Adaptive multi-domain thermal modeling and analysis for integrated circuit synthesis and design

Yonghong Yang*, Changyun Zhu*, Zhenyu (Peter) Gu†, Li Shang*, and Robert P. Dick†

*ECE Department
Queen's University
Kingston, ON K7L 3N6, Canada
{4yy6, 4cz1}@qlink.queensu.ca, li.shang@queensu.ca

†EECS Department
Northwestern University
Evanston, IL 60208, U.S.A.
{zgu646, dickrp}@eecs.northwestern.edu

Abstract— **Chip-package thermal analysis is necessary for the design and synthesis of reliable, high-performance, low-power, compact integrated circuits (ICs). Many methods of IC thermal analysis suffer performance or accuracy problems that prevent use in IC synthesis and hinder use in architectural design.**

This article describes ISAC, a novel, fast, accurate thermal analysis system for use in IC synthesis and design. We present new, cooperative, temporal and spatial adaptation methods to dramatically accelerate accurate analysis. The proposed system unifies steady-state, time-domain, and frequency-domain analysis techniques. It is composed of our spatially-adaptive multigrid iterative solver, a new temporally and spatially adaptive asynchronous time marching solver, and a new spatially-adaptive frequency-domain moment matching solver. Together, these cooperative adaptation and multi-domain analysis techniques allow the proposed system to efficiently solve the static, short time scale, and long time scale variants of the IC thermal analysis problem.

Experimental results demonstrate significant performance improvement over existing thermal analysis solutions. Our spatial adaptation techniques bring a 21.6×–690.0× speedup over recently-published steady-state thermal analysis techniques. Our unified spatial and temporal adaptation techniques, within our asynchronous time marching method, bring a 1,071×–1,890× speedup over other widely-used, time-domain thermal analysis techniques with less than 0.5% error. Our spatial adaptation techniques enable the efficient use of our frequency-domain thermal analysis technique, which brings a 10×–100× speedup over the fastest-known time-domain technique, when used for long time scale thermal analysis. The thermal analysis system described in this article has been implemented as a C/C++ library that has been publicly released for free academic and personal use.

I. Introduction

Temperature has a huge impact on IC performance, cooling cost, reliability, and power consumption. The latencies of transistors and metal wires increase with chip temperature, as do the probabilities of many lifetime reliability faults [1, 2]. For example, electromigration failure rate is an exponential function of temperature. Leakage power consumption is now responsible for a substantial proportion of overall power consumption in commercial designs and increases with temperature [3]. To enable reliable and low-power IC design, run-time thermal profiles must be predicted and optimized during design and synthesis. However, predicting the run-time thermal profile of an IC during design is a difficult, and potentially expensive, task.

IC thermal analysis is the modeling and simulation of heat flow in IC chips and packages. It is conducted using computationally-expensive numerical methods that have given designers the choice between accuracy or speed, but not both. Dynamic chip-package thermal analysis is the process of determining run-time three-dimensional temperature profiles, given a three-dimensional starting temperature profile, power consumption profiles, the chip-package structure, and the specific heat capacities and thermal conductivities of the materials composing the IC chip and package. The steady-state thermal analysis problem is similar. However, only the temperature profile as time proceeds to infinity need be calculated.

Researchers have developed steady-state and dynamic thermal analysis techniques that are suitable for manual design planning of ICs. However, use of thermal analysis in IC synthesis requires fast and accurate techniques that are capable of handling thousands of thermal elements. These techniques can be placed within five categories: non-linear finite element solvers [4], steady-state techniques based on discretization and direct matrix solvers [5], steady-state Green's function based techniques [6], steady-state techniques based on iterative matrix solvers [7], non-adaptive synchronous time marching techniques [5], and frequency-domain techniques [8]–[10].

Non-linear finite element solvers, such as the COMSOL Multiphysics package, are general and highly accurate but far too slow for use in IC synthesis. Steady-state techniques based on direct matrix solvers are too slow for use in accurate thermal analysis during IC synthesis. Steady-state Green's function integration kernel based techniques are fast. However, existing techniques are less accurate than those based on direct or iterative matrix solvers. Steady-state techniques based on iterative matrix solvers have the potential to be fast and accurate. They are suitable for thermal analysis, design, and synthesis of ICs that have unchanging power profiles. However, further improvement is possible. In addition, steady-state techniques cannot determine the dynamic, time-dependent temperature profile of an IC, i.e., for ICs with changing power profiles.

Dynamic thermal analysis is a substantially more demanding problem than steady-state thermal analysis. However, it is essential for determining the run-time temperature varation in ICs with varying power profiles. Synchronous time marching techniques iteratively update the temperatures of all thermal elements within an IC by taking small, synchronized, steps forward in time. These techniques are plagued by either low accuracy or poor performance due to the requirement that all thermal elements move forward in time at the same rate. Although synchronous time marching techniques, be they non-adaptive or globally adaptive, can quickly report temperature profiles over short time scales, their execution times increase linearly with increasing time scale. In addition, as a result of round-off errors, their accuracies are generally highest early in the analysis time period and degrade as time scales increase. Frequency-domain techniques determine the approximate frequency-domain responses of IC thermal element temperatures via methods such as moment matching [11]. This allows fast and accurate prediction of IC thermal profiles over long time scales. However, frequency-domain techniques are plagued by startup costs that make their use impractical for large problem instances, i.e., those containing numerous thermal elements.

This article describes ISAC, a comprehensive solution to the IC thermal analysis problem that is rapid and accurate for steady-state, short time scale dynamic, and long time scale dynamic problems. This work makes the following main contributions:

1) We present new, cooperative, temporal and spatial adaptation methods to dramatically accelerate accurate chip-package thermal analysis.
2) ISAC unifies steady-state, time-domain, and frequency-domain analysis techniques, allowing a broad range of thermal analysis problems to be rapidly and accurately solved.
3) We have developed a new temporally and spatially adaptive asynchronous time marching technique that brings a 1,071×–

This work is supported in part by the NSERC Discovery Grant #388694-01 and in part by the NSF under award CNS-0347941.

ICCAD'06, November 5-9, 2006, San Jose, CA

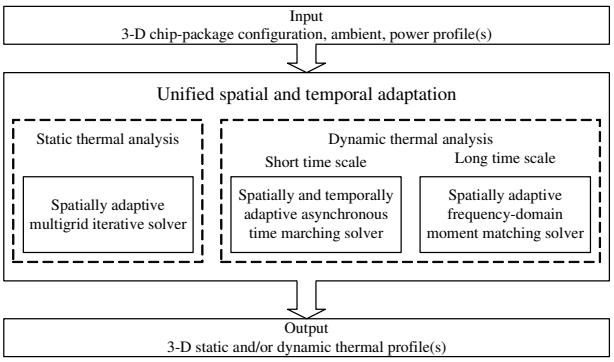


Fig. 1. ISAC Overview

1,890× speedup over other widely-used, time-domain thermal analysis techniques with less than 0.5% error.

4) The proposed spatial adaptation technique enables the use of our frequency-domain moment matching technique on large problems, bringing a 10×–100× speedup over the fastest-know time-domain technique, when used for long time scale thermal analysis.

The thermal analysis system described in this article has been implemented as a C/C++ library that has been publicly released for free academic and personal use: http://post.queensu.ca/~shangl/isac/.

The rest of this article is organized as follows. Section II introduces the IC thermal analysis problem and gives a brief overview of the proposed adaptive multi-domain thermal analysis system. Section III describes our cooperative adaptation methods, summarizes our steady-state thermal analysis technique, and describes our time-domain and frequency-domain thermal analysis techniques in detail. Section IV presents experimental results. Section V describes the C/C++ library interface for the software implementation of the proposed thermal analysis system. We draw conclusions in Section VI.

