UNIVERSITY OF CALIFORNIA
RIVERSIDE

Modeling, Characterization and Simulation of On-Chip Power Delivery Networks and Temperature Profile on Multi-Core Microprocessors

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Duo Li

December 2010

Dissertation Committee:

Dr. Sheldon X.-D. Tan, Chairperson
Dr. Yingbo Hua
Dr. Frank Vahid

The Dissertation of Duo Li is approved:

_____

_____

_____

Committee Chairperson

University of California, Riverside

ACKNOWLEDGMENT

There are many thanks to many people who made this dissertation possible.

First of all, I would like to thank my Ph.D. advisor Dr. Sheldon Tan for the continuous support to my Ph.D. study and research work. Thank him for guiding me on my research road and providing me the great lab research environment. I could not have finished my dissertation successfully without his encouragement, sound advice, good teaching and lots of good ideas.

Besides my advisor, I would like to thank the rest of my Ph.D. dissertation committee members Dr. Yingbo Hua and Dr. Frank Vahid, for their encouragement, comments and questions.

I would like to thank all my labmates for their support and encouragement. I could not have had the better understanding on my research without the frequent discussions with them.

I would like to thank my parents Daping Li and Yuqin Cong for giving birth to me, rasing me, teaching me, supporting me and loving me all the time.

Last but not the least, I would like to thank my lovely wife Shan Shan. Thanks for being with me together during my Ph.D. study in US.

ABSTRACT OF THE DISSERTATION

Modeling, Characterization and Simulation of On-Chip Power Delivery Networks and Temperature Profile on Multi-Core Microprocessors

by

Duo Li

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2010
Dr. Sheldon X.-D. Tan, Chairperson

Reliable on-chip power delivery is a challenging design task for sub-100nm and below VLSI technologies as voltage IR drops become more and more pronounced. This situation gets worse as technology continues to scale down. And efficient verification of power integrity becomes critical for design closure. In addition, the increasing process-induced variability makes it even worse for reliable power delivery networks. The process induced variations manifest themselves at different levels (wafer level, die-level and within a die) and they are caused by different sources (lithograph, materials, aging, etc.). In this dissertation, for power delivery networks without considering process variations, we propose an efficient simulation approach, called ETBR (Extended Truncated Balanced Realization), which uses MOR (Model Order Reduction) to speedup the simulation. To make ETBR more accuracy, we further introduce an error control mechanism into it. For power delivery networks with considering process variations, we propose varETBR (variational Extended Truncated Balanced Realization), a reduced Monte-Carlo simulation approach, which can handle a large number of variables and different variation distributions. To further speedup

the MOR process used in the fast simulation, a hierarchical Krylov subspace projection based MOR approach, hiePrimor, is proposed.

Besides the on-chip power delivery, excessive on-chip temperature has also become a first-tier design constraint as CMOS technology scales into the nanometer region. The exponential increase of power density of the high-performance microprocessors leads to the rapid rising of the average chip temperature. Higher temperature has significant adverse impacts on chip package cost, performance, and reliability. Multi-core techniques provide a viable solution to temperature/power problems. However, designing thermal efficient multi-core microprocessors remains a challenging problem as the temperature in each core can be dramatically different and the resulting large temperature gradients can produce mechanical stress and degrade the chip reliability. In this dissertation, we investigate a new architecture-level dynamic thermal characterization problem from a behavioral modeling perspective to address the emerging thermal related analysis and optimization problems for high-performance multi-core microprocessor design. We propose a new approach, called ThermPOF, to build the thermal behavioral models from the measured or simulated thermal and power information at the architecture level. And then we extend ThermPOF into ParThermPOF, a parameterized thermal behavioral modeling approach that can handle different parameters in multi-core microprocessor design and optimization.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations

1.1.1 Modeling and simulation of on-chip power delivery networks

Reliable on-chip power delivery is one of the difficult challenges for sub-100nm and below VLSI technology as voltage IR drops become more and more pronounced. This situation gets worse as technology continues to scale down. It has been reflected by the facts that more power has to be delivered into the chips for more packed devices, and supply voltage continues to decrease, which results in a decreased noise margin for signal transition, and makes transistor more vulnerable to supply voltage degradation. So efficient verification of power integrity becomes critical for final design power integrity closure.

The power delivery networks used in most of the research can be modeled as RC/RLC networks, as shown in Fig. 1.1 which is a part of large power grid networks. There are known time-variant current sources inside the power grid networks. Those current sources can be obtained by gate level logic simulations of the circuits. Some nodes having known

voltage are modeled as constant voltage sources.

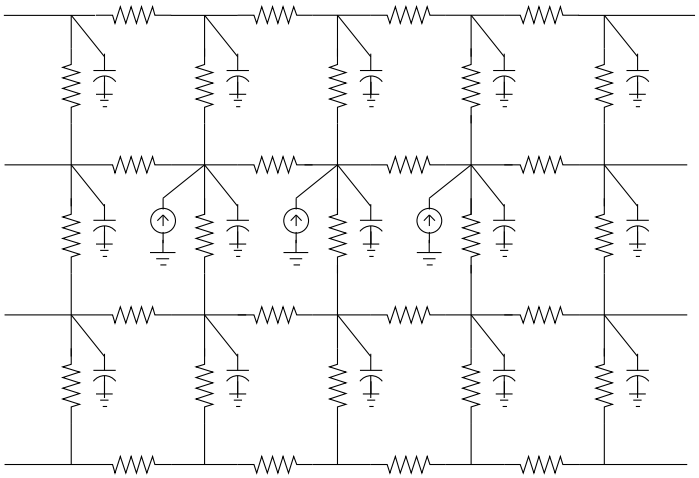


Figure 1.1: The power grid model.

Many research works have been done on efficient simulation of on-chip power grid networks. Methods such as multigrid-like [48, 73], hierarchical [83, 32], partition-based [37], fast iterative [9, 67] and random walk based [59] help improve scalability of power gird network analysis. Extended Krylov subspace based methods (EKS) [79, 32] uses both a power grid system and its input signals to reduce the original circuits before the simulation. Due to efficiency of Krylov subspace based reduction techniques, EKS can deal with very large power grid circuits.

Another issue for reliable on-chip power delivery is the increasing process-induced variability [62, 46]. The process induced variations manifest themselves at different levels (wafer level, die-level and within a die) and they are caused by different sources (lithograph, materials, aging etc) [10, 45]. One of the process variabilities comes from the voltage drop variations in on-chip power distribution networks. Voltage drop has significant impacts on the circuit timing [51]. The voltage drop of power grid networks subject to leakage current variations was first studied in [16, 17]. In [72, 50], impulse responses are used

to compute the mean and variances of node voltage responses caused by general current variations. Methods proposed in [20, 19] use orthogonal polynomial chaos expansion of random processes to represent and solve for the stochastic responses of linear systems. The methods have been improved by the StoEKS method [42, 41], where reduction is performed on the variational circuit matrices before the simulation.

1.1.2 Modeling and simulation of temperature profile on multi-core microprocessors

As CMOS technology is scaled into the nanometer region, the power density of high-performance microprocessors has increased drastically. The exponential power density increase will in turn lead to average chip temperature to raise rapidly [5]. Higher temperature has significant adverse impacts on chip package cost, performance, and reliability. Excessive on-chip temperature leads to slower transistor speed, more leakage power consumption, higher interconnect resistance, and reduced reliability [22, 6, 52].

Multi-Core techniques, where multiple CPU-cores and caches are integrated into a single chip, provide a viable solution to the temperature/power problems [38, 3, 4]. The architecture of the Intel Quad-Core microprocessor is shown in Fig. 1.2, where there are four CPU cores (die 0 to die 3) and one shared cache core (die 4). TIM here stands for thermal interface material.

Multi-Core processing allows one to increase the total throughput by task-level parallel computing with lower voltage and frequency to meet power and thermal constraints. The proliferation of this technique provides both opportunities and challenges for future massive parallel computing. One difficult issue confronting designers is the unpredictable heat and thermal effects, which are caused by the placement of cores and caches and changing

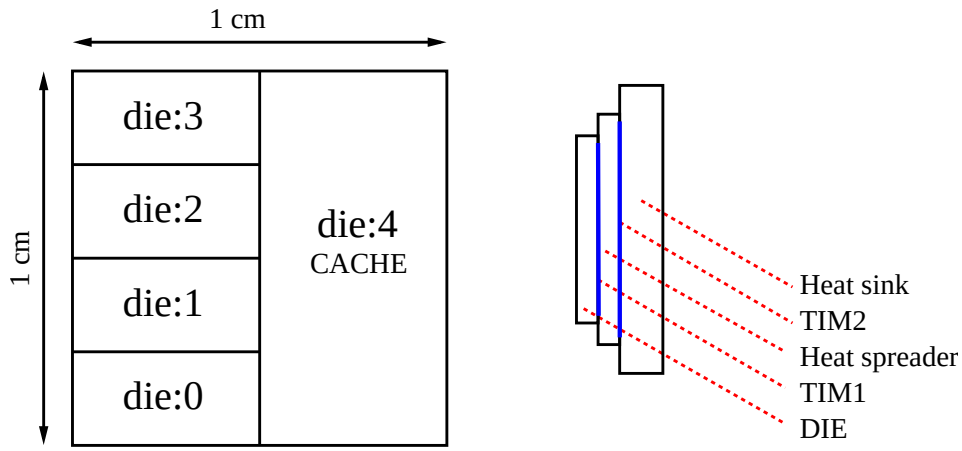


Figure 1.2: The quad-core architecture.

program loads. Furthermore, local hot spots, which may have much higher temperatures compared to the average die temperature, are becoming more prevalent in microprocessor chips [52]. This is especially the case for multicore processors as the temperature in each core can be dramatically different and the resulting large temperature gradients can produce mechanical stress and degrade the chip reliability. Hence it is very important to verify the temperatures and estimate the related performance (power, timing, yield) and reliability limits during the thermal-aware floorplanning and architecture design under various loads among different cores and caches [71].

To facilitate this temperature-aware architecture design, it is important to have accurate and fast thermal estimation at the architecture level [27]. Both architecture and CAD tool communities are currently lacking accurate and practical tools for thermal architecture modeling. Existing work on the HotSpot project [26, 71] tried to solve this problem by generating the architecture thermal model in a bottom-up way based on the floorplanning of the function units. But this method may not be accurate for real industry designs as many approximations are made during the modeling. It may also difficult to set up for new architectures with different thermal and packaging configurations [81]. To compute the

thermal responses by solving the basic thermal transfer equations using numerical methods like the finite element method, finite difference method is very expensive, especially for different thermal conditions and package configurations during the design stage. Hence, the need for efficient, accurate, and parameterized architecture thermal models, especially for emerging multicore microprocessors has never been greater.

1.2 Objectives and main results of this dissertation

The main objectives of this dissertation is to develop new modeling and simulation methods for on-chip power delivery networks and temperature profile on multi-core microprocessors. The major achievements accomplished in this dissertation as follows:

- A new model order reduction based simulation approach, called ETBR (Extended Truncated Balanced Realization), is proposed. In ETBR, both a system and its input signals are used to reduce the original circuit matrices. But different from the (improved) extended Krylov subspace methods, EKS/IEKS [79, 32], ETBR performs singular value decomposition (SVD) on response Gramian to reduce the original system while with the similar computation costs as EKS and a more accurate reduction framework: truncated balanced realization.

- We extend ETBR to ETBR_IR for efficient IR drop analysis based on the sampling-based reduction and simulation framework. ETBR_IR tries to dynamically compensate error losses from the reduction during the simulation process of reduced models. It introduces an error check mechanism based on the system residuals, which is an exact error indicator, as well as the novel effective resistance concept to compute the errors in terms of more useful voltage drop values.

- For large power grid network analysis considering process variations, a new scalable statistical simulation approach, called varETBR, is proposed To consider the variational parameters, we extend the concept of response Grammian proposed in ETBR to the variational response Grammian. Then Monte Carlo based numerical integration is employed to multiple-dimensional integrals. varETBR is very scalable for large networks with a large number of random variables.

- To speedup the MOR (Model Order Reduction) process used in the fast simulation, a hierarchical Krylov subspace projection based MOR approach, hiePrimor, is proposed. It combines the partitioning strategy and the Krylov subspace method to speed up the reduction process. hiePrimor is a very general hierarchical model order reduction technique and it works for general parasitic interconnect circuits modeled as RLC circuits.

- A new thermal behavioral modeling approach, called ThermPOF, is proposed for fast temperature estimation at the architecture level for multi-core microprocessors. ThermPOF builds the transfer function matrix from the measured or simulated thermal and power information. It improves generalized pencil-of-function (GPOF) method [24, 25, 64] to extract the poles and residues of the transfer functions. Further, the size of thermal models can be reduced by a Krylov subspace reduction method to speedup the simulation process [77]. ThermPOF is a top-down, black-box approach, meaning it does not require any internal structure of the systems and it is very general and flexible.

- We extend ThermPOF into ParThermPOF, a parameterized dynamic thermal behavioral modeling approach for emerging thermal-related analysis and optimization

problems for high-performance multi-core microprocessor design. ParThermPOF consists of two steps: first, a Response Surface Method (RSM) based on low-order polynomials is applied to build the parameterized models at each time point for all the given sampling nodes in the parameter space (except for time). Second, ThermPOF is employed to build the transfer-function-based models for each time-varying coefficient of the polynomials generated in the first step.

1.3 Organization of this dissertation

The organization of this dissertation is as follows: In Chapter 2, we first propose a new fast simulation method ETBR (Extended Truncated Balanced Realization) for on-chip power delivery network. To make ETBR more accurate, we then extend ETBR into ETBR_IR with an error control mechanism. For large power grid network analysis considering process variations, we propose a novel scalable statistical simulation approach, called varETBR, in Chapter 3. Then, in Chapter 4, we propose a new hierarchical Krylov subspace based reduction method, called hiePrimor, to speed up the traditional reduction process for RC/RLC circuits. In Chapter 5 and Chapter 6, we investigate a new architecture-level dynamic thermal characterization problem from a behavioral modeling perspective to address the emerging thermal related analysis and optimization problems for high-performance multi-core microprocessor design. We propose a new approach, called ThermPOF, to build the thermal behavioral models from the measured or simulated thermal and power information at the architecture level. And then we extend ThermPOF into ParThermPOF, a parameterized thermal behavioral modeling approach that can handle different parameters in multi-core microprocessor design and optimization. Finally Chapter7 concludes the dissertation.

Chapter 2

ETBR: Extended Truncated Balanced Realization for On-Chip Power Grid Network Analysis

In this chapter, we propose a novel model order reduction based simulation approach. This approach, called ETBR, performs singular value decomposition (SVD) on response Gramian to reduce the original system while with the similar computation costs of EKS/IEKS [79, 32]. ETBR is based on a more accurate reduction framework: truncated balanced realization, which was shown to be more accurate than Krylov subspace method used in EKS/IEKS method.

The proposed method is very amenable for threading-based parallel computing, as the response Gramian, which is used to construct the projection matrix, is computed in a Monte-Carlo-like sampling style and each sampling can be computed in parallel. This contrasts with all the Krylov subspace based methods like the EKS method, where moments have to be computed in a sequential order. The feature is important as the multi-core

architectures and multi-core computing are becoming commonplace [30, 70]. ETBR can naturally exploit task-level threading-oriented parallelism based on multicore architectures to significantly boost the simulation performance.ETBR also avoids the explicit moment representation of the input signals, which have well-known numerical problems in the past. Instead, it uses spectrum representation of input signals by fast Fourier transformation. As a result, ETBR is much more flexible for different types of input sources and can better capture the high frequency contents than EKS and this leads to more accurate results for fast changing input signals. Numerical examples, on a number of large RLC networks up to one million nodes, show that ETBR is indeed more accurate than the EKS/IEKS method especially for current sources rich in high-frequency components. ETBR also shows similar computational costs of EKS but smaller memory footprint in a single CPU, but is much faster than EKS when parallelism is explored.

Then we propose an efficient IR drop analysis approach, called ETBR_IR, based on the sampling-based reduction and simulation framework. The new approach tries to dynamically compensate error losses from the reduction during the simulation process of reduced models. ETBR_IR introduces an error check mechanism based on the system residuals, which is an exact error indicator, as well as the novel effective resistance concept to compute the errors in terms of more useful voltage drop values. The on-the-fly error reduction works well for compensating high frequency accuracy loss related to disruptive tap current waveforms in typical industry power grid networks. The new method also presents a new way to closely combine model order reduction and simulation to achieve the overall efficiency of simulation. The proposed method provide an efficient way to easily trade errors for speedup to suit different applications. Numerical examples show the proposed IR drop analysis method can significantly reduce the errors of the existing ETBR method,

and meanwhile it can lead to up 10X speedup over the the latest commercial power grid simulator, UPS, in UltraSim, with about 1-2% errors on a number of real industry circuits.

2.1 Power grid network models

The power grid networks in this chapter are modeled as RC networks with known time-variant current sources, which can be obtained by gate level logic simulations of the circuits. Fig. 2.1 shows the power grid models used in this chapter. For a power grid, some nodes having known voltage are modeled as constant voltage sources. For C4 power grids, the known voltage nodes can be internal nodes inside the power grid. Given the tap current source vector, $u(t)$, the node voltages can be obtained by solving the following differential equations, which is formulated using modified nodal analysis (MNA) approach,

$$\begin{aligned} Gv(t) + C\frac{dv(t)}{dt} &= Bu(t) \\ y(t) &= L^T v(t) \end{aligned} \tag{2.1}$$

where $G \in R^{n\times n}$ is the conductance matrix, $C \in R^{n\times n}$ is the matrix resulting from storage elements. $v(t)$ is the vector of time-varying node voltages and branch currents of voltage sources. $y(t)$ is the observed output voltage vector. $u(t)$ is the vector of independent power sources, and $B \in R^{n\times p}$ is the input selector matrix and $L \in R^{n\times l}$ is the output selector matrix. p and l are the number of input and output terminals respectively.

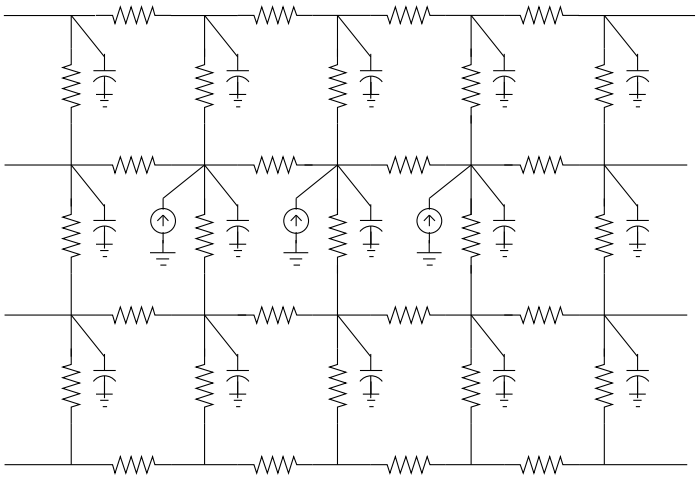


Figure 2.1: The power grid model used.

2.2 New extended balanced truncation method: ETBR

In this chapter, we propose an extended truncated balanced realization method, called ETBR, for efficient simulation of power grid networks. The new method features two improvements over existing approaches. First, the input signals are represented in its spectrum form in frequency domain directly by fast Fourier transformation. Second, fast balanced truncation method is used to perform the reduction, which has global accuracy [43, 56].

In the following, we first review the balanced truncation method and then the fast Gramian computation method.

2.2.1 Review of standard TBR

Given a system in a standard state-space form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2.2}$$

where $A \in R^{n\times n}$, $B \in R^{n\times p}$, $C \in R^{p\times n}$, $y(t)$, $u(t) \in R^p$. The controllable and observable Grammians are the unique symmetric positive definite solutions to the Lyapunov equations.

$$\begin{aligned} AX + XA^T + BB^T &= 0 \\ A^TY + YA + C^TC &= 0 \end{aligned} \tag{2.3}$$

Since the eigenvalues of the product XY are invariant under similarity transformation, we can perform a similarity transformation $(A_b = T^{-1}AT, B_b = T^{-1}B, C_b = CT)$ to diagonalize the product XY such that

$$T^{-1}XYT = \Sigma = diag({\sigma_1}^2, {\sigma_2}^2, \ldots, {\sigma_n}^2) \tag{2.4}$$

where the Hankel singular values of the system (σ_k), are arranged in a descending order. If we partition the matrices as

$$\begin{bmatrix} W_1^T \\ W_2^T \end{bmatrix} XY \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \tag{2.5}$$

where $\Sigma_1 = diag({\sigma_1}^2, {\sigma_2}^2, \ldots, {\sigma_r}^2)$ are the first r largest eigenvalues of Gramian product XY and W_1 and V_1 are corresponding eigenvectors. A reduced model can be obtained as follows

$$\begin{aligned} \dot{x}(t) &= A_r x(t) + B_r u(t) \\ y(t) &= C_r x(t) \end{aligned} \tag{2.6}$$

where $A_r = W_1^T A V_1$, $B_r = W_1^T B$, $C_r = CV_1$. The error in the transfer function of the order r approximation is bounded by $2\sum_{i=r+1}^{N} \sigma_k$. In the TBR procedure, the computational cost is dominated by solving Lyapunov equations $O(n^3)$, which makes it too expensive to

apply to integrated circuits problems and thus an efficient Gramian approximation technique is highly appreciated.

2.2.2 Review of fast TBR method: Poor man's TBR

Existing Gramian approximation technique, PMTBR [55], is restricted to a state-space model (2.2) with $A = A^T$ and $C = B^T$. This is the case for RC and RL circuits. In this symmetric case, it is easy to see that, both Gramians are equal and are obtained by solving the Lyapunov equation

$$AX + XA^T + BB^T = 0 \tag{2.7}$$

Since X is symmetric, it is orthogonally diagonalizable, i.e., there exists $T^{-1} = T^T$ such that $T^T XT = \Sigma$. Then, we have

$$T^T XXT = (T^T XT)(T^T XT) = (\Sigma)^2 \tag{2.8}$$

which means, in this symmetric case, the eigenspace of Gramian product XX is exactly the eigenspace of each X and we only need to find the dominant invariant subspace of an approximated Gramian $\hat{X}$. In frequency domain, the Gramian X can also be computed from the expression

$$X = \int_{-\infty}^{+\infty} (j\omega I - A)^{-1} BB^T (j\omega I - A)^{-H} d\omega \tag{2.9}$$

where superscript H denotes Hermitian transpose. Let ω_k be kth sampling point. If we define

$$z_k = (j\omega_k I - A)^{-1} B \tag{2.10}$$

then X can be approximated as

$$\hat{X} = \sum w_k z_k z_k^H = ZW^2 Z^H \tag{2.11}$$

where $Z = [z_1, z_2, \ldots, z_n]$. W a diagonal matrix with diagonal entries $w_{kk} = \sqrt{w_k}$. w_k comes from a specific numerical quadrature method. Since $\hat{X}$ is symmetric, it is orthogonally diagonalizable.

$$\hat{V}^T \hat{X} \hat{V} = \begin{bmatrix} \hat{V}_1^T \\ \hat{V}_2^T \end{bmatrix} \hat{X} \begin{bmatrix} \hat{V}_1 & \hat{V}_2 \end{bmatrix} = \begin{bmatrix} \hat{\Sigma}_1 & 0 \\ 0 & \hat{\Sigma}_2 \end{bmatrix} \tag{2.12}$$

where $\hat{V}^T \hat{V} = I$. $\hat{V}$ converges to the eigenspaces of X and the dominant eigenvectors $\hat{V}_1$ can be used as the projection matrix in a model reduction approach $(A_r = \hat{V}_1^T A \hat{V}_1, B_r = \hat{V}_1^T B)$.

2.2.3 Response Gramian and fast computation method

Follow the similar strategy of EKS method, we consider the input signals of the system into TBR based reduction framework so that efficient reduction can be done by converting an MIMO system into an SIMO system.

For a linear system in (2.1), we first define the frequency-domain *Response Gramian*,

$$X_r = \int_{-\infty}^{+\infty} (j\omega C + G)^{-1} B u(j\omega) u^T(j\omega) B^T (j\omega C + G)^{-H} d\omega \tag{2.13}$$

which is different from the Gramian concepts in the traditional TBR based reduction framework. Notice that in the new Gramian definition, the input signals $u(j\omega)$ is considered. As

a results, $(j\omega C + G)^{-1}Bu(j\omega)$ actually is the system response with respect to the input signal $u(j\omega)$ and resulting X_r becomes response Gramian.

To fast compute the response gramian X_r, which essential essentially one-dimensional integral with respect to the complex frequency ω. We can use Monte-Carlo-based method to estimate the numerical value as done in [53]. Specifically, let ω_k be kth sampling point over the frequency range. If we further define

$$z_k^r = (j\omega_k C + G)^{-1}Bu(j\omega_k) \tag{2.14}$$

then $\hat{X}$ can be computed approximately by numerical quadrature methods

$$\hat{X}_r = \sum_k w_k z_k^r z_k^{rH} = Z_r W^2 Z_r^H \tag{2.15}$$

where Z_r is a matrix whose columns are z_k^r and W a diagonal matrix with diagonal entries $w_{kk} = \sqrt{w_k}$. w_k comes from a specific quadrature method.

For the truncated balanced based reduction, we need to compute the eigven-composition of $\hat{X}_r$ to obtain the projection matrix, which consists of eigen vectors of $\hat{X}_r$. Since the approximate Gramian $\hat{X}_r$ is symmetric, we can obtain the project matrix by singular value decomposition of Z_r. To see this, if we have SVD of $Z_r = V_r S_r U_r^T$, then we can have the eigen decomposition of $\hat{X}_r = V_r S_r^2 V_r^T$. After this, we can reduce the original matrices into small ones and then perform the transient analysis on the reduced circuit matrices.

Also we find that weights w_k are not important for the SVD process. The weight matrix W will not change the subspace of Z_r as it simple multiplies each vector in Z_r with a constant. In our algorithm, we just simple ignore the weights and we use simple linear or logarithmic sampling methods to perform the sampling (to be discussed later).

Notice that we need frequency response of input signal $u(j\omega_k)$ in (2.14). This can be obtained by fast Fourier transformation on the input signals in time domain.

2.2.4 Extended truncated balanced realization method: ETBR

In this subsection, we give the algorithm flow of the proposed ETBR method, which is summarized in *Algorithm* 1.

Algorithm 1: Extended Truncated Balanced Realization method (ETBR)

Input: Circuit of G, C, B, $u(t)$, number of samples: q, transient simulation step interval.
Output: Node voltage responses $v(t)$ for the given simulation interval.

---

1. Convert all the input signals $u(t)$ into $u(s)$ using FFT.
2. Select q frequency points $s_1, s_2, \ldots, s_q$ over the frequency range
3. Compute $z_k^r = (s_k C + G)^{-1} B u(s_k)$
4. Form the matrix $Z_r = [z_1^r, z_2^r, \ldots, z_q^r]$
5. Perform SVD on Z_r, $Z_r = V_r S_r U_r^T$
6. $\hat{G} = V_r^T G V_r$,$\hat{C} = V_r^T C V_r$, $\hat{B} = V_r^T B$
7. Simulate ($\hat{G}, \hat{C}, \hat{B}, u(t)$)
8. Obtain the original waveforms $v(t) = V_r \hat{v(t)}$

Note that, like the EKS method, we use congruence transformation for the reduction process with orthogonal columns in the projection matrix (using Arnoldi or Arnoldi-like process), the reduced system must be stable. As far as simulation is concerned, this is good enough. If all the observable ports are also the current source nodes, i.e. $y(t) = B^T v(t)$, where $y(t)$ is the voltage vector at all observable ports, the reduced system is passive.

Compared with the existing approaches like EKS/IKES methods, ETBR shows several advantages and features. First ETBR method is much more amenable for parallel computing than EKS as each z_i^k in (2.14) can be computed in parallel. Thus ETBR is more efficient than EKS when the threading-based parallel computing is explored as shown in

the Numerical examples. Second, it is more accurate over wide band frequency ranges due to the global samplings. Third, it avoids the explicit moment representation of the input signals, which can lead more accurate results than the EKS method when signals are rich in high frequency components. ETBR can deal with any type of time-domain and frequency-domain input signals. While the EKS method can only deal with input signals in piecewise linear form.

2.2.5 Time complexity analysis and comparison

In this subsection, we analyze the computational costs for both ETBR and EKS and compare with the EKS methods.

In ETBR, there are two major computing steps, sampling and SVD. Let's look at the cost of each step. For sampling, we basically need to solve the (2.14) q times. Typically solving a $n \times n$ linear matrix takes $O(n^\beta)$ (typically, $1.1 \leq \beta \leq 1.5$ for matrix factorizations and $O(n^\alpha)$ (typically, $1 \leq \alpha \leq 1.2$) for solving (forward and backward substitutions). So the time complexity for this step is $O(qn^\beta + qn^\alpha)$. For the second step, the singular value decomposition (SVD) will take $O(nq^2)$ for a $n \times q$ matrix. Another computing cost comes from converting the input signals into the frequency spectrum form. Assume that we have m current sources, the samplings we use for the FFT is l, FFT takes $O(l\text{log}l)$ to finish. Hence the cost associated with input signals is $O(ml\text{log}l)$. Typically we set $l = 128$, which gives sufficient accuracy. So the total computational cost of ETBR is

$$O(qn^\beta + qn^\alpha + nq^2 + ml\text{log}l). \tag{2.16}$$

If all the sampling can be computed in parallel, computing (2.14) will become $O(n^\beta + n^\alpha)$

assuming very small overheads incurred to manage the threads. The total computational cost of ETBR will become

$$O(n^{\beta} + n^{\alpha} + nq^2 + ml\log l). \tag{2.17}$$

For one-point (expanded at one frequency point) EKS, it also have two major computing costs: compute the response moments and orthonormalize them similar to the QR decomposition. For the first step, it will take one matrix factorization and q steps solving (forward and backward substitutions). The computing cost is $O(qn^{\alpha} + n^{\beta})$, where $O(qn^{\alpha})$ (typically, $1 \leq \alpha \leq 1.2$ for sparse circuits) is q step solving. The orthonormalization will take about $O(nq^2)$ to finish. Again, we need to calculate the computing cost for transforming the input signals into the moment form. It can be shown that the computing cost is $O(qk^2m)$ [32], where k is the number of piecewise segments in each current sources, and m the number of current sources. Hence the final computational cost for EKS is

$$O(qn^{\alpha} + n^{\beta} + nq^2 + qk^2m) \tag{2.18}$$

It can be seen that EKS will be more efficient due to smaller number of factorizations in a single CPU. But if parallel computing is allowed, ETBR become much better. But if iteration solvers are used, which are typically more fast and memory efficient than the LU-based direct solvers for RLC networks [63], both approaches will have the same computational costs in a single CPU.

2.2.6 Statistical point of view

The proposed method in a sense can be viewed as special SVD-based principal component analysis (PCA) method used in statistical variable reduction transformation.

For a linear dynamic system formulated in state space equations (MNA) in (2.1), if complex frequency s is a vector of random variables with uniform distribution in the frequency domain. Then the state responses $V(s) = (G + sC)^{-1}Bu(s)$ become random variables in frequency domain. Its covariance matrix can be computed as

$$E\{V(s)V(s)^T\} = X_r \tag{2.19}$$

where $E\{x\}$ stands for computing the mean of random variable x. X_r is defined in (2.13). The response Gramian essentially can be viewed as the covariance matrix associated with state responses. ETBR procedure performs the principal component analysis transformation of the mentioned random process with a uniform distribution.

2.3 Numerical examples of ETBR

The proposed *ETBR* algorithm has been implemented using MATLAB 7.0 and tested on an Intel Xeon 3.0GHz dual CPU workstation with 2GB memory and an Intel quad-core 3.0GHz CPU workstation with 16GB memory. All the test circuits are randomly generated RC or RLC power grid networks up to one million nodes (R on the order of Ω, C on the order of pF and L on the order of pH), as shown in Table 2.1. Efficient matrix computations benefit from sparse matrix structure and a parser implemented by Python.

To solve circuits with one million nodes in MATLAB, an external linear solver package

Table 2.1: Test circuits

Test Ckts	#Nodes	#Sources
Ckt1	$1,000$	100
Ckt2	$10,000$	100
Ckt3	$10,000$	$1,000$
Ckt4	$100,000$	$1,000$
Ckt5	$100,000$	$4,000$
Ckt6	$500,000$	$5,000$
Ckt7	$500,000$	$20,000$
Ckt8	$1,000,000$	$50,000$
Ckt9 (RLC)	$6,000$	100
Ckt10 (RLC)	$250,000$	100

UMFPACK [2] is used, which is linked with MATLAB using MATLAB mexFunction. For ETBR, we use a non-LU-decomposition solver in UMFPACK. While for EKS, the LU decomposition solver is used. The reason why we choose different solver for them is ETBR only solves one column in the right hand side, so LU decomposition may cost too much and cannot be reused in ETBR. While in EKS, LU decomposition can be reused to solve several columns in the right hand side, number of columns depending on the selected reduced order, so doing LU decomposition is an efficient way in EKS. We remark that the selection of solvers is the best for both ETBR and EKS. The comparison is more fair for them.

2.3.1 Comparison with the EKS method

In sequel, we will compare our ETBR with IEKS [32], first in accuracy and then in CPU times. In all the test cases, to make a fair comparison, the reduction order q is set to 6 for IEKS and the number of frequency samples used for ETBR is also set to 6. Note that for the RLC circuits Ckt1-Ckt8, the results are collected on an Intel dual CPU workstation

with 2GB memory, and for the RLC circuits Ckt9 and Ckt10, the results are collected on an Intel quad-core workstation with 16GB memory.

Fig. 2.2 shows the simulation results of ETBR and IEKS at the 200th node of Ckt2. The simulation errors compared with SPICE results are shown in Fig. 2.3. One of the input signal waveforms in both time domain and frequency domain is as shown in Fig. 2.4. Through Fig. 2.3, we can see that ETBR is more accurate than IEKS over the entire simulation time.

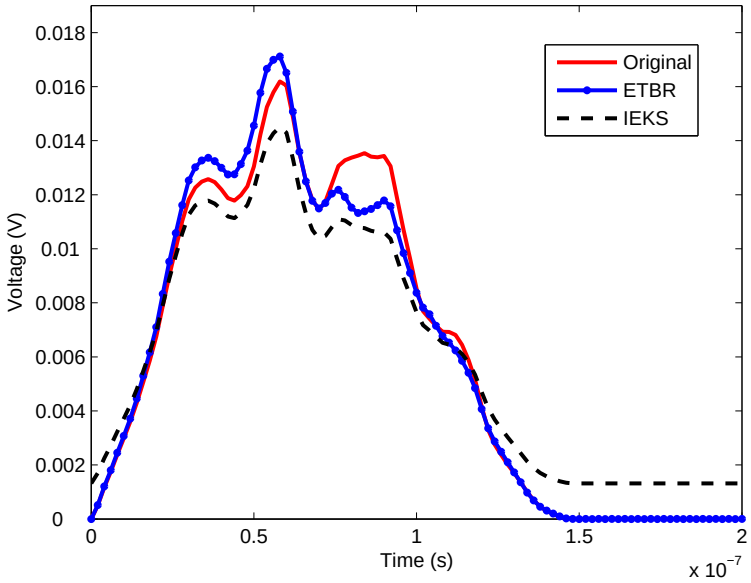


Figure 2.2: Transient waveform at the 200th node of Ckt2.

In the second testing case, we change the input signals so that they can have more fast changing spikes as shown in Fig. 2.7(a). In other words, current sources are rich in high-frequency components.

