UNIVERSITY OF CALIFORNIA
RIVERSIDE

Compact Modeling and Analysis for Electronic and Thermal Effects of Nanometer Integrated and Packaged Systems

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

Doctor of Philosophy

in

Electrical Engineering

by

Hai Wang

June 2012

Dissertation Committee:
Dr. Sheldon X.-D. Tan, Chairperson
Dr. Yingbo Hua
Dr. Esteban Tlelo-Cuautle

The Dissertation of Hai Wang is approved:

Committee Chairperson

University of California, Riverside

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my supervisor, Dr. Sheldon X.-D. Tan for his support throughout my Ph.D. study. His great teaching helped me a lot on my journey of research, and it is his exellent guidence and support which made this dissertation possible.

I am very grateful to my dissertation committee members, Dr. Esteban Tlelo-Cuautle, and Dr. Yingbo Hua, for their great direction, insightful comments, and invaluable advice.

I would like to thank my fellow labmates from Mixed-Signal Nanometer VLSI Research Lab at UC Riverside: Dr. Pu Liu, Jian Cui, Dr. Boyuan Yan, Dr. Ning Mi, Dr. Duo Li, Dr. Ruijing Shen, Xue-Xin Liu, Zao Liu, Thom Eguia, Ryan Rakib, Omar Olmos, Sahana Swarup, etc., for the collaborative research works and the fun we had together.

There are plenty of friends who have helped me and encouraged me during my study and research. Among many others, I thank Dr. Yiming Li, Dr. Guangdeng Liao, and Dr. Yiqian Li, who have supported me through every step of the way to this dissertation.

Finally, and most importantly, I would like to thank my parents for their constant love and support. To them I dedicate this dissertation.

ABSTRACT OF THE DISSERTATION

Compact Modeling and Analysis for Electronic and Thermal Effects of Nanometer Integrated and Packaged Systems

by

Hai Wang

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, June 2012
Dr. Sheldon X.-D. Tan, Chairperson

Design and verification of today's nanometer very-large-scale integrated (VLSI) system remain a very challenging problem. For instance, the sub-90-nm technology has caused extremely large parasitic global interconnects and complicated models such as clock networks, power delivery networks and thermal models for packaged systems, which are difficult to be analyzed directly due to the limited computing resources. In addition, the high performance VLSI systems such as multi-core and emerging 3D stacked integrated systems, also lead to excessive high temperature on chip due to the elevated power densities. As a result, temperature should be explicitly managed both at design time through thermal-aware optimization and design techniques and at runtime through on-chip dynamic thermal management (DTM). Hence, accurate yet compact thermal models are required for thermal-aware design and optimization.

In this dissertation, we focus on those challenging issues and have proposed three novel techniques to facilitate the verification of the electronic and thermal effects of the nanometer integrated systems. Specifically, first, we have introduced a wideband model order reduction algorithm (WBMOR) to provide a general solution for the large system analysis problems. With the novel imaginary axis sampling technique and adaptive sample point placement, WBMOR is able to generate a reduced system

accurate within the specified frequency band. Second, we have proposed a composable thermal modeling technique (ThermComp) for compact thermal modeling. ThermComp builds compact thermal models for each basic module, and uses these models to assemble different multi-core architecture thermal models, which improves the thermal modeling and analysis efficiency at design time. Last but not least, a runtime thermal estimation and prediction method (FRETEP) framework has been proposed to enable fine-grained DTM. With a thermal sensor based error compensation method utilizing only limited number of thermal sensors, FRETEP is able to estimate and predict the full-chip thermal behavior accurately with even inaccurate power estimation. Furthermore, a power-driven thermal sensor placement algorithm has been developed for FRETEP to further enhance the thermal estimation accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, the compact modeling and analysis of electronic and thermal system for the VLSI systems are introduced. Model order reduction (MOR) technique, thermal modeling problem and runtime thermal analysis are presented in sequence. Dissertation objectives and organizations are also given.

1.1 Electronic and thermal analysis problems in VLSI system design

The *very-large-scale integration* (VLSI) refers to integrating a large number of transistors onto a single chip. At present time (the year of 2012), the number of transistors has been increased to billions, on a chip with area around $1cm^2$. Due to the excessively large number of devices on a small area, designing of the VLSI system is a very complex process, involving both design engineers and design tools. Since it is extremely hard for design engineers and design tools to manage the whole design process, *design hierarchy* has been introduced to VLSI design, which divides the design process into multiple levels. There are two types of design flow for the design hier-

archy, the *top-down* design flow and the *bottom-up* design flow, both contains many design levels. After the design of each level, the design must be verified and optimized against the design specifications, which forms a verification and optimization loop. Given a design layout, verification is generally a process of layout extraction, modeling and simulation. In this dissertation, several modeling and simulation problems in both electronic and thermal perspectives are studied and solved.

For physical verification, such as timing, signal integrity and power grid analysis, the topological layout to be verified is written into the electronic circuit form by layout extraction tools. The circuit simulator, such as SPICE like device-level simulator, is used to simulate the extracted circuit. After the simulation, the chip performances, such as area and speed, are enhanced to fit the design specifications and without violating design restrictions such as timing constraints. The simulation and optimization processes form a design loop until all the design specifications are satisfied.

Except for the electronic side, the thermal behavior of the VLSI system is becoming more and more important due to the increasing power densities of the multi-core microprocessors. In addition, thermal analysis of the VLSI system is performed at both design time for thermal verification and optimization and runtime for dynamic thermal management. At design time, thermal verification is similar to the physical verification on the electronic side and is performed after the component placement process. A thermal model of the VLSI system is generated according to different materials (thermal conductivities) of the components. A steady-state thermal analysis is performed on the thermal model with the estimated power consumptions. Similar to the electronic counterpart, for the thermal verification, hotspots and the average temperature of the chip are captured in the simulation process and during the following optimization process, the number of hotspots is minimized and the average

temperature is lowered.

Three important problems in VLSI modeling and analysis are introduced in the following three sections. In Section 1.2, the general compact electronic and thermal modeling technique by model order reduction is introduced first. Then, in Section 1.3, the importance and challenges of thermal modeling are presented. Finally, in Section 1.4, we introduce the thermal estimation techniques.

1.2 Compact modeling methods in VLSI design

The very-large scale (VLSI) circuit is complicated, with billions of transistors. Designing such complicated device is an extremely challenging process. In order to make the design process possible, the IC design usually follow a top-down approach. Such design approach starts from the system concept and finalizes in the detailed layout and routing process. During the top-down design process, designers have to check if all the design specifications are met, as a result, bottom-up verification is equally important. The verification process starts from the behavior extraction. The design to be verified is represented in a behavior model which has the similar behavior as the design and, more importantly, can be simulated by a software. The example is the verification after layout. The layout has to be extracted into circuits expressed in netlist by extraction tools/softwares, and simulated by a simulator, for example, SPICE, to check the performance of the design. However, some circuits directly extracted from the extraction tools can be extremely large. One important example is the interconnects due to the high parasitic effects of the sub-90-nm design.

Model order reduction (MOR) is an efficient technique for reducing the complexity of parasitic interconnect circuits. Existing approaches based on the Krylov subspace are very efficient [24, 60, 49, 5]. These methods, which perform implicit moment-

matching by projecting the original system onto a Krylov subspace, preserve the stability, passivity and structural information for RLC interconnect circuits.

The Krylov subspace methods, suffer one long-standing problem: the lack of global error bound. Due to this problem, designers can't predict the errors of the reduced model over a given frequency range before the reduction. For computing wire delay, coupled noise and voltage noise of interconnects in the digital circuits, such lack of global errors is a less concern because matching a few moments is accurate enough. However, for analog and RF circuits, reduced models of the extracted RLCK circuits need to be accurate for a wide frequency range and in terms of the time domain waveforms (versus delay values). The existing Krylov subspace methods can't meet such requirement, although some techniques like multi-point Krylov subspace can partially mitigate this problem at high computational cost.

Truncated balanced realization based methods (TBR) [46, 36, 54, 71] have a global error bound, but generally suffer from high computing costs, poor scalability, although some works have been made to improve the computing efficiency [76, 66, 64]. The high cost comes from computing the controllability and observability Gramians, upon which the truncation and thus reduction of weak states is performed. This problem has been partially mitigated by the Gramian approximation techniques (also called the sampling based methods) [69, 55], where Gramians are approximated in the frequency domain by computing system impulse responses at many frequency points (samples).

There are still important problems exsit in MOR, two of the most important problems are the error control over wide frequency range and the efficiency control with massive number of ports. In this dissertation, the first problem is solved by introducing an improved sampling based algorithm with adaptive sampling scheme shown in Chapter 3.

1.3 Thermal modeling and challenges

Continuous process scaling and rising device densities lead to rapid power density increase and adverse thermal effects. This problem becomes more severe as the VLSI technology scales to the nanometer ranges. Excessively high on-chip temperature can cause many severe problems such as reduced reliability of chips and elevated cooling cost of the packaging [28, 10]. Thermal management and related design problems continue to be identified by the Semiconductor Industries Association Roadmap [3] as one of the five key challenges during the next decade to achieve the projected performance goals of the semiconductor industry. Thus, accurate and efficient thermal modeling and analysis are vital for the thermal-aware VLSI design [53] to improve performance, reliability, power reduction as well as online temperature regulation techniques [10, 61].

Multi-core techniques mitigate the exponential growth of temperature resulting from the exponential increase of power density [41, 33, 4, 43, 42]. Multi-core computing simply increases the total throughput via parallel computation with lower voltage and frequency to meet the thermal constraints. But thermal effects and the resulting reliability concerns, such as NBTI effect, are influenced by the placement of CPU cores and shared caches, loading of programs, and cooling solutions at the package level. So it is vital to accurately estimate the temperature during the floorplanning and architecture design of the multi-core microprocessors.

Traditional thermal analysis solves the thermal diffusion (partial differential) equation directly using numerical approaches such as FEM (finite element), FDM (finite difference) and CFD (computational fluid dynamics). These approaches are accurate given detailed thermal structures. However, the resulting equation sizes can be prohibitively large for design explorations, hence, thermal simulation starting from

detailed thermal structures by solving thermal diffusion equations no longer meets the demanding design tasks for efficient design space exploration. As thermal effects become the first-class design constraints, efficient thermal analysis calls for more efficient solutions.

Many compact static and transient thermal modeling methods at different levels (parts, package and board) have been proposed in the past. One popular approach is based on simple thermal resistance and capacitance networks subject to different thermal boundary conditions [37, 15, 6]. The traditional limitation of these methods is to determine appropriate RC values of elements, especially for the complex geometries and boundary conditions. The RC values are typically determined and optimized against field numerical or analytic results [27, 52] and the measured data [57]. A transient simulation technique using alternating direction implicit (ADI) method is proposed to enhance the simulation efficiency of the three dimensional thermal RC circuit [67]. Another viable approach is by means of model order reduction of the large linear dynamic thermal systems after the spatial discretization. Existing approaches apply the multi-point Krylov subspace method [19] and the multivariate moment matching method [20]. Recently, many other fast thermal modeling and simulation techniques have been proposed, such as the compact thermal models based on the floorplan at architecture level [31, 61, 32], the black-box behavioral thermal models [40, 39] and the spatially adaptive thermal modeling technique ISAC [72].

There are several unsolved problems in thermal modeling. One of them is the model order reduction efficiency problem. If the original thermal model is extremely large, none of existing MOR method is able to generate the reduced thermal model due to the excessive memory required. Another problem is the flexibility and reusability of thermal models. Because the thermal analysis and floorplan optimization form a design verification and optimization loop, as introduced in Section 1.1, the opti-

mization on the floorplan may require the re-built of the whole thermal model, which is a waste of computing time and resources. In Chapter 4, a composable thermal modeling technique is proposed which solved all the mentioned problems.

1.4 Thermal analysis at both design time and runtime

The high power density caused by the increasing integration has led to excessive temperature on chips. Although multi-core architectures partially relieve the thermal density problem, local hot spots still exist due to different loads for different cores. In order to ensure the proper working conditions of transistors and chip reliability, many dynamic thermal management (DTM) methods have been proposed, including dynamic voltage and frequency scaling (DVFS), task scheduling and computing migration [22, 61]. Recently, more effective predictive dynamic thermal management methods [74, 68, 73] have been introduced to predict the future thermal behavior and perform thermal control far before the real thermal violation occurs. However, most of the DTM methods nowadays rely only on the temperature information given by a few physical thermal sensors. To watch for temperatures in the whole chip, DTM methods have to rely on the thermal estimation based on simplified, less accurate thermal models and performance counter based power estimation, which is error-prone in practice. As a result, accurate and efficient runtime full-chip thermal estimation and prediction under those realistic and non-ideal conditions (less accurate thermal models and power estimations) are crucial for the success of practical dynamic thermal management.

Recently, many thermal modeling and simulation methods were proposed for ar-

chitecture level thermal estimation [32, 72]. These methods are usually performed off-line to get the full-chip thermal behavior. Although relatively accurate, they are usually too expensive for runtime thermal estimation and mainly used for the design time thermal aware verification and optimization.

Several runtime thermal estimation techniques have been studied for DTM, including the Kalman filter based methods [58, 75], the spectral based method [17], the interpolation based method [45] and the fast simulation based method [44], etc. But both the Kalman filter and fast simulation based methods are accurate only when the mean value of the power is accurately estimated by the power estimator. Both the spectral based and interpolation based methods lack the prediction capability. And the spectral based method also requires regular placement of thermal sensors.

For the runtime thermal prediction side, some methods have been proposed to predict the future temperatures, such as autoregressive moving average (ARMA) based method [21], the recursive least square based method [73] and the workload phase based method [18]. Nevertheless, these methods can only predict the temperatures at the thermal sensors and may fail to capture the potential hot spots on a chip.

The runtime thermal estimation and prediction under realistic conditions, where the estimated power is inaccurate, is not well addressed in the past. In order to solve this problem, an error-tolerant runtime thermal estimation and prediction algorithm is presented in Chapter 5.

1.5 Objectives of this dissertation

This dissertation presents new advances in the modeling and analysis for electronic and thermal effects of VLSI systems at both design stage and runtime.

The major contributions of this dissertation are summarized as follows:

- For the general compact modeling of electronic and thermal systems, a model order reduction with wideband accuracy has been proposed. This new MOR technique, called WBMOR, can be widely used in interconnect analysis for design verification and thermal analysis at both design stage and runtime. Compared to the traditional MOR methods, WBMOR achieves higher wide frequency band accuracy and full automation thanks to the imaginary axis sampling and automatic sample point selection.

- For thermal modeling, a composable thermal modeling technique , ThermComp, has been introduced. The new composable modeling method can be used for fast thermal design space exploration for thermal-aware multi-core microprocessors design. The new approach can easily build accurate thermal systems from compact composable models for fast architecture thermal analysis and optimization and is much faster than the existing HotSpot method with similar accuracy.

- In order to assist the dynamic thermal management of microprocessors, an error-tolerant runtime thermal analysis, FRETEP, has been proposed. FRETEP is able to estimate and predict the full-chip thermal behavior accurately with inaccurate power estimation. It has very low overhead introduced and compares very favorably with the Kalman filter based approach on standard SPEC benchmarks. In addition, a power-driven thermal sensor placement algorithm has been proposed to further enhance the accuracy of FRETEP.

1.6 Organization

The rest of this dissertation is organized as follows. Chapter 2 gives the backgrounds and basics of the electronic and thermal analysis. Chapter 3 presents the new model order reduction method, called WBMOR, for general compact modeling of the electronic and thermal system. A new thermal modeling technique, named ThermComp, is shown in Chapter 4. A new runtime thermal estimation and prediction algorithm, FRETEP, is presented in Chapter 5. Finally, Chapter 6 concludes the dissertation.

Chapter 2

Basics of Modeling and Analysis of Nanometer Integrated and Packaged Systems

In this chapter, the backgrounds and basics of the VLSI modeling and analysis considering electronic and thermal effects are presented. In Section 2.1, the model order reduction problem is shown mathmatically first, following with the review of the classical MOR methods. Next, the basics of thermal modeling is shown in Section 2.2 and the standard thermal simulation technology is presented in Section 2.3. Finally, Section 2.4 summarizes this chapter.

2.1 Compact modeling of the linear time-invariant system

Compact linear system modeling is the basic of efficient electronic and thermal system analysis. In this section, the state space model and its corresponding reduced model are introduced. The basics of the model order reduction (MOR) techniques are also given in this section.

2.1.1 The standard state space model

For a linear time-invariant (LTI) system, *standard state space model* is shown as the following in time domain [5]

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t). \end{aligned} \tag{2.1}$$

After a Laplace transform, the standard state space model in the frequency/"s" domain is

$$\begin{aligned} sx(s) &= Ax(s) + Bu(s) \\ y(s) &= Cx(s). \end{aligned} \tag{2.2}$$

For the dimensions of the state space components, we have $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{p\times n}$ and $u(t) \in \mathbb{R}^p$. This means x, u are vectors while A, B, C are matrices[1].

From now on, the state space model in this dissertation will be in the frequency domain by default unless mentioned specially.

[1] B and C can also be vectors when $p = 1$.

The components in the state space model are categorized into two groups. The first group is the variables, including the state variable x, output variable y and input variable u. The second group is the mapping/scaling matrices, including A, B, and C, which have *fixed* values.

2.1.2 The descriptor state space model

Although the standard state space model has been studied intensively, it does not suit for the VLSI modeling where the Modified Nodal Analysis (MNA) is the standard. MNA is a descriptor state space model which is slightly different from the standard state space model.

The *descriptor state space model* has a form like

$$\begin{aligned} sEx(s) &= Ax(s) + Bu(s) \\ y(s) &= Cx(s). \end{aligned} \tag{2.3}$$

The only difference between descriptor form and standard form is that the descriptor form has an extra $E \in \mathbb{R}^{n\times n}$ matrix. It is easy to see that if we multiply E^{-1} (assume E is invertible) on each side of the first equation in (2.3), we can transform descriptor form into the standard form.

Electronic circuits are usually modeled using MNA as

$$\begin{aligned} Gx(s) + sCx(s) &= Bu(s), \\ y(s) &= Lx(s), \end{aligned} \tag{2.4}$$

which is in descriptor form[2]. Similar to the standard form, for dimension, there are $x \in \mathbb{R}^n$, $G \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $L \in \mathbb{R}^{p\times n}$ and $u(t) \in \mathbb{R}^p$. Every

[2]It would be more obvious if we write the first equation as $sCx(s) = -Gx(s) + Bu(s)$.

component has its physical meaning. The state $x(s)$ contains node voltages and/or branch currents; $u(s)$ contains voltage sources and/or current sources; $y(s)$ includes node voltage and/or branch current observations; G mainly has conductance information (and voltage source and inductor position information); C contains capacitor and inductor information; B and L connect the inputs and outputs to the states: they specify where the inputs are injected into and where the outputs come from, respectively.

2.1.3 The reduced model

The reduced model, of course, should be smaller in size than the original model.

It must be emphasized that the general idea of MOR is to generate a model with smaller size but similar input and output behavior compared to the original model, i.e., given the same inputs to both the original and the reduced systems, the outputs should be similar. As shown in 2.1.1, the variables u and y represent the input and output and can be observed. Thus, the size of u and y cannot be reduced. In contrast, the internal variable x is invisible from outside. If it is reduced to a smaller size, with the same input, the reduced model still has the chance to produce the similar output as the original model. Thus, it is very clear that we can *only* reduce x. If we have reduced $x \in \mathbb{R}^n$ to $\hat{x} \in \mathbb{R}^q$ where $q < n$, the corresponding reduced model is represented as

$$\begin{aligned} \hat{G}\hat{x}(s) + s\hat{C}\hat{x}(s) &= \hat{B}u(s), \\ y(s) &= \hat{L}\hat{x}(s). \end{aligned} \tag{2.5}$$

It can be seen from (2.5) that $G \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$ and $L \in \mathbb{R}^{p\times n}$ matrices in (2.4) are also reduced to $\hat{G} \in \mathbb{R}^{q\times q}$, $\hat{C} \in \mathbb{R}^{q\times q}$, $\hat{B} \in \mathbb{R}^{q\times p}$ and $\hat{L} \in \mathbb{R}^{p\times q}$ respectively.

Once we have generated the $\hat{G}$, $\hat{C}$, $\hat{B}$ and $\hat{L}$ matrices, the MOR procedure is done.

2.1.4 Model order reduction method

We have presented the form of the reduced model, a natural question is how can we obtain this reduced model from the original model, that is, how to obtain $\hat{G}$, $\hat{C}$, $\hat{B}$ and $\hat{L}$. Please note that it is usually impossible to generate a reduced model with exactly the same behavior as the original model. For example, usually there is no guarantee that the frequency response of the reduced model is the same as the one of the original model for $s \in (-\infty, \infty)$. However, practically, people usually do not need such perfect reduced system since people may only require certain information of the original system to be retained, for example, only a frequency band.

According to the way of generating the reduced system, the MOR methods can be divided into two categories: the projection based methods and non-projection based methods. Due to the numerical advantages of projection, Modern MOR methods are mostly projection based and only the projection based methods are discuss in this dissertation. For the projection based method, there are two major MOR techniques, one is the moment matching based method, the other one is the sampling based method. In this section, the projection framework will be introduced first, following with the two major projection based MOR techniques.

The projection framework

In this section, we will show how to obtain the reduced matrices $\hat{G}$, $\hat{C}$, $\hat{B}$ and $\hat{L}$ through projection methods. Because of the important role the projection method plays in the modern MOR technology, it is explained in a very detailed way.

Assume there is a matrix $V \in \mathbb{R}^{n\times q}$, where $q < n$, satisfies

$$\tilde{x}(s) \equiv V\hat{x}(s) \approx x(s). \tag{2.6}$$

Here we introduced a new variable $\tilde{x}(s)$ as an *approximation* of x. Note that although the reduced state variable $\hat{x}(s)$ has a size of q, $\tilde{x}(s)$ has a size of n, which is the same as the original state variable $x(s)$.

Assume the input u contains impulse signals with magnitude 1, that is $u(s) = [e_1, e_2, ..., e_p]$ where e_i is a vector with 1 at the i-th position and 0 elsewhere, the approximation in (2.6) together with (2.4) will generate a **residual** $R(s)$ as

$$R(s) \equiv GV\hat{x}(s) + sCV\hat{x}(s) - B. \tag{2.7}$$

Generally, $R(s) \neq 0$. We left multiply $R(s)$ with a matrix W^T where $W \in \mathbb{R}^{n\times q}$ and force it to 0 like

$$W^T R(s) = W^T GV\hat{x}(s) + sW^T CV\hat{x}(s) - W^T B = 0. \tag{2.8}$$

Now, we have successfully generated the reduced model if we rewrite (2.8) as

$$\begin{aligned} W^T GV\hat{x}(s) + sW^T CV\hat{x}(s) &= W^T B, \\ y(s) &= LV\hat{x}(s). \end{aligned} \tag{2.9}$$

Obviously, $\hat{G}$, $\hat{C}$, $\hat{B}$ and $\hat{L}$ are obtained as the following

$$\hat{G} = W^T GV,\ \hat{C} = W^T CV,\ \hat{B} = W^T B,\ \hat{L} = LV. \tag{2.10}$$

Match the original system by projection

Consider the original system shown in (2.4) and the reduced system shown in (2.5) where the original state $x(s)$ and the reduced state $\hat{x}(s)$ are related by the approximation (2.6). We can write the approximation further as

$$\tilde{x}(s) = V(V^T Q(s) V)^{-1} V^T Q(s) x(s) \tag{2.11}$$

where $Q(s) = G + sC$ for simplicity.

Let us denote $P = V(V^T Q(s) V)^{-1} V^T Q(s)$. It is clear that P is an oblique projector along $Q^T V$ onto the range space of V. As the goal of model reduction, we want $\tilde{x}(s)$ approximates $x(s)$ as good as possible, i.e., in the ideal case, we want $\tilde{x}(s) = x(s)$. There is an *invariant property* of the projection that if the subspace spanned by a matrix F is inside the range space of a projector P, F remains invariant after the projection by P, i.e., $F = PF$. According to the invariant property, if the range space of V includes $x(s)$ for some frequencies, there will be $\tilde{x}(s) = x(s)$ for these frequencies. What is more, the range space of V can also include the Taylor series of $x(s)$ at certain frequencies, such that the Taylor series of $\tilde{x}(s)$ will be the same as those of $x(s)$ at these frequencies.

Different projection based MOR methods find different W and V through different ways. For example, V can be formed such that its range space should contain only the values of $x(s)$ at certain frequencies or it can be also formed by allowing its higher order Taylor terms at these frequencies. W can be formed to retain the passivity property of the original system, force the stability of the reduced system or further increase the approximation accuracy. In the following subsections, two most popular projection based MOR methods are presented.

The moment matching based method

As introduced previously, it is usually impossible to generate a reduced system without accuracy lost. The moment matching based methods find a reduced system with accuracy guaranteed around an expansion frequency point. The basic idea of moment matching is to form the projection matrix V to span the same subspace as the one spanned by the first q Taylor terms (called moments here) at a frequency point, where q is chosen to balance the accuracy and the reduced model size. According to the previously introduced projection framework, the approximation state $\tilde{x}(s)$ will have the same first q Taylor expansion terms as the original state $x(s)$ because the first q Taylor terms are included inside the projection matrix V's range space and the invariant property of the projector will retain these terms in the reduced approximation state $\tilde{x}(s)$.

We take the expansion point at the frequency $s = 0$ for example, which is known as the DC expansion. The original state $x(s)$ can be written in Taylor series as

$$\begin{aligned} x(s) &= (sC + G)^{-1}B \\ &= (G(sG^{-1}C + I))^{-1}B \\ &= (I - (-sG^{-1}C))^{-1}G^{-1}B \\ &= \sum_{i=0}^{\infty}(-G^{-1}C)^i G^{-1}Bs^i. \end{aligned} \tag{2.12}$$

Denote $A = -G^{-1}C$ and $K = G^{-1}B$, (2.12) can be rewritten as

$$x(s) = \sum_{i=0}^{\infty} A^i K s^i, \tag{2.13}$$

and the moments are defined as

$$M_i = A^i K. \tag{2.14}$$

If we can find a V which spans the first q moments like

$$\operatorname{span} V = \operatorname{span}\{M_0, M_1, M_2, \ldots, M_{q-1}\}, \tag{2.15}$$

the first q moments in the reduced system and original system are matched through projection, as shown in the projection part.

Directly generating the projection matrix V in (2.15) is expensive. Luckily, the moments actually span a Krylov subspace

$$\operatorname{span} V = \operatorname{span}\{M_0, M_1, M_2, \ldots, M_{q-1}\} = \mathcal{K}\{A, K\}, \tag{2.16}$$

where a Krylov subspace $\mathcal{K}\{A, K\}$ is defined as

$$\mathcal{K}\{A, K\} = \operatorname{span}\{K, AK, A^2K, \ldots, A^{q-1}K\}. \tag{2.17}$$

Forming a Krylov subspace is computationally cheap and numerically stable. Especially, Anoldi algorithm can be used to generate the desired Krylov space V in an efficient way.

The sampling based method

Different from the moment matching based method, which expands at one frequency point and matches the expansion terms up to a given order, the sampling based method matches the state frequency response (the zero-order moment, as in the moment matching based method) at different frequencies.

The sampling based method first takes state frequency responses (samples) at different frequencies, for example at the k-th frequency, s_k, as

$$z(s_k) = (s_k C + G)^{-1} B. \tag{2.18}$$

Then, the projection matrix V is generated to span the same subspace as spanned by the samples, like

$$\text{span}\, V = \text{span}\{z(s_1), z(s_2), \ldots, z(z_q)\}, \tag{2.19}$$

assume q samples are taken.

The sampling based method can be simply viewed as the multiple expansion point version of the moment matching based method, with each point expanded only up to the zero-th moment. It is computationally more expensive than the moment matching based method (single expansion point version), since there is only one original system solve (LU decomposition) required (at the only expansion point) in moment matching, while there are many solves (one for each sample point) for the sampling based method. However, the sampling based method has better global accuracy control, according to the flexible distribution of the sample points.

Moreover, the sampling based method can be easily related to the truncated balanced realization (TBR) method, a well known MOR algorithm developed in the control community, to further reduce the model size. It is as simple as introducing an SVD process in the projection matrix (V) generation process.

More details of the sampling based MOR method and its relation with the TBR method can be found in Chapter 3.

2.2 Thermal modeling and analysis

Thermal analysis of the VLSI system is one of the focuses of this dissertation. In this section, how the thermal system is equivalenced as an electronic system and described as an LTI model are shown.

2.2.1 Thermal modeling from the first principles

At the circuit, package and board levels, the heat transfer phenomena is governed by the following heat differential equation [13]:

$$\rho C_p \frac{\partial T(\vec{r},t)}{\partial t} = \nabla \cdot [\kappa(\vec{r},T) \cdot \nabla T(\vec{r},t)] + g(\vec{r},t), \tag{2.20}$$

which is subject to the following general thermal boundary condition (Robin's boundary condition)

$$\kappa(\vec{r},T) \frac{\partial T(\vec{r},t)}{\partial n_i} = h_i(T(\vec{r},t) - T_{amb}). \tag{2.21}$$

In (2.20), T (K) is the temperature, ρ (Kg/m^3) is the density of the material, C_p $(J/Kg{\cdot}K)$ is the mass heat capacity, κ $(W/m{\cdot}K)$ is the thermal conductivity, and g (W/m^3) is the heat energy generation rate. In (2.21), n_i is the outward direction normal to the boundary condition i, h_i (W/m^2K) is the heat-transfer coefficient (for the convective interface), and T_{amb} is the ambient temperature surrounding the thermal systems. If $h_i = 0$, the boundary condition is adiabatic (isolated) , otherwise, it is convective. Note that the thermal conductivity κ differs for different materials and also depends on the temperature.

