Journal of
Low Power Electronics
Vol. 3, 1–9, 2007

TILTS: A Fast Architectural-Level Transient Thermal Simulation Method

Yongkui Han*, Israel Koren, and C. M. Krishna

Department of ECE, University of Massachusetts, Amherst, MA 01002, USA

(Received: 6 September 2006; Accepted: 19 February 2006)

As power density of microprocessors is increasing rapidly and resulting in high temperatures, the reliability of chips is greatly affected, making thermal simulation a necessity for CPU designs. Current thermal simulation methods (for example, the HotSpot simulator) are very useful, but are still inefficient when performing thermal analysis for long simulation times. In this paper, we propose a novel transient thermal simulation method for CPU chips at the architecture level, TILTS (Time Invariant Linear Thermal System), which utilizes the fact that the input power trace is discretized over a fixed sampling interval to accelerate thermal simulations. TILTS allows us to calculate transient temperatures on a chip over long simulation times. Based on a linear system formulation, TILTS has the same accuracy as that of traditional thermal simulation tools and is orders of magnitude faster than previous algorithms. Compared to the HotSpot simulator, TILTS achieves speedups of 1300 for the processors in our experiments for an appropriate sampling interval of 100 μs. With some additional memory space, the improved algorithm CONTILTS (Convolutional TILTS) is about 6000 times faster than the HotSpot simulator for the processors in our experiments.

Keywords: Thermal Simulation, Fast Method, Linear System, Performance Counters, Runtime Temperature Monitoring.

1. INTRODUCTION

Power density is increasing in each generation of microprocessors, since feature size and frequency are scaling faster than the operating voltage. Power density directly translates into heat, and consequently processors are getting hotter. Operating temperature has a significant impact on microprocessor design. At higher temperatures, transistors work slower because of the degradation of carrier mobility. The interconnect metal resistivity is also higher at higher temperatures, causing longer interconnect RC delays, and therefore performance degradation. In addition, due to its exponential dependence on operating temperature, leakage power can be orders of magnitude greater at higher temperatures. Reliability is also strongly related to temperature, and increasing the temperature will exponentially decrease the lifetime of the chip. Last, but not least, a higher operating temperature increases the cost of cooling solutions. Therefore, keeping the chip at low temperatures is an important goal for chip designers.

Dynamic thermal management (DTM)[7,15] has been proposed to provide real-time regulation of on-chip temperature for today's high-performance microprocessors. Fast on-chip thermal estimation is necessary to study the effectiveness of various DTM techniques.

The finite element method (FEM) is the traditional approach to thermal simulation. In FEM, the problem domain under study, which is usually complex, is discretized into a series of smaller regions (called elements) where the differential equations associated with the thermal simulation are solved. The behavior over the entire problem domain is determined by assembling the set of equations for each region. FEM is very accurate but also very computation-intensive, and it is thus only appropriate for detailed thermal analysis of small scale chips. There is a tradeoff between accuracy and computation time in choosing granularity in FEM. Several FEM tools are available for thermal package designers, such as FLOTHERM[3] and COMSOL.[2] Recently, Yang et al. presented a spatially and temporally adaptive thermal analysis technique, ISAC,[16] which considerately accelerates the FEM. Despite its higher implementation complexity, ISAC achieves one to two orders of magnitude speedup over COMSOL, making practical its use within IC synthesis algorithms.

There exists a well-known duality between heat flow and electrical current flow, since both are described by exactly the same differential equations. Heat flow can be described as a current passing through a thermal "resistor",

*Author to whom correspondence should be addressed.
Email: yhan@ecs.umass.edu

leading to a temperature difference analogous to a voltage. Lumped values of thermal resistance and capacitance can be computed to represent the heat flow among units. Based on this duality, equivalent circuits called compact RC models can be constructed.[15] Dynamic compact RC models at the architecture level are preferable over FEM due to their faster simulation speed.

Based on a compact thermal RC model, a transient thermal simulator HotSpot[9,15] has been developed to study different DTM techniques in microprocessors. HotSpot generates the thermal RC circuit dynamically when initialized with a CPU floorplan. During thermal simulation, the power values of blocks are fed into the HotSpot model by a dynamic architectural power/performance simulator, such as Wattch.[8] Based on the present temperature of blocks, HotSpot uses the fourth-order Runge-Kutta method (rk4) to solve the differential equations that describe the RC circuit, and returns the new temperature of blocks at each time step. In this way, the complete temperature profile of an application can be computed step by step.

However, due to the use of a conventional integration-based transient simulation method, HotSpot becomes inefficient when attempting to obtain the temperature profile of an entire chip for a given benchmark program. A typical benchmark program has tens to hundreds of billions of instructions and the approach followed in HotSpot can lead to a very long simulation time.

We notice the fact that the input power trace to a compact thermal RC model is usually disctretized over a fixed sampling interval, e.g., 3.3 μs, 5 μs, or 40 ms. During the sampling interval time, the input power is assumed to be constant. Traditional simulation methods used in HotSpot are intended for continuous input power curves in general cases. Such general purpose methods perform a large amount of redundant computations in calculating thermal responses to a constant power in a sampling interval.

