Properties of and Improvements to Time-Domain Dynamic Thermal Analysis Algorithms

Xi Chen*, Robert P. Dick*, and Li Shang†

*EECS Department
University of Michigan
Ann Arbor, MI
{chexi, dickrp}@umich.edu

†ECEE Department
University of Colorado
Boulder, CO
li.shang@colorado.edu

***Abstract*—Temperature has a strong influence on integrated circuit (IC) performance, power consumption, and reliability. However, accurate thermal analysis can impose high computation costs during the IC design process. We analyze the performance and accuracies of a variety of time-domain dynamic thermal analysis techniques and use our findings to propose a new analysis technique that improves performance by 38–138× relative to popular methods such as the fourth-order globally adaptive Runge-Kutta method while maintaining accuracy. More precisely, we prove that the step sizes of step doubling based globally adaptive fourth-order Runge-Kutta method and Runge-Kutta-Fehlberg methods always converge to a constant value regardless of the initial power profile, thermal profile, and error threshold during dynamic thermal analysis. Thus, these widely-used techniques are unable to adapt to the requirements of individual problems, resulting in poor performance. We also determine the effect of using a number of temperature update functions and step size adaptation methods for dynamic thermal analysis, and identify the most promising approach considered. Based on these observations, we propose FATA, a temporally-adaptive technique for fast and accurate dynamic thermal analysis.**

I. Introduction and Past Work

Temperature has a strong influence on integrated circuit (IC) performance, power consumption, and reliability. Accurate and fast thermal analysis can therefore benefit IC design and control. Thermal analysis can be separated into two subproblems: steady-state thermal analysis and dynamic thermal analysis. Steady-state thermal analysis determines the thermal profile (i.e., a temperature at each physical position) as time proceeds to infinity resulting from a power profile (i.e., a power consumption at each physical position). Dynamic thermal analysis determines the thermal profile as a function of time resulting from time-varying power profiles. Although more computationally intensive, dynamic thermal analysis is necessary when the power profile varies before the thermal profile converges and to reliably detect transient violation of thermal constraints. This paper will focus on dynamic techniques.

IC dynamic thermal analysis models the thermal conduction from an IC's power sources through cooling packages to the ambient environment, usually using partial differential equations. In order to approximate the solutions to these equations using numerical methods, an IC model is decomposed into a large number of thermal elements such that the thermal variation within an element is negligible. This permits accurate thermal simulation. The resulting large system matrix makes direct solutions such as LU decomposition prohibitive. Traditionally, the dynamic thermal analysis problem can be solved using either frequency-domain or time-domain techniques [1], [2], [3]. Compared with frequency-domain techniques, time-domain techniques are fast and accurate for the shorter simulation runs typically encountered when power profiles frequently change. This paper focuses on time-domain techniques.

This work was supported in part by NSF under awards CCF-0702761, CNS-0347941, and CNS-0720820 and in part by SRC under awards 2007-HJ-1593 and 2007-TJ-1589.

A number of researchers have worked on the time-domain dynamic thermal analysis problem. Skadron et al. developed HotSpot [2], which uses an adaptive fourth-order Runge-Kutta method that dynamically adjusts the step size according to the local error at each time step to solve the finite difference equations. However, it is a synchronous time marching method: all the thermal elements have the same step size at each time point. Recently Yang et al. [3] developed an IC thermal analysis technique called ISAC. This technique adapts to spatial and temporal variation in material properties and power profiles. Although it achieved speedups over the fourth-order Runge-Kutta method in some circumstances, its assumption about the temperatures of the neighbors when solving finite difference equations is inaccurate (see Section IV), often reducing performance and/or accuracy. We know of no work that does a thorough analysis of the properties of commonly used time-domain thermal analysis techniques or points out fundamental problems with using popular finite difference techniques, such as adaptive Runge-Kutta methods, to solve the dynamic thermal analysis problem.

This paper makes the following contributions. First, it proves that the step size will always converge to a constant value for (a) a step doubling based globally adaptive fourth-order Runge-Kutta (GARK4) method and (b) Runge-Kutta-Fehlberg method regardless of the initial power profile, thermal profile, and error threshold during dynamic thermal analysis. This reveals a fundamental and surprising performance limitation when these techniques are used for thermal analysis. Second, we propose FATA, a new fast asynchronous dynamic thermal analysis technique. This technique improves performance by 38–138× relative to popular methods such as the GARK4 method [2] and by 118.59–222.60× relative to existing work in asynchronous time-domain thermal analysis [3] with similar accuracy. Third, this article indicates the impact of various combinations of temperature update functions and step adaptation methods on performance and accuracy. Our analysis suggests that combining the trapezoidal method with third-order finite temperature difference based step size adaptation yields the best combination of performance and accuracy.

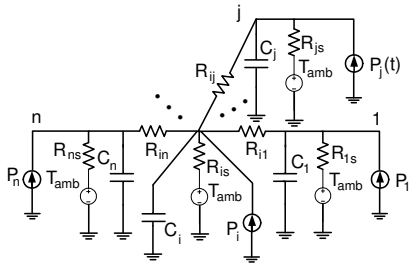

Figure 1. Model for a single thermal element.

II. Thermal Model and Problem Formulation

In this section, we first give details on the IC thermal model that characterizes the heat transfer among IC, package, heatsink, and the ambient environment. We then formulate the dynamic thermal analysis problem in the matrix-vector form.