We find that ETBR's results are much better than EKS's as shown in Fig. 2.5. From the simulation errors comparison in Fig. 2.6, we can see that ETBR is almost $3\times$ more accurate than IEKS (the maximum error: ETBR 0.003 vs IEKS 0.01). This is not a surprise for us if we notice that the input signals shown in Fig. 2.7(b) have much more high frequency

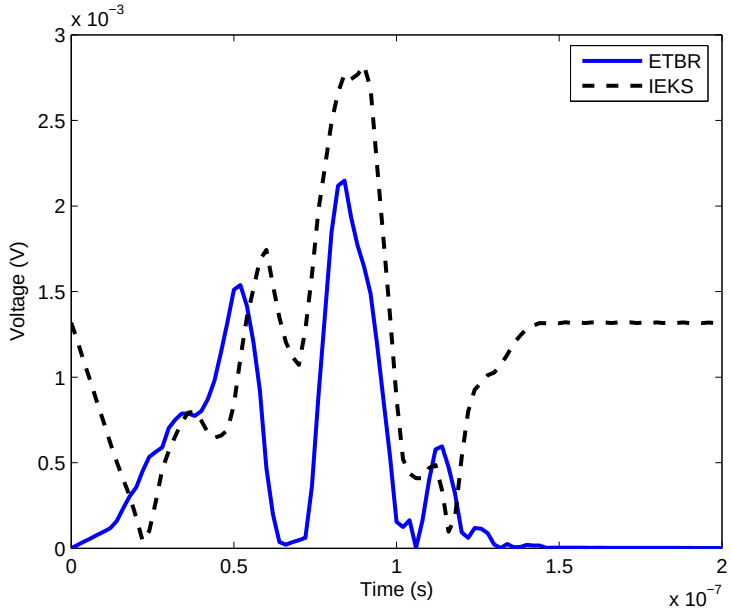
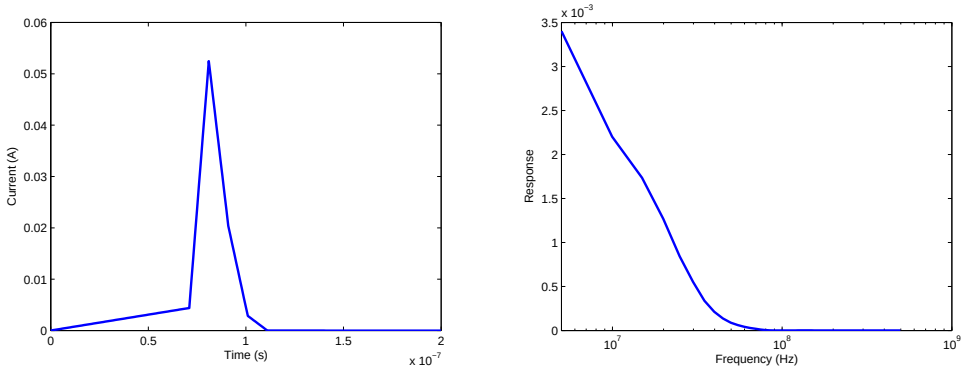


Figure 2.3: The simulation errors of ETBR and IEKS of Ckt2.



(a) In time domain (b) In frequency domain

Figure 2.4: Transient waveform at the 5th current source of Ckt2.

components from 10^7MHz to 10^8MHz than the input signals shown in Fig. 2.4(b).

We can try different reduced orders for Ckt2 to set $q = 5$ and $q = 7$. The results are shown in Fig. 2.8 and Fig. 2.9. We can see that ETBR is still more accurate than EKS as long as both of them use the same reduced order. And the CPU times of ETBR depends on the reduced order. If we want to achieve more accuracy, we need more reduced orders

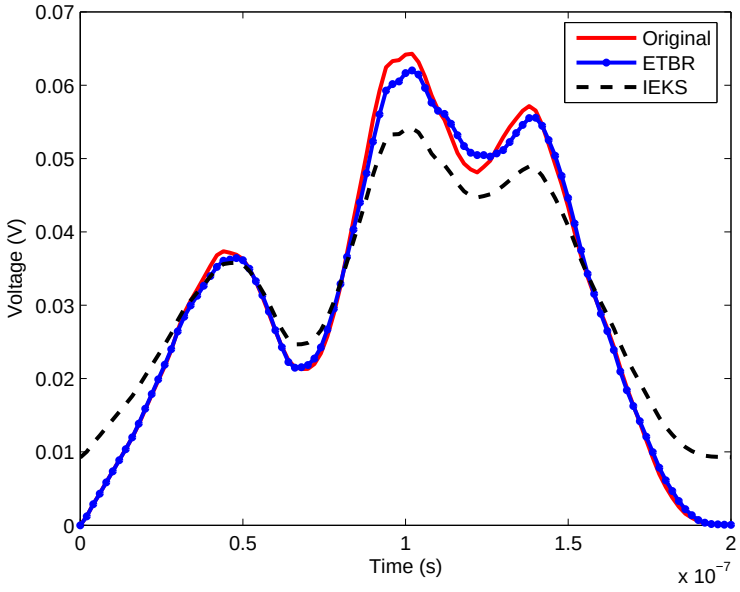


Figure 2.5: Transient waveform at the 200th node of Ckt2 with fast changing inputs.

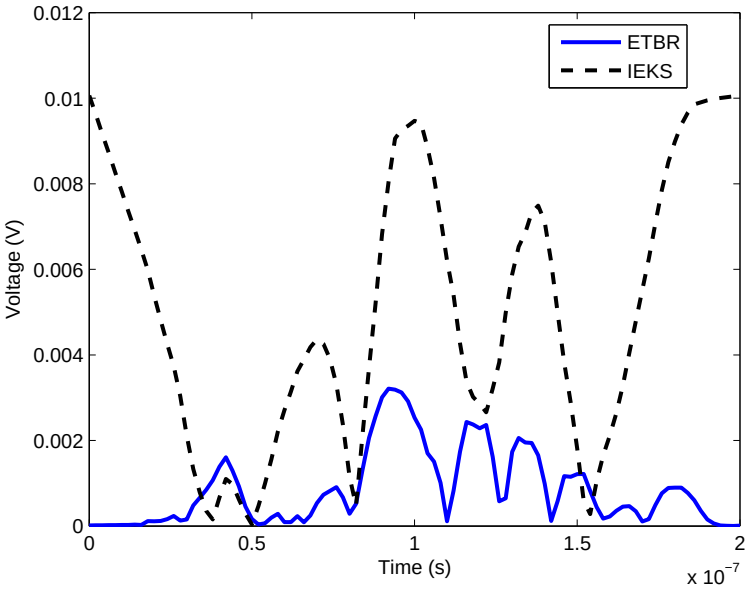
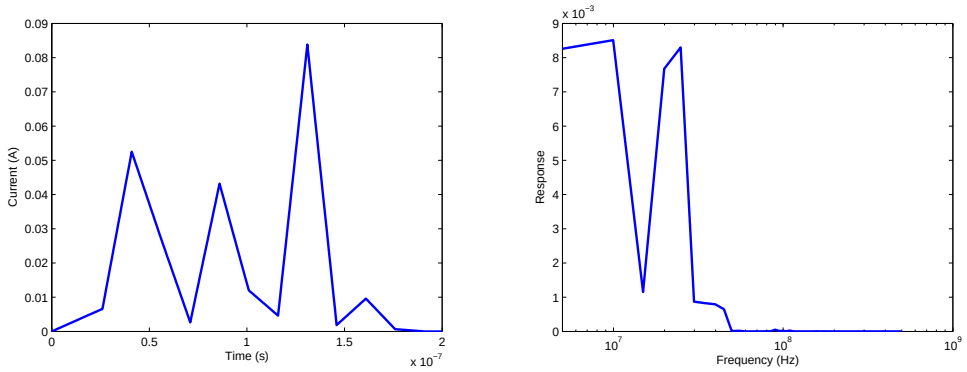


Figure 2.6: The simulation errors of ETBR and IEKS on Ckt2 with fast changing inputs.

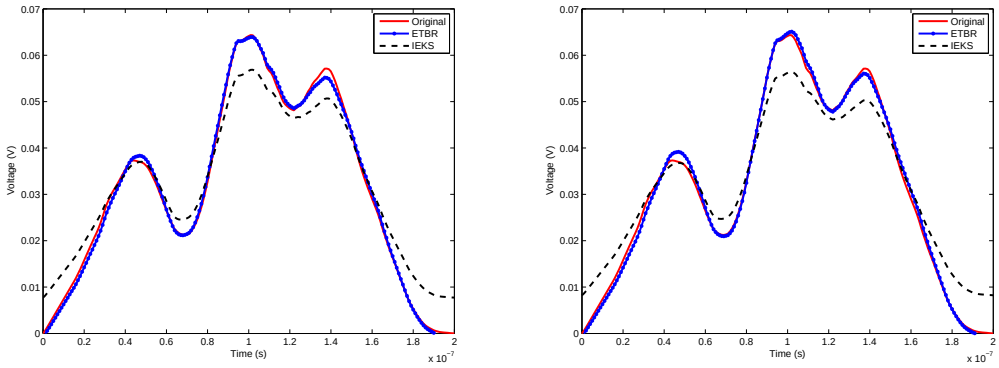
which results in more CPU times.

For the RLC circuits, ETBR also holds much more accuracy than EKS. Fig. 2.10 and Fig. 2.11 show the transient simulation waveforms and errors of both ETBR and EKS at



(a) In time domain (b) In frequency domain

Figure 2.7: The transient waveform at the 5th current source of Ckt2.

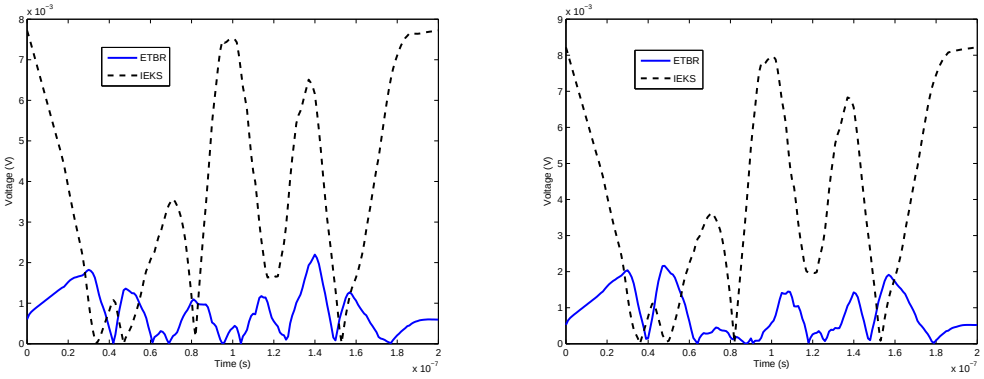


(a) $q = 5$ (b) $q = 7$

Figure 2.8: Transient waveform at the 200th node of Ckt2 with fast changing inputs.

the 200th node of Ckt9 (RLC).

There may be many high frequency components in the input signals in the real industry circuits. In this situation, we in general need more samplings to improve the accuracy. Now we perform ETBR on a real industry circuit of 154514 nodes, 624 current sources and 25001 simulation time steps. We also perform latest UltraSim UPS (power network solver) on the same case. UltraSim UPS is a commercial power grid simulator from Cadence



(a) $q = 5$ (b) $q = 7$

Figure 2.9: The simulation errors of ETBR and IEKS on Ckt2 with fast changing inputs.

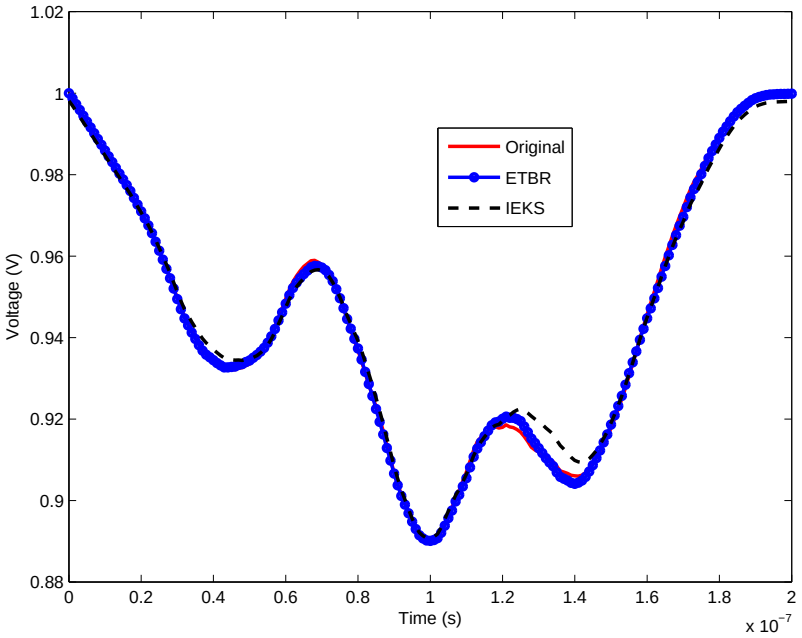


Figure 2.10: Transient waveform at the 200th node of Ckt9 (RLC).

and the results of UltraSim UPS are considered as golden in this chapter. The number of samplings in ETBR is still set to 10. But still the results are accurate enough as shown in Fig. 2.12. This circuit has rapid changing transient waveforms due to the reason that the

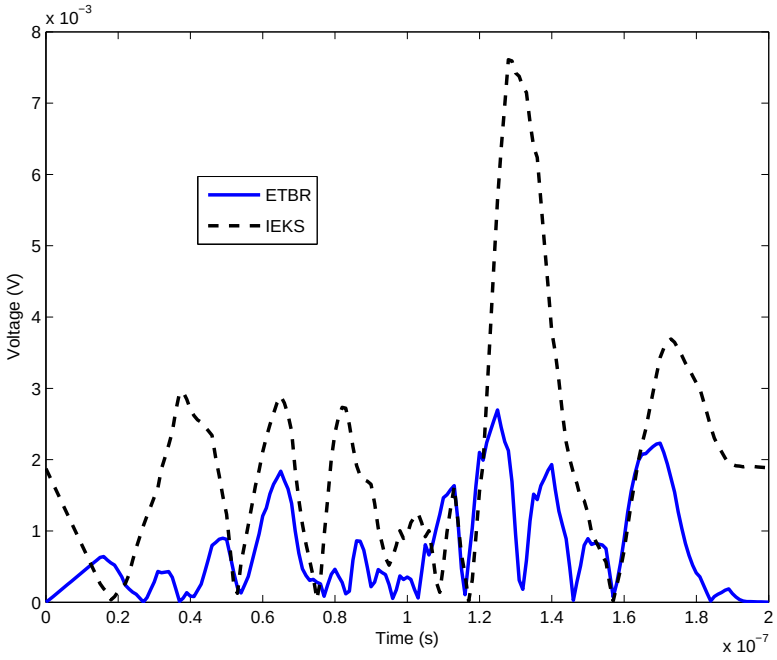


Figure 2.11: The simulation errors of ETBR and IEKS on Ckt9 (RLC).

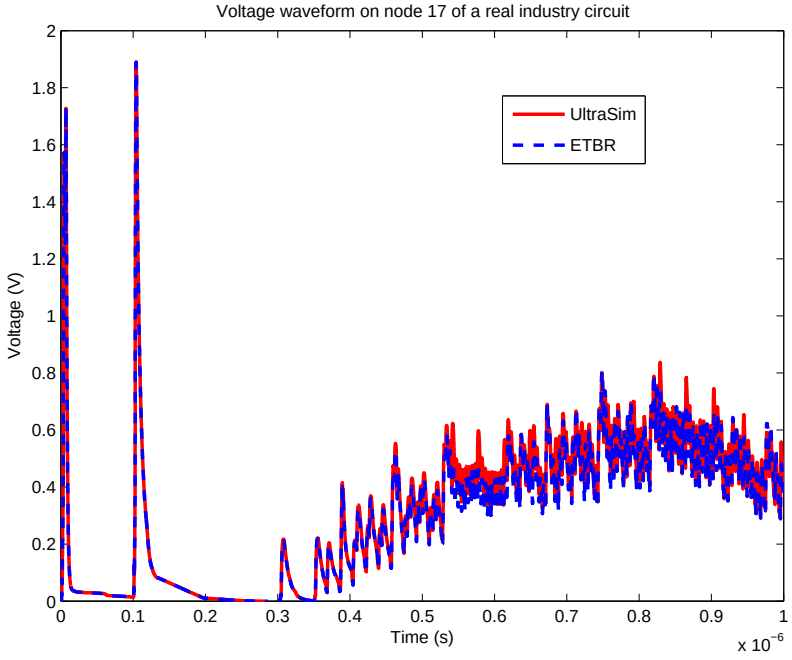


Figure 2.12: Transient waveform at node 17 of a real industry circuit.

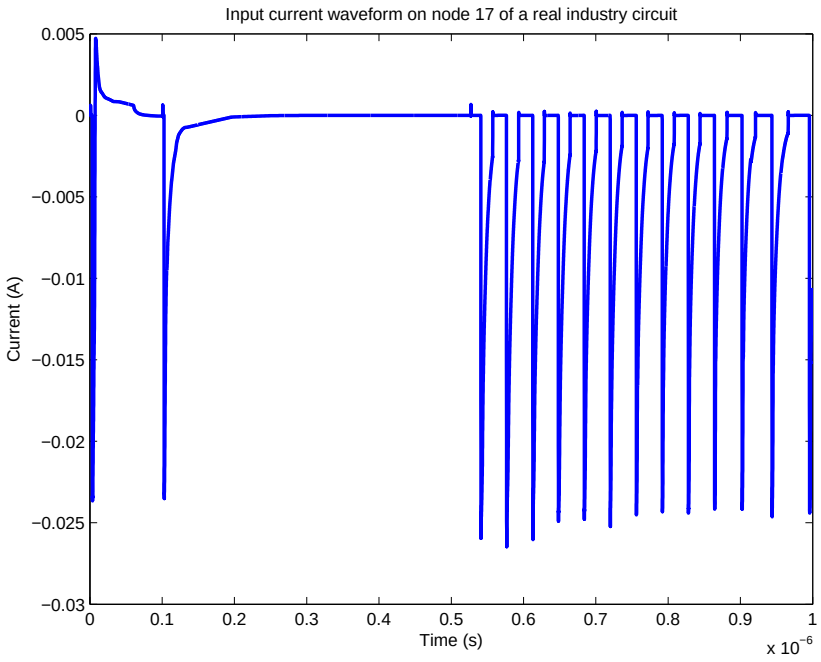


Figure 2.13: Input current waveform at node 17 of a real industry circuit.

current sources are changing very fast, as shown in Fig. 2.13.

Finally, we compare the CPU time of the two algorithms on a set of power grid networks up to one million nodes. The capacity of our implementation is mainly limited for Ckt1-Ckt8 by the physical memory of our machine (2GB).

Table 2.2 shows the CPU times of both ETBR (including the cost of FFT) and IEKS on the given set of circuits using the same reduction order $q = 6$. "$-$" means out-of-memory error. We find that EKS is a bit faster for small circuits. But for Ckt6 and larger circuits, the CPU times are almost the same for both methods. For the largest circuit Ckt8, EKS cannot even finish owning to the memory constraint; while ETBR runs through all the circuits. This clearly shows that ETBR is more memory efficient by using a non-LU decomposition solver than EKS.

Table 2.3 shows the CPU times if parallelism is explored in ETBT. *PETBR* means par-

Table 2.2: CPU times (in seconds) comparison of ETBR and IEKS ($q = 6$)

Test Ckts	ETBR (s)	EKS (s)
Ckt1	0.23	0.08
Ckt2	1.28	0.89
Ckt3	1.8	1.4
Ckt4	20.4	18.8
Ckt5	28.6	25.3
Ckt6	152	151
Ckt7	162	160
Ckt8	562	–
Ckt9 (RLC)	0.20	0.11
Ckt10 (RLC)	6.5	4.4

Table 2.3: CPU times (in seconds) comparison of ETBR, parallelized ETBR and IEKS ($q = 10$)

Test Ckts	# Nodes	# Sources	ETBR	PETBR	EKS
Ckt11	1,750,000	25,000	232	32	355
Ckt12	3,400,000	50,000	514	68	640
Ckt13	7,000,000	100,000	1349	167	–

allelized ETBR. The results are collected on an Intel quad-core (3GHz CPU) workstation with 16GB memory. We assume that Step 3 in *Algorithm* 1 can be fully parallelized. So the total CPU time of parallelized ETBR is the max CPU time out of all the sub-processes in parallelized Step 3 plus CPU time of serial parts in ETBR, such as FFT and SVD. For Krylov subspace method, such as EKS/IEKS, each moment is computed based on previous one, hence it is hardly to be parallelized. We can see that ETBR is now is a order of magnitude faster than EKS and ETBR.

2.3.2 Results on circuits with many different switching timings

In this subsection, we show the results of ETBR and EKS are also very accurate for power grid circuits excited by input currents with many different switching timings (peaks).

The used benchmark circuit has 1000 nodes and each node has a current source, which switches at a different time (the peaks are different for each of them) as shown in Fig. 2.14. The resistor and capacitor values of this circuit are randomly generated. R is on the order of $10^{-2}\Omega$, and C is on the order of 10^{-15}F. The capacitance is really small. Fig. 2.15 and Fig. 2.16 show the simulation waveforms and errors on 100th and 300th nodes ($q = 5$). We can see ETBR is still very accurate.

We remark that if the circuit has very small capacitances such that the whole circuit become DC with respect to their input signal spectrum, then both ETBR and model order reduction in general cannot be applied in this case. But this is a very unrealistic case for general interconnect circuits modeled as RLC/RLCK circuits.

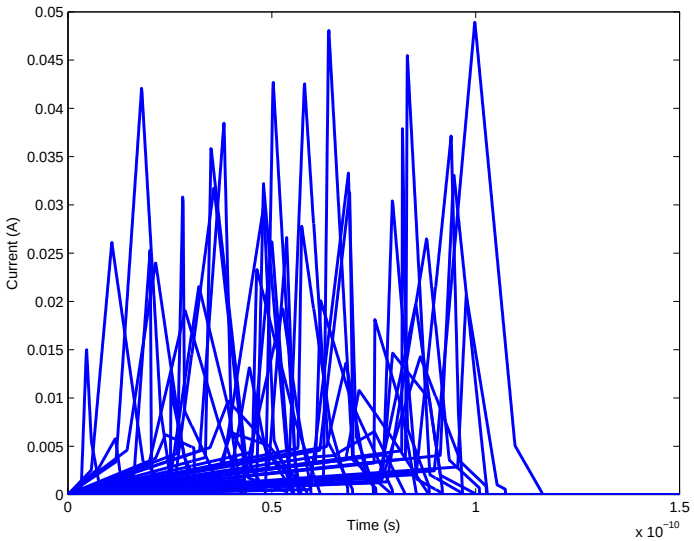
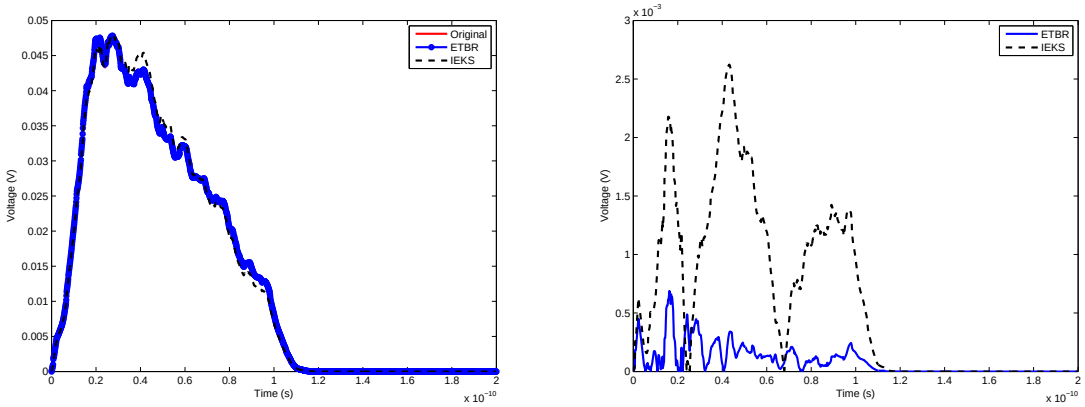
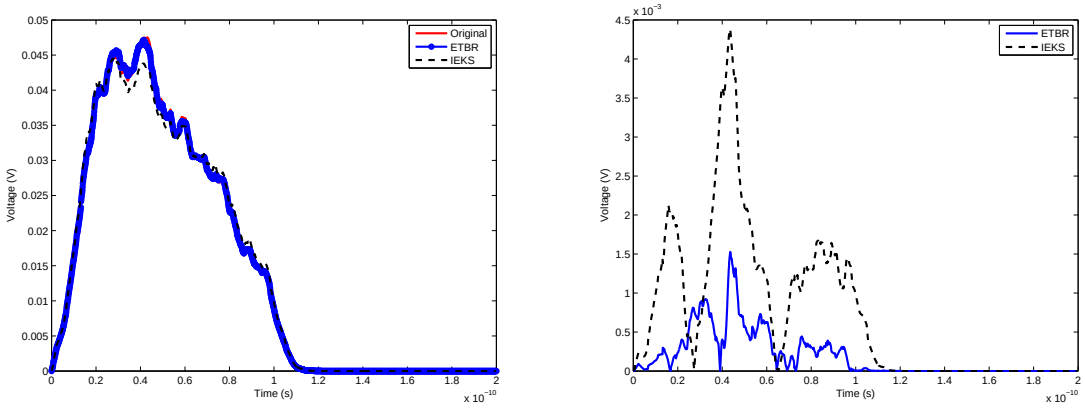


Figure 2.14: Transient waveforms of current sources switching at different time



(a) Transient waveform (b) Errors

Figure 2.15: The transient waveform and errors at the 100th node ($q = 5$).



(a) Transient waveform (b) Errors

Figure 2.16: The transient waveform and errors at the 300th node ($q = 5$).

2.4 IR drop analysis problem

The power grid networks in this charpter are modeled as RC networks with known time-variant tap current sources as shown in 2.1, which can be obtained by gate level logic simulations of the circuits under assumption that transistor circuit simulation and power grid network simulation are separated. Such RC model is still valid at least for the on-chip

level power grid networks for current technologies [47].

The on-chip power grids, one important integrity issue is excessive voltage IR drops due to the unavoidable wire resistance (and inductive effects when inductance are large). IR drop based power grid integrity analysis is different from the general transient analysis in that designers are mainly interested in the voltage drops in the tap current sources as the tap currents are where the power grid network are connected with the logic circuits and IR drops mainly matter from the logic circuit perspective. As a result in our program $L = B$ and $p = l$ in (2.1). This implies the passive model order reduction can be achieved and it will also lead to more efficient reduction-based simulation for power grid networks as shown later.

Second, for IR drop analysis, what matter are the excessive voltage drops occurring at a few time instances over the simulation period for each node. This is especially the case for real industry power grid networks, where the tap currents are very disruptive in nature as shown in Fig. 2.17 and so are the IR drops as shown in Fig. 2.18. Fig. 2.19 shows the frequency spectrum of the current shown in Fig. 2.17, which have shapes like sinc functions due to the impulse shapes of currents in time domain.

2.5 New reduction based IR drop analysis method

For IR drop analysis, many industry circuits exhibit rapid changing tap current waveforms as shown in Fig. 2.17. Such impulse-like current waveforms will have the frequency spectrum similar to sinc function in frequency domain as shown in Fig. 2.19, which has a long tail and thus is significant across wide frequency range. This requires large number of samplings to make ETBR accurate, which degrades its performance.

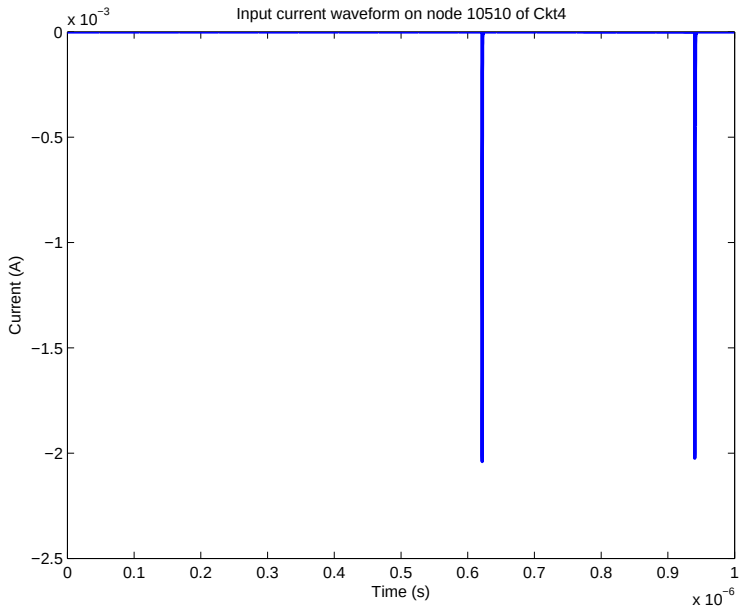


Figure 2.17: Input current waveform at the node 10510 of Ckt4 (the first one-tenth).

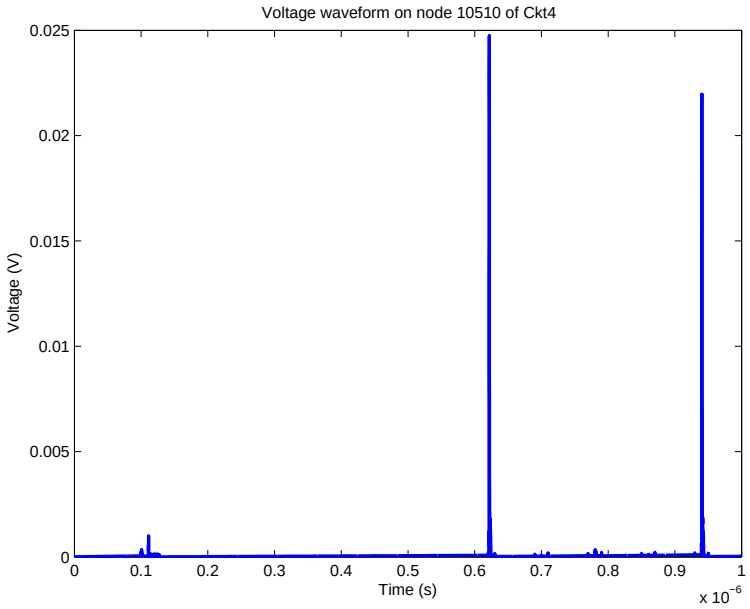


Figure 2.18: Voltage waveform at the node 10510 of Ckt4 (the first one-tenth).

In this chapter, we propose to reduce the errors during the transient simulation of the reduced models. In the new method, we monitor errors for the transient waveforms from

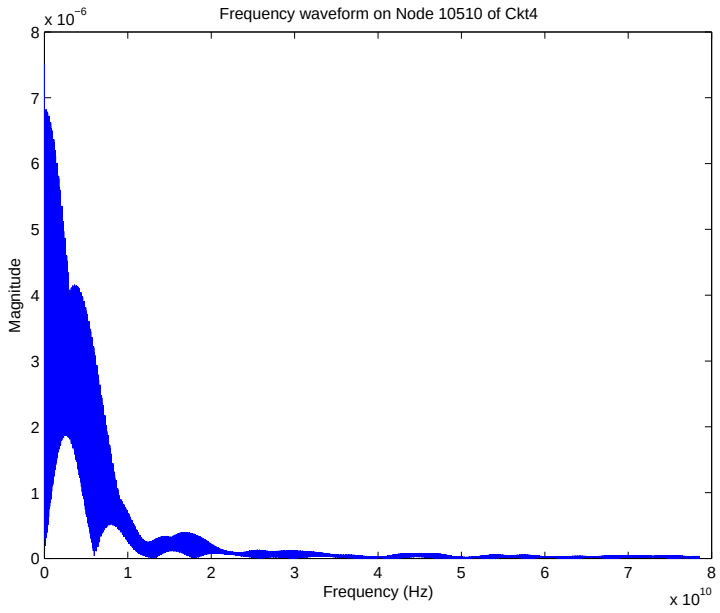


Figure 2.19: Frequency waveform at the node 10510 of Ckt4 (the first one-tenth).

the reduced model and switch to the original models when errors are large. Our Numerical examples show such large errors typically occur around the large voltage drop (spikes) and the proposed method can accurately estimate large voltage drops while still maintain decent speedup over traditional methods. We first present how errors are estimated in the time domain.

2.5.1 Error estimation in the time domain

One important aspect of the proposed method is to have accurate *a priori* error estimation at each time step t_i.

We propose to use the residual error information of the original on the states obtained from the reduced models. Specifically, for system (2.1), assume that h_i is the time step at time t_i and $v_r(t_i)$ and $v_r(t_{i-1})$ are the voltage vectors in the reduced systems at time t_i and t_{i-1} after the time discretization. Then we can define the residual error in time domain as

$$R(t_i) = GVv_r(t_i) + (C/h_i)V(v_r(t_i) - v_r(t_{i-1})) - Bu(t_i) \quad (2.20)$$

$$= (G + C/h_i)Vv_r(t_i) - (C/h_i)Vv_r(t_{i-1}) - Bu(t_i) \quad (2.21)$$

where $Vv_r(t_i)$ is an approximation of the original state $v(t_i)$, V is the project matrix computed from ETBR and $V \in R^{n\times q}$, q is the reduced order. Notice that if $Vv_r(t_i)$ is exactly equal to $x(t_i)$, the residual error should be zero. As a result, the norm of $R(t_i)$, $||R(t_i)||$ can serve as a good error indicator for the reduced model at t_i. Practically, we take $||R(t_i)||_\infty$ as the error indicator, which is the maximum absolute value of the element in $R(t_i)$.

Notice that we are only interested in the tap current nodes and the largest IR drop must happen in one tap current node. As a result, we do not need to check the all the nodes. The new residual formula considering only tap current nodes becomes

$$R_{tap}(t_i) = B^T(G + C/h_i)Vv_r(t_i) - B^T(C/h_i)Vv_r(t_{i-1}) - B^TBu(t_i) \quad (2.22)$$

Although $R(t_i)$ still involves the original matrices G and C, only matrix multiplications are involved. The time complexity of (2.21) is $O(p \times q)$, where p is the number of nodes and q is the size of the reduced model.

2.5.2 Effective resistance

The residual definition in (2.22) mainly give the current residual as $u(t_i)$ mainly contains the tap current sources (with only a few voltage sources normally). However, to effectively

control the errors, we need to know how much voltage errors such as current residuals will cause. As a result, we need to map from the current residual to the voltage residual (difference).

We introduce the effective resistance to perform the mapping. The effective resistance at time t_i is defined as

$$r_{eff}(t_i) = \frac{\max(v(t_i) - v_{dc}(t_i))}{\max(R_{tap}(t_i))} \tag{2.23}$$

where max means taking the maximum value of a vector. To compute $r_{eff}(t_i)$, we have to know the exact response solved from the original system $v(t_i)$. Actually we do not need to compute the effective response at every time step. Instead, we only compute it at the first step and the steps where errors are large and the original solutions are solved.

Our Numerical examples show that the effective resistances are quite consistent through the time steps for each circuit. Fig. 2.20 show the histogram for the effective resistance distribution all over the time steps of Ckt4 in the experimental section. It can seen that the effective resistance is dominated by values around 12. Practically we compute the average effective resistance r_{avg} all over the effective resistance computed seen so far to estimate the allowed maximum current residual.

2.5.3 Dynamic error control

To control the errors, we need to determine the maximum allowed current residual $i_{r,max}$. If the $max(R_{tap}(t_i))$ is larger than $i_{r,max}$, the original model will be solved. Otherwise, the reduced model is solved. The $i_{r,max}$ will be computed as

$$i_{r,max} = \frac{v_{ir,max} \times \alpha_{th}}{r_{avg}} \tag{2.24}$$

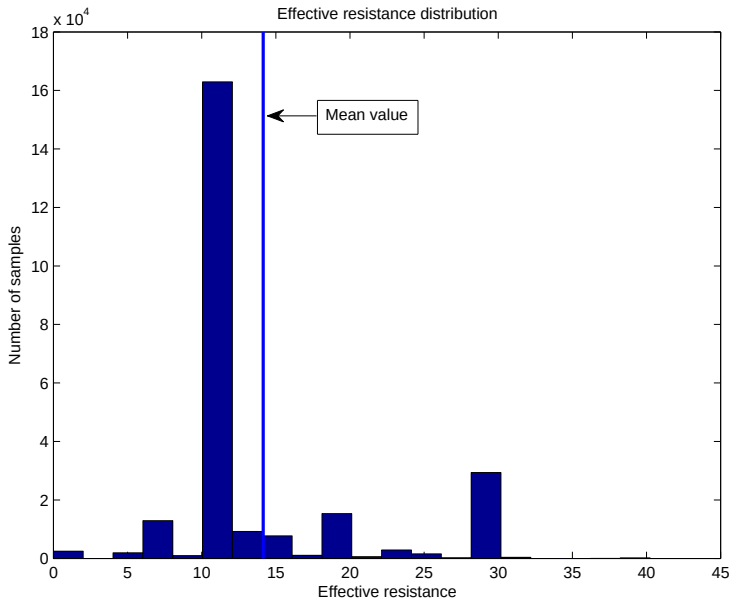


Figure 2.20: Effective resistance distribution of Ckt4.

where $v_{ir,max}$ is the largest IR drop seen so far and $0 < \alpha_{th} < 1$ is a user-defined threshold specifying the percentage of the allow voltage difference with respect to the largest voltage drop seen so far. Typically α_{th} is around 0.01 to 0.05.

At the beginning, the maximum voltage drop $v_{ir,max}$ may be small and it can lead to the necessary solving of the original models. To avoid this problem, the initial current residual is determined by the largest current value, I_{max}, of all the current sources over all the time steps.

$$i_{r,max} = I_{max} \times \alpha_{th} \tag{2.25}$$

So the actual allowed current residual will take the larger one of the two $i_{r,max}$s'.

2.5.4 The new IR drop analysis algorithm flow

In this subsection, we summarize all the steps we discuss before. We first present the proposed ETBR_IR method in *Algorithm* 2.

