this sparse matrix solve on CPU and fully utilize the GPU computing power, the GMD hierarchy is purposely made deep. In our implementation, four to five grid levels are used, making the size of the coarsest problem vary from a few

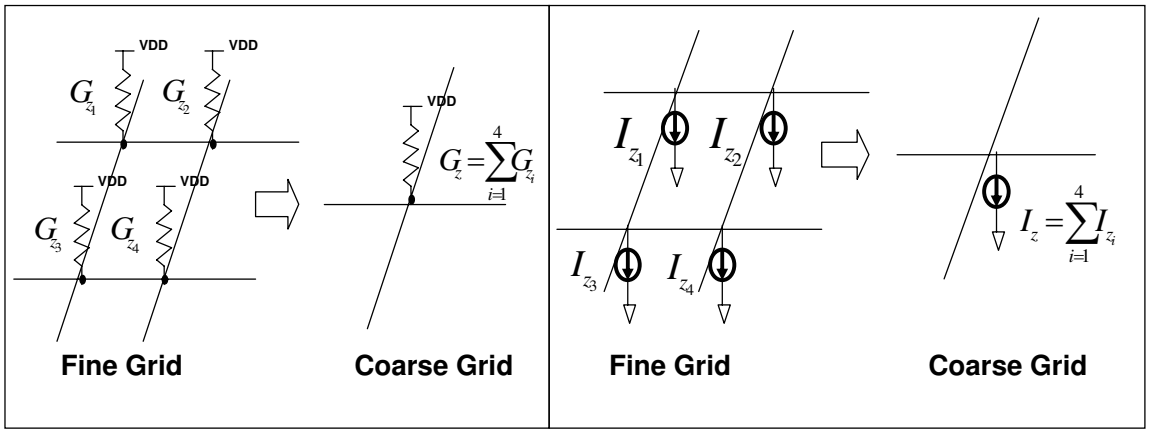

Fig. 40. VDD pads (G_z) and current sources (residues) in fine/coarse grids.

hundred to a few thousand times smaller than the finest grid. This choice may push, say 95%, of the overall computation onto the GPU.

2. Point vs. Block Smoothers

The choice of smoother is critical in GMD. Typically, point Gauss-Seidel or weighted Jacobi smoothers are used for CPU based GMD methods. However, a block based smoother is adopted in our approach to fully utilize the SIMT GPU computer power. On GPU, a number (more precisely a warp [27]) of threads may be simultaneously executed in a single-instruction multiple-data fashion on a multiprocessor. This implies that multiple circuit nodes can be processed in the smoothing step at the same time. In our approach, a block of circuit nodes are loaded into a multiprocessor at a time. Then, multiple treads are launched to simultaneously smooth the circuit nodes in the block for a number of iterations. As a result, such processing step (almost) completely solves the circuit block, effectively leading to a block smoother. This approach ensures that a meaningful amount of compute work is done before the data is released and a new memory access takes place. In other words, it contributes to

efficient GPU computing by increasing the arithmetic intensity. This block smoother is discussed in detail in Section D.

D. Accelerating GMD on GPU

To gain good efficiency on the GPU platform, care must be taken to facilitate thread organization, memory and register allocation, workload balancing as well as hardware-specific algorithms.

1. Thread Organization

Through a suitable programming model (e.g. CUDA [27]), threads shall be packed properly for efficient execution on multiprocessors. On a multiprocessor, threads are organized in units of blocks, where the number of blocks should be properly chosen to maximize the performance. The optimal block size shall be multiples of 32 threads for a commercial GPU [27]. In our implementation, the actual optimal block size is chosen experimentally.

2. Memory and Register Allocation

Before the GMD solve starts on GPU, $1D$ tables are allocated on the CPU to store all the regular grids in the multigrid hierarchy. Then, the data are transferred to the device (CPU). We bind the conductance tables (G_h, G_v and G_z) to the texture memory and other data to the on-board GPU device memory. Texture memory is cached, so its access latency is significantly smaller than the device memory. However, the texture memory is read-only and cannot be used for solution updates. Therefore, residues, solution and error vectors are stored in the device memory. Since the device memory is not cached, coalesced memory accesses are employed to achieve the best

memory bandwidth.

The fast on-chip shared memory and registers are very limited resources on GPU. If the required shared memory and registers exceed what are available, an application will fail. On the other hand, more than one block of threads should be run on the same stream multiprocessor (SM). This will hide the memory read/write latency in a better way, leading to a much higher performance throughput. With this in mind, all components of our GPU GMD method are developed carefully to fully utilize GPU resources. As an example, in the smoothing steps, the solution and right hand side (RHS) vectors are loaded from the global memory to the shared memory, while the resistance grid data are loaded from the the texture memory to the registers. The above scheme allows more than two blocks of threads to be launched concurrently within the resource limitation on an SM. Otherwise, if the grid data were loaded to the shared memory, only one block of threads could be run, making the memory access latency a higher impact.

3. Mixed Block-wise Smoother

In our GMD solver, the relaxation (smoothing) steps dominate the overall computation. Hence, an efficient implementation of the smoother is critical. On CPU, point-wise iterative methods such as Gauss-Seidel or weighted Jacobi are often adopted. However, to improve the arithmetic intensity and work better with efficient coalesced (block) memory accesses and control flows, *global block Gauss-Seidel iteration* (GBG iteration) and *local block weighted Jacobi iteration* (LBJ iteration) schemes are introduced.

As illustrated in Fig. 41, during each GBG iteration, the whole $2D$ regular grid is partitioned into small blocks which are subsequently transferred to streaming processors. Next, k times LBJ iterations are conducted within each block locally.

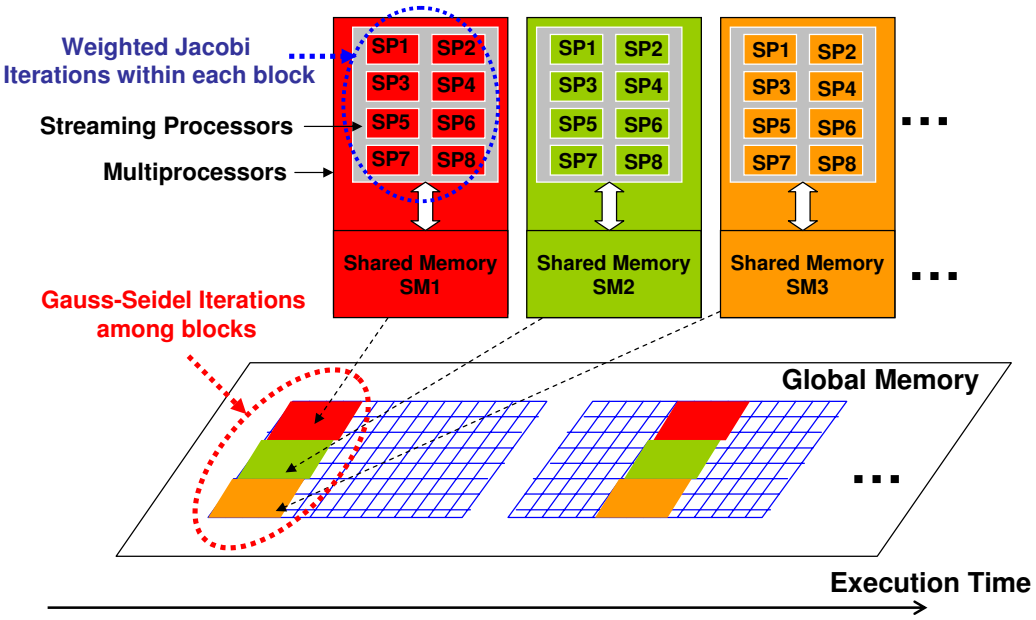

Fig. 41. Mixed block relaxation (smoother) on GPU.

Since only the threads within the same thread block can share the data with each other, the solution of this local block can not be shared by others unless it is sent back to the global memory. As processed block solutions are written back to the global memory, the smoothing of subsequent blocks will be based upon the most recent solutions of the neighboring blocks. Therefore, from this global point of view, the smoother is a block Gauss-Seidel iterative (or GBG) method. On the other hand, when each block is being smoothed, all its nodes are processed by multiple threads simultaneously in a weighted Jacobi fashion, referred to as LBJ iterations. The above mixed block-iteration scheme has been carefully tailored for our GPU based GMD engine, particulary through the following considerations:

1. To increase the arithmetic intensity, we perform k times LBJ iterations for each global memory access. k can be determined based upon the block size: larger block size may include more local iterations. However, excessive local iterations

may not help the overall convergence since the boundary information is not updated.

2. To hide the memory latency and thread synchronization time, we allow two or more blocks to run concurrently on each multiprocessor to avoid idle processors during the thread synchronization and device memory access.

The block size may impact the overall performance significantly. A too large block size may lead to slow convergence while a too small size may cause bad memory efficiency and shared memory bank conflicts. To minimize shared memory and register bank conflicts, block sizes such as 4×4 or 8×8 are observed to offer good performance.

4. Selection of Block Size

The block size used for the above relaxation may impact the overall performance significantly. Too large block size may lead to slow convergence while too small size may cause bad memory access latency. We apply a fixed hybrid block-wise relaxation scheme ($k = 5$) but different block sizes on IBM power grid benchmark *ibmpg*5, and demonstrate the results in Table VIII. From the table it can be found that when using five ($k = 5$) local iteration for each global update, the optimal block size is 4×4. Empirically, it is possible to find optimal number (k) of local iterations for different block sizes. For instance, the optimal $k = 10$ can be used for the block size of 4×4 and $k = 20$ for the block size of 8×8.

5. Dummy Grid Nodes

As discussed before, GPU data processing favors block-like operations. If the grid dimensions are not multiples of the block size, extra handling is required. For example, assume one smoothing kernel of the GMD solver is executed on all multigrid levels

Table VIII. Impact of partition size on the performance of the GMD solver implemented on GPU for the GND grid of *ibmpg*5 benchmark. $Part.\ Size$ is the block size for GPU processing, $\#of\ Iter.$ is the total number of iterations ($N_{GBG} * k = 5N_{GBG}$) for each level in the smoothing step, $\#of\ Vcyc.$ is the number of V cycles needed for the desired accuracy level, and $Runtime$ is the overall execution time of the GMD solve. CPU refers to the results obtained on the host using point-wise iteration scheme.

$Part.\ Size$	16×16	8×8	4×4	2×2	CPU
$\#of\ Iter.$	150	150	50	50	50
$\#of\ Vcyc.$	17	10	8	5	10
$Runtime$	$1.21s$	$0.54s$	$0.48s$	$0.72s$	$12.9s$

based on 8×8 thread blocks. Then, all the grid widths and heights need to be modified to be multiples of the block size. To this end, certain dummy grids can be attached to the periphery of the original grid. It is important to isolate these dummy grids from the original grid, as shown in Fig. 42. Otherwise, the GMD convergence can be significantly impacted.

