ISAC: Integrated Space and Time Adaptive Chip-Package Thermal Analysis

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

† 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*— Ever-increasing integrated circuit (IC) power densities and peak temperatures threaten reliability, performance, and economical cooling. To address these challenges, thermal analysis must be embedded within IC synthesis. However, this requires accurate three-dimensional chip-package heat flow analysis. This has typically been based on numerical methods that are too computationally intensive for numerous repeated applications during synthesis or design. Thermal analysis techniques must be both accurate and fast for use in IC synthesis.**

This article presents a novel, accurate, incremental, spatially and temporally adaptive chip-package thermal analysis technique, called ISAC, for use in IC synthesis and design. It is common for IC temperature variation to strongly depend on position and time. ISAC dynamically adapts spatial and temporal modeling granularity to achieve high efficiency while maintaining accuracy. Both steady-state and dynamic thermal analysis are accelerated by the proposed heterogeneous spatial resolution adaptation and asynchronous thermal element time-marching techniques. Each technique enables orders of magnitude improvement in performance while preserving accuracy when compared with other state-of-the-art adaptive steady-state and dynamic IC thermal analysis techniques. Experimental results indicate that these improvements are sufficient to make accurate dynamic and static thermal analysis practical within the inner loops of IC synthesis algorithms. ISAC has been validated against reliable commercial thermal analysis tools using industrial and academic synthesis test cases and chip designs. It has been implemented as a software package suitable for integration in IC synthesis and design flows and will be publicly released.

I. Introduction

With increasing integrated circuit (IC) power densities and performance requirements, thermal issues have become critical challenges in IC design [1]. If not properly addressed, increased IC temperature affects other design metrics including performance (via decreased transistor switching speed resulting from decreased charge carrier mobility and increased interconnect latency), power and energy consumption (via increased leakage power), reliability (via electromigration, thermal cycling, time-dependent dielectric breakdown, etc.), and price (via increased system cooling cost). It is thus critical to consider thermal issues during IC design and synthesis. When determining the impact of each decision in the synthesis or design process, the impacts of changed thermal profile on performance, power, price, and reliability must be considered. This requires repeated use of detailed chip-package thermal analysis. This analysis is generally based on computationally expensive numerical methods. In order to support IC synthesis, a thermal simulator must be capable of accurately analyzing models containing tens of thousands of discrete elements. Moreover, the solver must be fast enough to support numerous evaluations in the inner loop of a synthesis flow. Reliance on non-adaptive matrix operations that increase in space and time complexity superlinearly with matrix size (and model element count), has made achieving both accuracy and speed elusive.

The IC thermal analysis problem may be separated into two subproblems: steady-state (or static) analysis and dynamic analysis. Steady-state analysis determines the temperature profile to which an IC converges as time approaches infinity, given power and thermal conductivity profiles. Steady-state analysis is sufficient when an IC thermal profile converges before subsequent changes to its power profile and when transient thermal profiles, which might indicate short-term thermal peaks, may be neglected. Dynamic thermal analysis determines the temperature profile of an IC at any time given initial temperature, power, heat capacity, and thermal conductivity profiles. Although more computationally intensive than steady-state thermal analysis, dynamic thermal analysis is necessary when an IC power profile varies before its thermal profile has converged or when transient features of the thermal profile are significant.

Thermal analysis has a long history. Traditionally, thermal issues were solely addressed during cooling and packaging design based on worst-case analysis; in the past, thermal issues were typically ignored during IC design, or transfered as power constraints, e.g., a predefined peak power budget. A number of industrial tools were developed and widely used by packaging designers, such as FLOMERICS [2], ANSYS [3], and COMSOL (formerly known as FEMLAB) [4]. Since thermal analysis was conducted only a few times during the design process, efficiency was not a major concern. Typically, it took minutes or hours to conduct each simulation. Due to the increasing power density and cooling costs, such worst-case based cooling design has become increasingly difficult,

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

if not infeasible. Researchers started addressing thermal issues during IC design, for which both the efficiency and accuracy of thermal analysis are critical. Recently, Skadron et al. developed a steady-state and dynamic thermal analysis tools, called HotSpot, for microarchitectural evaluation [5]. In HotSpot, matrix operations are based on LU decomposition. Therefore, only coarse-grained modeling is supported. In addition, neither the matrix techniques of the steady-state analysis tool nor the lock-step fourth-order Runge-Kutta time-marching technique used for dynamic analysis make use of spatial or asynchronous temporal adaptation; accuracy or performance suffer. Li et al. proposed a full-chip steady-state thermal analysis method [6]. In this work, matrix operations are handled using the multigrid method, which can efficiently support fine modeling granularity with a large number of grid elements. However, although the advantages of heterogeneous element discretization is noted, in this work, no systematic adaptation method is provided. Smy et al. proposed a quad-tree mesh refinement technique for thermal analysis [7] but did not consider local temporal adaptation. Zhan and Sapatnekar [8] proposed a steady-state thermal analysis method based on the Green's function formalism that was accelerated by using discrete cosine transforms and a look-up table. However, these methods [6]–[8] do not support dynamic thermal analysis. Liu et al. proposed a moment matching based thermal analysis method suitable for accelerating thermal analysis of course-grained architectural models [9]. Numerical analysis techniques were also proposed to characterize the thermal profile of on-chip interconnect layers [10]–[12].

Existing IC thermal analysis tools are capable of providing either accuracy or speed, but not both. Accurate thermal analysis requires expensive computation for many elements in some regions, at some times. Conventional IC thermal analysis techniques ensure accuracy by choosing uniformly fine levels of detail across time and space, i.e., they use equivalent physical sizes or time step durations for all thermal elements. The large number of elements and time steps resulting from such techniques makes them computationally intensive and, therefore, impractical for use within IC synthesis. This article presents validated, synthesis-oriented IC thermal analysis techniques that differ from existing work by doing operation-by-operation dynamic adaptation of temporal and spatial resolution in order to dramatically reduce computational overhead without sacrificing accuracy. Experimental results indicate that the proposed spatial adaptation technique improves CPU time by 21.64–690.00× and that the temporal adaptation technique improves CPU time by 122.81–337.23×. Although much faster than conventional analysis techniques, the proposed techniques have been designed for accuracy even when this increases complexity and run time, e.g., by correctly modeling the dependence of thermal conductivity on temperature. These algorithms have been validated against COMSOL Multiphysics [4], a reliable commercial finite element physical process modeling package, and a high-resolution numerical spatially and temporally homogeneous simultaneous differential equation solver. Experimental results indicate that using existing thermal analysis techniques within the IC synthesis flow would increase CPU time by many orders of magnitude, making it impractical to synthesize complex ICs. The proposed techniques make accurate dynamic and static thermal analysis practical within the inner loop of IC synthesis algorithms. They have been implemented as a software tool called ISAC, which will be publicly released [13].

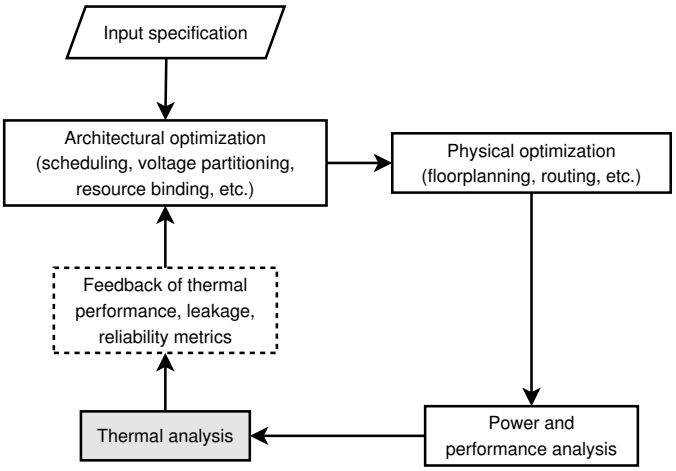


Fig. 1. Thermal-aware synthesis flow.

This article is organized as follows. Section II gives a motivating example, which illustrates the need for fast and accurate thermal analysis during IC synthesis and suggests techniques to reach this goal. Section III describes the model, algorithms, and implementation of ISAC, the proposed steady-state and dynamic thermal analysis tool. Section IV presents experimental results validating ISAC and demonstrating the dramatic performance advantages resulting from spatial and temporal adaptation during thermal analysis. Section V presents conclusions and Section VI acknowledges helpful suggestions from colleagues.

II. Motivating Examples

In this section, we use a thermal-aware IC synthesis flow to demonstrate the challenges of fast and accurate IC thermal analysis. Figure 1 shows an integrated behavioral-level and physical-level IC synthesis system [14]. This synthesis system uses a simulated annealing algorithm to jointly optimize several design metrics, including performance, area, power consumption, and peak IC temperature. It conducts both behavioral-level and physical-level stochastic optimization moves, including scheduling, voltage assignment, resource binding, floorplanning, etc. An intermediate solution is generated after each optimization move. A detailed two-dimensional active layer power profile is then reported based on the physical floorplan. Thermal analysis algorithms are invoked to guide optimization moves.