II. Problem definition and ISAC architecture

This section introduces the IC thermal analysis problem and describes the system architecture of ISAC. Note that this version of ISAC has grown in capabilities and techniques compared with a more preliminary system with the same name [12]. However, we thought it would be least confusing to current users of the software to keep the same name.

Problem definition

ISAC characterizes the heat diffusion process through an IC chip and cooling package. Thermal conduction is governed by the following partial differential equation.

$$\rho c \frac{\partial T(\vec{r},t)}{\partial t} = \nabla \cdot (k(\vec{r}) \nabla T(\vec{r},t)) + p(\vec{r},t) \quad (1)$$

where ρ is material density; c is mass heat capacity; $T(\vec{r},t)$ and $k(\vec{r})$ are temperature and thermal conductivity of the material at position $\vec{r}$ and time t; and $p(\vec{r},t)$ is heat source power density.

To conduct numerical thermal analysis, the IC chip and package are partitioned into numerous elements through a discretization process. The distributed thermal characteristics of the IC, package, and cooling solution are then modeled using finite difference discretization in which each element is a node connected to neighboring elements via thermal resistors and connected to a node at the ambient temperature via a thermal capacitor. Thus,

$$\mathbf{C}\frac{d\mathbf{T}(t)}{dt} = \mathbf{A}\mathbf{T}(t) + \mathbf{P}U(t) \quad (2)$$

where the thermal capacitance matrix, $\mathbf{C}$, is an $[N \times N]$ diagonal matrix; the thermal conductivity matrix, $\mathbf{A}$, is an $[N \times N]$ sparse matrix; $\mathbf{T(t)}$ and $\mathbf{P(t)}$ are $[N \times 1]$ temperature and power vectors; and $\mathbf{U(t)}$ is the time step function. As we return to the roots of this formalism, it is interesting to note that Georg Simon Ohm's work on electrical current in circuits [13] was based on Fourier's study of heat transfer.

ISAC architecture overview

Figure 1 shows the overview of ISAC, which unifies steady-state, time domain, and frequency domain techniques to efficiently and accurately address steady-state and dynamic IC thermal analysis problems.

Steady-State thermal analysis characterizes temperature distribution when heat flow does not vary with time. In IC designs, steady-state thermal analysis is sufficient for applications with stable power profiles or periodically changing power profiles that cycle quickly. For steady-state thermal analysis, the left term in Equation 2 expressing temperature variation as function of time, t, is dropped.

ISAC conducts steady-state thermal analysis using an efficient spatially-adaptive multigrid relaxation method. In ISAC, the discretization process of IC chip and package is performed using recursive refinement, and maintained hierarchically based on discretization granularity (see Section III-A). This adaptive refinement technique makes use of a hybrid tree structure to bound the temperature difference between adjacent elements (for accuracy) while minimizing the number of elements (for efficiency). The multigrid solver is also used for matrix inversion in the frequency-domain dynamic solver, as described in Section III-C.1.

Dynamic thermal analysis characterizes run-time IC thermal profile when the transient features of power profile are significant. ISAC consists of two dynamic thermal analysis algorithms: a time marching method and a moment matching method to handle short time scale and long time scale dynamic thermal analysis, respectively.

In the proposed spatially and temporally adaptive asynchronous time matching method, run-time temperature changes are discretized into numerous time steps and estimated via a limited-order expansion of the actual temperature function around each time instant. Its computational complexity is linearly proportional to the number of time steps and elements. Short time steps are sometimes necessary for accuracy. In ISAC, the time step magnitude of each element is adjusted independently, allowing elements to progress forward in time asynchronously. Care is taken to prevent time deviations from growing large enough to introduce error (see Section III-B).

The proposed spatially-adaptive frequency-domain numerical method derives an approximate analytical solution, which is used to compute run-time thermal profiles without the need of expensive time-domain iteration. However, deriving this analytical solution is expensive. Therefore, ISAC uses the spatially-adaptive moment matching method for long time scale dynamic thermal analysis, allowing the cost of deriving the analytical approximation to be amortized (Section III-C).

The major challenges of numerical IC thermal analysis are high computational complexity and memory usage. Stringent modeling accuracy constraints require fine-grain discretization, resulting in numerous grid elements. For multigrid-based steady-state thermal analysis, both computational complexity and memory usage are superlinearly proportional to the number of thermal elements. For dynamic thermal analysis using the time matching method, higher modeling accuracy requires the reduction of both spatial and temporal discretization granularities, increasing the computational overhead of this method. For dynamic thermal analysis using moment matching, deriving initial analytical approximations is both computation and memory intensive. High thermal element count may hinder or prevent the applicability of the frequency-domain method. In addition, the time complexity of this method increases linearly with increasing simulation time scale. Spatial adaptation is critical for efficiency.

III. Proposed adaptive thermal analysis algorithms

This section describes the proposed adaptive multi-domain thermal analysis techniques. Section III-A presents our spatial adaptation method and spatially-adaptive multigrid iterative technique for steady-state thermal analysis. Section III-B presents our cooperative temporal and spatial adaptation methods, and describes our asynchronous time matching technique for short time scale dynamic thermal analysis. Section III-C presents our spatially adaptive frequency-domain technique for long time scale dynamic thermal analysis.

III-A. Steady-state thermal analysis and spatial adaptation

Algorithm 1 shows the steady-state thermal analysis algorithm used in ISAC. Given the input power profile and an initial spatial

Algorithm 1 Spatially adaptive steady-state thermal analysis

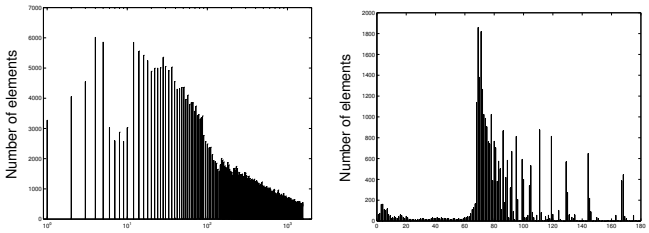
```
1: INPUT: A: initial spatial discretization of the IC chip and package
2: INPUT: P: steady-state power profile, flag_refinement ← true
3: while (flag_refinement) do
4:    flag_refinement ← false
5:    T ← multigrid_solver(A, P) (Algorithm 3)
6:    for Every adjacent thermal element pair {E_i, E_j} do
7:       if |T_i − T_j| > T_th then
8:          Hierarchical partitioning {E_i, E_j} by ⌈log_2 (|T_i − T_j| /T_th)⌉
9:          flag_refinement ← true
10:      end if
11:   end for
12:   if flag_refinement = false then
13:      return T: 3-D chip-package thermal profile
14:   end if
15:   Update matrix A
16: end while
```

discretization of the IC chip and package (line 1 and 2), steady-state thermal analysis is conducted using a multigrid relaxation kernel (line 5). A multigrid method with Gauss-Seidel smoothing iteratively solves (typically sparse) systems of linear equations using a multi-level scheme [14]. This solver, which is also used for matrix inversion within dynamic thermal analysis, will be explained in detail (Algorithm 3) in Section III-C. Note that, in Algorithm 3, for steady-state thermal analysis, the multigrid relaxation kernel is invoked only once. In this use, $\mathbf{e}_i$ is replaced by $\mathbf{P}$, the input power profile and the solution, $\mathbf{X}$, is the steady-state temperature profile. Steady-state thermal analysis reports the temperature of each individual thermal element. The spatial thermal difference between adjacent thermal elements are evaluated. Thermal grid elements with temperature differences exceeding T_{th} will be further hierarchically refined (line 6–11). The thermal conductivity matrix $\mathbf{A}$ is then updated (line 15). This process continues until the thermal difference constraint is satisfied. Finally, the thermal profile of the IC chip and package is reported (line 13).