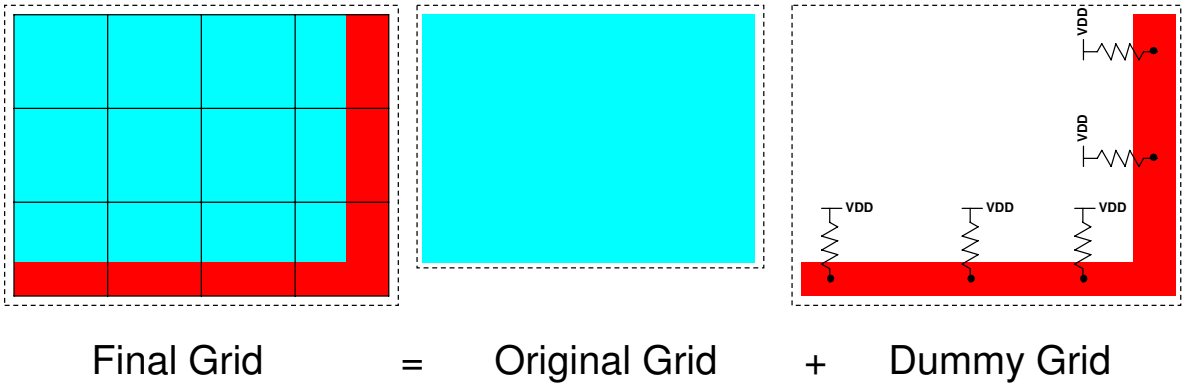

Fig. 42. Appending dummy grid nodes for a chosen block size.

6. A Simple Example of GBG Iteration

Denoting the solution by V, we demonstrate the key steps of one GBG iteration (block size is 8×8 in this example) using shared memories and registers on GPU as follows:

1. Load 8×8 V data from the global memory to the the shared memory and 8×8 $G_h,\ G_v,\ G_z,\ RHS$ data to the registers;
2. Load the four boundary data of $V,\ G_h,\ G_v,\ G_z,\ RHS$ to the shared memory and registers as well;
3. Do k times LBJ iterations using a 8×8 thread array;
4. Return the 8×8 solution V to the global memory for updating;

As shown above, the central 8×8 nodes can be updated by the 8×8 threads simultaneously, which is not possible for CPU implementations.

E. Hybrid Multigrid for Power Grid Analysis

Although solving the mapped 2D regular grids on GPU typically provides pretty accurate results, the solution quality may not be completely guaranteed since grid approximations can lead to various accuracy levels. To have a robust error control scheme, interactions between the 2D regular grid and the original 3D irregular grid are important. In this work, we propose a hybrid multigrid (HMD) analysis framework to iteratively correct the error components that are caused by grid approximation. The main steps of our HMD flow is shown in Fig. 38 and Fig. 43, and also outlined in Section 3.

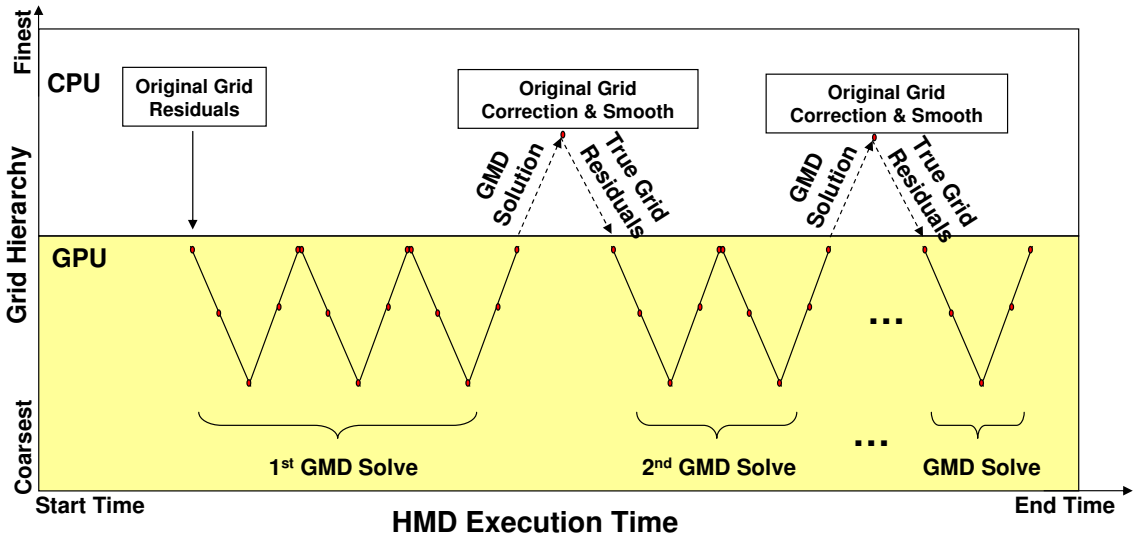

Fig. 43. Hybrid GPU-CPU multigrid iterations.

1. Problem Formulation

Assuming for a 3D irregular power grid ($Grid_O$), the following large linear system of equations need to be solved:

$$A \cdot x = b, \tag{3.9}$$

where $A \in \mathbb{R}^{n \times n}$ is the original grid system matrix, representing a linear operator $A(x) : \mathbb{R}^n \to \mathbb{R}^n$, $x = x^* \in \mathbb{R}^n$ is the exact solution vector to be solved, and $b \in \mathbb{R}^n$ is the right hand side (RHS). Denote the system matrix of the mapped 2D regular grid ($Grid_R$) as $A_r \in \mathbb{R}^{m \times m}$, which is a linear operator $A_r(x) : \mathbb{R}^m \to \mathbb{R}^m$. Denote the solution of the original grid in the k-th HMD iteration as $x^{(k)} \in \mathbb{R}^n$. The following steps are performed in the k-th HMD iteration. The residue $r^{(k)}$ associated with $x^{(k)}$ is computed and mapped onto $r_r^{(k)}$ on the 2D regular grid ($Grid_R$) as

$$r^{(k)} = b - A \cdot x^{(k)}, \quad r_r^{(k)} = V_o^r \cdot r^{(k)}, \tag{3.10}$$

where $V_o^r \in \mathbb{R}^{m \times n}$ is a proper linear operator ($\mathbb{R}^n \rightarrow \mathbb{R}^m$). Note that the above simple computations are done on the CPU. With $r_r^{(k)}$, a solution correction $e_r^{(k)}$ is computed on the regular grid using the GPU GMD method:

$$A_r \cdot e_r^{(k)} = r_r^{(k)}. \tag{3.11}$$

$e_r^{(k)}$ is returned to the CPU host for further processing. $e_r^{(k)}$ is mapped back to the original grid ($Grid_O$) via:

$$e^{(k)} = V_r^o \cdot e_r^{(k)}, \tag{3.12}$$

where $V_r^o \in \mathbb{R}^{n \times m}$ is a proper linear operator ($\mathbb{R}^m \rightarrow \mathbb{R}^n$). The solution for the original grid is updated:

$$x^{(k+1)} = x^{(k)} + e^{(k)}. \tag{3.13}$$

Finally, if the solution correction ($e^{(k)}$) is below a user-defined threshold, $x^{(k+1)}$ is returned as the final solution; otherwise, proceed to the $k+1$-th HMD iteration. The inter-grid ($Grid_O$ and $Grid_R$) mapping operators V_o^r and V_r^o may be interpreted as an prolongation or restriction operator, respectively, as in a classical multigrid method, depending on the relative sizes of $Grid_O$ and $Grid_R$. They are also constructed in a way similar to prolongation or restriction operators.

2. Convergence Analysis

Experimentally, it is observed that the proposed HMD approach can converge in a few iterations. To gain further insights on the converge property, the following theoretical result is proved.

Theorem 4 *Denote the spectral radius of an $l \times l$ matrix M by $\rho(M)$, where $\rho(M) = max_{i=1,\cdots,l}|\lambda_i|$, λ_i is an eigenvalue of M. The HMD iteration converges to the true*

solution x^* for any chosen initial guess $x^{(0)}$ if and only if:

$$\rho\left(I - V_r^o \left(A_r\right)^{-1} V_o^r A\right) < 1. \tag{3.14}$$

Proof At the k-th HMD iteration, the residue on $Grid_O$ ((3.10)) can be written as:

$$r^{(k)} = A\varepsilon^{(k)} = A\left[x^* - x^{(k)}\right], \tag{3.15}$$

where $\varepsilon^{(k)}$ is the solution error w.r.t. x^* and shall not be confused with $e^{(k)}$ in (3.12). Combining (3.10), (3.11) and (3.12) leads to

$$e^{(k)} = V_r^o A_r^{-1} V_o^r r^{(k)}. \tag{3.16}$$

From (3.13), (3.15) and (3.16), we have

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + e^{(k)} \\ &= x^{(k)} + V_r^o A_r^{-1} V_o^r A\left[x^* - x^{(k)}\right]. \end{aligned} \tag{3.17}$$

Let $B = I - V_r^o A_r^{-1} V_o^r A$. Substituting the definitions $\varepsilon^{(k)} = x^* - x^{(k)}$ and $\varepsilon^{(k+1)} = x^* - x^{(k+1)}$ into (3.17), we have:

$$\begin{aligned} \varepsilon^{(k+1)} &= \left[I - V_r^o A_r^{-1} V_o^r A\right] \varepsilon^{(k)} \\ &= \left[I - V_r^o A_r^{-1} V_o^r A\right]^{k+1} \varepsilon^{(0)} \\ &= B^{k+1} \varepsilon^{(0)}. \end{aligned} \tag{3.18}$$

It is not difficult to see that for any $\varepsilon^{(0)}$ (or $x^{(0)}$), if $\rho\left(B\right) < 1$, $\varepsilon^{(k)}$ converges to a zero vector. Furthermore,

$$\left\|\varepsilon^{(k+1)}\right\| \leq \|B\|^{k+1} \left\|\varepsilon^{(0)}\right\| \geq \left[\rho\left(B\right)\right]^{k+1} \left\|\varepsilon^{(0)}\right\|. \tag{3.19}$$

It implies that if $\varepsilon^{(k)}$ converges to zero for any $\varepsilon^{(0)}$ (or $x^{(0)}$), it must be true that $\rho\left(B\right) < 1$.

Theorem 1 provides a very intuitive understanding of the convergence property of the HMD iterations and offers a theoretical basis to further improve the convergence rate. Let $C = V_r^o A_r^{-1} V_o^r$, $C \in \mathbb{R}^{n \times n}$. C can be interpreted as a linear operator ($\mathbb{R}^n \rightarrow \mathbb{R}^n$), which corresponds to the correction operator of the $Grid_O$ solution $x^{(k)}$ by solving an approximate problem defined by $Grid_R$. If there exists no grid approximation in $Grid_R$, then the original power grid can be solved exactly on the regular grid: $C = A^{-1}$. In this ideal case, $\rho(B) = \rho(I - CA) = 0$, implying that HMD converges in one iteration. In practice, the regular grid problem needs not to be exactly identical to the original grid problem to have HMD converge fast, as long as it is sufficiently close. Here, the *closeness* is measured by $\rho(B)$ (the smaller the better).