Heat and electrical conduction are both governed by the diffusion equation and can be similarly modeled; thermal conductance corresponds to electrical conductance, heat capacity corresponds to electrical capacitance, temperature corresponds to voltage, and power dissipation corresponds to electrical current. We model a chip as a regular mesh containing N discretized elements, or a *thermal grid*, with the ambient temperature corresponding to ground. Each element has a ground capacitance and a ground thermal conductance. In the case where no heat dissipation channel exists between a thermal element and the ambient, the corresponding ground thermal conductance is set to zero. Physically adjacent elements are connected with resistors. The power consumption of each thermal element is modeled as a current source with current flowing into the element. Using this model, the thermal grid can be modeled as a linear system.

Figure 1 illustrates the model for a given element i, which is connected to n neighbors via resistors. Ground represents the ambient temperature. We use N_i to represent element i's neighbors. Given $n \in N_i$, $T_i(t)$ is the temperature of element i at time t, $T_n(t)$ is the temperature of element i's neighbor n, T_{amb} is the ambient temperature, C_i is the ground capacitance at element i, $P_i(t)$ is the heat source associated with element i, R_{is} is the ground thermal resistance at element i, and R_{in} is the resistance between element i and its neighbor n, Kirchhoff's Current Law can be used to derive the following equation:

$$\left(\sum_{n\in N_i}\frac{T_i(t)-T_n(t)}{R_{in}}\right)+\frac{T_i(t)-T_{amb}}{R_{is}}+C_i\frac{dT_i(t)}{dt}-P_i(t)=0. \quad (1)$$

Equation 1 can be simplified as follows:

$$\text{Letting } T_i' = T_i - T_{amb}, \forall i \text{ in the thermal grid and} \quad (2)$$

$$\alpha_i = \sum_{n\in N_i}\frac{1}{R_{in}}+\frac{1}{R_{is}}, \quad (3)$$

$$\frac{dT_i'(t)}{dt} = \frac{1}{C_i}\left(\sum_{n\in N_i}\frac{T_n'(t)}{R_{in}}-\alpha_i T_i'(t)+P_i(t)\right). \quad (4)$$

For convenience, we will use $T_i(t)$ instead of $T_i'(t)$ to represent the normalized temperature of element i, i.e., the difference between element i's temperature and the ambient temperature. Assume the jth neighbor of element i is k_j, we define $\overrightarrow{g_i}$ as

$$\overrightarrow{g_i} = \left(\underbrace{0,\cdots,0}_{k_1-1},\frac{1}{R_{i1}},\underbrace{0,\cdots,0}_{k_2-1},\frac{1}{R_{i2}},\cdots,-\alpha,\cdots,\frac{1}{R_{in_i}},\cdots,0\right), \quad (5)$$

where $\frac{1}{R_{ij}}$ is the k_jth entry of vector $\overrightarrow{g_i}$, representing the thermal conductance between i and its jth neighbor k_j, $-\alpha$ (defined in Equation 3) is the ith entry of the vector, and all other entries are 0s. Similarly, we use $\overrightarrow{T(t)}$ and $\overrightarrow{T^{(1)}(t)}$ to represent the temperature and first-order temperature derivatives of all N elements in the system at time t, i.e.,

$$\overrightarrow{T(t)} = (T_1(t), T_2(t), \cdots, T_N(t))^T \text{ and} \quad (6)$$

$$\overrightarrow{T^{(1)}(t)} = (dT_1(t)/dt, dT_2(t)/dt, \cdots, dT_N(t)/dt)^T. \quad (7)$$

Hence, Equation 4 can be written as

$$\frac{dT_i(t)}{dt} = \left(\overrightarrow{g_i}\cdot\overrightarrow{T(t)}+P_i(t)\right)/C_i. \quad (8)$$

Equation 4 holds for all elements in the system. If we define system matrix $\mathbf{A}$ (size $N\times N$), thermal capacitance matrix $\mathbf{C}$ (size $N\times N$), and N-dimensional power vector $\overrightarrow{P(t)}$ as

$$\mathbf{A} = \left(\overrightarrow{g_1}^T, \overrightarrow{g_2}^T, \cdots, \overrightarrow{g_N}^T\right)^T, \quad (9)$$

$$\mathbf{C} = diag\left(C_1, C_2, \cdots, C_N\right), \text{and} \quad (10)$$

$$\overrightarrow{P(t)} = (P_1(t), P_2(t), \cdots, P_N(t))^T, \quad (11)$$

the dynamic thermal analysis problem can be described as follows: $\overrightarrow{T^{(1)}(t)} = \mathbf{C}^{-1}\left(\mathbf{A}\overrightarrow{T(t)}+\overrightarrow{P(t)}\right)$. However, it is computationally expensive to directly solve the system equation due to the high order of system matrices $\mathbf{A}$ and $\mathbf{C}$ for a typical thermal model with tens of thousands of discrete elements. A common approach is to divide time into discrete time steps and approximate the solutions using finite temperature difference equations, i.e., using finite difference methods.

III. Globally Adaptive Runge-Kutta Methods: Are They Really Adaptive?

Traditionally, finite difference equations are solved by synchronous numerical methods. Runge-Kutta methods such as the fourth-order Runge-Kutta method and the Runge-Kutta-Fehlberg (RKF) method are commonly used solve ordinary differential equations [2], [3]. We have determined that, despite their popularity, these step size control methods are not appropriate for thermal analysis.

Theorem 3.1 (Step Size Convergence Property): Given an IC thermal model with N thermal elements that satisfy Equation 4, the step size of (1) a step doubling based globally adaptive 4th-order Runge-Kutta (GARK4) method and (2) RKF method converge to a constant value c_h regardless of initial step size and specified error threshold.