During thermal analysis, both time complexity and memory usage are linearly or superlinearly dependant on the number of thermal elements. Therefore, it is critical to limit discretization granularity. On the other hand, fine-grain discretization is crucial to accurately characterize the three dimensional thermal profile of IC chip and package. Achieving right balance of modeling accuracy and efficiency is challenging.

The spatial thermal gradient of IC chip and package (defined as $dT(\vec{r})/d\vec{r}$) exhibits significant spatial variation due to the heterogeneity of thermal conductivity and heat capacity in different materials, as well as the variation of power profiles. For instance, significant spatial thermal variation is commonly observed within the active layer of the silicon die, while the lateral thermal profile within the heat sink is normally smoother. Figure 2(a) shows the normalized inter-element temperature differences in steady-state thermal analysis of an IC implementation of a digital signal processing benchmark. The x-axis indicates the temperature difference between a pair of neighboring elements. The y-axis indicates the number of neighboring elements with the given temperature difference. The wide distribution of temperature differences shown in this figure suggests that some neighboring elements might be combined with little loss of accuracy in order to improve performance. This motivated us to design an efficient, thermal gradient driven, adaptive spatial discretization refinement technique that automatically adjusts the spatial partitioning resolution to maximize thermal modeling efficiency and guarantee modeling accuracy.

In this technique, the spatial discretization process is governed by temperature difference constraints. Iterative refinement is conducted in a hierarchical fashion. During each iteration, temperature approximation is performed until convergence to a stable profile. Neighboring grid elements with temperature differences exceeding thermal difference constraints are recursively partitioned. Given adjacent thermal element temperatures, T_i and T_j, and the spatial thermal difference constraint, T_{th}, the number of partitions for these two elements is $|T_i - T_j|/T_{th}$. The position of each individual cut is a function of the thermal conductivities and the sizes of the elements. Using hierarchical partitioning, $\lceil \log_2 (|T_i - T_j| /T_{th}) \rceil$ cuts are required. It is possible that two neighboring thermal elements may have equivalent average temperatures, resulting in premature termination



(a) Normalized inter-element temperature differences (b) Normalized maximum step size

Fig. 2. The potential of adaptive thermal modeling

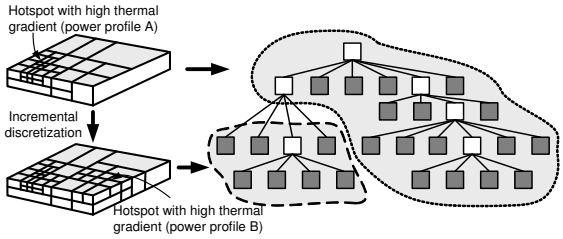


Fig. 3. Spatial discretization refinement

of the refinement procedure. At present, starting from a moderately fine-grained partitioning is sufficient to allow accurate results for all the problem instances we have encountered. However, we are in the process of adding the capability of adjusting the initial partitioning based on the locations and sizes of the blocks in the supplied power profile.

To support incremental spatial discretization refinement, we propose the hybrid tree structure illustrated on the right side of Figure 3. This structure provides an efficient representation of the incremental refinement process, which corresponds to the incremental growth of the tree. In this hybrid tree, all the leaf nodes, shown as grey blocks in Figure 3, refer to the thermal elements used in both steady-state and dynamic thermal analysis. This hybrid tree structure also provides an efficient hierarchical representation for multigrid analysis, by enabling efficient traversal through different levels of the tree.

III-B. Asynchronous time marching method

We will contrast the proposed asynchronous time marching technique used in ISAC with the popular Runge-Kutta family of finite difference techniques [5, 15]. When Runge-Kutta time marching techniques are used for thermal analysis, thermal elements advance in time in lock step. At each time step, the temperature at a fixed time offset is computed via a bounded-order function approximating the element's true temperature. This function is a reformulation of the Taylor series expansion of the thermal element's temperature function around its current time [15]. An element's bounded-order function depends on the thermal conductivities, heat capacities, and temperatures of its (transitive) neighboring thermal elements. The time complexity of this method is reduced by amortizing computations over use by many (transitive) neighbors, permitting the use of high-order methods. For many problems, higher-order methods are useful because they allow the bounded-order approximation function to accurately approximate the real temperature over long time scales, thereby allowing large time steps and speeding analysis. However, our analysis and experiments show that the benefits of asynchronous time progression far outweigh the benefits of using high-order method, for the IC thermal analysis problem.

There are two categories of Runge-Kutta methods: non-adaptive and adaptive. Non-adaptive Runge-Kutta methods use the same time step size throughout analysis. Unfortunately, this means that performance is bounded by the smallest time step required by any element at any time. A non-adaptive fourth-order Runge-Kutta method is in common use for dynamic IC thermal analysis [5]. We found that non-adaptive fourth-order Runge-Kutta techniques were incapable of handling thermal models with enough discrete elements to permit accuracy, while running with adequate performance for use within IC

synthesis. It is possible to improve performance substantially without loss of accuracy by using an adaptive fourth-order Runge-Kutta technique in which thermal elements advance in time in lock-step but, at each time, the step size is adjusted to the minimal size required for accuracy by any thermal element. The adaptive fourth-order Runge-Kutta method appears to be near a local optimum, performing better than lower-order and non-adaptive Runge-Kutta techniques without loss of accuracy. However, we have found it to be far from the global optimum for IC thermal analysis.

Maximum safe step size is the largest time step a thermal element may use when updating its temperature without resulting in unacceptable error. If a local approximation function is used with too large a step size for the approximated function, truncation error may exceed an error bound threshold. Therefore, before each time step, the maximum safe step size satisfying a truncation error bound on an element's temperature as a function of the states of its (transitive) neighbors is computed (see Equation 16).

Figure 2(b) is a histogram illustrating the distribution of maximum step sizes satisfying a temperature error bound of 1×10^{-5} K at a single time during the time-domain thermal analysis of the same benchmark used in Figure 2(a). In this figure, the time step sizes are normalized to the minimum over all thermal elements. Allowing most thermal elements to take time steps larger than the minimum has the potential to greatly improve efficiency. ISAC uses a temporally and spatially adaptive asynchronous element time marching technique for short time scale dynamic thermal analysis. Instead of advancing all thermal elements forward in time synchronously, our method allows thermal elements to advance asynchronously. In addition, it uses a heterogeneous thermal element discretization to minimize the number of thermal elements under a constraint on neighboring thermal element temperature differences (see Section III-A).

Spatial discretization refinement takes all input power profiles into consideration through incremental refinement. Consider the example illustrated in Figure 3. The initial spatial refinement is based on the input power profile *A*. A hotspot occurs at bottom-left corner of the chip active layer, increasing the thermal gradient in that region. As a consequence, finer-granularity spatial discretization is used in that region. Power profile *B* is later examined, causing further refinement of a region near the right of the chip. For each thermal element, partitioning decisions are based on the maximum difference between neighboring element temperatures over the steady-state thermal profiles associated with all power profiles in the trace provided for analysis.