In our implementation, we have found that applying a few, say m, additional simple point Gauss-Jacobi relaxations to further improve the solution obtained in (3.13) is very beneficial. In this case, the spectral radius (3.18) of the HMD iterations becomes $\rho\left(B\left(I - D^{-1}A\right)^m\right)$, where D is the diagonal matrix corresponding to Gauss-Jacobi iterations. This only adds a small additional cost on the host, but makes the spectral radius even smaller, improving the overall convergence rate.

3. HMD on Multi-Core-Multi-GPU System

The proposed HMD algorithm is highly parallelable, and the workload can be easily partitioned based on the geometrical information of the power grid circuit. In this work, we propose to parallelize the HMD simulation on multi-core-multi-GPU system (Fig. 44). As a simple example, if we want to solve an N-node power grid on a quad-core machine with four GPUs, each core is used to talk with a GPU card such that a smaller grid partition (e.g. a grid partition with N/4 nodes) can be solved independently on GPU by GMD. When the solutions of all partitions are fed back to the full grid, a few smoothing steps can be performed and residuals can be computed

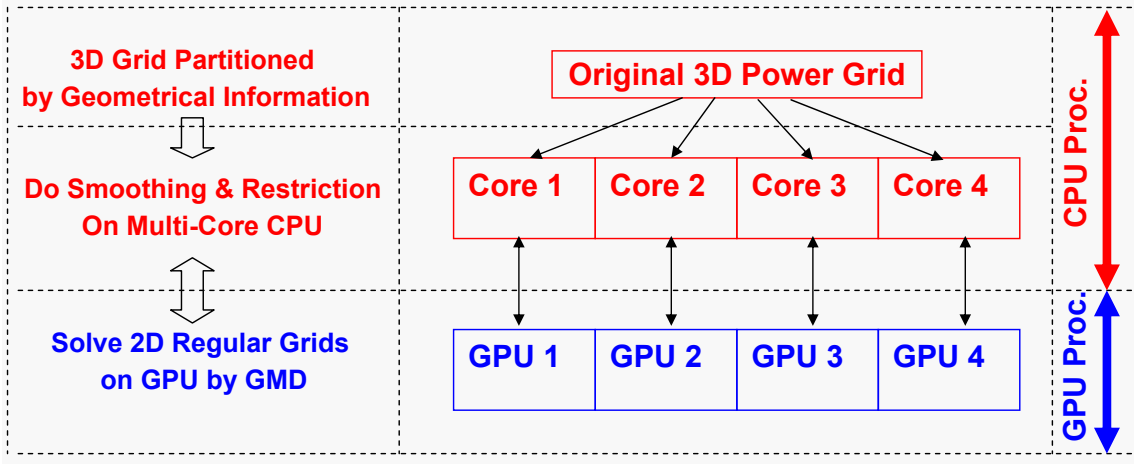

Fig. 44. Power grid simulation scheme on multi-core-multi-GPU platform.

subsequently. New RHS vectors (residuals on full grid) are sent to individual GPUs for the subsequent GMD solving. In this way, even very huge power grid designs can be handled very efficiently. For instance, if each of the GPU can solve up to ten million node (with 512 Mb device memory), then the multi-core-multi-GPU system can solve up to 40 million node power grid within a few seconds, which is not possible ever before.

To implement the multi-core-multi-GPU multigrid algorithm, the GPU code (written in Nvidia's CUDA language [27]) can be compiled to a static library that can be further invoked in the C/C++ code. Pthreads library [69] is used for the multithreading programming and each thread will control a single GPU card throughout the computation. Care must be given to designing data structure of the GPU code: Different grid partitions must be stored in the on-board memories of different GPU devices and the full grid solution will be updated in the shared memory by the multi-core CPU during each CPU-GPU iteration procedure. To minimize to the total data communications between the host and device, we suggest to exchange only the boundary data of each partitioned grid.

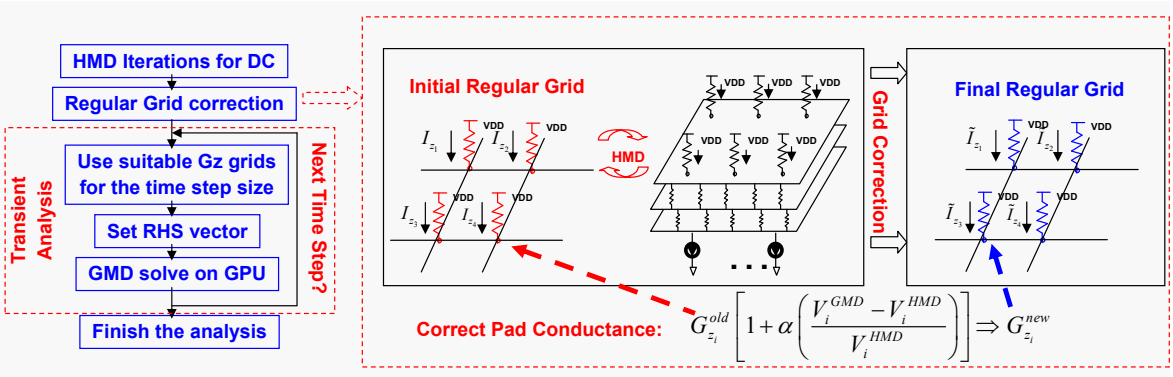

Fig. 45. Runtime composition of GpuHMD.

4. Accuracy and Overhead

The proposed HMD iteration scheme favorably enhances the robustness of the proposed algorithm and relaxes the need for a very accurate 3D-irregular-to-2D-regular grid mapping (Section B). For a set of power grid benchmarks, as the regular grid size is varied from 80% to 120% of the original grid size, running two HMD iterations can always reach a very satisfactory accuracy level of less than $1mV$ average node voltage error. Increasing to three iterations will cut the average error down to less than $0.5mV$. In addition to the solution of a small sparse matrix problem (the coarsest grid), required by the GMD method, the host (CPU) also conducts simple smoothing, correction and residue computation steps. The CPU runtime is typically only 1/3 to 1/10 of the total GpuHMD runtime.

F. Power Grid Transient Analysis on GPU Using HMD

In this section, details for transient power grid analysis based upon the GpuGMD and GpuHMD solvers will be introduced. As discussed in [70, 71], on-chip power grid analysis only requires taking into account capacitive effects while the inductance can be ignored. In this work, we only consider capacitors for the transient analysis.

To facilitate more efficient transient simulation procedure, a regular grid correction scheme is proposed to improve the simulation efficiency of each time step. From the convergence analysis (as shown in Section 2) we know that a better regular grid approximation can potentially reduce the number of HMD iterations, thus improve the simulation efficiency. It is therefore helpful for us to make a better 2D regular grid based upon the DC analysis results which in turn improve the overall transient simulation efficiency. Furthermore, the new multigrid-based analysis flow allows more flexible time step controlling during the transient simulation than the fixed time step scheme which is based on matrix factorizations. Our transient analysis flow is depicted in Fig. 45 and includes the following key steps:

1. HMD iteration for DC solution. Some of the HMD results will be used for the regular grid correction step.

2. By examining the DC solution difference (between the GMD and HMD solvers) at the VDD/GND pad locations, the VDD/GND pad conductance values of the regular 2D grids are modified to compensate the effects due to the 3D-to-2D grid approximation.

3. For each time step, choose a suitable G_z grid based on the current time step size and set the RHS vector for multigrid solver. The solution obtained from GPU will be used for the smoothing and residual computation on CPU for improving the accuracy.

1. Regular Grid Correction Scheme

In the regular grid approximation step, the via resistors are simply neglected and the errors are corrected by the HMD iterations. It is obvious that the proposed regular

grid approximation neglects the via resistance effects and will always lead to underestimated IR drop/bounce values, since the electrical current across the vias are not taken into consideration. Fortunately, by making use of the DC results given by the HMD iterations, we are able to obtain a more accurate 2D regular grid for approximating the original 3D grid, which in turn will effectively reduce the HMD iteration numbers in the transient power grid analysis. The idea of our grid correction scheme

Algorithm 7 Regular Grid Correction Algorithm.

Input: The original 3D power grid ($Grid_O$) that has **N** nodes connected with the VDD/GND pads, the initial 2D regular grid ($Grid_R^{(0)}$) and the number of grid correction iterations **K**.

Output: The updated 2D regular grid.

1: **for k = 1 to K: do**
2: Do DC analysis for $Grid_O$ using **GpuHMD** solver with the latest 2D regular grid $Grid_R^{(k)}$. Store the solution of node i as V_i^{HMD}, if this node is connected to VDD/GND pads;
3: Do DC analysis for $Grid_R^{(k)}$ using **GpuGMD** solver. Store the solution of node i as V_i^{GMD}, if this node is connected to VDD/GND pads;
4: **for i = 1 to N: do**
5: Correct the VDD/GND pad conductances of the regular grid using: $\mathbf{G_{z_i}^{new} = G_{z_i}^{old}\left[1+\alpha\left(\frac{V_i^{GMD}-V_i^{HMD}}{V_i^{HMD}}\right)\right]}$;
6: **end for**
7: Update the regular grid to $Grid_R^{(k+1)}$ using the latest pad conductance values;
8: **end for**
9: Return the final 2D regular grid.

is shown in Fig. 45, where the VDD/GND pad values can be slightly varied according to the difference between the HMD solution and the GMD solution. For instance, for a VDD/GND location if the GMD solver gives overestimated/underestimated solutions, then the original 2D regular grid pad conductances should be reduced by some value to alleviate the overestimation/underestimation. Although there may be other ways for improving the regular grid approximation (such as changing the local wire conductance values), we observe that the VDD/GND pad conductance values can

significantly impact the DC solutions of the nearby grid nodes. Therefore, adjusting the VDD/GND pad conductance values can be very effective for better grid approximation, which aims to compensate the influences of the via resistors that sit between multiple metal layers. The detailed grid correction procedure can be performed in an iterative fashion, which is concluded in Algorithm 7.

In practical implementations, the coefficient α shown in Fig. 45 and Algorithm 7 can be empirically selected, which is usually no greater than 5. We emphasize that selection of the coefficient α will not impact the final grid correction accuracy but the number of grid correction iterations. Intuitively, the lager α value used, the more pessimistic GMD results will be obtained. In practise, using only one or two grid correction iterations are adequate for getting a pretty good regular grid. In our experiments, we found this simple grid correction scheme will produce more accurate regular grid approximation of the original 3D grid, and can lead to faster convergence of the HMD iterations, which may bring more advantages to the transient simulation of power grid circuits.