For the finite difference based numerical analysis, a seven-point discretization scheme can be applied for (2.20) in three dimensions. The thermal structure will be decomposed into numerous rectangular parallelepipeds, which may be of non uniform

sizes and shapes. The adjacent elements interact with each other via heat diffusion and the elements interact with boundaries via specific boundary conditions. Each element may have power sources, temperature, and equivalent thermal capacitance and resistance to its adjacent elements. Assuming homogeneous material and temperature independent κ, $\nabla \cdot [\kappa(\vec{r}, T) \cdot \nabla T(\vec{r}, t)]$ becomes $\kappa \nabla^2 T(\vec{r}, t)$. Equation (2.20) becomes a linear partial differential equation

$$\rho C_p \frac{\partial T(\vec{r}, t)}{\partial t} = \kappa [\frac{\partial^2 T(\vec{r}, t)}{\partial x^2} + \frac{\partial^2 T(\vec{r}, t)}{\partial y^2} + \frac{\partial^2 T(\vec{r}, t)}{\partial z^2}] + g(\vec{r}, t). \tag{2.22}$$

After the space discretization, the temperature $T(x, y, z, t)$ at the grid point (i, j, k) is replaced by $T(i\Delta x, j\Delta y, k\Delta z, t)$. We denote $T(i\Delta x, j\Delta y, k\Delta z, t)$ by $T_{i,j,k}$ in this dissertation. According to the central difference discretization, we will have the following discretized equation for grid (i, j, k)

$$\begin{aligned} \rho C_p M \frac{\partial T(x, y, z, t)}{\partial t} &= -2(G_x + G_y + G_z) T_{i,j,k} + G_x T_{i-1,j,k} \\ &+ G_x T_{i+1,j,k} + G_x T_{i,j-1,k} + G_x T_{i,j+1,k} \\ &+ G_x T_{i,j,k-1} + G_x T_{i,j,k+1} + M g_{i,j,k,t}, \end{aligned} \tag{2.23}$$

where Δx, Δy, Δz are the discretization steps along the x, y, z axes, $M = \Delta x \Delta y \Delta z$. $G_x = \kappa \Delta y \Delta z / \Delta x$, $G_y = \kappa \Delta x \Delta z / \Delta y$ and $G_z = \kappa \Delta x \Delta y / \Delta z$.

2.2.2 Boundary condition modeling

For the boundaries, which interact with the outside directly via convection or other heat exchange mechanisms, a proper thermal condition (Robin's boundary condition) in terms of equivalent thermal resistance and independent source should be added at the ports as shown below.

For better illustration, let us consider only one dimension (x direction). From (2.21), we carry out the discretization on the x direction as

$$\begin{aligned} \kappa\frac{\partial T}{\partial x} &= h_x(T_{amb} - T), \\ \kappa\frac{T_0 - T_1}{\Delta x} &= h_x(T_{amb} - T_0). \end{aligned} \tag{2.24}$$

Here T_0 represents the temperature of a node just at the boundary and T_1 is the temperature of the adjacent node inside the module.

(2.24) is interpreted as the temperature relation between two branches (T_0, T_1) and (T_1, T_{amb}). Since we already know the thermal conductance between T_0 and T_1 through discretization, which is denoted as G_x, (2.24) readily becomes a three-branch KCL equation at T_0

$$(T_0 - T_1)G_x = \frac{h_x \Delta x G_x}{\kappa} T_{amb} - \frac{h_x \Delta x G_x}{\kappa} T_0. \tag{2.25}$$

Then, an equivalent circuit is generated with a current source (with current $\frac{h_x \Delta x G_x}{\kappa} T_{amb}$) and a resistor (with conductance $\frac{h_x \Delta x G_x}{\kappa}$) connected to the ground at T_0.

2.2.3 Equivalent circuit of the thermal model

According to the previous discussions, if there are n discretized grids with specific boundary conditions, the equivalent thermal circuit can be modeled using an ordinary differential equation [13]

$$C\frac{dT(t)}{dt} + GT(t) = BU(t), \tag{2.26}$$

where $T(t) \in \mathbb{R}^n$ is the temperature vector containing the temperatures of the n thermal nodes, $C \in \mathbb{R}^{n \times n}$ is the thermal capacitance matrix, $G \in \mathbb{R}^{n \times n}$ is the thermal conductance matrix, $B \in \mathbb{R}^{n \times p}$ is the position matrix of the input where $B_{i,j}$ denotes

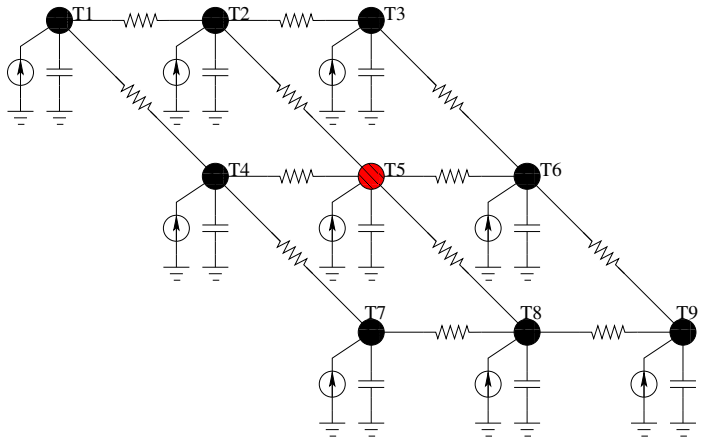


Figure 2.1: A nine-grid equivalent thermal circuit. Each grid has a thermal node T_i denoted as a solid circle (black or red dashed), a thermal capacitor and a current source representing the power dissipation at the grid. There is also a thermal resistor between each pair of the adjacent thermal nodes. A thermal sensor, denoted as the red dashed circle (T_5), is placed at the center grid.

the portion of the jth functional block power injects into the ith thermal node and $U(t) \in \mathbb{R}^p$ contains the power dissipations of the p functional blocks. The right hand side of (2.26) is also written as

$$J(t) = BU(t), \tag{2.27}$$

where $J(t) \in \mathbb{R}^n$ represents the power dissipations of n grids. It is obvious that (2.26) has the same structure as (2.4), both in MNA form. As a result, the thermal system can be equivalenced into an electronic circuit as shown in the following.

A two dimensional nine-grid equivalent thermal circuit example is shown in Fig. 2.1. It can be seen in the figure, each grid has a thermal node T_i, a thermal capacitor and a current source representing the power dissipation at the grid. There is also a thermal resistor between the adjacent thermal nodes. One thermal sensor, denoted as the red dashed circle, is placed at the center grid in this example.

2.3 Basics of runtime thermal estimation

So far, we have already shown how to get the compact thermal/electronic model in Section 2.1 and how to generate thermal model in Section 2.2. In this section, we present the runtime thermal estimation techniques, using the generated thermal model. In short, runtime thermal estimator feeds the estimated runtime power into a thermal model (compact thermal model should be used to reduce overhead) as the input and takes the output as the runtime temperature estimation. The input of the thermal estimator, i.e., runtime power estimation, is introduced first in Section 2.3.1. Next, in Section 2.3.2 we show how to perform thermal estimation numerically using the power input and the thermal model.

2.3.1 Runtime power estimation

Runtime power of a microprocessor can be divided into two parts: the static power (P_{sta}) and the dynamic power (P_{dyn}). The static power depends on temperature and can be relatively accurately estimated. The dynamic power, including capacitor power and short circuit power, is caused by the signal switching between Vdd and Gnd at the transistors and as a result is a function of utilization. Due to the complex behavior of the microprocessor at runtime, accurately estimating the functional block (FB) level dynamic power is very hard. There are several off-line FB level power estimators available [29, 11]. They take the power event counts and multiply the counts by their corresponding unit powers to get the power estimations for FBs. These power estimators are considered to be accurate, for example, accurate to within 5% to 10% [56]. However, they are too expensive for runtime usage because there are too many power events to be monitored. Runtime power estimators [70, 56] have much smaller overhead by monitoring only a few carefully chosen power events, and as a trade-off,

has larger errors compared to the off-line power estimators. For example, in [70], one power event is monitored for each FB and the total dynamic power is estimated as

$$P_{dyn} = \sum_{i=1}^{n_p} E_i \times P_i, \tag{2.28}$$

where n_p is the number of FBs, E_i is the count of power event at the ith FB, P_i is the unit power per power event for the ith FB, which is determined through linear regression using a number of benchmarks. In [56], the authors further reduced the number of power events from n_p to around 10 by analyzing the power event correlations among FBs and choosing only the most important ones.

2.3.2 Runtime thermal estimation

The heat differential equation of the chip can be spatially discretized using finite difference method in the three dimensional space to generate an equivalent thermal circuit, as shown previously in Section 2.2. If there are n discretized grids with specific boundary conditions, the equivalent thermal circuit modeled using an ordinary differential equation is rewritten here for convenience as

$$C\frac{dT(t)}{dt} + GT(t) = BU(t), \tag{2.29}$$

where $T(t) \in \mathbb{R}^n$ is the temperature vector containing the temperatures of the n thermal nodes, $C \in \mathbb{R}^{n\times n}$ is the thermal capacitance matrix, $G \in \mathbb{R}^{n\times n}$ is the thermal conductance matrix, $B \in \mathbb{R}^{n\times n_p}$ is the position matrix of the input where $B_{i,j}$ denotes the portion of the j-th functional block power injects into the i-th thermal node and $U(t) \in \mathbb{R}^{n_p}$ contains the power dissipations of the n_p functional blocks. The right

hand side of (2.29) is also written as

$$J(t) = BU(t), \tag{2.30}$$

where $J(t) \in \mathbb{R}^n$ represents the power dissipations of n grids.

In order to calculate the temperature T in time domain, Backward Euler (BE) is used to discretize (2.29) as

$$(\frac{C}{h} + G)T(t+h) = \frac{C}{h}T(t) + J(t+h), \tag{2.31}$$

where h is the time step. Given the initial value $T(0)$ and the input $J(t)$ for all time points, the subsequent temperature $T(t)$ can be calculated iteratively using (2.31).

2.4 Summary

This chapter has provided the basics of modeling and analysis for electronic and thermal effects of nanometer integrated and packaged systems. The model order reduction method has been introduced focusing on the projection framework and two projection based methods: the moment matching based method and the sampling based method. The thermal model in the LTI form has been derived from the basic heat differential equation and the corresponding boundary conditions. We have also shown the basics of runtime power estimation and the numerical runtime thermal estimation method. In the following chapters of this dissertation, the new model order reduction, thermal modeling and thermal estimation methods will be presented.

Chapter 3

Wideband Model Order Reduction for Compact Modeling of Nanometer Integrated and Packaged Systems

The LTI system modeled as the ordinary differential equations is widely used in electronic and thermal simulation. The simulation efficiency can be significantly boosted by using a reduced model, which has much smaller size than the original model but with similar input and output behaviors, generated by model order reduction (MOR) methods. The first problem to be solved in this dissertation is to develop a new MOR method, which is able to generate reduced models with balanced size and accuracy at an acceptable computational cost. The generated reduced model will be used extensively at the analysis stage, as shown later in Chapter 4 and Chapter 5.

The basics of the MOR method can be found in Section 2.1. In this Chapter, the sampling based MOR method, based on which the new MOR method is built, is

reviewed first in Section 3.1. Next, the new MOR method, WBMOR, is presented in Section 3.2. The experimental results are shown in Section 3.3. Finally, Section 3.4 concludes this chapter.

3.1 Review of the sampling based model order reduction method

In this section, the sampling based MOR method is reviewed. The standard TBR-based method, which has a strong connection with the sampling based method, is introduced first followed by the derivation of the sampling based method.

3.1.1 The standard TBR based reduction method

We first review the standard TBR method.

Consider a linear dynamic system in a standard state-space form, rewritten from (2.1) as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned} \tag{3.1}$$

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

$$\begin{aligned} AX + XA^T + BB^T &= 0, \\ A^TY + YA + C^TC &= 0. \end{aligned} \tag{3.2}$$

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

$$T^{-1}XYT = \Sigma = diag({\sigma_1}^2, {\sigma_2}^2, \ldots, {\sigma_n}^2), \tag{3.3}$$

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

$$\begin{bmatrix} W_1^T \\ W_2^T \end{bmatrix} XY \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}, \tag{3.4}$$

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

$$\begin{aligned} \dot{x}(t) &= A_r x(t) + B_r u(t), \\ y(t) &= C_r x(t), \end{aligned} \tag{3.5}$$

where $A_r = W_1^T A V_1$, $B_r = W_1^T B$, $C_r = C V_1$. 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.

3.1.2 The sampling based reduction framework

To mitigate the high computational cost of standard TBR method, a fast TBR method was proposed (called PMTBR) [55], where the Gramians are approximated using Monte-Carlo sampling approach. Specifically, if we look at the controllability Gramian, we need to solve the following Lyapunov equation:

$$AX + XA^T + BB^T = 0. \tag{3.6}$$

Alternatively, the Gramian X can also be represented in frequency domain as

$$X = \int_{-\infty}^{+\infty} (j\omega I - A)^{-1} BB^T (j\omega I - A)^{-H} d\omega, \tag{3.7}$$

where superscript H denotes Hermitian transpose. As a result, computing Gramian X boils down to evaluating the definite integral in (3.7) [35]. This can be done using numerical quadrature methods.

For an integral function $f(x)$, numerical quadrature methods try to approximate it as

$$\int_a^b f(x)dx \approx \sum_{k=1}^{m} w_k f(x_k), \tag{3.8}$$

where w_k, $k = 1, 2, ..., m$, are referred to as the quadrature point weights, while the interpolation points x_k, $k = 1, 2, ..., m$, are called quadrature points. The selections of w_k and x_k depend on the quadrature methods such as Newton-Cotes, Gaussian quadrature rules [35].

For the sampling-based reduction, our goal is not just computing the Gramian X, but the dominant eigenspace to form the projection matrix. As a result, let s_k be the

kth sample point, if we define

$$z_k = z(s_k) = (s_k I - A)^{-1} B, \tag{3.9}$$

which is called kth snapshot of the system in (3.1) in the frequency domain. Then X can be approximated as

$$\hat{X} = \sum_{k=1}^{m} w_k z_k z_k^H = Z W^2 Z^H, \tag{3.10}$$

where

$$Z = [z_1, z_2, \ldots, z_m], \tag{3.11}$$

and W is a diagonal matrix with diagonal entries $w_{kk} = \sqrt{w_k}$. w_k comes from a specific quadrature method. If we perform the singular value decomposition (SVD) on ZW and obtain

$$ZW = V \Sigma U, \tag{3.12}$$

then V, which gives the dominant eigenspace of X, is used as the projection matrix.

The main advantage of the sampling-based TBR methods over the Krylov subspace methods is that they are globally more accurate since they sample in a wide frequency range. However, how to efficiently perform the sampling (how many points and which points should be chosen) to control the errors of the reduced models still remains an open problem. Two methods were proposed recently to resolve this problem: a re-sampling scheme [59] and the ARMS method [65]. In [59], a statistical re-sampling scheme is used. In each iteration, many reduced models are computed by re-sampling from a common pool of candidate sample points in a given frequency range, and the variations among all the reduced models are calculated at many fre-

quency points called search points. The search point with the largest variation will be taken as the sample point. The ARMS method uses residue-minimization technique to estimate the error (without building the reduced models) and add sample points where the largest residue exists. However, these methods may not find the best sample points or suffer from the large computational cost.

3.2 New wideband sampling based reduction method

In this section, we address the efficient sampling problem of the sampling-based reduction methods. In Section 3.2.1, we first present the problem and three challenges of solving the problem. Then, we conquer the first one with the complex-valued reduction and present some of its important properties. Next, in Section 3.2.2, we conquer the second challenge and show how error is estimated in the proposed method in Section 3.2.3. The third challenge is solved in Section 3.2.4 using an adaptive sampling scheme. Finally, the whole algorithm flow and analysis are presented in Section 3.2.5.

3.2.1 Challenges of efficient sampling

Typically, in the frequency band where the frequency response of a system is smooth, a sample point (its corresponding subspace) can cover[1] a wide frequency range. While in the spiky district, where many system poles exist, a sample point will only cover a small interval. As a result, pure uniform or random Monte Carlo sampling schemes may over sample at the smooth frequency bands and suffer accuracy loss at the spiky bands. Let's illustrate this via an example of a RLC circuit of size 640. 20

[1] *cover* here means the reduced model generated by the projection matrix matches the original system in the specified frequency band.

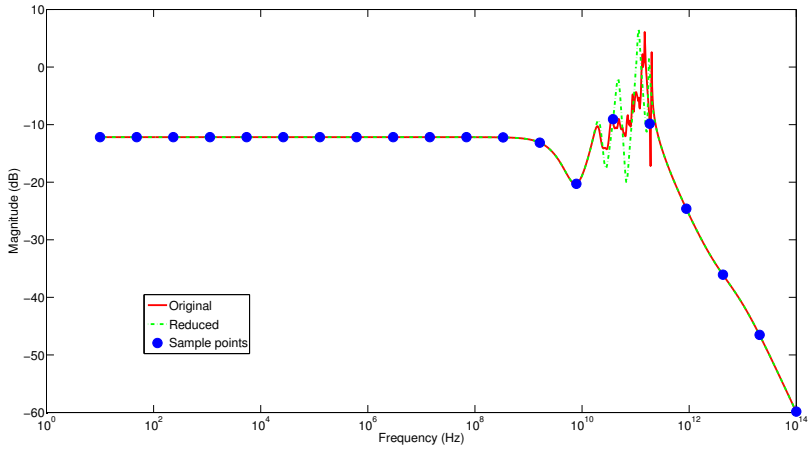


Figure 3.1: Real axis sampling with 20 sample points.

uniformly distributed sample points are used to generate a reduced model of size 20. Obviously, as shown in Fig. 3.1, the accuracy for this reduction is not satisfactory – the approximation is good for the smooth frequency bands, but it fails to match the area from 10^{10}Hz to 10^{12}Hz which has a lot of peaks.

One way to mitigate this problem is by a static sampling scheme which simply adds more sample points at the spiky frequency bands. Unfortunately, knowing the frequency response of a large system as *a priori* is not feasible since it requires to solve the large system over the wide frequency range of interest. Moreover, we will not known how many samples will lead to an acceptable reduced model unless we perform the reduction many times.

A better way to perform reduction efficiently is via adaptive sampling. In the iterative adaptive sampling method, new sample points are added at the current iteration based on the information of the last iteration. However, there are three challenges we need to solve in this scheme. First, we desire the reduced system to have zero errors at the sample points and thus guarantee the convergence as more samples are added. This seems simple, but as will be shown later, existing real axis sampling

does not have this property. Secondly, in order to guide the adaptive sampling at the next iteration and to indicate the convergence of the algorithm, we need to know the errors of the reduced system at the current iteration over the frequency range. However, calculating the errors explicitly requires expensive solving of the original model at many points, which should be avoided as much as possible. As a result, an error estimator with a cheaper computational cost is preferred. Thirdly, assume we have solved the first two problems, the sample points still need to be placed carefully. For example, how can we accelerate the convergence of the iteration? How can we prevent missing the potential sample points? The following subsections will provide our solutions to all the three challenges.

3.2.2 New complex-valued sampling based reduction method

As shown in Fig 3.1, the reduced model may fail to match the original model exactly even at the sample points, such as at the fifth sample points from right to left. In this subsection, we show through sampling along the imaginary axis, we will obtain the exact match at the sample points.

Existing reduction techniques based on Krylov subspace method are mainly expanded at $s = 0$, while multi-point expansion methods and fast TBR methods are expanded at multiple frequency points, but along the real axis. In other words, $s_k = \sigma_k$ instead of $s_k = j\omega_k$ is used in (4.19) [2]. The reason is that if $s = j\omega$, the moments in Krylov subspace methods and snapshots in the sampling based approaches will become complex, causing the reduced matrices $\hat{G}$, $\hat{C}$, $\hat{B}$ and $\hat{L}$ to be complex matrices. This complex reduced system does not exist in the real world and is hard to be realized into a reduced RLC circuit. We notice that expanding along the imag-

[2] We remark that in [55], the authors mentioned PMTBR can sample in the whole complex plane, but no further detail was provided. Here we claim that the imaginary axis should be the only place to sample.

inary axis is not an issue for explicit moment matching method like AWE and CFH methods [14] as complex Padé approximation can be carried out to compute the poles and residues.

In this dissertation, we propose a new sampling scheme Complex-Valued Sampling based Truncated Balanced Realization (CVSTBR) as a part of our WBMOR algorithm. CVSTBR samples along the imaginary axis $s = j\omega$ to preserve the physical meaning of the Gramian approximation. In addition, the new CVSTBR method leads to **real** reduced system. We show that the resulting reduced system matches exactly with the original system at the sample points, which is not the case for sampling along the real axis (except for $s = 0$).

The projection framework of sampling based model reduction

The projection framework has been introduced briefly in Section 2.1.3 for general projection based MOR method. Here, it is demonstrated in details specially for the sampling based model reduction method.

For an interconnect circuit modeled as a RLC dynamic system with n states and p ports, the system equation (3.1) can be formulated in descriptor form in the Laplace domain as

$$\begin{aligned} Gx(s) + sCx(s) &= Bu(s), \\ y(s) &= Lx(s), \end{aligned} \tag{3.13}$$

where $G \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{n\times n}$ contain elements information such as conductance, capacitance and inductance. B and L matrices describe the positions of inputs and outputs in the network respectively. And typically there is $B = L^T \in \mathbb{R}^{n\times p}$ which means the inputs and outputs are identical. $x(s) \in \mathbb{C}^n$ is the state variable vector

represents node voltages and branch currents, where $s = \sigma + j\omega \in \mathbb{C}$. Then we rewrite (3.9), (3.10), (3.11) and (3.12) here for convenience:

$$z_k = z(s_k) = (s_k C + G)^{-1} B, \tag{3.14}$$

$$\hat{X} = \sum_{k=1}^{m} w_k z_k z_k^H = Z W^2 Z^H, \tag{3.15}$$

$$Z = [z_1, z_2, \ldots, z_m], \tag{3.16}$$

$$ZW = V \Sigma U. \tag{3.17}$$

After the reduction, we will have a reduced model

$$\begin{aligned} \hat{G} x_r(s) + s\hat{C} x_r(s) &= \hat{B} u(s), \\ y_r(s) &= \hat{L} x_r(s), \end{aligned} \tag{3.18}$$

where $\hat{G} = V^T G V$, $\hat{C} = V^T C V$, $\hat{B} = V^T B$, $\hat{L} = LV$. V is computed from (3.17) and $V \in \mathbb{R}^{n \times q}$. $q \ll n$ is the dimension of the reduced system. $x_r(s) \in \mathbb{C}^q$ is the reduced state vector in the reduced system.

Assume we have p impulse inputs $u(s) = [e_1, e_2, ..., e_p]$ applied to the system in (3.13), where e_i is a $p \times 1$ vector whose i-th position is 1 and the rest is 0. Define the impulse response matrix as

$$z(s) = (G + sC)^{-1} B = [x^1(s), x^2(s), ..., x^p(s)] \in \mathbb{C}^{n \times p}, \tag{3.19}$$

where $x^i(s)$ is the state response according to e_i. Correspondingly, we have

$$z_r(s) = (\hat{G} + s\hat{C})^{-1} \hat{B} = [x_r^1(s), x_r^2(s), ..., x_r^p(s)] \in \mathbb{C}^{q \times p} \tag{3.20}$$

in the reduced system. From now on, we will use $z(s)$ and $z_r(s)$ instead of $x(s)$ and $x_r(s)$ for the identity $u(s)$ case.

The approximate impulse response $\tilde{z}(s) \in \mathbb{C}^{n \times p}$, which is the approximation of the original impulse response $z(s)$, can be recovered from the reduced impulse response $z_r(s)$ as

$$\tilde{z}(s) = V z_r(s) \approx z(s). \tag{3.21}$$

It can be also written as

$$\tilde{z}(s) = V(V^T Q(s) V)^{-1} V^T Q(s) z(s), \tag{3.22}$$

where $Q(s) = (G + sC)$ for simplicity [26] and we will use Q later for short.

Let's denote $P = V(V^T Q V)^{-1} V^T Q$. Since $P^2 = P$, it is clear that the matrix P is a projector along $Q^T V$ onto V. The range space of P is the subspace spanned by V and the orthogonal complement of its null space is spanned by $Q^T V$. Also notice that P is a function of s. Then, combining (3.21) and (3.22), we have

$$\tilde{z}(s) = V z_r(s) = P z(s) \approx z(s). \tag{3.23}$$

We have shown the relationship between the impulse responses of the original system and the reduced system in (3.23). It is obvious that the quality of the approximation in (3.23) is determined by the projector P. Next, we show how to obtain a good approximation by choosing the appropriate projector.

It is well known that if the subspace spanned by a matrix F is inside the range space of a projector P, F remains invariant after the projection by P, i.e., $F = PF$. In our case, in order to find a reduced system without accuracy lost, V, which spans the range space of P, should include all the subspaces spanned by $z(s)$. This results

in $\tilde{z}(s) = z(s)$ which looks like an excellent result. However, this "perfect" projector usually leads to a "reduced system" with the same size of the original system. Notice $z(s) \in \mathbb{C}^{n\times p}$ is a function of s and for a fixed value $s = s_i$, $z(s_i)$ spans a subspace of dimension p. For all values of s, the infinite number of p dimensional subspaces usually fill the whole n dimensional space. In order to include all these subspaces, V needs to have the dimension n leading to a $n \times n$ "reduced" system which has the same size as the original system and is unacceptable for model reduction.

Although it is usually impossible to generate a reduced system accurate for all values of s as discussed above, it is possible to generate a reduced system accurate for certain values of s. If we have generated $V \in \mathbb{C}^{n\times mp}$ to include m subspaces spanned by m values of s, then the resulting $\tilde{z}(s)$ matches $z(s)$ at all these values of s. Since $V \in \mathbb{C}^{n\times mp}$, the reduced system has an order of mp. In order to form this special V matrix, we have to sample the original system at the m values of s using (3.14) and the V matrix has the following property:

$$\text{span}(V) = \text{span}(z(s_1), z(s_2), \ldots, z(s_m)). \tag{3.24}$$

We summarize the discussions above in the following theory:

Theorem 1. *If s_k is sampled , then $\tilde{z}(s_k) = z(s_k)$ for $k = 1, 2, ..., m$, where m is the number of sample points.*

Proof. According to the definition of projection, P, which is a projector onto V, is the identity operator on the space spanned by V, that is, $\forall x \in V : Px = x$. According to (3.24), because $z(s_k) \in \text{span}(z(s_1), z(s_2), \ldots, z(s_m)) = \text{span}(V)$,

$$\tilde{z}(s_k) = Pz(s_k) = z(s_k),\ k = 1, 2, ..., m.$$

□

Accuracy of the imaginary axis sampling

The previous subsection shows if we sample the original system at m values of s and form the V matrix, the approximated impulse response matrix $\tilde{z}(s)$ equals to the original $z(s)$ at these sampled points. In this section, we discuss the sample point should be limited to the imaginary axis to capture the frequency response of the original system.

Assume (3.13) is a stable causal linear system, the system poles are on the left hand side of the complex plane. On the imaginary axis $s = j\omega$, the Laplace transform converges and is equivalent to the fourier transform. The frequency response of the system is

$$H(j\omega) = L(G + j\omega C)^{-1}B, \tag{3.25}$$

and the impulse response of the system is the inverse fourier transform of the frequency response

$$h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(j\omega)e^{j\omega t}\, d\omega. \tag{3.26}$$

In order to have the same frequency behavior as the original system at a frequency point ω_k, the reduced system needs to have the same frequency response at both $-j\omega_k$ and $j\omega_k$. It is important because for the original system, $H(j\omega)$ is conjugate symmetric, i.e. $H(-j\omega) = H^*(j\omega)$. As a result, for the time domain waveforms, for example the impulse response (3.26), the contributions by the $\omega \in (-\infty, 0)$ (i.e., $\frac{1}{2\pi}\int_{-\infty}^{0} H(j\omega)e^{j\omega t}\, d\omega$) and $\omega \in (0, \infty)$ (i.e., $\frac{1}{2\pi}\int_{0}^{\infty} H(j\omega)e^{j\omega t}\, d\omega$) are conjugate to each other such that their imaginary parts are eliminated leading to the real time domain waveform. Failing to retain this symmetric property in the reduced system is unacceptable. In addition, it is even impossible to generate a real system if the

sampling is not performed symmetrically on the imaginary axis.

We introduce V_c to denote the V matrix in the imaginary sampling case. According to (3.24), the V_c matrix of the imaginary sampling is generated as

$$\text{span}(V_c) = \text{span}(z(j\omega_1), z(j\omega_2), \ldots, z(j\omega_m), z(-j\omega_1), z(-j\omega_2), \ldots, z(-j\omega_m)), \tag{3.27}$$

for the frequency points $\omega_1, \omega_2, \ldots, \omega_m$ we would like to match exactly. The practical way of generating the real valued V_c matrix is presented later.

We are ready to present the following result as the extension of Theorem 1.

Corollary 3.2.1. *The reduced system has exact frequency response at the frequency ω_k if $j\omega_k$ and $-j\omega_k$ on the imaginary axis are sampled.*

Proof. Consider the reduced frequency response $\hat{H}(j\omega)$. If $j\omega_k$ and $-j\omega_k$ are sampled, according to Theorem 1, there are $\tilde{z}(j\omega_k) = z(j\omega_k)$ and $\tilde{z}(-j\omega_k) = z(-j\omega_k)$, then we have

$$\hat{H}(j\omega_k) = L\tilde{z}(j\omega_k) = Lz(j\omega_k) = H(j\omega_k),$$

$$\hat{H}(-j\omega_k) = L\tilde{z}(-j\omega_k) = Lz(-j\omega_k) = H(-j\omega_k).$$

□

Accuracy of the real axis sampling

For real axis sampling, according to (3.24), the V matrix is

$$\text{span}(V) = \text{span}(z(\sigma_1), z(\sigma_2), \ldots, z(\sigma_m)). \tag{3.28}$$

Similar to the imaginary axis sampling case, we have $\tilde{z}(\sigma_k) = z(\sigma_k)$. However, there is only $H(\sigma_k) = \hat{H}(\sigma_k)$. Generally for all frequencies except for $\omega = 0$, there is

no $H(j\omega) = \hat{H}(j\omega)$ guaranteed. In other words, the real axis sampling does not guarantee to be accurate for any frequency except for the DC point.

Sampling along the imaginary axis

Previously, we have shown by sampling along the imaginary axis, the frequency response of the reduced system matches that of the original system exactly. In this section, we present how the imaginary sampling is performed in a practical way.

Specifically, by performing the sampling along the imaginary axis, Z, as defined in (3.16), is a complex matrix. As explained previously, for every sampling at $j\omega_k$, we also need to sample $-j\omega_k$. Actually, it is not necessary to perform the sampling at both points. Due to the conjugate property, if we have sampled $j\omega_k$, $k = 1, 2, \ldots, m$ and generated the corresponding Z, we only need to introduce Z^* as the conjugate of Z and form a new complex subspace $Z_c = [Z\ Z^*]$. Also assume W is an identity matrix without loss of generality, we have new versions of (3.15) and (3.17) as

$$\hat{X}_c = Z_c Z_c^H \tag{3.29}$$

and

$$Z_c = V_c S_c U_c. \tag{3.30}$$

As a result, there is

$$\hat{X}_c = V_c S_c U_c U_c^H S_c V_c^H = V_c S_c^2 V_c^H, \tag{3.31}$$

which is the eigendecomposition of the approximated Gramian $\hat{X}_c$. In contrast to the complex $\hat{X}$ in (3.15), $\hat{X}_c$ here is a symmetric real matrix even though Z_c is complex. According to the properties of the eigendecomposition, V_c should be a real unitary matrix as it is the eigenspace of a symmetric real matrix, i.e. $V_c^H = V_c^T$, which will

ALGORITHM 1: CVSTBR algorithm

Input: Circuit: G, C, B, L; sample points: ω_k, $k = 1, 2, ..., m$; the reduced order: q.
Output: Reduced system matrices: $\hat{G}$, $\hat{C}$, $\hat{B}$, $\hat{L}$.