As illustrated by the example synthesis flow, for each intermediate solution, detailed thermal characterization requires full chip-package thermal modeling and analysis using computationally-intensive numerical methods. Figures 2 and 3 show a full chip-package thermal modeling example from an IBM IC design (see Section IV-A for more detail). The steady-state thermal profile of the active layer of the silicon die in conjunction with the top layer of the cooling package, shown in

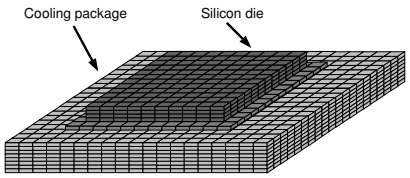


Fig. 2. Silicon chip and package.

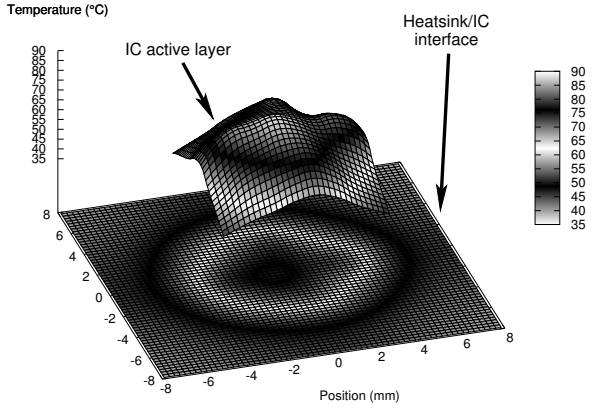


Fig. 3. Temperature profile for active layer and heatsink.

Figure 3, were characterized using a multigrid thermal solver by partitioning the chip and the cooling package into 131,072 homogeneous thermal elements. Without spatial and temporal adaptation, the solver requires many seconds or minutes when run on a high-performance workstation. Compared to steady-state thermal modeling, characterizing IC dynamic thermal profile is even more time consuming. IC synthesis requires a large number of optimization steps; thermal modeling can easily become its performance bottleneck.

A key challenge in thermal-aware IC synthesis is the development of fast and accurate thermal analysis techniques. Fundamentally, IC thermal modeling is the simulation of heat transfer from heat producers (transistors and interconnect), through silicon die and cooling package, to the ambient environment. This process is modeled with partial differential equations. In order to approximate the solutions of these equations using numerical methods, finite discretization is used, i.e., an IC model is decomposed into numerous three-dimensional elements. Adjacent elements interact via heat diffusion. Each element is sufficiently small to permit its temperature to be expressed as a difference equation that is a function of time, its material characteristics, its power dissipation, and the temperatures of its neighboring elements.

In an approach analogous to electric circuit analysis, thermal RC (or R) networks are constructed to perform dynamic (or steady-state) thermal analysis. Direct matrix operations, e.g., inversion, may be used for steady-state thermal analysis. However, the computational demand of this technique hinders its use within synthesis. Dynamic thermal analysis may be conducted by partitioning the simulation period into small time steps. The local times of all elements are then advanced, in lock-step, using transient temperature approximations yielded

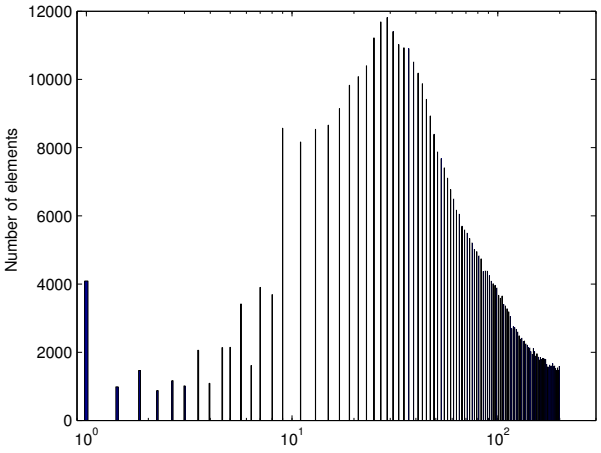


Fig. 4. Inter-element thermal gradient distribution.

by difference equations. The computational complexity of dynamic thermal analysis is a function of the number of grid elements and time steps. Therefore, to improve the efficiency of thermal modeling, the key issue is to optimize the spatial and temporal modeling granularity, eliminating non-essential elements and stages.

There is a tension between accuracy and efficiency when choosing modeling granularity. Increasing modeling granularity reduces analysis complexity but may also decrease accuracy. Uniform temperature is assumed within each thermal element; intra-element thermal gradients are neglected. Therefore, increasing spatial modeling granularity naturally increases modeling errors. Similarly, increasing time step duration may result in failure to capture transient thermal fluctuation or may increase truncation error when the actual temperature functions of some elements are of higher order than the difference equations used to approximate them.

IC thermal profiles contain significant spatial and temporal variation due to the heterogeneity of thermal conductivity and heat capacity in different materials, as well as varying power profiles resulting from non-uniform functional unit activities, placements, and schedules. Figure 4 shows the inter-element thermal temperature difference distribution using homogeneous meshing of the example shown in Figure 3. The temperature differences between all pairs of adjacent thermal elements are considered. These values are normalized to the smallest value encountered for this example. This figure contains a wide distribution of temperature differences: heterogeneous spatial element discretization refinement based on temperature differences has the potential to improve performance without impacting accuracy.

For dynamic thermal simulation, the size of each thermal element's time steps should permit accurate approximation by the element difference equations. An IC may experience different thermal fluctuations at different locations. Therefore, the best time step durations for elements at different locations may vary. Figure 5 shows the maximum potential time step duration of each individual block based on local thermal variation. These values are normalized to the smallest maximum

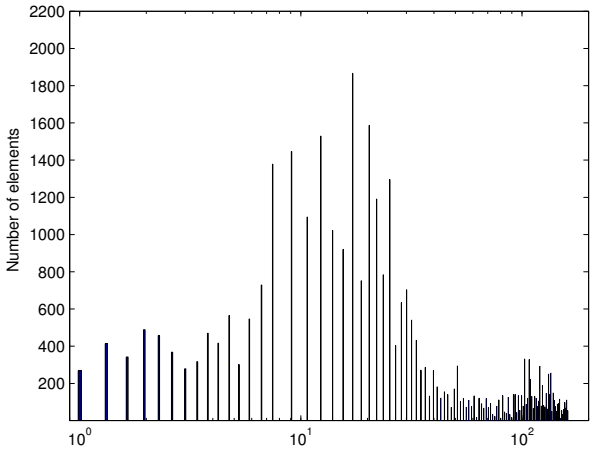


Fig. 5. Normalized maximum step size distribution.

potential time step duration of any block. Local adaptation of time step sizes has the potential to improve performance without degrading accuracy.

III. Thermal Analysis Model and Algorithms

This section gives details on the proposed thermal analysis techniques. Section III-A defines the steady-state and dynamic IC thermal analysis problems. Section III-B gives an overview of the algorithms used by ISAC, the proposed thermal analysis tool. Section III-C gives an overview of multigrid analysis and describes the spatial adaptation techniques used by ISAC to accelerate analysis. Section III-D gives an overview of time-marching and describes the temporal adaptation techniques used by ISAC. Section III-E points out the accuracy benefits of considering the dependence of thermal conductivity upon temperature within a thermal model. Finally, Section III-F explains the interface between ISAC and an IC synthesis algorithm.

A. IC Thermal Analysis Problem Definition

IC thermal analysis is the simulation of heat transfer through heterogeneous material among heat producers (e.g., transistors) and heat consumers (e.g., heat sinks attached to IC packages). Modeling thermal conduction is analogous to modeling electrical conduction, with thermal conductivity corresponding to electrical conductivity, power dissipation corresponding to electrical current, heat capacity corresponding to electrical capacitance, and temperature corresponding to voltage.

The equation governing heat diffusion via thermal conduction in an IC follows.

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

subject to the boundary condition

$$k(\vec{r},t)\frac{\partial T(\vec{r},t)}{\partial n_i} + h_i T(\vec{r},t) = f_i(\vec{r},t) \tag{2}$$

In Equation 1, ρ is the material density; c is the mass heat capacity; $T(\vec{r},t)$ and $k(\vec{r})$ are the temperature and thermal conductivity of the material at position $\vec{r}$ and time t; and $p(\vec{r},t)$ is the power density of the heat source. In Equation 2, n_i is the outward direction normal to the boundary surface i, h_i is the heat transfer coefficient; and f_i is an arbitrary function at the surface i. Note that, in reality, the thermal conductivity, k, also depends on temperature (see Section III-E). ISAC supports arbitrary heterogeneous three-dimensional thermal conduction models. For example, a model may be composed of a heat sink in a forced-air ambient environment, heat spreader, bulk silicon, active layer, and packaging material or any other geometry and combination of materials.