This characteristic of the input power trace provides an opportunity to accelerate the thermal simulation. The basic idea is to identify redundant computations and replace them with more efficient computations. We developed such a method, TILTS (Time Invariant Linear Thermal System), which is much faster than conventional methods without any loss in accuracy for such type of input power traces. We have integrated TILTS in the widely used architectural level thermal simulator HotSpot. Experimental results show that TILTS is hundreds of times, or even millions of times faster than the original HotSpot, while generating the same results as HotSpot. Based on TILTS, we also implemented a lightweight runtime temperature monitoring tool called *Temptor* to demonstrate an example application of TILTS. To the best of our knowledge, TILTS is the first method to utilize linear system formulation to accelerate thermal simulations.

In Ref. [12], the authors propose a thermal moment matching (TMM) method to accelerate the thermal simulation by applying fast moment matching techniques in the frequency domain. Their method is 79 times faster than HotSpot with a maximum error of 0.37 °C, but is restricted to estimating the thermal response to the DC component in the input power trace: the AC components are disregarded. Their estimate is an approximation due to the reduced order of the dynamic system.

In Ref. [12], the authors extend the TMM method to calculate the steady state thermal response to a periodic power trace via the discrete Fourier transform, and use the steady-state thermal response as an approximation for the transient thermal response to the periodic power trace. The error of the revised method, based on the figures in Ref. [13], is about 0.3 °C.

The TMM method is a model order reduction (MOR) method.[6] Our TILTS method is complementary to MOR methods. TILTS reduces the amount of computations in the time domain, while MOR methods reduce the state space size of the dynamic system. If both methods are used, it is possible to perform fast thermal simulation for large-scale systems. MOR methods usually cause some degree of accuracy loss due to state space approximation. TILTS does not incur any accuracy loss since the input power trace is already discretized in its intended usage scenarios.

Based on a linear system formulation, the time-consuming integration computations are replaced in TILTS by more efficient matrix multiplications, as is presented in the following section.

2. LINEAR SYSTEM THEORY OVERVIEW

A linear system[14] is described by its state equations, where the state variables are system internal variables. Denote the number of state variables by N, the number of inputs to the system by M, and the state and input vectors by $\mathbf{x}(t)$ and $\mathbf{u}(t)$, respectively:

$$\mathbf{x}(t) = (x_1(t), x_2(t), \ldots, x_N(t))^T$$
$$\mathbf{u}(t) = (u_1(t), u_2(t), \ldots, u_M(t))^T$$

The linear system equation is:

$$\dot{\mathbf{x}}(t) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) \tag{1}$$

where $\mathbf{F}$ is an $N \times N$ matrix, and $\mathbf{G}$ is an $N \times M$ matrix. These matrices are fixed for a time-invariant linear system.

For such a linear system, the complete response is the sum of the zero-input response and the zero-state response:[1]

$$\mathbf{x}(t) = e^{\mathbf{F}t}x(0) + \int_0^t e^{\mathbf{F}(t-\tau)}\mathbf{G}\mathbf{u}(\tau)d\tau \tag{2}$$

The first term on the right hand side is the zero input response due to the initial condition and the second term is the response to the input impulse, $\mathbf{u}(t)$.

2.1. CPU Chip as a Linear System

A CPU chip is a thermal system that can be described by its equivalent thermal circuit composed of thermal resistors and capacitors. All of these components are linear components, making it a linear system. The input to this linear system is the power dissipated by each functional unit on the chip, and its state variables are the temperatures of the internal nodes in the thermal circuit.

Let M be the number of functional units which dissipate power, and N the number of internal nodes in the thermal circuit ($N \geq M$). Denote the thermal resistance between node i and j by r_{ij}, and the thermal capacitance to the thermal ground (the ambient environment) by c_i for node i. For convenience, let $1/r_{ii} = 0$ in Eq. (3). The CPU thermal system obeys the following differential equation:

$$c_i \dot{x}_i(t) = -\sum_{j=1}^{N} \frac{1}{r_{ij}}(x_i(t) - x_j(t)) + \tilde{u}_i(t), \quad i = 1, 2, \ldots, N \tag{3}$$

where $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))^T$ is the temperature vector and $\tilde{\mathbf{u}}(t) = (u_1(t), \ldots, u_M(t), 0, \ldots, 0)^T$, i.e., $\tilde{\mathbf{u}}(t)$ is $\mathbf{u}(t)$ (the power vector) extended by N–M zeros, corresponding to nodes which have no power dissipation associated with them (e.g., thermal interface material, heat spreaders, heat sinks, etc.).

Let $\mathbf{D} = (d_{ij})_{N\times N}$, $\mathbf{C} = (c_{ij})_{N\times N}$, where

$$d_{ij} = \begin{cases} -\sum_{k=1}^{N} \frac{1}{r_{ik}} & \text{if } i = j \\ \frac{1}{r_{ij}} & \text{if } i \neq j \end{cases} \tag{4}$$

$$c_{ij} = \begin{cases} c_i & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{5}$$

Then, Eq. (3) can be rewritten as:

$$\mathbf{C}\dot{\mathbf{x}}(t) = \mathbf{D}\mathbf{x}(t) + \tilde{\mathbf{u}}(t) \tag{6}$$

$\mathbf{C}$ is a diagonal matrix, and thus it is easy to compute its inverse $\mathbf{C}^{-1}$ and obtain the standard differential equation:

$$\dot{\mathbf{x}}(t) = \mathbf{C}^{-1}\mathbf{D}\mathbf{x}(t) + \mathbf{C}^{-1}\tilde{\mathbf{u}}(t) \tag{7}$$

Note that since $\tilde{u}_i = 0$ for $i > M$, only the leftmost M columns of $\mathbf{C}^{-1}$ are useful in the second term in Eq. (7). We can, therefore, construct an $N \times M$ matrix G out of the left M columns of $\mathbf{C}^{-1}$ and replace $\tilde{\mathbf{u}}$ by $\mathbf{u}$. Thus, we obtain an equation similar to (1) with

$$\mathbf{F} = \mathbf{C}^{-1}\mathbf{D} \text{ and } \mathbf{G} = \text{left } M \text{ columns of } \mathbf{C}^{-1} \tag{8}$$

Therefore, a formula similar to Eq. (2) can be used to calculate the transient temperature of the CPU.