1. Solve $z_k = (G + j\omega_k C)^{-1} B$ for $k = 1, 2, ..., m$.
2. Combine all the solved z_k to form Z.
3. Construct the complex Gramian subspace $Z_c = [Z, Z^*]$.
4. Perform economic SVD on Z_c and obtain the left singular matrix V_c and singular value matrix S. Keep only q dominant columns of V_c.
5. Rotate V_c to the real axis through multiplying the kth column of V_c with $exp(-j\phi_k)$, where ϕ_k is the phase of the kth column in the complex plane.
6. Build the reduced model $\hat{G}$, $\hat{C}$, $\hat{B}$, $\hat{L}$ using the projection matrix V_c and V_c^T.

generate the real reduced systems even if we sample along the imaginary axis.

Note that we do not perform the eigendecomposition to compute V_c, instead, we use SVD on Z_c in order to save the computation time. However, V_c is still not a real matrix in general. This is because the singular vectors of the SVD are not unique in the complex plane, although the singular values are uniquely computed [63]. For instance, for SVD of matrix $\mathcal{A} \in \mathbb{C}^{n \times n}$, $\mathcal{A} = VSU^T = \sum_{i=1}^{n} \sigma_i v_i u_i^T = \sum_{i=1}^{n} \sigma_i (v_i e^{j\phi})(u_i^T e^{-j\phi})$. In other words, singular vector v_i and u_j can rotate with the same angle but in the opposite directions without changing the subspace. Specifically, for the kth column v_k in V_c, multiplying with $exp(-j\phi_k)$ will rotate it to the real axis. The phase ϕ_k is the angle between v_k and the real axis.

The new complex-valued sampling based reduction method, named CVSTBR, is summarized in Algorithm 1.

3.2.3 Residual based error estimator and its relationship with imaginary axis sampling

We have introduced the new imaginary axis sampling scheme CVSTBR which matches the frequency response of the original system at the sampled frequencies. However,

CVSTBR works in a static way with provided sample frequencies and an adaptive sample point selecting scheme is demanded. In order to perform the adaptive sampling, we need an error indicator to guide the adaptive sample point picking process. Calculating the errors explicitly requires expensive solving of the original model at many points, which should be avoided if possible. In this subsection, we present a good error estimator based on the residual of the reduced system. Combined with the imaginary sampling scheme, the residual error estimator is zero at the sampled frequencies and is good for guiding the adaptive sampling process.

Remind in (3.23) we have the reduced state approximation $\tilde{z}(s) = Vz_r(s) \approx z$. Plugging it into the original system with identity input

$$Gz(s) + sCz(s) = B \tag{3.32}$$

leads to the error matrix, which is called the *residual* matrix in this paper

$$\begin{aligned} R(s) &= GVz_r(s) + sCVz_r(s) - B \\ &= G\tilde{z}(s) + sC\tilde{z}(s) - B, \end{aligned} \tag{3.33}$$

where $R(s) \in \mathbb{C}^{n\times p}$. Notice that if $\tilde{z}(s) = Vz_r(s)$ is very close to $z(s)$, the residual should be very small. As a result, the norm of $R(s)$, $||R(s)||$ can serve as a good error indicator for the reduced model.

We remark that $R(s)$ is not a dimensionless quantity for each element in it. Each element in it actually represents either node voltage or branch current residual. And each column of $R(s)$ can be viewed as normalized error at all the nodes/branches of the system (3.18) excited by the impulse response applied at one port. As a result, the residual matrix $R(s)$ can be used as the input-normalized error indicator.

From Theorem 1, we have the following corollary:

Corollary 3.2.2. *The reduced system has zero residual at the sample points in the complex plane.*

Proof. If s_k is sampled, from Theorem 1, there is $\tilde{z}(s_k) = z(s_k)$. By plugging it into (3.33), we have

$$\begin{aligned} R(s_k) &= G\tilde{z}(s_k) + s_k C\tilde{z}(s_k) - B \\ &= Gz(s_k) + s_k Cz(s_k) - B \\ &= 0. \end{aligned}$$

□

Furthermore, we would like to show how the residual is related to the imaginary sampling. If the sampling is performed on the imaginary axis, the residual in (3.33) becomes

$$\begin{aligned} R(j\omega) &= GV_c z_r(j\omega) + j\omega CV_c z_r(j\omega) - B \\ &= G\tilde{z}(j\omega) + j\omega C\tilde{z}(j\omega) - B, \end{aligned} \tag{3.34}$$

and there is a corollary similar to Corollary 3.2.2.

Corollary 3.2.3. *The reduced system residual is zero at the sampled frequency points if the sampling is performed on the imaginary axis.*

Proof. If we have sampled the frequency ω_k, there are $\tilde{z}(j\omega_k) = z(j\omega_k)$ and $\tilde{z}(-j\omega_k) =$

$z(-j\omega_k)$. As a special case of sampling in the complex plane, we have

$$\begin{aligned} R(j\omega_k) &= G\tilde{z}(j\omega_k) + j\omega_k C\tilde{z}(j\omega_k) - B \\ &= Gz(j\omega_k) + j\omega_k Cz(j\omega_k) - B \\ &= 0, \\ R(-j\omega_k) &= G\tilde{z}(-j\omega_k) - j\omega_k C\tilde{z}(-j\omega_k) - B \\ &= Gz(-j\omega_k) - j\omega_k Cz(-j\omega_k) - B \\ &= 0. \end{aligned}$$

□

Besides showing small values around the sampled frequencies, the residual should also reveal the errors at the unsampled frequencies to guide the adaptive sampling process. Now, we discuss why sampling using residual as the error estimator leads to good results. Assume we have sampled m points $\{\omega_1, \omega_2, \ldots, \omega_m\}$ of the n dimensional system, the resulting reduced model has the order $2mp$. We have already shown at the sampled frequency point, for example ω_k, the approximate impulse response $\tilde{z}(\pm j\omega_k)$ equals to the original impulse response $z(\pm j\omega_k)$ such that the reduced system has exact frequency response as the original system. Consider a frequency far away from the sampled frequencies, for example at ω_l, the approximate impulse response $\tilde{z}(\pm j\omega_l)$ lays inside the subspace $\text{span}(V_c) = \text{span}(z(\pm j\omega_1), z(\pm j\omega_2), \ldots, z(\pm j\omega_m))$ which is a $2mp$ dimensional subspace inside the n dimensional space. Since ω_l is not sampled, the original impulse response $z(\pm j\omega_l)$ generally dose not equal to $\tilde{z}(\pm j\omega_l)$ in this case. It can be decomposed into two components, one equals to $\tilde{z}(\pm j\omega_l)$ inside $\text{span}(V_c)$, the other one equals to $(z(\pm j\omega_l) - \tilde{z}(\pm j\omega_l))$ inside the $n - 2mp$ dimensional left null space of V_c. The component lays inside the left nullspace causes the error of the reduced

system and is revealed through the residual as $R(\pm j\omega_l) = (G \pm j\omega_l C)(z(\pm j\omega_l) - \tilde{z}(\pm j\omega_l))$. If $||R(\pm j\omega_l)||$ is a local peak, it means $||z(\pm j\omega_l) - \tilde{z}(\pm j\omega_l)||$ is relatively large. In this case, we need to sample the frequency ω_l to make the new subspace span(V_c) include the component of $z(\pm j\omega_l)$ inside the left null space of the previous V_c.

3.2.4 Adaptive sample point placement

Now we demonstrate how to place the sample points along the imaginary axis in an adaptive way with the residual based error estimation.

Before presenting the new method, we show some theoretical results regarding to the residual errors and the complex-valued sampling.

Theorem 2. *Given sufficient sample points in imaginary axis, the residual function defined in* (3.34) *will become monotone decreasing.*

Proof. Based on the Corollary 3.2.1, it is known that at the sample points, the responses of the reduced model match exactly with those of the original model. Their residuals are all zero. The sampling approach can be viewed as the multi-point complex Krylov subspace method where only the zero-th moment is matched at every frequency point. For any frequency point $j\omega_k$, the reduced state response in (3.18) can be written in the moment form as

$$z_r(j\omega_k + \delta) = (I + \delta M_k + \delta^2 M_k^2 + ...)\mathbf{r}_k, \tag{3.35}$$

where $M_k = -(\hat{G} + j\omega_k \hat{C})^{-1}\hat{C}$ and $\mathbf{r}_k = (\hat{G} + j\omega_k \hat{C})^{-1}\hat{B}$, $\delta \in \mathbb{C}$ is a small complex value change from $j\omega_k$.

If we sample sufficiently, δ, which can be viewed as the half distance between two

adjacent points, will be small enough. As a result, the error between $z_r(j\omega_k)$ and $z_r(j\omega_k + \delta)$ can be approximated as $\delta M_k \mathbf{r}_k$. So the error will go down monotonically with δ when δ is small enough.

Now we look at the residual $R(j\omega_k+\delta)$. Notice that we have $R(j\omega_k) = 0$ according to Corollary 3.2.3. If we ignore the second order terms, we have

$$\begin{aligned} R(j\omega_k + \delta) &= (G + (j\omega_k + \delta)C)V_c z_r(j\omega_k + \delta) - B \\ &= R(j\omega_k) + \delta(GV_c M_k + j\omega_k C V_c M_k + CV_c)\mathbf{r}_k \\ &= \delta(GV_c M_k + j\omega_k C V_c M_k + CV_c)\mathbf{r}_k. \end{aligned}$$

As a result, the residual $R(j\omega_k+\delta)$ will decrease monotonically with δ when δ is small enough. □

Theorem 2 states if a point is getting close enough to a sampled point, its residual error becomes controllable. As a result, the proposed method can always achieve the error bound by sufficient samplings. But practically, it is not necessary to sample densely over all frequencies. Instead, we develop a new adaptive sampling algorithm in which sample points are added dynamically to frequencies having more poles near the imaginary axis and thus more likely to show large residuals.

The new algorithm as demonstrated in Fig. 3.2 has two iterative steps. At step (a) in Fig. 3.2, WBMOR first builds a reduced model using the minimum sample points picked in the previous iterations. Then it tests all the candidate points which are initially evenly distributed in the frequency range (in log scale). Residuals are computed at all the candidate points (they needs to be computed or evaluated only once). At step (b), WBMOR drops all the satisfied candidate points whose residuals are smaller than the threshold from the candidate set. And for each unsatisfied point, two of its adjacent middle points are inserted to the candidate set. The step of

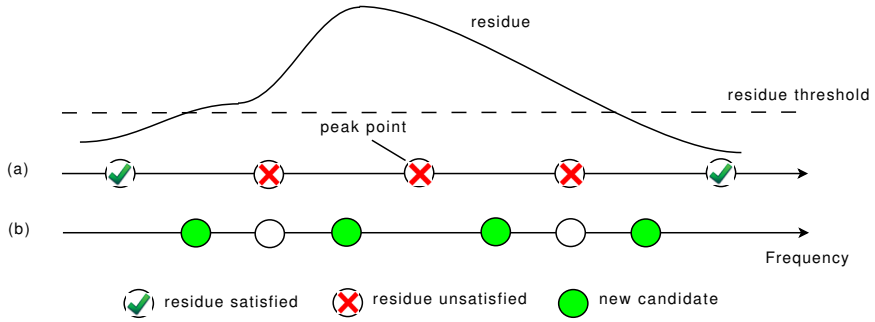


Figure 3.2: An illustrative example for the WBMOR adaptive sampling scheme.

inserting more samples is optional if the initial candidate points are sufficiently dense. At the same time, WBMOR takes the *peak points*, which are the unsatisfied candidate points and are local maximums in residual values, as the sample points. The peak points are dropped from the candidate set as they have guaranteed zero residuals in the next iteration. The whole process continues until there are no unsatisfied candidates left.

For most of the time, there are several peak points in one iteration. Thus the number of sample points selected in one iteration is not limited to one. This greatly enhances the convergence speed. Moreover, the peak points are usually not close to each other, which means they are relatively independent. This enables us to sample them all at the same time without causing over sampling problems.

The dynamic sample insertion in step (b) guarantees WBMOR can generate good model even when the initial candidate points are sparsely or badly distributed. This is important because sometimes there is too little information about the original model to guide the placement of the initial candidate points. Although the dynamic sample insertion in step (b) can be enabled to avoid missing the potential range with large errors, we notice that if the initial candidates are placed dense enough (around 100 points per decade is enough in our experiments), WBMOR will still reach very good results with cheap computational cost without dynamic sample insertion.

In order to achieve a more efficient reduction, after the complex SVD on Z_c in (3.30), we select the dominant singular values and corresponding singular vectors in V_c based on a user set SVD threshold th_{svd}. th_{svd} is defined as the threshold for the weight of the trivial singular values.

3.2.5 WBMOR algorithm flow and analysis

The whole WBMOR flow is shown in Algorithm 2.

The further truncation based on the singular values is performed after the iteration. As a result, SVD is only needed in the last iteration, and we use QR factorization instead of SVD in the loop to accelerate the algorithm. In addition, incremental QR algorithms, which only orthonormalize the newly sampled columns, can also be used to further save the CPU time.

Now we take a look at the computational cost of WBMOR. Most CPU time is spent on the residual matrix computation (3.34), since it is required for each candidate point in every iteration. GV_c and CV_c in (3.34) is calculated only once in an iteration because they are the same for every candidate and only vary over iterations. Also, $z_r(s)$ can be cheaply solved from the reduced system at each candidate. Thus, the cost of (3.34) at one candidate point is mainly the multiplication cost of $GV_c \cdot z_r$ and $CV_c \cdot z_r$, which is $O(npq)$ where n is the size of the original model, p is the port number and q is the reduced model size at the current iteration. The cost for each candidate increases as the reduced model size q increases during the iterations. However, the number of candidate points will decrease as all the candidates smaller than the threshold are dropped. Thus, the later iteration will cost less as the algorithm proceeds. Moreover, WBMOR takes peak candidate points as sample points, whose number is usually larger than one. This greatly improved the convergence speed and

ALGORITHM 2: WBMOR algorithm

Input: Circuit: G, C, B, L; frequency range: ω_{min}, ω_{max}; the residue threshold: th_{res} and the SVD threshold: th_{svd}.

Output: Reduced system matrices: $\hat{G}$, $\hat{C}$, $\hat{B}$, $\hat{L}$.

1. Place initial candidate points at a reasonable density in the given frequency range $[\omega_{min}, \omega_{max}]$ evenly in log scale.
2. Obtain the initial reduced model by sampling at ω_{min} and ω_{max} using the *CVSTBR* algorithm.
3. While max residue $> th_{res}$, do
4. Calculate the residues at the candidate points using (3.34).
5. Scan all the candidate points. If residue norm $< th_{res}$, drop this point; otherwise (optional), add two middle points between the current point and two adjacent points as new candidate points.
6. Add the candidates which are peak points with excessive residues as the sample points. Use the *CVSTBR* algorithm to compute the new reduced model with all the selected sample points.
7. Take the singular subspace V_c from the *CVSTBR* in the last iteration. Only keep its dominant singular vectors according to th_{svd}.
8. Build the real reduced model $\hat{G}$, $\hat{C}$, $\hat{B}$, $\hat{L}$ using the projection matrix V_c and V_c^T.

makes WBMOR usually stop in less than 10 iterations. In summary, although more expensive than the static sampling based methods, WBMOR still has a relatively low computational cost considering its high accuracy.

3.3 Experimental Results

3.3.1 Implementation and settings

The proposed method WBMOR together with the re-sampling method [59] and recently proposed adaptive sampling based reduction method method, ARMS [65], as well as other mentioned sampling methods have been implemented in Matlab 7.0. All the experimental results are collected on a Linux workstation with Intel Quadcore Xeon CPU with 2.99Ghz and 16GB memory. The WBMOR method has been integrated into the *UC Riverside Model Order Reduction Tool Suite* (UiMOR) [62] which

is available for download online [2].

For the proposed WBMOR method, the default setting is as follows: the residual threshold is chosen to be 0.1 and the number of initial points per decade is 100 and we only drop candidate points without adding new ones. The SVD threshold th_{svd} is 10^{-7}.

For the re-sampling method, the number of points in the pool is 20, the number of reduced models used in each iteration is 10, the size of the reduced model is 20. We also used 20 search points, 1/3 of which will be replaced. One point in the pool will be substituted in every iteration. We also implemented the speedup techniques such as efficient construction of projectors and heuristic search [59].

For the ARMS method, however, many implementation details were not given in [65]. In our implementation and comparison, it shares the same initial sampling candidate set with WBMOR. We apply QR factorization to do the orthogonalization to generate the residuals. One may argue that the incremental QR algorithm is more efficient. However, incremental QR needs to store the large and dense orthogonalized basis (the Q matrix in QR factorization) for every candidate point, and the resulting algorithm will soon become memory-limited.

Since our comparison includes both statical sampling methods and adaptive methods with different stop criteria, for a fair comparison, we have to keep the same or similar size of the final reduced order for all methods for accuracy comparison.

The benchmark circuits are published benchmarks such as the transmission line (TL) model, the PEEC model from [12], which are available online for download. Some additional RLC circuits are also used for further accuracy and scalability comparisons.

We first show that the results of the static complex-valued sampling based reduction method on the transmission line model example. Then, we show the results of

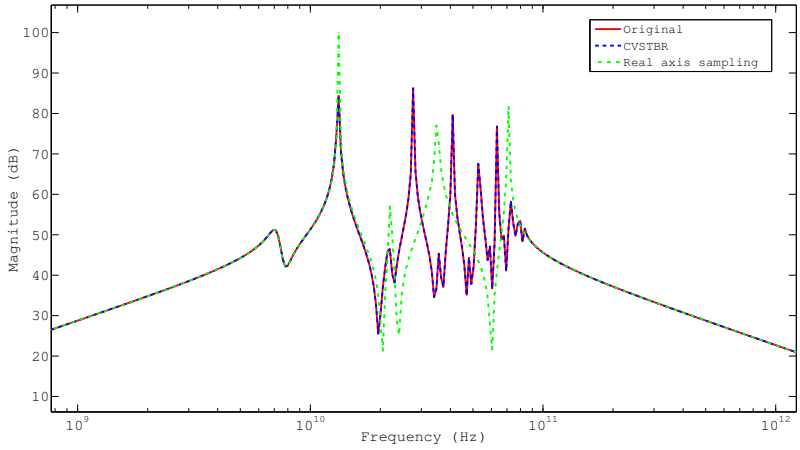


Figure 3.3: Accuracy comparison of imaginary axis sampling and real axis sampling methods on TL.

the adaptive WBMOR method on the TL, PEEC models. Finally, the runtime and maximum error of each method are evaluated using more RLC circuits.

3.3.2 Comparison of complex-valued sampling scheme and real-valued sampling scheme

First, we would like to show the accuracy of CVSTBR. Note that in order to generate the same dimension of Z_c, the real sampling method has to use twice the sample points. In the example, the real sampling method uses 100 sample points that are uniformly distributed in the given frequency range while the CVSTBR picks only half of these points, that is 50 points. After SVD, both of the two methods keep 50 dominant columns of the left singular matrix V_c as the projection matrix, and thus the size of the two final reduced models is 50. All the samples are generated randomly so that the real-valued sampling method is similar to the proposed PMTBR method [55]. Fig. 3.3 shows the results of the comparison. It is clear that the model generated by the CVSTBR reaches a frequency response that cannot be distinguished from the original one while the real sampling method has very large errors.

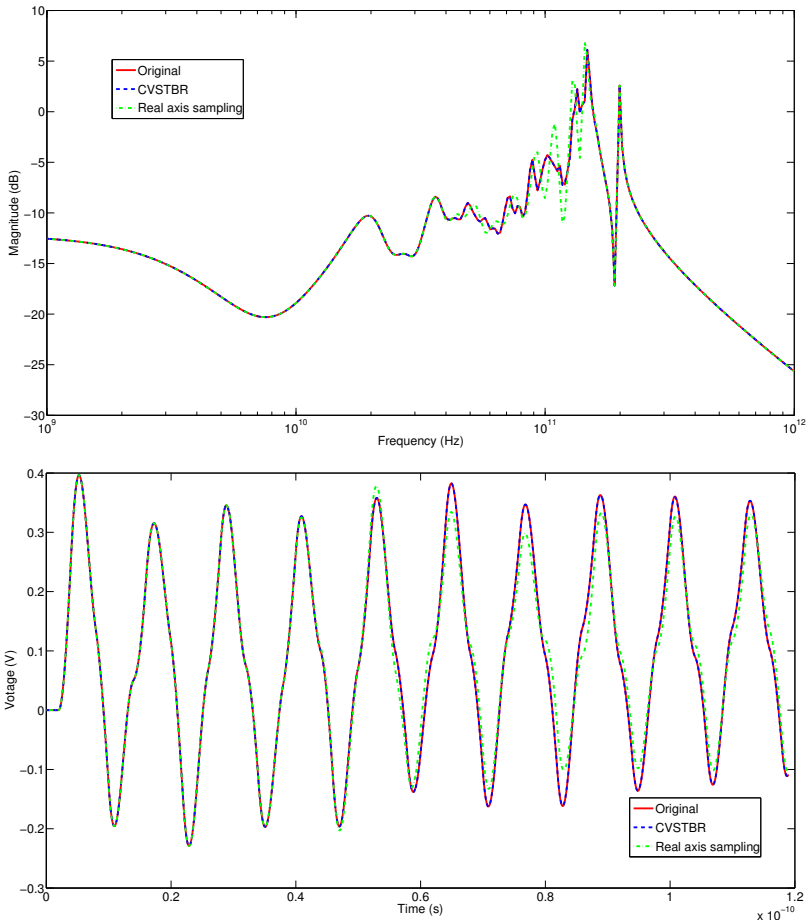


Figure 3.4: Frequency response (top) and transient simulation (bottom) comparison of imaginary axis sampling and real axis sampling methods on rlc1.

For the computational cost, since both schemes generate the same dimension of the Z_c matrix, they have the same SVD cost of Z_c. However, the real axis sampling scheme samples twice more than the CVSTBR to reach the same size of Z_c, and thus, it is *more expensive*. To make the computational cost similar, the real axis sampling method will sample half of the points (both methods sample 50 points). Obviously, it will become further inaccurate. As a result, we can clearly observe that *given the same computing cost*, the complex-valued method CVSTBR is more accurate than the real axis sampling method like PMTBR [55]. Actually, this is the case for all other benchmarks we test for the reasonable number of samples.

Detailed analysis shows that instead of 50 sample points, only 13 points are enough to produce the 50 states reduced model in the ideal case (recall that every point generates 4 states in the final reduced model without the SVD process). This ideal case happens when we sample at only the critical points such that in the SVD process, all the singular vectors are important and cannot be truncated. In the next section, we show how we insert new samples around those critical regions more intelligently in the WBMOR algorithm.

A better accuracy comparison is in the time domain for the reduced models. We test on a RLC circuit *rlc1*, which has a 640 states and 1 port. It has many peaks around 10^{11}Hz and is very hard to approximate. 100 sample points are used for CVSTBR while 200 points for the real axis sampling method. The frequency range of interest is chosen to be $[10^9, 10^{12}]$, which covers the whole spiky area. The final size for both reduced models is kept manually as 100. A periodic 1Amp pulse current with 2×10^{-12}s rise time and 4×10^{-12}s hold time is used as the input signal at the port. The output voltage is measured from the same port. The frequency response and transient simulation results are shown in Fig. 3.4. It can be seen from the time domain simulation, for the model generated by real-value sampling, there are obvious discrepancy at some peak and valley values in the transient response wave.

3.3.3 Adaptive process and comparison with the re-sampling scheme

We now compare the new adaptive WBMOR method with several methods based on real axis sampling, including the recently proposed re-sampling method [59], the simple logarithmic and Monte Carlo sampling methods. In order to reach a fair comparison, we set all of the four methods with the same dimension of Z_c and the

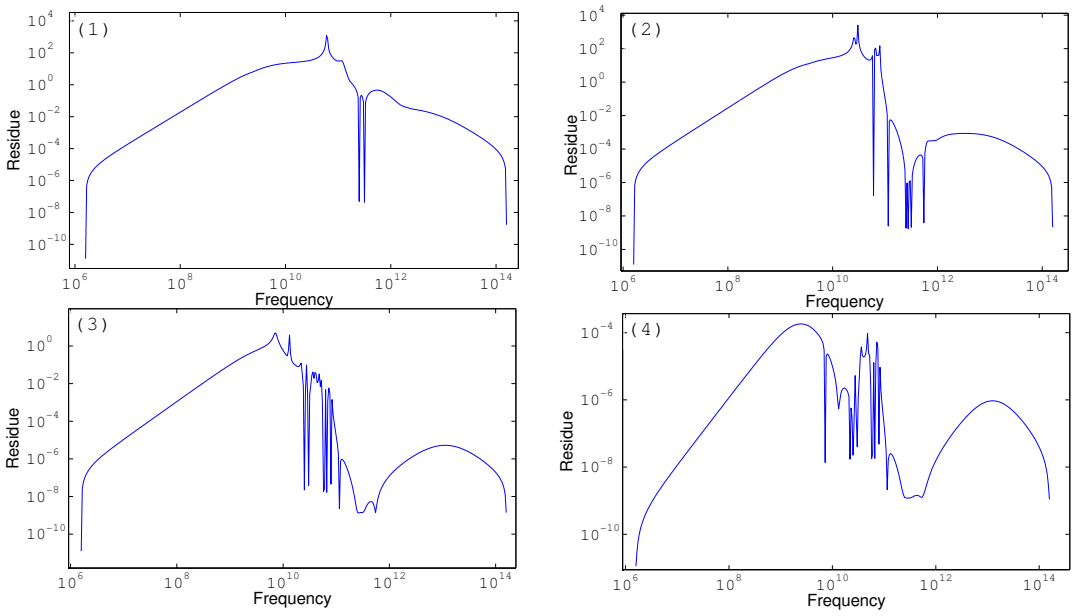


Figure 3.5: The residual convergence process of WBMOR for four iterations.

same size of final reduced system after SVD.

The first example is the transmission line model used in 3.3.2. In this case, WBMOR uses only 14 sample points and generates a reduced model of dimension 37. Fig. 3.5 shows the residual convergence process of our WBMOR during four iterations. The maximum residual of the reduced system drops very fast from the value as large as 10^4 to below 10^{-3} after four iterations. From Fig. 3.6, it is clear that the new WBMOR method produces a reduced model as accurate as the the static CVSTBR (Fig. 3.3) with much less sample points and more compact size. Even with twice the number of samples and the same size of the final reduced model, none of them captures the fine details of the original system frequency response. Among these three real sampling methods, re-sampling method has the best results, which however gradually become less accurate after the third large peak.

Fig. 3.7 shows the results of the widely used PEEC model. With 42 sampling points, WBMOR shows good results in most of the frequency bands and only a little

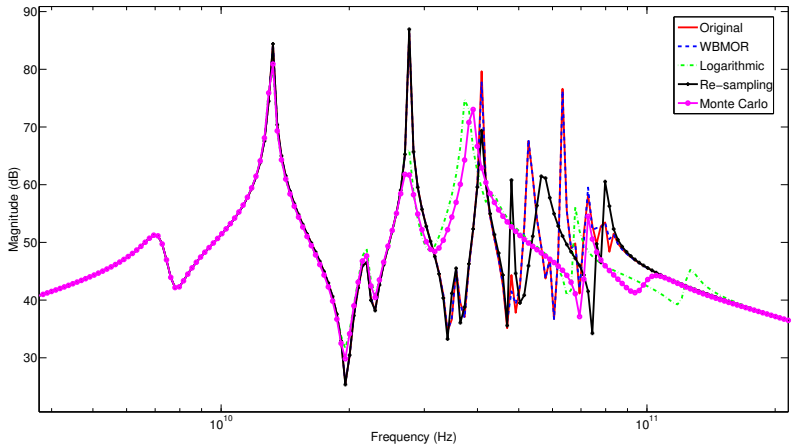


Figure 3.6: Comparison with re-sampling method on the transmission line example.

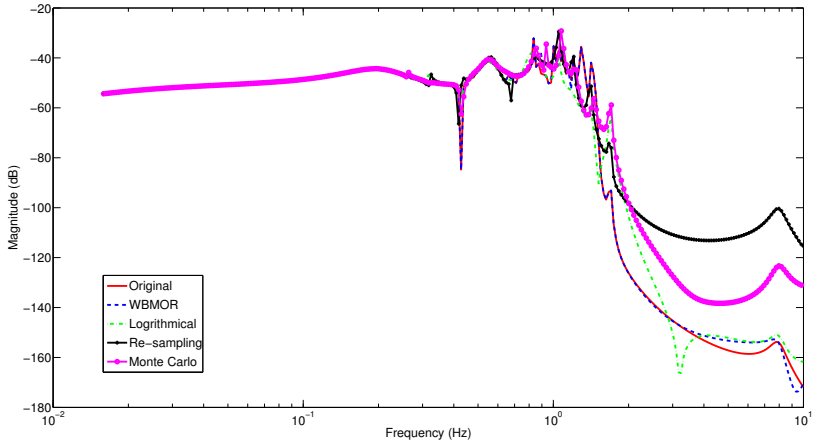


Figure 3.7: Comparison with re-sampling method on the PEEC example.

off around $10Hz$. This is because the frequency responses at these frequencies are quite small (around -160dB) and are very hard to match due to the numerical errors.