Proof: Since the proofs for both methods are similar, we present only the proof for the synchronous, adaptive GARK4 method used in HotSpot [2]. We assume the power profile, $\overrightarrow{P}$, is constant during a simulation run. Given that $\overrightarrow{T_k}$ is the temperature vector at iteration k, $\overrightarrow{T_k^{(1)}}$ is the first-order temperature derivative at iteration k, and h_{k+1} is the step size at iteration $k+1$, the temperature vector at iteration $k+1$, i.e.,

$\overrightarrow{T_{k+1}}$ can be expressed as follows:

$$\begin{aligned}
\overrightarrow{k_1} &= \overrightarrow{T_k^{(1)}} = \mathbf{C}^{-1}\left(\mathbf{A}\overrightarrow{T_k} + \overrightarrow{P}\right), && (12)\\
\overrightarrow{k_2} &= \mathbf{C}^{-1}\left[\mathbf{A}\left(\overrightarrow{T_k} + 1/2h_{k+1}\overrightarrow{k_1}\right) + \overrightarrow{P}\right]\\
&= \overrightarrow{k_1} + 1/2h_{k+1}\mathbf{C}^{-1}\mathbf{A}\overrightarrow{k_1}, && (13)\\
\overrightarrow{k_3} &= \overrightarrow{k_1} + 1/2h_{k+1}\mathbf{C}^{-1}\mathbf{A}\overrightarrow{k_2}, \text{ and} && (14)\\
\overrightarrow{k_4} &= \overrightarrow{k_1} + h_{k+1}\mathbf{C}^{-1}\mathbf{A}\overrightarrow{k_3}, \text{ yielding} && (15)\\
\overrightarrow{T_{k+1}} &= \overrightarrow{T_k} + 1/6h_{k+1}\left(\overrightarrow{k_1} + 2\overrightarrow{k_2} + 2\overrightarrow{k_3} + \overrightarrow{k_4}\right). && (16)
\end{aligned}$$

Starting from Equations 12, 13, 14, and 15, substitute the corresponding terms into Equation 16, resulting in

$$\overrightarrow{T_{k+1}} = \overrightarrow{T_k} + \sum_{n=1}^{4}\frac{h_{k+1}^n}{n!}(\mathbf{C}^{-1}\mathbf{A})^{n-1}\overrightarrow{k_1}. \quad (17)$$

Next, we present the temperature and step size update functions in the matrix-vector form. Using Equation 12, we introduce a few variables to simplify Equation 17.

$$\begin{aligned}
\text{Let } \mathbf{B} &= \mathbf{C}^{-1}\mathbf{A} \text{ and } \mathbf{D} = \mathbf{C}^{-1}\overrightarrow{P}.\\
\overrightarrow{T_{k+1}} &= \overrightarrow{T_k} + \left(\mathbf{B}\overrightarrow{T_k} + \mathbf{D}\right)\sum_{n=1}^{4}\frac{h_{k+1}^n}{n!}\mathbf{B}^{n-1}\\
&= \sum_{n=0}^{4}\frac{(h_{k+1}\mathbf{B})^n}{n!}\overrightarrow{T_k} + \sum_{n=1}^{4}\frac{(h_{k+1}^n\mathbf{B}^{n-1})}{n!}\mathbf{D}. \quad (18)
\end{aligned}$$

Given that $f_{\mathbf{B}}(h) = \sum_{n=0}^{4}\frac{(h\mathbf{B})^n}{n!}$, the temperature vector at iteration $k+1$ is

$$\overrightarrow{T_{k+1}} = f_{\mathbf{B}}(h_{k+1})\overrightarrow{T_k} + (f_{\mathbf{B}}(h_{k+1}) - \mathbf{I}_{N\times N})\mathbf{B}^{-1}\mathbf{D}, \quad (19)$$

where $\mathbf{I}_{N\times N}$ is a $N \times N$ unit matrix. Given that $\overrightarrow{T_k}$ and h_k are the temperature vector and step size at iteration k, step doubling based step size adaptation first determines the absolute difference between the results computed by taking one h_k step and two $\frac{h_k}{2}$ steps, where $\overrightarrow{T_{k+1,s}}$ is the temperature vector at iteration $k+1$ using one step and $\overrightarrow{T_{k+1,d}}$ is the predicted temperature vector at iteration $k+1$ using two steps.

$$\begin{aligned}
\overrightarrow{T_{k+\frac{1}{2}}} &= f_{\mathbf{B}}(\frac{h_k}{2})\overrightarrow{T_k} + (f_{\mathbf{B}}(\frac{h_k}{2}) - \mathbf{I}_{N\times N})\mathbf{B}^{-1}\mathbf{D}, && (20)\\
\overrightarrow{T_{k+1,s}} &= f_{\mathbf{B}}(h_k)\overrightarrow{T_k} + (f_{\mathbf{B}}(h_k) - \mathbf{I}_{N\times N})\mathbf{B}^{-1}\mathbf{D}, && (21)\\
\overrightarrow{T_{k+1,d}} &= f_{\mathbf{B}}(\frac{h_k}{2})\overrightarrow{T_{k+\frac{1}{2}}} + (f_{\mathbf{B}}(\frac{h_k}{2}) - \mathbf{I}_{N\times N})\mathbf{B}^{-1}\mathbf{D} && (22)
\end{aligned}$$