In order to do numerical thermal analysis, a seven point finite difference discretization method can be applied to the left and right side of Equation 1, i.e., the IC thermal behavior may be modeled by decomposing it into numerous rectangular parallelepipeds, which may be of non-uniform sizes and shapes. Adjacent elements interact via heat diffusion. Each element has a power dissipation, temperature, thermal capacitance, as well as a thermal resistance to adjacent elements. The discretized equation at an interior point of a grid element follows.

$$\begin{aligned}\rho c V \frac{T^{m+1}_{i,j,l} - T^m_{i,j,l}}{\Delta t} &= -2(G_x + G_y + G_z)T^m_{i,j,l} \\ &+ G_x T^m_{i-1,j,l} + G_x T^m_{i+1,j,l} + G_y T^m_{i,j-1,l} + G_y T^m_{i,j+1,l} \\ &+ G_z T^m_{i,j,l-1} + G_z T^m_{i,j,l+1} + V p_{i,j,l}\end{aligned} \tag{3}$$

where i, j, and l are discrete offsets along the x, y, and z axes. Given that Δx, Δy, and Δz are discretization steps along the x, y, and z axes, $V = \Delta x \Delta y \Delta z$. G_x, G_y and G_z are the thermal conductivities between adjacent elements. They are defined as follows: $G_x = k\Delta y \Delta z/\Delta x, G_y = k\Delta x \Delta z/\Delta y,$ and $G_z = k\Delta x \Delta y/\Delta z$. Δt is the discretization step in time t.

For an IC chip-package design with N discretized elements, the thermal analysis problem can be described as follows.

$$\mathbf{C}T(t)' + \mathbf{A}T(t) = Pu(t) \tag{4}$$

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; $T(t)$ and $P(t)$ are $[N \times 1]$ temperature and power vectors; and $u(t)$ is the time step function. For steady-state analysis, the left term in Equation 4 expressing temperature variation as function of time, t, is dropped. For either the dynamic or steady-state version of the problem, although direct solutions are theoretically possible, the computational expense is too high for use on high-resolution thermal models.

B. ISAC Overview

Figure 6 gives an overview of ISAC, our proposed incremental, space and time adaptive, chip-package thermal analysis tool. When used for steady-state thermal analysis, it takes, as input, a three-dimensional chip and package thermal conductivity profile, as well as a power dissipation profile. A multigrid incremental solver is used to progressively refine thermal element discretization to rapidly produce a temperature profile.

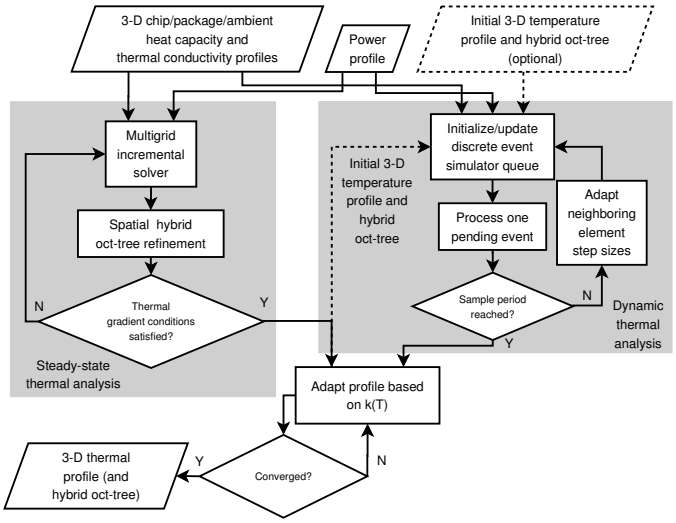


Fig. 6. Overview of ISAC.

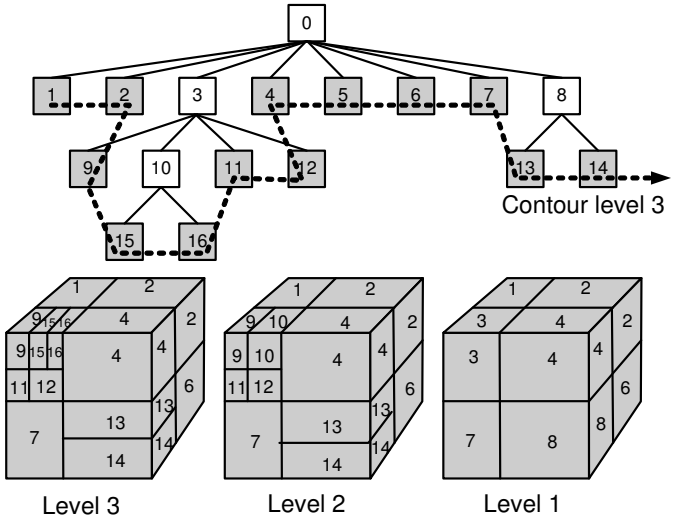


Fig. 7. Heterogeneous spatial resolution adaptation.

When used for dynamic thermal analysis, in addition to the input data required for steady-state analysis, ISAC requires the chip-package heat capacity profile. In addition, it may accept an initial temperature profile and an efficient thermal element discretization. If these inputs are not provided, the dynamic analysis technique uses steady-state analysis to produce its initial temperature profile and element discretization. It then repeatedly updates the local temperatures and times of elements at asynchronous time steps, appropriately adapting the step sizes of neighbors to maintain accuracy.

As described in Section III-E, after analysis is finished, the temperature profile may be adapted using a feedback loop in which thermal conductivity is modified based upon temperature in order to account for non-linearities induced by the dependence of the thermal conductivity or leakage power consumption on temperature. Upon convergence, the temperature profile is reported to the IC synthesis tool or designer.

C. Spatial Adaptation in Thermal Analysis

During thermal analysis, both time complexity and memory usage are linearly or superlinearly related to the number of thermal elements. Therefore, it is critical to limit the discretization granularity. As shown in Figure 4, IC thermal profiles may contain significant spatial variation due to the heterogeneity of thermal conductivity and heat capacity in different materials, as well as the variation of power profile.

In this section, we present an efficient technique for adapting thermal element spatial resolution during thermal analysis. This technique uses incremental refinement to generate a tree of heterogeneous rectangular parallelepipeds that supports fast thermal analysis without loss of accuracy. Within ISAC, this technique is incorporated with an efficient multigrid numerical analysis method, yielding a comprehensive steady-state thermal analysis solution. Dynamic thermal analysis also benefits from the proposed spatial adaptation technique due to the dramatic reduction of the number of grid elements that must be considered during time-marching simulation.

1) Hybrid Data Structure: Efficient spatial adaptation in thermal analysis relies on sophisticated data structures, i.e., it requires the efficient organization of large data sets and representation of multi-level modeling resolutions. In addition, efficient algorithms for inter-level transition are necessary for adaptive thermal modeling and numerical analysis. In ISAC, the proposed spatial adaptation technique is supported by a hybrid oct-tree data structure, which provides an efficient and flexible representation to enable spatial resolution adaptation. A hybrid oct-tree is a tree that maintains spatial relationships among rectangular parallelepipeds in three dimensions. In a hybrid oct-tree, each node may have two, four, or eight immediate children. Figure 7 shows an example of hybrid oct-tree representation. As shown in this figure, in the hybrid oct-tree, different modeling resolutions are organized into contours along the tree hierarchy. In this example, nodes (elements) 1,. . . ,8 form a level 1 contour; nodes (elements) 1,2,4,. . . ,7,9,. . . ,14 form a level 2 contour; leaf nodes (elements), shown as shaded blocks, 1,2,4,. . . ,7,10,. . . ,16 form a level 3 contour. Heterogeneous spatial resolution may result in a thermal element residing at multiple resolution levels, e.g., element 2 resides at level 1, 2, and 3. This information is represented as nodes existing in multi-level contours in the tree.

The hybrid tree structure also enables a compact representation of the thermal conductivity matrix. Within the thermal conductivity matrix, each non-zero item, $g_{i,j}$, represents the thermal conductivity between two adjacent thermal grid elements i and j. Since the hybrid tree structure contains complete connectivity information of the thermal grid elements within each contour level, it enables efficient matrix indexing, and minimizes both the memory use and the computational time required by matrix operations.