3.3.4 Comparison with the ARMS method

The results compared with recently proposed adaptive sampling method ARMS is shown in this subsection. Although the ARMS's implementation is based on real

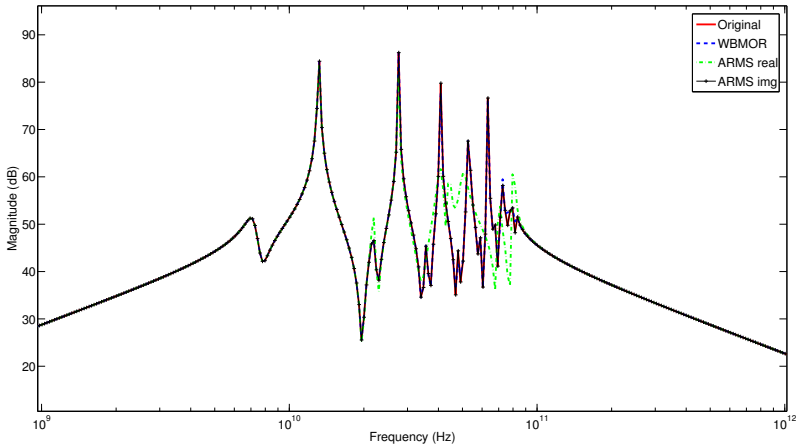


Figure 3.8: Comparison with ARMS on the TL example

valued sampling, we modify it by using the proposed imaginary axis sampling. In this way, we can see more clearly the performance of the two methods for their adaptive schemes. Specifically, in order to perform complex-valued reduction, an additional conjugate column should be put into the Gramian matrix Z in ARMS. In addition, SVD is used in the last iteration in order to generate the real valued system.

The results on the TL model is shown in Fig 3.8. The order of the reduced models is 37 except for the imaginary sampling based ARMS, which has a order of 36. Obviously, the real sampling based ARMS method failed to find the critical samples while the imaginary axis sampling based ARMS does a relatively better job while it still lacks some accuracy at certain frequency bands.

Also, the comparison on the PEEC model is presented in Fig 3.9.

Until now, all the results of WBMOR are obtained by only dropping the candidates during iterations. In other words, we do not add any new candidate points. This generates good results as shown in the previous experiments. However, sometimes setting the initial set of candidate points become a tricky problem. We may start with a sparse candidate set and result in a poor approximation since the initial set

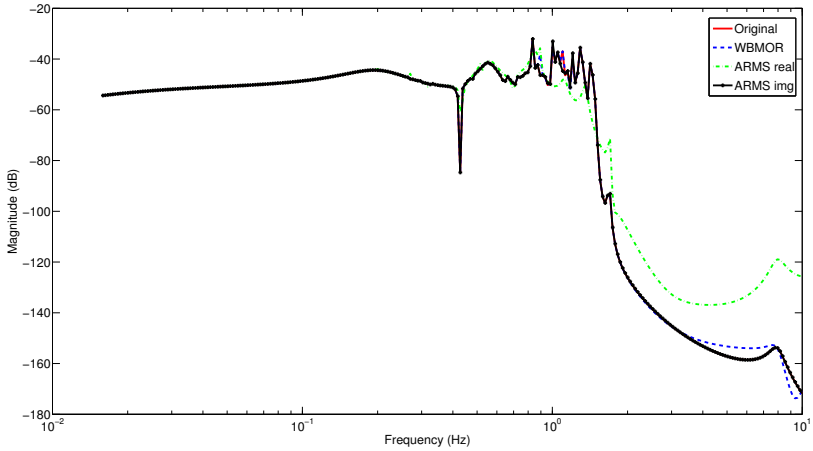


Figure 3.9: Comparison with ARMS on the PEEC example

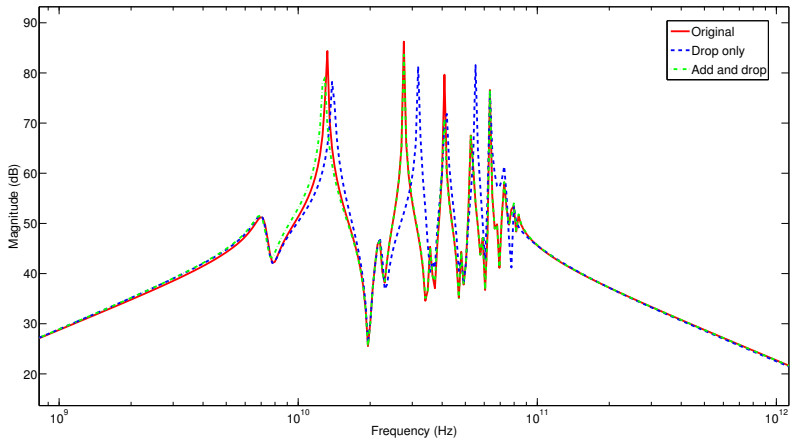


Figure 3.10: Demonstrate of the adding candidates feature of WBMOR

may fail to cover some critical regions with large errors. This can be avoided by inserting candidate points in a dynamic and adaptive way. Take the TL model again for example. Now we start with only 10 candidates per decade instead of 100 used previously. Fig. 3.10 shows the results. By detailed inspection, it is clear that most of the important potential samples, which would have been missed, are correctly located through the dynamic point insertion process. The resulting WBMOR successfully reaches a good approximation even with the sparse initial candidate set.

1	2	3	4	5	6	7	8	9	10	11	12
Ckt	Node	Port	Order	WBMOR		Re-sampling		Monte Carlo		Logarithmical	
				Time(s)	Max error	Time(s)	Max error	Time(s)	Max error	Time(s)	Max error
TL	256	2	37	0.9	0.3	1.4	21.6	0.09	8.5	0.09	17.7
PEEC	480	1	79	1.9	1.5	2.4	2000.1	0.71	33.1	0.70	32.9
rlc1	640	1	62	0.7	0.1	1.2	1.1	0.14	1.7	0.12	2.08
rlc2	1180	2	160	4.8	0.1	4.4	0.5	0.36	0.9	0.34	1.6
rlc3	2680	3	192	12.1	0.1	8.8	1.2	0.8	2.1	0.8	1.5
rlc4	10960	1	58	11.2	0.2	6.3	1.0	2.7	2.8	2.7	2.1

Table 3.1: Scalability comparison of runtime and relative errors for WBMOR and the re-sampling method.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ckt	Node	Port	WBMOR				ARMS real				ARMS img			
			order	iter	Time(s)	Max error	order	iter	Time(s)	Max error	order	iter	Time(s)	Max error
TL	256	2	37	4	0.9	0.3	38	19	5.3	11.6	40	10	10.7	2.8
PEEC	480	1	79	10	1.9	1.5	79	79	46.7	81.0	80	40	56.9	1.7
rlc1	640	1	62	11	0.7	0.1	62	62	135	1.2	62	31	126	0.1
rlc2	1180	2	160	9	4.8	0.1	160	80	1099	0.5	160	40	2402	0.1
rlc3	2680	3	192	7	12.1	0.1	192	64	2918	0.4	192	32	3705	0.05
rlc4	10960	1	58	9	11.2	0.2	58	58	2737	1.2	58	29	3273	0.2

Table 3.2: Scalability comparison of runtime and relative errors for WBMOR and the ARMS method.

3.3.5 CPU runtime and error comparison

Finally, we report the runtime and the maximum relative errors of the examples in 3.3.3 together with several additional RLC circuits in Table 3.1. The comparison with ARMS method is presented in Table 3.2. The relative errors used in two tables are computed as

$$error(j\omega) = \frac{||H(j\omega) - \hat{H}(j\omega)||}{||H(j\omega)||}$$

at all the simulation frequency points.

We observe that the proposed method is much more accurate than all the real axis sampling based methods. The modified imaginary axis based ARMS has almost the same accuracy as WBMOR. We notice that ARMS computing costs are much higher

than the other two methods. The main computing costs are the QR operations. We need to do this for every remaining candidate point in each iteration, which is expensive for large candidate pool. One can argue that incremental QR will be better off in this case. We actually implemented incremental QR in ARMS. But we need to store all the previous Q_i and R_i for each remaining candidate points. The sizes of Q_i and R_i grow during iterations and their numbers are large because of the large number of initial candidate points (100 in our implementation). As a result, we soon run out of memory for some testing cases.

3.4 Summary

In this chapter, we have presented a novel model order reduction method, WBMOR, for wide frequency band model reduction. WBMOR explicitly computes the exact residual errors to guide the sampling process in an adaptive way. It has been shown that by sampling along the imaginary axis and performing a new complex-valued sampling based reduction, the reduced model will match exactly with the original model at the sample points. Theoretically, the proposed method can achieve the error bound over a given frequency range with sufficient sampling. Practically, we have designed an adaptive scheme to help designers choose the best order of the reduced model for the given frequency range and error bound. We have compared several sampling schemes such as Monte Carlo, logarithmic, and recently proposed re-sampling methods. Experimental results on a number of RLC circuits show that WBMOR is much more efficient than all the other sampling methods including the recently proposed re-sampling and ARMS schemes with the same reduction orders. Compared with the real-valued sampling methods, the complex-valued sampling method is more accurate for the same computational costs.

Chapter 4

Composable Thermal Modeling of Multi-Core Microprocessors

In this chapter, we address the emerging problem introduced above by proposing a novel composable thermal modeling approach. We present the new approach in the context of fast thermal analysis and design for multi-core microprocessors at the architecture level. The new approach, called *ThermComp*, which stands for thermal modeling with composable modules, builds the compact thermal models for the basic modules (CPU core and cache) from detailed thermal models generated by the finite difference method. Then, it applies the sampling-based reduction technique to reduce the complexity of those models. To make the complexity reduction efficient for thermal models with many ports, port reduction by means of adjacent port merging has been introduced. Such port reduction naturally leads to a new two-grid discretization scheme where a uniform global coarse grid is used for all the boundary grids and a fine grid is used for the internal grids of each module. The coarse grid is also justified by the smooth thermal gradients at the boundary.

The compact and *composable* model-based simulation provides a viable solution to

this difficult problem. *Composable* is defined as the ability to build the basic module models, which are then used to assemble different large systems. We also call the assembling process as *composition*. In addition, the large system assembled from the basic modules is called the *composite model*. This strategy is similar to the model-based electronic circuit simulation method, such as SPICE, which no longer solves the basic Poisson's equations directly at the device level to obtain the voltage and current information. Similar to the SPICE device models of the CMOS and BJT transistors, we propose new composable thermal models, which can be re-used and easily connected to build various thermal circuits and systems.

The composable modeling has three main advantages over traditional thermal modeling: first, it enables the reuse of the compact components in the thermal modeling process, especially for the multi/many-core systems: only few compact models need to be built for very large system with multiple identical components; second, the change in the design architecture only requires recomposing the compact components instead of rebuilding the whole model; third, it makes compact modeling of extremely large system possible because directly applying model order reduction on the large system is impossible due to the memory and computational cost limitations.

Experimental results on a number of multi-core microprocessor architectures show the new approach can easily build accurate composite thermal models from compact models of different modules for the fast architecture thermal analysis and optimization. The compact models lead to orders of magnitude speedup over the standard finite difference method with marginal error.

The rest of this chapter is organized as follows: Section 4.1 introduces the thermal modeling problem we are trying to solve. Section 4.2 presents the new composable thermal modeling method. Section 4.3 shows the experimental results. Finally, the summary is provided in Section 4.4.

4.1 Thermal simulation and composable modeling problems

In this section, we briefly present the problems of the composable thermal modeling and simulation of the VLSI system. The compact modeling technique and its corresponding model complexity reduction problem are shown in Section 4.1.1 and Section 4.1.2, respectively. The boundary condition handlings are introduced next, in Section 4.1.3.

4.1.1 Simulation by compact thermal modeling of multi-core systems

The basics of thermal modeling and analysis techniques can be found in Section 2.2. A traditional thermal analysis method directly solves the discretized thermal system in (2.22). In order to capture the junction temperature of a multi-core chip, a fine grid needs to be used, resulting in a very large thermal system. In this case, directly solving the system can be very expensive or even prohibitive. A solution is to build a reduced system using the model reduction techniques. However, building a reduced model from a very large scale system is still expensive. Moreover, a slight change in the multi-core architecture requires redoing the whole finite difference discretization and rebuilding the reduced model from the very large discretized system. As a result, it will be unacceptable for the thermal design exploration for various multi-core architectures as the spatial discretization, reduction and simulation have to be done for every architecture during the optimization steps.

Instead of directly using (2.23) or its reduced model for thermal analysis, a more efficient composable thermal-model based approach is more desirable. As shown in

Fig. 4.1 (a), we first build two compact composable models for CPU core and cache modules through the finite difference method and model reduction. Then, using the two models, we can build the multi-core thermal system such as the quad-core system shown in Fig. 4.1 (b), the 16-core system shown in Fig. 4.1 (c) or other multi-core thermal systems. We assume the CPU cores are the same for simplicity, but our approach can be easily extended to different CPU cores and other functional blocks. Fig. 4.2 is the lateral structure view of a typical package for the multi-core system. Typically the heat generated at the die is conducted from the bottom to the heat spreader and then to the heat sink. In this paper, we do not directly consider those package structures. Instead, we capture the effects of those package structures on the die using proper thermal conditions. For brevity's sake, we assume the other sides of the die do not have heat exchange (adiabatic condition).

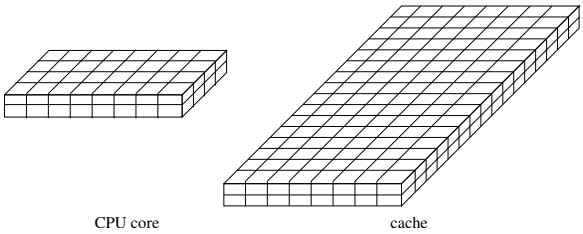
As discussed above, our goal is to build the compact composable thermal model for each module (CPU core and cache) so we can quickly build the composite models for different multi-core architectures instead of building the whole thermal systems from scratch every time.

4.1.2 Model complexity reduction problem

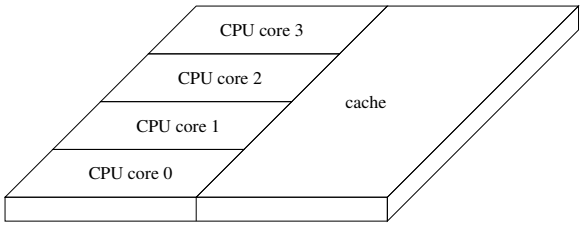
We already have detailed thermal models for the modules in the finite difference framework with certain boundary conditions. The modeling problem now is to build the compact thermal model for each module.

Specifically, if we have n discretized elements (grids) with specific boundary conditions, equation (2.23) becomes a linear ordinary differential equation

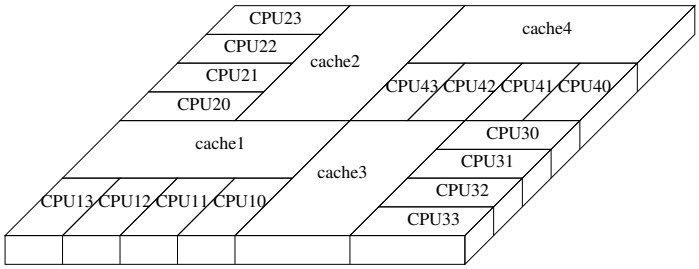
$$C\frac{dT(t)}{dt} + GT(t) = Bg(t), \tag{4.1}$$



(a) CPU core and cache modules.



(b) A quad-core architecture.



(c) A 16-core architecture.

Figure 4.1: CPU core, cache, a quad-core architecture and a 16-core architecture.

where $C \in \mathbb{R}^{n\times n}$ is the thermal capacitance matrix, $G \in \mathbb{R}^{n\times n}$ is the thermal conductance matrix. $B \in \mathbb{R}^{n\times p}$ is the position matrix for a total of p ports including the boundary ports and the power dissipation sources. $g(t) \in \mathbb{R}^{p\times 1}$ represents the power injections from the boundary ports and the power dissipation sources. The boundary port sources modeling has been discussed in Section 2.2.2 and the power dissipation sources modeling will be presented in Section 4.2.3.

To reduce the model complexity, model reduction techniques, which has been

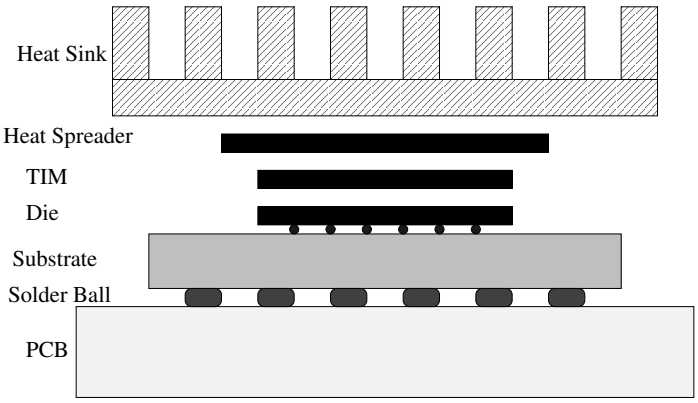


Figure 4.2: The literal structure view of the multi-core system package.

studied intensively in Chapter 3, can be applied to (4.1). However, we will show that existing reduction methods do not work well for the thermal models with many ports in general, which will be addressed in Section 4.2.1.

4.1.3 Adiabatic thermal condition for composibility

The composable models need to be constructed such that the composite system connected by these modules is the same as the system constructed directly from the original structure. It can be proved that to make the thermal model composable, the *adiabatic* thermal conditions (special Neumann's boundary condition) [13]

$$\kappa(\vec{r}, T)\frac{\partial T(\vec{r}, t)}{\partial n_i} = 0 \tag{4.2}$$

should be added at the thermal modules.

Specifically, for the *adiabatic* condition, there are no thermal exchanges between the boundary nodes and outside. When two modules are connected together, the connected boundary nodes of the two modules will become one node in the new system. Heat will flow via normal diffusion as there is no boundary between the two modules. The interface with *adiabatic* thermal conditions is called the *composable*

interface in this paper.

4.2 New composable thermal modeling method

In this section, the new composable thermal modeling method is presented. A two-grid discretization scheme is introduced in Section 4.2.1 to reduce the number of ports of each module in order to enhance the model order reduction efficiency. The stability and the property of the new discretization is shown in Section 4.2.2. Next, in Section 4.2.3, three power dissipation models are presented and their handling by the composable thermal modeling method is studied. Section 4.2.4 introduces the thermal model order reduction technique which generates smaller model for faster simulation speed. Finally, we show how to realize the reduced thermal model into SPICE sub-circuit for easier composition in Section 4.2.5 and how to retrieve the internal node temperature from the reduced thermal model in Section 4.2.6.

4.2.1 Two-grid scheme for discretization

To build compact thermal models from detailed models generated by the finite difference method (or other discretization schemes), one viable way is to partition the original thermal structure into many building blocks and perform reduction on these modules. In case of the multi-core processor example shown in Fig. 4.1 (b) and (c), the natural building blocks are the CPU core module and the cache module.

However, such a simple strategy does not work well in practice. The main reason is that many ports will be generated for such thermal modules since each grid in one direction will generate one port as shown in Fig. 4.3. As a result, the number of ports can be huge if the mesh is large. [1] Existing model order reduction techniques such

[1]The internal power dissipation sources also contribute to the number of ports. But as shown in

as the Krylov subspace or the Gramian based methods do not work well when the number of ports is large because the projection space and the computational time are tightly determined by the port counts. Reducing the number of ports is vital for building the compact thermal models.

The many ports problem mentioned above can be solved by assuming the adjacent boundary nodes are isothermal. In this way, we can reduce the number of ports by merging some adjacent ports into one port. Meanwhile, such port merging or reduction needs to ensure the composability of the thermal modules for fast thermal design exploration. So the port reduction needs to follow geometrical patterns such that the resulting thermal modules can be easily assembled to build different thermal systems.

To achieve the two goals: reducing the port number and ensuring composability, the port merging should equivalently lead to a coarser global grid among all the modules so they can be easily assembled based on this global grid. In the following, we use a $2 \times 2 \times 2$ meshed structure example (in finite difference scheme) shown in Fig. 4.3 to illustrate the idea.[2]

For this meshed structure, there are 8 nodes (cubes) denoted by light solid circles and 24 ports by dark solid circles. Please note, for this special case, every node is on the boundary and is the vertex of the cube. Thus, each of them has 3 ports connected. We also indicate the connection between the nodes by an equivalent thermal resistance. Each node also has an equivalent thermal capacitance connected to the ground, which is not displayed for simplicity. If we apply the adiabatic conditions to all the 6 interfaces, the G, C, T, g and B matrices are shown in (4.3)-(4.7)

Section 4.2.3, only one port is enough to represent all the power dissipation sources in a module.

[2]For simplicity, in this example, we ignore the internal power sources whose effects will be discussed later. Also, we do not show the thermal capacitors and only the ports on the three front faces are shown.

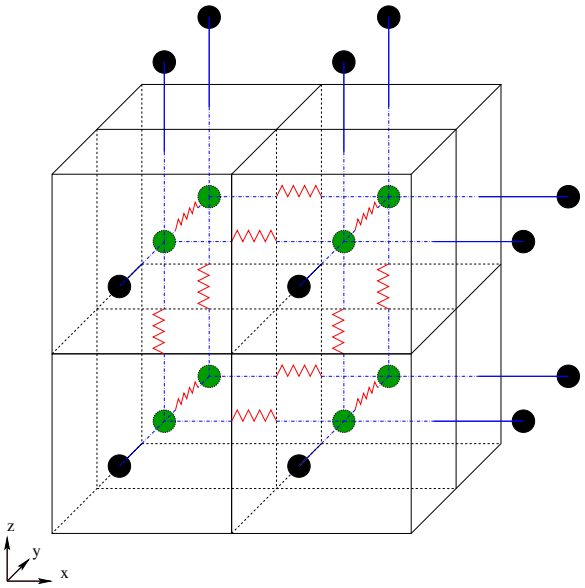


Figure 4.3: A 2×2×2 meshed structure case. The nodes and the ports are represented by light solid circles and dark solid circles, respectively.

respectively.

$$G = \kappa \begin{bmatrix} 3 & -1 & -1 & & -1 & & & \\ -1 & 3 & & -1 & & -1 & & \\ -1 & & 3 & -1 & & & -1 & \\ & -1 & -1 & 3 & & & & -1 \\ -1 & & & & 3 & -1 & -1 & \\ & -1 & & & -1 & 3 & & -1 \\ & & -1 & & -1 & & 3 & -1 \\ & & & -1 & & -1 & -1 & 3 \end{bmatrix} \tag{4.3}$$

$$C = \rho C_p I_{8\times 8} \tag{4.4}$$

$$T(t) = \begin{bmatrix} T_{1,1,1} & T_{2,1,1} & T_{1,2,1} & T_{2,2,1} & T_{1,1,2} & T_{2,1,2} & T_{1,2,2} & T_{2,2,2} \end{bmatrix}^T \tag{4.5}$$

$$g(t) = [\ g_{1,1,0}\ \ g_{2,1,0}\ \ g_{1,2,0}\ \ g_{2,2,0}\ \ g_{1,1,3}\ \ g_{2,1,3}\ \ g_{1,2,3}\ \ g_{2,2,3}\ \ g_{1,0,1}\ \ g_{2,0,1}\ \ g_{1,0,2}\ \ g_{2,0,2}$$
$$g_{1,3,1}\ \ g_{2,3,1}\ \ g_{1,3,2}\ \ g_{2,3,2}\ \ g_{0,1,1}\ \ g_{0,2,1}\ \ g_{0,1,2}\ \ g_{0,2,2}\ \ g_{3,1,1}\ \ g_{3,2,1}\ \ g_{3,1,2}\ \ g_{3,2,2}\ ]^T \tag{4.6}$$

$$B = \begin{bmatrix}
1 & & & & 1 & & & & 1 & & & \\
& 1 & & & & 1 & & & & & 1 & \\
& & 1 & & & & 1 & & 1 & & & \\
& & & 1 & & & & 1 & & & 1 & \\
& & & & 1 & & 1 & & & 1 & & \\
& & & & & 1 & & 1 & & & & 1 \\
& & & & & & 1 & & 1 & & & \\
& & & & & & & 1 & & 1 & &
\end{bmatrix} \tag{4.7}$$

$T_{i,j,k}$ is the temperature of the node at (i, j, k) in Fig. 4.3 where i, j and k are indexes in the x, y, z directions, respectively. $g_{i,j,k}$ is the power injection into the boundary port at (i, j, k). (i, j, k) can be the position of a node (with $T_{i,j,k}$) or a port (with $g_{i,j,k}$). Notice that $I_{8\times 8}$ is an 8×8 unit matrix. So we have a symmetric positive definite (s.p.d.) matrix G and a unit matrix C. Matrix G will still be s.p.d. when we have power dissipation sources since the power dissipation sources will only change the B matrix. The s.p.d. property can be preserved during the projection based reduction, allowing the reduced models to be further optimized and realized. It will be shown in Section 4.2.5.

In this simple case, we have more ports than nodes. As a result, existing model reduction methods such as the Krylov subspace based method do not work. For instance, even with only one moment matching, one will end up with a larger reduced

model (24×24 matrices, which are also full matrices!) than the original one (8×8 sparse matrices). Although for the large real finite difference thermal model, the port number is one order of magnitude smaller than the grid number, it still greatly degenerates the model reduction efficiency.

In general, if the number of grids in the x, y and z directions are X, Y and Z, the number of nodes in the detailed thermal model is XYZ, while the number of ports is

$$f_{port} = 2(XY + XZ + YZ), \tag{4.8}$$

which indicates the number of ports grows quadratically with the grid count. It is impossible to build very compact thermal models if the port number is not reduced.

To reduce the number of ports, one idea is to reduce the number of grids at the boundaries using a large grid size assuming the boundary surface of the large grid is isothermal. We can simply merge the adjacent boundary nodes into one node and then build the new finite difference matrices. But this scheme requires the construction of the new G and C matrices in (4.1). A better way is to just merge the adjacent ports of the boundary nodes into one port assuming all the involved nodes are connected to the same port. Using the example in Fig. 4.3, if we merge 4 adjacent ports into 1 port in each boundary face as show in Fig. 4.4, the port number will be reduced to

6. The B and $g(t)$ matrices after this boundary merge are shown as

$$B_m = \begin{bmatrix} 1 & & 1 & & 1 & \\ 1 & & 1 & & & 1 \\ 1 & & & 1 & 1 & \\ 1 & & & 1 & & 1 \\ & 1 & 1 & & 1 & \\ & 1 & 1 & & & 1 \\ & 1 & & 1 & 1 & \\ & 1 & & 1 & & 1 \end{bmatrix}, \tag{4.9}$$

$$g_m(t) = [g_{x,y,0}, g_{x,y,3}, g_{x,0,z}, g_{x,3,z}, g_{0,y,z}, g_{3,y,z}]^T, \tag{4.10}$$

where the subscripts x, y and z represent the merged indexes in the x, y and z directions, respectively. Such a port reduction also leads to a larger grid size at the boundaries as shown in Fig. 4.4.

The boundary merging will introduce errors because some adjacent nodes with different temperatures are merged together. The errors, however, are relatively small when the power dissipation is uniformly distributed inside the module and will be larger when the power dissipation distribution has a large gradient just next to the boundary. As shown in our experiment, even in the latter case, the errors are still within 2-3$^\circ C$ and the large errors only appear at the boundaries. In order to further improve the accuracy, a finer boundary grid can be used through merging less ports into one port. This will result in a less compact reduced model as the model reduction efficiency is the trade-off in this case.

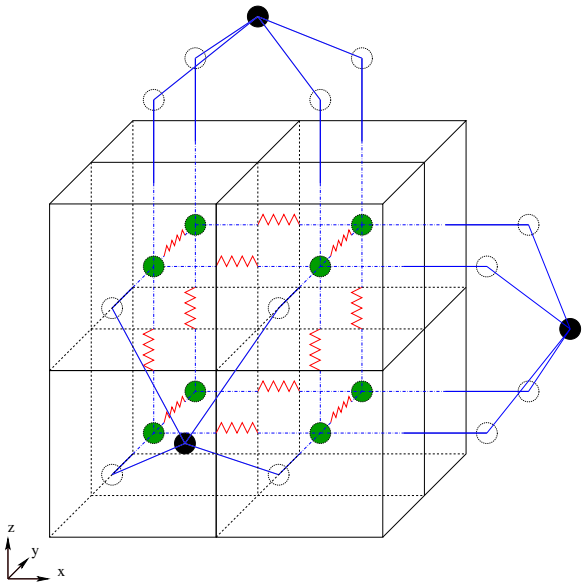


Figure 4.4: A $2 \times 2 \times 2$ meshed structure case where the boundary faces (ports) are merged. The original ports are shown as the hollow circles and the new ports are represented by the dark solid circles.

4.2.2 Stability and property of the new discretization

In the new two-grid discretization, we only change the right-hand side matrix B without altering the G and C matrices in (4.1), so the resulting thermal matrices will still be symmetric positive definite. This is important for more accurate reduction and passivity preservation during the reduction process.

For the finite difference method, if h is the space difference (assume it is the same for all the three dimensions), the time step Δt will be restricted by the following equation (when the explicit time discretization method is used) [51]:

$$0 < \Delta t < \frac{1}{2\beta(1/\Delta x^2 + 1/\Delta y^2 + 1/\Delta z^2)} \tag{4.11}$$

where $\beta = \frac{\kappa}{\rho C_p}$. For the new method, since equivalently we increase the space difference at the boundary, it will give a larger upper bound for Δt. As a result, if we stick with the Δt for the model without port merging, the simulation will be stable.

4.2.3 Power dissipation modeling

Each module has its own power dissipation property, approximately characterized as the power dissipation distribution within its 3D structure. In this paper, the power dissipation is modeled in three ways. The first model assumes the power inside a module is uniformly distributed. We call it the *uniform model*. The second model assumes the power is concentrated in some power centers, and is called the *center model*. The third model, called the *distribution model*, follows the measured or given power distribution across the module. The first two models are simple but less accurate as some simplifications are made. Specifically, the uniform model generally gives optimistic junction temperature predictions while the center model generally gives pessimistic ones. Although the distribution model is more accurate, it requires the detailed power distribution, which may be unavailable. Please note that the first two models are just two special cases of the third model: the uniform model is the distribution model with a uniform distribution and the center model is the distribution model with zeros everywhere except for the power centers.

In our composable thermal model, all of the three power dissipation models can be easily handled with a simple formulation. Specifically, the power dissipation of a module is represented at the right hand side of our thermal model as $B_{pw} \times g_{pw}(t)$. $B_{pw} \in \mathbb{R}^{n\times 1}$ represents the power distribution in the module and is one column of the $B \in \mathbb{R}^{n\times p}$ matrix in (4.1) (the other columns in B are for the boundary condition power sources). $g_{pw}(t)$ is the total power dissipation in the module and is an element of $g(t)$. Each element of B_{pw} corresponds to a finite difference node and its value is the weight of the power dissipation of the finite difference node against the total power dissipation of the module. For the uniform model, each element of B_{pw} is $1/n$ and for the center model, all elements have the value 0 except for the power center

nodes whose value is $1/n_c$ where n_c is the number of the power centers. The element values of the distribution model are set according to the provided power distribution. Given the power distribution function as $f(x, y, z)$, the ith element of B_{pw}, denoted as b^i_{pw}, is calculated as

$$b^i_{pw} = \frac{\int_{x_i^-}^{x_i^+} \int_{y_i^-}^{y_i^+} \int_{z_i^-}^{z_i^+} f(x, y, z)\, dz\, dy\, dx}{\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y, z)\, dz\, dy\, dx}, \tag{4.12}$$

where x_i^-, y_i^- and z_i^- are the starting coordinates of the ith finite difference element and x_i^+, y_i^+ and z_i^+ are its ending coordinates.