The step size for the next iteration, h_{k+1}, is calculated by dividing the error threshold by that difference, as shown below:

$$h_{k+1} = h_k \times \left(\epsilon/||\overrightarrow{T_{k+1,d}} - \overrightarrow{T_{k+1,s}}||_\infty\right)^{\frac{1}{5}}, \quad (23)$$

where ϵ is a user-specified error threshold used to control accuracy. Combining Equations 20, 22, and 21 yields the next safe step size:

$$h_{k+1} = h_k \times \left(\epsilon/||[f_{\mathbf{B}}^2(\frac{h_k}{2}) - f_{\mathbf{B}}(h_k)](\overrightarrow{T_k} + \mathbf{B}^{-1}\mathbf{D})||_\infty\right)^{\frac{1}{5}}. \quad (24)$$

Equation 19 and Equation 24 can be simplified by defining $\overrightarrow{Y_k} = \overrightarrow{T_k} + \mathbf{B}^{-1}\mathbf{D}$, yielding the following temperature and step size update functions:

$$\begin{aligned}
h_{k+1} &= h_k \times \left(\epsilon/||[f_{\mathbf{B}}^2(\frac{h_k}{2}) - f_{\mathbf{B}}(h_k)]\overrightarrow{Y_k}||_\infty\right)^{\frac{1}{5}} && (25)\\
\overrightarrow{Y_{k+1}} &= f_{\mathbf{B}}(h_{k+1})\overrightarrow{Y_k}. && (26)
\end{aligned}$$

We now prove that the numerical solution of $\overrightarrow{Y}$ converges to a constant vector regardless of the user-specified error threshold ϵ. This will be used to prove step size convergence, i.e., h_k will converge to a constant. Due to the characteristics of RC linear systems, the exact temperature vector (or $\overrightarrow{Y_{true}}$) will become stable as time proceeds to infinity, i.e., $\lim_{k\to\infty}|\overrightarrow{Y_{k+1,true}} - \overrightarrow{Y_{k,true}}| = 0$. Given that step doubling is used to control the step size, we have $|\overrightarrow{Y_k} - \overrightarrow{Y_{k,true}}| = O(h_k^6)$ and $|\overrightarrow{Y_{k+1}} - \overrightarrow{Y_{k+1,true}}| = O(h_{k+1}^6)$, where $\overrightarrow{Y_k}$ and $\overrightarrow{Y_{k+1}}$ are obtained by GARK4 at iterations k and $k+1$ [4]. Therefore, when the thermal profile reaches steady state,

$$\begin{aligned}
|\overrightarrow{Y_{k+1}} - \overrightarrow{Y_k}| &= |\overrightarrow{Y_{k+1}} - \overrightarrow{Y_{k+1,true}} + \overrightarrow{Y_{k+1,true}} - \overrightarrow{Y_{k,true}} +\\
&\quad \overrightarrow{Y_{k,true}} - \overrightarrow{Y_k}|\\
&\le |\overrightarrow{Y_{k+1}} - \overrightarrow{Y_{k+1,true}}| +\\
&\quad |\overrightarrow{Y_{k+1,true}} - \overrightarrow{Y_{k,true}}| + |\overrightarrow{Y_{k,true}} - \overrightarrow{Y_k}| && (27)\\
&= O(h_k^6) + O(h_{k+1}^6). && (28)
\end{aligned}$$

As shown later in the section, the steady-state step size is on the order of 10^{-6}, i.e., the difference between two consecutive $\overrightarrow{Y}$ vectors, is on the order of 10^{-36}, which is dominated by numerical error. Hence, we assume $\lim_{k\to\infty}\overrightarrow{Y_k} = \overrightarrow{c_Y}$, where $\overrightarrow{c_Y}$ is a constant vector. If we use c_h to represent the step size in steady state, Equation 26 gives us

$$\overrightarrow{c_Y} = \lim_{k\to\infty}\overrightarrow{Y_{k+1}} = f_{\mathbf{B}}(c_h)\lim_{k\to\infty}\overrightarrow{Y_k} = f_{\mathbf{B}}(c_h)\overrightarrow{c_Y}. \quad (29)$$

Therefore, $f_{\mathbf{B}}(c_h)$ has an eigenvalue of 1, with $\overrightarrow{c_Y}$ being one of the corresponding eigenvectors. Note that this argument still holds in the presence of a typical numerical error of 10^{-16}. We omit the numerical analysis here due to space limitations. According to Equation 29, $\overrightarrow{c_Y} = f_{\mathbf{B}}(c_h)\overrightarrow{c_Y} = \cdots = \lim_{k\to\infty} f_{\mathbf{B}}^k(c_h)\overrightarrow{c_Y}$. Based on the matrix analysis theory on the relationship between convergence of matrix powers and eigenvalues, given that $\lambda_1(f_{\mathbf{B}}(h)), \lambda_2(f_{\mathbf{B}}(h)), \cdots, \lambda_N(f_{\mathbf{B}}(h))$ are the N eigenvalues of $f_{\mathbf{B}}(h)$, $\lim_{k\to\infty} f_{\mathbf{B}}^k(h)$ exists if and only if $\max_{1\le i\le N}|\lambda_i(f_{\mathbf{B}}(h))| \le 1$, with $|\lambda_i(f_{\mathbf{B}}(h))| = 1$ only if $\lambda_i(f_{\mathbf{B}}(h))$ is not defective and $\lambda_i(f_{\mathbf{B}}(h)) = 1$ [5].