Recall that computing the next temperature of a thermal element requires knowledge of the temperatures of its (transitive) neighbors at the element's current time. This poses no special problem for conventional finite difference techniques. However, allowing asynchronous thermal element times makes it necessary to compute the temperatures of an element's (transitive) neighbors at the local time of the advancing element in order to compute the element's next temperature using a bounded-order approximation function. This prevents the amortization of temperature computations over multiple (transitive) neighbors (which was permitted by the Runge-Kutta methods). In other words, asynchronous element times make high-order approximation methods more computationally expensive. However, asynchronous operation relaxes the constraint that each step size is bounded by the minimum step size required by any element at that time. For the dynamic thermal analysis problem, the gains possible via asynchronous step size adaptation hugely outweigh the disadvantages of using a lower-order approximation function. While maintaining an error lower than 0.5%, the proposed approach improves performance by at least 1,071× over adaptive fourth-order Runge-Kutta (see Section IV).

We now give the derivation of the thermal element update equation used in the asynchronous time marching method. Noting the definitions in Equation 1, and given that $T_i(t)$ is the temperature of element i at time t, G_{in} is the thermal conductivity between thermal elements i and n, V_i is the volume of thermal element i, and N are the element's neighbors, we know that the net heat flow for a given thermal element, i, is zero.

$$0 = \sum_{n \in N_i} \left(T_i(t) - T_n \cdot u(t)\right) G_{in} + \rho_i c_i V_i \frac{dT}{dt} - p_i V_i \cdot u(t) \quad (3)$$

This can be simplified by introducing a few variables.

$$\text{Let } \alpha = \sum_{n \in N_i} G_{in}, \beta = \sum_{n \in N_i} T_n G_{in} + p_i V_i, \text{ and} \quad (4)$$

$$\kappa = \rho_i c_i V_i. \text{ Thus } 0 = T(t) \cdot \alpha - u(t) \cdot \beta + \kappa \frac{dT}{dt} \quad (5)$$

and solved for $T(t)$.

$$\mathcal{L}\left(T(t) \cdot \alpha - u(t) \cdot \beta + \kappa \frac{dT}{dt}\right) = T(s) \cdot \alpha - 1/s \cdot \beta + T(s) \cdot s \cdot \kappa - T(0^-) \cdot \kappa \quad (6)$$

$$T(s) = \frac{\beta + s \cdot T(0^-) \cdot \kappa}{s \cdot (\alpha + s \cdot \kappa)} \text{ (by 5 and 6)} \quad (7)$$

$$\text{Let } \gamma = \frac{1}{s \cdot (\alpha + s \cdot \kappa)}, \text{ thus } T(s) = \frac{T(0^-)}{s + \alpha/\kappa} + \beta \cdot \gamma \quad (8)$$

Linearity theorem for γ.

$$\frac{1}{s \cdot (\alpha + s \cdot \kappa)} = \frac{A}{s} + \frac{B}{\alpha + s \cdot \kappa}, a = A \cdot (\alpha + s \cdot \kappa) + B \cdot s \quad (9)$$

Let $s = 0$ to yield $A = 1/\alpha$ and let $s = -\alpha/\kappa$ to yield $B = -\kappa/\alpha$.

$$\gamma = \frac{1}{s \cdot (\alpha + s \cdot \kappa)} = \frac{1/\alpha}{s} - \frac{1/\alpha}{s + \alpha/\kappa} \quad (10)$$

$$T(s) = \frac{T(0^-)}{s + \alpha/\kappa} + \frac{\beta/\alpha}{s} - \frac{\beta/\alpha}{s + \alpha/\kappa} \quad (11)$$

$$\mathcal{L}^{-1}\left(\frac{T(0^-)}{s + \alpha/\kappa} + \frac{\beta/\alpha}{s} - \frac{\beta/\alpha}{s + \alpha/\kappa}\right) = u(t) \cdot \beta/\alpha + (T(0^-) - \beta/\alpha) e^{-t \cdot \alpha/\kappa} \quad (12)$$

$$\underset{t \geq 0}{T(t)} = \beta/\alpha + \left(T(0^-) - \beta/\alpha\right) e^{-t \cdot \alpha/\kappa} \quad (13)$$

Note that, although the impact of transitive neighbors is not explicitly stated, it may be considered in higher-order methods. Thus, β should be redefined to explicitly consider transitive neighbors.

$$\beta_i(t, M) = \begin{cases} \sum_{n \in N_i} T_n(t, M) \cdot G_{in} + p_i V_i & \text{if } M > 0 \\ p_i V_i & \text{otherwise} \end{cases} \quad (14)$$

given that M is the remaining transitive neighbor depth. In other words, it is necessary to consider the heat flow from and to transitive neighbors to a depth of M. Thus, the nearest-neighbor approximation of temperature of element i at time $t + h$ follows:

$$T_i(t + h, M) = \beta_i(t + h, M - 1)/\alpha_i + \frac{T_i(t) - \beta_i(t + h, M - 1)/\alpha_i}{e^{(h \cdot \alpha_i)/\kappa}} \quad (15)$$

Note that ensuring acceptable accuracy will still require the following bound on step size, h, remaining terse by neglecting the parameters of $T_i(t, M)$:

$$h_i^+(t_i) = u \cdot \left[\frac{1}{y}\left|\frac{dT_i}{dt}\frac{3t_i h_i}{2} - \frac{3h_i}{4}\left(\frac{dT_i}{dt}t_i + \frac{dT_i}{dt}\left(t_i + \frac{3h_i}{4}\right)\right)\right|\right]^{-1/v} \quad (16)$$

I.e., we compare the temperature after two $3/4h$ steps with the temperature after one $3/2h$ step in order to estimate truncation error. The difference between these values is related to the contribution of the truncated higher-order terms of the temperature function. This allows an h^+ satisfying bounds on local truncation error to be computed.

Boundary conditions are imposed by the chip, package, and cooling solution. Updates are carried out using a run queue of thermal elements sorted in order of increasing target step times ($t + h$), i.e., we always advance the element that will end up at a minimal time, thereby reducing time discrepancies among elements in the presence of asynchronous progression.

III-C. Moment matching algorithm for thermal analysis

This section describes the design and analysis of an efficient and accurate frequency-domain IC thermal analysis algorithm. As described in Section I, dynamic thermal analysis may be conducted via time marching techniques as well as frequency-domain moment

Algorithm 2 Moment matching for dynamic thermal analysis

Require: Chip-package region thermal conductivities
Require: Chip-package region heat capacities
Require: Time series of active layer power profiles
Ensure: Time series IC temperature profiles
1: **subtask** Moment matching {Expensive: Amortized over many uses}
2: Homogeneous discretization of IC
3: Spatial adaptation: Minimize elements, bound temperature difference
4: Multigrid technique for fast conductivity matrix inversion
5: Moment matrix calculation via iterated multiplication and extraction
6: Eigen decomposition
7: Compute poles
8: Compute system response matrix, $\mathbf{H}$
9: **end subtask**
10: **subtask** Periodic computation {Moderate cost and frequency}
11: Compute element power, initial temperature coefficient matrix
12: **end subtask**
13: **subtask** Dynamic time-domain calculations {Inexpensive but frequent}
14: Convert to time-domain representation
15: **end subtask**

matching; each technique is appropriate under certain circumstances. In this section, we will focus on moment matching, which can result in dramatic improvements in analysis time for long time span dynamic thermal analysis. A numerical method will be used to match the moments of the thermal profile's response to the power profile.

Moment matching based thermal analysis is composed of three stages: static, periodic, and dynamic. The static analysis phase need be completed only once for each IC chip-package configuration, i.e., once for a (potentially long) series of power profiles. In this phase, the reduced order thermal model for the chip, package, and heat sink is generated. The periodic phase occurs each time a change is reported in the power profile of the IC active layer, e.g., every 1 ms–100 ms in normal applications. In this phase, the moments of the reduced order model are used to compute system response coefficients that will be used to determine temperature profile as a function of time. In the final dynamic phase, the time-varying thermal profile of the IC is computed based on the system response coefficients. Multiple dynamic phases may occur within each period phase, i.e., it may be necessary to compute the temperature profile at multiple times between two changes to the power profile.