The input power trace to the CPU is usually given as a series of power vectors. In a sampling interval Δt, the power vector $\mathbf{u}(t)$ is constant, allowing us to simplify Eq. (2) as follows:

$$\mathbf{x}(\Delta t) = e^{\mathbf{F}\Delta t}\mathbf{x}(0) + \left[\int_0^{\Delta t} e^{\mathbf{F}(\Delta t - \tau)}\mathbf{G}\, d\tau\right] \cdot \mathbf{u} \tag{9}$$

We use this simplified equation to reduce the amount of computation. Denoting

$$\mathbf{A} = e^{\mathbf{F}\Delta t}, \quad \mathbf{B} = \int_0^{\Delta t} e^{\mathbf{F}(\Delta t - \tau)}\mathbf{G}\, d\tau \tag{10}$$

we obtain the equation:

$$\mathbf{x}(\Delta t) = \mathbf{A}\mathbf{x}(0) + \mathbf{B}\mathbf{u} \tag{11}$$

Because the system is a time-invariant linear system, we obtain the same equation for any interval Δt with the same matrices $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{x}(n\Delta t) = \mathbf{A}\mathbf{x}((n-1)\Delta t) + \mathbf{B}\mathbf{u}(n-1) \tag{12}$$

where $\mathbf{u}(n-1)$ is the power vector in the time interval $[(n-1)\Delta t, n\Delta t]$.

We will use $\mathbf{x}(n)$ to represent $\mathbf{x}(n\Delta t)$ for conciseness, resulting in:

$$\mathbf{x}(n) = \mathbf{A}\mathbf{x}(n-1) + \mathbf{B}\mathbf{u}(n-1) \tag{13}$$

3. TIME INVARIANT LINEAR THERMAL SYSTEM (TILTS) METHOD

Our transient thermal simulation method, TILTS, is based on Eq. (13). Suppose that the number of power vectors (called data points) in the input power trace $\mathbf{u}$ is n, and the initial temperature is $\mathbf{x}_0$. TILTS is shown below:

Algorithm TILTS($\mathbf{x}_0, \mathbf{u}, n$)

(1) Calculate matrices $\mathbf{D}$ and $\mathbf{C}$ using HotSpot. Then calculate matrices $\mathbf{F}$ and $\mathbf{G}$ using Eq. (8). Finally, calculate matrices $\mathbf{A}$ and $\mathbf{B}$ for the sampling interval Δt using Eq. (10).

(2) For $i = 1$ to n do $\mathbf{x}(i) = \mathbf{A}\mathbf{x}(i-1) + \mathbf{B}\mathbf{u}(i-1)$.

Denoting $\mathbf{A} = [\mathbf{a}_1\mathbf{a}_2 \ldots \mathbf{a}_N]$ and $\mathbf{B} = [\mathbf{b}_1\mathbf{b}_2 \ldots \mathbf{b}_M]$, where $\mathbf{a}_i, i = 1, \ldots, N$ and $\mathbf{b}_i, i = 1, \ldots M$ are the i-th column vectors in $\mathbf{A}$ and $\mathbf{B}$, respectively, then for a given time interval $[0, \Delta t]$,

$$\mathbf{x}(\Delta t) = \begin{cases} \mathbf{a}_i, & \text{if } x_i = 1,\ x_j = 0\,(j \neq i),\ u_j = 0 \\ \mathbf{b}_i, & \text{if } u_i = 1,\ u_j = 0\,(j \neq i),\ x_j = 0 \end{cases} \tag{14}$$

That is, each column vector of the matrices A and B is the step response to only one x_i or u_i. In our implementation, HotSpot is used to calculate the step response to a single x_i or u_i over time Δt, and then the matrices $\mathbf{A}$ and $\mathbf{B}$ are obtained using Eq. (14). Therefore, while the calculation of the matrices $\mathbf{A}$ and $\mathbf{B}$ still uses the conventional integration-based method, this calculation is only performed once with a computation time which is less than 0.01 seconds.

Table I. The number of inputs M and the number of internal nodes N in the equivalent thermal circuits for the Pentium Pro, Alpha, and Pentium 4 processors.

Processor	Number of nodes N	Number of inputs M	N/M
Pentium Pro	58	16	3.63
Alpha 21364	97	18	5.39
Pentium 4	76	22	3.45

3.1. Performance Analysis

In our experiments we studied the Intel Pentium Pro, Compaq Alpha 21364, and Intel Pentium 4 processors. The number of inputs M and the number of internal nodes N in the equivalent thermal circuits are shown in Table I. The floorplans of the Pentium Pro, Alpha 21364 and Pentium 4 processors are shown in Figures 1, 2, and 3, respectively.