Algorithm 2: ETBR based IR drop analysis (ETBR_IR)

Input: Circuit of G, C, B, $u(t)$, number of samples: q, transient simulation step interval.
Output: Max IR drop for the given simulation interval.

---

1. Convert all the input signals $u(t)$ into $u(s)$ using FFT.
2. Select q frequency points $s_1, s_2, \ldots, s_q$ over the frequency range.
3. Compute $z_k^r = (s_k C + G)^{-1} B u(s_k)$.
4. Form the matrix $Z_r = [z_1^r, z_2^r, \ldots, z_q^r]$.
5. Perform SVD on Z_r, $Z_r = V_r S_r U_r^T$.
6. $\hat{G} = V_r^T G V_r$,$\hat{C} = V_r^T C V_r$, $\hat{B} = V_r^T B$
7. Solve the ith step ($\hat{G}, \hat{C}, \hat{B}, u(t)$), and get $\hat{v}(t)$. $v(t) = V_r \hat{v}(t)$.
8. Substitute $v(t)$ into (G, C), and get right hand side w_1. $w = B \times u$.
9. Compute current residual error $R = abs(w - w1)$. If $||R||_\infty$ is less than allowed residual $i_{r,max}$, then goto step 11, else goto step 10.
10. Solve the ith step ($G, C, B, u(t)$), and get $v(t)$. Update r_{avg} and $i_{r,max}$. Goto step 11.
11. Compute max IR drop. $i = i + 1$. Goto step 7.
12. Finish all the time steps and return max IR drop.

In the new algorithm, ETBR_IR first reduces the original circuits from step 1 to 6 using ETBR method. Then from step 7 to step 11, it performs the simulation on the reduced model. At the same time, it watches out for the error in each time step. If the error is larger than the given voltage IR drop threshold, ETBR_IR switches the simulation to the original model to get accurate results and then switch back the reduced model for the next step until we finish all the time step.

2.6 Numerical examples of ETBR_IR

The proposed *ETBR_IR* algorithm has been implemented using C++ and CSparse package [12]. ETBR_IR has been tested on a workstation with Intel quad-core 2.0GHz CPU and 8GB memory. All the benchmarks are power or ground grids from real industry designs. The statistics are summarized in Table 2.4. In the table, #Nodes means the total number of nodes in one test circuit. #VS means the total number of voltage sources and #IS means the total number of current sources. #Time Steps means the total number of simulation time steps.

In the experimental setting, the α_{th} is set to 0.05 except for Ckt6 and Ckt7 where α_{th} is set to 0.01. Also the number of samplings is set to 10 for all the case in *ETBR_IR*.

Table 2.4: Benchmark circuits

Test Ckts	#Nodes	#VS	#IS	#Time Steps
Ckt1	249475	1	5177	25001
Ckt2	154514	0	624	25001
Ckt3	60999	1	20901	250001
Ckt4	42222	0	10654	250001
Ckt5	49303	0	48756	79001
Ckt6	70127	1	28928	100001
Ckt7	75758	1	28048	100001

We compare ETBR_IR with original ETBR and UltraSim version 7.1, which is a commercial simulation tool from Cadence. UltraSim UPS (UltraSim Power network Solver) is the power grid analysis tool in UltraSim. It is an improved LU solver for power grid network analysis. We consider UltraSim UPS as the standard one, due to the reason that those real industry benchmarks are too large and too challenging for SPICE to solve it. We first show the performance comparison results in Table 2.5.

Table 2.5 shows the performance in CPU seconds of ETBR_IR, comparing with original

Table 2.5: Performance comparison (CPU seconds) of UltraSim, ETBR and ETBR_IR

Test Ckts	UltraSim (s)	ETBR (s)	ETBR_IR (s)	ETBR speedup	ETBR_IR speedup
Ckt1	49653	236	278	210.39	178.61
Ckt2	6906	104	122	66.40	56.61
Ckt3	6130	350	1122	17.51	5.46
Ckt4	3969	234	629	16.96	6.31
Ckt5	3969	551	1182	7.20	3.34
Ckt6	6144	803	1020	7.65	6.02
Ckt7	6523	765	950	8.53	6.87
Avg.				47.81	37.60

ETBR and UltraSim. From Table 2.5, we can see ETBR_IR can finish much faster than UltraSim for Ckt1 and Ckt2. It can archive about 37X speedup in average. We notice that *ETBR_IR* favors circuits with less current sources as shown for Ckt1 and Ckt2 where we see much higher speedup. This is due to the less time spent on the mapping results from reduced models to the original one. For other cases such as Ckt5 where the #IS is almost equal to #Nodes, ETBR_IR still can finish 3x faster than UltraSim.

Table 2.6: Accuracy comparison (max IR drop values) of UltraSim, ETBR and ETBR_IR

Test Ckts	UltraSim (mV)	ETBR (mV)	ETBR_IR (mV)	ETBR error	ETBR_IR error
Ckt1	1087.855	1087.812	1087.812	0.00%	0.00%
Ckt2	1899.810	1890.496	1890.500	0.49%	0.49%
Ckt3	12.230	6.021	12.222	50.77%	0.07%
Ckt4	24.734	15.549	24.707	37.14%	0.13%
Ckt5	8.424	5.055	8.363	39.99%	0.72%
Ckt6	196.300	181.251	197.468	7.67%	0.60%
Ckt7	255.920	196.102	252.613	23.37%	1.29%
Avg.				22.78%	0.47%

Table 2.6 shows the accuracy in maximum IR drop values of ETBR_IR compared with ETBR and UltraSim. Here we consider results from UltraSim UPS as the golden and errors

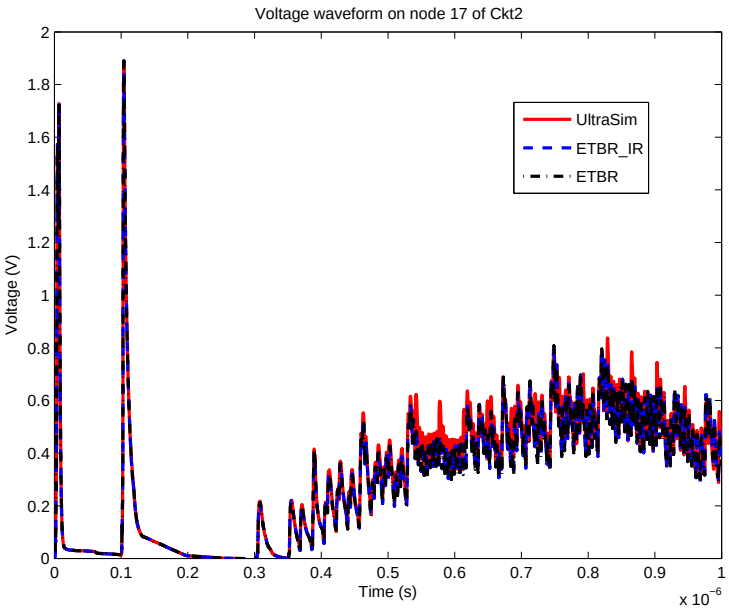


Figure 2.21: Voltage waveform at the node 17 of Ckt2.

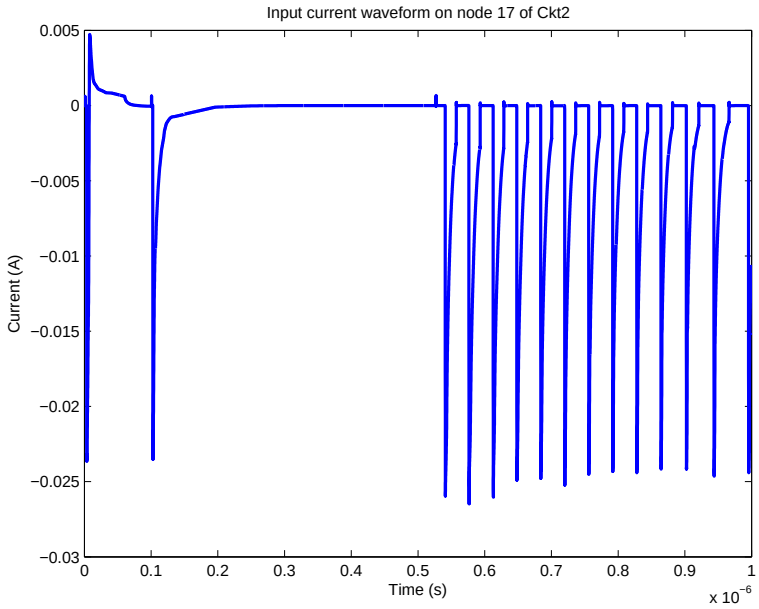


Figure 2.22: Input current waveform at the node 17 of Ckt2.

are computed as the relative errors to the golden results in percentage.

We can see that the max IR drop values computed by UltraSim and ETBR_IR are almost

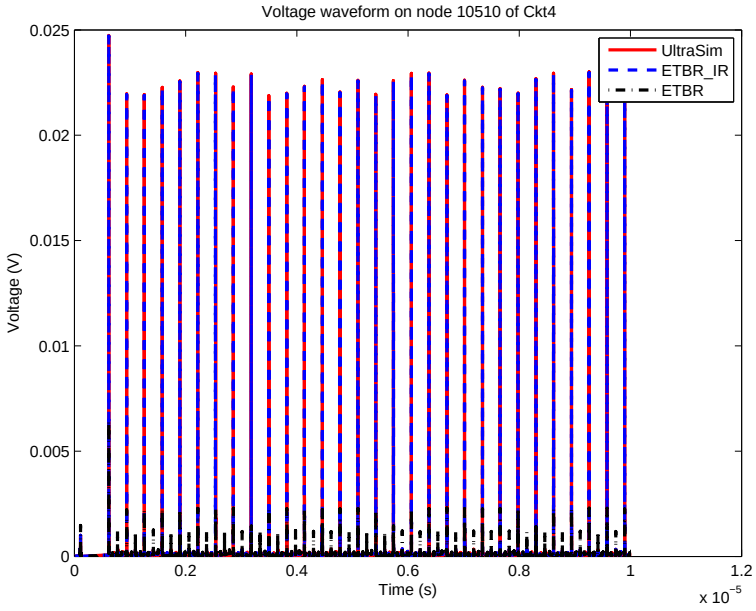


Figure 2.23: Voltage waveform at the node 10510 of Ckt4.

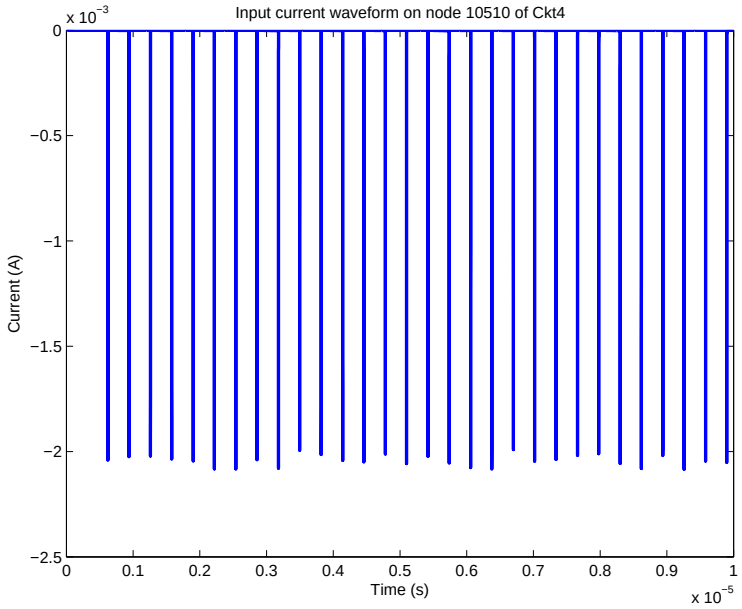


Figure 2.24: Input current waveform at the node 10510 of Ckt4.

the same. The max difference is less than 2%, and the average difference is less than 1%. Fig. 2.21, Fig. 2.23 and Fig. 2.25 show the voltage waveforms on the max IR drop node

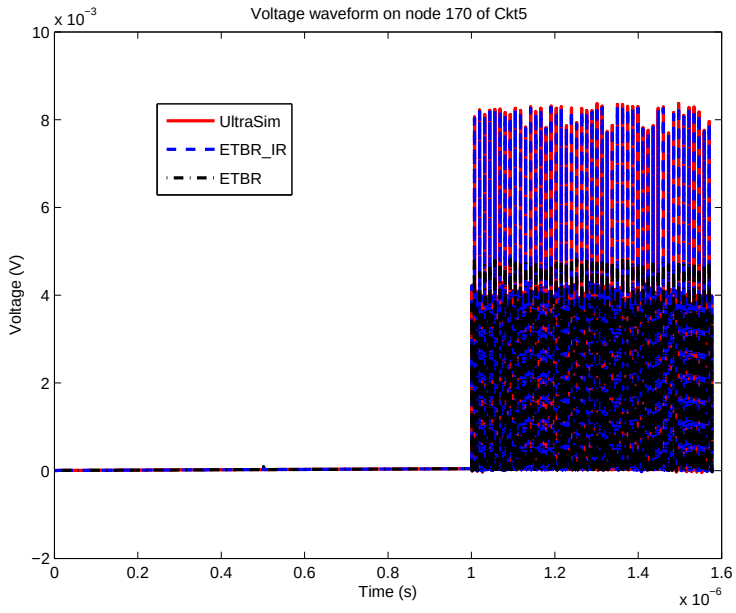


Figure 2.25: Voltage waveform at the node 107 of Ckt5.

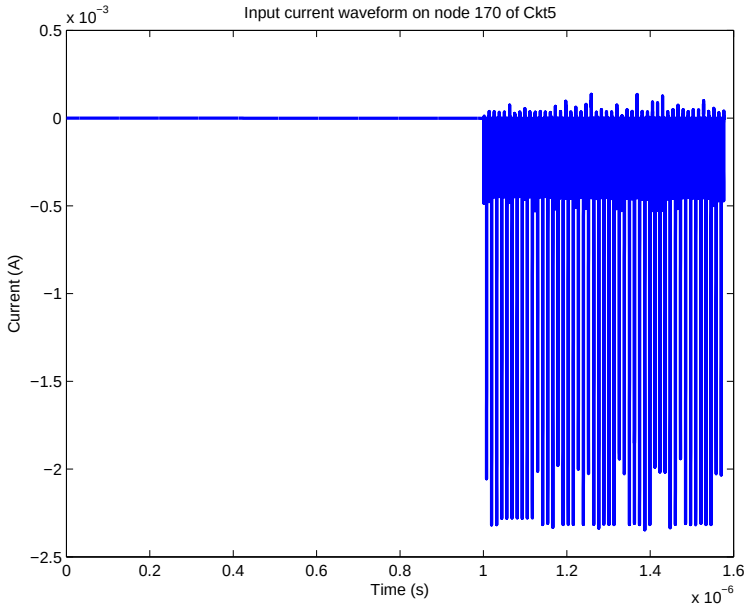


Figure 2.26: Input current waveform at the node 107 of Ckt5.

of Ckt2, Ckt4 and Ckt5, respectively. The max IR drop values computed by ETBR_IR are sufficiently accurate for the practice purpose. As we can see that the voltage drops are very

disruptive and shape. The maximum voltage drop only happens at a number of discrete time points over the the whole simulation period. Such disruptive waveforms comes from the similar input current waveforms as shown in Fig. 2.22, Fig. 2.24 and Fig. 2.26, which show the input current waveforms on the max IR drop node of Ckt2, Ckt4 and Ckt5, respectively. For the original ETBR, the errors for some circuits are quite large (22.78% in average) for the maximum voltage drops. We observe that ETBR works quite well for Ck1 and Ck2. The reason is that Ck1 and Ck2 have input waveforms that change less rapidly compared to other circuits as shown in Fig. 2.22, Fig. 2.24 and Fig. 2.26. We can increase the number of samples to improve the accuracy at much more computational costs.

Further, we can implement multithreading version of ETBR_IR to gain more speedup. ETBR_IR (*Algorithm* 2) mainly consists of two parts: one-time reduction (Step 1 - Step 6) and error-checking simulation (Step 7 - Step 11). When we look into *Algorithm* 2, Step 2 and Step 3 can be fully parallel computed without any overlaps. Here we use multithreading techniques as circuit matrices G, C, B can be shared between threads without using local copies. In this way, multithreading version ETBR_IR can save a lot of memory compared to multiprocessing version.

Table 2.7 shows the performance comparison of reduction time (Step 1 - Step 6) in ETBR_IR between single core and Quad-Core. The speedup we achieve is from multi-threading implementation of the one-time reduction in ETBR_IR flow. We can see that the reduction time in ETBR_IR is able to gain around 3x speedup.

The reason why we could not achieve 4x speedup on a Quad-Core machine is that in an algorithm there are always some parts that could not be parallel computed. Those parts must be done sequentially. So the total speedup must be less than 4x.

For the total run time of ETBR_IR, Table 2.8 shows the performance comparison be-

Table 2.7: Performance comparison (CPU seconds) of reduction time in ETBR_IR between single core and Quad-Core

Test Ckts	Single (s)	Quad-Core (s)	speedup
Ckt1	206	66	3.12
Ckt2	94	34	2.76
Ckt3	174	62	2.80
Ckt4	118	41	2.88
Ckt5	79	29	2.72
Ckt6	132	43	3.07
Ckt7	127	42	3.02
Avg.			2.91

tween ETBR_IR and ETBR_IR_THREAD on our Quad-Core machine. We can see that multithreading version of ETBR_IR is able to achieve up to 2x, 35% on average speedup compared to single process version of ETBR_IR.

Table 2.8: Performance comparison (CPU seconds) between ETBR_IR and ETBR_IR_THREAD

Test Ckts	ETBR_IR (s)	ETBR_IR_THREAD	speedup
Ckt1	278	138	2.01
Ckt2	122	62	1.97
Ckt3	1122	1010	1.11
Ckt4	629	552	1.14
Ckt5	1182	1132	1.04
Ckt6	1020	931	1.10
Ckt7	950	865	1.10
Avg.			1.35

2.7 Summary

In this chapter, we have proposed a new fast simulation method ETBR for extended truncated balanced realization. ETBR is based on a more accurate reduction framework: truncated balanced realization, which was shown to be more accurate than Krylov subspace

method used in EKS method. The proposed method is very amenable for threading-based parallel computing, as the response Gramian, which is used to construct the projection matrix, is computed in a Monte-Carlo-like sampling style and each sampling can be computed in parallel. This contrasts with all the Krylov subspace based methods like the EKS method, where moments have to be computed in a sequential order. The feature is important as the multi-core architectures and multi-core computing are becoming commonplace [30, 70]. ETBR can naturally exploit task-level threading-oriented parallelism based on multicore architectures to significantly boost the simulation performance. ETBR also avoids the explicit moment representation of the input signals, which have well-known numerical problems in the past. Instead, it uses spectrum representation of input signals by fast Fourier transformation. As a result, ETBR is much more flexible for different types of input sources and can better capture the high frequency contents than EKS and this leads to more accurate results for fast changing input signals.

To make ETBR more accuracy, we further introduce an error control mechanism into it. The improved method is called ETBR_IR. The error control mechanism is based on the system residuals as well as the novel effective resistance concept to compute the errors in terms of more useful voltage drop values. The on-the-fly error reduction works well for compensating high frequency accuracy loss related to disruptive tap current waveforms in typical industry power delivery networks. ETBR_IR provides an efficient way to easily trade errors for speedup to suit different applications. Numerical results show ETBR_IR can significantly reduce the errors of the existing ETBR method at the similar computing cost, while it can have 10X and more speedup over the the commercial power grid simulator in UltraSim with about 1-2% errors on a number of real industry benchmark circuits.

Chapter 3

varETBR: Variational Extended Truncated Balanced Realization for On-Chip Power Grid Network Analysis

Another issue for reliable on-chip power delivery is the increasing process-induced variability [62, 46]. The process induced variations manifest themselves at different levels (wafer level, die-level and within a die) and they are caused by different sources (lithograph, materials, aging etc) [10, 45]. Some of the variations are systematic, like those caused by chemical mechanical polishing (CMP), while some are purely random, like the doping density of impurities and edge roughness. As the technology moves to 65nm and comes near to 45nm, variation will become more and more pronounced for both systemic and random components.

One of the process variabilities comes from the voltage drop variations in on-chip power distribution networks. Voltage drop has significant impacts on the circuit timing [51]. Variability on voltage drops will also affect the statistical timing analysis. A number of research

works have been proposed recently to address the variational voltage drop issues in the on-chip power delivery networks under process variations. The voltage drop of power grid networks subject to leakage current variations was first studied in [16, 17]. This method assumes that the log-normal distribution of the node voltage drop is caused by log-normal leakage current inputs, and is based on a localized Monte Carlo (sampling) method to compute the variance of the node voltage drop. However, this localized sampling method is limited to the static DC solution of power grids modeled as resistor-only networks. Therefore, it can only compute the responses to the standby leakage currents. However, dynamic leakage currents are becoming more significant, due to the intensive use of sleep transistors for reducing leakage powers. In [72, 50], impulse responses are used to compute the mean and variances of node voltage responses caused by general current variations. But this method requires the impulse response from all the current sources to all the nodes, which is expensive to compute for a large network. Methods proposed in [20, 19] use orthogonal polynomial chaos expansion of random processes to represent and solve for the stochastic responses of linear systems. But existing approaches can only consider Gaussian distributions, and analysis times increase with the number of variables. The methods have been improved by the StoEKS method [42, 41], where reduction is performed on the variational circuit matrices before the simulation.

In this chapter, we present a novel scalable statistical simulation approach for large power grid network analysis considering process variations. The new algorithm is very scalable for large networks with a large number of random variables. Our work is inspired by the recent work on variational model order reduction using fast balanced truncation method (called variational Poor man's TBR method, or varPMTBR [53]).The new method, called *varETBR*, is based on the recently proposed extended truncated balanced

realization (ETBR) method [33, 35]. To consider the variational parameters, we extend the concept of response Grammian, which was used in ETBR to compute the reduction projection subspace, to the variational response Grammian. Then Monte Carlo based numerical integration is employed to multiple-dimensional integrals.

Different from traditional reduction approaches, varETBR calculates the variational response Grammians, considering both system and input source variations, to generate the projection subspace. In this way, much more efficient reduction can be performed for interconnects with massive terminals like power grid networks [77]. Furthermore, the new method is based on the globally more accurate balanced truncation reduction method instead of the less accurate Krylov subspace method as in EKS/IEKS [79, 32]. After the reduction, Monte Carlo based statistical simulation is performed on the reduced system and the statistical responses of the original systems are obtained thereafter. The varETBR only requires the simulation of the reduced circuit using any existing transient analysis method. It is insensitive to the number of variables and variation ranges in terms of computing costs and accuracy, which makes it very general and scalable. Numerical examples, on a number of the IBM benchmark circuits [47] up to 1.6 million nodes, show that the varETBR can be up to $1900X$ faster than the Monte Carlo method, and is much more scalable than the StoEKS method [42, 41]. varETBR can solve very large power grid networks with large numbers of random variables, large variation ranges and different variational distributions.

3.1 Variational model for power grid networks

In the presence of process variations, the G and C matrices and input currents $u(t)$ depend on variational circuit parameters, such as metal wire width, length, and metal thickness on

power grids, as well as transistor parameters, such as channel length, width, gate oxide thickness, etc. Process-induced random variations can be systematic and random and can be highly partially correlated [10]. For highly correlated variations like inter-die variations, the worst case corner can be easily found by setting the parameters to their range limits (mean plus 3σ). The difficulty lies in the intra-die variations, where circuit parameters are not correlated or spatially correlated. Intra-die variations also consist of local and layout dependent deterministic components and random components. In this chapter, we focus on the random variations, which are typically modeled as multivariate Gaussian processes with any spatial correlation [28].

We assume that we have a number of independent (uncorrelated) transformed orthonormal Gaussian random variables $\xi = [\xi_1, ..., \xi_M]$, which model the channel length, the device threshold voltage and the wire geometry variations. Therefore, the MNA equation for (2.1) becomes

$$G(\xi)v(t) + C(\xi)\frac{dv(t)}{dt} = Bu(t, \xi) \tag{3.1}$$

The spatial correlation in the intra-die variation can be processed by using the principal component analysis method (or other methods like K-L transformation or principal factor analysis, etc.) to transform the correlated variables into un-correlated variables before spectral statistical analysis [20].

Note that the input vector $u(t, \xi) = i(t, \xi) + u_0(t)$, where the current vector $i(t, \xi)$ follows the log-normal distribution and has both deterministic and random components, and the input voltage vector $u_0(t)$ is not effected by ξ. In this chapter, we assume the dynamic currents (power) due to circuit switching are still modeled as deterministic currents. Therefore, we only consider the leakage variations as they are more significant owing to their log-normal distributions. Specifically, we expand the variational G and C around their

mean values and keep the first order terms as in [40, 11, 53].

$$\begin{aligned} G(\xi) &= G_0 + G_1\xi_1 + G_2\xi_2 + ... + G_M\xi_M \\ C(\xi) &= C_0 + C_1\xi_1 + C_2\xi_2 + ... + C_M\xi_M \end{aligned} \tag{3.2}$$

We remark that the proposed method can be trivially extended to the second and higher order terms [53]. The input current variation $i(t, \xi)$ follows the log-normal distribution as leakage variations are dominant factors:

$$i(\xi) = e^{g(\xi)}, \quad g(\xi) = \mu + \sigma\xi \tag{3.3}$$

Note that input current variation $i(\xi)$ is not a function of time as we only model the static leakage variations for the simplicity of presentation. However, the proposed approach can be easily applied to time-variant variations with any distribution.

3.2 New variational analysis method: varETBR

In this section, we detail the new proposed *varETBR* method. We first present the recently proposed ETBR method for deterministic power grid analysis based on reduction techniques.

3.2.1 Extended truncated balanced realization scheme

The new method is based on the recently proposed extended truncated balanced realization method [33]. We first review this method.

For a linear system in (2.1), we first define the frequency-domain *Response Grammian*,

$$X_r = \int_{-\infty}^{+\infty} (j\omega C + G)^{-1} B u(j\omega) u^T(j\omega) B^T (j\omega C + G)^{-H} d\omega \tag{3.4}$$

which is different from the Grammian concepts in the traditional TBR based reduction framework. Notice that in the new Grammian definition, the input signals $u(j\omega)$ are considered. As a result, $(j\omega C + G)^{-1} B u(j\omega)$ serves as the system response with respect to the input signal $u(j\omega)$ and resulting X_r becomes the response Grammian.

To fast compute the response Grammian X_r, we can use Monte Carlo based method to estimate the numerical value as done in [53]. Specifically, let ω_k be kth sampling point over the frequency range. If we further define

$$z_k^r = (j\omega_k C + G)^{-1} B u(j\omega_k) \tag{3.5}$$

then $\hat{X}$ can be computed approximately by numerical quadrature methods

$$\hat{X}_r = \sum_k w_k z_k^r z_k^{rH} = Z_r W^2 Z_r^H \tag{3.6}$$

where Z_r is a matrix whose columns are z_k^r and W a diagonal matrix with diagonal entries $w_{kk} = \sqrt{w_k}$. w_k comes from a specific quadrature method.

The projection matrix can be obtained by singular value decomposition of Z_r. After this, we can reduce the original matrices into small ones and then perform the transient analysis on the reduced circuit matrices. The extended TBR algorithm is summarized in *Algorithm* 3.

Notice that we need the frequency response caused by input signal $u(j\omega_k)$ in (3.5). This

Algorithm 3: ETBR: Extended Truncated Balanced Realization method

---

Input: Circuit of G, C, B, $u(t)$, number of samples: q
Output: Transient voltage waveforms

---

1. Convert all the input signals $u(t)$ into $u(s)$ using Fast Fourier Transformation (FFT).
2. Select q frequency points $s_1, s_2, \ldots, s_q$ over the frequency range
3. Compute $z_k^r = (s_k C + G)^{-1} B u(s_k)$
4. Form the matrix $Z_r = [z_1^r, z_2^r, \ldots, z_q^r]$
5. Perform Singular Value Decomposition (SVD) on Z_r, $Z_r = V_r S_r U_r^T$
6. $\hat{G} = V_r^T G V_r$,$\hat{C} = V_r^T C V_r$, $\hat{B} = V_r^T B$
7. Perform the transient analysis on reduced system $[\hat{G}, \hat{C}, \hat{B}]$ to compute responses $\hat{v}(t)$
8. Compute the final transient waveforms $v(t) = V_r \hat{v}(t)$

---

can be obtained by fast Fourier transformation on the input signals in time domain. Using frequency spectrum representations for the input signals is a significant improvement over the EKS method as we avoid the explicit moment representation of the current sources, which are not accurate for currents rich in high frequency components due to the well-known problems in explicit moment matching methods [58]. Accuracy is also improved owing to the use of the fast balanced truncation method for the reduction, which has global accuracy [43, 56].

Note that we use congruence transformation for the reduction process with orthogonal columns in the projection matrix (by using Arnoldi or Arnoldi-like process), the reduced system must be stable. For simulation purposes, this is sufficient. If all the observable ports are also the current source nodes, i.e. $y(t) = B^T v(t)$, where $y(t)$ is the voltage vector at all observable ports, the reduced system is also passive. It was also shown in [56] that the fast TBR method has similar time complexity to multiple-point Krylov subspace based reduction methods. The extended TBR method also has similar computation costs as the EKS method.

3.2.2 The new variational ETBR method

We first start the new statistical interpretation of Grammian computation before introducing the new method.

3.2.3 Statistical interpretation of Grammian

For a linear dynamic system formulated in state space equations (MNA) in (2.1), if complex frequency $j\omega$ is a vector of random variables with uniform distribution in the frequency domain, then the state responses $V(j\omega) = (G + j\omega C)^{-1}Bu(\omega)$ become random variables in frequency domain. Its covariance matrix can be computed as

$$X_r = E\{V(j\omega)V(j\omega)^T\} = \int_{-\infty}^{+\infty} V(j\omega)V(j\omega)^T \, d\omega \tag{3.7}$$

where $E\{x\}$ stands for computing the mean of random variable x. X_r is defined in (3.4). The response Grammian essentially can be viewed as the covariance matrix associated with state responses. X_r can also be interpreted as the mean for function $P(j\omega)$ on evenly distributed random variables $j\omega$ over $[-\infty, +\infty]$ [1]. ETBR method actually performs the principal component analysis (PCA) transformation of the mentioned random process with uniform distribution.

3.3 Computation of variational response Grammian

Define $P(j\omega) = V(j\omega)V(j\omega)^T$. Now suppose in addition to the frequency variable $j\omega$, $P(j\omega, \xi)$ is also the function of the random variable ξ with probability density $f(\xi)$. The

[1] Practically, the interesting frequency range is always bounded

new *variational* response Grammian X_{vr} can be defined as

$$X_{vr} = \int_{s_\xi} \int_{-\infty}^{+\infty} f(\xi) P(j\omega, \xi) d\omega d\xi = E\{P(j\omega, \xi)\} \tag{3.8}$$

where s_ξ is the domain of variable ξ with a specific distribution. Hence, X_{vr} is essentially the mean of $P(j\omega, \xi)$ with respect to both $j\omega$ and ξ. The concept can be extended to more random variables $\xi = [\xi_1, \xi_2, ..., \xi_n]$ and each variable ξ_i adds one more dimension of integration for the integral.

As a result, calculating the variational Grammian is equivalent to computing the multi-dimensional integral in (3.8), which can be computed by numerical quadrature methods. For one dimensional integration, efficient methods like Gaussian quadrature rule [74] exist. For multi-dimension integral, quadrature points are created by taking tensor products of one-dimensional quadrature points, which, unfortunately, grow exponentially with the number of variables (dimensions) and makes the integration intractable for practical problems [68].

Practically, established techniques like Monte Carlo or quasi Monte Carlo are more amenable for computing the integrals [74] as the computation costs are not dependent on the number of variables (integral dimensions). In this chapter, we apply the standard Monte Carlo method to compute the variational Grammian X_{vr}. The Monte Carlo estimation of (3.8) consists of sampling N random points $x_i \in S$, where S is the domain for both frequency and other variables, from a uniform distribution, and then computing the estimate as

$$\hat{X}_{vr} = \frac{1}{N} \sum_{i=1}^{N} P(x_i) \tag{3.9}$$

The Monte Carlo method has a slow convergence rate ($1/\sqrt{N}$) in general although it can

be improved to $(1/N)$ by quasi Monte Carlo methods. But as observed by Phillips [53], the projection subspace constructed from the sampled points actually converge much faster than the value of $\hat{X}_{vr}$. As we are concerned with the projection subspace rather than the actual numerical values of X_{vr}, we require only the drawing of a small number of samples as shown in the experimental result. The *varETBR* algorithm flow is shown in *Algorithm* 4.

Algorithm 4: varETBR: Variational extended Truncated Balanced Realization method

---

Input: Circuit of $G(\xi)$, $C(\xi)$, B, $u(t,\xi)$, variables $\xi = [\xi_1, ..., \xi_M]$, number of samples: q
Output: The variational response $v(t)$

---

1. Convert all the nominal input signals $u(t)$ into $u(s)$ using FFT.
2. Select q points over an M+1 dimensional space $(s, \xi_1, ..., \xi_M)$
3. Compute $z_k^r = (s_k C(\xi_1^k, ..., \xi_M^k) + G(\xi_1^k, ..., \xi_M^k))^{-1} B u(s_k, \xi_1^k, ..., \xi_M^k))$ through Monte Carlo.
4. Form the matrix $Z_r = [z_1^r, z_2^r, \ldots, z_q^r]$
5. Perform SVD on Z_r, $Z_r = V_r S_r U_r^T$
6. $\hat{G}(\xi) = V_r^T G(\xi) V_r$, $\hat{C}((\xi)) = V_r^T C(\xi) V_r$, $\hat{B} = V_r^T B$
7. Perform the Monte Carlo simulation on $\hat{G}(\xi)\hat{v}(t) + \hat{C}(\xi)\frac{d\hat{v}(t)}{dt} = \hat{B}u(t,\xi)$
8. Obtain the variational response $v(t) = V_r \hat{v}(t)$.
9. End

Where $\hat{G}(\xi) = V_r^T G(\xi) V_r$ and $\hat{C}((\xi)) = V_r^T C(\xi) V_r$ stand for

$$\hat{G}(\xi) = V_r^T G_0 V_r + V_r^T G_1 V_r \xi_1 + V_r^T G_2 V_r \xi_2 + ... + V_r^T G_M V_r \xi_M \quad (3.10)$$

$$\hat{C}(\xi) = V_r^T C_0 V_r + V_r^T C_1 V_r \xi_1 + V_r^T C_2 V_r \xi_2 + ... + V_r^T C_M V_r \xi_M \quad (3.11)$$

The algorithm starts with the given power grid network and the number of samplings q, which are used for building the projection subspace. Then it computes the variational response $z_k^r = (s_k C(\xi_1^k, ..., \xi_M^k) + G(\xi_1^k, ..., \xi_M^k))^{-1} B u(s_k, \xi_1^k, ..., \xi_M^k))$ randomly. Then we perform the SVD on $Z_r = [z_1^r, z_2^r, \ldots, z_q^r]$ to construct the projection matrix. After the reduction, we perform the Monte Carlo based statistical analysis to obtain the variational

responses from $v(t) = V_r\hat{v}(t)$.

We remark that in both Algorithm 3 and Algorithm 4, we perform Monte-Carlo like random sampling to obtain q frequency sampling points over the M+1 dimensional space for given frequency range and parameter spaces (for Algorithm 3, sampling is on the given frequency range only). We note that the MC based sampling method is also used in the PMTBR method [53].

Compared with existing approaches, varETBR offers several advantages and features. First, varETBR only uses Monte Carlo sampling, it is easy to implement and is very general for dealing with different variation distributions and large variation ranges. It is also more amenable for parallel computing as each sampling in frequency domain can be done in parallel. Second, it is vary scalable for solving large networks with large number of variables as reduction is performed. Third, varETBR is more accurate over wide band frequency ranges as it samples over frequency band (compared with the less accurate moment-matching based EKS method). Last, it avoids the explicit moment representation of the input signals, leading to more accurate results than the EKS method when signals are rich in high frequency components.

3.4 Numerical examples

The proposed *varETBR* algorithm has been implemented using MATLAB and tested on an Intel quad-core workstation with 16GB memory under Linux environment.

All the benchmarks are real PG circuits from IBM provided by [47], but the circuits in [47] are resistor-only circuits. For transient analysis, we need to add capacitors and transient input waveforms. As a result, we modified the benchmark circuits. First we

Table 3.1: Power Grid (PG) benchmarks

Name	#Nodes	#V Sources	#I Sources
ibmpg1	30638	14308	10774
ibmpg2	127238	330	37926
ibmpg3	851584	955	201054
ibmpg4	953583	962	276976
ibmpg5	1079310	539087	540800
ibmpg6	1670494	836239	761484

added one grounded capacitor on each node with a random value in the magnitude of pF. Second we replaced the DC current sources by a piecewise linear signal in the benchmark. The values of these signals are also randomly generated based on their original values in the DC benchmarks. We implemented a parser using Python to transform the SPICE format benchmarks into MATLAB format.