2. Flexible Time Step Control

As observed in the equations (3.2) and (3.3), to perform the transient analysis using the existing GMD solver, only the RHS vector and the G_z grid need to be modified for simulating the dynamic effects due to capacitors. Consequently, the following formulas can be applied to set the RHS values as well as the G_z grids during the transient simulation steps.

$$RHS^{TR} = -[I_x(t) - \frac{C_x}{h} V_x(t-h)]; \tag{3.20}$$

$$G_z^{TR} = G_z^{DC} + \frac{C_x}{h}. \tag{3.21}$$

From the above formulas, we find that as long as different G_z grids that correspond to different time step sizes have been setup on GPU device in advance, the power grid transient analysis using adaptive time step controlling can be naturally performed. Adaptive time step controlling technique is important for reducing the simulation cost (by examining the local truncation errors) when nonlinear devices are combined with the linear power delivery network, since much less time steps are needed to be analyzed. However, achieving such a flexible time step controlling is rather expensive when using the direct methods (sparse matrix solvers), since for different time step sizes, many times of matrix factorizations are needed and much more memory space (supper linear complexity) has to be taken for the matrix factors. On the other hand, the HMD-based transient simulation method using flexible time step controlling scheme costs much less memory (linear complexity), since only the G_z grids corresponding to different time step sizes are stored. In this paper, we will only use the fixed time step size for transient analysis.

G. Experimental Results

Extensive experiments are conducted to demonstrate the promising performance of the proposed GpuHMD engine. A set of published industrial power grids [66, 67] and synthetic benchmarks are used to compare five solvers: proposed GPU accelerated GMD solver (GpuGMD), the CPU implementation of the same algorithm (CpuGMD), proposed GPU accelerated hybrid solver (GpuHMD), the GPU implementation of the same algorithm (CpuHMD), and the state-of-the-art CPU-based direct sparse matrix solver CHOLMOD [72]. All the algorithms are implemented using C++ and the GPU programming interface CUDA [27]. The hardware platform is a Linux PC with Intel Core 2 Quad CPU running at 2.66 GHz clock frequency and two Nvidia Geforce

Table IX. DC analysis results of the GMD solver. $GridSize$ is the number of nodes of the original power grid, N_{Vc} is the number of V-cycles, $\Delta V(mv)$ is the solution range ($V_{max} - V_{min}$), E_{avg} is the average error, and E_{max} is the maximum error (the data for the VDD and GND grids are shown in the form of VDD/GND). T_C/M_C is the runtime/memory using Cholmod solver, $T_{GPU}/T_{CPU}(s)$ is the runtime using GMD on GPU/CPU (the above runtime are the total runtime for solving both the VDD and GND grids).

CKT	$GridSize$	N_{Vc}	$\Delta V(mv)$	$E_{avg}(mv)$	$E_{max}(mv)$	$T_{GPU}(s)$	$T_{CPU}(s)$
$ibmpg2$	$127,238$	4/4	347/275	3.7/2.5	21/8.3	0.08	2.41
$ibmpg3$	$851,514$	4/4	181/153	4.2/2.9	32/20	0.33	19.50
$ibmpg4$	$953,583$	8/8	5.3/2.6	0.1/0.1	0.6/0.3	0.24	13.20
$ibmpg5$	$1,079,310$	10/8	48/28	1.5/1.0	4.4/4.6	0.38	26.95
$ibmpg6$	$1,670,494$	10/10	154/86	3.6/1.4	20.1/10.3	0.46	40.06

9800 GX2 cards (including four GPUs and each of them has a similar performance as Geforce 8800 GTX GPU).

The block size used for relaxations may also greatly impact the overall performance. An excessively large block size may lead to slow convergence or even divergence while a too small block size usually results in bad memory efficiency. As observed in our experiments, smaller block size achieve better convergence rate (less V-cycles) in spite of their lower efficiency for memory access. It is found that the overall performance obtained using a block size of 4×4 is comparable to the block size of 8×8 for all power grid test cases, whereas the block size of 16×16 may cause divergence sometimes.

Table X. DC analysis results of the HMD solvers. N_{Iter} is the number of HMD iterations, E_{avg} is the average errors of the HMD solvers, E_{max} is the maximum errors of the HMD solvers, and E_{wst} is the worst voltage drop/bounce error. T_{HMD} is the runtime of HMD solve. The data for the VDD and GND grids are shown in the form of VDD/GND.

CKT	N_{Iter}	$E_{avg}(mv)$	$E_{max}(mv)$	$E_{wst}(mv)$	$T_H(s)$	*Speedup*
*ibmpg*2	2/2	0.3/0.2	2.7/3.5	0.4/0.1	0.12	25X
*ibmpg*3	2/2	2.1/1.0	12.0/8.2	3.0/1.2	0.72	43X
*ibmpg*4	1/1	0.0/0.0	0.3/0.3	0.0/0.0	0.46	34X
*ibmpg*5	2/2	0.6/0.2	4.4/2.8	2.4/2.9	0.83	48X
*ibmpg*6	3/3	0.6/0.2	5.5/1.8	1.4/0.2	1.15	55X
CKT	N_{Iter}	$E_{avg}(mv)$	$E_{max}(mv)$	$E_{wst}(mv)$	$T_H(s)$	*Speedup*
*ibmpg*2	3/3	0.0/0.0	2.3/1.2	0.0/0.0	0.18	24X
*ibmpg*3	3/3	1.0/0.6	10.0/5.4	1.1/0.9	1.15	44X
*ibmpg*4	2/2	0.0/0.0	0.2/0.1	0.0/0.0	0.62	34X
*ibmpg*5	3/3	0.4/0.2	3.0/2.9	1.5/1.5	1.10	48X
*ibmpg*6	4/4	0.5/0.2	4.3/1.5	0.0/0.2	1.58	51X

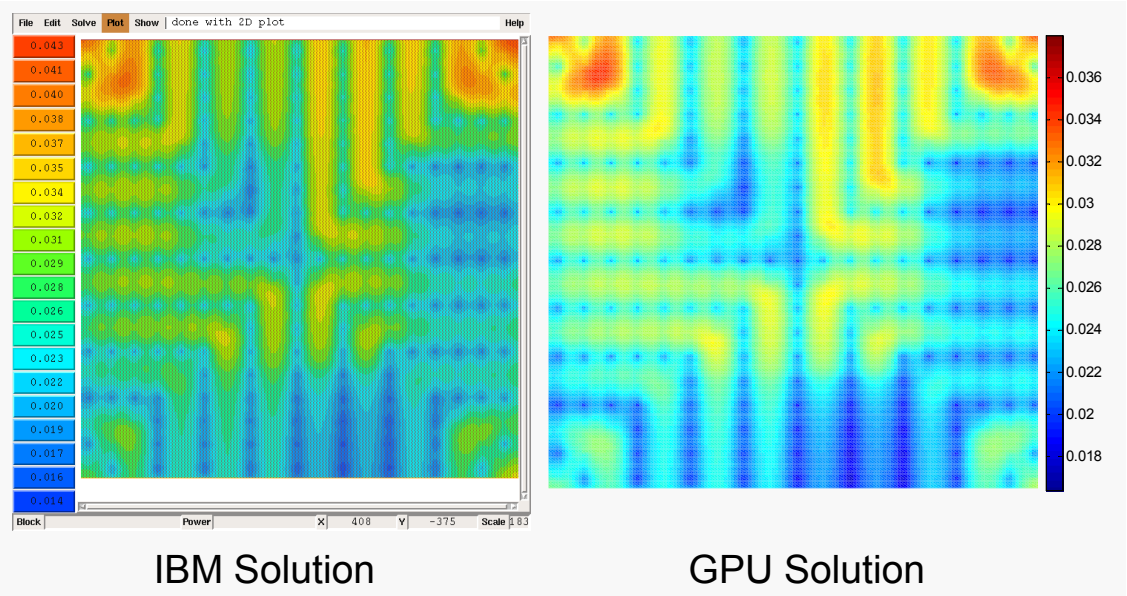

Fig. 46. Comparison of IBM solution and GpuGMD results.

1. DC Analysis Results

a. GMD and HMD Results

The GMD solvers are terminated when the residue reaches 0.5% of the initial residue. The HMD solvers are stopped when the (estimated) average node voltage error is less then $0.5mV$. The comprehensive results of GpuGMD and GpuHMD for all the industrial benchmarks are shown in Table IX and Table X. The results for VDD nets and GND nets are displayed as VDD/GND. GpuGMD is up to 87X faster than CpuGMD while GpuHMD is up to 55X faster than CpuHMD. In Fig. 46 we compare the spatial voltage distribution of *ibmpg*5 circuit given by IBM with our results obtained on GpuGMD solver, which indicates that without the help of HMD iterations, GpuGMD can provide a pretty accurate voltage distribution. Additionally, in Table X, we show the runtime/accuracy results when using different numbers of HMD iterations. As observed, using one more iteration, the accuracy can be improved significantly. For

Table XI. Runtime (ms) composition of 100 relaxations on GPU. The pure computation time T_c and total runtime T_t are listed as T_c/T_t. K is the number of local block-wise Jacobi (LBJ) iterations. Block size is 4×4.

CKT	$ibmpg2$	$ibmpg3$	$ibmpg4$	$ibmpg5$	$ibmpg6$
$k = 1$	25/60	45/223	60/357	49/254	73/471
$k = 5$	6.4/12.3	16/45	23/69	18/51	30/93
$k = 10$	4.2/6.9	13/24	20/37	15/28	26/49

most benchmarks, GpuHMD produces a less than $0.5mV$ average node voltage error and a less than $5mV$ maximum node voltage error.

b. Block Size Selection

As explained in Section 3, GPU memory access (read/write) latency can be dominant if the algorithm is not well implemented. When the block size is 4×4, for each choice of the local Jacobi (LBJ) iteration number k, the number of global iterations is empirically determined by $100/k$. The runtimes and ratios of the pure GPU computing time T_c over the total GPU runtime T_t (computing time+memory read/write time) for all industrial benchmark circuits are shown in Table XI and Fig. 47. From Fig. 47, we observe that the pure computation time T_c can only be a fraction (15% to 60%) of the total runtime T_t, while more local LBJ iterations (larger k) can better hide the memory access latency. However, it is less useful to do excessive local iterations, since they may not help the convergence of the overall GMD solve. Therefore, the number of local iterations (k) should be selected to tradeoff between the relaxation runtime and global convergence rate. We suggest $k = 10$ for the block size of 4×4 and $k = 20$ for the block size of 8×8 in practice.