The inter-grid thermal conductivity between two adjacent thermal grid elements i and j is determined as follows.

$$g_{i,j} = \frac{1}{\frac{t_i}{k_i A_i} + \frac{t_j}{k_j A_j}} \tag{5}$$

Algorithm 1 hybrid_tree_traversal($node_{root}$)

1: **if** $node_{root}$ is a leaf node **then**
2: Add $node_{root}$ to $contour_{finest\_level}$
3: Return *finest_level*
4: **end if**
5: **for** each intermediate child chi_node_i **do**
6: $level_{chi\_node_i}$ = *hybrid_tree_traversal*(chi_node_i)
7: $level_{min}$ = min($level_{min}$, $level_{chi\_node_i}$)
8: **end for**
9: **for** each intermediate child chi_node_i **do**
10: **if** $level_{chi\_node_i} > level_{min}$ **then**
11: Add chi_node_i to $contour_{level_{chi\_node_i}-1}, \ldots, contour_{level_{min}}$
12: **end if**
13: **end for**
14: Add $node_{root}$ to $contour_{level_{min}-1}$
15: Return $level_{min}$-1

Algorithm 2 Multigrid cycle

Require: Thermal conductivity matrix $\mathbf{A}$, power profile vector P
Ensure: $\mathbf{A}T = P$
1: Pre-smoothing step: Iteratively relax initial random solution. {HF error eliminated.}
2: **subtask** Coarse grid correction
3: Compute residue from finer grid.
4: Approximate residue in coarser grid.
5: Solve coarser grid problem using relaxation.
6: **if** coarsest level has been reached **then**
7: Directly solve problem at this level.
8: **else**
9: Recursively apply the multigrid method.
10: **end if**
11: Map the correction back from the coarser to finer grid.
12: **end subtask**
13: Post smoothing step: Add correction to solution at finest grid level.
14: Iteratively relax to obtain the final solution.

where k_i and k_j are the thermal conductivities of grid elements i and j. A_i is the cross-section area of grid element i along the plane parallel to the face contacting grid element j. The converse is true of A_j. t_i and t_j are the distances from the center of each grid element to its contact surface.

Spatial resolution adaptation requires two basic operations, partitioning and coarsening. In a hybrid oct-tree, partitioning is the process of breaking a leaf node along arbitrary orthogonal axes, e.g., nodes 13 and 14 result from partitioning node 8. Coarsening is the process of merging direct sub-nodes into their parent, e.g., node 9,...,12 merged into node 3.

To conduct thermal analysis across different discretization resolutions, we propose an efficient contour search algorithm with computational complexity $\mathcal{O}(N)$, that determines thermal grid elements belonging to a particular discretization resolution level. As shown in Algorithm 1, leaf nodes are assigned to the finest resolution level (lines 1–3). The resolution level of a parent node of a subtree equals the minimal resolution level of all of its intermediate children nodes, $level_{min}$, minus one (lines 4–7 and 13). An element may reside in multiple resolution levels (lines 8–12). More specifically, in each subtree, each intermediate child node, chi_node_i, belongs to contours from $level_{min}$ to $level_{chi\_node_i}$. Algorithm 1 provides an efficient solution to traverse different spatial resolutions, thereby supporting fast multigrid thermal analysis.

2) Multigrid Method: For steady-state thermal analysis, the heat diffusion problem is (approximately) described by the following linear equation, $\mathbf{A}T = P$. The size of the thermal conductivity matrix, $\mathbf{A}$, increases quadratically with the number of discretization grid elements. Therefore, directly solving this equation is intractable. Iterative numerical methods are thus widely used. The quality of iterative methods is typically characterized by their convergence rates. Convergence rate is a function of the error field frequency [15]. Standard iterative methods, such as those of Jacobi and Gauss-Siedel have slow convergence rates due to inefficiency in removing low frequency errors. This problem becomes more prominent under fine-grain discretization.

In this work, we developed a multigrid iterative relaxation solver for steady-state thermal analysis. Multigrid methods are among the most efficient techniques for solving large scale linear algebraic problems arising from the discretization of partial differential equations [15], [16]. In conjunction with linear solvers, the multigrid method provides an efficient multi-level relaxation scheme. Using this technique, low frequency errors, which limit the performance of standard iterative methods, are transferred into the high frequency domain through grid coarsening. Algorithm 2 shows the proposed multigrid method. This method consists of a set of relaxation stages across the discretization hierarchy, where each stage is responsible for eliminating a particular frequency bandwidth of errors. Given a thermal conductivity matrix $\mathbf{A}$ and power profile P, a multigrid cycle starts from the finest granularity level (line 1), at which iterative relaxation is conducted using a linear solver to remove high frequency errors until low frequency errors dominate. Next, the solution at the finest granularity level is transformed to a coarser level, in which the original low frequency errors from the finest granularity level manifest themselves as high frequency errors. This restriction procedure is applied recursively (line 9) until the coarsest level is reached (line 6). Then, a reverse procedure, called prolongation, is used to interpolate coarser corrections back to a finer level recursively across the grid discretization hierarchy (line 11). The final result is the estimated steady-state IC chip-package thermal profile.

3) Incremental Analysis: In this work, the spatial discretization process is governed by temperature difference constraints. Iterative refinement is conducted in a hierarchical fashion. Upon initialization, the steady-state thermal analysis tool generates a coarse homogeneous oct-tree based on the chip size. Iterative temperature approximation is repeated until convergence to a stable profile. Elements across which temperature varies by more than thermal difference constraints are further partitioned into sub-elements. Given that T_i is the temperature of element i and that S is the temperature threshold, for each ordered element pair, (i, j), the new number of elements, Q, along some partition g follows.

$$Q = \left\lceil \log_2 \frac{T_i - T_j}{S} \right\rceil \tag{6}$$

For each element, i, partitions along three dimensions are gathered into a three-tuple (x_i, y_i, z_i) that governs partitioning element i into a hybrid sub oct-tree. The number of sub-elements depends on the ratio of the temperature gradient to the temperature difference threshold. Therefore, some elements may be further partitioned and local thermal simulation

repeated. Simulation terminates when all element-to-element temperature differences are smaller than the predefined threshold, S. This method focuses computation on the most critical regions, increasing analysis speed while preserving accuracy. The temperature difference threshold, S, required to trigger further thermal element partitioning is an input to ISAC. Therefore, high thresholds may be used during early design exploration in order to decrease thermal analysis time. In this article, we used a low threshold of 1 K.

D. Temporal Adaptation in Thermal Analysis

ISAC uses an adaptive time-marching technique for dynamic thermal analysis. A number of authors have written wonderful introductions to time-marching methods [17], [18]. Time-marching is a numerical method to solve simultaneous partial differential equations by iteratively advancing the local times of elements. The proposed technique is a time-marching finite difference method [17]. The computational cost of such techniques is approximately $\sum_{e \in E} u_e c_e$ where E is the set of all elements, u_e is the number of time steps for a given element, and c_e is the time cost per evaluation for that element. The Runge-Kutta family of finite difference method are commonly used to solve discretized finite difference problems such as dynamic thermal analysis [17]. For Runge-Kutta methods, assuming a constant evaluation time and noting that all elements experience the same number of updates, run time can be expressed as $uc \sum_{e \in E} n_e$ where n is the number of a block's transitive neighbors. For these methods, element time synchronization eliminates the need to repeatedly evaluate transitive neighbors, yielding a time cost of $|E|uc$.

Analysis time is classically reduced by attacking u, the number of time steps, either by using higher-order methods that allow larger time steps under bounded error or by adapting global step size during analysis, e.g., the adaptive Runge-Kutta methods. Higher-order methods allows the actual temperature function to be accurately approximated for a longer span of time, reducing the number of steps necessary to reach the target time.

1) Two popular time-marching techniques: For conventional synchronous methods, it is necessary to select a fixed time step size that is small enough to satisfy an error bound, i.e.,

$$h^f = \min_{\substack{t \in U \\ e \in E}} S_{te} \tag{7}$$

where h^f is the fixed step size to use throughout analysis, t is the time from U, the set of all explicitly visited times within the sample period, e is an element from E, the set of all elements, and S_{te} is the maximum safe step size for element e at time t. Although the weaker assurance of accuracy at the sample period would be sufficient, in practice this requires that accuracy be maintained throughout time-marching due to the dependence of element temperatures on their predecessors. Non-adaptive time-marching is used in existing popular dynamic thermal analysis packages, e.g., HotSpot [5].

Further improvement is possible via the use of a synchronous adaptive time-marching method. In such methods, the time step is adjusted such that the largest globally safe step is taken, i.e.,

$$\forall_{t \in U} h_t^s = \min_{e \in E} S_{te} \tag{8}$$

where h_t^s is the step size to be used at time t.