The summary of our transient PG benchmarks is shown in Table 3.1. We use MNA formulation to set up the circuit matrices. To efficiently solve PG circuits with 1.6 million nodes in MATLAB, an external linear solver package UMFPACK [2] is used, which is linked with Matlab using Matlab mexFunction.

We will compare varETBR with the Monte Carlo method, first in accuracy and then in CPU times. In all the test cases, the number of samples used for forming the subspace in varETBR are 50, based on our experience. The reduced order is set to $p = 10$, which is sufficiently accurate in practice. Here we set the variation range, the ratio of the maximum variation value to the nominal value, to 10% and set the number of variables to 6 (2 for G, 2 for C and 2 for i). $G(\xi)$ and $C(\xi)$ follow Gaussian distribution. $i(t, \xi)$, which models the leakage variations [16], follows log-normal distribution.

varETBR is essentially a kind of reduced Monte Carlo method. It inherits the merits of Monte Carlo methods, which are less sensitive to the number of variables and can reflect

the real distribution very accurately for a sufficient number of samples. But the main disadvantage of Monte Carlo is that it is too slow to simulate on large scale circuits. varETBR first reduces the size of circuits to a small number while maintaining sufficient accuracy. Thus, varETBR can do Monte Carlo simulation on the reduced circuits very fast. Note that the reduction process is done only once during the simulation process.

To verify the accuracy of our varETBR method, we show the results of simulations on *ibmpg1* (100 samples) and *ibmpg6* (10 samples). Fig. 3.1 and Fig. 3.2 show the results of varETBR and the pure Monte Carlo method at the 1000th node
(named n1_20583_11663 in SPICE format) of *ibmpg1* and at the 1000th node (named n3_16800_9178400 in SPICE format) of *ibmpg6*, respectively. The circuit equations in Monte Carlo are solved by MATLAB.

The absolute errors and relative errors of *ibmpg1* and *ibmpg6* are shown in Fig. 3.3 and Fig. 3.4. We can briefly see that errors are very small and our varETBR is very accurate. Note that the errors are not only influenced by the variations but also depends on the reduced order. To increase the accuracy, we may increase the reduced order. In our tests, we set the reduced order to $p = 10$ for all the benchmarks.

Next we do accuracy comparison with Monte Carlo on the probability distributions including means and variances. Fig. 3.5 shows the voltage distributions of both varETBR and original Monte Carlo at the 1000th node of *ibmpg1* when $t = 50ns$ (200 time steps between $0ns$ and $200ns$ in total). We can also refer to simulation waveforms on $t = 50ns$ in Fig. 3.1. Note that the results do not follow Gaussian distribution as $G(\xi)$ and $C(\xi)$ follow Gaussian distribution and $i(t, \xi)$ follows log-normal distribution. From Fig. 3.5, we can see that not only are the means and the variances of varETBR and Monte Carlo almost the same, but so are their probability distributions.

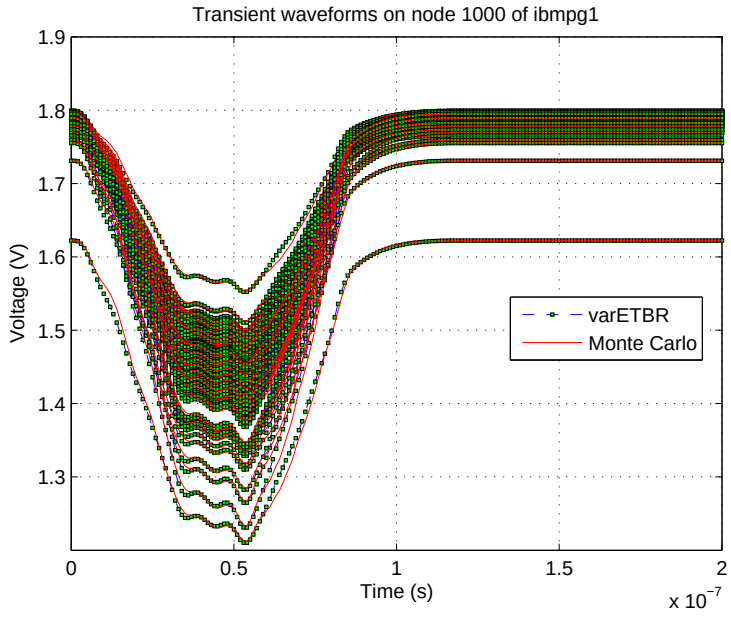


Figure 3.1: Transient waveform at the 1000th node (n1_20583_11663) of *ibmpg1* ($p = 10$, 100 samples).

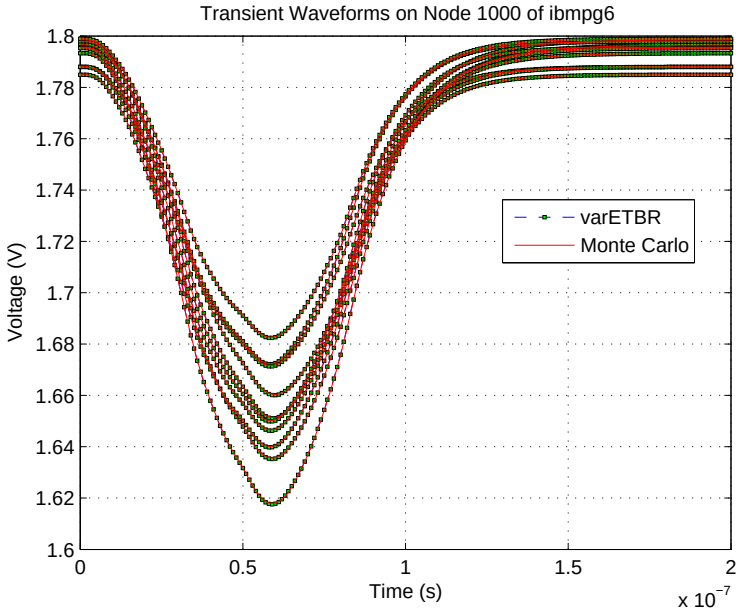
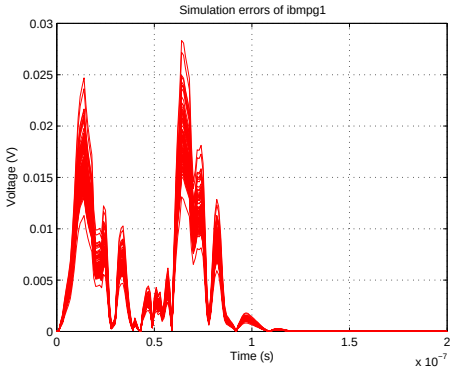
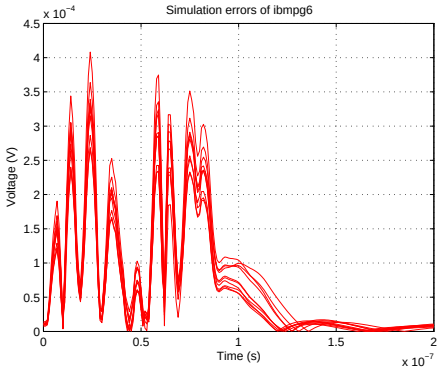


Figure 3.2: Transient waveform at the 1000th node (n3_16800_9178400) of *ibmpg6* ($p = 10$, 10 samples).

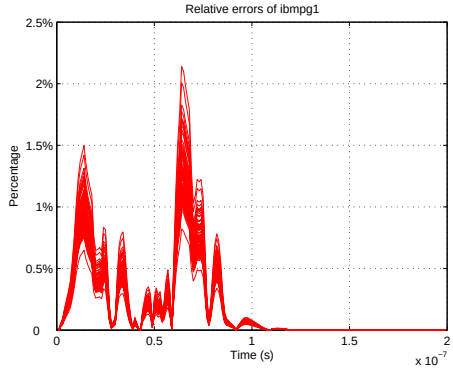


(a) Simulation errors of *ibmpg1* (100 samples).

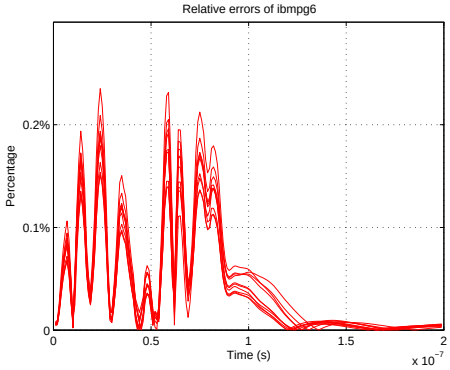


(b) Simulation errors of *ibmpg6* (10 samples).

Figure 3.3: Simulation errors of *ibmpg1* and *ibmpg6*



(a) Relative errors of *ibmpg1* (100 samples).



(b) Relative errors of *ibmpg6* (10 samples).

Figure 3.4: Relative errors of *ibmpg1* and *ibmpg6*

Finally, we compare the CPU times of varETBR and the pure Monte Carlo method. To verify the efficiency of varETBR on both CPU time and memory, we do not need to run simulations many times for both varETBR and Monte Carlo. We will run 10 or 100 samples for each benchmark to show the efficiency of varETBR since we already showed its accuracy. Although we only run a small number of samples, the speedup will be the same. Table 3.2 shows the actual CPU times of both varETBR (including FFT costs) and

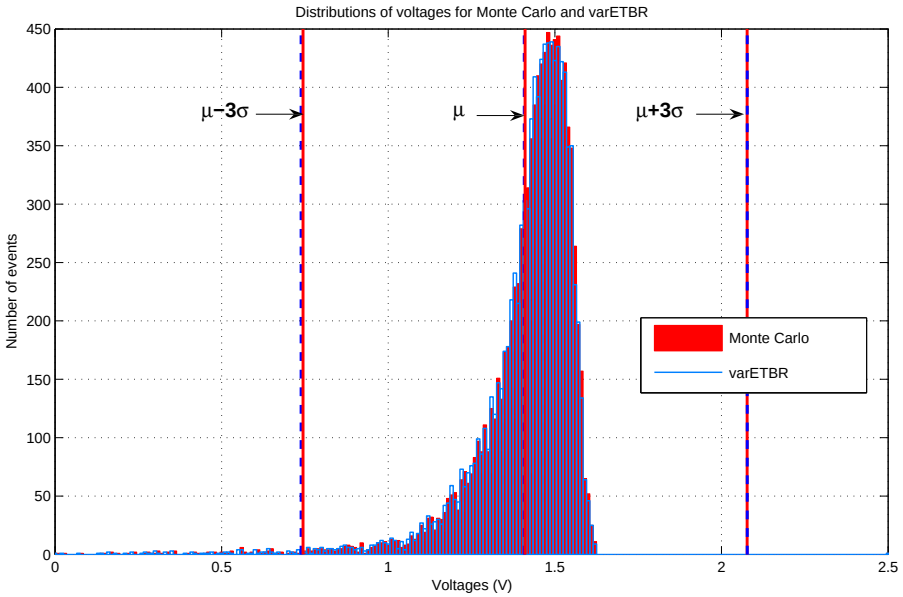


Figure 3.5: Voltage distribution at the 1000th node of *ibmpg1* (10000 samples) when $t = 50ns$.

Table 3.2: CPU times (s) comparison of varETBR and Monte Carlo ($q = 50$, $p = 10$)

Test Ckts	varETBR (s)		Monte Carlo
	Red. (s)	Sim. (s)	Sim. (s)
ibmpg1 (100)	23	14	739
ibmpg1 (10000)	23	1335	70719
ibmpg2 (10)	115	1.4	536
ibmpg3 (10)	1879	1.5	4973
ibmpg4 (10)	2130	1.3	5275
ibmpg5 (10)	1439	1.3	5130
ibmpg6 (10)	1957	1.5	6774

Monte Carlo on the given set of circuits. The number of sampling points in reduction is $q = 50$. The reduction order is $p = 10$. Table 3.3 shows the projected CPU times of varETBR (one-time reduction plus 10000 simulations) and Monte Carlo (10000 samples).

In varETBR, circuit model becomes much smaller after reduction and we only need to performa the reduction once. Therefore, the total time is much faster than Monte Carlo (up to $1960X$). Basically, the bigger the original circuit size is, the faster the simulation will

Table 3.3: Projected CPU times (s) comparison of varETBR and Monte Carlo ($q = 50$, $p = 10$, 10000 samples)

Test Ckts	varETBR (s)	Monte Carlo (s)	Speedup
ibmpg1	1358	70719	$53X$
ibmpg2	1515	53600	$354X$
ibmpg3	3379	497300	$1472X$
ibmpg4	3430	527500	$1538X$
ibmpg5	2739	513000	$1873X$
ibmpg6	3457	677400	$1960X$

be for varETBR. Compared to the Monte-Carlo method, the reduction time is negligible compared to the total simulation time.

Note that we run random simulation 10000 times for *ibmpg1*, as shown in Table 3.2, to show the efficiency of our varETBR in practice.

It can be seen that varETBR is very scalable. It is, in practice, almost independent of the variation range and numbers of variables. One possible reason is that varETBR already captures the most dominant subspaces even for small number of samples (50 in our case) as explained in Subsection 3.2.2.

When we increase the variation range and the number of variables, the accuracy of varETBR is almost unchanged. Table 3.4 and Table 3.5 shows that the mean and variance comparison between the two methods for 10K Monte Carlo runs, where we increase the number of variables from 6 to 15 and the variation range from 10% to 100%. The tables show that varETBR is very insensitive to the number of variables and variation range for a given circuit *ibmpg1*, where simulations are run on 10000 samples for both varETBR ($q = 50$, $p = 10$) and Monte Carlo.

The variation range var is the ratio of the maximum variation value to the nominal value. So "$var = 100\%$" means the maximum variation value may be as large as the

Table 3.4: Relative errors for the mean of max voltage drop of varETBR compared with Monte Carlo on the 2000th node of *ibmpg1* ($q = 50$, $p = 10$, 10000 samples) for different variation ranges and different numbers of variables

	Variation range			
#Variables	$var = 10\%$	$var = 30\%$	$var = 50\%$	$var = 100\%$
$M = 6$	0.16%	0.08%	0.17%	0.21%
$M = 9$	0.16%	0.25%	0.08%	0.23%
$M = 12$	0.25%	0.07%	0.07%	0.28%
$M = 15$	0.15%	0.06%	0.05%	0.06%

nominal value.

Table 3.5: Relative errors for the variance of max voltage drop of varETBR compared with Monte Carlo on the 2000th node of *ibmpg1* ($q = 50$, $p = 10$, 10000 samples) for different variation ranges and different numbers of variables

	Variation range			
#Variables	$var = 10\%$	$var = 30\%$	$var = 50\%$	$var = 100\%$
$M = 6$	0.27%	1.54%	1.38%	1.73%
$M = 9$	0.25%	0.67%	1.32%	1.27%
$M = 12$	0.42%	0.07%	0.68%	1.41%
$M = 15$	0.18%	1.11%	0.67%	2.14%

From Table 3.4 and Table 3.5, we observe that varETBR is basically insensitive to the number of variables and the variation range. Here we use the same sampling size ($q = 50$) and reduced order ($p = 10$) for all of the different combinations between number of variables and variation range. And the computation cost of varETBR is the almost same for different number of variables and different variation ranges. This actually is consistent with the observation in PMTBR [56]. One explanation for the insensitivities or nice feature of the new method is that the subspace obtained even with small number of samplings contains the dominant responses Grammian subspaces for the wide parameter and frequency ranges.

Finally, to demonstrate the efficiency of varETBR, we compare it with one recently

proposed similar approach, *StoEKS* method, which employs Krylov subspace reduction with orthogonal polynomials in [42] on the same suite of IBM circuit.

Table 3.6 shows the comparison results where '-' means out of memory error. StoEKS can only finish smaller circuits *ibmpg1* (30k) and *ibmpg2* (120k), while varETBR can go through all the benchmarks (up to 1.6M nodes) easily. The CPU time of StoEKS increases rapidly and could not complete computations as variables count increases. For varETBR, CPU time is independent of number of variables and only depends on the reduced order and number of samples used in the reduced Monte Carlo simulation. Here we select reduced order $p = 10$ and 10000 samples that are sufficient in practice to obtain the accurate probability distribution.

Table 3.6: CPU times (s) comparison of StoEKS and varETBR ($q = 50$, $p = 10$) with 10000 samples for different numbers of variables.

	M = 5		M = 7		M = 9	
Test Ckts	StoEKS	varETBR	StoEKS	varETBR	StoEKS	varETBR
ibmpg1	165	1315	572	1338	3748	1326
ibmpg2	1458	1387	–	1351	–	1377

3.5 Summary

In this chapter, we have proposed a novel scalable statistical simulation approach for large power grid network analysis considering process variations. The new algorithm is very scalable for large networks with a large number of random variables. The new method, called varETBR, is based on the previously proposed extended truncated balanced realization (ETBR) method. To consider the variational parameters, we extend the concept of response Grammian, which was used in ETBR to compute the reduction projection subspace,

to the variational response Grammian. Then Monte Carlo based numerical integration is employed to multiple-dimensional integrals. varETBR only requires the simulation of the reduced circuit using any existing transient analysis method. It is insensitive to the number of variables and variation ranges in terms of computing costs and accuracy, which makes it very general and scalable. Numerical examples, on a number of the IBM benchmark circuits [47] up to 1.6 million nodes, show that the varETBR can be up to $1900X$ faster than the Monte Carlo method, and is much more scalable than the StoEKS method [42, 41]. varETBR can solve very large power grid networks with large numbers of random variables, large variation ranges and different variational distributions.

Chapter 4

hiePrimor:Hierarchical Krylov Subspace Based Reduction of Large Interconnects

Compact modeling of passive RLC interconnect networks has been a research-intensive area in the past decade owing to increasing delays and signal integrity effects and increasing design complexity in today's nanometer VLSI designs. Reducing the parasitic interconnect circuits by approximate compact models can significantly speedup the simulation and verification process in nanometer VLSI designs. As the technology moves to 45nm, the massive extracted post-layout circuits will make the reduction imperative before any meaningful simulations and verifications. Hence the reduction algorithm must be able to scale to attack very large circuit sizes in the current and future technologies.

Reduction algorithms based on subspace projection have been proved to be very effective in the past [14, 69, 29, 49, 77]. Those methods typically project the original circuit into the dimensioned-reduced Krylov subspace to reduce the model order. Krylov sub-

space methods can lead to a localized moment matching link between the original model and the reduced one. It was introduced to the interconnect reduction by the Pade via Lanczos (PVL) [14] method, as it can mitigate the numerical problems in the explicit moment matching methods like Asymptotic Waveform Evaluation (AWE) algorithm [57]. Thereafter, some similar approaches such as Arnoldi Transformation method [69] was also proposed. Later, the congruence transformation method [29] and PRIMA [49] were further proposed, which produce passive models. At the same time, many other approaches also have been proposed, such as balanced truncation based reduction methods [54, 80, 82], local node reduction methods [65, 66] and general node reduction method [60, 76]. But Krylov subspace based-reduction method remains a viable approach for many practical interconnect reduction problems owning to its high efficiency. Existing projection-based reduction methods, however, lack a general way to exploit the parallel computing capabilities, which become more popular with emerging multi-core computing architectures.

But in this chapter, we investigate the parallelism within the reduction operations in one expansion point for one large interconnect circuit. Grimme has explored the parallel computation for multi-point Krylov based reduction where each Krylov subspace from each expansion point can be computed in parallel [21]. Hierarchical reduction of interconnects have also been studied from different perspectives in the past. In HiPRIME algorithm [32], hierarchical reduction has been extended in the extended Krylov subspace method (EKS) to compute the responses of on-chip power grid networks. The HiPRIME method reduces both system and input signals at the same time in a hierarchical way, but it does not produce a reduced model for general use. In the RecMOR method [15], Feldmann and Liu applied the combined terminal and model order reduction on the subcircuits based on the observation that partitioning may lead to many circuits with many new terminals, which will

affect the efficiency for projection-based reduction methods. However, terminal reduction in general still remains a difficult problem and may not be effective for many practical problems [54, 39].

In this chapter, we propose a new hierarchical Krylov subspace based reduction method. The new method combines the partitioning strategy and the Krylov subspace method to speed up the reduction process. It is more suitable for reducing many large global interconnects like coupled bus, transmission lines and large clock nets where the number of ports are general not significant. It is a very general hierarchical model order reduction technique and it works for general parasitic interconnect circuits modeled as RLC circuits.

The new method, called *hiePrimor*, first partitions a large RLC circuit into two or more levels and then perform the projection-based reduction on subcircuits in a bottom-up way. Our contributions are as follows: (1) theoretically we show that if kth order block moment order is preserved in all the reduction processes for all subcircuits and top level circuit, first k block moments will be preserved in the final reduced models; (2) we prove that the new hierarchical reduction method also preserves the passivity of the reduced models for interconnects at all the hierarchical levels; (3) we show that the proposed method not only can exploit parallel computing to speed up the reduction process, but also can significantly improve the analysis capacity by partitioning strategy; (4) we study the impacts of partitioning on the reduction efficiency and show that partitioning is critical for the hierarchical reduction process and min-span or min-cut objective should be attained for best reduction performance. We apply the existing hMETIS partitioning tools [1] to perform the min-cut partitioning.

The proposed method, for the first time, exploits the partitioning-based reduction strategy, which enable the parallel computing and more scalability for handle very large para-

sitic interconnect circuits. Numerical examples show that the proposed method can lead to significant speedup over the flat projection based method like PRIMA and order of magnitudes speedup over PRIMA if parallel computing is used. Interconnect circuits with millions of nodes can be analyzed by hiePrimor in a desktop PC using Matlab in a few minutes.

4.1 Review of subspace projection based MOR methods

In this section, we review the Krylov subspace projection-based methods, which are also used for the new hierarchical projection MOR method.

Without loss of generality, a linear m-port RLC circuit can be expressed as

$$\begin{aligned} C\dot{\mathbf{x}}_n &= -G\mathbf{x}_n + B\mathbf{u}_m \\ \mathbf{i}_m &= L^T\mathbf{x}_n \end{aligned} \tag{4.1}$$

where $\mathbf{x}_n$ is the vector of state variables and n is the number of state variables, m is the number of independence sources specified as ports. C, G are storage element and conductance matrices respectively. B and L are position matrices for input the output ports.

Define $A = -G^{-1}C$, $A \in \Re^{n\times n}$ and $R = G^{-1}B$, $R = [r_0, r_1, ..., r_m]$, $R \in \Re^{n\times m}$. The transfer function matrix after Laplace transformation is $H(s) = L^T(G + sC)^{-1}B = L^T(I_n - sA)^{-1}R$ where I_n is the $n \times n$ identity matrix. The block moments of $H(s)$ are defined as the coefficients of Taylor expansion of $H(s)$ around $s = 0$:

$$H(s) = M_0 + M_1 s + M_2 s^2 + ... \tag{4.2}$$

where $M_i \in \Re^{m\times m}$ and can be computed as $M_i = L^T A^i R$. In the sequel, we use m_i to

denote the terminal count for subcircuit i.

The idea of model order reduction is to find a compact system of a much smaller size than the original system. The Krylov subspace based method accomplishes this by projecting the original system on a special subspace which spans the same space as the block moments of the original system. Specifically, the block Krylov subspace is defined as

$$\begin{aligned} Kr(A,R,q) &= colsp[R, AR, A^2R, ..., A^{k-1}R, \\ &\quad A^k r_0, A^k r_1, ..., A^k r_l] \qquad (4.3) \\ &\quad k = \lfloor q/m \rfloor, \qquad l = q - km. \qquad (4.4) \end{aligned}$$

For simplicity of expression, we assume $q = m \times k$ in the following and k is the order of block moments used in the Krylov subspace. i.e. k order block moments will be matched if Krylov subspace $Kr(A, R, mk)$ is used. Then, projection MOR method tries to find orthogonal matrix $X \in \Re^{n \times q}$ such that $colsp(X) = Kr(A, R, q)$. With

$$\tilde{C} = X^T C X \qquad \tilde{G} = X^T G X$$
$$\tilde{B} = X^T B \qquad \tilde{L} = X^T L$$

the reduced system of size q is found as

$$\begin{aligned} \tilde{C}\dot{\tilde{\mathbf{x}}}_n &= -\tilde{G}\tilde{\mathbf{x}}_n + \tilde{B}\mathbf{u}_m \\ \mathbf{i}_m &= \tilde{L}^T\tilde{\mathbf{x}}_n \end{aligned} \qquad (4.5)$$

The reduced transfer function become $\tilde{Y}(s) = \tilde{L}^T(\tilde{G} + s\tilde{C})^{-1}\tilde{B}$. An important result for projection-based MOR methods is that the reduced system approximates the original

systems in terms of moment matching: if $Kr(A, R, q) \subseteq span(X)$, then the reduced transfer function $\tilde{Y}(s)$ and the original transfer function $H(s)$ matches the first k block moments where $k = q/m$. Also when $L = B$, the reduction process preserves passivity.

4.2 Hierarchical projection MOR method: hiePrimor

4.2.1 A walkthrough example

We introduce our method by using an illustrative RC example circuit shown in Fig. 4.1. This circuit has been partitioned into three parts, the two subcircuits I and II and the top part, which connects the two subcircuits. The two subcircuits are connected via the top level circuit only.

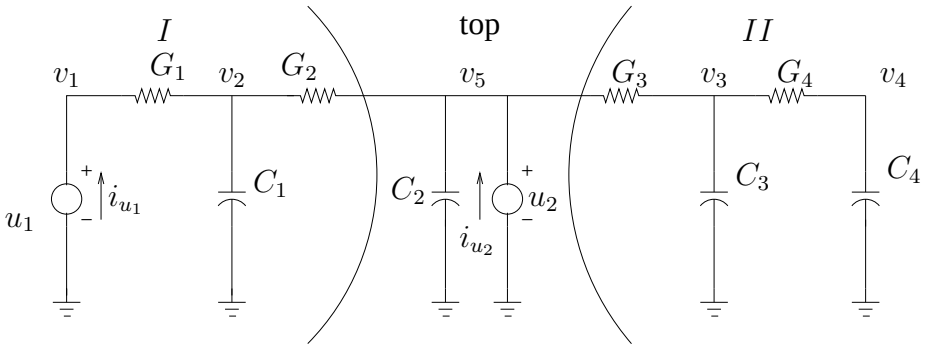


Figure 4.1: A partitioned RC circuit.

As a result, we have the partitioned MNA equations as shown in (4.6), where we partition the matrix into three parts and the input sources into two parts as input sources only appear in partition I and the top-level partition.

$$
\left[\begin{array}{ccc|cc|cc}
G_1 & -G_1 & -1 & 0 & 0 & 0 & 0 \\
-G_1 & G_1+G_2 & 0 & 0 & 0 & -G_2 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & (G_3+G_4) & -G_4 & -G_3 & 0 \\
0 & 0 & 0 & -G_4 & G_4 & 0 & 0 \\
\hline
0 & -G_2 & 0 & -G_3 & 0 & (G_2+G_3) & -1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}\right]
\left[\begin{array}{c} v_1 \\ v_2 \\ i_{u_1} \\ \hline v_3 \\ v_4 \\ \hline v_5 \\ i_{u_2} \end{array}\right] +
$$

$$
\left[\begin{array}{ccc|cc|cc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & C_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & C_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & C_4 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & C_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
\left[\begin{array}{c} \dot{v}_1 \\ \dot{v}_2 \\ \dot{i}_{u1} \\ \hline \dot{v}_3 \\ \dot{v}_4 \\ \hline \dot{v}_5 \\ \dot{i}_{u2} \end{array}\right] =
\left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ \hline 0 & 0 \\ 0 & 0 \\ \hline 0 & 0 \\ 0 & 1 \end{array}\right]
\left[\begin{array}{c} u_1 \\ \hline u_2 \end{array}\right] \tag{4.6}
$$

$$
\left[\begin{array}{c} i_{u_1} \\ \hline i_{u_2} \end{array}\right] =
\left[\begin{array}{ccc|cc|cc} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] x_n
$$

$$
x_n^T = [v_1 \; v_2 \; i_{u_1} \; v_3 \; v_4 \; v_5 \; i_{u_2}]
$$

In general, we can write a w-way partitioned RLC circuit into the following general

form:

$$\begin{bmatrix} G_1 & 0 & ... & G_{1t}^T \\ 0 & G_2 & ... & G_{2t}^T \\ \vdots & \vdots & ... & \vdots \\ G_{1t} & G_{2t} & ... & G_{tt} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_t \end{bmatrix} + \begin{bmatrix} C_1 & 0 & ... & 0 \\ 0 & C_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & C_{tt} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \\ \vdots \\ \dot{\mathbf{x}}_t \end{bmatrix} = \begin{bmatrix} B_1 & 0 & ... & 0 \\ 0 & B_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & B_{tt} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_t \end{bmatrix} \tag{4.7}$$

where the $\mathbf{x}_i$ is the internal variable vector for partition i and $\mathbf{u}_i$ is the external input vector for partition i. $\mathbf{x}_t$ and $\mathbf{u}_t$ are the variables and external input vectors of the top-level circuit. If there are no external inputs for partition i, then the corresponding columns in the position matrix can be removed as shown in (4.6).

4.2.2 The hiePrimor algorithm

For a general RLC circuit, we can rewrite (4.7) as

$$G\mathbf{x} + C\dot{\mathbf{x}} = B\mathbf{u} \tag{4.8}$$

The idea of hierarchical projection-based reduction is to first perform the reduction using projection MOR method for each subcircuit assuming that the subcircuits are disconnected from the rest of the circuit. After the subcircuits are reduced, we perform the reduction on their parent circuits of the subcircuits until we reach to the top-level circuit. The benefit of doing this is that we can reduce the computation complexity by performing

the reduction on the subcircuits and intermediate circuits and parallelism can be exploited to speed up the reduction process as subcircuits in one hierarchical level can be reduced independently.

To illustrate this idea, we still use the example in Fig. 4.1. To reduce the subcircuit I, we have the following subcircuit matrix:

$$\begin{bmatrix} G_1 & -G_1 & -1 \\ -G_1 & G_1+G_2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ i_{u_1} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & C_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \\ \dot{i}_{u_1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ i_2 \end{bmatrix} \tag{4.9}$$

where i_2 is the current source attached to node 2, which becomes a terminal node now. The added current source is just for reduction propose. Note that the position matrix $B_1 = [0\ 0\ 1]^T$ for this subcircuit has been changed to

$$B_1' = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.10}$$

This modification reflects the fact that the subcircuit I now has two terminal nodes: node 1 and node 2. Notice that all the internal nodes, which are inside a subcircuit and are connected to boundary node at the upper level via a device branch, become the terminal nodes of the subcircuits for the reduction propose (as the case of node 2). If a subcircuit does not have any external input (such as the subcircuit II), all the nodes incident on the boundary nodes will become the terminal nodes for the reduction of the subcircuit. As projection based-MOR method becomes less effective for increasing terminal counts, we should try to minimize the terminal counts of subcircuit. Therefore, the hierarchical

reduction requires the min-span [1] partitioning of the circuit. In this way, we can achieve better reduction performance. We will discuss the partitioning issue in the section 4.4.

After the projection matrix V_1 is computed using (4.9), where V_1 spans the kth order block Krylov subspace. i.e. $V_1 \subseteq Kr(G_1^{-1}B_1, G_1^{-1}C_1, km_1)$, where m_1 is the terminal count of subcircuit 1, we can perform the reduction. But now we need to look at the subcircuit in the context of the whole circuit. From (4.7), for the subcircuit 1, we have

$$G_1\mathbf{x}_1 + C_1\dot{\mathbf{x}}_1 + G_{1t}^T\mathbf{x}_t = B_1\mathbf{u}_1 \tag{4.11}$$

After the reduction, we have

$$\tilde{G}_1\mathbf{z}_1 + \tilde{C}_1\dot{\mathbf{z}}_1 + \tilde{G}_{1t}^T\mathbf{x}_t = \tilde{B}_1\mathbf{u}_1 \tag{4.12}$$

where $\mathbf{x} = V_1\mathbf{z}$, $\tilde{G}_1 = V_1^T G_1 V_1$, $\tilde{C}_1 = V_1^T C_1 V_1$, $\tilde{B}_1 = V_1^T B_1$, $\tilde{G}_{1t} = V_1 G_{1t}$. We remark that we use original B_1 here instead of B_1'. Since the $colsp(B_1) \subseteq colsp(B_1')$, we can use subspace defined in V_1 to perform the reduction.

We repeat the the reduction process on all the subcircuits as mentioned above until we

[1] *span* of cut net is the number of internal nodes that a cut net connects from all the partitions.

end up with the following order reduced system at the top level:

$$\begin{bmatrix} \tilde{G}_1 & 0 & ... & \tilde{G}_{1t}^T \\ 0 & \tilde{G}_2 & ... & \tilde{G}_{2t}^T \\ \vdots & \vdots & ... & \vdots \\ \tilde{G}_{1t} & \tilde{G}_{2t} & ... & G_{tt} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{x}_t \end{bmatrix} + \begin{bmatrix} \tilde{C}_1 & 0 & ... & 0 \\ 0 & \tilde{C}_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & C_{tt} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{z}}_1 \\ \dot{\mathbf{z}}_2 \\ \vdots \\ \dot{\mathbf{x}}_t \end{bmatrix} = \begin{bmatrix} \tilde{B}_1 & 0 & ... & 0 \\ 0 & \tilde{B}_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & B_{tt} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_t \end{bmatrix} \tag{4.13}$$

In this chapter, we only present the results for two level reduction as shown in (4.7). But the proposed method can be trivially extended to more hierarchical levels. We can also rewrite (4.13) as

$$G_r\tilde{\mathbf{x}}_r + C_r\dot{\tilde{\mathbf{x}}}_r = B_r\mathbf{u}_r \tag{4.14}$$

With (4.13), we can continue the reduction by performing the reduction at the top-level circuit using the projection-based reduction method again. Finally, we have the reduced model:

$$\tilde{G}\tilde{\mathbf{x}} + \tilde{C}\dot{\tilde{\mathbf{x}}} = \tilde{B}\mathbf{u} \tag{4.15}$$

where $\tilde{G} = V_t^T G_r V_t$, $\tilde{C} = V_t^T C_r V_t$, $\tilde{B} = V_t^T B_r$. G_r and C_r and B_r are the circuit matrices in (4.14) and $V_t \subseteq Kr(G_r^{-1}B_r, G_R^{-1}C_r, q_t)$, where $q_t = km_t$ and m_t is the terminal count at the top-level circuit.

4.2.3 The algorithm flow for hiePrimor

In this subsection, we summarize the algorithm flow of the hiePrimor method shown in Fig. 4.2.3.

Algorithm 5: Hierarchical Krylov Subspace Projection-based Model Order Reduction Method (hiePrimor)

Input: Circuit matrices G, C, B, reduced order q, partition number w
Output: Reduced matrices $\hat{G}$, $\hat{C}$, $\hat{B}$

---

1. Partition original large circuit into w small subcircuits using hMETIS.
2. Form original circuit matrices as in (4.7).
3. For each subcircuit i, find sub-level projection matrix V_i using Krylov subspace method.
4. Reduce subcircuit matrices
$\tilde{G}_i = V_i^T G_i V_i$, $\tilde{C}_i = V_i^T C_i V_i$, $\tilde{B}_i = V_i^T B_i$, $\tilde{G}_{it} = V_i G_{it}$.
5. Form top-level circuit matrices as in (4.13).
6. Compute top-level projection matrix V_t using Krylov subspace method.
7. $\hat{G} = V_t^T \tilde{G} V_t$,$\hat{C} = V_t^T \tilde{C} V_t$, $\hat{B} = V_t^T \tilde{B}$
8. End

4.3 Moment matching connection

In this section, we analyze the moment matching property of the proposed method. We show that if the kth order block moment is preserved/matched in the reductions for all the subcircuits and for the top-level circuit as well, the final reduced model preserves the first k block moments of the original system.

Assume that we have an interconnected circuit system with the transfer function $H(s)$, which consists of n subcircuits that connects together. Assume that we denote subcircuit i as (G_i, C_i, B_i) and we perform the projection based model order reduction on the subcircuit

i only

$$(\tilde{G}_i, \tilde{C}_i, \tilde{B}_i) = (V_i^T G_i V_i, V_i^T C_i V_i, V_i^T B_i) \tag{4.16}$$

and keep all the other system unchanged. We generate the projection matrix V_i such that

$$V_i \subseteq Kr(A_i, R_i, q_i) \tag{4.17}$$

where $A_i = -G_i^{-1} C_i$, $R_i = G_i^{-1} B_i$ and $q_i = km_i$. Then we have the following result:

Lemma 1 *The resulting interconnected circuit system transfer $\bar{H}_1(s)$, which consists of the order reduced subcircuit $(\tilde{G}_i, \tilde{C}_i, \tilde{B}_i)$ with rest of subcircuits unchanged, matches the first k block moments of $H(s)$.*

The detailed proof of this lemma can be found at [78]. Here we give an intuitive example to explain the Lemma. For instance, we have two connected subsystems A and B with two transfer function $H_A(s)$ and $H_B(s)$, where the outputs of A drive the inputs of B. So the whole system transfer function is $H(s) = H_A(s)H_B(s)$. If we replace $H_A(s)$ with $\hat{H}_A(s)$, which is accurate to qth order of $H_A(s)$. It can be easily see that $\hat{H}(s) = \hat{H}_A(s)H_B(s)$ will be accurate to the qth order of $H(s)$ if we write both $\hat{H}_A(s)$ and $H_B(s)$ in the moment (Taylor's series) form.