We then determine the relationship between c_h and $\max_{1\le i\le N}|\lambda_i(f_{\mathbf{B}}(h))|$. $\mathbf{A}$ is real and symmetric because $R_{ij} = R_{ji}, \forall 1 \le i,j \le N$. We also notice that each row of matrix $\mathbf{A}$ satisfies

$$\forall i : |a_{ii}| - \sum_{j=1,j\neq i}^{N}|a_{ij}| = \frac{1}{R_{is}} \ge 0. \quad (30)$$

Furthermore, there exists at least one positive thermal conductance $\frac{1}{R_{ks}}$ between element k and the ambient such that the heat flow into the chip can be conducted through R_{ks} to the ambient, i.e., $|a_{kk}| > \sum_{j=1,j\neq k}^{N}|a_{kj}|$. Therefore, $\mathbf{A}$ is a Hermitian diagonally dominant matrix with negative diagonal elements and non-positive off-diagonal elements. This indicates $\mathbf{A}$ is invertible and negative semi-definite. Define diagonal matrix $\mathbf{Q}$ to have an ith element that is the reciprocal of the square root of the ith element of the thermal capacitance

matrix $\mathbf{C}$, i.e., $\forall_{1\leq i\leq N}: \ q_{ii} = \frac{1}{\sqrt{c_{ii}}}$. Note that $\mathbf{Q}^{-1}\mathbf{C}^{-1} = \mathbf{Q}$. Thus, $\mathbf{Q}^{-1}\mathbf{B}\mathbf{Q} = \mathbf{Q}^{-1}\mathbf{C}^{-1}\mathbf{A}\mathbf{Q} = \mathbf{Q}\mathbf{A}\mathbf{Q} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$. Since $\mathbf{A}$ is negative semi-definite, $\mathbf{Q}^T\mathbf{A}\mathbf{Q}$ is also negative semi-definite. Given $\mathbf{Q}^T\mathbf{A}\mathbf{Q}$ is also real and symmetric, its eigenvalues are non-defective and non-positive. It has no zero eigenvalues because it is invertible ($\mathbf{A}$ is invertible). Therefore, $\mathbf{Q}^T\mathbf{A}\mathbf{Q}$ is diagonalizable with negative eigenvalues. $\mathbf{B}$ is similar to $\mathbf{Q}^T\mathbf{A}\mathbf{Q}$, $\mathbf{B}$ is also diagonalizable with negative eigenvalues, i.e., there exists an invertible matrix $\mathbf{P}$ such that $\mathbf{B} = \mathbf{P} \times \Lambda(\mathbf{B}) \times \mathbf{P}^{-1}$, where $\Lambda(\mathbf{B})$ is the diagonal matrix consisting of $\mathbf{B}$'s eigenvalues. Thus, we can express $f_{\mathbf{B}}(h)$ as follows: $f_{\mathbf{B}}(h) = \mathbf{P}\sum_{n=0}^{4}\frac{(h\Lambda(\mathbf{B}))^n}{n!}\mathbf{P}^{-1}$. Hence, if $\lambda_i(\mathbf{B})$ is the ith eigenvalue of $\mathbf{B}$, the ith eigenvalue of $f_{\mathbf{B}}(h)$ is $\lambda_i(f_{\mathbf{B}}(h)) = \sum_{n=0}^{4}\frac{(h\lambda_i(\mathbf{B}))^n}{n!}$. Since function $f(x) = \sum_{n=0}^{4}\frac{x^n}{n!}$ is monotonically decreasing when $x < 0$, we have $f(x) = 1 \Leftrightarrow x = -2.785$ when x is negative. We know (1) 1 is an eigenvalue of $f_{\mathbf{B}}(c_h)$ and (2) $h\lambda_i(\mathbf{B}) < 0, 1 \leq i \leq N$, so the steady-state step size is

$$c_h = 2.785/\max_{1\leq i\leq N}|\lambda_i(\mathbf{B})|. \tag{31}$$

Note that $\max_{1\leq i\leq N}|\lambda_i(\mathbf{B})|$ can be found numerically, e.g., using von Mises' power method [6]. Thus, we have proven that the step size of the GARK4 method with step doubling based step size adaptation converges to c_h regardless of initial thermal profile and error threshold. This conclusion has been validated using different thermal grid structures in Hotspot 4.0 [2] and ISAC [3]. Note that this argument also holds for step doubling based low-order explicit methods such as the forward Euler, other variants of step doubling adaptation [2], and RKF. For example, HotSpot uses a variant of step doubling method in which auxiliary bounds are imposed on the maximum and minimum safe step size relative to the current step size. However, the constraints have no impact on the steady-state thermal behavior, i.e., Equations 29 and 31 still hold, leading to the same steady-state step size. A similar proof also applies to RKF. ■

Implications of Step Size Convergence Property

In each iteration, the step size adaptation function calculates a new safe step size based on the current thermal profile activity. Intuition suggests that it will use small steps when the chip temperature is rapidly changing and large step sizes when there is little change in temperature. When the IC thermal profile reaches steady state, the temperature function can be accurately approximated with very large step sizes. Therefore, the steady-state step size will generally be the longest encountered during dynamic thermal analysis. We validated our conclusion using different benchmarks and different grid structures (see Section V-A). We found that the percentage of step sizes exceeding c_h (the step size after temperature convergence) ranges from 0.3% to 0.8%, i.e., the steady-state step size is only rarely exceeded. In summary, the step sizes of the step doubling based GARK4 method converge to c_h long before IC thermal profile reaches steady state, significantly degrading performance.