2) Asynchronous element time-marching: Although synchronous adaptive time-marching has the potential to outperform non-adaptive techniques, much greater gains are possible. The requirement that all thermal elements be synchronized in time implies that, at each time step, all elements must have their local times advanced by the smallest step required by any element in the model. As indicated by Figure 5, this implies that most elements are forced to take unnecessarily small steps. If, instead, it were possible to allow the thermal elements to progress forward in time asynchronously, it would be possible to allow elements for which the temperature approximation function accurately matches the actual temperature over a longer time span duration to choose larger steps. Thus,

$$\forall_{\substack{t \in U \\ e \in E}} h_{te}^a = S_{te} \tag{9}$$

where h_{te}^a is the asynchronous adaptive step size to use for element e at time t. If, at many times,

$$\sum_{e \in E} h_e^a \gg |E| h^s \tag{10}$$

i.e., the average step size is much greater than the adaptive synchronous step size, as is clearly the case for the dynamic IC thermal analysis problem (see Section II), then asynchronous element time-marching clearly holds the potential to dramatically accelerate dynamic thermal analysis compared with non-adaptive and synchronous adaptive techniques. However, reaching this potential requires that a number of problems first be identified and solved: asynchronous element time-marching increases the cost of using higher-order methods and increases the difficulty of maintaining numerical stability.

3) Impact of Asynchronous Elements on Order: Recall that thermal element temperature approximation functions depend on the temperatures of an element's (transitive) neighbors at a consistent time. Determining these temperatures is trivial in conventional synchronous time-marching techniques: all elements have the same time. However, asynchronous time-marching requires that consistency be achieved despite the differing thermal element local times.

Although many time-marching numerical methods for solving ordinary differential equations are based on methods that do not require explicit differentiation, these methods are conceptually based on repeated Taylor series expansions around increasing time instants. Revisiting these roots and basing time-marching on Taylor series expansion allows asynchronous element-by-element time step adaptation by supporting the extrapolation of temperatures to arbitrary times.

For many problems, the differentiation required for calculating Taylor series expansions is extremely complicated. Fortunately, for the dynamic IC thermal analysis problem, the problem is tractable. Noting the definitions in Equation 3, 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, N are the element's

neighbors, and M is the neighbor depth, we know that the net heat flow for a given thermal element, i, is zero.

$$0 = \sum_{n \in N_i} \left(T_i(t) - T_n \cdot u(t)\right) G_{in} + \rho_i c_i V_i \frac{dT}{dt} - p_i V_i \cdot u(t) \tag{11}$$

This can be simplified by introducing a few variables.

$$\text{Let } \alpha = \sum_{n \in N_i} G_{in} \tag{12}$$

$$\text{Let } \beta = \sum_{n \in N_i} T_n G_{in} + p_i V_i \tag{13}$$

$$\text{Let } \kappa = \rho_i c_i V_i \tag{14}$$

$$0 = T(t) \cdot \alpha - u(t) \cdot \beta + \kappa \frac{dT}{dt} \tag{15}$$

and solved for $T(t)$.

$$\mathcal{L}\left(T(t) \cdot \alpha - u(t) \cdot \beta + \kappa \frac{dT}{dt}\right) = T(s) \cdot \alpha - 1/s \cdot \beta + T(s) \cdot s \cdot \kappa - T(0^-) \cdot \kappa \tag{16}$$

$$T(s) = \frac{\beta + s \cdot T(0^-) \cdot \kappa}{s \cdot (\alpha + s \cdot \kappa)} \text{ by 15 and 16} \tag{17}$$

$$T(s) = \frac{\beta}{s \cdot (\alpha + s \cdot \kappa)} + \frac{T(0^-) \cdot \kappa}{\alpha + s \cdot \kappa} \tag{18}$$

$$\text{Let } \gamma = \frac{1}{s \cdot (\alpha + s \cdot \kappa)} \tag{19}$$

$$T(s) = \frac{T(0^-)}{s + \alpha/\kappa} + \beta \cdot \gamma \tag{20}$$

Linearity theorem for γ.

$$\frac{1}{s \cdot (\alpha + s \cdot \kappa)} = \frac{A}{s} + \frac{B}{\alpha + s \cdot \kappa} \tag{21}$$

$$a = A \cdot (\alpha + s \cdot \kappa) + B \cdot s \tag{22}$$

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

$$\gamma = \frac{1}{s \cdot (\alpha + s \cdot \kappa)} = \frac{1/\alpha}{s} - \frac{1/\alpha}{s + \alpha/\kappa} \tag{23}$$

$$T(s) = \frac{T(0^-)}{s + \alpha/\kappa} + \frac{\beta/\alpha}{s} - \frac{\beta/\alpha}{s + \alpha/\kappa} \tag{24}$$

$$\mathcal{L}^{-1}\left(\frac{T(0^-)}{s + \alpha/\kappa} + \frac{\beta/\alpha}{s} - \frac{\beta/\alpha}{s + \alpha/\kappa}\right) = u(t) \cdot \beta/\alpha + (T(0^-) - \beta/\alpha) e^{-t \cdot \alpha/\kappa} \tag{25}$$

$$\begin{matrix} T(t) \\ t \geq 0 \end{matrix} = \beta/\alpha + \left(T(0^-) - \beta/\alpha\right) e^{-t \cdot \alpha/\kappa} \tag{26}$$

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.

$$\beta_i(t, M) = \begin{cases} \sum_{n \in N_i} T_n(t, M) \cdot G_{in} + p_i V_i & \text{if } M > 0 \\ p_i V_i & \text{otherwise} \end{cases} \tag{27}$$

Thus, the nearest-neighbor approximation of temperature of element i at time $t + h$ follows.

$$T_i(t + h, M) = \beta_i(t + h, M - 1)/\alpha_i + \frac{T_i(t) - \beta_i(t + h, M - 1)/\alpha_i}{e^{(h \cdot \alpha_i)/\kappa}} \tag{28}$$

Boundary conditions are imposed by the chip, package, and cooling solution. Note that this derivation need not be carried out on-line during thermal analysis. It is done once, for an update function and the resulting equation. It is possible to use an exact local update function, such as Equation 26, or an approximation function based on low-order Taylor series expansion. In practice, we found that a first-order approximation was sufficient for local updates, as long as the impact of transitive neighbors was considered via Equations 27 and 28.

Note that the potentially differing values of step size, h, and local time, t, for all thermal elements implies that the number of transitive temperature extrapolations necessary for an element to advance by one time step may not be amortized over multiple uses as in the case in the synchronous Runge-Kutta methods. We will contrast a conventional Runge-Kutta method with ISAC to illustrate the changes necessary for asynchronous element time-marching. For the sake of explanation, consider the fourth-order Runge-Kutta method, which is used for the purpose of comparison in Section IV-B. Given that N_i is the set of block i's neighbors, p_i, T_i, T_i', and κ_i are the power consumption, current temperature, next temperature, and heat capacity of element i; G_{in} is the thermal conductivity between elements n and i; and h is the time-marching step size,

$$d_{1_i} = \frac{P_i - \sum_{n \in N_i} T_n G_{ni}}{\kappa_i} \tag{29}$$

$$d_{2_i} = \frac{P_i - \sum_{n \in N_i} \frac{(T_n + h \cdot d_{1_n}) G_{in}}{2}}{\kappa_i} \tag{30}$$

$$d_{3_i} = \frac{P_i - \sum_{n \in N_i} \frac{(T_n + h \cdot d_{2_n}) G_{in}}{2}}{\kappa_i} \tag{31}$$

$$d_{4_i} = \frac{P_i - \sum_{n \in N_i} (T_n + h \cdot d_{3_n}) G_{in}}{\kappa_i} \tag{32}$$

$$T_i' = T_i + \frac{h}{6}\left(d_{1_i} + 2d_{2_i} + 2d_{3_i} + d_{4_i}\right) \tag{33}$$

Clearly, each element update requires the computation of all d terms. This would at first seem to imply that each element temperature update requires extrapolation of the temperatures of transitive neighbors. However, because all d_{i+1} values can be computed as functions of previously-computed d_i values, the cost of computing d_i values may be amortized over many uses. This amortization allows increases in the order of Runge-Kutta and explicit synchronous time-marching methods without great increases in computational complexity. However, asynchronous thermal analysis requires the extrapolation of the temperature of a thermal element to the numerous different local times of its neighbors. This prevents the amortization described above. As a result, for three-dimensional thermal analysis using asynchronous time-marching, the number of evaluations, e, is related to the transitive neighbor count, d, as follows:

$$e = |E|\left(4/3d^3 + 2d^2 + 8/3d\right) \tag{34}$$

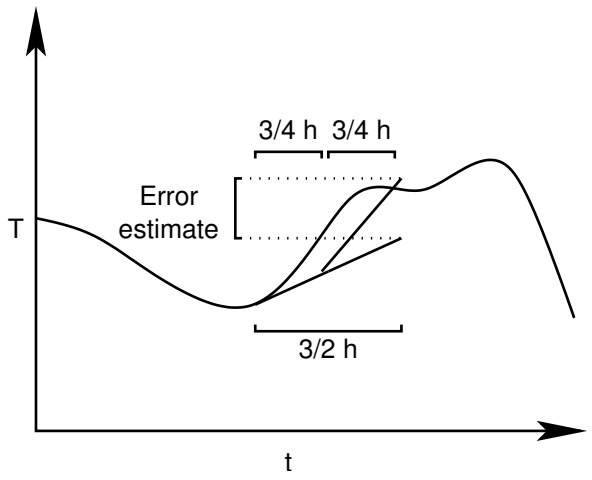


Fig. 8. Example of estimating error as a function of step size for a first-order method.