The following insightful experiments are also conducted. 1000 smoothing steps

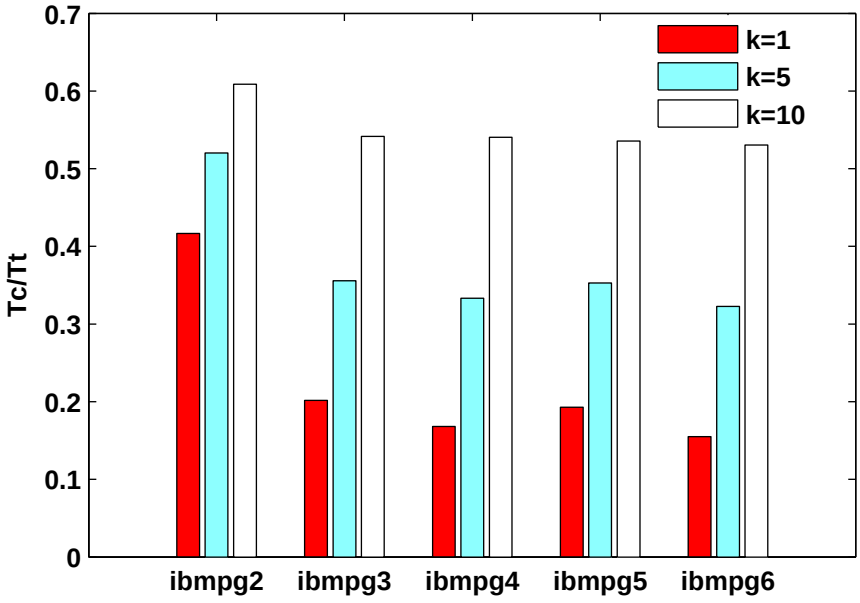

Fig. 47. Ratio of GPU computation time (T_c/T_t).

are run on both the CPU and GPU. As shown in Fig. 48 , the GpuGMD engine achieves $93X$ to $117X$ speedups over its CPU counterpart. The runtimes of the multi-V-cycle GMD solve are also compared on the GPU and CPU. As shown in Fig. 49, our GPU implementation achieves roughly $31X$ speedup for small grid and $87X$ speedup for large grids.

c. Errors of HMD Iterations

In Fig. 50, the runtimes of HMD iterations on CPU and GPU have been shown, where the GPU runtime takes more than 60 percent of the total runtime. To see the convergence behavior of the proposed HMD iterations, the spatial node voltage error distributions of the VDD net of IBM power grid benchmark *ibmpg*2 are shown in Fig. 51. The errors decrease drastically after two iterations, indicating the fast convergence of HMD. The average solution error as a function of the number of HMD

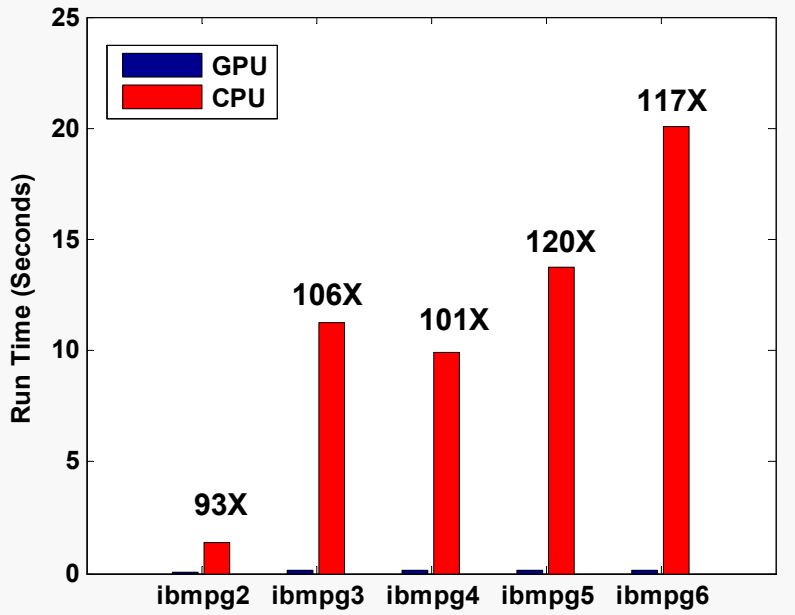

Fig. 48. Runtimes of $1K$ relaxations on CPU and GPU.

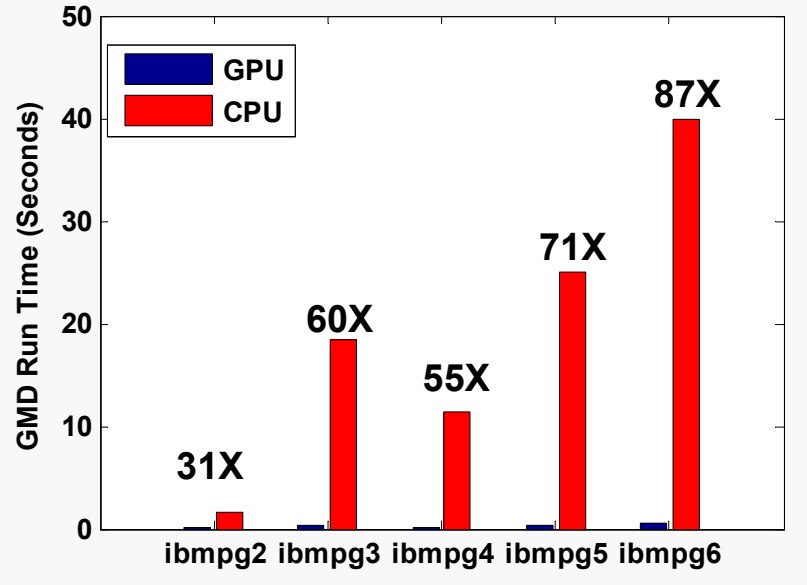

Fig. 49. Runtimes of CpuGMD and GpuGMD.

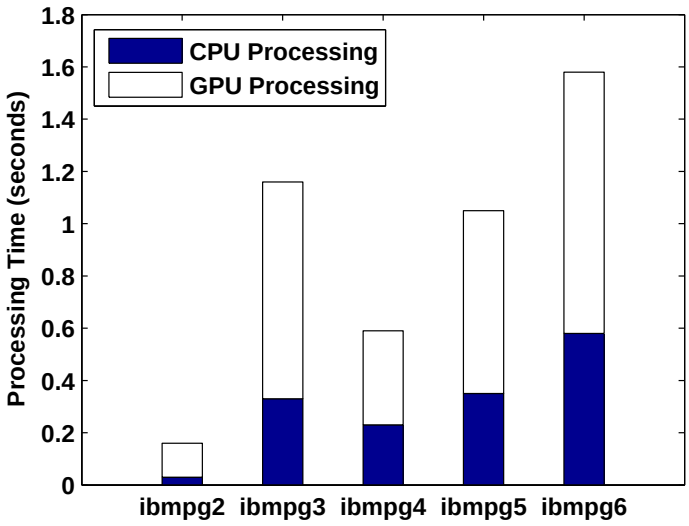

Fig. 50. Runtime composition of GpuHMD.

iterations is shown for the largest four industrial benchmarks in Fig. 52 (left). The average errors of all four benchmarks can be damped very quickly after two or three HMD iterations.

In Fig. 52 (right), the dependency of the total GpuHMD runtime on the regular grid size is shown for IBM benchmark *ibmpg*6. As the regular grid size is varied from 20% to 150% of the original grid size, the GpuHMD HMD runtime does not vary significantly under the same accuracy tolerance. This indicates that high accuracy in the 2D regular grid approximation is not needed. A reasonable regular grid approximation is sufficient for fast HMD convergence.

2. Grid Correction and Transient Analysis Results

The grid correction using the simple scheme proposed in Section 1 have been implemented and the test results on the largest benchmarks are shown in Table XII. As observed in the table, only after two to four HMD iterations, a more accurate regu-

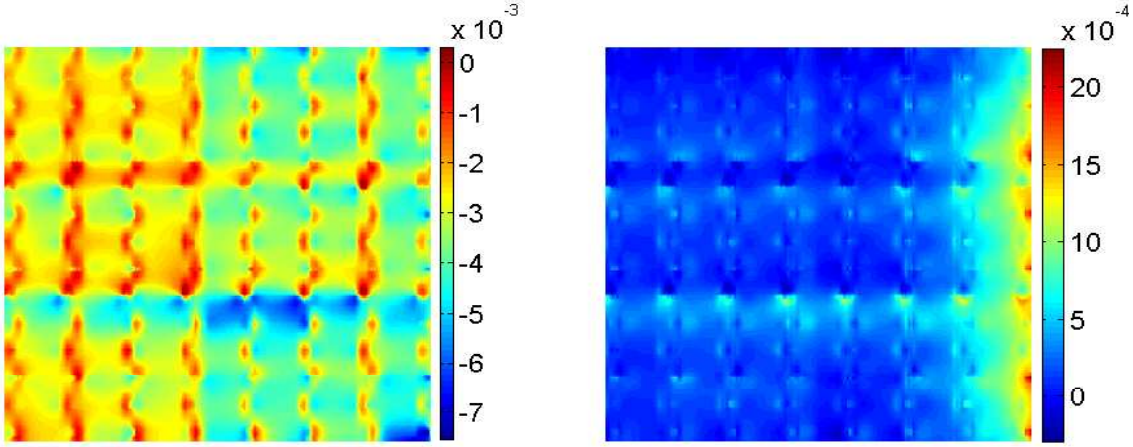

Fig. 51. Error distributions after the 1st (left) and 2nd (right) HMD iterations.

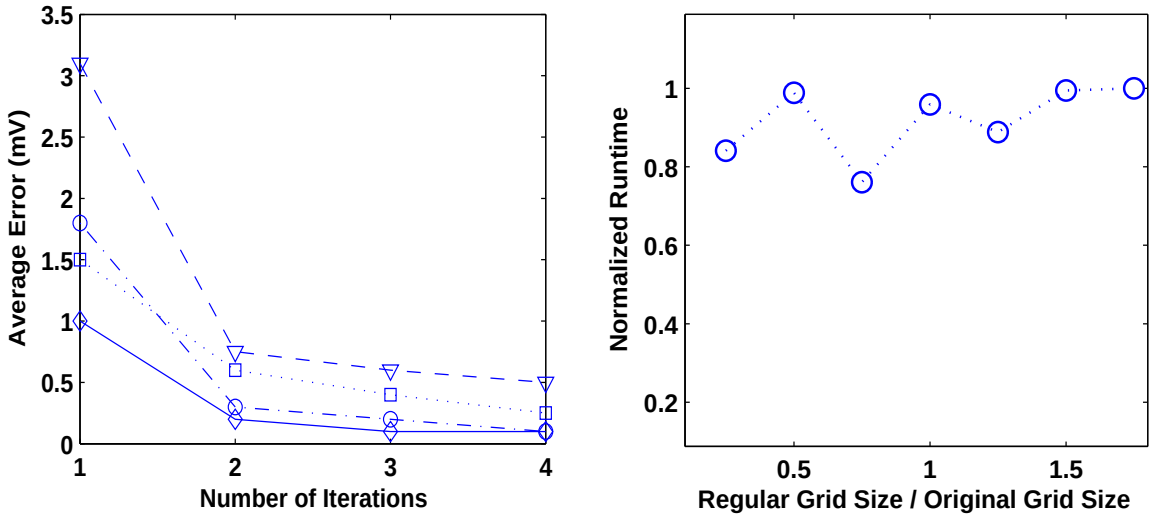

Fig. 52. Average error vs. HMD iteration count (left); regular grid size vs. runtime (right).