We use a one dimensional module (x direction) as an example to illustrate the proposed idea. Assume the module is positioned on $x \in [0, 1]$ and discretized into three nodes. For the uniform model, $B_{pw} = [1/3, 1/3, 1/3]^T$ and $B_{pw} \times g_{pw}(t) = [g_{pw}(t)/3, g_{pw}(t)/3, g_{pw}(t)/3]^T$ indicate the total power $g_{pw}(t)$ is equally divided into three parts which are injected into three finite difference nodes separately. For the center model with only one power center at $x = 0.5$, there will be no power injection into the first and the third nodes while all the power flows into the second node. As a result, the B_{pw} vector is $[0, 1, 0]^T$. For the distribution model, assume we have the simple power distribution $f(x) = x$, the B_{pw} vector should be $[1/9, 1/3, 5/9]^T$ according to (4.12).

In summary, all the three power dissipation models can be handled by our thermal model using the same formulation. Additionally, no matter which power dissipation model is used, there will be only one power dissipation handling column B_{pw} inside B in our thermal model. As a result, the model reduction efficiency will not be degraded.

4.2.4 Thermal model reduction

Reducing the complexity of linear dynamic systems by means of model order reduction has been introduced in Section 2.1 and studied intensively in Chapter 3. For compact thermal modeling, the Krylov subspace based approaches have been applied to reduce the large models [19, 20]. In this paper, we apply a more accurate sampling-based reduction technique [69, 55], which is based on the globally accurate balanced truncation realization reduction scheme.

For a dynamic thermal system described in (4.1), the goal is to find a subspace represented by the projection matrix $V \in \mathbb{R}^{n\times k}$, resulting in the approximation

$$T \approx V\tilde{T}, \tag{4.13}$$

where $\tilde{T} \in \mathbb{R}^{k\times 1}$ is the state in the reduced model and $k \ll n$. This is not an equation but an approximation because $\tilde{T}$ has a smaller dimension than T and the information in some dimensions may have been lost. The resulting thermal equation becomes

$$\tilde{C}\frac{d\tilde{T}(t)}{dt} + \tilde{G}\tilde{T}(t) = \tilde{B}g(t), \tag{4.14}$$

where

$$\tilde{G} = V^T GV,\ \tilde{C} = V^T CV,\ \tilde{B} = V^T B, \tag{4.15}$$

with $\tilde{G} \in \mathbb{R}^{k\times k}$, $\tilde{C} \in \mathbb{R}^{k\times k}$, $\tilde{B} \in \mathbb{R}^{k\times p}$. The resulting reduced model should preserve the port behaviors of the original system.

One way to obtain the projection matrix V is by means of the truncated balanced realization (TBR) method. TBR first performs the coordinate changes such that the controllability and observability (described by their corresponding Gramians X and Y) are the same for every state. In this way, the balanced weak states can

be truncated by picking only the dominant eigenspace of XY as the projection matrix V. However, the computational cost to obtain the Gramians is $O(n^3)$, which makes it too expensive for integrated circuits problems and thus an efficient Gramian approximation technique is highly appreciated.

Recently, some fast TBR methods have been proposed [69, 55] to mitigate the high computational cost of the standard TBR method, where the Gramians are approximated using the Monte-Carlo sampling approach. Specifically, the Gramian, denoted as X, [3] is explicitly written as the following integral in the frequency domain

$$X = \int_{-\infty}^{+\infty} (j\omega C + G)^{-1} B B^T (j\omega C + G)^{-H} d\omega, \tag{4.16}$$

where the superscript H denotes the Hermitian transpose. The dominant eigenspace of X is used as the projection matrix V [55] for model order reduction. As discussed before, computing the exact values of X in (4.16) is extremely expensive for large scale systems. A cheaper computation of the Gramian X is achieved by evaluating the definite integral in (4.16) by summation approximation, using numerical quadrature methods [35] as discussed below.

The integral of a function $f(x)$ can be approximated as summation by numerical quadrature methods as

$$\int_a^b f(x)dx \approx \sum_{k=1}^{m} w_k f(x_k), \tag{4.17}$$

where w_k, $k = 1, 2, ..., m$, are referred to as the quadrature point weights, while the interpolation points x_k, $k = 1, 2, ..., m$, are called the quadrature points. The selections of w_k and x_k depend on the quadrature methods, such as Newton-Cotes, Gaussian quadrature rules or even random-based Monte Carlo method [35].

[3] For thermal system, which can be interpreted as the equivalent RC circuit, both the controllability Gramian X and observability Gramian Y are identical.

According to (4.17), the Grammian X in the integral form (4.16) can be also approximated as $\hat{X}$ in summation form like the following

$$X \approx \hat{X} = \sum_{k=1}^{m} w_k z_k z_k^H, \tag{4.18}$$

where

$$z_k = z(j\omega_k) = (j\omega_k C + G)^{-1} B \tag{4.19}$$

is called the kth snapshot of the system in the frequency domain (let ω_k be the kth sample point in frequency) and w_k is the weight from a specific numerical quadrature method.

For conveniences, $\hat{X}$ in (4.18) can be equivalently written in matrix form as

$$\hat{X} = ZW^2Z^H, \tag{4.20}$$

where

$$Z = [z_1, z_2, \ldots, z_m], \tag{4.21}$$

and W is a diagonal matrix with diagonal entries $w_{kk} = \sqrt{w_k}$.

As the last step, the dominant eigenspace of X needs to be computed to serve as the projection matrix V. Because from (4.20), there is

$$\hat{X} = (ZW)(ZW)^H, \tag{4.22}$$

the eigenspace of $\hat{X}$ is the same as the left singular space of ZW. In order to save computational cost, instead of explicitly performing eigenvalue decomposition on $\hat{X}$,

singular value decomposition (SVD) on ZW is usually performed as

$$ZW = VSU. \tag{4.23}$$

Then V, which gives the dominant eigenspace of $\hat{X}$, is used as the projection matrix.

4.2.5 Circuit realization and model generation

We have obtained the reduced matrices of the composable models. The next step is to generate the composite system using these models. One method is directly composing the module matrices into the whole system matrices. However, the reduced composable model has complex structures and many new node connections will be introduced by composing the two or more modules. As a result, directly composing the matrices requires reformulating the structure of the composable model matrices, which is very complicated and error-prone. Instead, we need to package the composable model as a black box and use it for easy composition of the whole system. In this paper, we realize the reduced composable model matrices into a SPICE netlist and package it as a sub-circuit. SPICE will automatically build the composite system matrices from the hierarchical netlist.

Since the congruence transformation is used in (4.15), the reduced matrices $\tilde{G}$ and $\tilde{C}$ are symmetric and $\tilde{G}$ is positive definite. As a result, the reduced model can be further diagonalized by a generalized eigen-decomposition of a definite matrix pencil $\tilde{C} - \lambda\tilde{G}$ [7]. With the eigenvector matrix $P \in \mathbb{R}^{k\times k}$ of $\tilde{C} - \lambda\tilde{G}$, we can perform the coordinate change of (4.14) as

$$\tilde{T} = P\hat{T}, \tag{4.24}$$

and generate

$$\hat{C}\frac{d\hat{T}(t)}{dt} + \hat{G}\hat{T}(t) = \hat{B}g(t), \tag{4.25}$$

where

$$\hat{G} = P^T\tilde{G}P,\ \hat{C} = P^T\tilde{C}P,\ \hat{B} = P^T\tilde{B}. \tag{4.26}$$

$\hat{G}$ and $\hat{C}$ are two *diagonal* matrices here. The diagonalized system in (4.25) can be realized into RC circuits with controlled sources, and then simulated using SPICE-type simulators [19].

Next, all the reduced composable models are realized into the SPICE compatible format using SPICE *.subckt* command. Specifically, the i-th diagonal element in the diagonal matrix $\hat{G}$, $\hat{G}_{ii}$, is realized into a resistor from the i-th node to the ground with the value $1/\hat{G}_{ii}$; the i-th diagonal element in the diagonal matrix $\hat{C}$, $\hat{C}_{ii}$, is realized into a capacitor from the i-th node to the ground with the value $\hat{C}_{ii}$. Realization of the $\hat{B}$ matrix is more complicated. The element in the i-th row and j-th column in the dense matrix $\hat{B}$, $\hat{B}_{ij}$, is realized into two components: one component is a current controlled current source (CCCS) in parallel with the i-th resistor and capacitor with the j-th independent current source (power source) as the control current and $\hat{B}_{ij}$ as the current gain. The other component is a voltage controlled voltage source (VCVS) in series with the j-th independent current with the i-th node voltage as the control voltage and $\hat{B}_{ij}$ as the voltage gain.

We give a simple example to illustrate the circuit realization. Assume there is a small 2×2 diagonalized system with system matrices $\hat{C}$, $\hat{G}$ and $\hat{B}$ as the following

$$\hat{C} = \begin{bmatrix} \hat{C}_{11} & 0 \\ 0 & \hat{C}_{22} \end{bmatrix},\ \hat{G} = \begin{bmatrix} \hat{G}_{11} & 0 \\ 0 & \hat{G}_{22} \end{bmatrix},\ \hat{B} = \begin{bmatrix} \hat{B}_{11} & \hat{B}_{12} \\ \hat{B}_{21} & \hat{B}_{22} \end{bmatrix}. \tag{4.27}$$

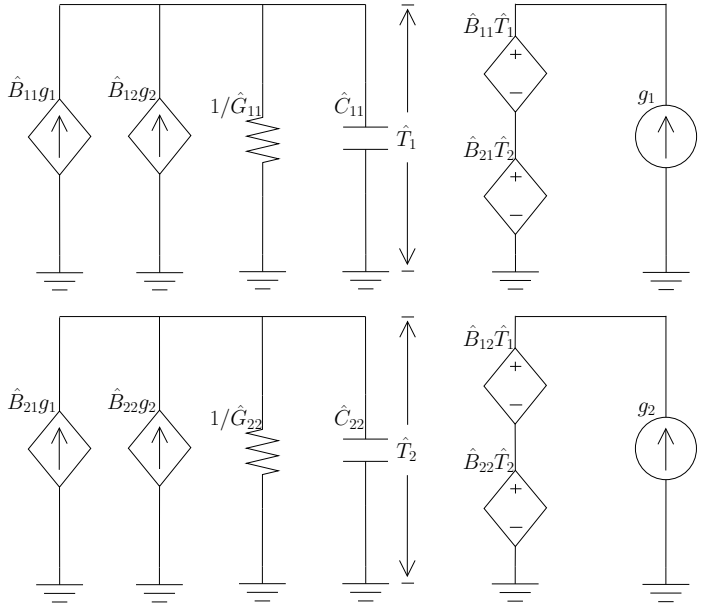


Figure 4.5: A simple circuit realization example.

The realized circuit is shown in Fig. 4.5. The two thermal nodes are shown on the left side with temperatures $\hat{T}_1$ and $\hat{T}_2$ respectively. The two independent current sources (power sources) are shown on the right side with values g_1 and g_2.

After the realization process, we can easily build different multi-core architectures (their thermal circuits) on top of these basic thermal building-block modules in SPICE netlists. All the boundary conditions in terms of equivalent resistances and sources will be added once the architectures are generated.

Although we are realizing matrices into the SPICE netlist, which needs to be re-constructed into matrices in the simulation stage, the overhead is very small since the matrices are reduced and have small sizes. In addition, circuit realizing will only be performed for the small number of basic modules, and will only be performed once at the composable model library building stage.

4.2.6 Internal node temperature retrieval technique

Model reduction presented in Section 4.2.4 and the diagonalizing technique introduced in Section 4.2.5 will destroy the internal state structures and only keep the port behaviors (This is indeed what model reduction tries to achieve). Specifically, the final diagonalized reduced model (4.25) has the state $\hat{T}$ with k dimensions while in the original finite difference model (4.1), the state T has n dimensions. From the reduced model, we can only observe the port temperatures but the temperatures of the original finite difference nodes cannot be observed directly.

Fortunately, the original internal node information can be approximately recovered from the reduced model. Remembering (4.13) and (4.24), we have

$$\bar{T} = V\tilde{T} = VP\hat{T}, \tag{4.28}$$

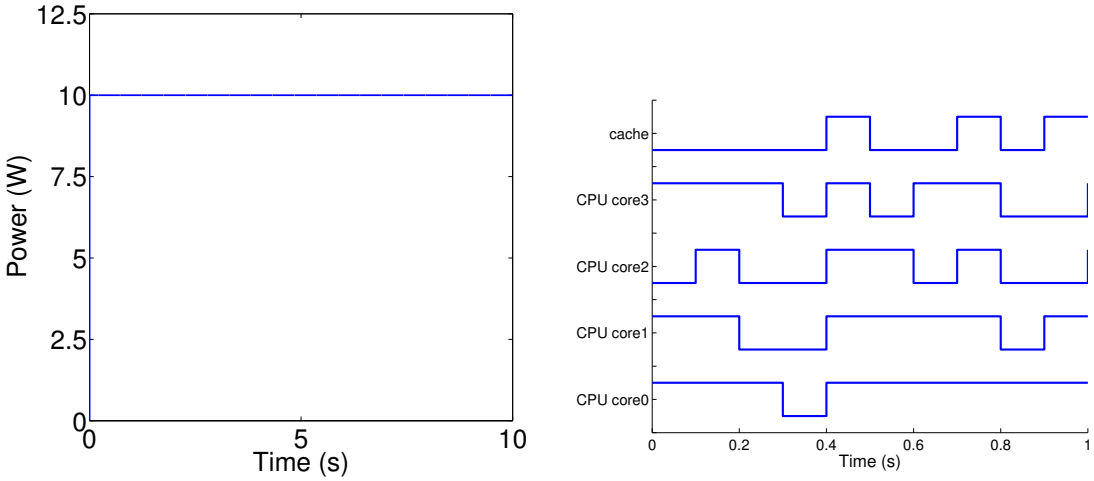
where

$$\bar{T} \approx T \in \mathbb{R}^{n\times 1} \tag{4.29}$$

is the original state approximation. As a result, the original state T can be easily recovered as $\bar{T}$ from the reduced state $\hat{T}$ through formulation (4.28).

4.3 Experimental results

The proposed method, *ThermComp*, has been implemented in MATLAB. First, we build the finite difference models and generate their reduced composable models for a single CPU module and a single cache module using the two-grid discretization based finite difference method and the sampling based model reduction technique. Then, we compose multi-core systems using the reduced composable CPU core and cache



(a) Power input source type 1: step input for steady state analysis, all the CPU and cache modules share the same input.

(b) Power input source type 2: random power traces for transient analysis.

Figure 4.6: Power input waveform.

modules. The original finite difference models of the corresponding multi-core systems are built directly (without port merge process, composition and model reduction) for the comparison purpose. Finally, thermal transient simulation is performed using HSPICE on a Linux server with Intel quad-core CPU and 16GB memory to obtain the temperature distribution for both the original and the reduced composite thermal systems.

To build the composable models for CPU core and cache modules, we set up the size of the discretization grid as $32 \times 16 \times 3$ for CPU core module ($8mm \times 4mm \times 0.75mm$) and $64 \times 32 \times 3$ for cache module ($16mm \times 8mm \times 0.75mm$), in order to keep both CPU core and cache modules sharing the same discretization step value $\Delta x = \Delta y = \Delta z = 0.25mm$. This is because the length and width of cache module are twice of the CPU core while the height is the same. Here, we choose the thermal conductivity $\kappa = 149W/(m^{\circ}C)$, material density $\rho = 2300Kg/m^3$ and

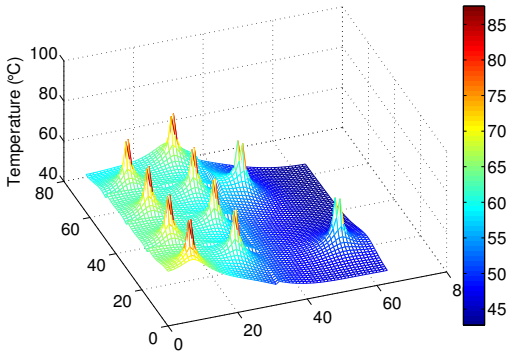
specific heat $c_p = 700J/(Kg°C)$. The lateral and top surfaces are supposed to be adiabatic ($h_i = 0W/m^2K$) and the heat exchanges with the ambient through the bottom surface ($h_i = 10000W/m^2K$), which is convective to model the heat spreader effects.

Two types of power dissipation waveforms are shown in Fig. 4.6. The one in (a) is the type 1 total power input in one module, used for steady state analysis. All the basic modules (CPU and cache) share this same total input. As discussed in Section 4.2.3, this total power may concentrate in some power centers, distribute across the module uniformly or following a distribution according to the power dissipation model used. In this experiment, we use both uniform model and center model for the power dissipation modeling. For the center model, four power centers are put into a module at its middle layer. We emphasis that *ThermComp* is able to handle all of the three power dissipation models (uniform, center and distribution power models). The waveforms in (b) is the type 2 total power inputs for different modules, used for transient analysis. As shown in the figure, different modules have different power input waveforms.

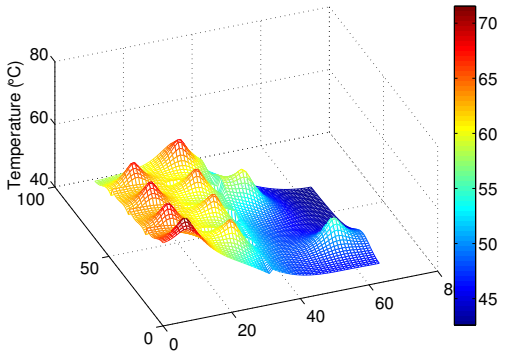
We have composed a quad-core CPU as shown in Fig. 4.1 (b) using the CPU core module and the cache module shown in Fig. 4.1 (a).

4.3.1 Comparison with finite difference simulation

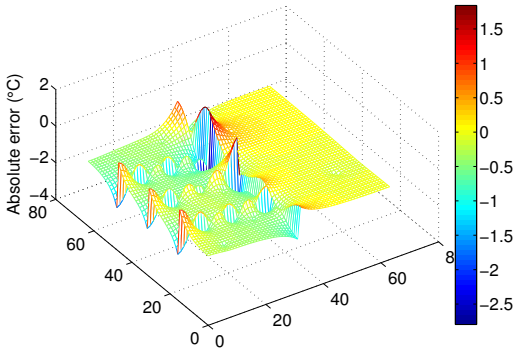
We first apply power source type 1 for the steady state analysis and use the center power dissipation model. There are four power centers in each module and all of them are very close to the boundaries in order to validate the port merge process. Fig. 4.7 (a) and (b) show the temperature distribution at the middle layer and the convective surface layer (the bottom surface) of the die, respectively. All of the internal nodes



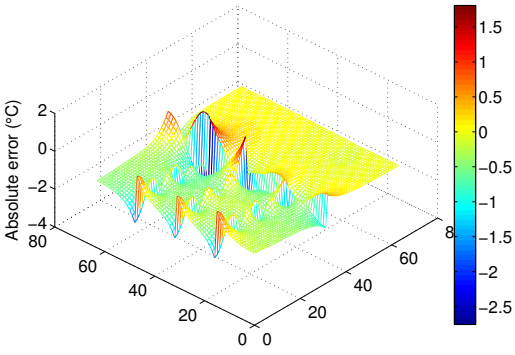
(a) Temperature distribution at the middle layer.



(b) Temperature distribution at the convective surface.

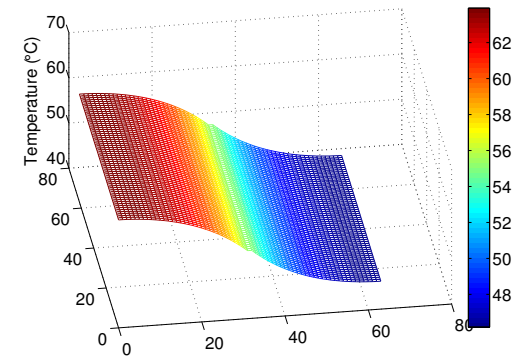


(c) Temperature error at the middle layer.

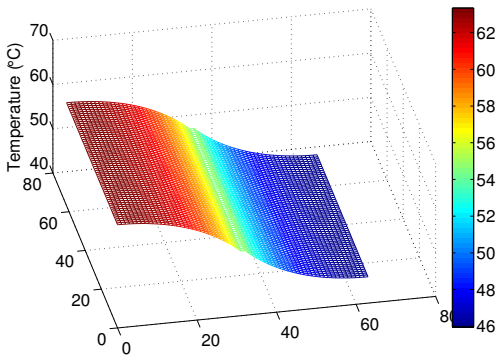


(d) Temperature error at the convective surface.

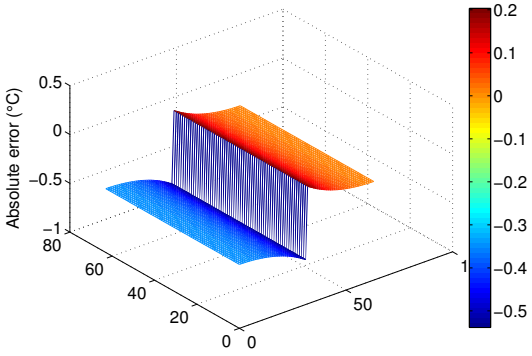
Figure 4.7: Composed quad-core microprocessor temperature distribution with type 1 power input and center power dissipation model.



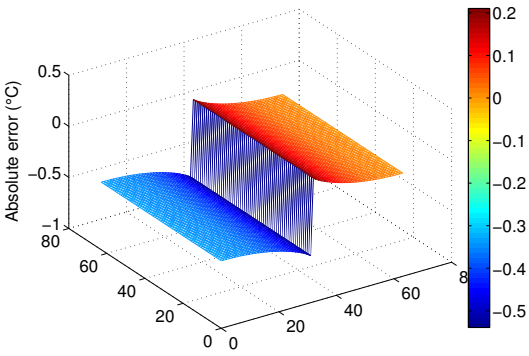
(a) Temperature distribution at the middle layer.



(b) Temperature distribution at the convective surface.



(c) Temperature error at the middle layer.



(d) Temperature error at the convective surface.

Figure 4.8: Composed quad-core microprocessor temperature distribution with type 1 power input and uniform power dissipation model.

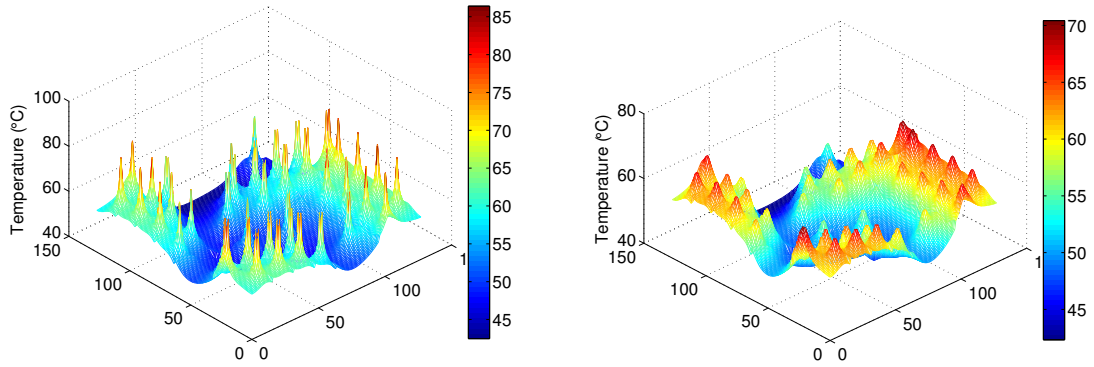
can be observed thanks to the internal node retrieval technique used. As we can see, the highest temperatures are on the left hand side and the hot spots appear at the power centers. This is expected as the CPU cores are on the left hand side.

The error plots of *ThermComp* are shown in Fig. 4.7 (c) and (d). We can see although the power centers generated large gradients at the boundaries, the largest temperature error is $2.5°C$. The errors are almost $0°C$ elsewhere other than at the boundaries. This indicates the high accuracy of our composite model.

Next, we change the power dissipation model to the uniform one, i.e., the power dissipation is uniformly distributed within the module. Fig. 4.8 (a) and (b) show the temperature distributions at the middle layer and the convective surface. It is observed that the temperature decreases from the CPU modules to the cache module smoothly and the convective surface has a temperature higher than the middle layer. This is because of the uniform power dissipation in the modules.

The errors of *ThermComp* with the uniform power distribution are shown in Fig. 4.8 (c) and (d). Since there are relatively small temperature differences on the boundaries, the errors of the port merge process are very small, within $0.5°C$ as shown in the figures. A non-smooth transition in the error between two modules (CPU core and cache) can be seem in the figure. This phenomenon is due to the boundary merging. As discussed in Section 4.2.1, several thermal nodes are merged together and the information of the temperature variations among these boundary thermal nodes are lost. As a trade-off between the simulation accuracy and reduction efficiency, this effect can be minimized by merging less boundary nodes.

We have also composed the 16-core architecture, which is shown in Fig. 4.1 (c), using the CPU core and cache in Fig. 4.1 (a). Fig. 4.9 shows the temperatures at the middle layer and convective surface given input type 1 with the center power model. The temperature distributions with the uniform power model are shown in Fig. 4.10.



(a) Temperature distribution at the middle layer.

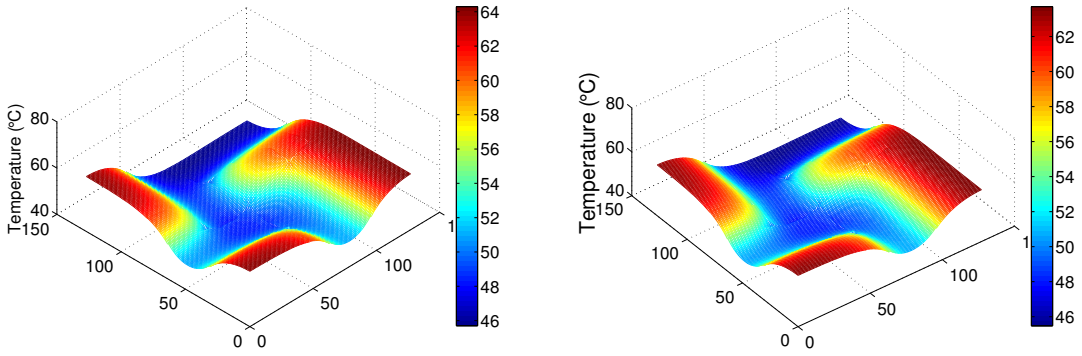
(b) Temperature distribution at the convective surface.

Figure 4.9: 16-core microprocessor temperature distribution with type 1 power input and center power dissipation model.

The highest temperatures are on the right hand side as we have more CPU modules there. The middle has the lowest temperatures as many cache modules are located close to the center. Similar to the quad-core case, the uniform power distribution model gives a relatively smoother temperature distribution.

Next, we analyze the dynamic thermal behavior of the 16-core architecture using the power input type 2 shown in Fig. 4.6 (b) with the center power model. Fig. 4.11 compares the accuracy between the original finite difference method and the composite model at several randomly picked power centers. The two transient results are almost the same and the large errors at the fast switch points are actually due to the small errors in the time axis. Since the waveforms at some time points switch very fast (like the ideal step), a tiny timing error will lead to a large temperature difference. But those differences do not represent any meaningful errors in practice.

Another dynamic thermal behavior analysis of the 16-core architecture is performed using a power trace obtained from power estimator Wattch [11] by running



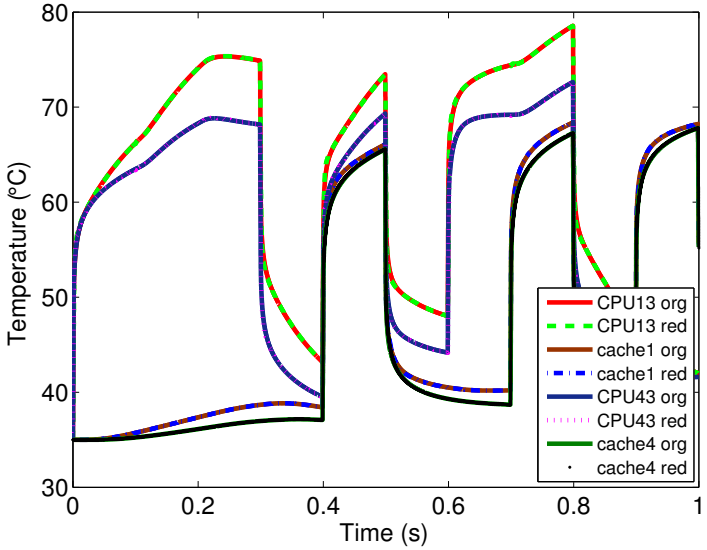
(a) Temperature distribution at the middle layer.

(b) Temperature distribution at the convective surface.

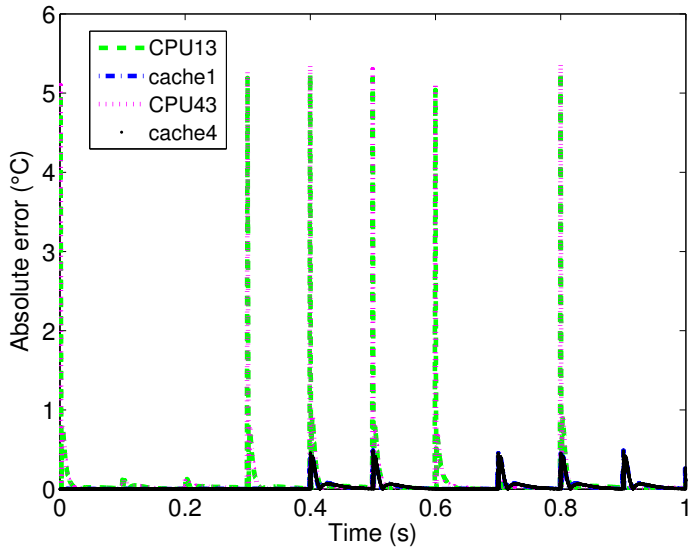
Figure 4.10: 16-core microprocessor temperature distribution with type 1 power input and uniform power dissipation model.

the SPEC [30] bzip2 benchmark. The bzip2 benchmark power trace, which has more complex transient behavior, is applied to each module using the center power model. Fig. 4.12 shows the transient results from the original finite difference model and the composite model. Similar to results from the previous experiment, the composite model generated accurate transient waveforms.