III-C.1. Spatial adaptation and multigrid analysis: We initially use the adaptive grid refinement technique described in Section III-A to spatially discretize the chip, package, and heat sink. This technique is of critical importance to permit moment matching to deal with large problem instances. It preserves detail in the regions with the most temperature variation, while permitting coarser grid resolution in regions within which temperature will be more uniform.

In the moment matching method, the first step is using the Laplace transform to derive the frequency domain form of the heat transfer equation (Equation 2):

$$\mathbf{C}(s\mathbf{T}(s) - \mathbf{T}(0^-)) = \mathbf{A}\mathbf{T}(s) + \mathbf{P}/s$$
$$\mathbf{T}(s) = -\mathbf{A}^{-1}(\mathbf{I} - s\mathbf{C}\mathbf{A}^{-1})^{-1}(\mathbf{P}/s + \mathbf{C}\mathbf{T}(0^-)) \quad (17)$$

Inverting the thermal conductivity matrix, $\mathbf{A}$, is the first step in moment matching. This step is one of the most critical for performance due to the size of $\mathbf{A}$. Accurate thermal analysis with fine-grain discretization results in a large $\mathbf{A}$ matrix, with a size quadratic in the number of thermal elements and a number of non-zero entries proportional to the number of thermal elements, e.g., given 32,768 homogeneous thermal elements $\mathbf{A}$ contains 1,073,741,824 elements, 196,608 of which are non-zero. Directly solving $\mathbf{A}$ can be computationally expensive. We have developed a hybrid, heterogeneous multigrid-based iterative relaxation technique for solving large discretized partial differential equations that is used for matrix inversion and steady-state thermal analysis.

In the $\mathbf{A}$ matrix, each non-zero value refers to the thermal conductivity between two neighboring thermal elements. Each thermal element has few neighbors, e.g., six in homogeneous partitioning. Therefore, $\mathbf{A}$ is sparse. However, spatial adaptation results in an irregular matrix; efficient solvers for band matrices are not applicable. The preconditioned conjugate gradient method can accelerate the solution of some sparse matrices. However, experiments show that, if the number of iteration steps required for convergence is much less than N, the number of grid elements, our multigrid iterative solver is more than ten times faster than an otherwise identical multigrid iterative solver using the preconditioned conjugate gradient method. Moreover, the hierarchical structure of the chip-package discretization naturally matches the nested iteration of the multigrid method. By iteratively traversing the discretization refinement hierarchy, our multigrid method speeds convergence.

Algorithm 3 Multigrid matrix solver

Require: matrix $\mathbf{A}$
Ensure: $B = \mathbf{A}^{-1}$
1: **for** $0 < i < N$ **do**
2: Define problem $\mathbf{A}\mathbf{X}_i = \mathbf{e}_i$
3: Pre-smoothing step: Iteratively relax initial random solution.
4: **subtask** coarse grid correction
5: Compute residue from finer grid.
6: Approximate residue in coarser grid.
7: Solve coarser grid problem using relaxation.
8: **if** coarsest level has been reached **then**
9: Directly solve problem at this level.
10: **else**
11: Recursively apply the multigrid method.
12: **end if**
13: Map the correction back from the coarser to finer grid.
14: **end subtask**
15: Post smoothing step: Add correction to solution at finest grid level.
16: Iteratively relax to obtain the final solution.
17: $\mathbf{B}[i] \leftarrow \mathbf{X}_i$
18: **end for**

Algorithm 3 describes the proposed multigrid iterative relaxation technique used for matrix inversion. Lines 4–16 show the multigrid relaxation kernel. To invert matrix $\mathbf{A}$, through each iteration i, the solver is invoked to compute $\mathbf{A}\mathbf{X}_i = \mathbf{e}_i$, in which $\mathbf{e}_i$ is the ith column of identity matrix $\mathbf{I}$. The solution $\mathbf{X}_i$ corresponds to the ith column of matrix $\mathbf{A}^{-1}$, the inverse of $\mathbf{A}$. Each $\mathbf{X}_i$ is computed using multigrid relaxation. An initial solution is first computed at the finest grid resolution using Gauss-Seidel method (Line 3). Low-frequency errors are characterized recursively at coarser grid resolutions (Lines 5–12) and mapped back to the initial solution for error correction (Line 13–15). Gauss-Seidel smoothing at the finest grid resolution is used to produce the final $\mathbf{X}_i$ (Line 16).

Despite its high efficiency relative to other direct and indirect solvers, the proposed multigrid relaxation method is still computation intensive. Its execution time is a superlinear in N, the number of rows of the matrix. For matrix inversion, the multigrid solver is invoked N times; for large problem instances, matrix inversion is a major performance bottleneck in moment matching.

III-C.2. Moment matrix calculation via iterated matrix multiplication: The second time-critical step in moment matching is the calculation of the moment matrix, $\mathbf{M}$. This matrix is composed of the first columns of matrices $\mathbf{F}_1, \mathbf{F}_2, \cdots, \mathbf{F}_Q$ where Q is the number of moments to which the model will be reduced. $\mathbf{A}$ is the $N \times N$ thermal conductivity matrix and $\mathbf{C}$ is the $N \times N$, diagonal, heat capacity matrix.

$$\forall_{i=0}^{Q-1}\mathbf{F}_i = -\mathbf{A}^{-1}(\mathbf{C}\mathbf{A}^{-1})^i \quad (18)$$

This matrix exponentiation can be reduced to a series of multiplications, each of which is necessary to compute the previous $\mathbf{F}$ matrix. However, each $\mathbf{F}$ is a dense $N \times N$ matrix and a multiplication is required for each moment. The time cost for this stage, using classical matrix multiplication, is QN^3 (note that this matrix contains few zeros). There exist fast matrix multiplication algorithms such as Winograd's $\mathcal{O}\left(N^{2.376}\right)$ time variant [16] of Strassen's method that might reduce the time complexity of this stage to $\mathcal{O}\left(QN^{2.376}\right)$. However, we tested the GEMMW implementation [17] of this technique and found that it took at least twice as long as the AMD core math library (ACML) [18] multiplication routines for values of N up to 4,096: we conclude that computing $\mathbf{M}$ is expensive. This emphasizes the importance of spatial partitioning algorithm described in Section III-A for increasing efficiency by reducing N.

As shown in Figure 4 we have found that 8–10 moments are sufficient for high accuracy. Although selecting an appropriate number of moments, Q, to permit accuracy without degrading performance is an interesting problem, it is less critical to the static phase of moment matching than controlling the number of thermal elements. In moment matching, a few time-consuming operations require time linear in Q. In practice, the Q required for accurate analysis is bounded by a small integer. Therefore, designers can be somewhat conservative

when selecting Q, knowing that they will suffer, at most, a linear penalty in run time for this phase of moment matching. The impact of moment count on subsequent phases of thermal analysis is, however, significant and will be discussed in Section III-C.4.

At this stage, eigen decomposition is used to determine the poles of the reduced thermal system. Fortunately, only a $Q \times Q$ matrix need be decomposed. Before eigen decomposition, the modified Gram-Schmidt transformation should be used to orthogonalize the vectors.