Table XII. Grid correction results using the HMD solver. N_{Iter} is the number of HMD iterations for grid correction, $E_{avg}/\widetilde{E}_{avg}$ and $E_{max}/\ \widetilde{E}_{max}$ are the average and maximum errors (mV) of the GMD solutions before/after the grid corrections. $E_{wst}/\widetilde{E}_{wst}$ is the worst voltage bounce errors (mV) before/after the grid corrections. $\widetilde{T}_{GMD}$ is the runtime of GMD solve using the regular grid after correction. Only the GND grid results are shown below.

CKT	N_{Iter}	E_{avg}	E_{max}	E_{wst}	$\widetilde{E}_{avg}$	$\widetilde{E}_{max}$	$\widetilde{E}_{wst}$	$\widetilde{T}_{GMD}$
*ibmpg*4	2	$< .1$	$< .1$	$< .1$	$< .1$	$< .1$	$< .1$	$< .1$
*ibmpg*5	3	1.0	4.9	-4.7	0.5	1.7	1.6	0.22
*ibmpg*6	4	1.7	8.9	-6.5	1.3	8.2	-3.2	0.37

lar grid can be obtained by simply modify the VDD/GND pad conductance values according to the GMD and HMD solutions. With such a better regular grid, the transient analysis can be performed more efficiently. In this work, we assume some typical decoupling capacitance values according to [73, 74], and run the transient analysis using the analysis flow in Section F. To guarantee the accuracy of each time step, we assure that the final residual is much smaller than the original ones. Interestingly, with the grid correction scheme, we can solve each time step using only one HMD iteration while the inside GMD solving usually converges in two V-cycles, which is much faster compared with DC analysis. Therefore, the transient simulation of each time step is much less expensive than the DC analysis. It is observed through several test cases that the runtime of each transient time step analysis is about 1/20 to 1/10 of the overall DC solving time.

Table XIII shows the results of 100 time steps' transient simulation using the GpuGMD solver, which is compared against the Cholmod solver. The running time of Cholmod only includes the re-solve time, and the matrix factorization time is not considered. The power grids considered in this experiment are generated using

Table XIII. Runtime (seconds) comparison of transient simulation (100 time steps). Average runtime for solving one time step is T_{GPU} (T_{chol}) when using the GpuGMD (Cholmod) solver. *Speedup* is defined as T_{chol}/T_{GPU}. Fixed time step is used for both cases.

$CKTSize$	$1M$	$2.25M$	$4M$	$7M$	$9M$
T_{GPU}	0.08	0.14	0.23	0.45	0.55
T_{Chol}	1.7	4.5	7.1	14.1	17.4
$Speedup$	$21X$	$32X$	$31X$	$31X$	$32X$

the typical wire/pad conductance values and current loadings observed in realistic industrial benchmarks [67]. As observed, the GPU transient simulator is roughly 20X to 30X faster than the direct matrix solver.

3. Scalability of GMD Solvers

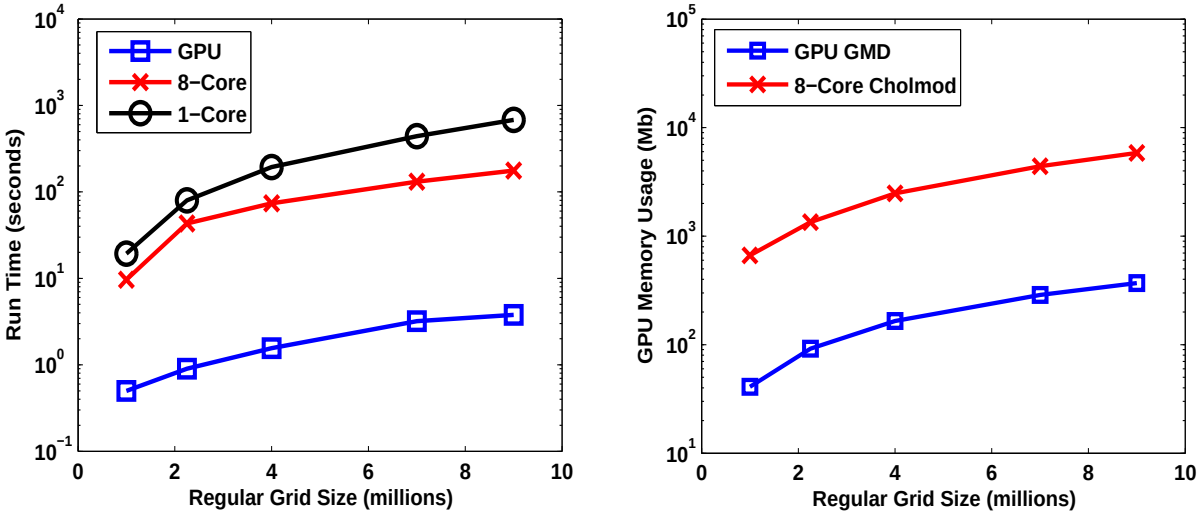

Fig. 53. Runtime scalability of GpuGMD (left); Memory scalability of GpuGMD (right).

Fig. 53 shows the runtime and memory comparison of GpuGMD and CHOLMOD

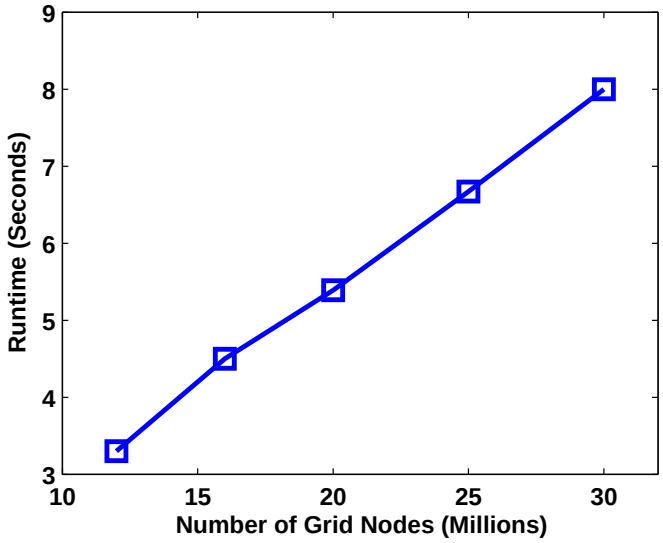

Fig. 54. Runtime of GpuGMD for solving very large grids.

for several large synthetic 2D (topologically) regular power grids (as the ones in Table XIII). GpuGMD is run on our four-core-four-GPU machine while the Cholmod is run on a more powerful computer (8-core Intel Xeon@2.33GHz with 8G RAM running 64-bit Linux). The runtime and memory consumption of GpuGMD solver increase linearly as the grid size increases, while the 1-threading (8-threading) Cholmod solver typically runs 100X (20X) slower and takes 20X more memory resources. The running times of GpuGMD for solving very large grids is plotted in Fig. 54, where the thirty-million grid has been solved in eight seconds. Our GpuGMD solver, the key component of GpuHMD, scales favorably with the circuit complexity, at a constant rate about one second (runtime) and $100Mb$ (memory) per two million nodes or more.

4. Multi-Core-Multi-GPU Results

In previous section, we mentioned how to utilize the multi-core computers and multi-GPU cards to further accelerate the HMD solver, where each CPU-GPU pair works

Table XIV. GMD results for multi-core-multi-GPU system. *Size* is number of nodes of the 2D regular grid, while T_N is the runtime of the GMD solver on N–core-N-GPU system. T_{Chol-N} is the runtime of Cholmod solver running on N-core CPU. *Spd.* is speedups of four-GPU GMD solver over the Cholmod solver running on eight-core CPU. All the runtime results are shown in seconds.

Size	T_1	T_2	T_3	T_4	T_{Chol-1}	T_{Chol-8}	*Spd.*
$4M$	1.7	1.1	0.75	0.56	194.2	73.7	$132X$
$8M$	3.5	2.1	1.35	1.1	561.4	154.3	$140X$

on a smaller partition of the original grid and the residuals and smoothing steps are computed on the full grid. In this section, we show the results of solving large 2D regular grids on multi-core-multi-GPU system, though the HMD irregular grid solver can be accelerated in the same way. The synthetic 2D regular grids are split into smaller partitions with similar sizes based on the grid geometries to well balance the workload. The runtime results of GMD solving using different numbers of GPUs and CPUs are shown in Table XIV, where we observe four-GPU system can achieve up to 140X speedups over the 8-core CHOLMOD solver.

H. Summary

In this work, we address the challenge of large-scale power grid analysis by developing a novel multi-core-multi-GPU acceleration engine. To gain good efficiency on GPUs, we propose to transform an irregular grid to a regular structure so as to eliminate most of random memory access patterns and simplify control flows. To properly exploit the massively parallel single instruction multiple thread (SIMT) GPU architecture, a parallel geometrical multigrid algorithm is specially designed. New coarse grid construction and block smoothing strategies are adopted to suit the SIMT GPU

platform. The robustness of the algorithm is well enhanced by an efficient CPU-GPU hybrid multigrid iteration scheme. Careful performance fine tuning is conducted to gain good analysis efficiency on the GPU. Extensive experiments have shown that the DC analysis accelerated on a single-core-single-GPU system can lead to $100X$ runtime speedups over a state-of-art direct solver and $50X$ speedups over the CPU based multigrid solver, while the transient analysis can be more than $20X$ faster than the direct method. It is also demonstrated that when utilizing a four-core-four-GPU system, a grid with eight million nodes can be solved within about one second.

CHAPTER IV

CONCLUSION AND FUTURE WORK

A. Conclusion of the dissertation

This dissertation presents methodologies for modeling and analysis of large scale on-chip interconnect networks. Two major circuit modeling and analysis issues have been covered:

1. Statistical parameter reduction methods have been proposed to facilitate modeling and analyzing VLSI circuits that are impacted by high dimensional parameter space: (a) Very compact parametric interconnect models can be obtained more efficiently than before; (b) Statistical design-dependent interconnect performance corners can be extracted in a more economic way; (c) Prior second order statistical static timing analysis algorithm has been extended to capture more local variation sources during the analysis.