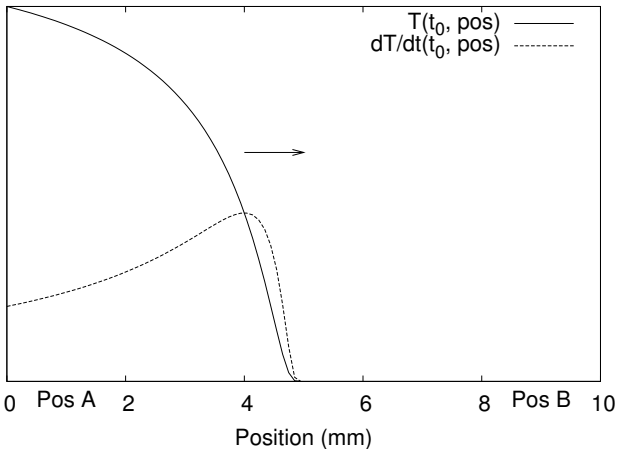


Fig. 9. Need for element local time deviation bound.

i.e., the discretized volume of the implied octahedron.

In summary, although it is common to improve the performance of time-marching techniques by increasing their orders, thereby increasing their step sizes, for the IC thermal analysis problem greater gains are possible by decoupling element local times, allowing most elements to take larger than minimum-sized steps. However, this requires explicit differentiation and prevents the amortization of neighbor temperature extrapolation, increasing the cost of using higher-order methods relative to that of using fully synchronized element time-marching techniques. As demonstrated in Section IV, this trade-off is an excellent one: the third-order element-by-element adaptation method yields speed-ups ranging from 122.81–337.23× when compared to the fourth-order adaptive Runge-Kutta method.

4) Step Size Computation: We now describe the element-by-element step size adaptation methods used by ISAC to improve performance while preserving accuracy. As illustrated in the right portion of Figure 6, dynamic analysis starts with an initial three-dimensional temperature profile and hybrid oct-tree that may have been provided by the synthesis tool or generated by ISAC using steady-state analysis; a chip/package/ambient heat capacity and thermal conductivity profile; and a power profile. After determining the initial maximum safe step sizes of all elements, ISAC initializes an event queue of elements sorted by their target times, i.e., the element's current time plus its step size. The element with the earliest target time is selected, its temperature is updated, a new maximum safe step size is calculated for the element, and it is reinserted in the event queue. The event queue serves to minimize the deviation between decoupled element current times, thereby avoiding temperature extrapolation beyond the limits of the local time bounded-order expansions. The new step size must take into account the truncation error of the numerical method in use as well as the step sizes of the neighbors. Given that h_i is element i's current step size, v is the order of the time-marching numerical method, u is a constant slightly less than one, y is the error threshold, F_i is element i's limited-order temperature approximation function, and t_i is i's current time, the safe next step size for a block, ignoring non-local effects, follows.

$$s_i(t_i) = u \times \left[\frac{y}{\left| F_i(t_i) \cdot \frac{3}{2} \cdot h_i - \frac{3}{4} \cdot h_i \left(F_i(t_i) + F_i(t_i + \frac{3}{4} \cdot h_i) \right) \right|} \right]^{1/v} \tag{35}$$

I.e., the new step size is computed by determining the absolute difference between the result of taking two $3/4h$ steps and one $3/2h$ step and dividing the error threshold by this value. This is illustrated, for a first-order method, in Figure 8. The result is then taken to the power of the reciprocal of the order of the method. Note that Equation 35 is the general expression. For the sake of example, the expression for a first-order method follows. Given that $dT_i(t)/dt$ is the derivative of i's temperature with respect to time at time t,

$$s_i(t_i) = \frac{uy}{\left| \frac{dT_i}{dt}(t_i) \cdot \frac{3}{2} \cdot h_i - \frac{3}{4} \cdot h_i \left(\frac{dT_i}{dt}(t_i) + \frac{dT_i}{dt}(t_i + \frac{3}{4} \cdot h_i) \right) \right|} \tag{36}$$

This method of computing a new step size is based on the literature [18]. However, it uses non-integer test step sizes to bracket the most probable new step size.

5) Neighborhood Time Deviation Bounds for Numerical Stability: It is necessary to further bound time-marching step sizes to ensure that the local times of neighbors are sufficiently close for accurate temperature extrapolation. For the sake of example, consider the following situation illustrated in Figure 9. To the far left of an IC active layer, at position A, the power consumption of a thermal element has recently increased. As a consequence, the temperatures of elements increase in a wave propagating rightward from position A. Note that the dashed line indicates the derivative of temperature with respect to time at time zero as a function of position, not the derivative of the temperature with respect to position.

The maximum asynchronous safe step size of an element to the far right of the IC, at position B, is large because the temperature function implied by the temperatures of other thermal elements in its neighborhood is easily approximated. For B, the rate of temperature change, based on its thermal profile of its neighborhood, is zero. Therefore, the element is

marked with an update time far in the future. Unfortunately, the temperature change resulting from A's power consumption increase may reach B's neighborhood before B's marked update time. Therefore, it is necessary to constrain the deviation of the local times of immediate neighbors in order to prevent instability due to unpredicted global effects. Given that N_i is the set of i's neighbors and w is a small constant, e.g., 3, the new step size follows.

$$h_i' = \min\left(s_i(t_i), \min_{n \in N_i}(w \cdot (t_n + h_n - t_i))\right) \quad (37)$$

For efficiency, the h_n of a neighbor at its own local time is used.

This temporal adaptation technique based upon Equations 27, 28, and 37 is general, and has been tested with first-order, second-order, and third-order numerical methods. As indicated in Section IV-B, the result is a 122.81–337.23× speedup without loss of accuracy when compared to the fourth-order adaptive Runge-Kutta method.

6) Comments on Adaptation in Simulation and Numerical Analysis: Although the asynchronous method described in this article is new, researchers have previously considered using asynchronous agents or elements in numerical simulation. Kozyakin provides a survey of, and tutorial on, asynchronous systems focusing on distributed computational networks [19] Esposito and Kumar describe event detection and synchronization methods to allow mostly-asynchronous simulation of multi-agent systems, e.g., multibody systems [20]. The work of Devgan and Roher on adaptively controlled explicit simulation is the most closely related to the proposed technique [21]. They propose using piecewise linear functions to model the responses of circuit elements. Instead of moving forward in time at a constant pace, each time step moves to the nearest time at which any circuit element becomes quiescent. In contrast, ISAC directly supports smooth functions, such as those appearing in the heat transfer problem, and allows step sizes to be adaptively controlled by error bounds instead of being imposed by the structure of the piecewise linear models used in the problem specifications.

E. Impact of Variable Thermal Conductivity

The thermal conductivity of a material, e.g., silicon, is its ratio of heat flux density to temperature gradient. It is a function of temperature, T. An ICs thermal conductivity, $k(\vec{r}, T)$, is also a function of position, $\vec{r}$. Most previous fast IC thermal analysis work ignores the dependence of thermal conductivity on temperature, approximating it with a constant. This introduces inaccuracy in analysis results. In contrast, ISAC models thermal conductivity as a function of temperature.

Position and temperature dependent thermal conductivity follows [22]:

$$k = k_0 \left(\frac{T}{300}\right)^{-\eta} \quad (38)$$

where k_0 is the material's conductivity value at temperature 300 K, η is a constant for the specific material. Recalculating the thermal conductivity value after each iteration for all the elements would be computationally expensive. In order to maintain both accuracy and performance, ISAC uses a post-processing feedback loop to determine the impact of variations in thermal conductivity upon temperature profile. As described in Section IV-A, neglecting the dependence of thermal conductivity on temperature can result in a 5 °C underestimation of peak temperature.

F. The Use of ISAC in IC Synthesis

ISAC was developed primarily for use within IC synthesis, although it may also be used to provide guidance during manual architectural decisions. ISAC may be used to solve both the steady-state and dynamic thermal analysis problems. For use in steady-state analysis, ISAC requires three-dimensional chip-package profiles of thermal conductivity and power density. The required IC power profiles are typically produced by a floorplanner used within the synthesis process [14], [23], [24]. In this application, it produces a three-dimensional steady-state temperature profile. When used for dynamic thermal analysis, ISAC requires three-dimensional chip-package profiles of temperature, power density, heat capacity, (optionally) initial temperature, and an elapsed IC duration after which to report results. In this application, it produces a three-dimensional temperature profile at any requested time.