By using the $\mathbf{F}$ matrices and the resulting poles, the coefficients of the frequency domain response of thermal element x corresponding to the power element j, can be calculated as follows.

$$\begin{bmatrix} -1/p_0 & -1/p_1 & \cdots & -1/p_{q-1} \\ -1/p_0^2 & -1/p_1^2 & \cdots & -1/p_{q-1}^2 \\ \cdots & \cdots & \cdots & \cdots \\ -1/p_0^q & -1/p_1^q & \cdots & -1/p_{q-1}^q \end{bmatrix} \begin{Bmatrix} H_{x,1,j} \\ H_{x,2,j} \\ \cdots \\ H_{x,q-1,j} \end{Bmatrix} = \begin{Bmatrix} m_{0(x,j)} \\ m_{1(x,j)} \\ \cdots \\ m_{q-1(x,j)} \end{Bmatrix} \tag{19}$$

The frequency domain response of thermal element x can be expressed by the coefficients $\mathbf{h}$ and the poles as follows.

$$T_x(s) = \sum_{i=0}^{q-1} \frac{\mathbf{h}_{xi}}{s - \mathbf{p}_i} \times \frac{\mathbf{P} + \mathbf{C}\mathbf{T}(0^-)s}{s} \tag{20}$$

For each pole, one thermal element has N coefficients, which correspond to N power elements. The preceding equations must be solved $N \times N$ times to derive the complete set of system response function coefficients. Therefore, the time complexity of these calculations is $\mathcal{O}\left(Q^2 N^2\right)$. At this point, the static moment matching phase is complete, and need not be carried out again for the given chip, package, heat sink, and (potentially long) series of power profiles.

III-C.3. Periodic phase: Power and initial temperature dependent coefficient computation: The computation of system response coefficients is expensive because, for each of the N thermal elements, it is necessary to iterate over $N \times Q$ matrix entries, where Q is the number of moments. One might attack the problem by attempting to adapt Q depending on the required number of moments for each element. However, independently reducing the number of moments used in the computation of the system response coefficients without changing the poles of the system would introduce substantial error because the values of all poles depend on the number of moments used to approximate the system response.

III-C.4. Dynamic phase: Time domain temperature computation: During this phase, given a power profile, the temperature profile of the IC may be calculated at (any) time. This phase requires $Q \times n$ operations to determine the temperature of each thermal element, in which Q is the number of moments in the reduced order thermal model and n is the number of elements under observation. Within the time span of each power profile, the run-time temperature of each element can be computed directly without the need of iteration. This is the reason for the superior performance of frequency-domain techniques over time-domain techniques in long time scale thermal analysis. Moreover, in some synthesis and architecture applications, only a subset of active-layer thermal elements need be observed, i.e., n can be much smaller than N.

IV. Experimental results

In this section, we evaluate the accuracy and performance of ISAC. Experiments were conducted on a Linux workstation with a 2.1 GHz Athlon processor and 1 GB of memory. ISAC is a unified thermal analysis platform containing a steady-state analysis technique, i.e., a spatially adaptive multigrid iterative method, and two dynamic analysis techniques, i.e., a spatially and temporally adaptive asynchronous time matching method for short time scales and a spatially adaptive moment matching method for long time scales.

We will compare the proposed adaptive algorithms with those used in other commercial and academic thermal analysis systems. In these comparisons, average error, e_{avg}, will be used as a measure of difference between thermal profiles:

$$e_{avg} = 1/|E| \sum_{e \in E} \left| T_e - T_e' \right| / T_e \tag{21}$$

where E is the set of elements on the surface of the active layer of the silicon die modeled by ISAC and T_e and T_e' are the temperatures of element e reported by another algorithm and ISAC, respectively. For the sake of consistency with existing work on IC thermal analysis, percentage error is computed with the fixed point of 0°C instead of 0 K (with apologies to purists). This is conservative; if comparisons were made in degrees Kelvin instead of degrees Celsius, the reported percentage error would be substantially lower. In all cases, we calculate average temperature difference for elements within a fine-grained homogeneous mesh, e.g., a large block with an average temperature of 80°C composed of two fine-grained blocks, one of which has a temperature of 75°C and one of which has a temperature of 85°C, has an average temperature difference of 5°C, not 0°C.

TABLE I
Results for asynchronous time matching method

Problem	ISAC				GARK4	
	CPU time (s)	Speedup (×)	Mem. (KB)	Error (%)	CPU time (s)	Mem. (KB)
chemical	1.35	1354	463.47	0.13	1827.41	4,506
dct_wang	0.39	1457	312.64	0.09	568.22	4,506
dct_dif	0.40	1807	332.91	0.05	722.64	4,506
dct_lee	0.85	1071	439.22	0.04	910.88	4,506
elliptic	2.24	1361	412.23	0.02	3042.61	4,506
iir77	0.86	1521	803.09	0.08	1305.25	4,506
jcb_sm	0.58	1890	357.30	0.11	1092.98	4,506
mac	1.65	1105	403.47	0.45	1817.71	4,506
paulin	0.77	1439	354.28	0.18	1111.68	4,506
pr2	1.06	1831	489.36	0.35	1932.95	4,506

Although we have partitioned the validation and performance evaluation of our thermal analysis methods by domain (steady-state, time-domain, and frequency-domain), either the time-domain or frequency-domain solvers can be used to solve dynamic thermal analysis problem instances. The correct method to use depends on which will yield better performance. This decision can be made by the user or automatically, based on time scale: the time-domain solver is generally faster for short time scales, e.g., tens of milliseconds, and frequency-domain solver is generally faster for long time scales.

This section is organized as follows. Section IV-A summarizes the validation of our spatially-adaptive steady-state thermal analysis algorithm. Section IV-B compares our new cooperative temporally and spatially adaptive asynchronous time marching method with a globally adaptive technique, as well as non-adaptive techniques in recently-published research. Section IV-C compares our new spatially-adaptive frequency-domain thermal analysis method with a commercial solver, and the fastest-known time-domain method.

IV-A. Steady-state analysis

ISAC contains algorithms for steady-state, short time scale, and long time scale thermal analysis. The steady-state thermal analysis algorithms used in ISAC have been validated in previous work [12]. However, we will briefly summarize those results. Spatial adaptation results in 27.5× and 690.0× speedup over a homogeneous multigrid steady-state thermal analysis for an IBM ASIC and a 16-core chip-level multiprocessor from the MIT Raw group. Temperatures had average deviations of less than 3% from the results produced by COMSOL Multiphysics [4]. When run on numerous high-level synthesis benchmarks, speedups ranged from 21.6×–202.9× and average temperature deviations were less than 1.5%, compared to a high-resolution non-adaptive multigrid method.

IV-B. Asynchronous time matching method

First, we evaluate the performance of the asynchronous time matching method implemented in ISAC, a third-order numerical method incorporating cooperative spatial and temporal adaption techniques. To demonstrate the effectiveness of these two techniques, we compare our proposed method with a globally-adaptive fourth-order Runge-Kutta method (GARK4), which uses global temporal adaptation with homogeneous partitioning. In addition, we compare with non-adaptive fourth-order Runge-Kutta, the dynamic method used in the HotSpot [5] thermal analysis system.