In HotSpot, the transient temperatures are computed by solving the thermal differential equation using a numerical integration method based on the fourth-order Runge-Kutta technique (rk4).[15] In order to keep the truncation error of rk4 small, the step size in rk4 must be very small. In one sampling interval Δt, tens of iterations of rk4 are required to keep the truncation error negligible. The number of iterations of rk4 in HotSpot during one sampling interval Δt is shown in Table II.

The computations in one iteration of rk4 consist of temperature calculations at 4 time instants, which are roughly equal to 4 times that of an $\mathbf{Ax}+\mathbf{Bu}$ operation.

The numbers of floating-point multiplications (FPM) in one iteration of rk4 and in the $\mathbf{Ax}+\mathbf{Bu}$ operation are shown in Table III. We can see from Table III that the number of FPMs in one iteration of rk4 is roughly 5 times that of the $\mathbf{Ax}+\mathbf{Bu}$ operation, validating our analysis.

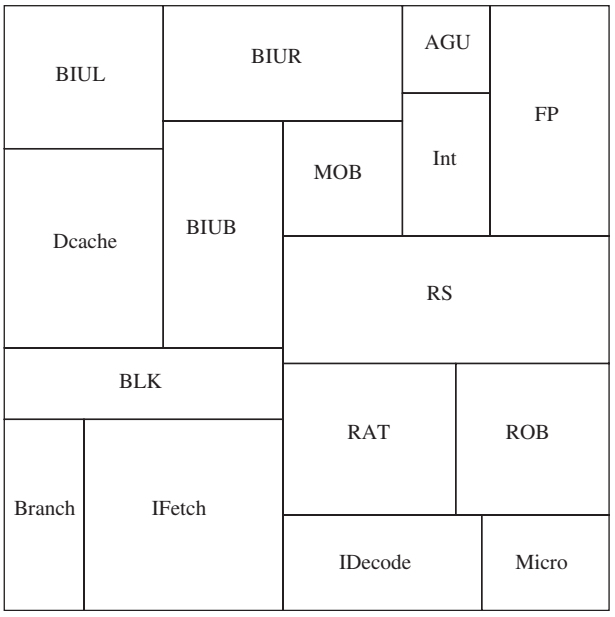

Fig. 1. Pentium Pro processor floorplan.

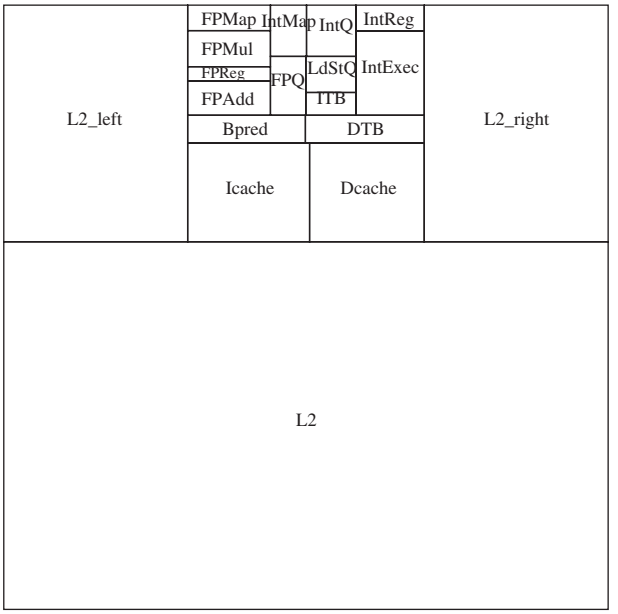

Fig. 2. Alpha 21364 processor floorplan.

In our TILTS method, we need only one step to calculate the temperature at $t = \Delta t$. The time consuming integration computations are replaced by one step of matrix multiplications. Our new method saves many computations performed by the fourth-order Runge-Kutta method used in HotSpot. This way, we achieve identical results in a much shorter computation time.

4. AN IMPROVED ALGORITHM CONTILTS

The performance of the TILTS algorithm can be further improved without any loss of accuracy. Based on

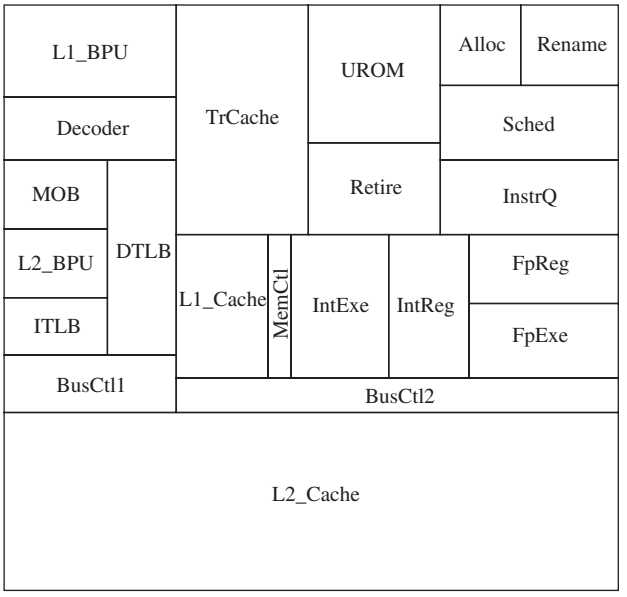

Fig. 3. Pentium 4 Pro processor floorplan.

Table II. The number of iterations of rk4 in the HotSpot simulator.