For the interconnected circuit system $H(s)$, all of its subcircuits are reduced by the projection based MOR method such that

$$(\tilde{G}_i, \tilde{C}_i, \tilde{B}_i) = (V_i^T G_i V_i, V_i^T C_i V_i, V_i^T B_i), \ i = 1, ..., w \tag{4.18}$$

such that $V_i \subseteq Kr(A_i, R_i, q_i)$, $q_i = km_i$ for all the subcircuits. Based on Lemma 1, we can easily obtain the following result:

Corollary 1 *The resulting interconnected circuit system transfer $\bar{H}_2(s)$, which consists of the order reduced subcircuit, $(\tilde{G}_i, \tilde{C}_i, \tilde{B}_i), i = 1, ..., w$, for all subcircuits, matches the first q block moments of $H(s)$.*

The proof of Corollary 1 can be obtained when we apply Lemma 1 w times to the interconnected circuit system $H(s)$ such that we reduce one subcircuit at a time.

Now we are ready to present the main result regarding the proposed hierarchical model order reduction method, hiePrimor.

Theorem 1 *Given a partitioned RLC circuit defined in* (4.7) *with transfer function $H(s)$, if we perform the projection based reduction on all the subcircuits and then on the top-level circuit such that kth order block moment is preserved in the all reduction processes, the transfer function $\tilde{H}(s)$ of the reduced system in* (4.15) *will match the first k block moments of $H(s)$.*

The proof of the theory is obvious in light of Corollary 1 and the fact the top-level reduction on (4.13) also preserves the kth order block moment. Theorem 1 also indicates that for the hierarchical reduction process, we should always use the same block moment order for all the reduction processes. For the same reduction order k, different subcircuit may have different reduced model sizes as the size of the reduced model is km_i, where m_i is the terminal count of the subcircuit i.

In summary, the proposed hierarchical projection based reduction method, hiePrimor, will have the same accuracy as the flat projection based method if both methods use the same block moment order.

4.4 Circuit partitioning

Partitioning plays an important rule for the performance of the proposed reduction method. The reason is that the nodes that are inside a subcircuit and are incident on the boundary nodes at the top level will become the terminal nodes for subcircuits. The sizes of the reduced models grow linearly with the terminal number of the original circuits in the projection based reduction framework as the size of the reduced model is km_i, where k is the block moment order and m_i is terminal count for subcircuit i. To have smaller sizes of the reduced matrices (thus smaller nonzero elements in the matrices), which will be stamped into the higher-level circuit matrix for further reduction, we need to reduce terminal count of subcircuits as much as possible. This calls for the minimum-span or minimum-cut partitioning to achieve this.

Also the size of subcircuits cannot be too small compared with the number of terminals to have meaningful reduction on subcircuits. As a result, the proposed method is more suitable for very large RLC networks like bus, coupled transmission lines and clock nets with loosely coupled subcircuits.

After partitioning, the subcircuit terminals generated by partitioning will be driven by current sources in general, which requires the subcircuit has DC path for all the nodes. If this is not the case, we have to introduce voltage sources at the terminal for the reduction purpose (to make the subcircuit G_i non-singular). This will add more interface terminals to the original subcircuits. As a result, we should minimize the capacitive cut, which can lead to non-DC path nodes. But the proposed method does not have any restrictions on types of boundary nodes.

To meet the partitioning requirement, we apply hMETIS partition tool suite [1], which employs the hierarchical partitioning strategy and is the best min-cut partitioning tools

available. Specifically, we abstract a circuit netlist into a hypergraph, where components (such as resistors, capacitors, inductors, etc.) are considered as vertices in abstracted hypergraph, and nodes in circuit netlist are considered as hyperedges. Then we use hMETIS partitioning suites of the hypergraph partitioning to partition the original large circuit into several small subcircuits.

hMETIS can balance the sizes of each partition automatically without any change to its cost function. In the experiments, we set 2-level partitioning: one top-level circuit and many second-level subcircuits. hMETIS tool suite is very efficient for partitioning very large networks. With hMETIS, the hiePrimor is able to reduce very large interconnect circuits with millions of nodes in a PC using Matlab.

For very densely coupled circuits, the proposed method can still be applied. First, for the capacitively coupled circuits, it is well known that the coupling is more localized, which means the coupling can be further reduced without loss of much accuracy. For inductive coupling, which has long-range effects owning to partial inductance formation [61], many methods have been proposed to reduce the coupling using various window-based truncation techniques [31, 13, 84] before the hierarchical reduction.

4.5 Numerical examples

In this section, we report the experiment results of hiePrimor on some interconnect circuits. We compare it with PRIMA [49] with and without parallel computing settings. We implement the hiePrimor method and PRIMA using Matlab 7.0 and Python. Sparse matrix structures are used in Matlab. Python is used for a parser converting Spice format netlist into Matlab format.

Our test circuits are created based on a bus circuit structure, where each circuit has capacitively-coupled bus lines with different length and each of them are modeled as RC ladder-like circuits. To partition the testing circuits in SPICE format, we transform the netlists into the ones that hMETIS can read and then partition the circuits into several small spice-formatted subcircuits of equal size with the min-cut objective.

We first show that hiePrimor and PRIMA give almost the same accuracy for the given block moment order k (our claim in Section 4.3). We set the reduction order q as $q = n \times k$, where n is the number of ports. Fig. 4.2 (*Ckt 1*, $25K$) and Fig. 4.4 (*Ckt 7*, $1M$) show the frequency responses of Y(1,1) and Y(1,2) from the reduced models by hiePrimor and PRIMA. Fig. 4.3 and Fig. 4.5 show the differences between hiePrimor and PRIMA. In all the test circuits, the accuracy of PRIMA and the hiePrimor are the almost the same numerically, although their results may be a little bit different from the exact one.

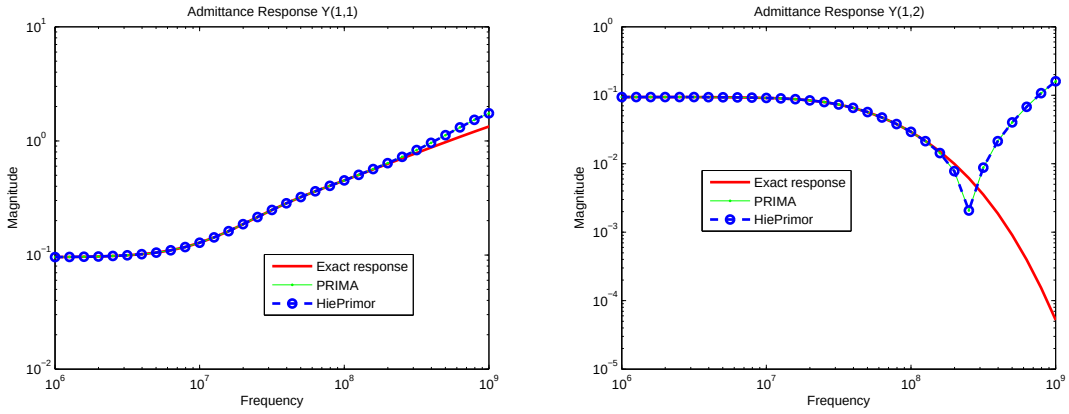


Figure 4.2: Accuracy comparison of PRIMA and hiePrimor in Ckt1 when $k = 4$.

If we increase the value of k, we can obtain the more accurate models. Fig. 4.6 and Fig. 4.7 show the comparison results for *Ckt 1* and *Ckt 7* for $k = 8$. We can see the results are much better than the previous cases when $k = 4$. Notice that the results from hiePrimor and PRIMA are still almost the same again and their sizes after reduction are the same too.

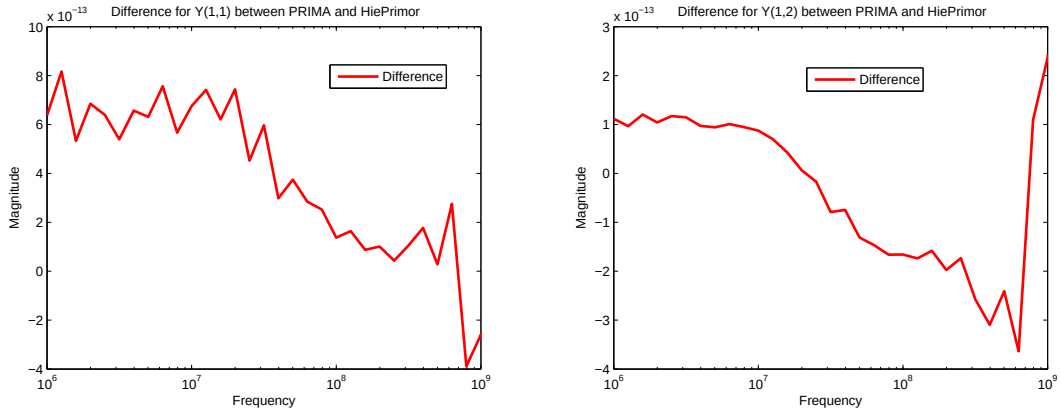


Figure 4.3: Difference between PRIMA and hiePrimor in Ckt1 when $k = 4$.

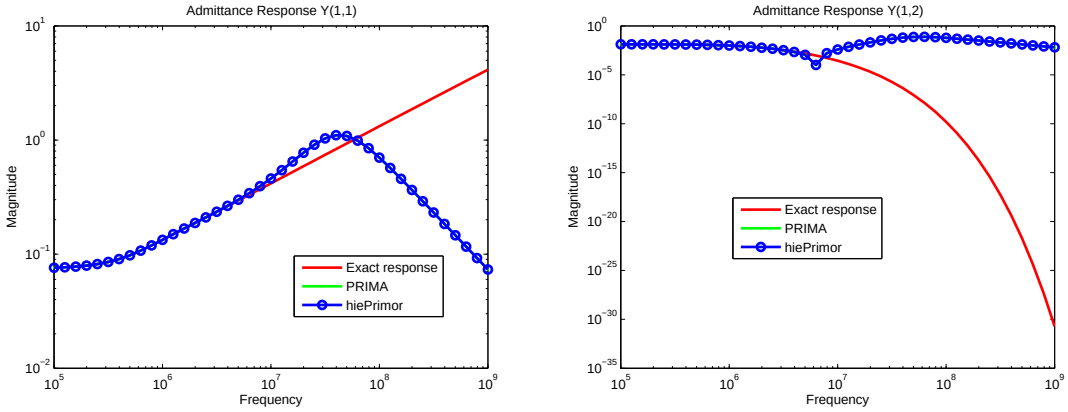


Figure 4.4: Accuracy comparison of PRIMA and hiePrimor in Ckt7 when $k = 4$.

Next, we compare hiePrimor with PRIMA in a single CPU setting in terms of reduction times. Table 4.1 shows the circuit statistics and comparison results of PRIMA and hiePrimor. *#Node* is the number of nodes, *#Sub* is the number of subcircuits, *#Ports* is number of ports (terminals) of the circuit and '-' means out of memory or could not end in a reasonable time. Note that *Ckt1* - *Ckt8* are run on an Intel Xeon 3.0GHz dual CPU workstation with 2GB memory; *Ckt9* and *Ckt10* are run on an workstation with an Intel Xeon Quad-Core CPU (3.0GHz and 16GB memory).

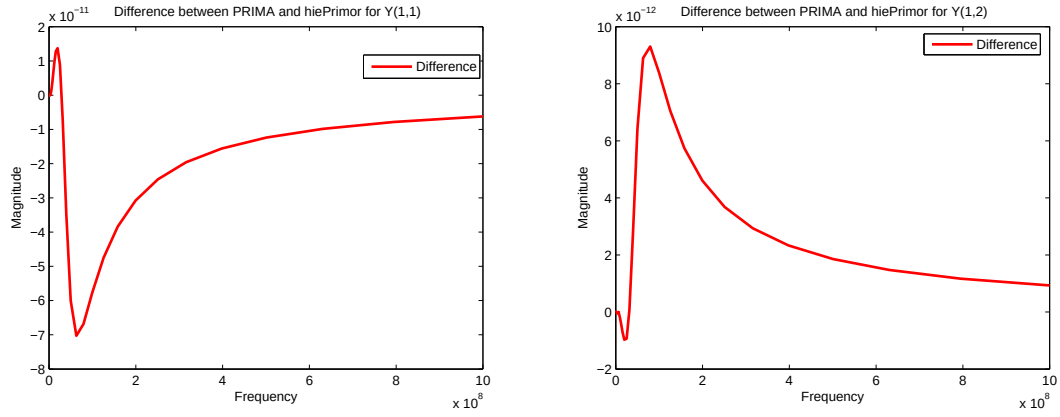


Figure 4.5: Difference between PRIMA and hiePrimor in Ckt7 when $k = 4$.

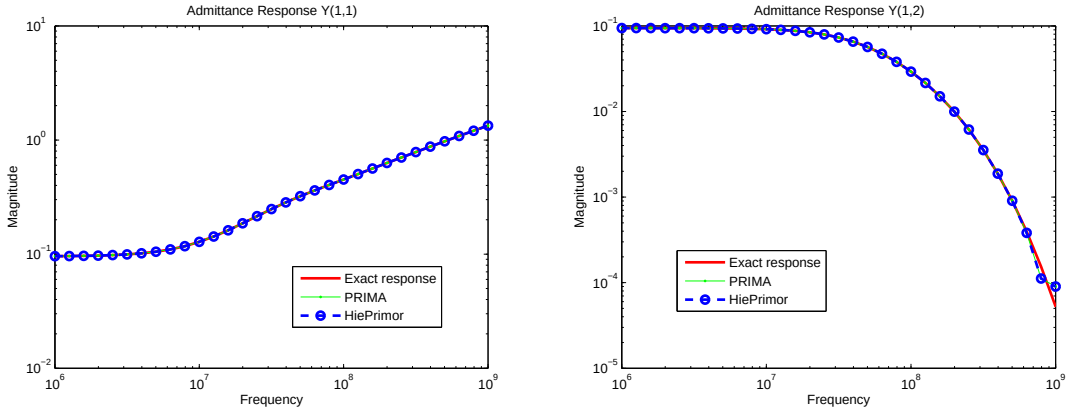


Figure 4.6: Accuracy comparison of PRIMA and hiePrimor in Ckt1 when $k = 8$.

We set the reduction (block moment) order to 4 ($k = 4$) in all the test circuits so that each circuit has the same reduced order (size) after reduction. It may be not accurate enough for $k = 4$ in all the circuits. But given that fact that hiePrimor gives almost the same accuracy as PRIMA, $k = 4$ is sufficient for us to compare the reduction CPU times for them. The last column is the speedup of hiePrimor over PRIMA. We can see that hiePrimor can roughly run $5\times$ faster than PRIMA for large scale circuits. It also shows that the hiePrimor has a better performance when the size of the circuits grow larger. Note that

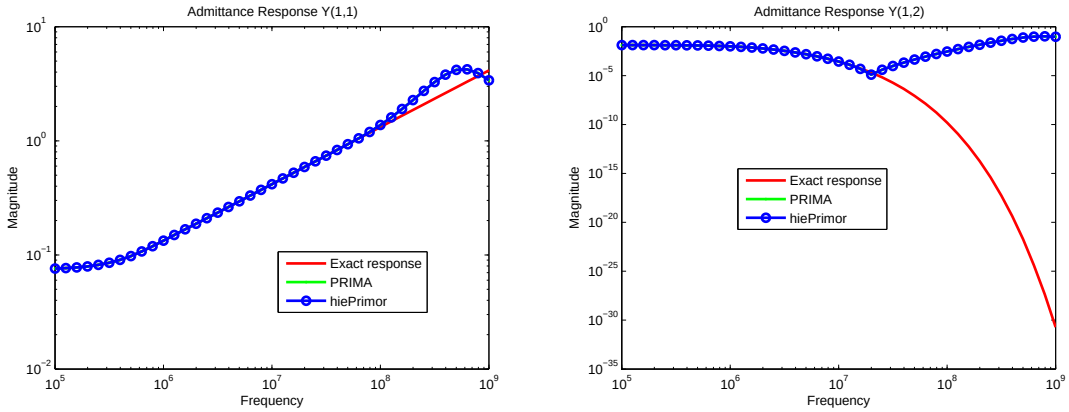


Figure 4.7: Accuracy comparison of PRIMA and hiePrimor in Ckt7 when $k = 8$.

Table 4.1: Reduction time comparison of PRIMA and hiePrimor ($k = 4$, $q = n \times k$).

Test Ckts	#Nodes	w = #Parts	#Ports	PRIMA (s)	hiePrimor (s)	Speedup
Ckt1	$25K$	2	8	5	4	1.25
Ckt2	$50K$	4	16	16	9	1.78
Ckt3	$100K$	8	16	32	13	2.46
Ckt4	$200K$	8	16	69	27	2.56
Ckt5	$500K$	16	24	248	60	4.13
Ckt6	$800K$	16	24	401	99	4.05
Ckt7	$1M$	16	32	863	154	5.60
Ckt8	$1.5M$	16	20	–	176	–
Ckt9	$2M$	32	32	–	136	–
Ckt10	$4M$	32	64	–	305	–

such a speedup is gained without any accuracy loss.

Typically, the more partition number we have, the more speedup is attained. But we also need to consider the cost of combining all the lower level subcircuits into higher level. Also as we get more partitions, the ratio of the terminal node count and the internal node count may get smaller, which may hurt the reduction efficiency as the subcircuits may not be effectively reduced. So number of partitions need to be properly selected practically based on the actual situation. Table 4.2 shows the relationship between the partition number and

the reduction time.

Another observation is that hiePrimor becomes more efficient than PRIMA when the number of ports increases. We use different number of ports for the same circuit (*Ckt7*) using both hiePrimor and PRIMA with the same reduction order. With larger number of ports, hiePimor become faster than PRIMA. Table 4.3 shows the reduction time comparison of PRIMA and hiePrimor for the same large circuit (*Ckt7*) with different number of ports. One reason is that ports are dispersed into subcircuits after partitioning and model order at the top level is already much smaller than the original. While for PRIMA, its time complexity is highly related to the number of ports given the same number of block moments k.

Table 4.2: Reduction time for different numbers of partitions ($k = 4$, $q = n \times k$).

Test Ckts	w = #Parts = 2	#Parts = 4	#Parts = 8	#Parts = 16
Ckt5	116	100	71	60
Ckt6	374	251	128	99
Ckt7	383	298	204	154
Ckt8	675	394	257	176
Ckt9	363	257	200	164
Ckt10	–	886	582	405

Table 4.3: Reduction time for different numbers of ports (Ckt7, $k = 4$, $q = n \times k$).

#Ports	PRIMA	hiePrimor	Speedup
8	189	56	3.38
16	339	96	3.53
32	863	154	5.60

Further, we compare the two methods in the artificial parallel computing settings. It is relatively easy to parallelize our method because each subcircuit can be reduced independently. In parallel computing setting, the running time of hiePrimor is only the sum

Table 4.4: Reduction time comparison of PRIMA and hiePrimor with parallel computing settings ($k = 4$, $q = n \times k$).

Test Ckts	Max Sub (s)	Top (s)	Sum (s)	Speedup
Ckt1	2	0	2	2.50
Ckt2	3	1	4	4.00
Ckt3	3	1	4	8.00
Ckt4	5	1	6	11.50
Ckt5	6	1	7	35.43
Ckt6	10	1	11	36.46
Ckt7	17	3	20	43.15
Ckt8	14	1	15	–
Ckt9	8	1	9	–
Ckt10	19	2	21	–

of the maximum subcircuit level reduction time among all the subcircuits and the top-level reduction time (for two level reduction). The results in Table 4.4 show that we can have one order of magnitude or more speedup if parallel computing is applied. With more levels, it is reasonable to expect more speedup as more parallelism can be exploited.

4.6 Summary

In this chapter, we have proposed a new hierarchical Krylov subspace based reduction method called hiePrimor. hiePrimor combines the partitioning strategy and the Krylov subspace method to speed up the reduction process. It is more suitable for reducing many large global interconnects like coupled bus, transmission lines and large clock nets where the number of ports are general not significant. It is a very general hierarchical model order reduction technique and it works for general parasitic interconnect circuits modeled as RLC circuits. Numerical examples show that the proposed method can lead to significant speedup over the flat projection based method like PRIMA and order of magnitudes

speedup over PRIMA if parallel computing is used.

Chapter 5

ThermPOF: Architecture-level Thermal Characterization For Multi-Core Microprocessors

In this chapter, we propose a new thermal behavioral modeling approach for fast temperature estimation at the architecture level for multi-core microprocessors. The new approach, called *ThermPOF*, builds the transfer function matrix from the measured or simulated thermal and power information. It first builds behavioral thermal models using the generalized pencil-of-function (GPOF) method [24, 25, 64], which was developed in the communication community to build the rational modeling from the given data of real-time and electromagnetism systems. However, the direct use of GPOF does not work for thermal systems. Based on the characteristics of transient chip-level temperature behaviors, we make two new improvements over the traditional GPOF: First we apply a logarithmic-scale sampling scheme instead of the traditional linear sampling to better capture the rapid temperatures change over the long period. Second, we modify the extracted thermal impulse response

such that the extracted poles from GPOF are guaranteed to be stable without accuracy loss. Finally we further reduce the size of thermal models by a Krylov subspace reduction method to further speedup the simulation process [77]. Numerical examples on a practical quad-core microprocessor show that the generated thermal behavioral models can be built very efficiently and the resulting model match the given temperature well.

The proposed method provides a different perspective for thermal modeling. Existing approach like HotSpot [26, 71] can be viewed as a bottom-up approach by considering the internal structures of the architectures of a processor. While our approach is a top-down approach as we only consider the port behaviors of the processors at the architecture level. The two methods in a sense are complementary in their solutions.

The advantage of the proposed method is that it is very simply and cheap to build the models as we only need the measured or computed thermal and power information, we do not need to know the internal structure of the microprocessors at a architecture or other more detailed levels. Its accuracy with respect to the hardware is automatically achieved during the modeling process. Also, the proposed method can be easily extended to consider variable parameters like thermal conductivities, measuring points (heat sink, heat spreader) etc, to build the parameterized thermal models.

In addition to the thermal modeling of the multi-core processor, the proposed method can also be used for many other thermal related design processes. In mobile platforms, it is important to understand the thermal interactions between different power components as they usually share the same cooling solution and thermal envelope. The thermal behavioral modeling will be quite useful to understand these influences by analyzing a variety of power scenarios. This will be almost impossible using experiments or FEM-based methods. Also as systems become smaller and new boundary conditions emerge (e.g. ergonomic lim-

its), the thermal behavioral models will be very useful to better understand the trade-offs between different design conditions.

5.1 Architecture-level thermal modeling problem

We first present the new thermal behavioral modeling problem. Basically we want to build the behavioral model, which is excited by the power input and produces the temperature outputs for the specific locations at the architecture level of the multi-core microprocessor. Our behavioral models are created and calibrated with the measured or simulated temperature and power information from the chips.

Our models are mainly built in the mathematic level and we model the power thermal relationships without regarding many other physical properties (like real poles the system should have) of the multi-core systems. But as far as simulation and verification are concerned, our models can work with any thermal simulators for thermal-related synthesis and optimization. Another benefit of such behavioral thermal models is that it can easily built for many different architectures with different thermal conditions and thermal parameters such as thermal conductivity, thermal cooling configuration, measuring locations (in chip dies, in heat spreader, in heat sinks and other locations), etc. It also has a clear path to build parameterized thermal models with variable parameters.

We remark that the proposed thermal modeling method is a general black-box modeling approach and can be easily applied to thermal modeling of microprocessors and other platform systems at different levels and granularities.

Since the given temperature data are transient and changing over time, we need to capture the transient behavior of the temperature, which can be attained by building an

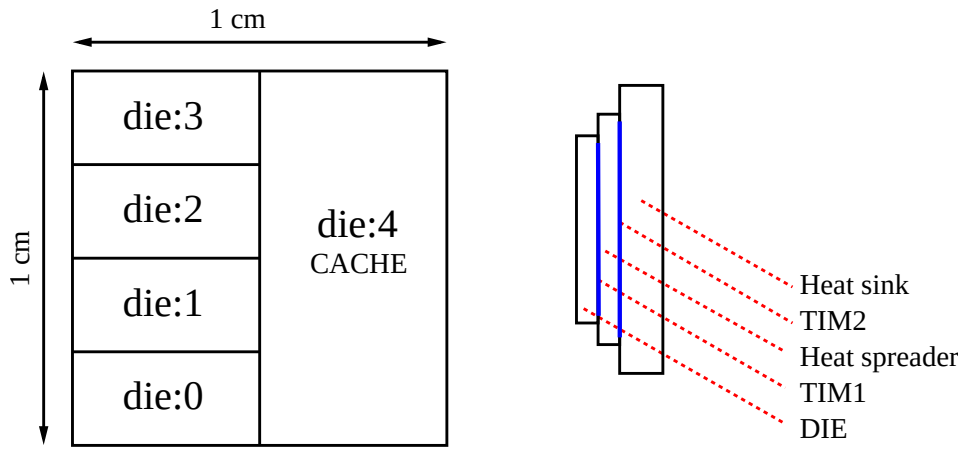


Figure 5.1: The quad-core architecture.

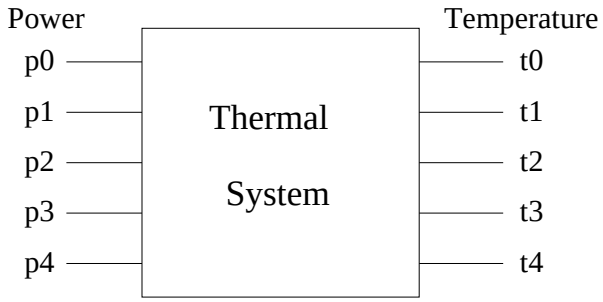


Figure 5.2: The abstracted model system.

impulse response function between temperature and power in the time domain.

In this chapter, we study a quad-core microprocessor architecture from Intel Corporation to validate the new thermal modeling method. The architecture of the multi-core microprocessor is shown in Fig. 5.1, where there are four CPU cores (die 0 to die 3) and one shared cache core (die 4). TIM here stands for thermal interface material. The temperature of each die is reported on the die bottom face in the center of each die. We can abstract this quad-core CPU into a linear system with 5 inputs and 5 outputs as shown in Fig. 5.2 (actually the input p_i and output port t_i will be shared as shown later). The inputs are the power traces of all the cores, and the outputs are the temperatures of them, respectively.

Such a system can be described by the impulse-response matrix-valued function $\mathbf{H}$

$$\mathbf{H(t)} = \begin{bmatrix} h_{00}(t) & h_{01}(t) & h_{02}(t) & h_{03}(t) & h_{04}(t) \\ h_{10}(t) & h_{11}(t) & h_{12}(t) & h_{13}(t) & h_{14}(t) \\ h_{20}(t) & h_{21}(t) & h_{22}(t) & h_{23}(t) & h_{24}(t) \\ h_{30}(t) & h_{31}(t) & h_{32}(t) & h_{33}(t) & h_{34}(t) \\ h_{40}(t) & h_{41}(t) & h_{42}(t) & h_{43}(t) & h_{44}(t) \end{bmatrix} \tag{5.1}$$

where h_{ij} is the impulse response function for output port i due to input port j. So totally we have 25 transfer functions.

Given a power input vector for each core $\mathbf{u}(t)$, the transient temperature vector (at all the ports) can be then computed

$$\mathbf{y}(t) = \int_0^t \mathbf{H}(t-\tau)\mathbf{u}(\tau)d\tau \tag{5.2}$$

Equation (5.2) can be further written in frequency domain as in (5.3).

$$\mathbf{y}(s) = \mathbf{H}(s)\mathbf{u}(s) \tag{5.3}$$

where $\mathbf{y}(s)$, $\mathbf{u}(s)$ and $\mathbf{H}(s)$ are the Laplace transform of $\mathbf{y}(t)$, $\mathbf{u}(t)$ and $\mathbf{H}(t)$, respectively. $\mathbf{H}(s)$ is called the transfer-function matrix of the system where each $h_{ij}(s)$ can be represented as the partial fraction form or the pole-residue form (5.4) as shown below:

$$h_{ij}(s) = \sum_{k=1}^{n} \frac{r_k}{s - p_k} \tag{5.4}$$

where $h_{ij}(s)$ is the transfer function between the jth input terminal and the ith output

terminal; p_k and r_k are the kth pole and residue respectively. Once transfer functions are computed, the transient responses can be easily computed.

We remark that the leakage current depends on the temperature exponentially. High temperature will leads to large leakages current. Such power-temperature dependency should be addressed in the power modeling for better accuracy. But this chapter is mainly focusing the thermal circuit modelings.

The remaining important problem is to find the poles and residues for each transfer function h_{ij} from the given thermal and power information. It turns out that the generalized pencil-of-function can be used for this propose. But we cannot simply apply GPOF method as we show in the Section 5.3. In the following section, we will briefly review the GPOF method before we present our improvements and the new method.

5.2 Review of generalized pencil-of-function method

Generalized pencil-of-function (GPOF) method can be used to extract the poles and residues from the transient response of a real-time dynamic (electrical, electromagnetic) systems [24, 25, 64]. The GPOF method essentially can be viewed as a special general eigen-value decomposition method, which finds the eigen-values of the sampled two non-square matrices from the output of the linear dynamic systems (see (5.16)). The eigen-values actually are the poles of the systems.

Specifically, GPOF can work for such a system that can be expressed in sum of complex exponentials:

$$y_k = \sum_{i=1}^{M} r_i e^{(p_i \Delta t k)} \tag{5.5}$$

where N is the number of sampled points, k = 0, 1, ..., N-1, r_i is the complex residues, p_i

are the complex poles, and Δt is the sampling interval. M is the number of poles used to build the transfer function. Let's define

$$z_i = e^{(p_i \Delta t k)} \tag{5.6}$$

which becomes a pole in Z-plane. For real value y_k, both r_i and p_i should be in complex conjugate pairs. Let's define the new vector of node temperatures (in our problem) as $\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_L$ where,

$$\mathbf{y}_i = [y_i, y_{i+1}, ..., y_{i+N-L-1}]^T \tag{5.7}$$

where L can be viewed as the sampling window size. Based on these vectors, we can define the matrices Y_1 and Y_2 as

$$Y_1 = [\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_{L-1}] \tag{5.8}$$

$$Y_2 = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_L] \tag{5.9}$$

Then one can obtain the following relationship among the Y_1, Y_2 and the pole and residue vectors Z_0 and R based on the structure of Y_1, Y_2:

$$Y_1 = Z_1 R Z_2 \tag{5.10}$$

$$Y_2 = Z_1 R Z_0 Z_2 \tag{5.11}$$

where

$$Z_1 = \begin{bmatrix} 1 & 1 & ... & 1 \\ z_1 & z_2 & ... & z_M \\ \vdots & \vdots & ... & \vdots \\ z_1^{N-L-1} & z_2^{N-L-1} & ... & z_M^{N-L-1} \end{bmatrix} \tag{5.12}$$

$$Z_2 = \begin{bmatrix} 1 & z_1 & ... & z_1^{L-1} \\ & & ... & \\ \vdots & \vdots & ... & \vdots \\ 1 & z_M & ... & z_M^{L-1} \end{bmatrix} \tag{5.13}$$

$$Z_0 = diag[z_1, z_2, ..., z_M] \tag{5.14}$$

$$R = diag[r_1, r_2, ..., r_M] \tag{5.15}$$

So the problem we need to solve is to find the pole and residue vector Z_0 and R efficiently. It turns out that this can be easily computed by observing that

$$\begin{aligned} Y_1^+ Y_2 &= Z_2^+ R^{-1} Z_1^+ Z_1 R Z_0 Z_2 \\ &= Z_2^+ Z_0 Z_2 \end{aligned} \tag{5.16}$$

Hence, the poles are the eigenvalues of $Y_1^+ Y_2$, where $+$ indicate the (Moore-Penrose) pseudo-inverse, as Y_1 is not a square matrix. As a result, one can obtain the Z_0 by using

$$Z = D^{-1} U^H Y_2 V \tag{5.17}$$

where $Z \in C^{M \times M}$ and D, V and U come from the singular value decomposition (SVD) of

Algorithm 6: Generalized Pencil of Function (GPOF)

Input: sampling vectors $\mathbf{y}_i = [y_i, y_{i+1}, ..., y_{i+N-L-1}]^T$
Output: poles vector $\mathbf{p}$ and residues vector $\mathbf{r}$

---

1. Construct matrices Y_1 and Y_2, as in (5.10) and (5.11).
2. Singular value decomposition (SVD) of Y_1, $Y_1 = UDV^H$.
3. Construct matrix Z, $Z = D^{-1}U^H Y_2 V$.
4. Eigen-decomposition of Z, $Z_0 = eig(Z)$.
5. Find poles vector: $\mathbf{p}$, $p_i = \frac{log(z_i)}{\Delta t}$.
6. Solve R_1 and R_2 from $Y_1 = Z_1 R Z_2$ and $Y_2 = Z_1 R Z_0 Z_2$, $Z1$ and $Z2$ are defined as in (5.12) and (5.13).
7. Find residues vector: $\mathbf{r} = \frac{R_1 + R_2}{2}$.
8. End.

Figure 5.3: GPOF algorithm for poles and residues extraction.

Y_1:

$$Y_1 = UDV^H \tag{5.18}$$

Where X^H means taking the conjugate transpose of X. After the Z is computed, we can obtain the pole vector Z_0 by performing the eigen-decomposition of Z, $Z_0 = eig(Z)$, where $eig(X)$ is to obtain the eigenvalue vector from matrix X. Once Z_0 is obtained, we can compute the residue vector R by using either (5.10) or (5.11).

We summerize the GPOF algorithm flow in Fig. 5.3, where N is the total number of sampled points, M is the number of poles used and L can be viewed as the sampling window size.

GPOF works on the sum of exponential forms, which can be represented in the partial fraction form in frequency domain like (5.4). So it can be used to extract poles and residues from the impulse responses for our problem. The GPOF method allows $M \leq L \leq N - M$, which means that we can allow different window sizes and pole numbers. Typically, choosing $L = N/2$ can yield good results. Obviously, more poles will lead more accurate

Algorithm 7: The flow of *ThermPOF* algorithm (extracting one transfer function)

Input: Step input of powers and step response of temperatures
Output: Transfer function in the pole-residue form

1.Calculate impulse response from step response by numerical differentiating.
2. Perform log-scale sampling on impulse response and offset the starting time to zero.
3. Extend the ending time of sampling for stabilizing the poles.
4. Improve the accuracy of computed model by selecting different L.
5. Extract poles and residues of the transfer function by GPOF.

Figure 5.4: The flow of extracting one transfer function.

results. For our problem we find $M = 50$ gives the good results.

5.3 New architecture-level thermal behavioral modeling method

In this section, we present our new thermal behavioral modeling approach based on the GPOF method mentioned in the previous section. We first present the *ThermPOF* algorithm flow. Then explain the several key steps in our proposed method.

5.3.1 The ThermPOF algorithm flow

Now, we describe all the important steps to obtain a transfer function of the thermal system, which is shown in Fig. 5.4.

Step 1 computes the impulse responses from the given step responses. Step 2-4 basically improve the sampling efficiency and stabilizing the models. Step 5 has been explained in the previous section. We will explain the key steps 2-4 in the following subsections.

We need to perform *ThermPOF* for all the transfer functions in our method to obtain

the complete thermal models.

5.3.2 Logarithmic scale sampling for poles and residues extraction

GPOF method should be applied to the thermal impulse response, which in general cannot be obtained directly from measurement or simulation. Instead, we are provided with the thermal step response for each core excited by the power input on the given multi-core microprocessor. Then impulse response can be obtained by performing the numerical differentiation on the step response.