Table I shows the experimental results for the proposed adaptive asynchronous time matching method. For each benchmark, each technique does thermal analysis for 1 ms with an initial time step size of 10 ns and a temperature error bound of 1×10^{-5} K. Columns two and six show the CPU time used by ISAC and GARK4. ISAC consistently speeds up dynamic analysis by three orders of magnitude (column three), and reduces memory usage by 5.6–14.4× (column four and column seven). This high performance gain results from the

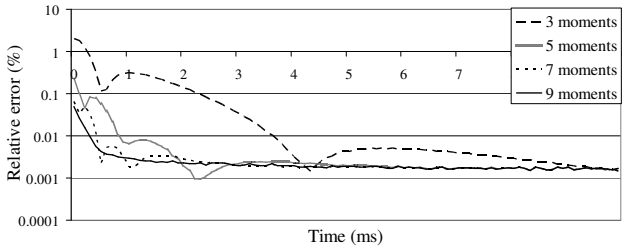


Fig. 4. Accuracy of moment matching method

TABLE II
EFFICIENCY OF ADAPTIVE MOMENT MATCHING METHOD

Problem	Elts.	Static $\mathbf{A}^{-1}$ (s)	Static $\mathbf{M}$ mul. (s)	Static $\mathbf{H}$ coeff. (s)	Periodic (ms)	Dynamic (μs)
chemical	3,383	93.44	16.53	0.80	104.35	0.26
dct_dif	2,282	32.28	8.54	0.42	55.37	0.20
dct_lee	2,430	42.23	7.78	0.40	50.91	0.18
dct_wang	3,206	371.31	23.39	0.84	106.25	0.20
elliptic	3,009	194.44	19.46	0.74	91.31	0.20
iir77	5,862	509.74	214.84	19.58	359.65	0.21
jcb_sm	2,608	125.93	12.63	0.58	72.45	0.20
mac	2,945	221.84	17.93	0.72	90.51	0.19
paulin	2,586	66.21	8.47	0.42	54.25	0.18
pr2	3,572	287.97	31.98	1.06	132.92	0.20

cooperative spatial and temporal adaptation techniques. As shown in column five, the deviation (e_{avg}) of ISAC's results from those of GARK4 is less than 0.5%.

Please note that the problem instances being considered here have 32,768 thermal elements, each. ISAC accelerates thermal analysis by constraining the number of thermal elements via spatial adaptation (as described in Section III-A) and by allowing different thermal elements to move through time asynchronously at different rates (as descried in Section III-B). Our spatial and temporal adaptation technique automatically consider impact on accuracy, allowing tremendous speedups with results that deviate from those of the globally-adaptive fourth-order Runge-Kutta method by less than 1%. In summary, the proposed adaptation methods accelerate time-domain dynamic thermal analysis by three orders of magnitude while producing solutions that are substantially equivalent to those produced by solvers using 32,768 thermal elements and synchronous time steps.

We believe that the globally-adaptive fourth-order Runge-Kutta method is a reasonable starting point for comparison. This method is commonly used by engineers interested in solving distributed differential equations via finite difference methods. However, in order to position ISAC relative to other recently-published work on chip-package thermal analysis, we will also discuss the merits of other techniques.

HotSpot is a widely-used thermal analysis package in the academic computer-aided design and computer architecture communities [5]. This tool initially used functional unit based partitioning and a fourth-order non-adaptive Runge-Kutta solver. Recently, support for homogeneous grid-based partitioning has been added. This *fast_tran_solver(·)* is still under development by its authors. We have encountered substantial irregularities when attempting to validate this solver and are presently in discussions with the authors. For the present, we will focus on comparisons with the fourth-order non-adaptive Runge-Kutta method.

In order to bound truncation error (which is generally more significant than round-off error for short time scale use) Runge-Kutta methods must use sufficiently small step sizes. Adaptive Runge-Kutta methods estimate truncation error before each time step, thereby selecting appropriate step sizes. Non-adaptive Runge-Kutta methods must use a single step size that is safe over the entire time range. In order to show the non-adaptive Runge-Kutta method in the best possible light, we used a globally-adaptive fourth-order Runge-Kutta method to determine the minimum of the step sizes used during the analysis run. This value was used in the non-adaptive fourth-order Runge-Kutta method to compare performance. Each of the problems in Table I required 9,436 seconds of CPU time to solve, i.e., the cooperative spatially and temporally adaptive methods used in ISAC consistently allow more than a 4,000× speedup over the non-adaptive fourth-order Runge-Kutta method.

IV-C. Adaptive moment matching method

We next evaluate the performance of the proposed spatially-adaptive moment matching method. This technique was developed for use in long time scale thermal analysis of large problem instances, making validation a challenging problem.

HSPICE failed to produce results for the benchmarks used in this work. We know of no other IC techniques capable of dynamic thermal analysis for the time scales, and problem instance sizes, handled by ISAC. Therefore, to compare with other techniques, it is necessary to bound problem instance sizes. For designs with 32 thermal elements, the proposed method produces results that differ from those of HSPICE by less than 1%.

We have characterized the analysis accuracy of the moment matching method as a function of number of moments and the time scale using the set of benchmarks described in Section IV-B. For each benchmark, a 100 ms simulation is performed using the proposed method with different moment counts: from one moment to ten moments. By using a tight error bound during numerical analysis, i.e., an error bound of 1×10^{-15} for matrix inversion, analysis using ten moments is highly accurate, and serves as a base case with which analysis with fewer moments may be compared. Figure 4 shows relative temperature error as a function of moment count and time averaged over a set of ten power profile transitions. For the sake of clarity, results using three, five, seven, and nine moments are plotted. As shown in this figure, as the number of moments increases, the relative error decreases superlinearly. For five or more moments, run-time analysis error after 1 ms is consistently less than 0.01%, relative to the ten-moment case, i.e., the frequency-domain approach achieves high analysis accuracy for long time scale analysis. Note that the proposed time-domain technique has low startup overhead and can be used for short time scale thermal analysis.

Table II shows detailed CPU times for the adaptive moment matching technique. Note that the static phase of moment matching is computation and memory intensive. ISAC greatly improves computation and memory efficiency via spatial adaption. Without spatial adaption, the moment matching method would be unable to handle these benchmarks using the original 32,768 element homogeneous partitioning. In this table, columns three to five show the CPU times of the three performance bottleneck in the static phase, i.e., $\mathbf{A}$ matrix inversion, moment matrix ($\mathbf{M}$) computation, and system response coefficient ($\mathbf{H}$) computation, respectively. The CPU times associated with one moment are reported. Based on the analysis in Section III-C, with an increase of number of moments, the CPU time of matrix inversion may or may not increase depending on whether a more stringent error bound must be applied, the CPU time of moment matrix computation increases linearly, and the CPU time of $\mathbf{H}$ coefficient computation increases quadratically. As we can see, the static phase is computation intensive. However, it need be carried out only once for an IC chip-package and cooling solution. Column six shows the CPU time of the periodic phase. Compared to the proposed time-domain method, the periodic phase is fairly efficient. This phase need only be performed once for every new power profile; for long time scale power profiles, this overhead is low. Column seven shows the CPU time of the dynamic phase for each element, which is much more efficient than the proposed time-domain method. These results demonstrate that the proposed adaptive moment matching method is well-suited for long time scale thermal analysis. Using a simple design case, in which the power profile is updated with a period of 10 ms–100 ms and temperature is reported every 100 us for elements on the active layer of the IC, the adaptive moment matching technique can achieve one or two orders of magnitude speedup compared to the proposed time-domain technique.

In summary, the results in this section demonstrate that the adaptive time matching method, combined with the adaptive moment matching method, provides a highly efficient and accurate multiple time scale thermal analysis solution.