Sampling interval Δt	Number of rk4 iterations in HotSpot
3.33 μs	9
5 μs	13
10 μs	26
20 μs	51
50 μs	128
100 μs	256
5 ms	12800
40 ms	102400

Eq. (13),

$$\mathbf{x}(p) = \mathbf{A}^p\mathbf{x}(0) + \sum_{j=0}^{p-1} \mathbf{A}^{p-1-j}\mathbf{B}\mathbf{u}(j)$$
$$= \mathbf{A}^p\mathbf{x}(0) + \sum_{j=0}^{p-1} \mathbf{L}_j\mathbf{u}(p-1-j) \qquad (15)$$

where

$$\mathbf{L}_j = \mathbf{A}^j\mathbf{B}, \quad j = 0, 1, \ldots, p-1 \qquad (16)$$

We can precompute the matrices $\mathbf{A}^p$ and $\mathbf{L}_j, j = 0, 1, \ldots, p-1$, and save them in a table for later use. This is the motivation behind the revised algorithm.

The second term on the righthand side of Equation (15) is the convolution of the input power trace and the step response of the thermal system. We call the revised algorithm CONTILTS (CONvolutional TILTS), and it is shown below:

Algorithm CONTILTS($\mathbf{x}_0, \mathbf{u}, n, p$)

(1) Calculate matrices $\mathbf{A}$ and $\mathbf{B}$ for the sampling interval Δt as in TILTS.

(2) Calculate matrices $\mathbf{A}^p$ and $\mathbf{L}_j, j = 0, \ldots, p-1$, from $\mathbf{A}$ and $\mathbf{B}$ using (16).

(3) Divide the input power trace $\mathbf{u}$ into groups of size p which are called windows. The n data points contain $\lfloor n/p \rfloor$ such windows.

(4) For window $i = 1$ to $\lfloor n/p \rfloor$ do

$$\mathbf{x}(i) = \mathbf{A}^p\mathbf{x}(i-1) + \sum_{j=0}^{p-1} \mathbf{L}_j\mathbf{u}((i-1)p + p - 1 - j)$$

where $\mathbf{x}(i)$ and $\mathbf{x}(i-1)$ are the output temperatures of windows i and $i-1$, respectively.

Table III. The comparison of the number of floating-point multiplications (FPMs) in one iteration of rk4 in HotSpot to that in the $\mathbf{Ax}+\mathbf{Bu}$ operation.

Processor	Number of FPMs in rk4	Number of FPMs in $\mathbf{Ax}+\mathbf{Bu}$	Ratio
Pentium Pro	20590	4292	4.797
Alpha 21364	57133	11155	5.122
Pentium 4	35188	7448	4.724

(5) Process the remaining (n mod p) data points using the TILTS algorithm.

The $\mathbf{L}_j$ matrices introduce some memory overhead. Assuming that a floating-point number is saved in 8 bytes of memory, the memory overhead for the $\mathbf{L}_j$ matrices is shown in Table IV. The memory overhead for a typical window size of 512 is only 3.6 MB for modeling the Pentium Pro processor, 6.8 MB for the Alpha processor, and 6.5 MB for the Pentium 4 processor, which are negligible for modern desktop computers.

Although the memory space requirement of CONTILTS is negligible for the main memory, it is larger than the cache size of a modern microprocessor, and thus it will cause cache corruption. This, however, will not have a significant impact on the performance since spatial locality can be exploited in matrix multiplication operations.

The calculation of the matrices $\mathbf{A}^p$ and $\mathbf{L}_j, j = 0, \ldots, p-1$ takes less than 1 second, and is performed only once. The method used to calculate matrix exponents is not important in CONTILTS: since p is always a power of 2, and $A^{1024} = A^{2^{10}}$, we only need 10 matrix multiplications to calculate A^{1024}. Matrix diagonalization is not needed to accelerate matrix exponent computations because the architectural level thermal model is always a small-scale system. Furthermore, since we pre-calculate these matrices before performing transient thermal simulation, the efficiency of calculating the matrix exponents does not affect the computation speed of the transient thermal simulation later on.

For a window of p data points, $p-1$ redundant $\mathbf{Ax}$ operations are removed in the algorithm CONTILTS. The number of floating-point multiplications is reduced from $p(N^2+NM)$ to N^2+pNM. The speedup ratio s is:

$$s = \frac{p(N^2+NM)}{N^2+pNM} = \frac{p(N+M)}{N+pM} \qquad (17)$$

Table IV. The memory overhead of the $\mathbf{L}_j$ matrices for a window of p data points and the speedup ratio of floating-point multiplications in the algorithm CONTILTS versus TILTS.

Processor	Window size p	Memory size	Number of FPMs in TILTS	Number of FPMs in CONTILTS	Ratio	$(N+M)/M$
Pentium Pro	512	3.63 MB	2197504	478500	4.59	4.63
Pentium Pro	1024	7.25 MB	4395008	953636	4.61	4.63
Alpha	512	6.82 MB	5711360	903361	6.32	6.39
Alpha	1024	13.64 MB	11422720	1797313	6.36	6.39
Pentium 4	512	6.53 MB	3813376	861840	4.42	4.45
Pentium 4	1024	13.06 MB	7626752	1717904	4.44	4.45

When $p \to \infty, s \to (N+M)/M$. Therefore, the maximum speedup ratio of the algorithm CONTILTS over TILTS is $(N+M)/M$.