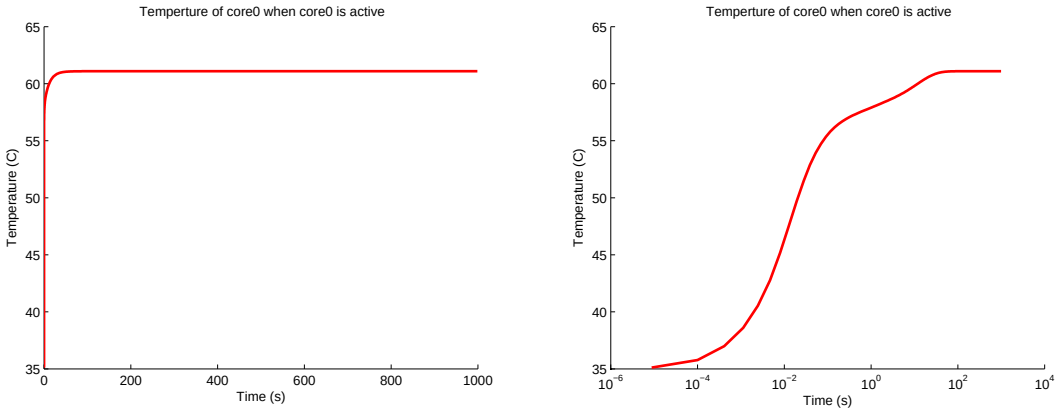
But directly applying the GPOF to the computed thermal impulse response may not lead to stable and accurate model. In the following, we will first show the problems and then present two improvement schemes in *ThermPOF* method such that the resulting model is stable and accurate.

The first problem we face for the thermal modeling is that linear sampling in the traditional GPOF method does not work for our thermal data.

According to GPOF method reviewed in Section 5.2, we know that matrices Y_1 and Y_2 are constructed from the sampled data and that the sampling time interval Δt must be the same. However, how to obtain sample observed temperature changes became a big issue as the step temperature response often goes up rapidly in the first few seconds and gradually tends to reach a steady state after a relatively long time.

This can be illustrated in Fig. 5.5(a), which is step temperature response for $core0$ ($die:0$) when only $core0$ is driven by a step $20W$ power source beginning at $t = 0$ (which is called *active* in this paper). The abient temperature, the initial temperature when no input power at the beginning, is $35°C$. We observe that almost all the temperature increase occurs within the first second, from $35°C$ to $57.9°C$, where $61.1°C$ is the final stable temperature

when $core0$ reaches a steady state after $1000s$ or more.



(a) Linear time scale.

(b) Logarithmic time scale.

Figure 5.5: The transient temperature change of core0 when core0 is excited by 20W power input.

However, the rapid temperature changes and a long observing period lead to problems for the GPOF method if linear sampling is used. The reason is that, to capture the thermal change information, the sampling interval tends to be very small, but this will lead to a very large number of samples owning to the long tail of the thermal response reaching the steady state. As a result, we have to use a very large N and consequently very large L, which causes large dimensions of matrices Y_1 and Y_2 in GPOF. Hence the following matrix operations such as multiplication, inverse or singular value decomposition (SVD) become very expensive.

In this chapter, we propose to use the logarithmic-scale sampling (log-scale sampling for short) to mitigate this problem. For the same temperature response in Fig. 5.5(a), we can obtain the log-sampled temperature response in Fig. 5.5(b), which clearly show how the temperature changes over the log-scale time gradually.

After the time is changed to the logarithmic scale, which is ln(time) and it may become

negative. So we need to offset it to make sure that temperature response always starts at $t = 0$ in the log scale. And the offsetting will not affect GPOF operations. After we obtain the transfer function from GPOF, we need to compute the response in original time scale. We can get the response back by using (5.19),

$$\mathbf{y}'(t) = \mathbf{y}(ln(t) - ln(t_0)) \tag{5.19}$$

where $\mathbf{y}'(t)$ is the response in normal time scale $\mathbf{y}(t)$ is the response in log-scale; t_0 is the offset and usually it is a very small value.

We remark that logarithmic sampling in time and frequency domain has been used in the numerical deconvolution method for RC network extraction in the past [75].

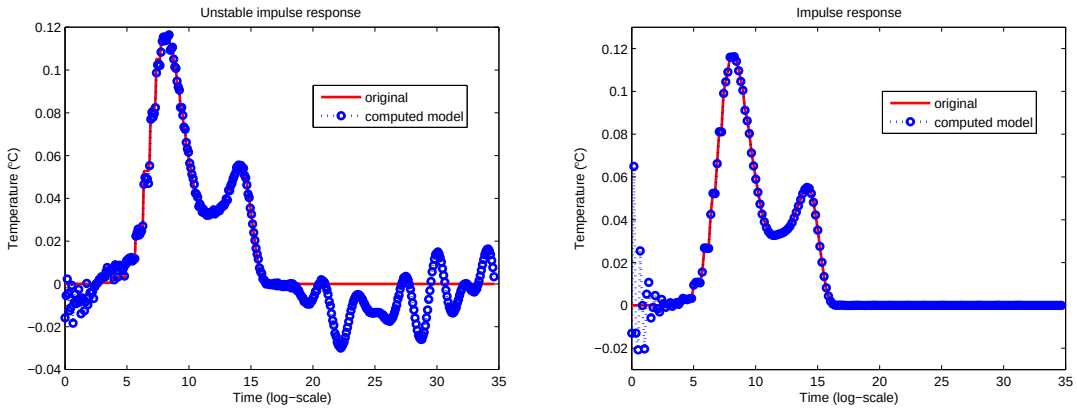
5.3.3 Stable poles and residues extraction

Stable pole extraction

The second problem with the GPOF method is that it will not always generate stable poles for a given impulse response. Actually GPOF model can give a very good matching for a given impulse response for the sampled interval by using positive poles. But outside the sampled interval, the response from the model by GPOF can be unbounded due to the positive poles.

Fig. 5.6(a) shows the extracted impulse response compared to the original one for one of the cores. For this example, the sampled time interval is from 0 to 1000 seconds. Except for the very beginning (we will address this issue later), it can be seen that the computed model matches very well with the original one from time 0 to $1000s$ (the corresponding $x = 18.55$ in log-scaled x-axis with offset being 8.8×10^{-6}). But outside the time interval,

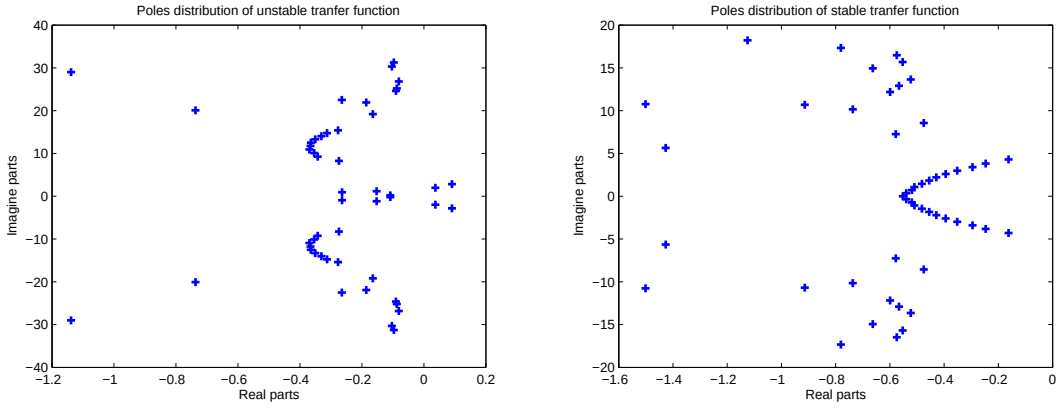
if we extend the time scale to 10^{10} seconds, they are significant differences between the two models. The computed models does not look like an impulse responses and will go unbounded actually owning to the positive poles. Fig. 5.7(a) shows the extracted poles where not all the poles extracted by GPOF are stable (having negative real parts).



(a) Impulse response with positive poles.

(b) Impulse response with only negative poles.

Figure 5.6: Unstable and stable impulse response for Core0.



(a) Existing positive poles.

(b) Only negative poles.

Figure 5.7: Poles distributions of unstable and stable extracted transfer function.

To resolve this problem, we propose to extend the time interval for zero-response time.

For any impulse response, after sufficient time, the response will become zero (or numerically become zero) as the area integration of the impulse curve below is a constant (should be 1 ideally). By sufficiently extending the time interval for zero-response time in an impulse response, we can make all the poles stable. The reason is that if we have positive poles, after sufficient long time, the response will always go non-zero and eventually become unbounded assuming all the poles are different numerically, which is always true practically. If we ensure the zero response for a sufficient long time, all the poles must be stable. The reason is that the response contributed only by those stable poles can decay to zero. Positive poles will lead to unbounded response for a long time interval.

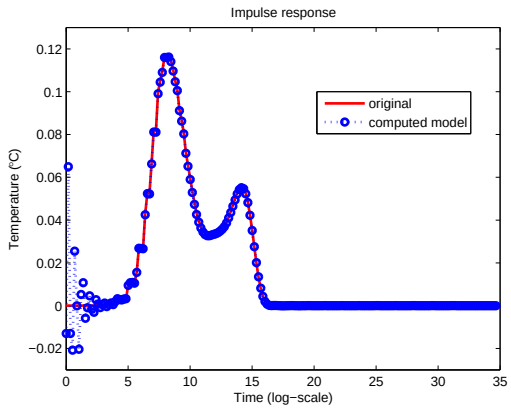
Using the same example, if we extend the time interval to 10^{10} seconds, which actually does not increase significantly in log-scale, all the extracted poles become stable. Fig. 5.7(b) shows the extracted poles by extending zero-response time to $10^{10}s$, where all the poles are stable (with negative real part) and Fig. 5.6(b) shows the extracted stable impulse response. For all our problem, we find $10^{10}s$ seems sufficient for our example. The proposed pole stabilization method can be applied to any stable dynamic system using the pencil of function method. If you sample zero-response sufficient enough, the generated poles will be negative. But the sufficient time is problem dependent. Typically, the new interval should be several order of magnitudes larger than the original time interval.

Stabilizing the starting response

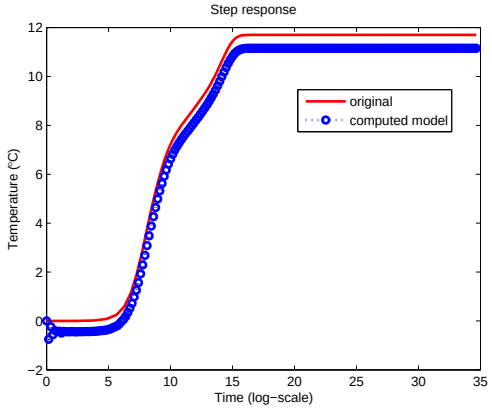
Temperature changes is very slow at the very beginning. As a result, the obtained impulse response may become zero numerically for a short period at the beginning.

However, the zero-response time at beginning may cause the significant discrepancies as shown in Fig. 5.6(b). For example, we consider the temperature of $core1$ when only

$core0$ is active. Assume that $core0$ is active at $t = 0$, in the first very short time, such as $t = 10^{-4}s$, temperature response of $core1$, due to the delay in thermal transmission, is probably still 0 and it may begin to increase at $t = 10^{-3}s$. Normally we consider the difference $10^{-4}s$ and $10^{-3}s$ as a small value in normal time scale, but in the log-scale, this difference is translated to a noticeable period of time with zero responses at the beginning. And long zero-response time at the beginning, however, may cause the significant discrepancies as shown in Fig. 5.8(a), although the computed response tends to be accurate after some time period. This means this transfer function we obtained is not accurate enough. Fig. 5.8(b) shows a step response computed by the transfer function obtained in Fig. 5.8(a). Obviously, it has visible differences compared to the original one.



(a) Impulse response with large errors in the starting time.

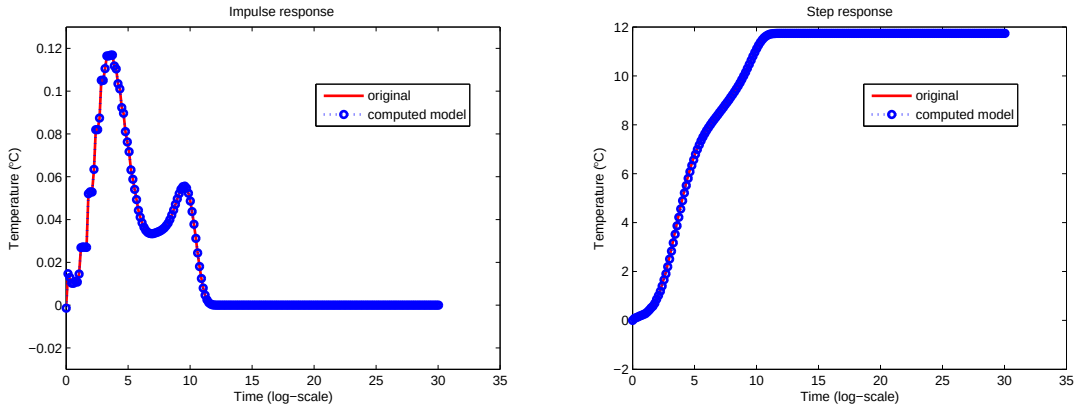


(b) The corresponding step response for the inaccurate model. Here the zero temperature means the room temperature.

Figure 5.8: Impulse and step response computed by inaccurate model with large error in the starting time.

The reason for this problem is that the log-scaled impulse response is different from the impulse response from a physical RLC electronic circuit in which the response goes to non-zero immediately after $t = 0$. To resolve this problem, we may truncate the beginning

zero-response time such that responses go to non-zero numerically immediately. This can be achieved by setting threshold temperature to locate the new zero time. During the simulation process of the model, in all the actual time before the new artificial zero time, the response will be set to zero. Fig. 5.9 shows the impulse and step response computed by accurate model after suppressing the beginning zeros.



(a) Impulse response.

(b) Step response.

Figure 5.9: Impulse and step response computed by accurate model with both improvements.

This problem can also be mitigated by increasing the value of L, which means more sampling points but more accuracy. The advantage of the second method is that we can set up the same offset for all the transfer functions, which can simplify the reduced models. Fig 5.10 shows the improved impulse and step responses from core0 to core1. Here $L = 200$, in contrast to $L = 100$ as used before. Notice that more L may not result in more accurate models. In our experiments, L varies from 150 to 300.

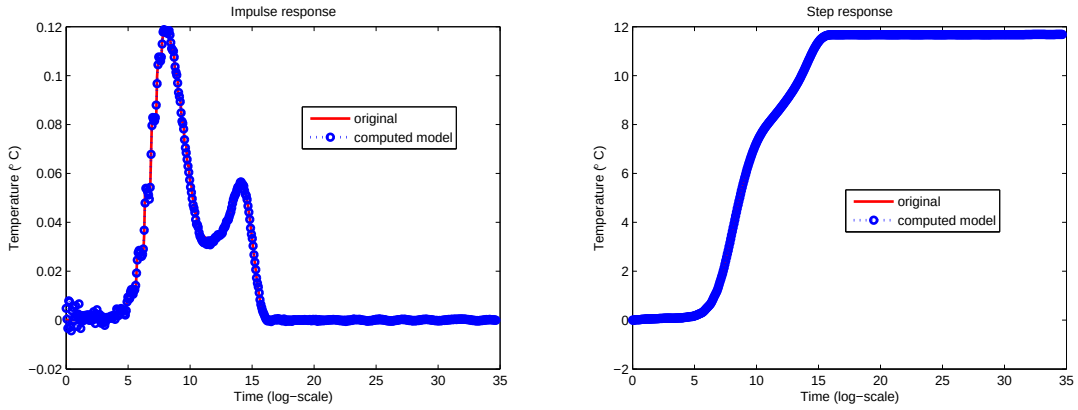


Figure 5.10: Impulse and step response with $L = 200$.

5.3.4 Recursive computation of temperature responses and time complexity

After we obtain the transfer-function matrix $\mathbf{H}$, responses of the system can be computed theoretically for whatever type of inputs. In this chapter, we introduce a fast recursive response computation method. Recursive convolution is a fast convolution with $O(n)$ time complexity instead of traditional $O(n^2)$ time complexity [8], where n is the number of time steps. This method requires to know the poles/residues of the transfer functions.

For our problem, the computation complexity becomes $O(nq)$, where n is the number of time segments or the number of power traces and q is the order of the thermal models (number of the poles in each transfer function). So it can be seen that the simulation time is linear with respect to both model size (number of poles in each transfer function) and the number of the time steps.

In contrast, for traditional integration methods, the time complexity for a $l \times l$ linear matrix, is $O(kl^{\alpha} + l^{\beta})$ where item $O(l^{\beta})$ (typically, $1.1 \leq \beta \leq 1.5$ for sparse circuits) is for the matrix factorization, $O(l^{\alpha})$ (typically, $1 \leq \alpha \leq 1.2$ for sparse circuits) is for solving

one step in transient analysis. The the time complexity is super-linear in general.

In our problem, if the power traces are clock-pulses like as shown in Fig. 5.14, i.e., the power inputs stay the same during one time segment. The recursive convolution can be simplified. Specifically, the power input can be seen as the sum of a group step inputs with different delays. This method works well for the general inputs as long as the time interval is small enough.

Given the power traces and time interval Δt, the response in each time segment not only depends on the current power inputs, but also depends on the previous power inputs. In total there are 3 cases of power inputs changes, as shown in the left part of Fig. 5.11, where T_{n-1} and T_n are two immediate time segments, Δt is the time interval.

Considering powers change at $t = 0$, we would like to compute the temperature $y_n(t)$ at t ($0 \leq t \leq \Delta t$) in the nth time segment T_n. And $y_n(t)$ can be computed as shown in the right part of Fig. 5.11, respectively, where $y_0(t)$ is the step response. The intuitive explanation of the Fig. 5.11 is that the final waveforms at any time point is the sum of step response waveforms generated from all previous power inputs at all the previous time steps. If the input becomes zero, it means adding a negative step response waveform (case 1); if input becomes positive from zero, it means adding a positive step response waveform (case 3). Otherwise, we stay at the same value (case 2). But the recursive convolution [8] can be used to compute any input waveforms in a linear time.

5.4 Reduction of thermal models

After we obtain our thermal models, we can further reduce the order of the models. In this chapter, a classic Krylov-subspace model order reduction method PRIMA [49, 77] is

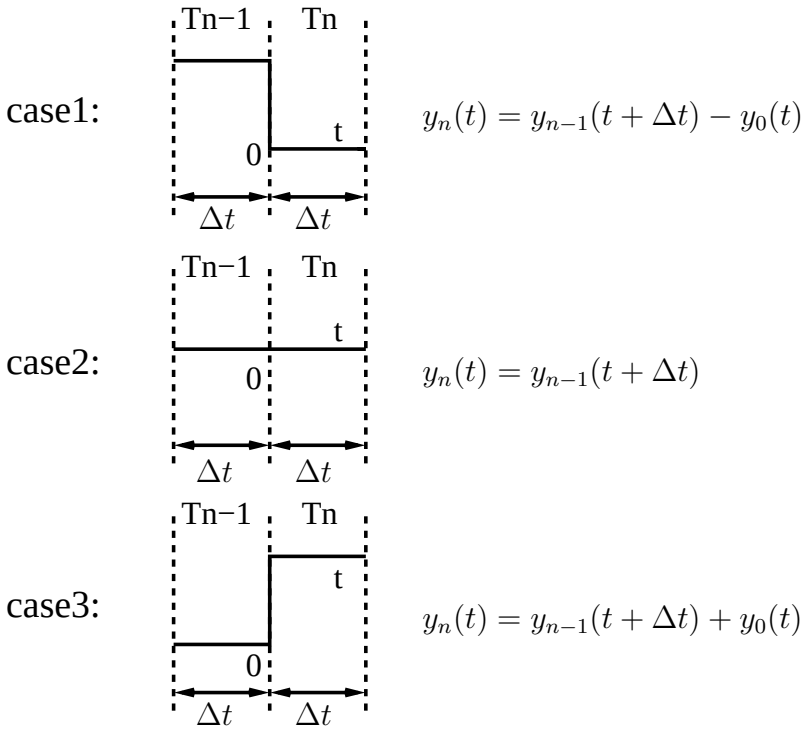


Figure 5.11: The recursive computation step.

used. Before using PRIMA, we need to have the state-space realization (5.20), which can be formed by poles and residues.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u \\ y &= \mathbf{c}^T\mathbf{x} \end{aligned} \tag{5.20}$$

In our model, poles and residues are both complex and appear in conjugate pairs. And for each pair (5.21), the state-space realization is in the form of (5.22) [7]:

$$h(s) = \frac{r}{s-p} + \frac{\overline{r}}{s-\overline{p}} \tag{5.21}$$

where $p = a + bj$ and $r = c + dj$. Let's further define

$$\mathbf{A}_i = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \qquad \mathbf{b}_i = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \qquad \mathbf{c}_i^T = \begin{bmatrix} c & d \end{bmatrix}. \tag{5.22}$$

So we have the state-space realization of (5.23):

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{A}_1 & 0 & ... & 0 \\ 0 & \mathbf{A}_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & \mathbf{A}_n \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} u$$

$$y = \begin{bmatrix} \mathbf{c}_1^T & \mathbf{c}_2^T & ... & \mathbf{c}_n^T \end{bmatrix} \mathbf{x} \tag{5.23}$$

where n is the number of the complex pole pairs in the transfer function and $n = M/2$. Then, the model order reduction is performed:

$$\mathbf{A}_r = \mathbf{V}^T \mathbf{A} \mathbf{V} \qquad \mathbf{b}_r = \mathbf{V}^T \mathbf{b} \qquad \mathbf{c}_r^T = \mathbf{c}^T \mathbf{V} \tag{5.24}$$

where $\mathbf{V}$ is the projection matrix obtained from PRIMA.

After model reduction, we need to extract the poles and residues from the reduced matrix (5.24) to go back to the partial fraction form (pole-residue form). This can be done by means of the eigen-decomposition of $\mathbf{A}_r$ [7], which leads to a diagonal matrix $\mathbf{\Lambda}$ containing eigenvalues and an orthogonal matrix $\mathbf{P}$ formed by the eigenvectors.

Thus, the transfer function of the pole-residue form in (5.4) can be computed as

$$h_{ij}(s) = \sum_{k=1}^{q} \frac{\mu_k \cdot v_k}{s - \lambda_k} \tag{5.25}$$

where λ_k is the kth diagonal element of $\mathbf{\Lambda}$, μ_k is the kth element of $\mathbf{c}_r^T\mathbf{P}$, v_k is the kth element of $\mathbf{P}^{-1}\mathbf{b}_r$ and q is the reduced order.

5.5 Numerical examples

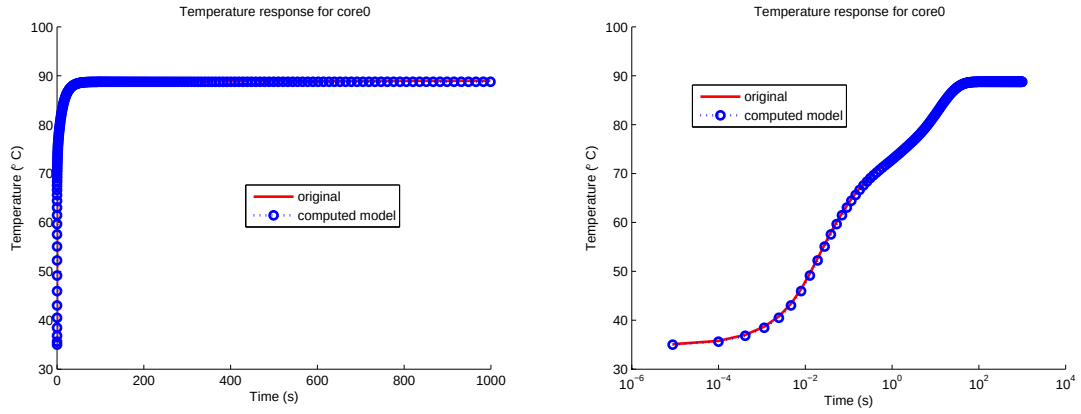
The proposed *ThermPOF* algorithm has been implemented in MATLAB 7.0 and tested on a quad-core microprocessor architecture as shown in Fig. 5.1 from industry partner Intel Corp. We first extracted the transfer function matrix of the system through a training data set, which consists of the step responses for each core from other cores. After generation of the transfer functions, we could validate our models by computing the thermal responses from other non-training power inputs and compare them with known responses.

Our experimental data contain each core's temperatures measured directly from the center of the dies, which are provided by Intel. At the beginning all the cores are in zero state and have an initial ambient temperature $35°C$.

We verify the correctness of our model based on two sets of given thermal data from Intel. First, from $t = 0$ each core is excited by a step power input of $20W$ simultaneously. And the temperature of each core is collected from $0s$ to $1000s$. For each transfer function, we set the order to 50. This is already enough for our model. In practice, temperature on each core or cache can be computed very fast by our model during any time interval as our model is directly based on the transfer function represented by poles and residues instead of state space equations.

We remark that in reality we cannot turn the cores to zero power. For the purposes of the simulations, we turned on and off cores and cache respectively so we can quantify or capture the influences. The power traces used are realistic for the workloads and the on/off scenario for different cores is not.

The results of core0 and cache are shown in Fig. 5.12 and Fig. 5.13 under the normal linear time scale and the log scale respectively. The solid curve represents the measured temperature and the dotted line represents our computed temperature. The simulation runs very fast and costs only few seconds. From these figures, we can see that our models have very good accuracy. Actually, the temperatures of other cores match as well as shown in the following tables.

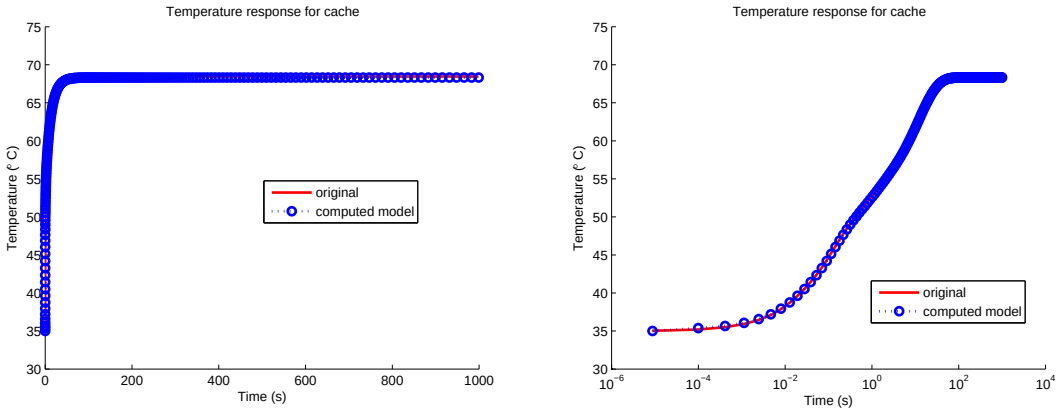


(a) Temperature response of core0 in the linear scale.

(b) Temperature response of core0 in the log scale.

Figure 5.12: Comparison results of core0's temperature when all cores are active (driven by 20W powers).

Table 5.1 shows the temperatures when all the cores achieve the steady state and the error differences in percentage. The difference is only around 0.2%. Furthermore, Table 5.2 shows some statistical features of the differences over all the sampling time points, including the maximums, the means and the standard deviations. Also, the maximum and average



(a) Temperature response of cache in the linear scale.

(b) Temperature response of cache in the log scale.

Figure 5.13: Comparison results of cache's temperature when all cores are active (driven by 20W powers).

percentages are given. From this table we can see that the maximum difference is less than $0.5°C$ and 1% and the average difference is less than $0.3°C$ and 0.3% for all the cores.

Table 5.1: Difference when temperatures achieve the steady state.

	Measured Temp. ($°C$)	Computed Temp. ($°C$)	Difference percentage
Core0	88.96	88.78	0.22%
Core1	90.60	90.52	0.08%
Core2	90.04	88.94	0.11%
Core3	88.96	88.78	0.20%
Cache	68.46	68.32	0.20%

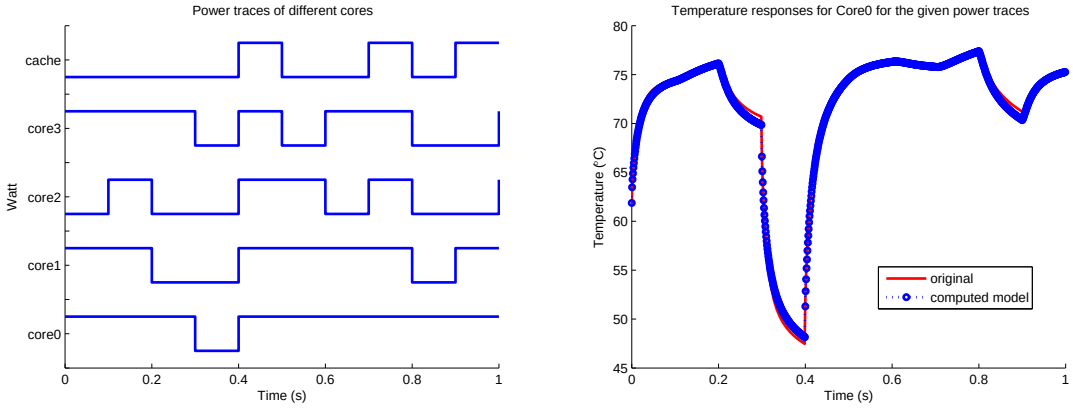
Now we test on the second set of thermal benchmarks also from Intel. The temperature on every core is driven to an initial state by a power of $10W$ on each of the cores for 1000 seconds. Then we start to apply a number of random power input traces as shown in Fig. 5.14(a), where the step power is $20W$ for all the cores.

We verify the correctness of our model from $0s$ to $1s$ based on the given thermal data

Table 5.2: Statistics of the difference between measured and computed temperatures.

	Difference ($^\circ C$)			Difference percentage	
	Maximum	Mean	Std. deviation	Maximum	On average
Core0	0.46	0.25	0.08	0.89%	0.32%
Core1	0.27	0.18	0.07	0.42%	0.15%
Core2	0.37	0.16	0.08	0.73%	0.20%
Core3	0.46	0.24	0.08	0.88%	0.31%
Cache	0.31	0.16	0.08	0.51%	0.26%

from Intel. During that time, power inputs stay the same in each time step of $0.1s$. In practice, temperature on each core can be computed very fast by our model during any time interval, as the computation complexity in our model is only $O(nq)$ by using recursive convolution, where n is the number of time steps and q is the order of the reduced models.

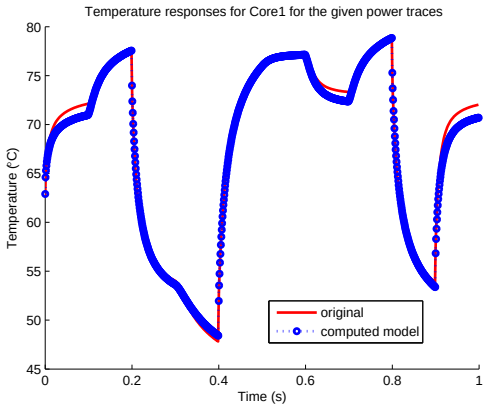


(a) Random power input traces.

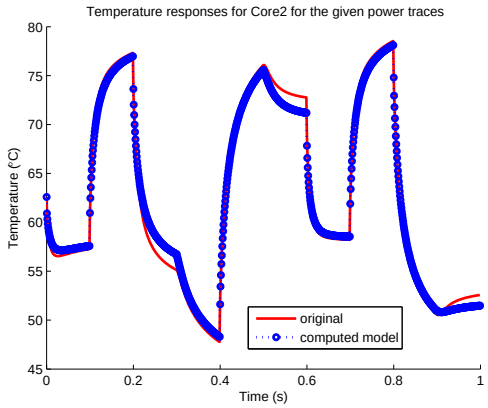
(b) Core0's temperature.

Figure 5.14: Thermal simulation results on given power input traces

Our simulation results for all cores are shown in Fig. 5.14(b), Fig. 5.15 and Fig. 5.16. The simulation runs very fast and costs only few seconds. From the figures, we can see that all the peak temperatures (both the maximum and the minimum) of each core during the whole time interval match well between the models and given measured data. The

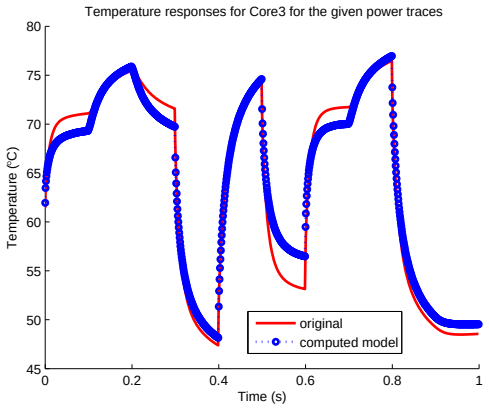


(a) Core1's temperature.

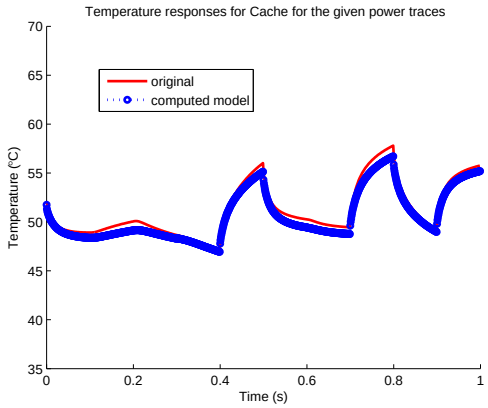


(b) Core2's temperature.

Figure 5.15: Thermal simulation results on given power input traces



(a) Core3's temperature.



(b) Cache's temperature.

Figure 5.16: Thermal simulation results on given power input traces

temperature errors are less than $1°C$, as shown in Table 5.3.

The errors (absolute values) between the original and our computed model are shown in Table 5.4, including the maximums, the means and the standard deviations. The maximal errors of $core0$, $core1$, $core2$ and $cache$ are around $1°C$ or less than $2°C$ at the most. The maximal error of $core3$ is a little bit larger, which occurred during the time interval

Table 5.3: The maximal and minimum error peaks ($M = 50$).

	Maximal peak			Minimum peak		
	Measured ($^\circ C$)	Error ($^\circ C$)	Percentage	Measured ($^\circ C$)	Error ($^\circ C$)	Percentage
Core0	77.27	0.45	0.58%	47.47	0.38	0.79%
Core1	78.86	0.04	0.05%	47.81	0.35	0.73%
Core2	78.55	0.38	0.48%	47.77	0.24	0.51%
Core3	76.48	0.75	0.98%	47.38	0.45	0.95%
Cache	57.80	0.99	1.72%	48.86	0.11	0.23%

Table 5.4: Statistics of the errors between measured and computed temperatures ($M = 50$).

	Error ($^\circ C$)			Error percentage	
	Maximum	Mean	Std. deviation	Maximum	On average
Core0	1.05	0.34	0.23	1.56%	0.50%
Core1	1.67	0.53	0.48	2.44%	0.78%
Core2	1.78	0.61	0.47	2.56%	0.98%
Core3	3.33	1.10	0.82	6.09%	1.80%
Cache	1.05	0.63	0.22	1.84%	1.22%

Table 5.5: The maximal and minimum error peaks and means ($M = 30$).

	Maximal peak		Minimum peak		Mean	
	Error ($^\circ C$)	Percentage	Error ($^\circ C$)	Percentage	Error ($^\circ C$)	Percentage
Core0	0.40	0.52%	0.46	0.96%	0.36	0.48%
Core1	0.12	0.15%	0.49	1.00%	0.47	0.69%
Core2	0.06	0.07%	0.34	0.70%	0.56	0.88%
Core3	0.76	0.98%	0.53	1.11%	1.11	1.66%
Cache	1.01	1.78%	0.01	0.02%	0.03	1.25%

$0.5s - 0.6s$. But the standard deviations of the errors of all cores show that the temperature variations on average are less than $1^\circ C$. Note that all the errors here are the absolute values of measured temperatures minus our computed temperatures.

Further we show the speedup gained in the simulation by the model reduction techniques presented in Section 5.4. In our experiments, we first set the order $M = 50$. After reduction, the order is reduced to $M = 30$ without loss of accuracy. All the errors are less

than $< 2\%$ compared to the exact ones, as shown in Table 5.5. We know that simulation running time $O(nq)$, q is the size or order of the model. So from order 50 to 30, we can approximately reduce 40% simulation time, as shown in Table 5.6. We also set up order $M = 100$ at first, but the reduced order is still 30 if we want to maintain the same high accuracy.

Table 5.6: Speedup when $M = 30$ compared to $M = 50$.

	Run time (s) when $M = 50$	Run time (s) when $M = 30$	Time reduced
Core0	1.31	0.80	38.9%
Core1	1.29	0.78	39.5%
Core2	1.28	0.78	39.1%
Core3	1.28	0.78	39.1%
Cache	1.30	0.79	39.2%

5.6 Summary

In this chapter, we have proposed a new thermal behavioral modeling approach for fast temperature estimation at the architecture level for multi-core microprocessors. The new approach, called ThermPOF, builds the transfer function matrix from the measured or simulated thermal and power information. It first builds behavioral thermal models using the generalized pencil-of-function (GPOF) method [24, 25, 64], which was developed in the communication community to build the rational modeling from the given data of real-time and electromagnetism systems. However, the direct use of GPOF does not work for thermal systems. Based on the characteristics of transient chip-level temperature behaviors, we make two new improvements over the traditional GPOF: First we apply a logarithmic-scale sampling scheme instead of the traditional linear sampling to better capture the rapid tem-

peratures change over the long period. Second, we modify the extracted thermal impulse response such that the extracted poles from GPOF are guaranteed to be stable without accuracy loss. Finally we further reduce the size of thermal models by a Krylov subspace reduction method to further speedup the simulation process [77]. Numerical examples on a practical quad-core microprocessor show that the generated thermal behavioral models can be built very efficiently and the resulting model match the given temperature well.