Steady-State Step Size Analysis

In this section, we give a rough estimate of the steady-state step size using the thermal resistances and thermal capacitances in the thermal grid based on Equation 31. Similar analysis can also be applied to RKF method. Given $\text{trace}(\mathbf{B}) = \sum_{i=1}^{N} b_{ii} = \sum_{i=1}^{N}\lambda_i(\mathbf{B}) \leq N \times \max_{1\leq i\leq N}|\lambda_i(\mathbf{B})|$, we can estimate the spectral radius of $\mathbf{B}$ as $\max_{1\leq i\leq N}|\lambda_i(\mathbf{B})| \geq \frac{\sum_{i=1}^{N} b_{ii}}{N}$. We define $\frac{1}{\tau_{avg}} = \frac{\sum_{1\leq i,j\leq N, i\neq j}\frac{1}{R_{ij}C_{ii}}}{m}$, where m is the number of non-zero off-diagonal entries in $\mathbf{B}$. Since $\sum_{i=1}^{N} b_{ii} = \sum_{1\leq i,j\leq N, i\neq j}\frac{1}{R_{ij}C_{ii}} = \frac{m}{\tau_{avg}}$, the estimated steady-state step size $c_{h,s}$ is

$$c_{h,s} = \frac{2.785}{\max_{1\leq i\leq N}|\lambda_i(\mathbf{B})|} \leq \frac{2.785}{\frac{m}{N\tau_{avg}}} = \frac{2.785N}{m}\tau_{avg}. \tag{32}$$

Note that τ_{avg} is the harmonic mean of the RC constants of different thermal elements. In our validation experiments, $\frac{m}{N} \approx 6$. Therefore, $c_{h,s} \approx 0.464\tau_{avg}$. Furthermore, τ_{avg} usually underestimates the RC constant of the chip. In our test cases, τ_{avg} is approximately 10 µs while the thermal RC constant associated with the IC, i.e., τ_{IC}, is on the order of 100 µs [7], i.e., $\tau_{IC} \approx 10\tau_{avg}$. This indicates the stable step size is approximately $\frac{1}{10} \times 0.464\tau_{IC} \approx 0.05\tau_{IC}$, even when the thermal profile is perfectly approximated by the temperature update function, i.e., the thermal profile is stable. Hence, the steady-state step size is severely limited for explicit step-doubling GARK4 methods.

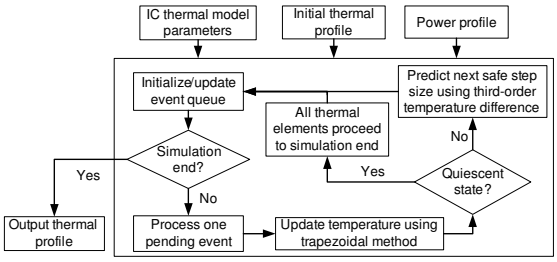

Figure 2. Overview of asynchronous time marching algorithm in FATA.

IV. FATA: Fast Asynchronous Time Marching Technique

In this section, we first give an overview of the proposed technique: FATA. We then describe the major components in FATA that enable fast and accurate simulation.

IV.A. Algorithm Overview

FATA is an adaptive asynchronous time marching finite-difference method. Compared to a synchronous method, FATA permits different elements to have different step sizes with their own local times. Figure 2 illustrates the algorithm in FATA. During dynamic thermal analysis, the algorithm takes, as input, an initial thermal profile, a power profile, and various IC thermal model parameters such as material thermal conductivities and heat capacities. After determining the initial step size for each element, FATA initializes an event queue containing temperature update events sorted by their target times, i.e., the element's current time plus its step size. In each iteration, the event with the earliest target time is selected and the corresponding element's temperature is updated. It then determines whether the thermal profile of the chip has reached quiescent state and if so, advances the local times of all the thermal elements to the user specified simulation end time. Otherwise, FATA calculates the element's next safe step size and reinserts the temperature update event into the event queue with a new target time. This process is repeated until the user specified simulation end time is reached. As illustrated

in Figure 2, the major components in FATA are temperature update, step size adaptation, and quiescence detection. The following sections explain these components.

IV.B. Temperature Update

Existing asynchronous methods [3] use exponential functions to update the temperature of the target element i, i.e., the element under consideration. The derivation is based on the assumption that the temperature of the neighbors of element i do not change over $[t_i, t_i + h_i]$, where t_i is element i's local time and h_i is element i's current step size. This assumption can lead to a large error when the neighboring nodes experience significant temperature changes during that period. Worse yet, these errors can accumulate as time advances, resulting in errors in temperature and step size calculation that degrade performance or accuracy. We therefore propose modeling temperature change using a variant of trapezoidal method that is tailored to asynchronous time marching. First, note that Equation 4 can be simplified as follows:

$$\text{Let } \beta = \sum_{n \in N} \frac{T_n(t)}{R_{in}} + P_i(t) = \alpha T_i(t) + C_i \frac{dT_i(t)}{dt}. \quad (33)$$

The trapezoidal method can be used to extrapolate element i's voltage at the target time $t_i + h_i$:

$$T_i(t_i + h_i) = T_i(t_i) + \frac{h_i}{2}(f_i'(t_i) + f_i'(t_i + h_i)), \quad (34)$$

where $f_i'(t_i)$ is the element i's temperature derivative at t_i and $f_i'(t_i+h_i)$ is the corresponding derivative at t_i+h_i. Given target time $t = t_i+h_i$, combining Equation 33 and Equation 34 yields

$$T_i(t_i + h_i) = \frac{\beta h_i + C_i h_i f_i'(t_i) + 2C_i T_i(t_i)}{\alpha h_i + 2C_i} \quad (35)$$