The numbers of floating-point multiplications in TILTS and CONTILTS are shown in Table IV. From Table IV, we can see that the speedup with a window size of 512 is already very close to the maximum speedup ratio, so 512 is large enough for the algorithm CONTILTS. N is about 4 to 5 times M in the Pentium Pro, Alpha, and Pentium 4 processors. A speedup of 5 to 6 is therefore expected with the revised algorithm CONTILTS compared to TILTS.

5. STEADY STATE TEMPERATURE

We next derive an equation to calculate the steady state temperature of a CPU chip.

$$\mathbf{x}(1) = \mathbf{A}\mathbf{x}(0) + \mathbf{B}\mathbf{u}(0)$$

$$\mathbf{x}(2) = \mathbf{A}\mathbf{x}(1) + \mathbf{B}\mathbf{u}(1) = \mathbf{A}^2\mathbf{x}(0) + \mathbf{A}\mathbf{B}\mathbf{u}(0) + \mathbf{B}\mathbf{u}(1)$$

$$\dots$$

$$\mathbf{x}(n) = \mathbf{A}^n\mathbf{x}(0) + \mathbf{A}^{n-1}\mathbf{B}\mathbf{u}(0) + \dots + \mathbf{B}\mathbf{u}(n-1)$$

$$= \mathbf{A}^n\mathbf{x}(0) + \sum_{i=0}^{n-1} \mathbf{A}^{n-1-i}\mathbf{B}\mathbf{u}(i) \tag{18}$$

When $n \to \infty, \mathbf{A}^n \to 0$, and if we use the average power vector $\overline{\mathbf{u}}$ to replace every power vector $\mathbf{u}(i)$ in (18), we can get the approximate steady state temperature $\mathbf{x}(\infty)$.

$$\overline{u}_j = \frac{\sum_{i=0}^{n-1} u_j(i)}{n}, \quad j = 1, 2, \dots, N \tag{19}$$

$$\overline{\mathbf{u}} = (u_1, u_2, \dots, u_N)^T \tag{20}$$

$$\mathbf{x}(n) \approx \mathbf{A}^n\mathbf{x}(0) + (\mathbf{A}^{n-1}\mathbf{B} + \dots + \mathbf{B})\overline{\mathbf{u}}$$

$$= \mathbf{A}^n\mathbf{x}(0) + \sum_{i=0}^{n-1} \mathbf{A}^i\mathbf{B}\overline{\mathbf{u}} \tag{21}$$

$$= \mathbf{A}^n\mathbf{x}(0) + (\mathbf{I}-\mathbf{A})^{-1}(\mathbf{I}-\mathbf{A}^n)\mathbf{B}\overline{\mathbf{u}}$$

$$\mathbf{x}(\infty) = (\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\overline{\mathbf{u}} \tag{22}$$

In HotSpot, the steady state temperature is calculated with the following method: Let $\Delta\mathbf{x}(t) = \dot{\mathbf{x}}(t)\Delta t = (\mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t))\Delta t = 0$, we can then get the equation:

$$\mathbf{x}(t) = -\mathbf{F}^{-1}\mathbf{G}\mathbf{u} \tag{23}$$

We notice that for a small time interval Δt,

$$\mathbf{A} = e^{\mathbf{F}\Delta t} \approx \mathbf{F}\Delta t + \mathbf{I} \tag{24}$$

and

$$\mathbf{B} = \int_0^{\Delta t} e^{\mathbf{F}(\Delta t - \tau)}\mathbf{G}\,d\tau \approx \mathbf{G}\Delta t \tag{25}$$

thus $(\mathbf{I}-\mathbf{A})^{-1}\mathbf{B} \approx -\mathbf{F}^{-1}\mathbf{G}$. Eq. (23) is approximately equivalent to Eq. (19) for a small time interval Δt.

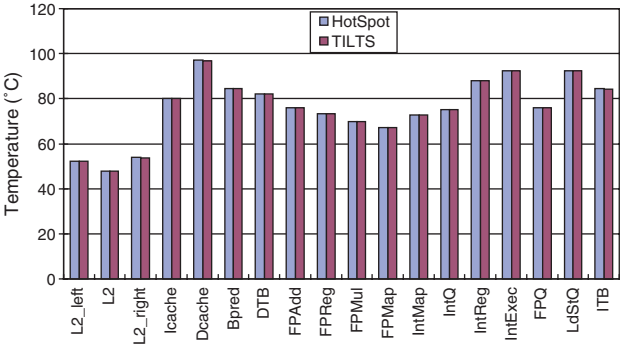

Fig. 4. The comparison of the computed steady state temperatures using the TILTS method to those computed by the HotSpot simulator.

The computed steady state temperatures for the *gcc* benchmark (for $\Delta t = 3.33\ \mu$s) are shown in Figure 4, which also shows the steady state temperatures computed by the HotSpot simulator. We can see from Figure 4 that they agree very well. The maximum temperature difference is 0.03 °C.

6. EXPERIMENTAL RESULTS

Pre-calculation of the matrices $\mathbf{A}, \mathbf{B}, \mathbf{A}^p$, and $\mathbf{L}_j, j = 0, 1, \dots, p-1$ takes less than 1 second for a typical window size of $p = 1024$. Since these pre-computations are done only once, this overhead is amortized over all simulation steps, making it negligible. For any given processor, these matrices are fixed, and we only need to calculate them once and use them for any input power trace.

We used in our experiments version 3.0.2 of the HotSpot simulator. Our algorithms are implemented in C, and have been incorporated into the HotSpot simulator.