2. A graphics processing unit (GPU) accelerated multigrid algorithm has been proposed to efficiently solve the very large scale power supply network. Much attention has been paid to the GPU algorithm design and implementation. The proposed approach starts by solving an approximate 2D regular grid that is close to the original 3D irregular grid using a novel GPU accelerated geometric multigrid solver. The error components introduced by the grid approximation step can be effectively damped out through the CPU-GPU iterative hybrid multigrid procedure. The GPU based multigrid solver exhibits very promising performances on various benchmark circuits, which is typically more than 100X faster than the state-of-art matrix solver CHOLMOD and 20X more memory efficient. Several key algorithm development issues have also been discussed.

B. Future Work

There are a few interesting topics to be investigated in the future, which can be briefly described as follows.

a. Modeling and Simulation of Complex Large-scale Systems

In the future, I plan to propose new methodologies and strategies, especially for the new microarchitectures, to better address the large-scale modeling and simulation challenges in VLSI design. Primary research efforts will be focused on developing efficient simulation and optimization methods for modern power supply network (PDN) designs, particularly for the low-power multi-core PDNs which involve complex power gating activities. The strategy of power gating is to switch between the low power modes and the active modes, at the appropriate time and in the appropriate manner to maximize power savings while minimizing the impact to performance, which has been widely adopted for nowadays multi-core processor designs. A main challenge in power gating design is how to efficiently and accurately assess the tradeoffs between the amount of leakage power savings, the entry/exit time penalties incurred, the energy dissipated during entering/leaving such leakage saving modes and the activity profile (assignment of asleep or active times), requiring a huge number of PDN simulation runs. However, the bottleneck of simulation speed of multi-million node systems makes power gating designs quite time-consuming with existing simulation methods. I plan to extend my recent research studies [75] to provide a highly efficient yet accurate hardware-accelerated PDN simulation strategy with promising speedups (potentially 100X faster than the traditional simulation methods).

As a following step, I will also work on developing new 3D integrated circuit (IC) thermal analysis and verification methods. With the ever-increasing power densities,

IC designs are experiencing bigger temperature variations across the chip [76]. Full-chip thermal analysis is quite important since it provides useful insights for revealing hotspots, selecting the appropriate packaging technologies and avoiding excessive temperature variations. Furthermore, full-chip thermal simulation results can be used for more accurate power, timing and electromigration simulations. However, full-chip thermal analysis down to the device and interconnect levels is very computational expensive in that 3D dense fine-grained mesh structure (with up to hundreds of millions of unknowns) has to been solved as a whole, resulting in excessively long runtime and huge memory consumption. I plan to develop much more efficient hardware acceleration algorithms to analyze the three-dimensional chip thermal effects, by extending my prior work [75]. Efficient 3D iterative algorithms will be investigated to properly handle the inhomogeneous multi-layer structure. In the long-term, this thermal solver can be integrated into our PDN analysis engine, to facilitate the dynamic thermal management for reliable VLSI system design and verification.

I also feel well prepared to conduct research on fast algorithms for addressing emerging modeling and simulation challenges. One immediate goal is to accelerate the optical lithography simulations. Optical lithography simulation is essential to enable the profitable continuation of Moores Law, which can assist with improving device yields and reducing the number of design iterations, allowing a fabrication house to make profits faster and save substantially in production costs. However, full-chip lithography simulation may takes many days with existing simulation methodologies. I plan to study how the geometries can be efficiently stored and processed [77], like graphics pixels on graphics processing unit (GPU), and the lithography simulation can be performed in a massively parallel manner (using hundreds or thousands of cores of GPU).

b. Variability Modeling and Analysis

I plan to continue with my previous research topics on statistical VLSI circuit modeling techniques [78, 60, 31], to include more complex statistical parameter model uncertainties. Process variation data can only be obtained from foundries of IC manufacturing companies, through specific test structure designs and extensive measurements across different dies. However, due to the very limited measurements (test structures) and complex process characterization fluctuations, even the statistical process variation data obtained from foundries may not be accurate enough. It is desirable to build statistical circuit models capturing the statistical model uncertainties. To achieve this goal, I would like to study the impacts of parameter uncertainties on the parameter dimension reduction algorithms [79] for a variety of VLSI modeling problems (digital, analog and RF circuit modelings). Advanced matrix perturbation theories can be adopted to analyze the reduced parameter sets obtained from uncertain statistical parameters. Nonlinear optimization methods can be also applied to find the upper and lower bounds of the reduced parameter sets due to the parameter uncertainties [31, 79, 80]. With the above modeling framework, VLSI designs with hundreds or thousands of process parameters can be accurately verified through a feasible means. The final goal is to make the chip-level yield analysis and optimizations more efficient and effective than ever before. In the long-term, I plan to integrate the dynamic power-thermal simulation engine into the variation-aware VLSI modeling and analysis flow, making the process-voltage-temperature (PVT) aware yield analysis unprecedented efficient and accurate, which is indispensable for nanometer VLSI design.

c. PVT-Aware VLSI Design and Optimization

The modeling and simulation methods always serve for practical circuit designs and optimizations. In the long-term, therefore, I hope to conduct research on VLSI design optimizations that take into account the complex power, thermal and process variation effects. Prior researches on VLSI optimizations usually use rather coarse-grained (approximate) modeling approaches to avoid long simulation time. On the other hand, my ultimate goal is to deliver practical optimization solutions based on our accurate yet efficient PVT modeling and simulation methods. I believe my background in VLSI design and numerical methods prepares me well for this challenge.

Looking forward, I am interested in conducting advanced researches in computational techniques for the simulation of a large variety of large-scale engineering and physical systems, and applying numerical techniques to the simulation and modeling of emerging and interdisciplinary technologies (photonic systems, nanoelectronics, biological sequencing, numerical weather prediction, etc).

REFERENCES

[1] S. R. Nassif, "Modeling and analysis of manufacturing variations," in *Proc. IEEE CICC*, 2001, pp. 223–228.

[2] K. K. Low and S. W. Director, "An efficient methodology for building macromodels of IC fabrication processes," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 8, no. 12, pp. 1299–1313, December 1989.

[3] C. Chao and L. Milor, "Performance modeling using additive regression splines," *IEEE Trans. on Semicond. Manuf.*, vol. 8, no. 3, pp. 239–251, August 1995.

[4] Y. Liu, L. T .Pileggi, and A. Strojwas, "Model order-reduction of RC(L) interconnect including variational analysis," in *Proc. IEEE/ACM DAC*, 1999, pp. 201–206.

[5] P. Heydari and M. Pedram, "Model reduction of variable-geometry interconnects using variational spectrally-weighted balanced truncation," in *Proc. IEEE/ACM ICCAD*, 2001, pp. 586–591.

[6] J. Phillips, "Variational interconnect analysis via PMTBR.," in *Proc. IEEE/ACM ICCAD*, 2004, pp. 872–879.

[7] J. Wang, P. Ghanta, and S. Vrudhula, "Stochastic analysis of interconnect performance in the presence of process variations," in *Proc. IEEE/ACM ICCAD*, 2004, pp. 880–886.

[8] L. Daniel, O. Siong, L. Chay, K. Lee, and J. White, "A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 5, pp. 678–693, May 2004.

[9] P. Li, F. Liu, X. Li, L. Pileggi, and S. Nassif, "Modeling interconnect variability using efficient parametric model order reduction," in *Proc. IEEE/ACM DATE*, 2005, pp. 958–963.

[10] X. Li, P. Li, and L. Pileggi, "Parameterized interconnect order reduction with explicit-and-implicit multi-parameter moment matching for inter/intra-die variations," in *Proc. IEEE/ACM ICCAD*, 2005, pp. 806–812.

[11] H. Chang and S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 9, pp. 1467–1482, September 2005.

[12] D. Morrison, *Multivariate Statistical Methods*, New York: McGraw-Hill, 1976.

[13] Z. Li, X. Lu, and W. Shi, "Process variation dimension reduction based on SVD," in *Proc. IEEE ISCAS*, 2003, pp. 672–675.

[14] T. H. Chen and C. C.-P Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," in *Proc. IEEE/ACM DAC*, 2001, pp. 559–562.

[15] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," *IEEE Trans. on Computer-Aided Design*, vol. 21, no. 10, pp. 1148–1160, 2002.

[16] S. R. Nassif and J. Kozhaya, "Fast power grid simulation," in *Proc. IEEE/ACM DAC*, 2000, pp. 156–161.

[17] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "Multigrid-like technique for power grid analysis," in *Proc. IEEE/ACM ICCAD*, 2001, pp. 480–487.

[18] M. Zhao, R. Panda, S. S. Sapatnekar, and D. T. Blaauw, "Hierarchical analysis of power distribution networks," *IEEE Trans. on Computer-Aided Design*, vol. 21, no. 2, pp. 159–168, 2002.

[19] H. Su, E. Acar, and S. R. Nassif, "Power grid reduction based on algebraic multigrid principles," in *Proc. IEEE/ACM DAC*, 2003, pp. 109–112.

[20] Y. Zhong and M. D. F. Wong, "Fast algorithms for IR drop analysis in large power grid," in *Proc. IEEE/ACM ICCAD*, 2005, pp. 351–357.

[21] Y. Zhong and M. D. F. Wong, "Efficient second-order iterative methods for ir drop analysis in power grid," in *Proc. IEEE ASP-DAC*, 2007, pp. 768–773.

[22] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Power grid analysis using random walks," *IEEE Trans. on Computer-Aided Design*, vol. 24, no. 8, pp. 1204–1224, 2005.

[23] C. Zhuo, J. Hu, M. Zhao, and K. Chen, "Power grid analysis and optimization using algebraic multigrid," *IEEE Trans. on Computer-Aided Design*, vol. 27, no. 4, pp. 738–751, 2008.

[24] K. Sun, Q. Zhou, K. Mohanram, and D. C. Sorensen, "Parallel domain decomposition for simulation of large-scale power grids," in *Proc. IEEE/ACM ICCAD*, 2007, pp. 54–59.

[25] J. Shi, Y. Cai, S. Tan, J. Fan, and X. Hong, "Pattern-based iterative method for extreme large power/ground analysis," *IEEE Trans. on Computer-Aided Design*, vol. 26, no. 4, pp. 680–692, 2007.

[26] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Random walks in a supply network," *Proc. IEEE/ACM DAC*, vol. 24, no. 8, pp. 93–98, 2003.

[27] NVIDIA Corporation, *NVIDIA CUDA Programming Guide*, [Online]. Available: http://www.nvidia.com/object/cuda.html, 2007.

[28] G. Reinsel and R. Velu, *Multivariate Reduced-Rank Regression, Theory and Applications*, New York: Springer-Verlag, 1998.

[29] G. H. Golub and C. F. Van Loan, *Matrix Computations 3rd ed*, Baltimore: Johns Hopkins University Press, 1996.