The CPU time results for different configurations of modules are shown in Table 4.1. In the table, xx *org* means the original models directly built from finite difference method and xx *red* means the reduced systems built from composable models (*ThermComp*). It can be seen that with the reduced composite thermal models, we can achieve about two orders of magnitude speedup for the transient simulation of the multi-core systems. In addition, the reduced composite thermal models lead to much smaller memory footprint than the original models.



(a) Transient simulation results for both full model (xx org) and reduced composite model (xx red).



(b) Error plot of the transient simulation.

Figure 4.11: Transient simulation accuracy comparison with the full model at some power centers for the 16-core architecture, using power input source type 2.

1	2	3	4	5	6	7	8
Circuit	#Node	#Elem	Run time (s)		Memory	Speedup	
			Trans	Total	(mb)	Trans	Total
CPU core org	3475	12340	4.7	10.9	32.6		
CPU core red	225	448	0.03	0.05	0.7	157	218
Cache org	13093	48232	31.9	100	30		
Cache red	2145	4288	0.2	3.2	6.2	160	31
2 CPU cores org	6949	24678	9.3	33.4	68		
2 CPU cores red	449	894	0.05	0.1	1.3	186	334
4 CPU cores org	13897	49352	42.3	85.8	28		
4 CPU cores red	897	1784	0.1	0.35	2.6	423	245
Quad-core chip org	26989	97582	167	520	93		
Quad-core chip red	3041	6070	0.48	6.9	8.5	348	75
16-core chip org	107953	390312	392	24146	224		
16-core chip red	12161	24264	15.5	114	28	25	212

Table 4.1: CPU time comparisons between the original models (xx *org*) and reduced thermal systems using composable models (xx *red*).

1	2	3	4	5
Circuit	ThermComp		HotSpot	
	node #	total time (s)	node #	total time (s)
Quad-core chip	26989	6.9	16384	86
16-core chip	107953	114	65536	586

Table 4.2: CPU time comparison between ThermComp and HotSpot.

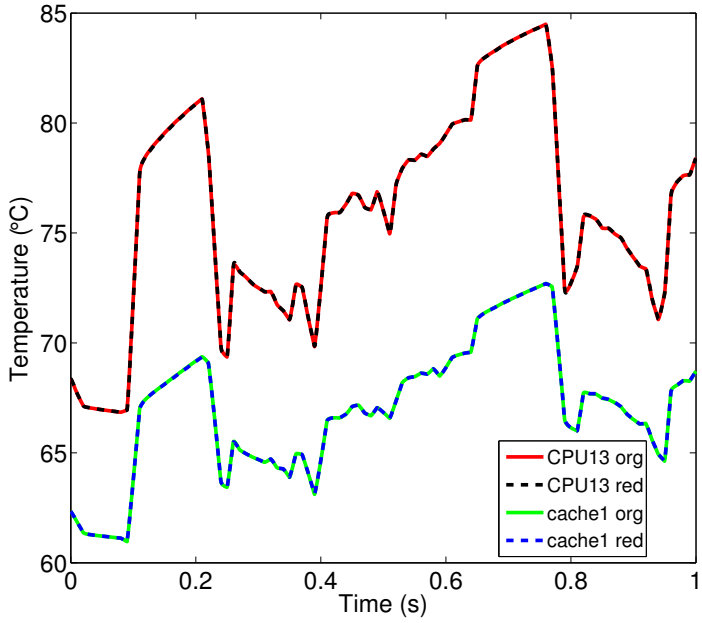
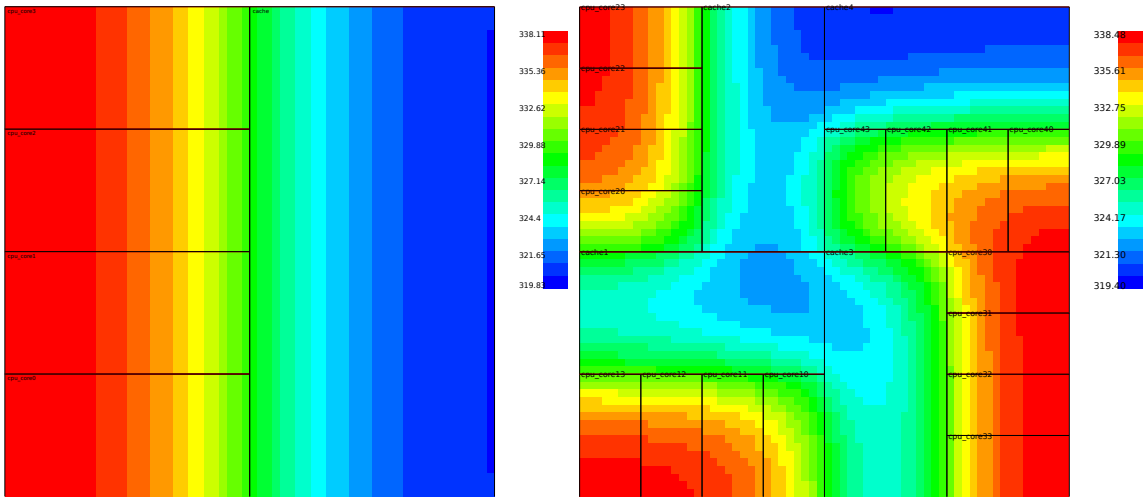


Figure 4.12: Transient simulation accuracy comparison with the full model at some power centers for the 16-core architecture, using power input from bzip2 benchmark.

4.3.2 Comparison with HotSpot program

To further illustrate the advantages of the proposed method, we compared the new modeling technique with existing thermal modeling program HotSpot [32]. To perform accuracy comparison between the two programs, we have configured HotSpot to have the same settings as *ThermComp*. HotSpot's heat sink and heat spreader are set to be very thin (10^{-7}m in thickness) in order to neglect their effects. HotSpot [32] steady state results of the quad-core and 16-core architectures using the power input in Fig. 4.6 (a) are shown in Fig. 4.13. Being a uniform power distribution based method, HotSpot gives very similar results compared to *ThermComp* models with uniform power distribution. The maximum difference between ThermComp and HotSpot is within $1°C$.

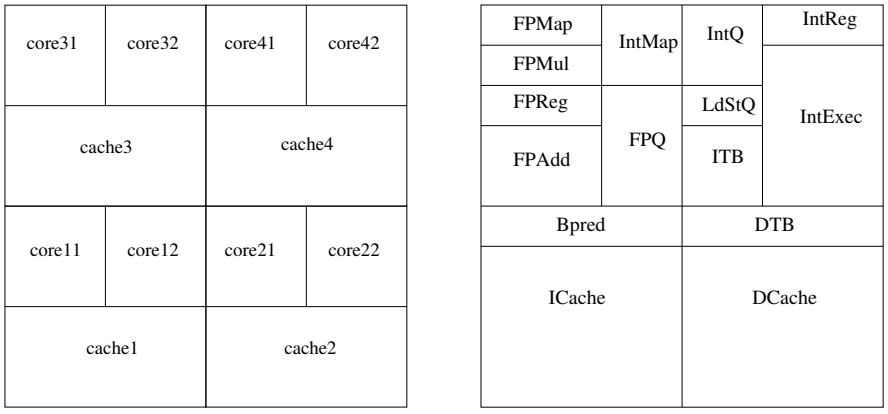
In addition, in order to further verify our method using more realistic CPU archi-



(a) Temperature distribution of the quad-core microprocessor.

(b) Temperature distribution of the 16-core microprocessor.

Figure 4.13: HotSpot steady state results of the quad-core and 16-core microprocessors.



(a) The basic structure of the 8-core alpha chip, composed of 8 cores and 4 caches.

(b) The detailed structure of each core, similar to the alpha ev6 processor.

Figure 4.14: The architecture of the 8-core alpha chip.

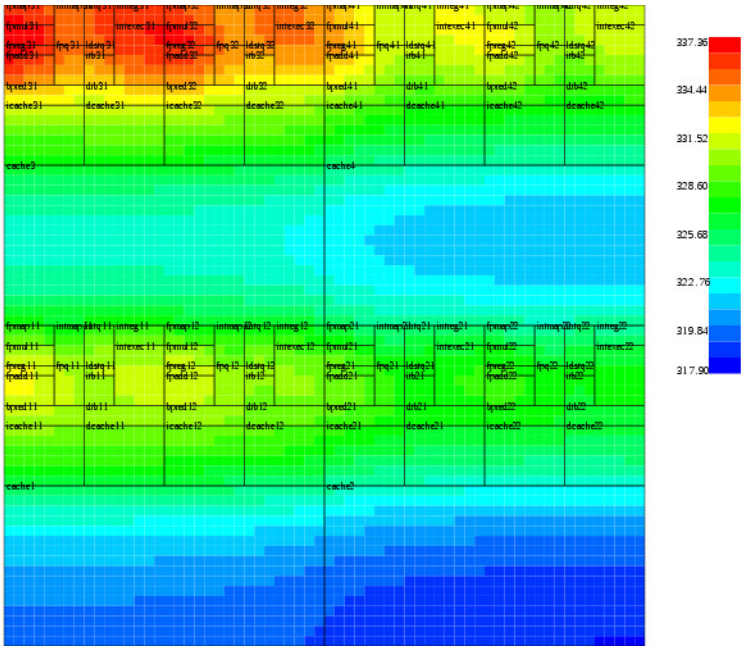
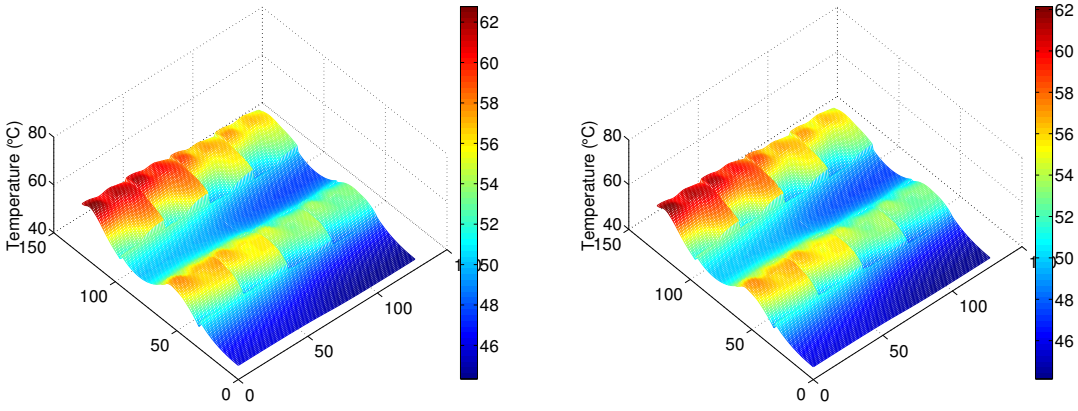


Figure 4.15: HotSpot steady state results of the 8-core microprocessor.



(a) Temperature distribution at the middle layer.

(b) Temperature distribution of the 16-core microprocessor.

Figure 4.16: Steady state results of the 8-core microprocessor with the composable thermal model with the distributed power model.

tecture with general distributed power model case, we have built an 8-core CPU model with the core structure similar to the alpha ev6 architecture as shown in Fig. 4.14. The power trace of the 8-core CPU is obtained by running Wattch [11] power estimator with SPEC benchmarks [30]. For the composite model, in order to verify the distributed power model, we treat each core as a module, and extract the power distribution by running different SPEC benchmarks and let the power distribution follow the average power ratio among all the functional blocks. The temperature distribution is obtained by running bizp2 benchmark on the bottom left dual-core structure (core11, core12, and cache1), mcf benchmark on the bottom right dual-core structure (core21, core22, and cache2), swim benchmark on the top left dual-core structure (core31, core32, and cache3), and mgrid benchmark on the top right dual-core structure (core41, core42, and cache4). The steady state results of the HotSpot results on the same 8-core structure is shown in Fig. 4.15. The steady state results from the composite model with the distributed power model are shown in Fig. 4.16. Comparing Fig. 4.15 and Fig. 4.16, except for the errors on the boundaries of the modules, the other parts of the temperature distribution from the composable model matches the HotSpot results very well.

The CPU time comparison with HotSpot using the power trace from Fig. 4.6 (a) is shown in Table 4.2. In the steady state case, we have configured the heat sink and heat spreader to a smaller value in order to neglect their effects, however, doing this will cause the transient simulation hard to converge and result in a very large transient simulation time. In order to make a fair comparison, in the transient simulation case, we have relaxed the thickness of both the heat sink and heat spreader to the default values. Since HotSpot grid mode restricts the grid dimensions to be powers of two, we make HotSpot node number to be smaller than *ThermComp* node number in the case where we cannot make them the same. *ThermComp* is faster than HotSpot even

with larger node number (higher resolution) as shown in the table.

4.4 Summary

In this chapter, we have proposed a new compact thermal modeling technique for architecture level thermal design space exploration for multi-core microprocessors. The new approach, called ThermComp, builds the models from the first principles by the finite difference method for each basic module and reduces the model complexity by the sampling based model order reduction technique. To improve the reduction efficiency, we try to merge the boundary nodes of modules, which lead to different space discretizations for the whole thermal system. ThermComp preserves the accuracy of fine-grained models while trying to achieve the accuracy of course-grained models. The resulting reduced models are then realized into equivalent RC circuits for easy assembling and simulation by circuit level SPICE simulators. Experimental results on a number of multi-core microprocessor architectures show that the new approach can easily build accurate thermal circuits from the composable thermal models. The reduced composite models lead to orders of magnitude speedup over the standard finite difference models and are much faster than the HotSpot method with similar accuracy.

Chapter 5

Runtime Thermal Estimation and Prediction for Dynamic Thermal Management of Microprocessors

In this chapter, the full-chip error-tolerant runtime thermal estimation and prediction method FRETEP is presented. FRETEP accurately estimates and predicts the temperature of microprocessors with low overhead at runtime. The estimated and predicted temperature information is able to help the dynamic thermal management (DTM) of the chip. The challenges of runtime thermal estimation and prediction are shown first in Section 5.1. Then, the thermal estimation and prediction method is presented in Section 5.2. Next, in order to further improve the estimation accuracy, a power-driven thermal sensor placement algorithm is demonstrated in Section 5.3. The algorithm flow is summarized in Section 5.4. Finally, experimental results are given in Section 5.5 and Section 5.6 concludes this chapter.

5.1 Challenges for accurate thermal estimation and prediction

The basics of the thermal simulation can be found in Section 2.3. Unfortunately, the traditional thermal simulation cannot be directly applied to the runtime thermal estimation and prediction of microprocessors. In this section, we summarize the challenges of current thermal estimation and prediction and provide solutions in the following sections.

For convenience, the thermal system equation (2.29) here

$$C\frac{dT(t)}{dt} + GT(t) = BU(t), \tag{5.1}$$

where its right hand side is also written as

$$J(t) = BU(t). \tag{5.2}$$

Our goal is to accurately compute all the thermal node temperatures $T(t)$ and even predict their values in the near future with small overhead. However, there are several practical problems preventing us from getting accurate temperatures in reality.

The first problem comes from the inaccurate power estimations and inaccurate thermal model. From (5.1), $T(t)$ can be solved numerically given the power inputs of all nodes provided by the runtime power estimator and the initial state $T(0)$. Recently, many accurate runtime functional block level power estimation methods have been proposed, for example [34, 56]. Most of these methods are based on the performance counters and are relatively accurate and computationally efficient. However, the

estimated powers still contain estimation errors, especially the mean value difference, compared to the real power dissipations of the functional blocks. So the power errors will lead to inaccurate temperature estimation and prediction. Furthermore, the inaccurate thermal model is the second source of errors. Calibration can be performed to improve the accuracy, but the thermal conductivity of materials are functions of temperature, which will introduce unavoidable temperature errors if only linear thermal models are used (as the case for most existing works). In this dissertation, we solve this problem first in Section 5.2.1 by assuming sufficient number of thermal sensors.

The second problem comes from the limited number of thermal sensors on a practical chip. We further propose a correlation based method to address this problem as shown in Section 5.2.1.

The third problem is the computational cost issue if we want to perform the full-chip thermal estimation at runtime. In other words, it is impractical to directly work on (5.1) as the matrix size can be extremely large. Compact modeling technique can be used to reduce the system size, but it cannot be directly applied to the new estimation algorithm because of the newly introduced error compensation. We show in Section 5.2.2 how to optimize the compact modeling method and integrate it into the new thermal estimation method with error compensation.

Finally, a full-chip thermal prediction usually requires high computational cost because of the large thermal node number. We show in Section 5.2.3 we can still predict the full-chip temperature with small overhead with the newly designed full-chip thermal prediction framework.

5.2 Full-chip runtime thermal estimation and prediction method

In this section, we present the new full-chip runtime thermal estimation and prediction method FRETEP.

5.2.1 Full-chip temperature estimation

Generally, a runtime thermal estimator cannot generate accurate results, as briefly introduced in Section 2.3. One type of error is the power estimation error. Usually based on performance counters, the power estimator results may contain errors with non-zero mean and variance. Its statistics may also change at runtime because of the change of the running applications or threads. It has been shown in [44] that most part of the power is concentrated around DC and runtime power average is usually used as the input for thermal estimation. As a result, the mean value of the power estimation error is more important than the variance. The other type of thermal estimation error comes from the thermal model. There are differences in the thermal resistance and capacitance values compared to the real thermal system mainly because of the thermal effect on the thermal resistors and capacitors. We will consider both types of errors and show how accurate full-chip temperature is estimated.

Assume the inaccurate power estimation provided by the power estimator is J and the system matrices G and C are not accurate, the resulting temperature estimation is T. In order to calculate T numerically, we need to discretize (5.1) in time domain. Backward Euler (BE) is used here for illustration. By choosing an appropriate time

step h, BE discretizes (5.1) in time domain as

$$(\frac{C}{h} + G)T(t+h) = \frac{C}{h}T(t) + J(t+h) \tag{5.3}$$

Through inverting $(\frac{C}{h} + G)$ to the right hand side, (5.3) is also written as

$$T(t+h) = (\frac{C}{h} + G)^{-1}(\frac{C}{h}T(t) + J(t+h)) \tag{5.4}$$

Given the initial value $T(0)$ and the input $J(t)$ for all time points, the subsequent temperature $T(t)$ can be calculated iteratively using (5.4).

However, the temperature $T(t)$ calculated from (5.4) is inaccurate due to the inaccurate input J and the inaccurate model G, C. Assume the actual system matrices are $\bar{G} = G + \delta G$ and $\bar{C} = C + \delta C$, and the actual power input is $\bar{J} = J + \delta J$. The real system response $\bar{T}$ can be calculated from

$$(\frac{\bar{C}}{h} + \bar{G})\bar{T}(t+h) = \frac{\bar{C}}{h}\bar{T}(t) + \bar{J}(t+h) \tag{5.5}$$

Error compensation with sufficient thermal sensors

We would like to compensate the power estimation and model errors to generate an accurate temperature estimation.

In the ideal case, assume there are thermal sensors everywhere on the chip, that is, we have the accurate temperature information $\bar{T}(t)$ already. [1] We define the

[1]Note that this ideal case does not exist in reality, where there are only limited number of thermal sensors available. It is introduced here only for the purpose of better presentation and easier understanding of the realistic case shown later.

temperature estimation error δT, power estimation error δJ and model error δM as

$$\delta T(t) := \bar{T}(t) - T(t) \tag{5.6}$$

$$\delta J(t) := \bar{J}(t) - J(t) \tag{5.7}$$

$$\delta M(t) := -(\frac{\delta C}{h} + \delta G)\bar{T}(t) + \frac{\delta C}{h}\bar{T}(t-h) \tag{5.8}$$

Then subtract (5.3) from (5.5) and neglect the second order term, we have

$$(\frac{C}{h} + G)\delta T(t+h) = \frac{C}{h}\delta T(t) + \delta J(t+h) + \delta M(t+h) \tag{5.9}$$

Because of the low-pass filter property of thermal system [61], the temperature estimation error over two successive time steps does not change too much, that is $\delta T(t+h) \approx \delta T(t)$. Therefore, (5.9) becomes

$$(\frac{C}{h} + G)\delta T(t) \approx \frac{C}{h}\delta T(t) + \delta J(t+h) + \delta M(t+h) \tag{5.10}$$

We define the *error compensation term*, determined at time $t+h$, as

$$\epsilon := \delta J(t+h) + \delta M(t+h) \tag{5.11}$$

and from (5.10), the error compensation term ϵ can be approximately solved as

$$\epsilon \approx G\delta T(t) \tag{5.12}$$

We do not express ϵ as a variable of t since it will not be calculated repeatedly at every time point.

After we obtain the error compensation term, the inputs of all the future time

points are updated as

$$J(t + ih) = J(t + ih) + \epsilon \tag{5.13}$$

where $i = 1, 2, \ldots$.

Note the error compensation term ϵ is accurate as long as the power estimation error statistics and the temperature do not change too much. In this case, one compensation is enough for the whole estimation time. If these conditions are not satisfied, we can perform the error compensation process (5.12) and (5.13) periodically or at the time when the temperature errors at the thermal sensors exceed a threshold.

Error compensation with limited number of thermal sensors

We have shown we are able to fully compensate the power estimation error and model error to generate accurate thermal estimation in the ideal case with sufficient number of thermal sensors. However, we cannot put thermal sensors all over the chip in reality. The number of sensors is always limited and as a result, it is impossible to obtain all the elements of $\delta T(t)$ in (5.12). In this subsection, we show how to exploit the power estimator and limited thermal sensor information and approximately recover the full-chip temperature.

Assume there are n_s thermal sensors placed on chip. For convenience, we first perform matrix permutation on (5.1) to group the thermal nodes with thermal sensors together as

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \frac{dT_s(t)}{dt} \\ \frac{dT_u(t)}{dt} \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} T_s(t) \\ T_u(t) \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(t) \tag{5.14}$$

and

$$\begin{bmatrix} J_s(t) \\ J_u(t) \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(t) \tag{5.15}$$

where $T_s(t) \in \mathbb{R}^{n_s}$ represents the temperatures at the nodes where thermal sensors are placed and $T_u(t) \in \mathbb{R}^{n-n_s}$ represents the temperatures at the nodes without thermal sensors.

Accordingly, (5.12) becomes

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} \delta T_s(t) \\ \delta T_u(t) \end{bmatrix} = \begin{bmatrix} \epsilon_s \\ \epsilon_u \end{bmatrix} \tag{5.16}$$

We know the value of δT_s since thermal sensors are placed at these nodes. However, δT_u is unknown due to the absence of thermal sensors. Since there are $2n - n_s$ unknowns in (5.16) with n equations, (5.16) is unsolvable (in the normal sense) unless the number of unknowns is reduced. Fortunately, we are able to reduce the number of unknowns by taking advantage of correlation among different functional blocks in a chip.

Our idea is based on the observation that many functional blocks in a chip are highly correlated in their power consumptions. For instance, when a integer register file is busy, most likely the integer ALU and nearby cache memory will also be busy. As a result, if we properly place the thermal sensors so that more correlated functional blocks are clustered around those thermal sensors, we should be able to have a good guess of the compensation errors around the thermal sensors. Specifically, based on the placement of the n_s thermal sensors, the chip is divided into n_s blocks by combining the correlated functional blocks around each thermal sensor. We call this kind of block as *sensor block*. The compensation errors of different nodes inside one func-

tional block are correlated and the correlation can be characterized. They are usually assumed to be the same or follow a given distribution from characterization. There are also compensation error correlations among different functional blocks inside the same sensor block, mainly because the power consumptions of these functional blocks rely strongly on a small number of common performance parameters. As a result, we introduce a *correlation matrix* $D \in \mathbb{R}^{(n-n_s)\times n_s}$ and represent ϵ_u in terms of ϵ_s as

$$\epsilon_u = D\epsilon_s \tag{5.17}$$

where each column of D shows the correlation of the compensation errors within a specific sensor block.

The details of constructing the correlation matrix D will be shown in Section 5.3, since it is also related to the thermal sensor placement. At this stage, for simplicity considerations, the correlation inside the functional block is temporally established by assuming the compensation errors are identical for all nodes. The relationship among different functional blocks inside the same sensor block is established by keeping the input ratios among different functional blocks invariant before and after the error compensation. We remark that more accurate correlation can be found through statistic characterization on realistic chips, as will be shown later in Section 5.3. In addition, the correlation matrix D will introduce errors if some functional blocks are not fully correlated. In this case, the errors can be minimized by placing thermal sensors properly or increasing the sensor number as shown in the experiments.

We would like to walk through a simple example to illustrate this idea. As shown in Fig. 5.1, there are only three functional blocks on chip. Two thermal sensors (red dashed circle) are placed, one inside FB1 and the other one inside FB2. According to the two thermal sensors, the chip is divided into two sensor blocks: sensor block

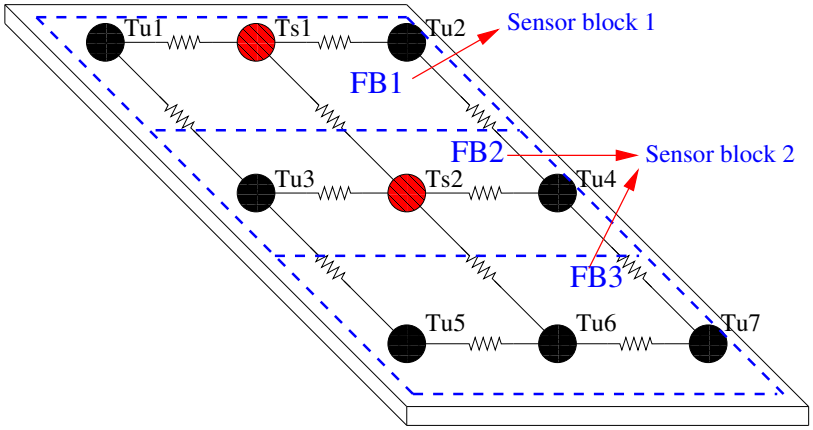


Figure 5.1: A simple example with three functional blocks (FB in short in the figure) to show the D matrix construction. The functional blocks are bounded by dashed lines. The thermal sensor nodes are represented by red dashed circles and the other thermal nodes by black solid circles. The power sources and capacitors are omitted here for simplicity.

1 contains FB1 and sensor block 2 includes FB2 and FB3. In this example, given ϵ_s and ϵ_u corresponding to T_s and T_u shown in Fig. 5.1 as

$$\epsilon_s^T = [\epsilon_{s1} \ \epsilon_{s2}] \tag{5.18}$$

$$\epsilon_u^T = [\epsilon_{u1} \ \epsilon_{u2} \ \epsilon_{u3} \ \epsilon_{u4} \ \epsilon_{u5} \ \epsilon_{u6} \ \epsilon_{u7}] \tag{5.19}$$

the D matrix can be formulated as

$$D^T = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & \frac{J_{fb3}}{J_{fb2}} & \frac{J_{fb3}}{J_{fb2}} & \frac{J_{fb3}}{J_{fb2}} \end{bmatrix} \tag{5.20}$$

where J_{fb2} and J_{fb3} are the power inputs of the nodes inside FB2 and FB3 respectively. Within each functional block, the compensation errors of different nodes are the same. Take FB1 for example, there is $D_{11} = D_{21} = 1$. In order to keep the input ratios invariant among different functional blocks inside the same sensor block, the input errors of different functional blocks are set so that they are proportional to their power

values. Take sensor block 2 (FB1 and FB2) for example, the input error ratio between FB2 and FB3 is J_{fb3}/J_{fb2} as shown in the second column of D. Note although J_{fb2} and J_{fb3} may change at runtime, their ratio J_{fb3}/J_{fb2} remains constant since FB2 and FB3 are correlated. We stress again that the construction of D matrix is not unique and more sophisticated characterization methods can be applied to get more accurate D.

After the introduction of the correlation matrix D, the number of unknowns has been reduced to n. Combined with (5.17), (5.16) is rearranged as

$$\begin{bmatrix} G_{12} & -I_{n_s \times n_s} \\ G_{22} & -D \end{bmatrix} \begin{bmatrix} \delta T_u(t) \\ \epsilon_s \end{bmatrix} = \begin{bmatrix} -G_{11}\delta T_s(t) \\ -G_{21}\delta T_s(t) \end{bmatrix} \tag{5.21}$$

where $I_{n_s \times n_s}$ is an identity matrix with dimension n_s. After ϵ_s is solved from (5.21) and ϵ_u is obtained from (5.17), the error compensation is performed with the permuted form of (5.13).

5.2.2 Compact modeling for fast runtime simulation

Runtime thermal estimation and prediction improve thermal management performance. However, at the same time, they introduce overhead and degrade the system performance. The overhead can be significant especially when the full-chip thermal model is used. Model order reduction (MOR) technique, which reduces the size of large dynamic system models, can be used to reduce the runtime overhead. MOR is introduced and studied intensively in this dissertation, in Section 2.1 and Chapter 3, respectively. However, its integration with the new thermal estimation method is not straightforward. For the completeness of section, we will first review MOR very briefly (interested readers are referred to Section 2.1, Chapter 3 and [5] for a compre-

hensive MOR introduction) and then show in detail how to optimize and integrate MOR into our thermal estimation method.

Assume we need to reduce the original n order model into a smaller k (usually $k \ll n$) order model. The projection based MOR method finds a projection matrix $V \in \mathbb{R}^{n \times k}$ which satisfies the following approximation

$$T(t) \approx V\tilde{T}(t) \tag{5.22}$$

where $\tilde{T} \in \mathbb{R}^k$ is the reduced state. In this case, the reduced model is

$$\tilde{C}\frac{d\tilde{T}(t)}{dt} + \tilde{G}\tilde{T}(t) = \tilde{B}u(t) \tag{5.23}$$

where $\tilde{C} = V^T C V$, $\tilde{G} = V^T G V$ and $\tilde{B} = V^T B$.

In order to integrate MOR into the new thermal estimation method with error compensation, the most important thing is to obtain the reduced error compensation term $\tilde{\epsilon}$ in the reduced model through a similar formulation of (5.21) and (5.17).