V. LIBRARY INTERFACE

The thermal analysis infrastructure described in this article has been implemented as a library with C and C++ bindings. ISAC is

- **Chip-package material layer:** Layer(double cond, double cap, double size_x, double size_y, double size_z);
- **Thermal element holding power or temperature value:**
  Element(double size_x, double size_y, double size_z, double center_x, double center_y, double center_z, double value);
- **Constructor for thermal analysis class:** ISAC(const std::vector<Layer> & chip_package, const Boundary & boundary_cond, const std::vector<Element> & power_profile);
- **Steady-state thermal analysis solver interface:** std::vector<Element> solve_static(const std::vector<Element> & power_profile);
- **Dynamic thermal analysis solver interface:**
  - std::vector<Element> init_dynamic(const std::vector<Element> & power_profile);
  - std::vector<Element> step_dynamic(const std::vector<Element> & power_profile, double duration);
- **Example of generic interface:**
  template <typename ELT_ITER> std::vector<Element> solve_static(ELT_ITER power_profile_begin, ELT_ITER power_profile_end);
- **Example of C binding interface:**
  void tams_solve_static(tams_handle h, const struct CElement * power_profile_begin, const struct CElement * power_profile_end, struct CElement ** thermal_profile_begin, struct CElement ** thermal_profile_end);

Fig. 5. Core C/C++ interface to ISAC

presently being used by teams at numerous universities working on thermal-aware IC synthesis and computer architecture.

Figure 5 shows the primary C++ binding interface to the ISAC thermal analysis system, as well examples from the secondary, more generic interface, and the C binding interface. More detailed documentation is included with the software. The *ISAC(·)* constructor creates an instance of a thermal analysis object. This constructor accepts, as input, a *Boundary* object specifying the chip-package boundary conditions. The ISAC constructor also accepts a C++ standard template library (STL) vector of *Layers*, which describes the construction of the chip and package. Each material *Layer* has a thermal conductivity, a specific heat capacity, a width, a height, and a depth. The ordering of layers in the chip and package is determined by their order in the vector. Finally, the constructor uses an initial power profile in order to determine appropriate thermal element spatial discretization.

Steady-state thermal analysis is conducted via the *solve_static(·)* method, which accepts an STL vector of thermal elements describing the chip-package power consumption profile. Each element is a rectangular parallelipiped having a width, height, depth, a position in three-dimensional space, and a power consumption value. This interface allows three-dimensional power profiles to be specified. The *solve_static(·)* method returns an STL vector of thermal elements, each of which has a temperature.

Dynamic thermal analysis is conducted via the *init_dynamic(·)* and *step_dynamic(·)* methods. *init_dynamic(·)* initializes the chip-package to the steady-state thermal profile associated with power profile argument. This method is similar to *solve_static(·)*. However, it stores the initial temperature profile as internal state to allow subsequent dynamic analysis. Starting from the current thermal profile state, *step_dynamic(·)* method determines the new thermal profile if *duration* seconds are spent with the power profile argument. This temperature profile is returned from the method and stored as internal state for further time advances.

For the sake of user convenience, we have provided a general mirror interface allowing the use of any STL container, including a plain array, to provide power profiles. This interface is otherwise identical to the primary interface.

Specifying power and thermal profiles using unordered containers of rectangular parallelipipeds is general. However, some users may want to provide power profiles, or receive thermal profiles, in homogeneous arrays. We have provided a number of generic, template-based, conversion routines to convert to and from containers of elements to two-dimensional and three-dimensional homogeneous arrays. We decided to support element basis conversion via routines instead of complicating the interface of ISAC.

Recall that ISAC is spatially adaptive. The elements in the output thermal profiles may be heterogeneous, and may not have the same positions and sizes as the elements in the input power profiles. The element basis conversion routines can also be used to find the average, minimum, and maximum temperatures encountered within three-dimensional parallelipipeds corresponding to regions, e.g., processor functional units.

VI. Conclusions

This article has described ISAC, a comprehensive solution to the IC thermal analysis problem. It has both the speed and accuracy required for use during IC synthesis. ISAC uses cooperative temporal and spatial adaptation to accelerate thermal analysis while maintaining accuracy. It unifies steady-state, time-domain, and frequency-domain thermal analysis, thereby allowing the fastest appropriate solver to be used for a given thermal analysis problem instance. ISAC contains a spatially-adaptive multigrid solver of our own design for steady-state thermal analysis, a new spatially and temporally-adaptive asynchronous time marching technique for short time scale analysis, and a new spatially adaptive frequency-domain moment matching technique for long time scale analysis. Our spatial adaptation techniques bring a 22.6×–690.0× speedup over recently published steady-state thermal analysis techniques [7]. Our unified spatial and temporal adaptation techniques, within our asynchronous time marching method, bring a 1,071×–1,890× speedup over other widely-used, time-domain thermal analysis techniques [5] with less than 0.5% error. Our spatial adaptation techniques enables the efficient use of our new frequency-domain thermal analysis technique, which brings a 10×–100× speedup over the fastest-know time-domain technique, when used for long time scale thermal analysis. ISAC efficiently and accurately solves the static, short time scale dynamic, and long time scale dynamic thermal analysis problems. Our C/C++ library implementation of ISAC is publicly available at http://post.queensu.ca/˜shangl/isac/.

References

[1] International Technology Roadmap for Semiconductors, http://public.-itrs.net.

[2] J. Srinivasan, *et al.*, "The impact of technology scaling on lifetime reliability," in *Proc. International Conf. Dependable Systems and Networks*, June 2004, pp. 177–186.

[3] P. Li, Y. Deng, and L. T. Pileggi, "Temperature-dependent optimization of cache leakage power dissipation," in *Proc. Int. Conf. Computer Design*, Oct. 2005.

[4] "COMSOL Multiphysics," http://www.comsol.com/products/multiphysics.

[5] K. Skadron, *et al.*, "Temperature-aware microarchitecture," in *Proc. Int. Symp. Computer Architecture*, June 2003, pp. 2–13.

[6] Y. Zhan and S. S. Sapatnekar, "A high efficiency full-chip thermal simulation algorithm," in *Proc. Int. Conf. Computer-Aided Design*, Oct. 2005.

[7] P. Li, *et al.*, "Efficient full-chip thermal modeling and analysis," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2004, pp. 319–326.

[8] P. Liu, *et al.*, "Fast thermal simulation for architecture level dynamic thermal management," in *Proc. Int. Conf. Computer-Aided Design*, Oct. 2005.

[9] T.-Y. Wang and C. C.-P. Chen, "SPICE-compatible thermal simulation with lumped circuit modeling for thermal reliability analysis based on modeling order reduction," in *Proc. Int. Symp. Quality Electronic Design*, Mar. 2004, pp. 357–362.

[10] L. Codecasa, D. D'Amore, and P. Maffezzoni, "An Arnoldi based thermal network reduction method for electro-thermal analysis," *Trans. Components and Packaging Technologies*, vol. 26, no. 1, pp. 168–192, Mar. 2003.

[11] T. L. Pillage, R. A. Rohrer, and C. Visweswariah, *Electronic Circuit System Simulation*. McGraw-Hill Book Company, NY, 1995.

[12] Y. Yang, *et al.*, "Adaptive chip-package thermal analysis for synthesis and design," in *Proc. Design, Automation, and Test in Europe*, Mar. 2006, pp. 844–849.

[13] G. S. Ohm, "The Galvanic circuit investigated mathematically," 1827.

[14] W. Briggs, *A Multigrid Tutorial*. SIAM Press, 1987.

[15] S. S. Rao, *Applied Numerical Methods for Engineers and Scientists*. Prentice-Hall, Englewood Cliffs, NJ, 2002.

[16] S. Winograd, "Some remarks on fast multiplication of polynomials," *Complexity of Sequential and Parallel Numerical Algorithms*, pp. 181–196, 1973.

[17] C. C. Douglas, *et al.*, "GEMMW: A portable level 3 BLAS Winograd variant of Strassen's matrix–matrix multiply algorithm," *J. Computational Physics*, vol. 110, pp. 1–10, 1994.

[18] "AMD core math library (ACML)," http://developer.amd.com/acml.aspx.