We evaluated our results by running several SPEC2000 benchmark programs[4] to generate the power trace files. After that, the same power trace file was fed into both the original HotSpot simulator and the modified HotSpot simulator with our proposed methods, to enable a comparison.

The power traces for the Pentium Pro and Alpha processors have been generated using the Wattch simulator.[8] The sampling interval for the Pentium Pro processor in our experiments is 5 μs, and for the Alpha 21364 processor is 3.3 μs. The power trace for the Pentium 4 processor was generated through power modeling based on runtime performance counter readings from a real Pentium 4 processor. The sampling interval of the Pentium 4 processor is 40 ms, which is the smallest interval allowed by the tool for collecting power traces in a real Pentium 4 processor. It is a limitation imposed by the power trace collecting platform. Although such a long interval is not reasonable (or practical) for the HotSpot simulator, we can still see from such experiments the benefits we can expect from TILTS in some special applications.

The experiments were performed on a 3 GHz Intel Pentium 4 machine. The speedups of our methods over

Table V. Comparisons of computation times for different benchmark programs between our TILTS and CONTILTS methods and the HotSpot method.

Processor	Bench-mark	Interval Δt (μs)	Number of intervals	HotSpot (s)	TILTS (s)	Speed-up	CONTILTS	Speed-up
Pentium Pro	*gcc*	5	20 M	15.8 K	237	67	59 s	268
Pentium Pro	*mgrid*	5	20 M	15.8 K	238	66	59 s	268
Alpha	*gcc*	3.33	20 M	28 K	592	47	129 s	217
Alpha	*mgrid*	3.33	20 M	28 K	592	47	129 s	217
Pentium 4	*gcc*	40000	2500	26 K	0.05	520 K	13 ms	2 M
Pentium 4	*mgrid*	40000	2500	26 K	0.05	520 K	13 ms	2 M
Pentium Pro	*gcc*	100	1 M	15.6 K	12	1300	3 s	5200
Alpha	*gcc*	100	1 M	40 K	30	1333	6.5 s	6154
Pentium 4	*gcc*	100	1 M	26 K	20	1300	5 s	5200

the method used in HotSpot are shown in Table V. For the algorithm CONTILTS, we used $p = 1024$ in our experiments.

When generating the same temperature trace, our TILTS algorithm was 67 times faster than the HotSpot simulator for the Pentium Pro processor ($\Delta t = 5$ μs), 47 times faster for the Alpha processor ($\Delta t = 3.3$ μs), and 0.5 million times faster for the Pentium 4 processor ($\Delta t = 40$ ms). The extent of the speedup for the Pentium 4 is mainly due to the very large interval which we had to use. Our CONTILTS algorithm is 268 times faster than the HotSpot simulator for the Pentium Pro processor ($\Delta t = 5$ μs), 217 times faster for the Alpha processor ($\Delta t = 3.3$ μs), and 2 million times faster for the Pentium 4 processor ($\Delta t =$ 40 ms). None of our methods loses any accuracy compared to HotSpot.

We also did some experiments for a sampling interval of 100 μs for all three processors. A 100 μs interval results in high performance of TILTS while preserving simulation accuracy (described in the next section). For such a sampling interval, TILTS is 1300 times faster than HotSpot, and CONTILTS is about 6000 times faster than HotSpot.

The speedup ratio is determined only by the number of inputs M, the number of internal nodes N in the equivalent thermal circuit, and the sampling interval Δt. We can see from Table V that the speedup is independent of the benchmark program.

7. APPLICABILITY AND LIMITATIONS

The TILTS method is only intended for small linear systems such as the architectural level thermal model, and not for large-scale linear systems with millions of nodes. For large-scale systems, model order reduction is necessary, which is complementary to our TILTS method. The TILTS method is not scalable for large systems by itself.

Also, the TILTS method is not intended to calculate the thermal response to continuous input power traces. It is only intended to calculate the thermal response of discretized power inputs, e.g., the application of runtime temperature monitoring in *Temptor* (described below).

Although linear system modeling of thermal systems is not new, the TILTS method is novel in utilizing the specific properties of the input power trace to accelerate thermal simulations. As a result, TILTS is simple and effective in reducing the amount of computations in transient thermal simulations, helping HotSpot users accelerate their thermal simulations without any accuracy loss. To the best of our knowledge, no such linear system based method that utilizes the fact of constant power trace during a fixed sampling interval has been proposed for architectural level thermal simulation.

We also want to emphasize that the matrices **A** and **B** in TILTS are calculated using the rk4 method. TILTS makes use of linear system formulation to eliminate unnecessary computations when the input power trace is constant over a fixed sampling interval. Therefore, TILTS does not lose any accuracy if the input power trace is discretized.

One characteristic of the TILTS method is that the number of FPMs is fixed for one temperature calculation interval. We can thus choose different temperature calculation intervals for different purposes. To reduce the overhead of temperature calculations, we can use a longer interval, or if a more accurate temperature estimation or finer granularity is required, we can use shorter intervals. In such cases, error-vs-computation overhead tradeoffs must be made for certain purposes.