[30] G. A. F. Seber and C. J. Wild, *Nonlinear Regression (Wiley Series in Probability and Statistics)*, New York: Wiley-Interscience, 2003.

[31] Z. Feng and P. Li, "Performance-oriented statistical parameter reduction of parameterized systems via reduced rank regression," in *Proc. IEEE/ACM ICCAD*, November 2006, pp. 868–875.

[32] X. Ye, F. Liu, and P. Li, "Fast variational interconnect delay and slew computation using quadratic models," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 8, pp. 913–926, August 2007.

[33] M. Stein, "Large sample properties of simulations using Latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, May 1987.

[34] S. Wong, T. Lee, D. Ma, and C. Chao, "An empirical three-dimensional crossover capacitance model for multilevel interconnect VLSI circuits," *IEEE Trans. Semicond. Manuf.*, vol. 13, no. 2, pp. 219–227, May 2000.

[35] X. Lu and W. Shi, *Layout and Parasitic Information for ISCAS Circuits*, [Online]. Available: http://dropzone.tamu.edu/ xiang/iscas.html, 2005.

[36] Z. Feng, P. Li, and Z. Ren, "SICE: Design-dependent statistical interconnect corner extraction under inter/intra-die variations," in *Proc. SRC Techcon Conference*, 2007, pp. 54–59.

[37] C. J. Alpert, A. Devgan, and C. V. Kashyap, "RC delay metrics for performance optimization," *IEEE Trans. on Computer-Aided Design*, vol. 20, no. 5, pp. 571–582, May 2001.

[38] Z. Ren, D. Petranovic, and J. Falbo, "Interconnect parasitics sensitivity for modeling and analysis of process variation in nanometer technology," in *VLSI Multilevel Interconnect Conference (VMIC)*, 2007, pp. 365–372.

[39] B. N. Sheehan, "TICER: realizable reduction of extracted RC circuits," in *Proc. IEEE/ACM ICCAD*, November 1999, pp. 200–203.

[40] C. S. Amin, C. V. Kashyap, N. Menezes, K. Killpack, and E. Chiprout, "A multiport current source model for multiple-input switching effects in cmos library cells," in *Proc. IEEE/ACM DAC*, 2006, pp. 247–252.

[41] P. Li, Z. Feng, and E. Acar, "Characterizing multistage nonlinear drivers and variability for accurate timing and noise analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 11, pp. 1205–1214, November 2007.

[42] P. Li and E. Acar, "A waveform independent gate model for accurate timing analysis," in *Proc. IEEE ICCD*, 2005, pp. 363–365.

[43] R. Myers and D. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments.*, New York:Wiley Interscience, 2002.

[44] F. Huebbers, A. Dasdan, and Y. I. Ismail, "Computation of accurate interconnect process parameter values for performance corners under process variations," in *Proc. IEEE/ACM DAC*, 2006, pp. 797–800.

[45] F. Huebbers, A. Dasdan, and Y. I. Ismail, "Multi-layer interconnect performance corners for variation-aware timing analysis," in *Proc. IEEE/ACM ICCAD*, 2007, pp. 713–718.

[46] M. Sengupta, S. Saxena, L. Daldoss, G. Kramer, S. Minehane, and J. Cheng, "Application-specific worst case corners using response surfaces and statistical models," *IEEE Trans. on Computer-Aided Design*, vol. 24, no. 9, pp. 1372–1380, September 2005.

[47] Y. Xu, K. Hsiung, X. Li, I. Nausieda, S. P. Boyd, and L. T. Pileggi, "OPERA: optimization with ellipsoidal uncertainty for robust analog IC design," in *Proc. IEEE/ACM DAC*, 2005, pp. 632–637.

[48] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *Proc. IEEE/ACM ICCAD*, November 2003, pp. 621–625.

[49] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Proc. IEEE/ACM ICCAD*, November 2003, pp. 900–907.

[50] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," in *Proc. IEEE/ACM ICCAD*, November 2003, pp. 607–614.

[51] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proc. IEEE/ACM*

DAC, June 2004, pp. 331–336.

[52] L. C. Wang, J. J. Liou, and K. T. Cheng, "Critical path selection for delay fault testing based upon a statistical timing," *IEEE Trans. on Computer-Aided Design*, vol. 23, no. 11, pp. 1550–1565, November 2004.

[53] V. Khandelwal, A. Davoodi, and A. Srivastava, "Efficient statistical timing analysis through error budgeting," in *Proc. IEEE/ACM ICCAD*, November 2004, pp. 473–477.

[54] H. Chang, V. Zolotov, S. Narayan, and C. Visweswariah, "Parameterized block-based statistical timing analysis with non-Gaussian parameters, nonlinear delay functions," in *Proc. IEEE/ACM DAC*, June 2005, pp. 71–76.

[55] Y. Zhan, A. Strojwas, X. Li, and L. Pileggi, "Correlation-aware statistical timing analysis with non-Guassian delay distributions," in *Proc. IEEE/ACM DAC*, June 2005, pp. 77–82.

[56] L. Zhang, W. Chen, Y. Hu, A. Gubner, and C. Chen, "Correlation-preserved non-Gaussian statistical timing analysis with quadratic timing model," in *Proc. IEEE/ACM DAC*, June 2005, pp. 83–88.

[57] L. Zhang, W. Chen, Y. Hu, J. A. Gubner, and C. C. Chen, "Correlation-preserved statistical timing with a quadratic form of Gaussian variables," *IEEE Trans. on Computer-Aided Design*, vol. 25, no. 1, pp. 2437–2449, January 2004.

[58] X. Yin and R. D. Cook, "Dimension reduction for the conditional k-th moment in regression," *Journal of the Royal Statistical Society B*, vol. 64, no. Part 2, pp. 159–175, 2002.

[59] X. Yin and E. Bura, "Moment based dimension reduction for multivariate response regression," *Journal of Statistical Planning and Inference*, vol. 136, no. 10, pp. 3675–3688, 2006.

[60] Z. Feng, P. Li, and Y. Zhan, "Fast second-order statistical static timing analysis using parameter dimension reduction," in *Proc. IEEE/ACM DAC*, June 2007, pp. 244 – 249.

[61] K. C. Li, "On principal Hessian directions for data visualization and dimension reduction: another application of Stein's lemma," *J. Ameri. Stat. Assoc.*, vol. 87, no. 420, pp. 1025–1039, December 1992.

[62] A. V. Mitev, M. Marefat, D. Ma, and J. M. Wang, "Principle Hessian direction based parameter reduction for interconnect networks with process variation," in *Proc. IEEE/ACM ICCAD*, November 2007, pp. 632 – 637.

[63] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W. W. Hwu, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *Proc. ACM PPOPP*, 2008, pp. 73–82.

[64] J. Bolz, I. Farmer, E. Grinspun, and P. Schroder, "Sparse matrix solvers on the GPU: conjugate gradients and multigrid," *ACM Trans. on Graphics*, vol. 22, no. 3, pp. 917–924, 2003.

[65] L. Buatois, G. Caumon, and B. Levy, "Concurrent number cruncher: An efficient sparse linear solver on the GPU," *High Performance Computing and Communications 2007, Lecture Notes in Computer Science*, vol. 4782/2007, pp. 358–371, 2007.

[66] S. R. Nassif, "Power grid analysis benchmarks," in *Proc. IEEE/ACM ASP-DAC*, 2008, pp. 376–381.

[67] S. R. Nassif, *IBM power grid benchmarks*, [Online]. Available: http://dropzone.tamu.edu/ pli/PGBench/, 2008.

[68] W. Briggs, *A multigrid tutorial*, Philadelphia:SIAM Publications, 1987.

[69] B. Barney, *POSIX Threads Programming*, [Online]. Available: www.llnl.gov/computing/tutorials/pthreads/, 2008.

[70] S. Pant and E. Chiprout, "Power grid physics and implications for CAD," in *Proc. IEEE/ACM DAC*, 2006, pp. 199–204.

[71] S. Pant, E. Chiprout, and D. Blaauw, "Power grid physics and implications for CAD," *IEEE Design & Test of Computers*, vol. 24, no. 3, pp. 246–254, 2007.

[72] T. Davis, *CHOLMOD: sparse supernodal Cholesky factorization and update/downdate*, [Online]. Available: http://www.cise.ufl.edu/research/sparse/cholmod/, 2008.

[73] L. Smith, R. Anderson, and T. Roy, "Chip-package resonance in core power supply structures for a high power microprocessor," in *Proc. of IPACK*, 2001.

[74] H. Zheng, B. Krauter, and L. T. Pileggi, "Electrical modeling of integrated-package power and ground distributions," *IEEE Design & Test of Computers*, vol. 20, no. 3, pp. 24–31, 2003.

[75] Z. Feng and P. Li, "Multigrid on GPU: Tackling power grid analysis on parallel SIMT platforms," in *Proc. IEEE/ACM ICCAD*, 2008, pp. 480–487.

[76] P. Li, L. Pileggi, M. Asheghi, and R. Chandra, "IC thermal simulation and modeling via efficient multigrid-based techniques," *IEEE Trans. on Computer-Aided Design*, vol. 25, no. 9, pp. 1763–1776, 2006.

[77] A. Lvov, C. Visweswariah, and U. Finkler, "Exact basic geometric operations on arbitrary angle polygons using only fixed size integer coordinates," in *Proc. IEEE/ACM ICCAD*, 2008, pp. 494–498.

[78] Z. Feng and P. Li, "A methodology for timing model characterization for statistical static timing analysis," in *Proc. IEEE/ACM ICCAD*, 2007, pp. 725–729.

[79] G. Yu, W. Dong, Z. Feng, and P. Li, "Statistical static timing analysis considering process variation model uncertainty," *IEEE Trans. on Computer-Aided Design*, vol. 27, no. 10, pp. 1880–1890, 2008.

[80] G. Yu, W. Dong, Z. Feng, and P. Li, "A framework for accounting for process model uncertainty in statistical static timing analysis," in *Proc. IEEE/ACM DAC*, 2007, pp. 829–834.

VITA

Zhuo Feng received the B.Eng. degree in information engineering from Xi'an Jiaotong University, Xi'an, China, in 2003 and the M.Eng. degree in electrical engineering from the National University of Singapore, Singapore, in 2005. He enrolled as a Ph.D. student in the Department of Electrical and Computer Engineering at Texas A&M University in the Fall of 2005, receiving his Ph.D. in December 2009. His research interests include hardware (e.g. Graphics Processing Unit) acceleration methods for circuit simulations, fast power grid analysis, model order reduction, statistical VLSI circuit modeling/analysis and statistical timing analysis.

His permanent address is: Department of Electrical and Computer Engineering, Texas A&M University, 214 Zachry Engineering Center, TAMU 3128 College Station, Texas 77843-3128.

The typist for this dissertation was Zhuo Feng.