Both steady-state and dynamic thermal analysis solvers within ISAC have been accelerated, using the techniques described in Sections III-C and III-D, in order to permit efficient use after each tentative change to an IC power profile during synthesis or design. Use within synthesis has been validated (see Section IV) by integrating ISAC within a behavioral synthesis algorithm [14].

IV. Experimental Results

In this section, we validate and evaluate the performance of ISAC. Experiments were conducted on Linux workstations of similar performance. Evaluation focuses on accuracy and efficiency. ISAC supports both steady-state and dynamic thermal analysis. Steady-state thermal analysis is validated against COMSOL Multiphysics, a widely-used commercial physics modeling package, using two actual chip designs from IBM and the MIT Raw group. Dynamic thermal simulation is validated against a fourth-order adaptive Runge-Kutta method using a set of synthesis benchmarks. Efficiency determines whether using adaptive thermal analysis during synthesis and design is tractable. To characterize the efficiency of ISAC, we compare it with alternative thermal analysis methods by conducting steady-state and dynamic thermal analysis on the power profiles produced during IC synthesis.

In all cases, convection was modeled as a thermal resistance from the top layer of the heatsink to the ambient. The thermal resistance along the heat convection path can be estimated as follows:

$$R_{convection} = \frac{1}{h_c A_s} \quad (39)$$

where h_c is the convection heat transfer coefficient, which is a function of air flow rate. A_s is the effective surface area of the heat sink.

A. Steady-State Thermal Analysis Results

This section reports the accuracy and efficiency of the steady-state thermal simulation techniques used in ISAC. We first conduct the following experiments using two actual chip designs. The first IC is designed by IBM. The silicon die is 13 mm×13 mm×0.625 mm, which is soldered to a ceramic carrier using flip-chip packaging and attached to a heat sink. A detailed 11×11 block static power profile was produced using a power simulator. The second IC is a chip-level multiprocessor designed by the MIT Raw group. This IC contains 16 on-chip MIPS processor cores organized in a 4×4 array. The die area is 18.2 mm×18.2 mm. It uses an IBM ceramic column grid array package with direct lid attach thermal enhancement. The static power profile is based on data provided in the literature [25]. We validate ISAC by comparing its results with those produced by COMSOL Multiphysics, a widely-used commercial three-dimensional finite element based physics modeling package. Table I provides thermal analysis results produced by ISAC and COMSOL Multiphysics for these ICs.

Average error, e_{avg} will be used as a measure of difference between thermal profiles:

$$e_{avg} = 1/|E| \sum_{e \in E} |T_e - T'_e| / T_e \tag{40}$$

where E is the set of elements on the surface of the active layer of the silicon die modeled by ISAC. T_e and T'_e are the temperatures of element e reported by COMSOL Multiphysics (FEMLAB) and ISAC, respectively. Percentage error is computed with the fixed point of 25°C, i.e., the ambient temperature, instead of 0 K (with apologies to purists). This is conservative; if comparisons were made relative to 0 K instead of 298.15 K, the reported percentage error would be substantially lower.

In Table I, the second and third columns show the peak and average temperatures of the surface of the active layer of the silicon dies of these chips, as reported by ISAC. Compared to COMSOL, the average errors, e_{avg}, are 3.1% and 1.0%. The next four columns show the efficiency of ISAC in terms of CPU time, speedup, memory use, and number of elements. For comparison, the next three columns show the efficiency of a multigrid analysis technique with homogeneous meshing. These results clearly demonstrate that element resolution adaptation allows ISAC to achieve dramatic improvements in efficiency compared to the conventional multigrid technique. ISAC achieves speedups of 27.50× and 690.00× relative to an efficient but homogeneous element partitioning approach. Memory usage decreases to 5.6% and 2.4% of that required by the homogeneous technique. Note that multigrid steady-state analysis itself is a highly-efficient approach [6]. Using COMSOL, both simulations take at least 20 minutes.

Existing academic IC thermal analysis tools neglect the dependence of thermal conductivity on temperature, potentially resulting in substantial errors in peak temperature. In previous work, this error was not detected during validation because the models against which they were validated also used constant values for thermal conductivity. Temperature varies through the silicon die. Therefore, ignoring the dependence of thermal conductivity on temperature may introduce significant errors. As described in Section III-E, ISAC supports modeling of temperature-dependent thermal conductivity. The last two columns of Table I show the peak and average temperatures reported by COMSOL Multiphysics when the thermal conductivity at 25 °C, i.e., room temperature, is assumed. It shows that, for both chips, the peak temperatures are underestimated by approximately 5 °C. This effect will be even more serious in designs with higher peak temperatures. Note that the source of inaccuracy is not the specific value of thermal conductivity chosen. No constant value will allow accurate results in general: an accurate IC thermal model must consider the dependence of silicon thermal conductivity upon temperature.

To further evaluate its efficiency, we use ISAC to conduct thermal analysis for the behavioral synthesis algorithm described in Section II. This iterative algorithm does both behavioral-level and physical-level optimization. In this experiment, ISAC performs steady-state thermal analysis for each intermediate solution generated during synthesis of ten commonly-used behavioral synthesis benchmarks.

Table II shows the performance of ISAC when used for steady-state thermal analysis during behavioral synthesis. The second, third, and fourth columns show the overall CPU time, speedup, and average memory used by ISAC to conduct steady-state thermal analysis for all the intermediate solutions. Column five shows the average error compared to a conventional homogeneous meshing multigrid method, the overall CPU time and average memory use of which are shown in columns six and seven. ISAC achieves almost the same accuracy with much lower run-time. The last column shows the CPU time used by the behavioral synthesis algorithm. Comparing column two and column seven makes it clear that, when used for steady-state thermal analysis, ISAC consumes only a fraction of the CPU time required for synthesis: it is feasible to use ISAC during synthesis.

B. Dynamic Thermal Analysis Results

In this section, we evaluate the accuracy and efficiency of the dynamic thermal analysis techniques used in ISAC. Heterogeneous spatial resolution adaptation was evaluated via steady-state thermal analysis. We will now focus on evaluating the proposed temporal adaptation technique. We apply this technique to second-order (ISAC-2nd-order) and third-order (ISAC) numerical methods, which are then used to conduct dynamic thermal analysis on power profiles produced by our thermal-aware behavioral synthesis algorithm during optimization. Efficiency and accuracy are compared with a fourth-order adaptive Runge-Kutta method, which uses global temporal adaptation. The CPU time of ISAC is also compared to the CPU time for IC synthesis.

We use the same set of benchmarks described in the previous section. To generate dynamic power profiles, on-line power analysis is conducted during synthesis using a switching activity model proposed in the literature [26]. For each architectural unit, input data with a Gaussian distribution are fed through an auto-regression filter to model the dependence of switching activity, and therefore power consumption, on operand bit position.

TABLE I
STEADY-STATE ARCHITECTURAL THERMAL ANALYSIS EVALUATION

Test cases	ISAC							Multigrid_HM			Const. k	
	Peak temp. (°C)	Average temp. (°C)	Error (%)	CPU time (s)	Speedup (×)	Memory use (KB)	Elements	CPU time (s)	Memory use (KB)	Elements	Peak temp. (°C)	Average temp. (°C)
IBM chip	90.7	54.8	3.1	0.08	27.50	252	1,800	2.2	4,506	32,768	85.2	53.8
MIT Raw	88.0	81.3	1.0	0.01	690.00	108	888	6.9	4,506	32,768	83.1	77.5

TABLE II
STEADY-STATE THERMAL ANALYSIS IN IC SYNTHESIS

Problem	ISAC				Multigrid_HM		HLS
	CPU time (s)	Speedup (×)	Mem. (KB)	Error (%)	CPU time (s)	Mem. (KB)	CPU time (s)
chemical	0.78	53.06	265	0.35	41.39	4,506	40.02
dct_wang	2.52	37.08	264	0.24	93.43	4,506	301.37
dct_dif	2.40	37.63	266	1.50	90.31	4,506	71.60
dct_lee	6.10	27.64	268	0.50	168.60	4,506	132.15
elliptic	2.31	32.38	267	0.43	74.79	4,506	38.07
iir77	3.35	29.27	265	0.20	98.06	4,506	77.93
jcb_sm	1.63	21.64	277	0.13	35.27	4,506	151.95
mac	0.26	79.08	264	0.12	20.56	4,506	12.32
paulin	0.13	202.85	264	0.25	26.37	4,506	4.06
pr2	8.29	22.53	285	0.55	186.75	4,506	220.81