First, in order to preserve the structure of (5.14), the structure preserving reduction [25] is used instead of the traditional reduction method. After we get the projection matrix V of (5.14) using a traditional reduction method, the structure preserving reduction method divides V into two blocks according to (5.14) as

$$V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{5.24}$$

and modify V into the structure preserving projection matrix V_{sp}

$$V_{sp} = \begin{bmatrix} \text{orth}(V_1) & 0 \\ 0 & \text{orth}(V_2) \end{bmatrix} \tag{5.25}$$

where orth means V_1 and V_2 are orthonomalized to enhance the numerical performance. In this case, the reduced model has the formulation

$$\begin{bmatrix} \tilde{C}_{11} & \tilde{C}_{12} \\ \tilde{C}_{21} & \tilde{C}_{22} \end{bmatrix} \begin{bmatrix} \frac{d\tilde{T}_s(t)}{dt} \\ \frac{d\tilde{T}_u(t)}{dt} \end{bmatrix} + \begin{bmatrix} \tilde{G}_{11} & \tilde{G}_{12} \\ \tilde{G}_{21} & \tilde{G}_{22} \end{bmatrix} \begin{bmatrix} \tilde{T}_s(t) \\ \tilde{T}_u(t) \end{bmatrix} = \begin{bmatrix} \tilde{B}_1 u(t) \\ \tilde{B}_2 u(t) \end{bmatrix} \tag{5.26}$$

where, take the G and B matrices for example, $\tilde{G}_{11} = V_1^T G_{11} V_1$, $\tilde{G}_{12} = V_1^T G_{12} V_2$, $\tilde{G}_{21} = V_2^T G_{21} V_1$, $\tilde{G}_{22} = V_2^T G_{22} V_2$, $\tilde{B}_1 = V_1^T B_1$, $\tilde{B}_2 = V_2^T B_2$.

In the reduced model, (5.16) becomes

$$\begin{bmatrix} \tilde{G}_{11} & \tilde{G}_{12} \\ \tilde{G}_{21} & \tilde{G}_{22} \end{bmatrix} \begin{bmatrix} \delta\tilde{T}_s(t) \\ \delta\tilde{T}_u(t) \end{bmatrix} = \begin{bmatrix} \tilde{\epsilon}_s \\ \tilde{\epsilon}_u \end{bmatrix} \tag{5.27}$$

where $\tilde{\epsilon}_s = V_1^T \epsilon_s$, $\tilde{\epsilon}_u = V_2^T \epsilon_u$.

There is $\epsilon_u = D\epsilon_s$ in the original model, we also need to find a similar relationship between $\tilde{\epsilon}_u$ and $\tilde{\epsilon}_s$ to reduce the number of unkowns in (5.27). Since there is an approximation

$$\epsilon_s \approx V_1 \tilde{\epsilon}_s = V_1 V_1^T \epsilon_s \tag{5.28}$$

then we have the relationship of $\tilde{\epsilon}_u$ and $\tilde{\epsilon}_s$ as

$$\tilde{\epsilon}_u = V_2^T D \epsilon_s \approx V_2^T D V_1 \tilde{\epsilon}_s \tag{5.29}$$

Generally, this is not a good approximation, because $\epsilon_s = B_1\delta u + \delta M_1$ and the projection matrix V_1 may not contain the subspace spanned by B_1. In order to achieve a good approximation, V_1 in (5.25) is updated by appending B_1 as

$$V_1 = [V_1, B_1] \tag{5.30}$$

and the accurate relationship of $\tilde{\epsilon}_u$ and $\tilde{\epsilon}_s$ is

$$\tilde{\epsilon}_u \approx V_2^T D V_1 \tilde{\epsilon}_s = \tilde{D}\tilde{\epsilon}_s \tag{5.31}$$

where

$$\tilde{D} = V_2^T D V_1 \tag{5.32}$$

Combining (5.27) and (5.31), $\tilde{\epsilon}_s$ is solved from the reduced version of (5.21) as

$$\begin{bmatrix} \tilde{G}_{12} & -I_{k\times k} \\ \tilde{G}_{22} & -\tilde{D} \end{bmatrix} \begin{bmatrix} \delta\tilde{T}_u(t) \\ \tilde{\epsilon}_s \end{bmatrix} = \begin{bmatrix} -\tilde{G}_{11}\delta\tilde{T}_s(t) \\ -\tilde{G}_{21}\delta\tilde{T}_s(t) \end{bmatrix} \tag{5.33}$$

and $\tilde{\epsilon}_u$ is obtained using (5.31).

The reduced model simulation is performed by iteratively using

$$\tilde{T}(t+h) = (\frac{\tilde{C}}{h} + \tilde{G})^{-1}(\frac{\tilde{C}}{h}\tilde{T}(t) + \tilde{J}(t+h)) \tag{5.34}$$

with the error compensation

$$\tilde{J}(t+ih) = \tilde{J}(t+ih) + \tilde{\epsilon} \tag{5.35}$$

where $i = 1, 2, \ldots$. The full-chip temperature T is recovered using (5.22).

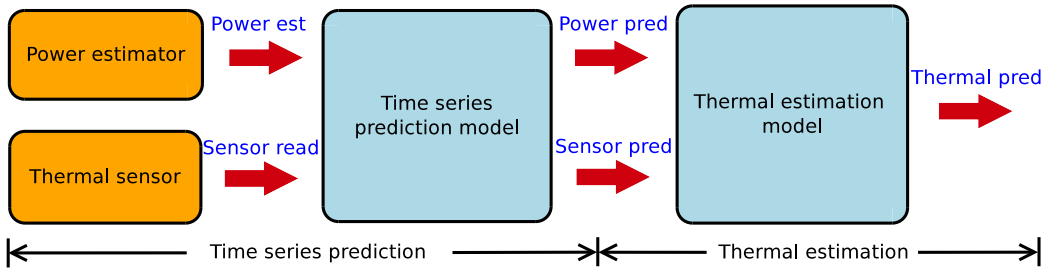


Figure 5.2: Full-chip thermal prediction framework.

5.2.3 Full-chip runtime thermal prediction

Predicting full-chip thermal behaviors at runtime is important for the emerging predictive dynamic thermal management [74, 68, 73]. However, directly performing prediction on the estimated full-chip thermal data is very expensive due to the large number of thermal nodes, since each thermal node contains a thermal time series to be predicted. We have designed a full-chip thermal prediction framework shown in Fig. 5.2 by taking advantage of the small number of functional blocks, the small number of thermal sensors and the high efficiency of the newly introduced full-chip thermal estimation method. In the prediction framework, the power estimations and thermal sensor readings are first predicted using time series prediction models [9]. Then, the future full-chip temperature is calculated using the efficient thermal estimation model with the predicted power and thermal sensor information. Because the number of functional blocks (number of power estimation time series) and the number of thermal sensors (number of thermal sensor temperature reading time series) are both small (usually in dozens) compared to the number of full-chip temperature nodes (usually in thousands) and the new thermal estimation method is very efficient, the new full-chip thermal prediction framework is able to predict the full-chip thermal behaviors with small overhead.

Among many widely used time series prediction methods [9], we choose the autore-

gressive moving average (ARMA) model in this paper because of its long prediction range. An ARMA model consists of two parts, the autoregressive (AR) part and the moving average (MA) part. With p orders of AR part and q orders of MA part, an ARMA(p,q) model is represented as

$$y(t) + \Sigma_{i=1}^{p}(a_i y(t-i)) = e(t) + \Sigma_{i=1}^{q}(c_i e(t-i)) \tag{5.36}$$

where $y(t)$ is the series' value at time t (in our case, the power or thermal sensor temperature at t), $e(t)$ is the white noise term, a_i are the AR parameters and c_i are the MA parameters. Given a time frame of training data, an ARMA(p,q) model can be generated and used to predict the future data. The details of the ARMA model and other time series prediction methods can be found in [9].

5.3 Power-driven thermal sensor placement algorithm

As shown previously in Section 5.2, the construction of the correlation matrix D is highly related to the thermal sensor placement. It is obvious that thermal sensor placement has impact on the estimation accuracy as long as thermal sensor is used in the thermal estimation process. Specifically for FRETEP, note that (5.17) may not work well if the thermal sensor placement is not considered. If ϵ_u and ϵ_s are strongly correlated, then (5.17) will hold for all the time over a wide variety of applications provided that the error correlation matrix D is correctly formed. However, if ϵ_u and ϵ_s are weakly correlated, or even independent, there will be no D exist which can make (5.17) valid. Our objective in this section is to find the optimal thermal sensor placement which will maximize the correlation between ϵ_u and ϵ_s and at the same

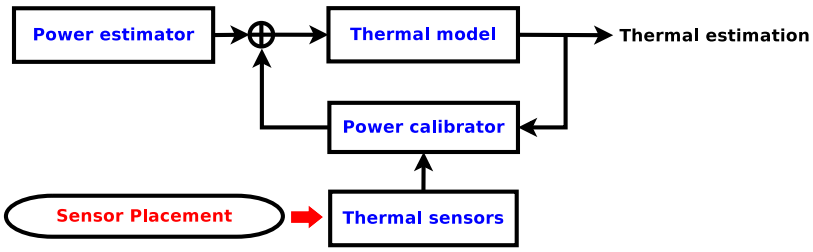
time minimize the number of thermal sensors.

In this section, the power-driven thermal sensor placement algorithm is proposed to assist the thermal estimation and prediction method FRETEP. First, two types of thermal sensor placement methods are introduced in Section 5.3.1. Then, the new power-driven thermal sensor placement algorithm is presented from Section 5.3.2 to Section 5.3.5.

5.3.1 Two types of thermal sensor placement methods

Thermal sensor placement problem for microprocessors has been studied in the past few years and several methods have been proposed [38, 50, 47, 58]. In [38], the maximum temperature difference from the hot spot to a certain spatial point on chip is shown analytically. The authors demonstrated how this error bound will determine the maximum hot spot temperature error from a thermal sensor and how it will guide the thermal sensor placement. In [50], a systematic thermal sensor placement method was proposed: an interpolation method was introduced to recover the full-chip thermal map, and the k-means clustering algorithm was applied to determine the sensor locations according to the hot spot distribution. The main goal of this work is to cover as many hot spots as possible with the limited thermal sensor resources. However it has limited accuracy for the thermal map recovery since the interpolation accuracy is directly related to the thermal sensor density. Another thermal sensor placement work is the spatial thermal spectral-driven method [47]. It employs the fact that the fast temperature change in space leads to high spatial temperature frequency. By placing more thermal sensors at the places with high frequencies, the full-chip temperature can be recovered with higher accuracy. However, in order to get accurate results, a lot of thermal sensors must be placed according to the Nyquist-

(a) The traditional sensor based thermal estimation flow and the corresponding thermal sensor placement.



(b) The proposed power calibration based thermal estimation flow and the corresponding thermal sensor placement.

Figure 5.3: Comparison of the traditional sensor based thermal estimation flow and the proposed power calibration based thermal estimation flow. The thermal sensors play completely different roles in the two approaches.

Shannon sampling theorem. This requires very dense sensor placement at the fast temperature changing areas (high spatial frequency areas). An optimization technique based thermal sensor placement method was introduced in [58]. According to this work, a point of interest (usually hot spot) can be monitored if a thermal sensor is placed within the observing area of this point. The number of thermal sensors is minimized with the constraint that all the hot spots are covered.

Although quite different in details, all the existing thermal sensor placement methods exploit only the chip thermal information and properties. As shown in Fig. 5.3 (a), full-chip thermal estimation is achieved by a full-chip temperature recovery technique with the thermal sensor readings as input. Power information, which is the source of the temperature, remains un-explored for thermal sensor placement. Power

information is particularly relevant because power consumptions of many functional blocks are correlated and this can lead to less number of required sensors or better accuracy given the same number of sensors. In this paper, we propose a new thermal sensor placement method by looking into the information from the power consumption side. As shown in Fig. 5.3 (b), the proposed thermal sensor placement method is based on a different thermal estimation flow with two additional components: a performance counter based runtime power estimator and a thermal estimator with power calibration. The new thermal sensor placement method serves to boost the power calibration efficiency to achieve accurate thermal estimation.

The new thermal sensor placement algorithm mainly includes two steps: first, experiments will be performed on a variety of benchmarks to collect the sample data from measurement and runtime power estimation to form a correlation graph. Then, a correlation clustering algorithm is applied on the correlation graph. The functional blocks are automatically clustered into sensor blocks without pre-specifying the sensor block count (number of thermal sensors). One thermal sensor is placed for each sensor block and the correlation matrix D is determined.

5.3.2 Correlation graph generation

Please note that instead of finding the error relation for each thermal node as (5.17), it is only necessary to find the correlation among functional blocks since the powers of the nodes inside each functional block are extremely correlated and are usually considered to be the same or follow a static distribution. As a result, we only need to find the relation of the total power error

$$\delta U_u = D_p \delta U_s \tag{5.37}$$

where δU_s and δU_u represent the total power error of the functional blocks with thermal sensors and the total power error of the functional blocks without thermal sensors, respectively. The fine-grained power error relation (5.17) can then be easily calculated.

As the first step, the correlation graph is generated for all the functional blocks, using the collected sample data, both from measurement and power estimator simulation.

Assume there are b benchmarks with steady power configurations. First, we run the benchmarks using the power estimator and record the power results

$$\hat{U} = [\hat{U}^1, \hat{U}^2, \ldots, \hat{U}^b] \tag{5.38}$$

where, for example, the ith sample

$$\hat{U}^i = [\hat{u}^i_1, \hat{u}^i_2, \ldots, \hat{u}^i_p]^T \tag{5.39}$$

since there are n_p functional blocks. Next, the benchmarks are run on the test chip until the temperatures reach steady state. The steady state temperature is measured as T. The real power of the chip is reversely calculated as

$$U = [U^1, U^2, \ldots, U^b] \tag{5.40}$$

using the measured temperatures. Note that all these steps should be performed off-line, such that the error can be better controlled and no overhead is introduced at runtime. Please see [48, 16] for details of the reverse power calculation. The errors of

$$corr_{\delta u} = \begin{bmatrix} \frac{E[(\delta u_1-\mu_1)(\delta u_1-\mu_1)]}{\sigma^2_{\delta u_1}} & \frac{E[(\delta u_1-\mu_1)(\delta u_2-\mu_2)]}{\sigma_{\delta u_1}\sigma_{\delta u_2}} & \cdots & \frac{E[(\delta u_1-\mu_1)(\delta u_p-\mu_p)]}{\sigma_{\delta u_1}\sigma_{\delta u_p}} \\ \frac{E[(\delta u_2-\mu_2)(\delta u_1-\mu_1)]}{\sigma_{\delta u_2}\sigma_{\delta u_1}} & \frac{E[(\delta u_2-\mu_2)(\delta u_2-\mu_2)]}{\sigma^2_{\delta u_2}} & \cdots & \frac{E[(\delta u_2-\mu_2)(\delta u_p-\mu_p)]}{\sigma_{\delta u_2}\sigma_{\delta u_p}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{E[(\delta u_p-\mu_p)(\delta u_1-\mu_1)]}{\sigma_{\delta u_p}\sigma_{\delta u_1}} & \frac{E[(\delta u_p-\mu_p)(\delta u_2-\mu_2)]}{\sigma_{\delta u_p}\sigma_{\delta u_2}} & \cdots & \frac{E[(\delta u_p-\mu_p)(\delta u_p-\mu_p)]}{\sigma^2_{\delta u_p}} \end{bmatrix} \tag{5.42}$$

the functional block powers are obtained as

$$\delta U = U - \hat{U} \tag{5.41}$$

The next step is to form a correlation matrix, such that functional blocks with high power error correlations can be identified and put into one sensor block. Using the data samples δU, the correlation matrix is calculated as (5.42) shown on top of the next page, where μ_i is the expected value of δu_i.

By definition, correlation matrix is a symmetric matrix containing the correlation values of each random variable pair. The correlation value is a number between -1 and 1 which reveals the dependence of a random variable pair, where 1 and -1 indicate the two random variables are fully dependent and 0 means totally independence. In our case, we can take the absolute value of the correlation regardless of the sign.

The correlation graph is easily generated by observing the correlation matrix. Assume the correlation matrix of a chip with four functional blocks is

$$corr_{\delta u} = \begin{bmatrix} 1 & 0.9 & 0.4 & 0.8 \\ 0.9 & 1 & 0.3 & 0.8 \\ 0.4 & 0.3 & 1 & 0.7 \\ 0.8 & 0.8 & 0.7 & 1 \end{bmatrix} \tag{5.43}$$

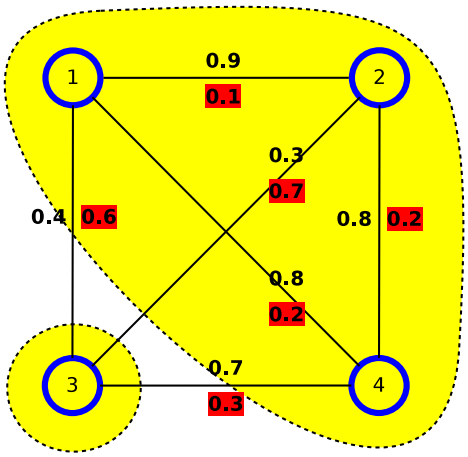


Figure 5.4: A correlation clustering example for a complete undirected weighted graph with four vertices. On each edge e_{ij}, w_{ij}^- is shown as red shaded while w^+ is in normal. The outputed clusters are surrounded by dashed lines filled with yellow color.

The corresponding correlation graph is generated as shown in Fig. 5.4.

5.3.3 Correlation clustering algorithm

The highly correlated functional blocks need to be clustered into the same sensor block in order to enhance the relation in (5.17). There are many clustering algorithms which can be used to cluster the functional blocks into sensor blocks according to the correlation graph, such as k-means method used in [50] (although the objective is quit different). However, most of these clustering algorithms require the number of clusters (in our case, the number of sensors) to be known as *a priori*, which typically is not the case or estimation needs to be performed. As a result, it is better to determine the number of sensors in the clustering algorithm. In this paper, we introduce the correlation clustering algorithm [8] which can automatically determine the number of thermal sensors and their locations based on the generated correlation graph.

In our case, the input of the correlation clustering algorithm is the correlation graph generated previously, which can be expressed as a complete undirected weighted

graph $\mathcal{G} = (V, E)$ (see Fig. 5.4 for example) . There are n_p vertices, where each vertex in the graph represents a functional block. Let v_i denote the ith vertex, e_{ij} denote the edge (v_i, v_j) and the correlation on the edge is c_{ij}. There are two weights on each edge e_{ij}, denoted as w_{ij}^+ and w_{ij}^-, where w_{ij}^+ is the cost of cutting the edge and w_{ij}^- is the cost of keeping the edge. We use the additive weights which are calculated as $w_{ij}^+ = c_{ij}$ and $w_{ij}^- = 1 - c_{ij}$. The output of the correlation clustering algorithm is a new graph $\mathcal{G}'$ with edges $x_{ij} \in \{0, 1\}$, where $x_{ij} = 0$ means vertices v_i and v_j are assigned into the same cluster while $x_{ij} = 1$ means e_{ij} is cut and v_i and v_j are assigned into different clusters. Please note that for the clustering problem, the values of x_{ij} should satisfy the triangle inequality: $x_{ij} + x_{jk} \geq x_{ik}$.

The basic idea of the correlation clustering algorithm is to find the optimal clusters (and the number of clusters) such that the uncorrelations inside each cluster is minimized and at the same time, the correlations among clusters are minimized. We can measure the uncorrelations inside clusters as

$$W_{inside} = \sum_{i,j:i<j} (1 - x_{ij}) w_{ij}^- \tag{5.44}$$

and the correlations among clusters as

$$W_{among} = \sum_{i,j:i<j} x_{ij} w_{ij}^+ \tag{5.45}$$

Please note $1 - x_{ij}$ is 1 if v_i and v_j are in the same cluster while 0 means they are separated. The total cost can be written as $W_{inside} + W_{among}$ and the formulation of

the optimization problem is

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j:i<j} (1 - x_{ij}) w_{ij}^{-} + x_{ij} w_{ij}^{+} \\ \text{subject to} \quad & x_{ij} \in \{0, 1\} \\ & x_{ij} + x_{jk} \geq x_{ik} \end{aligned} \tag{5.46}$$

As an example, consider the graph shown in Fig. 5.4. The weights w_{ij}^{-} and w_{ij}^{+} are shown on each edge. The clusters determined by the correlation clustering algorithm are shown surrounded by dashed lines filled with yellow color. Obviously, v_1, v_2 and v_4 are highly correlated to each other and are assigned into the same cluster. For v_3, although it is relatively correlated to v_4, it is uncorrelated to v_1 and v_2. As a result, it is assigned into another cluster. These two clusters minimize the cost function in (5.46) and 2 is automatically determined as the number of clusters.

5.3.4 Locate thermal sensors

We have clustered functional blocks into sensor blocks, then one thermal sensor has to be placed for each sensor block. We call the functional block with a thermal sensor located as the *sensor functional block*. Although several functional blocks are clustered into the same sensor block by power error correlations, their physical locations may be distributed all over the chip. We decide the sensor functional block as the one closest to the centroid of the sensor block. The thermal sensor is then put on the center of this functional sensor block. Please note that thermal sensors cannot always be put at the specified location because of the design considerations and limitations [58]. In this case, thermal sensor can be fine tuned within the sensor functional block. If the design constraint is not satisfied, the functional block which

is the second closest to the sensor block centroid can be used as the sensor functional block instead.

5.3.5 Error correlation matrix generation

In this subsection, we will present how to generate the error correlation matrix D. As introduced in 5.3.2, we have to form D_p first, then generate D. We use the linear regression method to find the relations among the functional blocks within each sensor block. Assume the ith functional block is associated with the jth sensor functional block (which means they are clustered into the same sensor block and the jth functional block has a thermal sensor placed), the relation

$$\delta u_j = a_j \delta u_i \tag{5.47}$$

is found using the sample data information $[\delta u_i^1, \delta u_i^2, \ldots, \delta u_i^b]$ and $[\delta u_j^1, \delta u_j^2, \ldots, \delta u_j^b]$. With (5.47) for each functional block without thermal sensors, i.e. $j = 1, 2, \ldots, n-n_s$, D_p in (5.37) is populated with a_j and the correlation matrix D in (5.17) is derived subsequently.

5.4 Algorithm flow and practical considerations

The algorithm flow of FRETEP is shown in Fig. 5.5.

The new power-driven thermal sensor placement flow is summarized in Fig. 5.6.

The practical implementation of FRETEP contains off-line part and on-line part. The off-line part includes the modeling and calibration of the full thermal model (5.1), thermal sensor placement, structure preserving reduction, and pre-factorization of $(\frac{\tilde{C}}{h} + \tilde{G})$ in the reduced model simulation (5.34). The on-line part contains tem-

FRETEP: Full-chip Runtime Error-Tolerant Thermal Estimation and Prediction method

▷ *Preparation*

1. Build and calibrate thermal model (5.1).
2. Place thermal sensors using the power-driven thermal sensor placement algorithm and do the matrix permutation.
3. Perform structure preserving reduction on (5.1) and get V_1 and V_2.
4. Update V_1 by appending B_1 as $V_1 = [V_1, B_1]$, form the projection matrix V_{sp} as (5.25) to generate the reduced system (5.26) and $\tilde{D}$.

▷ *Thermal estimation*

1. Perform *Preparation* 1-5.
2. **while** 1:
3. Calculate $\tilde{T}$ one time step forward using (5.34).
4. **if** temperature senor error > tolerance:
5. Calculate ϵ_u and ϵ_s using (5.33) and (5.31).
6. Update input using (5.35).
7. **end if**
8. Recover full-chip temperature T using (5.22).
9. **end while**

▷ *Thermal prediction*

1. Perform *Preparation* 1-5.
2. Build time series prediction models using the past power estimator and thermal sensor information.
3. Perform *Thermal estimation* 2-9 with predicted power estimator and thermal sensor information from the time series prediction models.

Figure 5.5: FRETEP algorithm flow.

POWER-DRIVEN THERMAL SENSOR PLACEMENT FLOW

1. Collect power error samples using runtime power estimations and thermal map measurements.
2. Build the correlation graph.
3. Cluster functional blocks into sensor blocks by applying correlation clustering algorithm on the correlation graph. The number of clusters is determined automatically.
4. Determine the sensor functional block for each sensor block.
5. Place one thermal sensor at the center of each sensor functional block.
6. Permute thermal model matrices G, C, B and generate the error correlation matrix D according to the thermal sensor placement.

Figure 5.6: The new power-driven thermal sensor placement algorithm flow.

perature calculation using the pre-factorized $(\frac{\tilde{C}}{h} + \tilde{G})$ and error compensation. The thermal prediction contains extra on-line parts: form the ARMA model and predict temperature using the ARMA model.

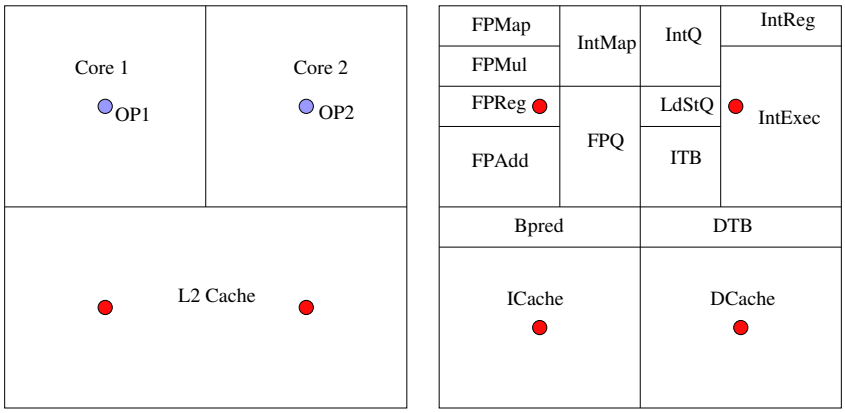
5.5 Experimental results

The proposed method FRETEP with the power-driven thermal sensor placement algorithm are implemented in Matlab. All the results are collected on a Linux server with 3.0Ghz Intel Quadcore Xeon CPU and 16GB memory. In order to validate the new thermal estimation and prediction method, we build a dual-core processor with a shared L2 cache which is shown in Fig. 5.7 (a). The size of the processor is $10mm \times 10mm \times 0.7mm$. The core architecture shown in Fig. 5.7 (b) is similar to the Alpha ev6 processor. The power information is obtained using the power estimator Wattch [11] by running the standard SPEC benchmarks [30]. One core of the dual-

core processor is assumed to be active and the other one is assumed to be idle, they can be switched when the temperature on one core is too high. The original order of the thermal model is 3200 and the reduced model, which is used in FRETEP, has the order of 106. The simulation time step h is chosen to be $0.1s$ to balance the speed and accuracy. The thermal estimator performs the error compensation periodically every 5 samples, that is, every 0.5 seconds.

In order to validate FRETEP, there are 10 thermal sensors placed on chip in total, 4 for each core and 2 for the L2 cache as shown in Fig. 5.7. We also set two observing points (OP1 and OP2) which are far away from any thermal sensors in order to demonstrate the transient estimation and prediction results. The power estimations given by the power estimator is modeled with up to 20% mean value error. For example, the actual power and the estimated power of L2 cache of the dual-core processor by running bzip2 benchmark is shown in Fig. 5.8. The system model error is set to be 10%. Besides FRETEP, Kalman filter based method [58] using the same reduced model as FRETEP is also implemented for comparison. For the sake of consistency, we use bzip2 benchmark to show all the transient plots and thermal map plots. All the other benchmarks show similar results and are summarized in Table 5.1.

For the validation of the power-driven thermal sensor placement algorithm, a runtime power estimator has been built by using two most important performance counts for each functional block power, where the parameter of each performance count is obtained through a linear regressor with samples from several benchmarks similar to [70, 56]. For the correlation clustering, we use the opensource software for [23] available at [1].



(a) The dual-core microprocessor architecture.

(b) The architecture for each core composed of functional blocks.

Figure 5.7: The dual-core microprocessor architecture, with two cores and one shared L2 cache. 10 thermal sensors (red solid circle) are placed on chip, 2 on the L2 cache and 4 on each core. Two observing points (light blue circle) OP1 and OP2 are set in order to show the transient thermal estimation and prediction results.

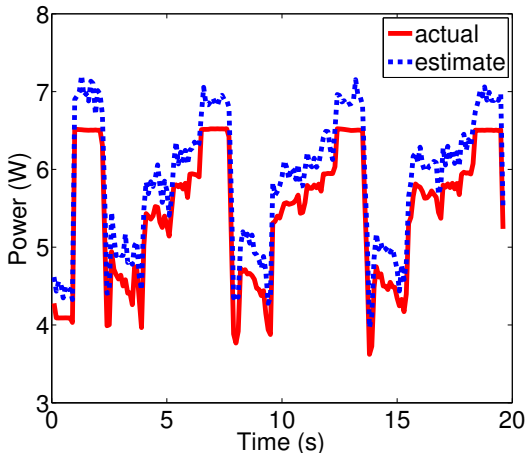
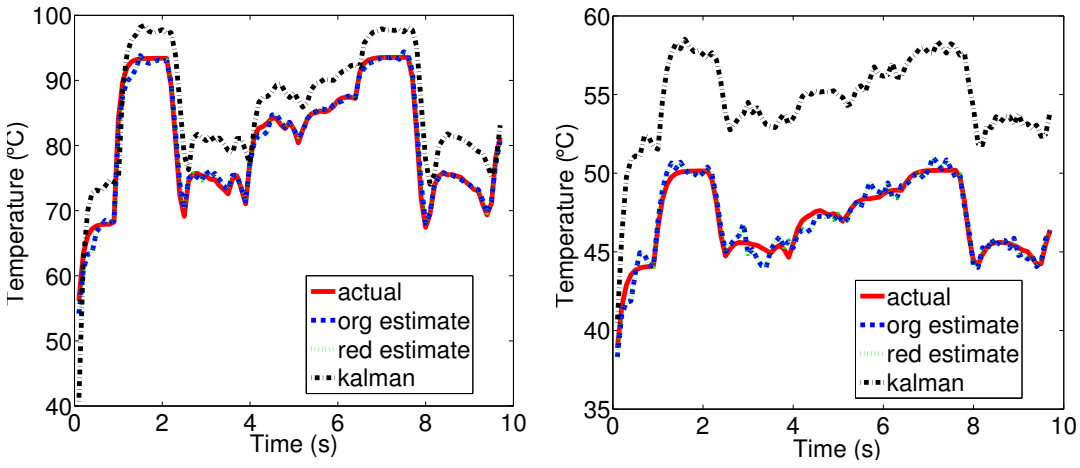


Figure 5.8: The actual power and the estimated power of L2 cache running bzip2 benchmark. The estimated power has significant mean value difference compared to the actual power.