Chapter 6

ParThermPOF: Parameterized Architecture-Level Dynamic Thermal Models For Multi-Core Microprocessors

In this chapter, we propose a new architecture-level parameterized dynamic thermal behavioral modeling algorithm for emerging thermal-related analysis and optimization problems for high-performance chip-multiprocessor design. The proposed compact thermal model will be used to predict the thermal response of new package designs once its accuracy has been calibrated and validated with the detailed models. This is the design methodology to be used by our industry partner. We propose a new approach, called ParThermPOF, to build the parameterized dynamic thermal behavioral models from accurately computed thermal and power information using the sophisticated FEA (Finite Element Analysis) or CFD (Computational Fluid Dynamics) tools at architecture level. The new method is a top-down, black-box approach, meaning it does not require any internal structure of the systems and it is very general and flexible. ParThermPOF is able to include a number of

parameters such as location of thermal sensors in a heat sink, different components (heat sink, heat spreader, core, cache, etc.), thermal conductivity of heat sink materials, etc. The new method consists of two steps: first, a Response Surface Method (RSM) based on low-order polynomials is applied to build the parameterized models at each time point for all the given sampling nodes in the parameter space (except for time). Second, an improved Generalized Pencil-Of-Function (GPOF) method [36, 34], specifically for thermal modeling, called ThermPOF, is employed to build the transfer-function-based models for each time-varying coefficient of the polynomials generated in the first step.

We remark that the detailed model for generating the realistic temperatures for training compact models was developed in *FloTHERM* [18], which is a typical 3D Computational Fluid Dynamics (CFD) commercial software used in cooling of electronics. Also, the material properties and boundary conditions are representatives of conditions found in the thermal test vehicles at Intel's lab rather than real-life systems. The specific workloads only mimic those of real-life applications. The model does not include the specifics of the board. The power distribution in the real die is known in advance (for example from Thermal Test Vehicles (TTVs)). The detailed model only tries to mimic these profiles. Although sufficient detail was put in the model, the focus was to include the relevant parameters that needed to be calibrated with parameterized methods.

Simulation results on a practical quad-core microprocessor show that the generated parameterized thermal behavioral models can be built very efficiently and the temperatures computed from resulting models match the given temperatures well for given parameter space in the time domain. The compact models by ParThermPOF offer two order of magnitudes speedup over the commercial thermal analysis tool FloTHERM [18] on the given examples from our industry partner.

6.1 Parameterized Transient Thermal Behavioral Modeling Problem

Two types of parameters are considered in our modeling problems. The first one is time; the second one are other parameters to be discussed below.

Our modeling problem is to build parameterized transient thermal models considering the both time and other variable parameters of multi-core processors. Basically we want to build the behavioral model, whose inputs are the powers and outputs are temperatures that not only depend on the input powers but also depend on the system parameters. Our parameterized behavioral models are created and calibrated with the simulated temperature and power information using a commercial thermal analysis tool based on a realistic multi-core processor.

In this paper, we specifically look at a quad-core microprocessor architecture from Intel to validate the new thermal modeling method. The architecture of this multi-core microprocessor is shown in Fig. 6.1, where there are four CPU cores (die 0 to die 3) and one cache core (die 4). The temperatures reported are on the die bottom face and centered with each die region.

Fig. 6.2 shows 3-D structure of this quad-core microprocessor in a package, where the CPU die (with quad-cores) is in the bottom in contact with intermediate heat spreader (IHS). At the top is the heater sink (HS), which has the top and bottom parts. The thermocouples (thermal sensors) are used to measure the temperatures on these specific locations. Fig. 6.3 shows the temperature changes when only core0 is active ($20W$ power at the input) at difference locations using a copper heat sink. HS_$5mm$ means that the temperature changes at $5mm$ away from the center of the heat sink. As we can see, temperatures go

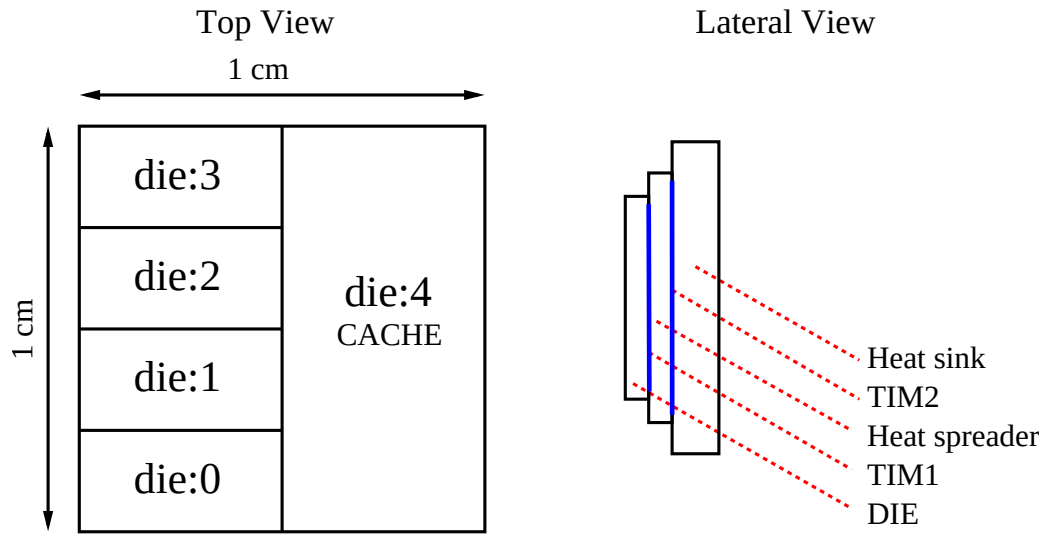


Figure 6.1: Quad-core architecture

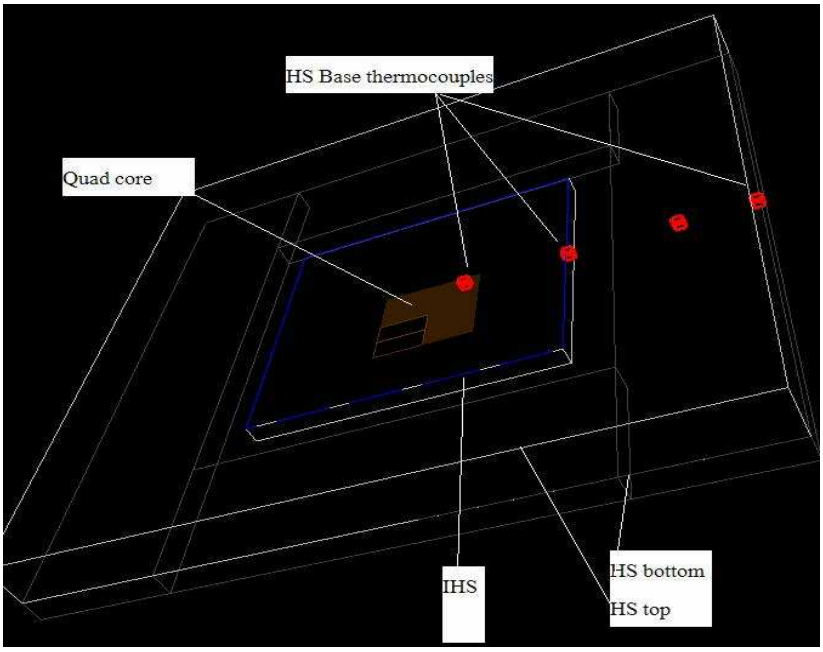
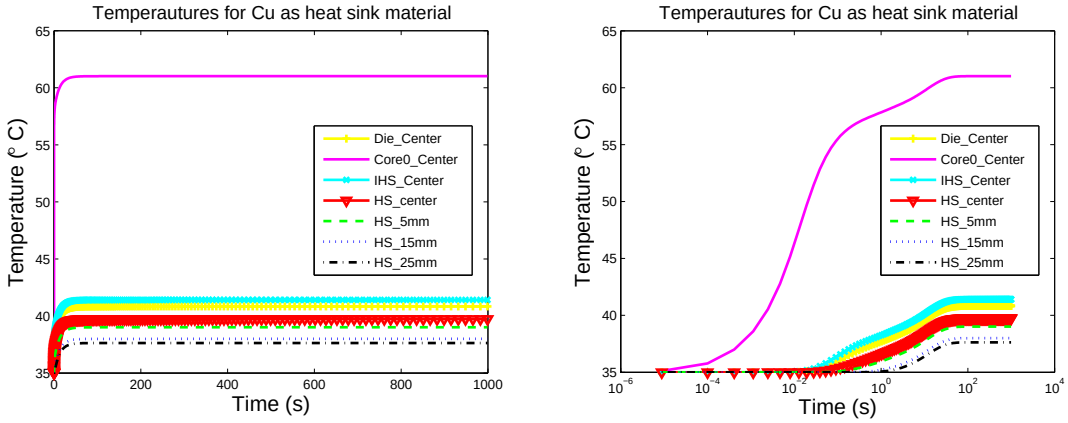


Figure 6.2: 3D structure of quad-core processor

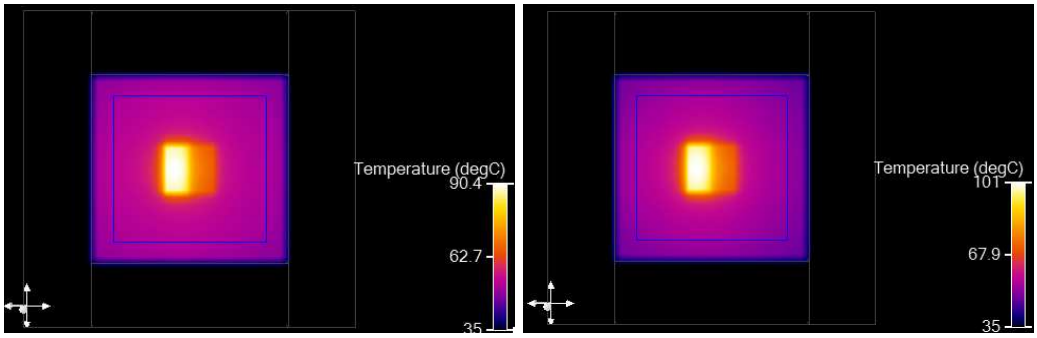
down as we move away from the center and away from the bottom parts of the chip. The temperatures at the core center are hottest. Also, in addition to the distance parameter in a specific component (heat sink), we may select different observation components such as individual core, cache, heat spreader or heat sink as other indicator parameters.



(a) Temperature responses in normal time scale.

(b) Temperature responses in log time scale

Figure 6.3: Temperature responses at various locations in quad-core processor when only core0 is active.



(a) Using copper heat sink

(b) Using aluminium heat sink

Figure 6.4: Temperature distributions on the whole chip with package using different heat sink materials when all cores and cache are active.

Furthermore, we may consider the thermal conductivity of the heat sink material as another parameter. Normally, heat sink is made of copper (Cu) or aluminium (Al). Cu and Al have different thermal conductivities, such as Cu is $390W/(m \cdot K)$ and Al is $240W/(m \cdot K)$. Different heat sink materials may induce the different temperature distributions on the chip. Fig. 6.4 shows the simulated temperature distributions on the whole chip with package using copper heat sink and aluminum heat sink respectively. The supply power for both cases

is about 40W. We can see from the two figures that the maximum temperature of the chip on a copper heat sink is $10°C$ less than the one on an aluminum heat sink due to different thermal conductivities of the two materials. But the prices of copper and aluminum are different. So the designers need a trade-off between hot spot temperatures and package cost. In our work, we set up such a parameter to indicate the thermal conductivity of the heat sink material properties. Such parameterized thermal models may be very helpful for the design exploration and optimization.

We can abstract this quad-core processor into a linear system with 5 inputs, 1 output and several parameters as shown in Fig. 6.5. The inputs are the power traces of all the cores and the output is the temperature response for given parameters. The parameters can be the location of the thermal sensors (distance to a center point), the observation component or measure point, thermal conductivity of the materials used for heat sinks, etc.

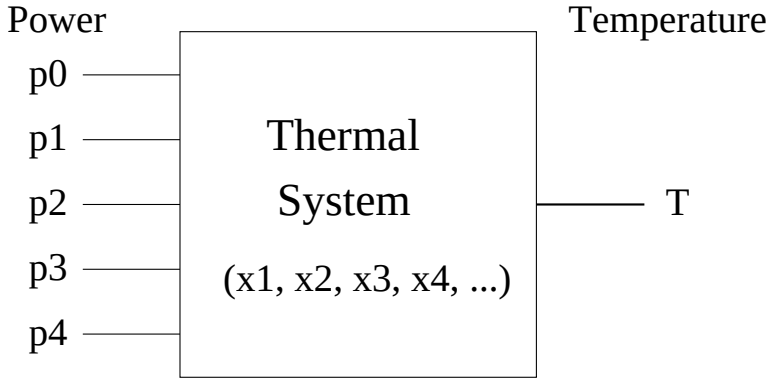


Figure 6.5: Abstracted system

Such system can be described by the parameterized impulse-response (transfer) function matrix $\mathbf{H}$

$$\mathbf{H}(t, \boldsymbol{\xi}) = [h_0(t, \boldsymbol{\xi})\ h_1(t, \boldsymbol{\xi})\ h_2(t, \boldsymbol{\xi})\ h_3(t, \boldsymbol{\xi})\ h_4(t, \boldsymbol{\xi})] \tag{6.1}$$

where h_i are the impulse response function for output due to input port i and $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3, ...)$ are the parameters of this system.

Given a power input vector for each core $\mathbf{u}(t)$ and a specific set of parameters $\boldsymbol{\xi}$, the transient temperature can be then computed by

$$\mathbf{y}(t) = \int_0^t \mathbf{H}(t-\tau,\xi)\mathbf{u}(\tau)d\tau \tag{6.2}$$

Equation (6.2) can be written in frequency domain as in (6.3).

$$\mathbf{y}(s) = \mathbf{H}(s,\xi)\mathbf{u}(s) \tag{6.3}$$

where $\mathbf{y}(s)$, $\mathbf{u}(s)$ and $\mathbf{H}(s,\xi)$ are the Laplace transform of $\mathbf{y}(t)$, $\mathbf{u}(t)$ and $\mathbf{H}(t,\xi)$, respectively. Each h_i can be expressed as the partial fraction form or the pole-residue form (6.4) [58]:

$$h_i(s,\xi) = \sum_{k=1}^{n} \frac{r_k(\xi)}{s - p_k(\xi)} \tag{6.4}$$

where $h_i(s,\xi)$ is the transfer function between the ith input terminal and the output terminal; p_k and r_k are the kth pole and residue. Once transfer functions are obtained, the transient responses can be easily computed.

To build the parameterized behavioral model, we need to solve the following two problems: (1) to find a response polynomial functions that can approximate the given temperatures for all the controllable variables (parameters) with enough accuracy; (2) to find the poles and residues for each transfer function h_i from thermal coefficients (of the polynomials from step (1)) and power information to capture the transient behavior of the temperature.

For problem (1), we introduce response surface method to capture linear or nonlinear relationship between the parameters and response (temperatures) at each time point. For

problem (2), we can handle it by using improved generalized pencil-of-function (GPOF) to extract the poles and residues from transient thermal response. Combine (1) and (2), we can build the parameterized transient thermal behavioral models.

In the following section, we will first briefly review the improved GPOF method for transient thermal behavioral modeling before we present our new parameterized thermal behavior models.

6.2 GPOF and Improved GPOF for thermal modeling

6.2.1 Review of generalized pencil-of-function (GPOF) method

Generalized pencil-of-function (GPOF) method has been used to extract the poles and residues from the transient response of a real-time system and electromagnetism [24, 25, 64]. It works on the sum of exponential forms, which can be expressed in the partial fraction form in frequency domain like (6.4). GPOF method can be viewed as a special generalized eigenvalue computing method, in which we not only compute the eigenvalue (poles) of the given two matrices made by the sampled data, but also produce the residue for each pole in the partial fraction form [23]. As a result, it can be used to extract poles and residues from the thermal-related impulse responses for our problem. GPOF algorithm flow can be shown in Fig. 6.6, where N is the total number of sampled points, M is the order or the number of poles and L can be viewed as sampling window size.

For GPOF method, it allows $M \leq L \leq N - M$, which means that we can allow the different window size and pole numbers. Typically, choosing $L = N/2$ can yield better results.

ALGORITHM: GPOF
Input: sampling vectors $\mathbf{y}_i = [y_i, y_{i+1}, ..., y_{i+N-L-1}]^T$
Output: poles vector $\mathbf{p}$ and residues vector $\mathbf{r}$

1. Construct matrices Y_1 and Y_2.
   $Y_1 = [\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_{L-1}] \quad Y_2 = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_L]$
2. Singular value decomposition (SVD) of Y_1. $Y_1 = UDV^H$
3. Construct matrix Z. $Z = D^{-1}U^H Y_2 V$
4. Eigen-decomposition of Z. $Z_0 = eig(Z)$
   find poles vector: $p_i = \frac{log(z_i)}{\Delta t}$
5. Solve R_1 and R_2 from $Y_1 = Z_1 R Z_2$ and $Y_2 = Z_1 R Z_0 Z_2$.
   $$Z_1 = \begin{bmatrix} 1 & 1 & ... & 1 \\ z_1 & z_2 & ... & z_M \\ \vdots & \vdots & ... & \vdots \\ z_1^{N-L-1} & z_2^{N-L-1} & ... & z_M^{N-L-1} \end{bmatrix}$$
   $$Z_2 = \begin{bmatrix} 1 & z_1 & ... & z_1^{L-1} \\ & & ... & \\ \vdots & \vdots & ... & \vdots \\ 1 & z_M & ... & z_M^{L-1} \end{bmatrix}$$
   find residues vector: $\mathbf{r} = \frac{R_1 + R_2}{2}$

Figure 6.6: GPOF algorithm for poles and residues extraction

6.2.2 Improved GPOF method for thermal modeling

Directly applying the GPOF to the computed thermal impulse response may not lead to stable and accurate model. We improve the GPOF method by using logarithmic-scale and stabilization process mentioned below. The resulting method, called *ThermPOF*, was proposed recently by the authors [36, 34]. *ThermPOF* builds the linear transient thermal models for given power and temperature information and is briefly reviewed in the following.

Because the temperatures change very rapidly in a very short time and gradually reach

a steady state for a long time. This feature results in the modeling problem for GPOF method if linear sampling is used. A logarithmic-scale sampling technique is presented in *ThermPOF* to mitigate this problem. After obtaining the transfer function from GPOF, *ThermPOF* can get the response back in original time scale.

Also, GPOF method will not always generate stable poles for a given impulse response. The response from the model by GPOF can be unbounded outside the sampled interval while using positive poles, although GPOF model can give a very good matching for a given impulse response for the sampled interval. To mitigate this problem, *ThermPOF* artificially extends the time interval when the impulse response is zero. By sufficiently extending the time interval of zero-response in an impulse response, it can make all the poles stable.

Further more, the obtained impulse response may become zero numerically for a short period because temperature changes at the beginning is very slow. And long zero-response time at beginning may cause the significant discrepancies in the reduced models. To resolve this problem, *ThermPOF* truncates the beginning zero-response time such that response goes to non-zero numerically immediately. The second method to mitigate this problem is by increasing the value of L, which means more sampling points but more accuracy. The advantage of the second method is that it can use the same offset for all the transfer functions, which can reduce the complexity in the thermal simulation.

6.3 Parameterized thermal behavioral modeling method

In this section, we present our new parameterized thermal behavioral modeling approach. We first present the *ParThermPOF* algorithm overall flow and then present the important

steps in the algorithm.

6.3.1 The ParThermPOF algorithm flow

The proposed *ParThermPOF* consists of two major steps: first, building parameterized models by *response surface methods* on every time point; second, building subsystem response behavior models by *ThermPOF*. Let assume that we have k parameter variables in the parameter space $\Omega = \{\xi | \xi = (\xi_1, \xi_2, ..., \xi_k)\}$.

The proposed *ParThermPOF* is given in Fig. 6.7. Steps 1-2 build the response surface

ALGORITHM: PARTHERMPOF
Input: the sampled transient power and temperature information in parameter space Ω
Output: parametered thermal models $T(t, P, \Omega)$

1. Perform coding for the variable parameters where original parameter variables are converted to a new and expanded variable spaces.
2. Obtain the coefficients of the fitting polynomials of response surface models (RSM) in a minimum square way.

   1.
3. Perform the numerical differentiation on each coefficient of the polynomials in resulting RSM and optimize the resulting waveforms for stability and accuracy.
4. Compute the transient thermal model for each coefficient of the polynomials in resulting RSM.

Figure 6.7: The proposed *ParThermPOF* flow.

models in the parameter space Ω at each time step t. Steps 3-4 build the transient thermal models on top of the RSM models.

As an illustration, Fig. 6.8 shows the response surfaces generated by *ParThermPOF* over 3 selected time points when only core0 is active. In the following, we discuss the

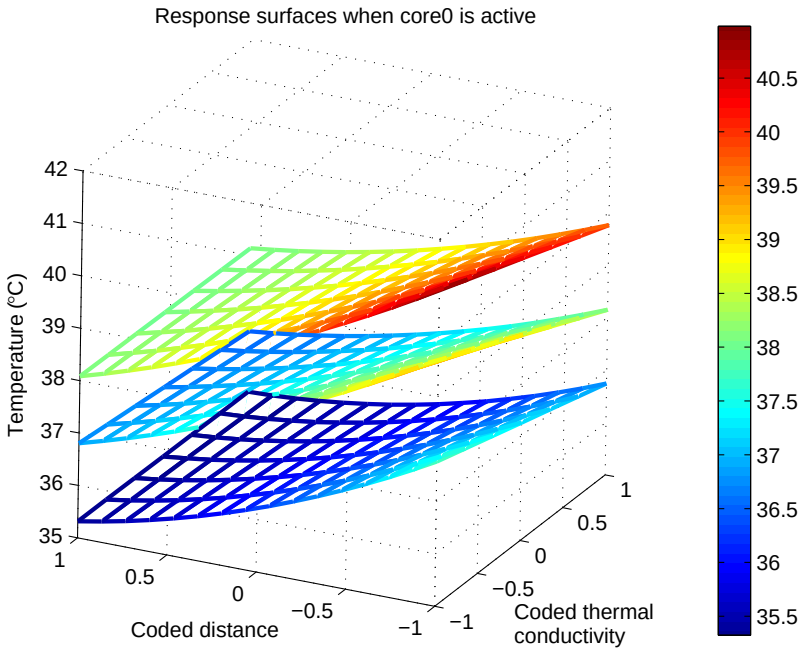


Figure 6.8: Response surfaces at 3 time points when only core0 is active

important steps in the proposed method.

6.3.2 Response surface method

Response surface methodology (RSM) explores the relationships between several input variables and one or more responses. The main principle of RSM is to use a set of designed experiments to obtain an optimal response. There are many applications of RSM in real industry, particularly in situations where several input variables potentially influence some performance measure or quality characteristic of the product or process [44]. This performance measure or quality characteristic is called response. And the input variables are sometimes called independent variables.

Specifically, suppose that a response y depends on serval controllable input variables

$(\xi_1, \xi_2, ..., \xi_k)$

$$y = f(\xi_1, \xi_2, ..., \xi_k) + \varepsilon \tag{6.5}$$

where the form of the true response function f is unknown and perhaps very complicated. We need to minimize the error ε when building response surface models.

Usually, a low-order polynomial in some relatively small region of the independent variable space is appropriate. In many cases, either a first-order or a second-order model is used. First-order model is easy to estimate and apply, but it can only accurately approximate the true response surface over a relatively small region of the variable space where there is little curvature in f. But if the curvature is strong enough that the first-order model is inadequate to fit the true response surface, a second-order model will be required.

6.3.3 Building parameterized thermal behavioral models

Coding for the variable parameters

The first thing we do is to transform the natural variables ξ in a range $[a, b]$ into coded variables x in a range $[-1, 1]$. After coding, the variable matrix X will have all orthogonal columns. It may reduce the numerical errors and increase the numerical stability.

A simple linear transformation can be used on the original measure scale so that the highest value becomes "1" and the lowest value becomes "-1". Assume that a variable ξ_i is in a range $[a, b]$, using the linear transformation in (6.6), we can convert the coded variable x_i into a range $[-1, 1]$. Now we can use the coded variables $x_1, ...x_8$ instead of the natural ones $\xi_1, ...\xi_8$.

$$x_i = \frac{\xi_i - (b + a)/2}{(b - a)/2} \tag{6.6}$$

Build the response surface models

In this chapter, we use second-order response surface model. A second-order response y depending on variables $(x_1, x_2, ..., x_k)$ can be written as

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum_{j=2}^{k} \sum_{i<j} \beta_{ij} x_i x_j + \sum_{j=1}^{k} \beta_j x_j^2 + \varepsilon \tag{6.7}$$

If we let $x_{k+1} = x_1 x_2$, $x_{k+2} = x_2 x_3$, ... ,$x_{k(k+1)/2+1} = x_1^2$, $x_{k(k+1)/2+2} = x_2^2$, ... , $\beta_{k+1} = \beta_{12}$, $\beta_{k+2} = \beta_{23}$, ... ,$\beta_{k(k+1)/2+1} = \beta_{11}$, $\beta_{k(k+1)/2+2} = \beta_{22}$, ..., then (6.7) becomes

$$y = \beta_0 + \sum_{j=1}^{q} \beta_j x_j + \varepsilon \tag{6.8}$$

which is a linear regression model for coefficients $(\beta_0, \beta_1, ..., \beta_q)$, where $q = k(k+3)/2$. We can use least squares method to estimate the regression coefficients in the multiple linear regression model in (6.8).

Suppose that we have n observed responses $\mathbf{y} = (y_1, y_2, ..., y_n)$ and for each y_i we have one set of parameter values $\mathbf{x_i} = (x_{i1}, x_{i2}, ..., x_{iq})$. So (6.8) can be written in matrix notation as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \varepsilon \tag{6.9}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & ... & x_{1q} \\ 1 & x_{21} & x_{22} & ... & x_{2q} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & ... & x_{nq} \end{bmatrix}, \tag{6.10}$$
$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_q \end{bmatrix}, \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

We would like to find the coefficient solution vector $\hat{\boldsymbol{\beta}}$ that minimizes the squares of errors E, where

$$E = \sum_{i=1}^{n} \varepsilon_i^2 = \boldsymbol{\varepsilon}'\boldsymbol{\varepsilon} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{6.11}$$

And E can be expressed as

$$\begin{aligned} E &= \mathbf{y}'\mathbf{y} - \boldsymbol{\beta}'\mathbf{X}'\mathbf{y} - \mathbf{y}'\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \\ &= \mathbf{y}'\mathbf{y} - 2\boldsymbol{\beta}'\mathbf{X}'\mathbf{y} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \end{aligned} \tag{6.12}$$

To minimize E, we have

$$\left.\frac{\partial E}{\partial \boldsymbol{\beta}}\right|_{\hat{\boldsymbol{\beta}}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{0} \tag{6.13}$$

So the least squares estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{6.14}$$

In practice, we do QR decomposition on X to make the computation numerically more stable. So we obtain $\mathbf{R}\boldsymbol{\beta} = \mathbf{Q}^T\mathbf{y}$. After solving the linear equations, we get the estimated coefficient vector $\hat{\boldsymbol{\beta}}$.

Building generalized linear thermal models for coefficients

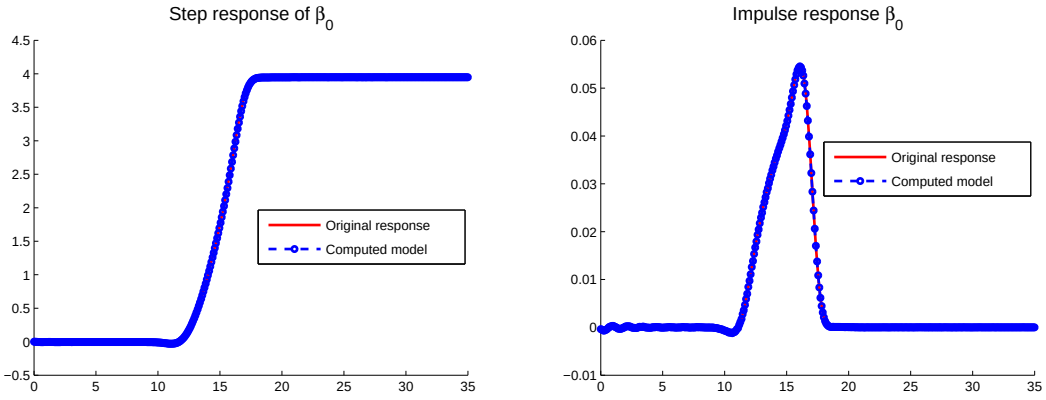
After we obtain coded variable matrix X, the coefficients of our model can be computed using (6.14). At this point, we obtain the parameterized thermal model only on a single time point. We then compute the response surface models on all the time points, which could generate a set of response surface models, or more precisely, a set of coefficients, which are functions of time now. Since we can consider the temperature response as a linear combination of such coefficients, the original thermal system is decomposed into a number of linear dynamic subsystems. Each coefficient is considered as temperature output of each subsystem and these subsystems share the same power inputs.

To build transient models, we need to incorporate the time into our model. Now we apply the *ThermPOF*[36] to each coefficient, which is a function of time only and is computed from the previous RSM step. The coefficient function, which can be viewed as a special transient temperature function along with the input powers (the real temperature function is the combination of those coefficients), will become a multiple-input and single-output (MISO) linear dynamic system. In our specific thermal problem as shown in Fig. 6.5, each coefficient function consists of 5 power inputs and 1 temperature output. Once we have the coefficient models, we can compute the total temperature response of the whole system, which is just the sum of all the responses from all the subsystems together. Note that the modeling process above is only for one power output. We need to repeat it 5 times in order to obtain the models of thermal system with 5 power outputs.

6.3.4 The thermal-coefficient step and impulse response

In *ParThermPOF*, instead of having the thermal power step responses, we obtain the thermal-coefficient power step responses. Although such responses do not have direct physical meaning, but the resulting step and impulse responses still resemble the thermal-power step and impulse response.

Most important is that GPOF can be still applied to obtain the transient thermal models for the coefficients, which show the flexibility of the new approach. Fig. 6.9 shows the step and impulse responses of one of the coefficients in the resulting thermal models (versus the original ones), which are similar to the actual thermal step and impulse responses.



(a) Step response of one coefficient.

(b) Impulse response of coefficient.

Figure 6.9: Step and impulse responses of one of the coefficients. The x axis is the time in logarithmic scale and y axis is the relative temperature to the ambient temperature.

6.3.5 A walkthrough example

We illustrate the new method by using a real example. Specifically, the temperature response y is a function of the following parameters: two variables (ξ_1, ξ_2) are distance

away from the center and thermal conductivity of the heat sink materials; six variables $(\xi_3, \xi_4, ..., \xi_8)$ are used to indicate observation components, such as core0-core3, cache, heat spreader and heat sink. Such variables are called indicator variables, because values in them are binary (0 or 1), while values in (ξ_1, ξ_2) are continuous.

We obtained the data from Intel and the data was computed from the commercial thermal analysis tool based on a real quad-cord microprocessor. The observed temperature responses are on ξ_1 = $0mm$, $5mm$, $15mm$ and $\xi_2 = 240W/(m \cdot K)$(Al), $390W/(m \cdot K)$(Cu). $\xi_3 = 1$ if we observe the temperature on core0, $\xi_4 = 1$ if we observe the temperature are on core1. The setting for $\xi_5, ...\xi_8$ are the same. They represent core2, core3, cache and heat spreader when they are set to 1, respectively. At any time, there is at most one variable which is set to 1 in $\xi_3, ...\xi_8$. When $\xi_3, ...\xi_8$ are all zeros, it means that we observe on heat sink.

In our setting, we set x_1 as a full second-order form, which consists of the linear terms, the crossing terms amid different variables and squared terms. For x_2, we first consider the temperatures on two thermal conductivity points ($240W/(m \cdot K)$ and $390W/(m \cdot K)$ (we consider one more material in the experimental section). So in our models we consider x_2 as a second-order form including only linear and crossing terms. We may extend x_2 to a full second-order or even high-order form for more training data.

For $x_3, ...x_8$, because they are indicator variables only binary value (0 or 1), we also consider them as a second-order form including only linear and crossing terms. Also, based on our current given data, $x_3, ...x_8$ only have the crossing terms with x_2, because the temperatures we obtained on $0mm$, $5mm$ and $15mm$ away from center is only for heat sink. For other components, such as core0-core3, cache and heat spreader, we only know the temperature on their centers. So currently indicator variables are independent

of distance x_1. Note that we indicate the heat sink by setting $\xi_3, ...\xi_8$ to all zeros. So our models can still work well to capture the temperature responses on heat sink for different values of distance variable x_1.

Now we can begin to set up variable matrix $\mathbf{X}$ based on given temperature data like the form in (6.10). There are 17 terms in total, including constant, linear, crossing and squared terms. For each time point, we have 18 given temperature samples for different distances, different thermal conductivities of heat sink materials and different observation components. So we obtain the coded variable matrix $\mathbf{X}$ as shown in Fig. 6.10.

$$
\left[\begin{array}{cccccccccccccccc}
1 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_1x_2 & x_2x_3 & x_2x_4 & x_2x_5 & x_2x_6 & x_2x_7 & x_2x_8 & x_1^2 \\
\hline
1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\
1 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\
1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\
1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1/3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 1/9 \\
1 & -1/3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 1/9 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right] \tag{6.15}
$$

Figure 6.10: Coded variable matrix $\mathbf{X}$

6.3.6 More remarks for the proposed method

We remarks that response surface model works fine when the response can be approximated by low-order models. Our Numerical examples show that the given parameters like

locations of thermal sensors in a heat sink, thermal conductivity of heat sink materials etc, second order approximation can give quite good approximation. For strong nonlinear parameters, new modeling techniques will be explored such as using orthogonal polynomials in RSM or piecewise linear modeling methods.

We also remark that currently the number of samples will depend exponentially on the number of variables for sufficient accuracy. We used a simple sampling method as sampling is not the main focus of this chapter. More efficient sampling methods will be investigated in the future to accommodate more parameters.

Also for fine granularity modeling where a large number of input power sources exist, the proposed method can still work. More power inputs will leads to more transfer functions and more matrix pencil operations. But the number of power inputs does not add the sampling dimensionality as all the inputs share the same time steps from one detailed simulation of a particular setting. The only difference is that each transfer function (or the coefficient function) will have more inputs.

To consider the dependency of leakage powers and thermal conductivity on temperature, the simple way is to build the thermal models on the actual measured data (we are working on that with Intel). Another way is to build the models on the detailed simulation in which such dependencies are considered. Although such a model is still linear, but we at least have first-order approximation to the nonlinear effects.

In addition to the parameter variables, there are many other package variables which will affect the thermal characterization of the whole packages such as the thermal conductivities of the materials used TIM1 and TIM2. In this chapter, we just demonstrate that the proposed method can accommodate different parameters. Our next step is to make it more practical for use in industry setting.

6.4 Numerical examples

The proposed *ParThermPOF* algorithm has been implemented in MATLAB 7.0 and the Numerical examples are obtained on a Dell PowerEdge 1900 workstation (using a Linux system) with Intel Quadcore Xeon CPUs with 2.99Ghz and 16GB memory.

The example we use is the quad-core microprocessor as shown in Fig. 6.1, from Intel. We first build parameterized thermal behavior models from a training data set, using commercial thermal analysis tool *FloTHERM* [18], which is a 3D computational fluid dynamics (CFD) commercial software, where we collect the computed step temperature response when only one single step power source is applied at one time. After parameterized thermal models are built, we could apply them to generate the temperature responses for any type of time-varying input power sources and different parameter settings.

In our experiments, the training data used first to build the models have different time scales from the benchmark data used later to verify the models. Both the training data and benchmark data from Intel are the powers and computed temperatures (using *FloTHERM*) on a realistic quad-core microprocessor under some operating conditions. The given temperature distribution when using a Copper heat sink at $t = 1s$ is shown in Fig. 6.11. The power input traces in the benchmark are shown in Fig. 6.12(a), where the step power is $20W$ for all the cores.

In practice, temperature response can be computed very fast by our models during any time interval, as the computation complexity in our model is only $O(n)$ by using the recursive convolution on the pole-residue expression, where n is the number of time steps. Note that the simulation results at one time point are obtained for all the parameter space. In other words, when we change the values of parameters at one time point, the results can be computed directly without doing transient simulation again.