TABLE III
DYNAMIC THERMAL ANALYSIS EVALUATION

Problem	ISAC-2nd-order			ISAC			ARK	HLS
	CPU time (s)	Error (%)	Speedup (×)	CPU time (s)	Error (%)	Speedup (×)	CPU time (s)	CPU time (s)
chemical	79.80	0.01	135.74	48.19	0.02	224.75	10831.24	82.43
dct_wang	77.56	0.05	88.20	29.35	0.05	233.06	6840.83	110.24
dct_dif	104.47	0.03	71.02	57.67	0.02	128.65	7420.05	68.15
dct_lee	360.02	0.03	61.38	179.93	0.03	122.81	22097.77	90.51
elliptic	48.68	0.02	179.75	25.95	0.02	337.23	8750.10	80.79
irr77	97.82	0.02	111.00	48.87	0.02	222.15	10857.33	110.01
jcb_sm	73.19	0.05	108.42	32.00	0.04	247.98	7935.64	160.52
mac	19.80	0.01	97.50	14.48	0.01	133.30	1930.00	29.05
paulin	16.20	0.02	109.83	10.86	0.02	163.86	1779.10	7.73
pr2	82.65	0.02	106.82	34.08	0.03	259.04	8828.50	123.43

Table III shows the experimental results for dynamic thermal analysis. For each benchmark, columns two and five show the CPU time used by ISAC-2nd-order and ISAC to conduct dynamic thermal analysis for all the intermediate solutions generated by the behavioral synthesis algorithm. Column eight shows the CPU time used by the fourth-order adaptive Runge-Kutta method. In comparison, our techniques consistently speeds up analysis by at least two orders of magnitude, as indicated by column four and column seven. As shown in column three and column six, ISAC produces results that deviate from those of the adaptive fourth-order Runge-Kutta method by no more than 0.05%. The last column shows the CPU time used by the behavioral IC synthesis algorithm. As indicated in the table, it would be impractical to use the adaptive fourth-order Runge-Kutta method within IC synthesis due to its high computational overhead. The CPU times required by the proposed thermal analysis techniques are similar to those required by IC synthesis. Therefore, it is practical to incorporate them within IC synthesis.

V. CONCLUSIONS

This article has presented spatially and temporarily adaptive techniques for steady-state and dynamic thermal analysis during IC synthesis and design. The proposed techniques were used on a number of IC designs and synthesis test cases. They were validated against COMSOL, a reliable commercial finite element physical process modeling package as well as a high-resolution spatially and temporally homogeneous solvers. Dynamic spatial and temporal adaptation result in 21.64–690.00× and 122.81–337.23× speedups, respectively, while preserving accuracy. Moreover, improvements in the underlying model, e.g., considering the dependence of thermal conductivity on temperature, have allowed accuracy improvements of 5 °C when compared with IC thermal models that neglect this dependence. The proposed techniques make accurate dynamic and static thermal analysis practical within the inner loops of IC synthesis algorithms. They have been implemented as a software tool called ISAC, which will be publicly released [13].

VI. ACKNOWLEDGMENTS

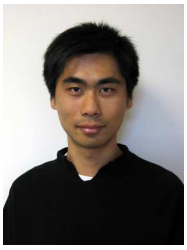
The authors would like to acknowledge Alvin Bayliss at Northwestern University for his suggestions on this work.

REFERENCES

[1] "International Technology Roadmap for Semiconductors," 2006, http:// public. itrs. net.

[2] FLOMERICS. http:// www. flomerics. com.
[3] ANSYS. http:// www. ansys. com.
[4] COMSOL Multiphysics. COMSOL, Inc. 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] P. Li, et al., "Efficient full-chip thermal modeling and analysis," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2004, pp. 319–326.
[7] T. Smy, D. Walkey, and S. Dew, "Transient 3D heat flow analysis for integrated circuit devices using the transmission line matrix method on a quad tree mesh," *Solid-State Electronics*, vol. 45, no. 7, pp. 1137–1148, July 2001.
[8] Y. Zhan and S. S. Sapatnekar, "A high efficiency full-chip thermal simulation algorithm," in *Proc. Int. Conf. Computer-Aided Design*, Oct. 2005.
[9] P. Liu, et al., "Fast thermal simulation for architecture level dynamic thermal management," in *Proc. Int. Conf. Computer-Aided Design*, Oct. 2005.
[10] T.-Y. Chiang, K. Banerjee, and K. C. Saraswat, "Analytical thermal model for multilevel VLSI interconnects incorporating via effect," *IEEE Electron Device Ltrs.*, vol. 23, no. 1, pp. 31–33, Jan. 2002.
[11] D. Chen, et al., "Interconnect thermal modeling for accurate simulation of circuit timing and relability," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 197–205, Feb. 2000.
[12] Z. Lu, et al., "Interconnect lifetime prediction under dynamic stress for reliability-aware design," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2004, pp. 327–334.
[13] Y. Yang, et al., "Incremental self-adaptive chip-package thermal analysis software," ISAC link at http:// post. queensu. ca/ shangl/ isac/ index. html and http:// robertdick. org/ projects. html.
[14] Z. P. Gu, et al., "TAPHS: Thermal-aware unified physical-level and high-level synthesis," in *Proc. Asia & South Pacific Design Automation Conf.*, Jan. 2006, pp. 879–885.
[15] P. Wesseling, *An Introduction to Multigrid Methods*. John Wiley & Sons, 1992.
[16] D. Braess, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 2001.
[17] S. S. Rao, *Applied Numerical Methods for Engineers and Scientists*. Prentice-Hall, NJ, 2002.
[18] W. H. Press, B. P. F. S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, 1992.
[19] V. S. Kozyakin, "Asynchronous systems: A short survey and problems," Institute for Information Transimission Problems: Russian Academy of Sciences, Tech. Rep., 2000.
[20] J. M. Esposito and V. Kumar, "An asynchronous integration and event detection algorithm for simulating multi-agent hybrid systems," *ACM Trans. Modeling and Computer Simulation*, vol. 14, pp. 363–388, Oct. 2004.
[21] A. Devgan and R. A. Rohrer, "Adaptive controlled explicit simulation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, pp. 746–762, June 1994.
[22] Z. Yu, et al., "Full chip thermal simulation," in *Proc. Int. Symp. Quality of Electronic Design*, Mar. 2000, pp. 145–149.
[23] J. Cong and M. Sarrafzadeh, "Incremental physical design," in *Proc. Int. Symp. Physical Design*, Apr. 2000.
[24] W. Choi and K. Bazargan, "Incremental placement for timing optimization," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003.
[25] J. S. Kim, et al., "Energy characterization of a tiled architecture processor with on-chip networks," in *Proc. Int. Symp. Low Power Electronics & Design*, Aug. 2003, pp. 424–427.
[26] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, MA, 1997.

Yonghong Yang (S'06) received her B.Sc. degree from Xiamen University, China and her M.Sc. degree from the University of Western Ontario, Canada. She is currently a Ph.D. student at Queen's University, Canada. Her research interests include computer-aided design of integrated circuits, thermal modeling, thermal optimization, and reconfigurable computing.



Zhenyu (Peter) Gu (S'04) received his B.S. and M.S. degrees from Fudan University, China in 2000 and 2003. He is currently a Ph.D. student at Northwestern University's Department of Electrical Engineering and Computer Science. Gu has published in the areas of behavioral synthesis and thermal analysis of integrated circuits.

Changyun Zhu (S'06) received his B.E. and M.E. degrees from Tsinghua University in 2002 and 2005. He is currently a Ph.D. student at Queen's University's Department of Electrical and Computer Engineering. His research interests include computer-aided design of integrated circuits, reliability modeling and optimization, and nanocomputing.

Robert P. Dick (S'95-M'02) received his B.S. degree from Clarkson University and his Ph.D. degree from Princeton University. He worked as a Visiting Researcher at NEC Labs America, a Visiting Professor at Tsinghua University's Department of Electronic Engineering, and is currently an Assistant Professor at Northwestern University's Department of Electrical Engineering and Computer Science. Robert received an NSF CAREER award and won his department's Best Teacher of the Year award in 2004. He has published in the areas of embedded system synthesis, mobile ad-hoc network protocols, reliability, behavioral synthesis, data compression, embedded operating systems, and thermal analysis of integrated circuits.

Li Shang (S'99-M'04) received his B.E. and M.E. degrees from Tsinghua University and his Ph.D. degree from Princeton University. He is currently an Assistant Professor at the Department of Electrical and Computer Engineering, Queen's University, Canada. Li has published in the areas of computer architecture, design automation, thermal/power modeling and optimization, reconfigurable computing, mobile computing, and nanocomputing. He won the Best Paper Award at PDCS'02 and his department's teaching award in 2006. He is the Walter F. Light Scholar of Queen's University.