5.5.1 Full-chip thermal estimation results

First, we give the full-chip thermal estimation results of FRETEP using the first half of the runtime power estimator information ($0 \sim 10s$). The new thermal estimator



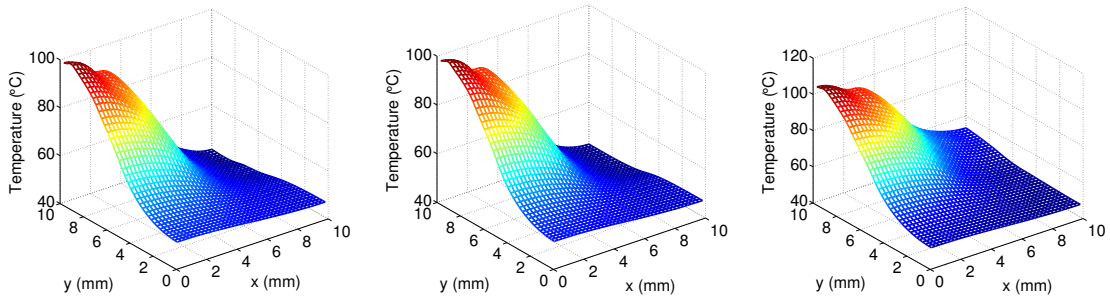
(a) Transient thermal estimation results for OP1.

(b) Transient thermal estimation results for OP2.

Figure 5.9: Transient thermal estimation results of bzip2 benchmark, where *org* represents the new method with original model before MOR, *red* represents the new method with reduced model after MOR and Kalman represents the Kalman filter based method [58].

should give relatively accurate results even though the power estimation is not accurate as shown in Fig. 5.8. The transient estimation results for the two observing points, OP1 and OP2, are presented in Fig. 5.9. The error of the new thermal estimation method is within 1°C and there is no observable difference between the reduced model results and the original model results which means model order reduction is accurate enough. The Kalman filter based method has the error up to 5°C mainly because of the power estimation mean value error and the thermal model error.

In order to see the accuracy of the full-chip thermal estimation, we take a full-chip thermal map snapshot at $5s$. The results are shown in Fig. 5.10. It is clear that the new thermal estimation method has a thermal map very similar to the actual one while Kalman filter based method generates a thermal map with significant errors.



(a) Actual thermal map.

(b) Estimated thermal map by the new thermal estimation method.

(c) Estimated thermal map by the Kalman filter based method.

Figure 5.10: Full-chip thermal map comparison of bzip2 benchmark at 5s.

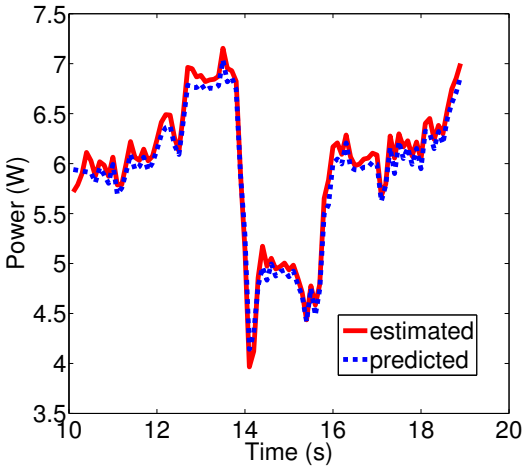
5.5.2 Full-chip thermal prediction results

Next, we demonstrate the prediction ability of FRETEP. The power estimator and thermal sensor readings are predicted by an ARMA(5, 0) model. The ARMA model is trained using 5 seconds of the past data and the prediction is performed 1 second into the future. Fig. 5.11 shows the prediction values of the power estimations for the L2 cache and the thermal readings from the thermal sensor on FPReg of core 1.

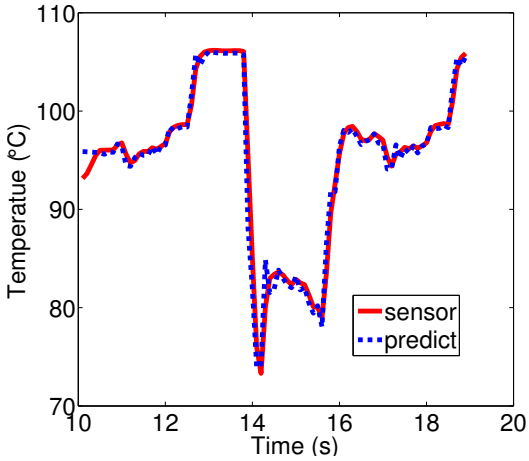
The accuracy of the predicted transient thermal result is shown in Fig. 5.12. Although the prediction errors are larger than the ones because of the errors introduced by ARMA prediction, the average error is still very small.

The comparison of the predicted thermal map snapshot at 15s and the actual thermal map is shown in Fig. 5.13. FRETEP successfully predicted the full-chip temperature.

The runtime and accuracy comparison of FRETEP on a variety of SPEC benchmarks are shown in Table 5.1. In the table, *avg err* is the absolute error averaged on both space and time with the unit °C, *KF* means the Kalman filter based method, *X org* and *X kf* denote the speedup against the original model and the Kalman filter

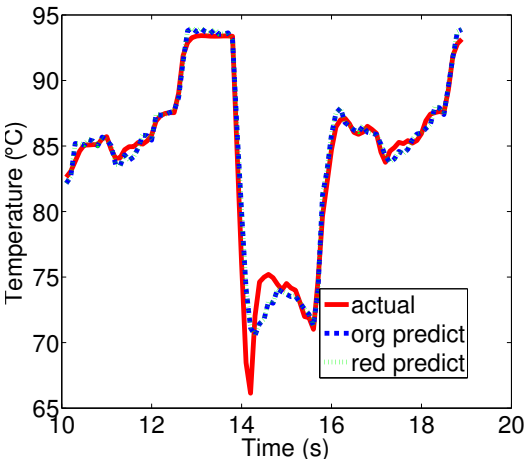


(a) Power estimator data prediction for L2 cache.

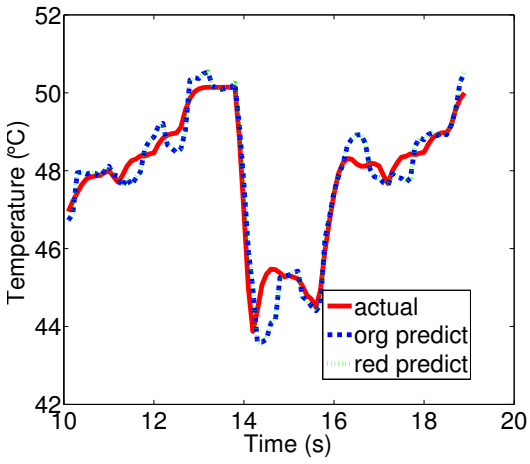


(b) Prediction of the readings from the thermal sensor on FPReg of core 1.

Figure 5.11: Power estimator and thermal sensor data prediction of the bzip2 benchmark.

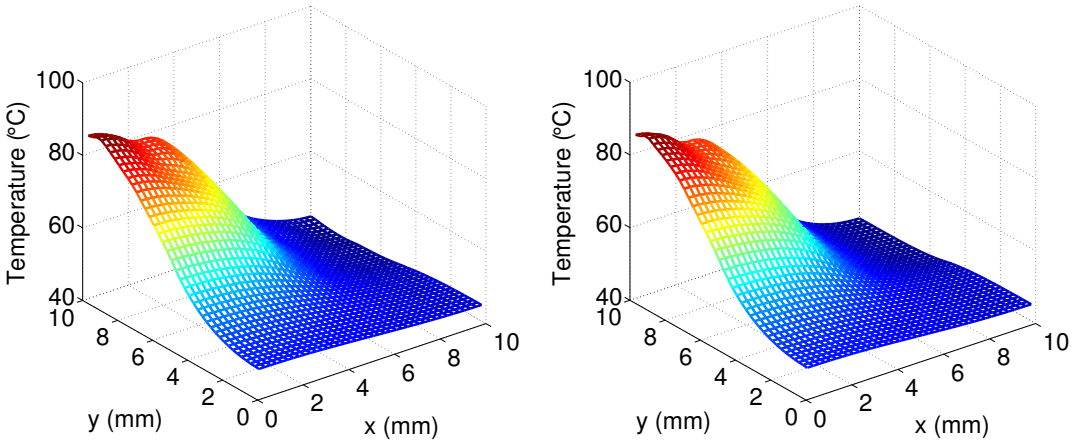


(a) Transient thermal prediction results for OP1.



(b) Transient thermal prediction results for OP2.

Figure 5.12: Transient thermal prediction results of bzip2 benchmark where *org* represents the new method with original model before MOR and *red* represents the new method with reduced model after MOR.



(a) Actual thermal map.

(b) Predicted thermal map by the new thermal prediction method.

Figure 5.13: Accuracy of the predicted full-chip thermal map of bzip2 benchmark at $15s$.

Benchmark	KF based estimation		FRETEP estimation					FRETEP prediction		
	avg err	sim time	avg err	org time	red time	X org	X kf	avg err	arma time	sim time
bzip2	3.8	0.018	0.48	0.04	0.0011	37	18	0.52	0.026	0.0011
gzip	3.9	0.006	0.36	0.14	0.0017	80	3	0.92	0.008	0.0016
mcf	3.4	0.016	0.43	0.05	0.0012	46	14	1.51	0.019	0.0011
mgrid	3.9	0.031	0.46	0.04	0.0013	34	31	0.73	0.032	0.0008
swim	4.1	0.021	0.41	0.05	0.0011	45	19	1.08	0.027	0.0013
galgel	4.5	0.008	0.37	0.11	0.0014	72	6	0.72	0.012	0.0012

Table 5.1: Runtime and accuracy comparison of FRETEP on SPEC benchmarks.

method with the reduced model, respectively. To be fair, all the simulation times are measured as the time spent to estimate/predict 1 second thermal behavior, with the unit s. *arma time* includes the ARMA model building time and time series prediction time. For all the benchmarks, FRETEP has better accuracy than the Kalman based method with the estimation error within 0.5°C and prediction error within 1.5°C. It also introduces very low overhead, only around 0.002 seconds for 1 second estimation and 0.03 seconds for each second of prediction. It is faster than the Kalman filter based method using the same reduced model up to $20X$.

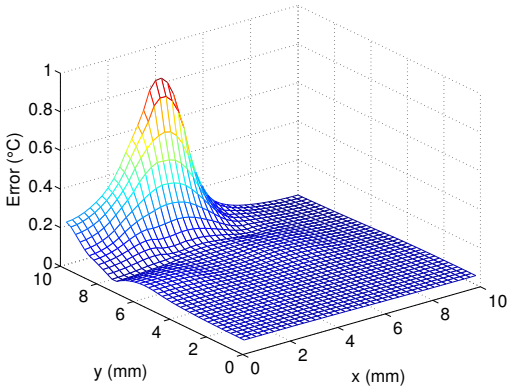
sensor #	avg est err	avg pred err
6	0.98	0.97
8	0.58	0.61
10	0.48	0.52
14	0.42	0.44
18	0.35	0.43

Table 5.2: Thermal sensor number effects on the estimation and prediction accuracies running bzip2.

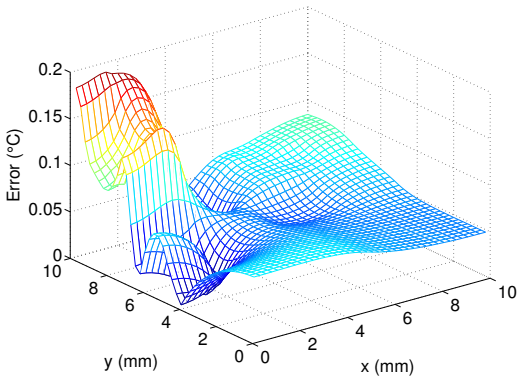
In order to study the impacts of different numbers of thermal sensor on the estimation and prediction performances, we change the sensor number from 6 to 18 and collect the average absolute errors (in °C) in Table 5.2. Not surprisingly, the results show increasing the number of thermal sensors reduces both the estimation and prediction errors with extra die area overhead for sensor placement.

5.5.3 The effect of the power-driven thermal sensor placement algorithm

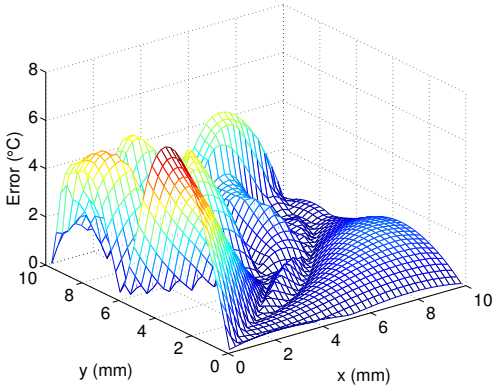
We first compare the new thermal sensor placement algorithm with the simple uniform placement method. Because in the new thermal sensor placement algorithm, the number of thermal sensors is determined automatically, we force the other placement methods to have the same number of thermal sensors for a fair comparison. In order to validate the new placement algorithm, the uniform thermal sensor placement has the same thermal estimation flow (shown in Fig. 5.3 (b)) as the new placement method. The correlation clustering algorithm inside the new thermal sensor placement method gives 6 clusters, which means there are totally 6 sensor blocks (thus 6 thermal sensors). The sensor placement information is summarized in Table 5.3, where *SB idx* is for sensor block index and *Functional blocks* is for list of functional blocks in each sensor block and *Sensor FB* is the functional block on which the sensor is located. Since the



(a) Error plot of the uniform thermal sensor placement.



(b) Error plot of the new thermal sensor placement.



(c) Error plot of the k-means sensor placement with interpolation thermal map recovering.

Figure 5.14: Error snapshot plot with the bzip2 benchmark at 15s. For both cases, 6 thermal sensors are placed on chip.

Method	SB idx	Functional blocks	Sensor FB
Uniform	1	L2 Cache	L2 Cache
	2	FPReg, FPQ, FPMap, IntMap FPMul, FPAdd, Bpred, Icache	FPAdd
	3	DCache, DTB, ITB, LdStQ, IntQ, IntReg, IntExec	IntExec
New	1	L2 Cache	L2 Cache
	2	ICache, DCache, DTB, ITB, FPReg, FPQ, LdStQ, IntQ, IntReg	FPReg
	3	Bpred, FPMul, FPAdd, FPMap, IntMap, IntExec	FPAdd

Table 5.3: The sensor placement information for uniform sensor placement and the new sensor placement method.

Bench	6 sensors						10 sensors						14 sensors					
	Uniform		New		k-means		Uniform		New		k-means		Uniform		New		k-means	
	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max
bzip2	0.08	0.83	0.06	0.19	2.0	7.2	0.07	0.85	0.05	0.20	1.4	7.7	0.06	0.33	0.05	0.12	1.3	7.7
gzip	0.15	0.42	0.12	0.37	1.9	8.0	0.11	0.37	0.08	0.25	1.5	8.1	0.09	0.34	0.09	0.26	1.2	7.0
mcf	0.07	0.65	0.06	0.41	1.0	4.7	0.08	0.52	0.04	0.26	0.91	4.7	0.09	0.47	0.02	0.10	0.8	3.9
mgrid	0.04	0.66	0.04	0.22	2.3	10.4	0.04	0.48	0.03	0.16	1.7	10.4	0.03	0.31	0.02	0.12	1.5	7.6
swim	0.22	2.7	0.17	1.2	1.3	8.1	0.16	1.8	0.15	0.86	1.3	8.2	0.15	0.93	0.15	1.1	1.3	6.5
galgel	0.08	1.3	0.06	0.28	1.0	5.2	0.07	0.93	0.04	0.18	0.94	5.6	0.05	0.66	0.03	0.15	0.93	6.2

Table 5.4: Accuracy comparison of the new thermal sensor placement algorithm with the uniform and the k-means thermal sensor placement methods.

sensor placement is the same for the two cores, only information of one core is shown (so only 3 sensor blocks are shown).

The error snapshot plots with the *bzip2* benchmark at 15s for the uniform and the new method are shown in Fig. 5.14 (a) and (b). Thanks to the runtime power estimator and the power calibrator, both the uniform and the new placement method generate quite good thermal map estimations with only 6 thermal sensors. However, the uniform thermal sensor placement does not consider the power error correlation at the sensor placement stage, such that the functional blocks inside each sensor block are not fully correlated. As a result, it can be seen in Fig. 5.14 (a), the error can be relatively large at some positions. Since the new thermal sensor placement

algorithm automatically groups the functional blocks with respective to power error correlations, the power calibrator works more efficiently in this case and generates the power map with much less error as shown in Fig. 5.14.

The new method is then compared against the existing k-means based thermal sensor placement method with interpolation thermal map recovery [50] on the same benchmark. The new thermal sensor placement method takes advantage of the runtime power estimator which is additional information compared to the k-means placement method. As a result, although quite efficient indeed, the k-means method still has larger error compared to the new method because it is extremely hard (perhaps impossible) to accurately recover the full-chip temperature with *only* temperature information from 6 thermal sensors.

More accuracy comparison results on the other benchmarks are summarized in Table 5.4. We have also increased the number of thermal sensors by multiplying the correlation matrix (5.42) with a constant term (although this trick is not a part of the new algorithm), where the case of 10 thermal sensors is achieved by multiplying 0.65 and the case of 14 thermal sensors is reached by multiplying 0.6. We can see more thermal sensors actually improve the accuracy, but the number of sensors automatically determined by the new algorithm is already fairly enough.

5.6 Summary

In this chapter, we have presented a new method FRETEP which accurately estimates and predicts the full-chip temperature at runtime under more practical conditions where we have inaccurate thermal models, less accurate power inputs and limited number of on-chip physical thermal sensors. FRETEP employs a number of new techniques to address the practical conditions problem. The proposed techniques

enable FRETEP to estimate the temperature everywhere on the chip and detect the hot spots so that on-line thermal regulator can act properly to reduce the thermal gradients across the chip. In addition, a power-driven thermal sensor placement method has been proposed to further enhance the accuracy of FRETEP. Experimental results show FRETEP accurately estimates and predicts the full-chip thermal behavior with very low overhead introduced and compares very favorably with the Kalman filter based approach on standard SPEC benchmarks.

Chapter 6

Conclusion

This chapter concludes the dissertation by summarizing the research contributions for electronic and thermal modeling and analysis of nanometer integrated and packaged systems. The contributions of this dissertation covers three steps in the electronic and thermal analysis flow: general model order reduction, thermal modeling and thermal simulation. WBMOR has increased the wideband accuracy in general VLSI electronic and thermal compact modeling. ThermComp has enhanced the flexibility and reusability in thermal modeling. Finally, fast and accurate thermal estimation and prediction are achieved by the proposed algorithm FRETEP.

In Chapter 3, we have proposed a novel model order reduction method, WBMOR, for wide frequency band modeling of interconnect circuits. WBMOR explicitly computes the exact residual errors to guide the sampling process in an adaptive way. We showed that by sampling along the imaginary axis and performing a new complex-valued sampling based reduction, the reduced model will match exactly with the original model at the sample points. Theoretically, the proposed method can achieve the error bound over a given frequency range with sufficient sampling. Practically, we designed an adaptive scheme to help designers choose the best order of the reduced

model for the given frequency range and error bound. We compared several sampling schemes such as Monte Carlo, logarithmic, and recently proposed re-sampling methods. Experimental results on a number of RLC circuits show that WBMOR is much more efficient than all the other sampling methods including the recently proposed re-sampling and ARMS schemes with the same reduction orders. Compared with the real-valued sampling methods, the complex-valued sampling method is more accurate for the same computational costs.

Chapter 4 presented a new compact thermal modeling technique for architecture level thermal design space exploration for multi-core microprocessors. The new approach, called ThermComp, builds the models from the first principles by the finite difference method for each basic module and reduces the model complexity by the sampling based model order reduction technique. We have applied a new two-grid discretization scheme, where a uniform global coarse grid is used for all the boundary grids and a fine grid is used for the internal grids for each module. This discretization scheme makes the thermal modules easily composable for building large thermal systems. The coarse global grid allows significant reduction of the number of the boundary ports and enables the efficient reduction of thermal modules possible. Experimental results on a number of multi-core microprocessor architectures show that the new approach can easily build accurate thermal circuits from the composable thermal models. The reduced composite models lead to orders of magnitude speedup over the standard finite difference models and are much faster than the HotSpot method with similar accuracy.

In chapter 5, we have presented a new method FRETEP which accurately estimates and predicts the full-chip temperature at runtime under more practical conditions where we have inaccurate thermal models, less accurate power inputs and limited number of on-chip physical thermal sensors. FRETEP employs a number of

new techniques to address the practical conditions problem. The proposed techniques enable FRETEP to estimate the temperature everywhere in the chip and detect the hot spots so that on-line thermal regulator can act properly to reduce the thermal gradients across the chip. Experimental results show FRETEP accurately estimates and predicts the full-chip thermal behavior with very low overhead introduced and compares very favorably with the Kalman filter based approach on standard SPEC benchmarks.

Bibliography

[1] Correlation clustering system. `http://www.cs.brown.edu/~melsner/`.

[2] UiMOR – UC Riverside Model Order Reduction Tool Suite. http://www.ee.ucr.edu/ stan/project/uimor/uimor_main.htm.

[3] International technology roadmap for semiconductors(ITRS), 2009 update. http://public.itrs.net.

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

[5] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. The Society for Industrial and Applied Mathematics (SIAM), 2005.

[6] A. Augustin, B. Maj, and A. Kostka. A structure oriented compact thermal model for multiple heat source ASICs. *Mircroelectronics Journal*, 36(8):700–704, August 2005.

[7] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, 2000.

[8] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning Journal*, 56(1-3):89–113, 2004.

[9] George Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice-Hall, 3rd edition, 1994.

[10] 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.

[11] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. Int. Symp. on Computer Architecture (ISCA)*, pages 83–94, 2000.

[12] Youns Chahlaoui and Paul Van Dooren. A collection of benchmark examples for model reduction of linear time invariant dynamical systems. SLICOT Working Note 2002-2, February 2002.

[13] Y.-K. Cheng, C.-H. Tsai, C.-C. Teng, and S.-M. Kang. *Electrothermal Analysis of VLSI Systems*. Kluwer Academic Publishers, 2000.

[14] E. Chiprout and M. S. Nakhla. Analysis of interconnect networks using complex frequency hopping. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(2):186–200, Feb. 1995.

[15] Filip Christiaens, Bart Vandevelde, Eric Beyne, Robert Mertens, and Jan Berghmans. A generic methodology for deriving compact dynamic thermal models, applied to the PSGA package. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part A*, 21(4):565–576, December 1998.

[16] R. Cochran, A. Nowroz, and S. Reda. Post-silicon power characterization using thermal infrared emissions. In *Proc. Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 331–336, 2010.

[17] Ryan Cochran and Sherief Reda. Spectral techniques for high-resolution thermal characterization with limited sensor data. In *Proc. Design Automation Conf. (DAC)*, pages 478–483, 2009.

[18] Ryan Cochran and Sherief Reda. Consistent runtime thermal prediction and control through workload phase detection. In *Proc. Design Automation Conf. (DAC)*, pages 62–67, 2010.

[19] Lorenzo Codecasa, Dario D'Amore, and Paolo Maffezzoni. An arnoldi based thermal network reduction method for electro-thermal analysis. *IEEE Tran. on Components and Pacakaging Technologies*, 26(1):186–192, March 2003.

[20] Lorenzo Codecasa, Dario D'Amore, and Paolo Maffezzoni. Boundary condition independent compact models of dynamic thermal networks with many heat sources. In *Thermal and Thermomechanical Phenomena in Electronic Systems*, pages 685–689, 2006.

[21] Ayse Kivilcim Coskun, Tajana Simunic Rosing, and Kenny C. Gross. Utilizing predictors for efficient thermal management in multiprocessor SoCs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 28:1503–1516, October 2009.

[22] James Donald and Margaret Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *Proceedings of the 33rd annual international symposium on Computer Architecture*, ISCA '06, pages 78–88, Washington, DC, USA, 2006. IEEE Computer Society.

[23] Micha Elsner and Warren Schudy. Bounding and comparing methods for correlation clustering beyond ILP. In *Workshop on Integer Linear Programming for Natural Language Processing*, June 2009.

[24] 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.

[25] R. W. Freund. SPRIM: structure-preserving reduced-order interconnect macromodeling. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 80–87, 2004.

[26] Aurél Galántai. *Projectors and Projection Methods*. Kluwer Academic Publishers, 2004.

[27] Y. C. Gerstenmaier and G. Wachutka. Rigorous model and network for transient thermal problems. *Mircroelectronics Journal*, 33:719–725, September 2002.

[28] 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.

[29] Stephen Gunther, Frank Binns, Douglas Carmean, and Jonathan Hall. Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal*, 5, February 2001.

[30] John L. Henning. SPEC CPU 2000: Measuring CPU performance in the new millennium. *IEEE computer*, 1(7):28–35, July 2000.

[31] 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.

[32] Wei Huang, Shougata Ghosh, Siva Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R. Stan. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, May 2006.

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

[34] Canturk Isci and Margaret Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proceedings of MICRO*, 2003.

[35] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University, 3 edition, 1996.

[36] M. Kamon, F. Wang, and J. White. Generating nearly optimally compact models from Krylov-subspace based reduced-order models. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 47(4):239–248, 2000.

[37] C. Lasance, H Vinke, H Rosten, and K.-L. Weiner. A novel approach for the thermal characterization of electronic parts. In *Proceedings of the IEEE 11th Annual Semiconductor Thermal Measurement and Management Symposium*, pages 1–9, 1995.

[38] K. Lee, K. Skadron, and W. Huang. Analytical model for sensor placement on microprocessors. In *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, pages 24–27, 2005.

[39] D. Li, S. X-.D. Tan, E. H. Pacheco, and M. Tirumala. Parameterized transient thermal behavioral modeling for chip multiprocessors. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 611–617, Nov. 2008.

[40] 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.

[41] 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.

[42] Guangdeng Liao, Xia Zhu, and Laxmi N. Bhuyan. A new server I/O architecture for high speed networks. In *Proc. IEEE Int. Symp. on High-Performance Computer Architecture (HPCA)*, 2011.

[43] Guangdeng Liao, Xia Zhu, Steen Larsen, Laxmi N. Bhuyan, and Ram Huggahalli. Understanding power efficiency of TCP/IP packet processing over 10GbE. In *18th Symposium on High-Performance Interconnects*, pages 32–39, 2010.

[44] 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.

[45] Jieyi Long, Seda Ogrenci Memik, Gokhan Memik, and Rajarshi Mukherjee. Thermal monitoring mechanisms for chip multiprocessors. *ACM Transactions on Architecture and Code Optimization*, 5(2):9:1–9:33, August 2008.

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

[47] A. Nowroz, R. Cochran, and S. Reda. Thermal monitoring of real processors: Techniques for sensor allocation and full characterization. In *Proc. Design Automation Conf. (DAC)*, 2010.

[48] A. Nowroz, G. Woods, and S. Reda. Improved post-silicon power modeling using AC lock-in techniques. In *Proc. Design Automation Conf. (DAC)*, 2011.

[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] Seda Ogrenci Memik, Rajarshi Mukherjee, Min Ni, and Jieyi Long. Optimizing thermal sensor allocation for microprocessors. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):516–527, March 2008.

[51] M. Necati. Ozisik. *Finite Difference Methods in Heat Transfer*. Taylor & Francis, Inc., 1994.

[52] Heinz Pape, Dirk Schweitzer, John H. J. Janssen, Arianna Morelli, and Claudio M. Villa. Thermal transient modeling and experimental validation in the european project PROFIT. *IEEE Tran. on Components and Pacakaging Technologies*, 27(3):530–538, September 2004.

[53] M. Pedram and S. Nazarian. Thermal modeling, analysis, and management in VLSI circuits: Principles and methods. *Proc. of the IEEE*, 94(8):1487–1501, Aug. 2006.

[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. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(1):43– 55, 2005.

[56] Michael D. Powell, Arijit Biswas, Joel S. Emer, Shubhendu S. Mukherjee, Basit R. Sheikh, and Shrirang Yardi. CAMP: A technique to estimate per-structure power at run-time using a few simple parameters. In *Proc. IEEE Int. Symp. on High-Performance Computer Architecture (HPCA)*, pages 289–300, 2009.

[57] M. Rencz, G. Farkas, V. Székely, A. Poppe, and B. Courtois. Boundary condition independent dynamic compact models of packages and heat sinks from thermal transient measurements. In *Proceedings of the 5th Electronics Packaging Technology Conference*, pages 479–484, 2003.

[58] Shervin Sharifi and Tajana Simunic Rosing. Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 29(10):1586–1599, Oct. 2010.

[59] L. M. Silveira and J. R. Phillips. Resampling plans for sample point selection in multipoint model-order reduction. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(12):2775–2783, 2006.

[60] 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.

[61] Kevin Skadron, Mircea R. Stan, Wei Huang, Siva Velusamy, Karthik Sankaranarayanan, and David Tarjan. Temperature-aware microarchitecture. In *Proc. Int. Symp. on Computer Architecture (ISCA)*, pages 2–13, 2003.

[62] S. X-.D. Tan, H. Wang, and B. Yan. UiMOR – UC Riverside model order reduction tool for post-layout wideband interconnect modeling. In *International Conf. Solid State and Integrated Circuit Technology (ICSICT)*, Oct. 2010.

[63] L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra*. The Society for Industrial and Applied Mathematics (SIAM), 1997.

[64] D. Vasilyev and J. White. A more reliable reduction algorithm for behavioral model extraction. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 813–820, 2005.

[65] J. Villena and L. Silveira. Arms - automatic residue-minimization based sampling for multi-point modeling techniques. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, 2009.

[66] 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.

[67] T. Y. Wang and C. C. Chen. 3-D thermal-ADI: a linear-time chip level transient thermal simulator. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(12):1434–1445, Dec. 2002.

[68] Yefu Wang, Kai Ma, and Xiaorui Wang. Temperature-constrained power control for chip multiprocessors with online model estimation. In *Proc. Int. Symp. on Computer Architecture (ISCA)*, pages 314–324, 2009.

[69] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 2002.

[70] 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.

[71] 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.

[72] Y. Yang, Z. P. Gu, C. Zhu, R. P. Dick, and L. Shang. ISAC: Integrated space and time adaptive chip-package thermal analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 16(1):86–99, 2007.

[73] Inchoon Yeo, Chih Chun Liu, and Eun Jung Kim. Predictive dynamic thermal management for multicore systems. In *Proc. Design Automation Conf. (DAC)*, DAC '08, pages 734–739, New York, NY, USA, 2008. ACM.

[74] F. Zanini, D. Atienza, L. Benini, and G. De Micheli. Multicore thermal management with model predictive control. In *Proc. 19th European Conference on Cicuit Theory and Design*, pages 90–95, Piscataway, NJ, USA, 2009. IEEE Press.

[75] Yufu Zhang and Ankur Srivastava. Adaptive and autonomous thermal tracking for high performance computing systems. In *Proc. Design Automation Conf. (DAC)*, pages 68–73, 2010.

[76] Yunkai Zhou. *Numerical methods for large scale matrix equations with applications in LTI system model reduction (Ph.D. Thesis)*. Rice University, 2002.