However, we still face the problem of computing $T_n(t_i + h_i)$ ($n \in N_i$) to obtain β at target time $t_i + h_i$. The trapezoidal method cannot be used to compute neighbor temperatures, for that would result in circular dependency problems. More specifically, $T_n(t_i + h_i)$ must be known before $T_i(t_i + h_i)$ is computed. Similarly, $T_n(t_i + h_i)$ depends on $T_i(t_i + h_i)$. To solve this problem, we use the forward Euler method to extrapolate $T_n(t_i + h_i)$ based on $T_n(t_n)$ and $f_n'(t_n)$, where $T_n(t_n)$ represents element n's temperature at t_n and $f_n'(t_n)$ represents the derivative of element n's temperature at t_n. We experimented with approximation functions with different orders and determined combining the trapezoidal method with the forward Euler method achieves a good balance between accuracy and performance.

IV.C. Step Size Adaptation

Given the trapezoidal method we use to estimate $T_i(t)$, the local truncation error in step n of element i can be expressed as $\epsilon_{in} = \frac{h_{i,n}^3 f_i'''(\zeta)}{12}$, where $h_{i,n}$ is the size of step n, ζ is between local times t_n and t_{n+1}, and $f_i'''(\zeta)$ is the third-order derivative of i's temperature at time ζ. In practice, we approximate the step size using a third-order divided difference. For element i at time step n, the third-order finite difference is expressed as follows:

$$DD_1(t_n) = (T_i(t_n) - T_i(t_{n-1}))/(t_n - t_{n-1}), \quad (36)$$

$$DD_2(t_n) = (DD_1(t_n) - DD_1(t_{n-1})/(t_n - t_{n-1}), \quad (37)$$

$$DD_3(t_n) = (DD_2(t_n) - DD_2(t_{n-1})/(t_n - t_{n-1}), \quad (38)$$

$$\epsilon_{in} = h_{i,n}^3 f_i'''(\zeta)/12 = DD_3(t_n)/2, \quad (39)$$

TABLE I
COMPARISON OF GARK4, ISAC, AND FATA

Problem	GARK4		ISAC			FATA		
	CPU time (s)	Error (%)	CPU time (s)	Error (%)	Speedup (×)	CPU time (s)	Error (%)	Speedup (×)
dct_ijpeg	2.05	0.01	10.67	0.04	0.19	0.05	0.03	37.86
dct_lee	32.74	0.00	43.16	0.08	0.76	0.27	0.03	122.55
dct_wang	36.93	0.00	32.02	0.1	1.15	0.27	0.02	138.30
jacobi	2.70	0.01	8.82	0.02	0.31	0.04	0.1	70.68
mac	2.04	0.01	11.13	0.03	0.18	0.05	0.03	38.33
pr2	31.71	0.00	85.15	0.03	0.37	0.39	0.09	80.56
rand100	33.87	0.00	47.51	0.08	0.71	0.31	0.05	109.58
rand200	14.60	0.00	48.14	0.02	0.30	0.28	0.05	52.57

where t_n and t_{n-1} are the local times at time step n and $n-1$. The $(n+1)th$ step size estimation is thus given by

$$h_{i,n+1} = k_1 * \left(\frac{(\text{RELTOL} * |T_i(t_n)| + \text{ABSTOL})}{\max(\text{ABSTOL}, \epsilon_{in})} \right)^{k_2}. \quad (40)$$

where ABSTOL and RELTOL are the maximum tolerable absolute and relative temperature errors. k_1 and k_2 are determined empirically to achieve a good balance between accuracy and speed. In practice, we use a k_1 of 1.5 and a k_2 of 0.3. This method of computing a new step size is similar to that in SPICE3 [8]. However, that formula uses a more complicated step control algorithm that also considers the maximum number of iteration at a given time point during its iterative solving process.

When the power profile is mostly static for a long time and the IC thermal profile approaches its steady state, i.e., the system becomes quiescent, we advance all nodes to the simulation end time. This step is called *quiescence detection*.[1]

V. EVALUATION

In this section, we evaluate FATA and compare it with existing thermal analysis techniques. Experiments were conducted on a Linux workstation equipped with a 1 GHz AMD Sempron 3100 Processor and 1 GB memory. We use a non-adaptive lock-step RK4 method with a very small step size as our accuracy (but not speed) reference; error values are computed relative to this reference. We first compare the accuracy and analysis times of the step doubling based GARK4 method, ISAC, and FATA. We also compare temperature update functions and step adaptation methods.

V.A. Comparison of GARK4, ISAC, and FATA

This section reports the accuracy and analysis time of the GARK4, ISAC, and FATA. Note that speedup over GARK4 claimed for ISAC in prior work [3] was incorrect due to an implementation error in the step size adjustment algorithm of the reference GARK4 method. We used eight real and synthetic behavioral synthesis benchmarks with different grid structures to evaluate the candidate analysis techniques. The dynamic power profiles are generated using a switching model proposed in the literature [4]. Three different power profiles were used to simulate different input patterns during behavioral synthesis and the average runtime for a power profile was reported. To permit a fair comparison among different techniques, we set the parameters for each technique to maximize performance while constraining the peak temperature error over all benchmarks and all time to 0.1%.

[1]We omit the details due to space constraints. A extended technical report will be published.