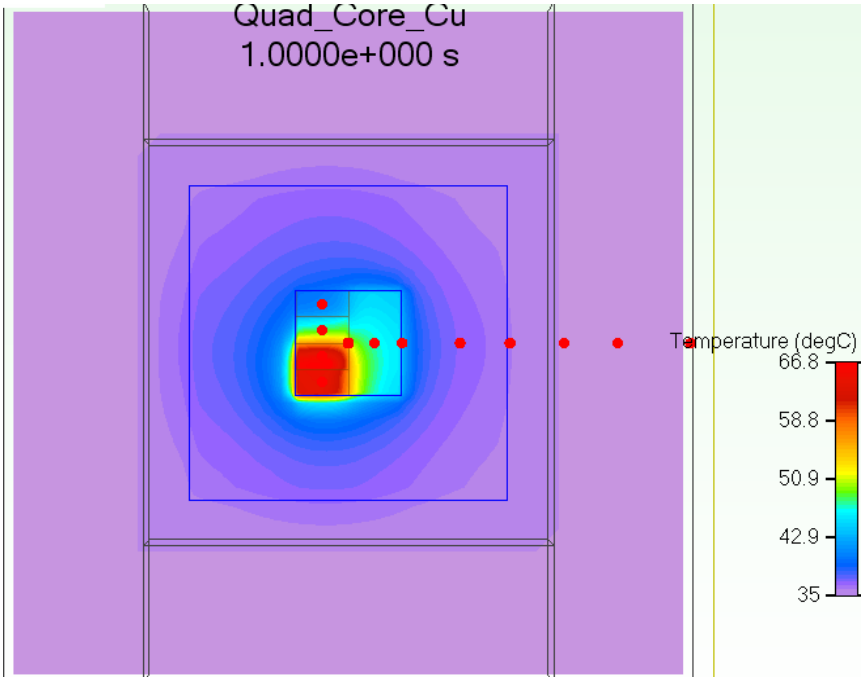
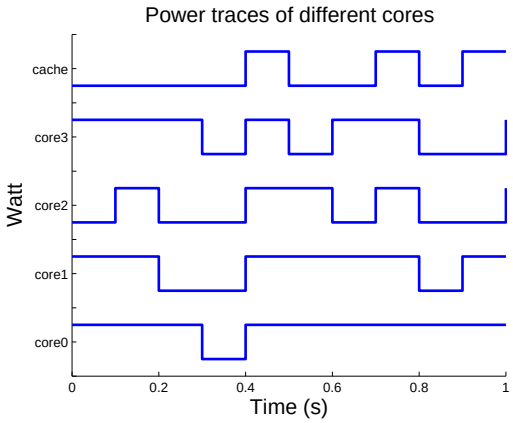
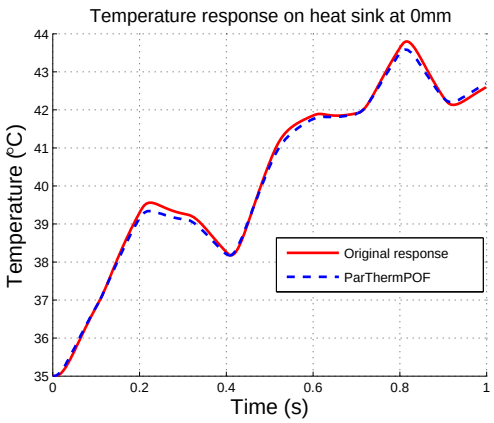


Figure 6.11: Given temperature distribution on the whole chip package when using a Copper heat sink at $t = 1s$.



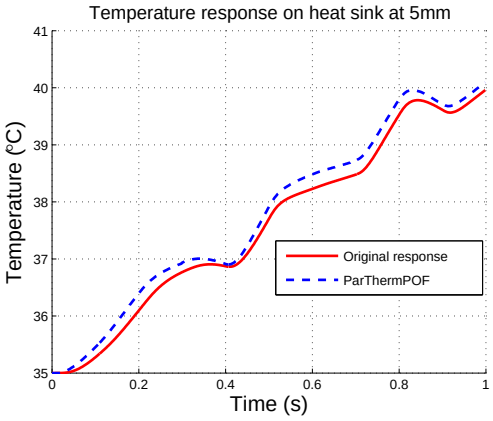
(a) Power input traces in the benchmark.



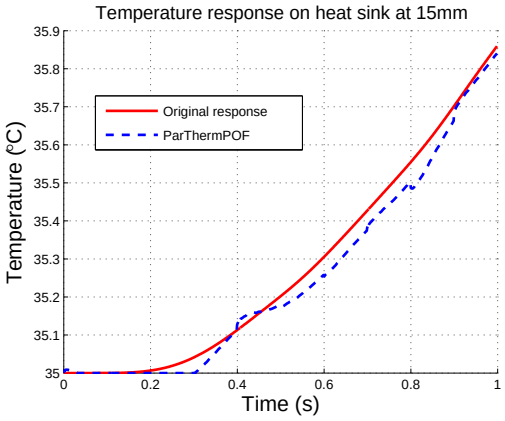
(b) Temperatures on a heat sink made of Aluminum at 0mm.

Figure 6.12: Thermal simulation results on specific values of parameters

Now we will show the accuracy of *ParThermPOF*. The calculation of temperature responses at each coefficient is only done once. Then we can obtain thermal response for any

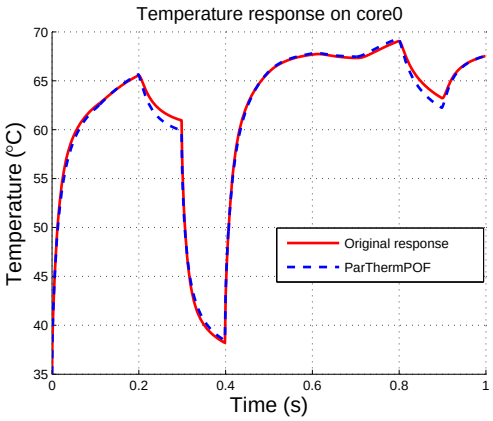


(a) Temperatures on heat sink made of Aluminum at 5mm.

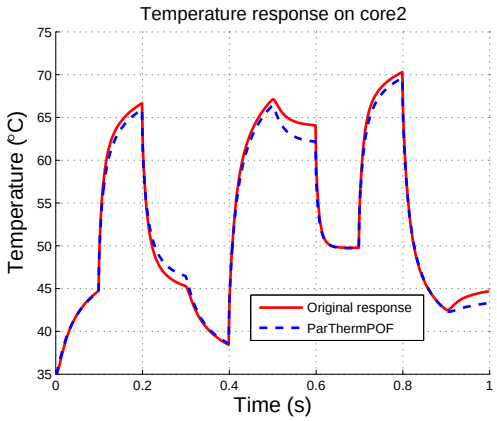


(b) Temperatures on a heat sink made of Aluminum at 15mm.

Figure 6.13: Thermal simulation results on specific values of parameters



(a) Temperatures on core0 when using a Copper heat sink.



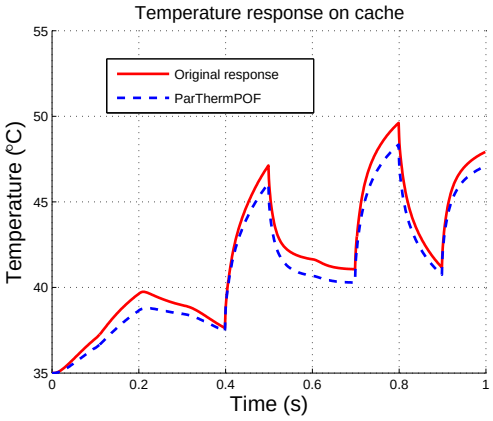
(b) Temperatures on core2 when using a Copper heat sink.

Figure 6.14: Thermal simulation results on specific values of parameters

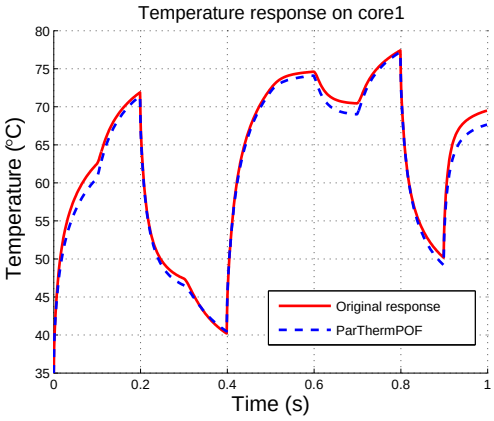
specific values for parameters $(\xi_1, \xi_2, ..., \xi_8)$ by setting them directly in the models.

Fig. 6.12(b) and Fig. 6.13 show the computed temperature results at the points $0mm$, $5mm$ and $15mm$ away from the center when using an Aluminum heat sink. In another word, we set $\xi_1 = 0, 5, 15$, $\xi_2 = 240$ and others to zeros.

Fig. 6.14 and Fig. 6.15(a) show the temperatures on the center of core0, core2 and cache

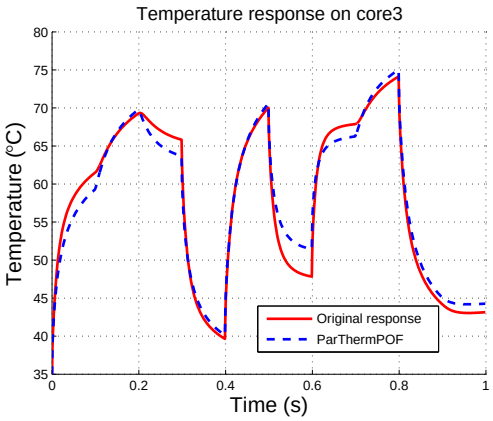


(a) Temperatures on cache when using a Copper heat sink.

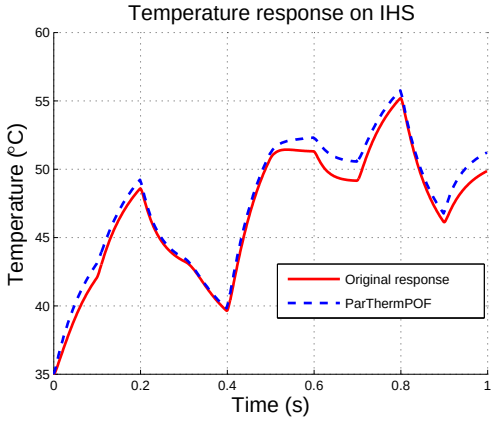


(b) Temperatures on core1 when using an Aluminium heat sink.

Figure 6.15: Thermal simulation results on specific values of parameters



(a) Temperatures on core3 when using an Aluminium heat sink.



(b) Temperatures on heat spreader when using an Aluminum heat sink.

Figure 6.16: Thermal simulation results on specific values of parameters

when using a Copper heat sink. In these cases we set $\xi_1 = 0$, $\xi_2 = 390$, $\xi_3 = 1$ or $\xi_5 = 1$ or $\xi_7 = 1$ and others to zeros.

Fig. 6.15(b) and Fig. 6.16 show the temperatures on the center of core1, core3 and the heat spreader when using an Aluminum heat sink. In these cases we set $\xi_1 = 0$, $\xi_2 = 240$, $\xi_4 = 1$ or $\xi_6 = 1$ or $\xi_8 = 1$ and others to zeros.

From the figures, we can see that all the peak temperatures for each set of parameters during the whole time interval match well between computed data and given data. The models work well for the nine sets of specific parameters as we just showed sequentially. The errors and percentages are shown in Table 6.1. All the temperature errors except for set6 (cache with a Copper heat sink) is less than $1°C$.

The average errors and relative errors (computed temperature over given temperature on each time point) between computed data and given data are shown in Table 6.2. From Table 6.1 and Table 6.2, we can see that *ParThermPOF* is very accurate.

Table 6.1: Errors of the peaks

Parameter settings	Maximal peak		
	Given ($°C$)	Error ($°C$)	Percentage
set1	43.8	0.21	0.48%
set2	40.0	0.14	0.35%
set3	35.9	0.02	0.06%
set4	69.1	0.29	0.42%
set5	70.3	0.69	0.98%
set6	49.6	1.27	2.56%
set7	77.4	0.22	0.28%
set8	74.2	0.87	1.17%
set9	55.2	0.57	1.03%

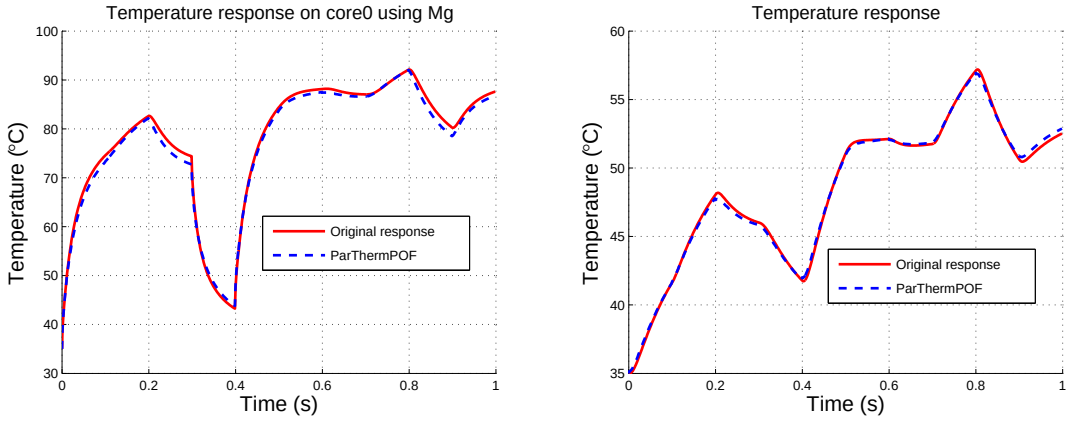
Finally, we add one sampling point for the thermal conductivity parameter of sink materials after we did for Al and Cu in our model and to see how the model works. Specifically, we add the thermal-power data for Magnesium (Mg), whose thermal conductivity is $160W/(m \cdot K)$ as a heat sink material (Cu has $390W/(m \cdot K)$ and Al has $240W/(m \cdot K)$.). In this case, x_2 in Section 6.3 are not only including linear and crossing terms, but also including squared terms. We also need to add one column x_2^2 to coded variable matrix $\mathbf{X}$ and update the corresponding item values in $\mathbf{X}$.

Fig. 6.17(a) and Fig. 6.17(b) show the temperatures on the center of core0 and heat

Table 6.2: Average errors and relative errors between the computed and given temperatures

Parameter settings	Average Error ($^\circ C$)	Average Relative Error ($^\circ C$)
set1	0.06	0.16%
set2	0.18	0.49%
set3	0.02	0.07%
set4	0.15	0.23%
set5	0.39	0.59%
set6	0.72	1.69%
set7	0.81	1.28%
set8	0.17	0.55%
set9	0.71	1.52%

sink when using an Magnesium heat sink when pulse-like power inputs are excited for the generated models using the new training data. In these cases we set $\xi_1 = 0$, $\xi_2 = 160$, $\xi_3 = 1$ or $\xi_3 = 0$, and others to zeros. ParThermPOF can still obtain enough accurate results.



(a) Temperatures on core0 when using a Magnesium (Mg) heat sink.

(b) Temperatures on the center of heat sink when using a Magnesium (Mg) heat sink.

Figure 6.17: Thermal simulation results on specific values of parameters

Now we report some CPU times for the proposed method and compare them with *FloTHERM* [18], which uses advanced numerical techniques to compute the thermal re-

sponses. In the FloTHERM, each run for one setting (with fixed thermal materials for heat sink, heat spreader, ambient temperature and thermal conditions) takes about 25 minutes for 1000 transient steps.

While in ParThermPOF, the training process takes 40 seconds for 5 inputs and 19 coefficients, which means performs 5×19 matrix pencil operations [36]. After we obtain the models from training part, the simulation time is much less. For the problem we have, it takes 2.81 seconds to compute the 19 coefficients for the whole simulation period (1000 steps). For one particular response at one time step, it only costs 0.002s. The reduced model has 535X speedup over FloTHERM if we only consider the transient simulation time, and 35X speedup over FloTHERM if we consider both training and transient simulation time. As a result, the proposed compact modeling is much fast than the full-blown numerical simulation.

6.5 Summary

In this chapter, we have proposed a new architecture-level parameterized dynamic thermal behavioral modeling algorithm for emerging thermal-related analysis and optimization problems for high-performance chip-multiprocessor design. The proposed compact thermal model will be used to predict the thermal response of new package designs once its accuracy has been calibrated and validated with the detailed models. This is the design methodology to be used by our industry partner. We propose a new approach, called ParThermPOF, to build the parameterized dynamic thermal behavioral models from accurately computed thermal and power information using the sophisticated FEA (Finite Element Analysis) or CFD (Computational Fluid Dynamics) tools at architecture level. The

new method is a top-down, black-box approach, meaning it does not require any internal structure of the systems and it is very general and flexible. ParThermPOF is able to include a number of parameters such as location of thermal sensors in a heat sink, different components (heat sink, heat spreader, core, cache, etc.), thermal conductivity of heat sink materials, etc. The new method consists of two steps: first, a Response Surface Method (RSM) based on low-order polynomials is applied to build the parameterized models at each time point for all the given sampling nodes in the parameter space (except for time). Second, an improved Generalized Pencil-Of-Function (GPOF) method, called ThermPOF, is employed to build the transfer-function-based models for each time-varying coefficient of the polynomials generated in the first step. Simulation results on a practical quad-core microprocessor show that the generated parameterized thermal behavioral models can be built very efficiently and the temperatures computed from resulting models match the given temperatures well for given parameter space in the time domain. The compact models by ParThermPOF offer two order of magnitudes speedup over the commercial thermal analysis tool FloTHERM [18] on the given examples from our industry partner.

Chapter 7

Conclusion

In this dissertation, we have finished deep studies on several modeling and simulation algorithms of on-chip power delivery networks and temperature profile on multi-core microprocessors.

7.1 Modeling and simulation of on-chip power delivery networks

Reliable on-chip power delivery is a challenging design task for sub-100nm and below VLSI technologies as voltage IR drops become more and more pronounced. This situation gets worse as technology continues to scale down. And efficient verification of power integrity becomes critical for design closure. In addition, the increasing process-induced variability makes it even worse for reliable power delivery networks. The process induced variations manifest themselves at different levels (wafer level, die-level and within a die) and they are caused by different sources (lithograph, materials, aging, etc.).

For power delivery networks without considering process variations, we have proposed a new fast simulation method ETBR for extended truncated balanced realization, which uses MOR (Model Order Reduction) to speedup the simulation. ETBR is based on a more accurate reduction framework: truncated balanced realization, which was shown to be more accurate than Krylov subspace method used in EKS method. ETBR also avoids the explicit moment representation of the input signals, which have well-known numerical problems in the past. Instead, it uses spectrum representation of input signals by fast Fourier transformation. As a result, ETBR is much more flexible for different types of input sources and can better capture the high frequency contents than EKS and this leads to more accurate results for fast changing input signals. To make ETBR more accuracy, we further introduce an error control mechanism into it. The improved method is called ETBR_IR. The error control mechanism is based on the system residuals as well as the novel effective resistance concept to compute the errors in terms of more useful voltage drop values. The on-the-fly error reduction works well for compensating high frequency accuracy loss related to disruptive tap current waveforms in typical industry power delivery networks. ETBR_IR provides an efficient way to easily trade errors for speedup to suit different applications. Numerical results show ETBR_IR can significantly reduce the errors of the existing ETBR method at the similar computing cost, while it can have 10X and more speedup over the the commercial power grid simulator in UltraSim with about 1-2% errors on a number of real industry benchmark circuits.

For power delivery networks with considering process variations, we have proposed a novel scalable statistical simulation approach for large power grid network analysis considering process variations. The new algorithm is very scalable for large networks with a large number of random variables. The new method, called varETBR, is based on the previously

proposed extended truncated balanced realization (ETBR) method. To consider the variational parameters, we extend the concept of response Grammian, which was used in ETBR to compute the reduction projection subspace, to the variational response Grammian. Then Monte Carlo based numerical integration is employed to multiple-dimensional integrals. Numerical examples, on a number of the IBM benchmark circuits [47] up to 1.6 million nodes, show that the varETBR can be up to $1900X$ faster than the Monte Carlo method, and is much more scalable than the StoEKS method [42, 41]. varETBR can solve very large power grid networks with large numbers of random variables, large variation ranges and different variational distributions.

To further speedup the MOR process used in the fast simulation, a hierarchical Krylov subspace projection based MOR approach, hiePrimor, is proposed. Different from the traditional flat MOR, The new method combines the partitioning strategy and the Krylov subspace method to speed up the reduction process. hiePrimor is more suitable for reducing many large global interconnects like coupled bus, transmission lines and large clock nets where the number of ports are general not significant. The new method is a very general hierarchical model order reduction technique and it works for general parasitic interconnect circuits modeled as RLC circuits. Numerical results demonstrate that hiePrimor can be significantly faster (up to 5x) and more scalable than the flat projection methods like PRIMA and be order of magnitude faster than PRIMA with parallel computing without loss of accuracy. Interconnect circuits with up to 4 million nodes can be analyzed in a few minutes even in Matlab.

7.2 Modeling and simulation of temperature profile on multi-core microprocessors

Besides the on-chip power delivery, excessive on-chip temperature has also become a first-tier design constraint as CMOS technology scales into the nanometer region. The exponential increase of power density of the high-performance microprocessors leads to the rapid rising of the average chip temperature. Higher temperature has significant adverse impacts on chip package cost, performance, and reliability. Multi-core techniques provide a viable solution to temperature/power problems. However, designing thermal efficient multi-core microprocessors remains a challenging problem as the temperature in each core can be dramatically different and the resulting large temperature gradients can produce mechanical stress and degrade the chip reliability.

In this dissertation, we have investigated a new architecture-level dynamic thermal characterization problem from a behavioral modeling perspective to address the emerging thermal related analysis and optimization problems for high-performance multi-core microprocessor design. We have proposed a new thermal behavioral modeling approach for fast temperature estimation at the architecture level for multi-core microprocessors. The new approach, called ThermPOF, , builds the transfer function matrix from the measured or simulated thermal and power information. It first builds behavioral thermal models using the generalized pencil-of-function (GPOF) method. However, the direct use of GPOF does not work for thermal systems. Based on the characteristics of transient chip-level temperature behaviors, we make two new improvements over the traditional GPOF: First we apply a logarithmic-scale sampling scheme instead of the traditional linear sampling to better capture the rapid temperatures change over the long period. Second, we modify the extracted

thermal impulse response such that the extracted poles from GPOF are guaranteed to be stable without accuracy loss. Finally we further reduce the size of thermal models by a Krylov subspace reduction method to speedup the simulation process. Numerical results on a real quad-core microprocessor show that generated thermal behavioral models match the measured temperature very well.

Further, we have extended ThermPOF into ParThermPOF, a parameterized dynamic thermal behavioral modeling algorithm for emerging thermal-related analysis and optimization problems for high-performance chip-multiprocessor design. The proposed compact thermal model will be used to predict the thermal response of new package designs once its accuracy has been calibrated and validated with the detailed models. ParThermPOF builds the parameterized dynamic thermal behavioral models from accurately computed thermal and power information using the sophisticated FEA (Finite Element Analysis) or CFD (Computational Fluid Dynamics) tools at architecture level. ParThermPOF consists of two steps: first, a Response Surface Method (RSM) based on low-order polynomials is applied to build the parameterized models at each time point for all the given sampling nodes in the parameter space (except for time). Second, ThermPOF is employed to build the transfer-function-based models for each time-varying coefficient of the polynomials generated in the first step. Numerical results on a practical quad-core microprocessor show that the generated parameterized thermal model matches the given data very well. The compact models by ParThermPOF offer two order of magnitudes speedup over the commercial thermal analysis tool FloTHERM [18] on the given examples.

Bibliography

[1] http://www.cs.umn.edu/ metis.

[2] Umfpack. http://www.cise.ufl.edu/research/sparse/umfpack/.

[3] Intel multi-core processors, making the move to quad-core and beyond (White Paper), 2006. `http://www.intel.com/multi-core`.

[4] Multi-core processors - the next evolution in computing (White Paper), 2006. `http://multicore.amd.com`.

[5] International technology roadmap for semiconductors(itrs) 2007 edition, 2007. http://public.itrs.net.

[6] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. of Intl. Symp. on High-Performance Comp. Architecture*, pages 171–182, 2001.

[7] M. Celik, L. Pileggi, and A. Odabasioglu. *IC interconnect analysis*. Kluwer Academic Publishers, 2002.

[8] N. Chang, L. Barford, and B. Troyanovsky. Fast time domain simulation in spice with frequency domain data. *Electronic Components and Technology Conference*, pages 689–695, 1997.

[9] T. Chen and C. C. Chen. Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative method. In *Proc. Design Automation Conf. (DAC)*, pages 559–562, 2001.

[10] C. Chiang and J. Kawa. *Design for Manufacturability*. Springer, 2007.

[11] L. Daniel, O. C. Siong, L. S. Chay, K. H. Lee, and J. White. Multi-parameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 23(5):678–693, May 2004.

[12] T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2006.

[13] A. Devgan, H. Ji, and W. Dai. How to efficiently capture on-chip inductance effects: introducing a new circuit element K. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 150–155, 2000.

[14] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Pade approximation via the Lanczos process. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(5):639–649, May 1995.

[15] P. Feldmann and F. Liu. Sparse and efficient reduced order modeling of linear subcircuits with large number of terminals. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 88–92, 2004.

[16] I. A. Ferzli and F. N. Najm. Statistical estimation of leakage-induced power grid voltage drop considering within-die process variations. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 865–859, 2003.

[17] I. A. Ferzli and F. N. Najm. Statistical verification of power grids considering process-induced leakage current variations. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 770–777, 2003.

[18] Flotherm. http://www.flomerics.com/products/flotherm/.

[19] P. Ghanta, S. Vrudhula, and S. Bhardwaj. Stochasic variational analysis of large power grids considering intra-die correlations. In *Proc. Design Automation Conf. (DAC)*, pages 211–216, July 2006.

[20] P. Ghanta, S. Vrudhula, R. Panda, and J. Wang. Stochastic power grid analysis considering process variations. In *Proc. Design, Automation and Test In Europe. (DATE)*, volume 2, pages 964–969, 2005.

[21] E. J. Grimme. *Krylov projection methods for model reduction (Ph.D. Thesis)*. University of Illinois at Urbana-Champaign, 1997.

[22] S.H. Gunther, F. Binns, D.M. Carmean, and J.C. Hall. Managing the impact of increasing microprocessor power consumption. In *Intel Technology Journal*, First Quarter 2001.

[23] Y. Hua, A. B. Gershman, and Q. Cheng, editors. *High-Resolution and Robust Signal Processing*, volume 19 of *Signal Processing and Communications*. CRC Press, 2004.

[24] Y. Hua and T.K. Sarkar. Generalized pencil of function method for extracting poles of an em system from its transient responses. *IEEE Trans. on Antennas and Propagation*, 37(2):229–234, Feb. 1989.

[25] Y. Hua and T.K. Sarkar. On SVD for estimating generalized eigenvalues of singular matrix pencils in noise. *IEEE Trans. on Signal Processing*, 39(4):892–900, Apr. 1991.

[26] W. Huang, M. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy. Compact thermal modeling for temperature-aware design. In *Proc. Design Automation Conf. (DAC)*, pages 878–883, 2004.

[27] Wei Huang, Eric Humenay, Kevin Skadron, and Mircea R. Stan. The need for a full-chip and package thermal model for thermally optimized ic designs. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 245–250, New York, NY, USA, 2005. ACM.

[28] Andrew B. Kahng. DFM tools and methodologies for 65nm and below. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, 2006. Tutorial.

[29] K. J. Kerns and A. T. Yang. Stable and efficient reduction of large, multiport rc network by pole analysis via congruence transformations. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 16(7):734–744, July 1998.

[30] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: A 32-Way multithreaded sparc processor. *IEEE Micro*, 25(2):21–29, 2005.

[31] B. Krauter and L. Pileggi. Generating sparse partial inductance matrices with guaranteed stability. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 45–52, 1995.

[32] Y. Lee, Y. Cao, T. Chen, J. Wang, and C. Chen. HiPRIME: Hierarchical and passivity preserved interconnect macromodeling engine for RLKC power delivery. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):797–806, 2005.

[33] D. Li, S. X.-D. Tan, and B. McGaughy. ETBR: Extended truncated balanced realization method for on-chip power grid network analysis. In *Proc. Design, Automation and Test In Europe. (DATE)*, pages 432–437, 2008.

[34] D. Li, S. X.-D. Tan, E. H. Pacheco, and M. Tirumala. Architecture-level thermal characterization for multi-core microprocessors. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 17(10):1495–1507, 2009.

[35] D. Li, S. X.-D. Tan, E. H. Pacheco, and M. Tirumala. Fast analysis of on-chip power grid circuits by extended truncated balanced realization method. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Science(IEICE)*, E92-A(12):3061–3069, 2009.

[36] D. Li, S. X-.D. Tan, and M. Tirumala. Architecture-level thermal behavioral characterization for multi-core microprocessors. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 456–461, Jan. 2008.

[37] H. Li, Z. Qi, S. X.-D. Tan, L. Wu, Y. Cai, and X. Hong. Partitioning-based approach to fast on-chip decap budgeting and minimization. In *Proc. Design Automation Conf. (DAC)*, pages 170–175, June 2005.

[38] Y. Li, B. C. Lee, D. Brooks, Z. Hu, and K. Skadron. CMP design space exploration subject to physical constraints. *Proc. IEEE Int. Symp. on High Performance Computer Architecture (HPCA)*, pages 15–26, 2006.

[39] P. Liu, H. Li, L. Jin, W. Wu, S. X.-D. Tan, and J. yang. Fast thermal simulation for runtime temperature tracking and management. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(12):2882–2893, Dec. 2006.

[40] Ying Liu, Lawrence T. Pileggi, and Andrzej J. Strojwas. Model order-reduction of rc(l) interconnect including variational analysis. In *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 201–206, 1999.

[41] N. Mi, S. X.-D. Tan, Y. Cai, and X. Hong. Fast variational analysis of on-chip power grids by stochastic extended krylov subspace method. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27(11):1996–2006, 2008.

[42] N. Mi, S. X.-D. Tan, P. Liu, J. Cui, Y. Cai, and X. Hong. Stochastic extended Krylov subspace method for variational analysis of on-chip power grid networks. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 48–53, 2007.

[43] B. Moore. Principal component analysis in linear systems: Controllability, and observability, and model reduction. *IEEE Trans. Automat. Contr.*, 26(1):17–32, 1981.

[44] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley-Interscience, 2002.

[45] S. Nassif. Delay variability: sources, impact and trends. In *Proc. IEEE Int. Solid-State Circuits Conf.*, pages 368–369, San Francisco, CA, Feb 2000.

[46] S. Nassif. Model to hardware correlation for nm-scale technologies. In *Proc. IEEE International Workshop on Behavioral Modeling and Simulation (BMAS)*, Sept 2007. Keynote speech.

[47] S. R. Nassif. Power grid analysis benchmarks. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 376–381, 2008.

[48] S. R. Nassif and J. N. Kozhaya. Fast power grid simulation. In *Proc. Design Automation Conf. (DAC)*, pages 156–161, 2000.

[49] A. Odabasioglu, M. Celik, and L.T. Pileggi. PRIMA: Passive reduced-order interconnect macromodeling algorithm. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 645–654, 1998.

[50] S. Pant, D. Blaauw, V. Zolotov, S. Sundareswaran, and R. Panda. A stochastic approach to power grid analysis. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 171–176, 2004.

[51] Sanjay Pant and D. Blaauw. Static timing analysis considering power supply variations. *Computer-Aided Design, International Conference on*, 0:365–371, 2005.

[52] M. Pedram and S. Nazarian. Thermal modeling, analysis and management in VLSI circuits: principles and methods. *Proc. of IEEE, Special Issue on Thermal Analysis of ULSI*, 94(8):1487–1501, 2006.

[53] J. Phillips. Variational interconnect analysis via PMTBR. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 872–879, Nov. 2004.

[54] J. R. Phillips, L. Daniel, and L. M. Silveira. Guaranteed passive balancing transformation for model order reduction. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(8):1027–1041, 2003.

[55] J. R. Phillips and L. M. Silveira. Poor man's TBR: a simple model reduction scheme. In *Proc. Design, Automation and Test In Europe. (DATE)*, pages 938–943, 2004.

[56] J. R. Phillips and L. M. Silveira. Poor man's TBR: a simple model reduction scheme. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(1):43– 55, 2005.

[57] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 352–366, April 1990.

[58] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic Circuit and System Simulation Methods*. McGraw-Hill, New York, 1994.

[59] H. F. Qian, S. R. Nassif, and S. S. Sapatnekar. Random walks in a supply network. In *Proc. Design Automation Conf. (DAC)*, pages 93–98, 2003.

[60] Zhanhai Qin and C.K. Cheng. Realizable parasitic reduction using generalized Y-Δ transformation. In *Proc. Design Automation Conf. (DAC)*, pages 220–225, 2003.

[61] A. E. Ruehli. Equivalent circuits models for three dimensional multiconductor systems. *IEEE Trans. on Microwave Theory and Techniques*, pages 216–220, 1974.

[62] R. Rutenbar. Next-generation design and EDA challenges. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, January 2007. Keynote speech.

[63] Yousef Saad. *Iterative methods for linear systems*. PWS publishing, 2000.

[64] T.K. Sarkar, F. Hu, Y. Hua, and M. Wick. A real-time signal processing technique for approximating a function by a sum of complex exponentials utilizing the matrix pencil approach. *Digital Signal Processing - A Review Journal*, 4(2):127–140, Apr. 1994.

[65] B. N. Sheehan. TICER: Realizable reduction of extracted RC circuits. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 200–203, 1999.

[66] B. N. Sheehan. Branch merge reduction of RLCM networks. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 658–664, 2003.

[67] J. Shi, Y. Cai, J. Fan, S. X.-D. Tan, and X. Hong. Pattern-based iterative method for extreme large power/ground analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 26(4):680–692, Apr. 2007.

[68] R. W. Shonkwiler and L. Lefton. *An introduction to parallel and vector scientific computing*. Cambridge University Press, 2006.

[69] M. Silveira, M. Kamon, I. Elfadel, and J. White. A coordinate-transformed Arnoldi algorithm for generating guaranteed stable reduced-order models of RLC circuits. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 288–294, 1996.

[70] B. Sinharoy, R.N. Kalla, J.M. Tendler, R.J. Eickemeyer, and J.B. Joyner. Power5 system microarchitecture. *IBM J. Research and Development*, 49(4):505–521, 2005.

[71] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature aware microarchitecture. In *Proc. IEEE International Symposium on Computer Architecture (ISCA)*, pages 2–13, 2003.

[72] A. Srivastava, R. Bai, D. Blaauw, and D. Sylvester. Modeling and analysis of leakage power considering within-die process variations. In *Proc. Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 64–67, Aug. 2002.

[73] H. Su, E. Acar, and S. R. Nassif. Power grid reduction based on algebraic multigrid principles. In *Proc. Design Automation Conf. (DAC)*, pages 109–112, 2003.

[74] E. Suli and D. Mayers. *An Introduction to Numerical Analysis*. Cambridge University, 2006.

[75] V. Szekely. Identification of RC networks by deconvolution: Chances and limits. *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, 45(3):244–258, 1998.

[76] S. X.-D. Tan. A general s-domain hierarchical network reduction algorithm. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 650–657, Nov. 2003.

[77] Sheldon X.-D. Tan and L. He. *Advanced Model Order Reduction Techniques in VLSI Design*. Cambridge University Press, 2007.

[78] A. Vandendorpe and P. Van Dooren. On model reduction of interconnected systems. In *Proceedings International Symposium Math. Th. Netw. Syst.*, 2004.

[79] J. M. Wang and T. V. Nguyen. Extended Krylov subspace method for reduced order analysis of linear circuit with multiple sources. In *Proc. Design Automation Conf. (DAC)*, pages 247–252, 2000.

[80] N. Wang and V. Balakrishnan. Fast balanced stochastic truncation via a quadratic extension of the alternating direction implicit iteration. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 801–805, 2005.

[81] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan. A systematic method for functional unit power estimation in microprocessors. In *Proc. Design Automation Conf. (DAC)*, pages 554–557, June 2006.

[82] B. Yan, S. X-.D. Tan, P. Liu, and B. McGaughy. SBPOR: second-order balanced truncation for passive model order reduction of RLC circuits. In *Proc. Design Automation Conf. (DAC)*, pages 158–161, June 2007.

[83] M. Zhao, R. V. Panda, S. S. Sapatnekar, T. Edwards, R. Chaudhry, and D. Blaauw. Hierarchical analysis of power distribution networks. *Proc. Design Automation Conf. (DAC)*, pages 150–155, 2000.

[84] G. Zhong, C. Koh, and K. Roy. On-chip interconnect modeling by wire duplication. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 341–346, 2002.