TABLE II
COMPARISON AMONG DIFFERENT COMBINATIONS OF TEMPERATURE UPDATE FUNCTIONS AND STEP SIZE ADAPTATION TECHNIQUES

Problem	TR w. Step Doubling			FE w. Step Doubling			FE w. DD3			TR w. DD3	
	CPU time (s)	Error (%)	Slowdown (×)	CPU time (s)	Error (%)	Slowdown (×)	CPU time (s)	Error (%)	Slowdown (×)	CPU time (s)	Error (%)
dct_ijpeg	0.39	0.02	6.93	9.84	0.06	174.68	5.65	0.07	100.23	0.06	0.04
dct_lee	1.57	0.05	3.29	47.62	0.03	99.90	40.60	0.03	85.17	0.48	0.02
dct_wang	1.46	0.06	2.64	49.00	0.02	88.23	41.22	0.02	74.22	0.56	0.02
jacobi	0.27	0.01	5.17	6.52	0.03	123.29	4.88	0.04	92.31	0.05	0.07
mac	0.39	0.01	7.06	9.79	0.05	177.20	5.66	0.07	102.48	0.06	0.05
pr2	2.17	0.05	4.34	66.70	0.06	133.25	50.29	0.07	100.47	0.50	0.05
rand100	1.69	0.07	3.33	50.32	0.03	98.86	43.52	0.04	85.49	0.51	0.02
rand200	1.40	0.02	4.17	39.33	0.06	117.20	33.47	0.06	99.75	0.34	0.03

Table I presents the experimental results. Each row shows the results for a specific benchmark. Columns two, four, and seven indicate the CPU time used by GARK4, ISAC, and FATA. Columns six and nine indicate the speedups achieved by ISAC and FATA compared to GARK4 given the same error constraint. FATA speeds up analysis by 37.86–138.30× compared to GARK4 method and 118.59–222.60× compared to ISAC, while maintaining accuracy.

We also examined the performance of synchronous methods. We first compared the accuracy and performance of non-adaptive lock-step synchronous methods, namely forward Euler (FE), backward Euler (BE), trapezoidal (TR), and explicit RK4. Due to space limitations, we omit the full results. FE is inaccurate, with a peak error of 0.52% regardless of step size. FE and RK4 are impractical because they require manual specification of safe step size. FATA is 4.4–20.6× faster than all the non-adaptive synchronous methods. Adaptive implicit methods are impractical because they require inversion of the system matrix at each time step.

V.B. Combining Temperature Update Functions with Step Size Adaptation Methods

This section analyzes the impact of temperature update function and step size adaptation method on asynchronous method performance. We use FE and the variant of TR describe in Section IV as temperature update candidates and consider step doubling and third-order divided difference (DD3) as potential step size adaptation methods. We use the approach adopted by FATA, i.e., TR combined with DD3 as the base case.

Columns two, five, eight, and eleven of Table II indicate the runtime for each combination, while columns four, seven, and ten show the slowdown using the corresponding combination compared to the base case, i.e., TR combined with DD3. Note that the runtime for TR combined with DD3 is slightly larger than FATA because quiescence detection is not used. In comparison, DD3 is generally better than step doubling, resulting in a speedup of 2.61–6.5× for TR and a speedup of 1.16–1.74× for FE. The temperature update function is critical and TR is consistently faster than FE. This explains why FATA is much faster than ISAC, which makes the inaccurate assumption that neighboring element temperatures do not change during the integration interval of the current element. This can cause temperature estimation errors that decrease step size and degrade performance. The combination of TR and DD3 based step size adaptation is the best among all candidates. Although a higher-order temperature update function might further improve accuracy, this would impose additional computational overhead. In addition, high-order numerical methods are generally more likely to cause numerical instability problems in asynchronous methods. We experimented with a third-order temperature update function. Experimental results indicate the increase in the computational cost outweighs the improvement in accuracy: their use is not recommended.

VI. CONCLUSIONS

This paper proves that step doubling based globally adaptive fourth-order Runge-Kutta and Runge-Kutta-Fehlberg methods will fail to appropriately adapt step sizes regardless of initial power profile, thermal profile, and user-specified error threshold, i.e., the steady-state step size will converge to a constant value even when the thermal profile has converged, leading to poor performance. We proposed FATA, an asynchronous time marching thermal analysis technique that corrects this problem and uses other ideas to improve speed and accuracy. Experimental results indicate that FATA speeds up dynamic thermal analysis by 37.86–138.30× compared to an existing synchronous globally-adaptive fourth-order Runge-Kutta method and by 118.59–222.60× relative to an existing asynchronous adaptive dynamic thermal analysis technique, while maintaining accuracy. FATA will replace the time-domain solver used in ISAC; a new version will be released after integration [9]. Note that our findings do not imply that Runga-Kutta based thermal analysis methods are inaccurate, only that they may be inefficient due to inappropriately adapting step sizes to problem conditions.

REFERENCES

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

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

[3] Y. Yang, et al., "ISAC: Integrated space and time adaptive chip-package thermal analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 86–99, Jan. 2007.

[4] 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.

[5] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, 2002.

[6] M. T. Heath, *Scientific Computing: An Introductory Survey*. McGraw-Hill Science Engineering, 2001.

[7] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management," in *Proc. Int. Symp. High-Performance Computer Architecture*, Feb. 2002.

[8] T. Quarles, "The SPICE3 implementation guide," EECS Department, University of California, Berkeley, Tech. Rep., 1989. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1989/1218.html

[9] N. Allec, et al., "ISAC2: Incremental self-adaptive chip-package thermal analysis software, version 2," ISAC2 link at http://ziyang.eecs.umich.edu/projects/isac and http://eces.colorado.edu/~hassanz/ThermalScope.