We did some experiments to see the impact of the sampling interval length on temperature calculation accuracy. The calculated temperature for a sampling interval of $\Delta t =$ 3.333 μs served as the baseline. We tried 2, 4,..., 128 times of the base sampling interval Δt, and calculated the temperature errors. The maximum temperature errors for 24 SPEC2000 benchmarks are shown in Figure 5. We can see from Figure 5 that when the sampling interval is $16\Delta t = 53$ μs, the maximum temperature error is only 0.002 °C, which is clearly acceptable. When the sampling interval is $32\Delta t = 107$ μs, the maximum temperature error is only 0.003 °C, still acceptable. Even when the sampling interval is $128\Delta t = 427$ μs, the maximum temperature error is less than 0.04 °C. Thus, a very small sampling interval length does not always increase accuracy. A sampling interval of 100 μs will give almost the same results as the original interval. Therefore, we can use a sampling

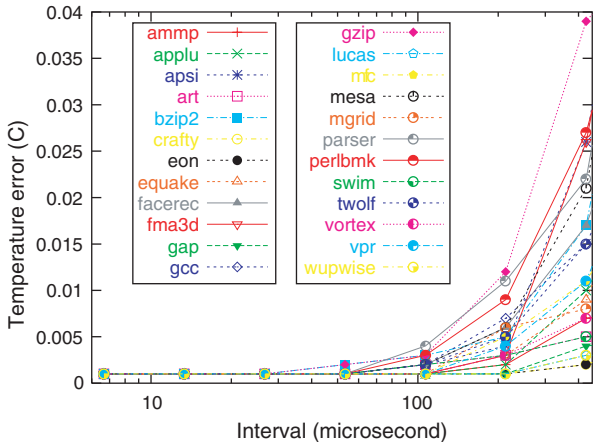

Fig. 5. Temperature errors for different sampling intervals.

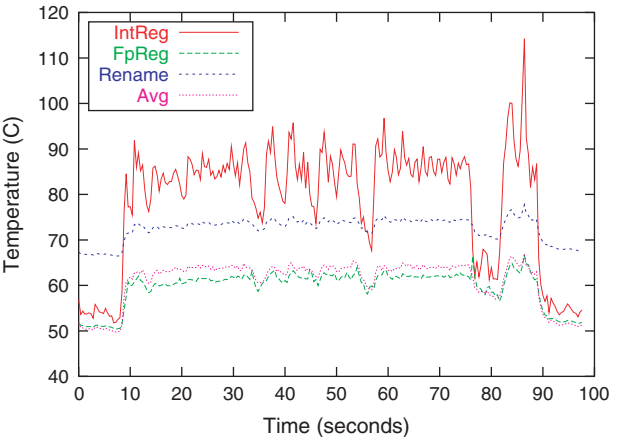

Fig. 6. Temperatures for *gcc* benchmark.

interval of 100 μs and still guarantee a sufficiently high accuracy in most cases while taking advantage of higher speedup when using TILTS or CONTILTS. The speedups of TILTS and CONTILTS for a 100 μs interval are shown in Table V.

8. APPLICATION TO RUNTIME TEMPERATURE MONITORING

To show the effectiveness and usefulness of TILTS, it has been applied to runtime temperature monitoring. Based on TILTS, we developed a lightweight runtime temperature monitoring tool called *Temptor*. Using the internal performance counters in the Pentium 4 microprocessor,[5] *Temptor* is able to estimate runtime temperature distributions in this chip.

Temptor uses hardware performance counters to measure processor activities, based on which the power consumption of each functional unit is estimated.[10] The temperature of each functional unit is then calculated using the TILTS method, providing detailed temperature distribution information at the architecture level.

Most importantly, the bottleneck of inefficient temperature calculations in[11] is resolved in *Temptor*. The system overhead of *Temptor* is negligible. In,[11] to reduce the large overhead of temperature calculations, instead of using the original rk4 method in Hotspot, the authors implement an improved Runge-Kutta-Fehlberg method (rkf) with adaptive step size to minimize the temperature calculation time. The rkf method is more efficient than the rk4 method despite its higher complexity. Even with the improved rkf method, the authors of Ref. [11] have observed more than 50 % performance degradation for some SPEC2000 benchmarks. They point out that a more efficient temperature calculation method is needed, which is exactly what the TILTS method attempts to do.

The experimental results for the *gcc* benchmark using *Temptor* are shown in Figure 6, which shows the temperatures of the *IntReg*, *FPReg*, and *Rename* units, and the average temperature of the whole chip. Without TILTS, temperature calculations for such a long time would affect the application performance significantly as reported in Ref. [11].

9. CONCLUSIONS

In this paper, we have proposed a new transient thermal simulation method for CPU chips, which utilizes the fact that the input power trace is discretized over a fixed sampling interval, thus allowing us to calculate transient temperatures on a chip for long simulation times. Based on a linear system formulation, our TILTS algorithm achieves the same accuracy as that of conventional thermal simulation tools, and is orders of magnitude faster. Compared to the HotSpot simulator, TILTS achieves speedups of 1300 for the processors in our experiments for an appropriate sampling interval of 100 μs. With some additional memory space, the revised algorithm CONTILTS is about 6000 times faster than the HotSpot simulator for the processors in our experiments. Since thermal aware designs and dynamic thermal management are becoming more important today, the proposed methods in this paper can prove to be very useful for temperature aware chip designs.

Acknowledgments: This work has been supported by NSF grant ITR-0205212. The authors wish to thank the LAVA group (the Laboratory for Computer Architecture at Virginia) at the University of Virginia for providing the HotSpot simulator, and especially Karthik Sankaranarayanan for providing the power traces of SPEC benchmark programs.

References

1. http://www.cds.caltech.edu/~murray/books/am05/wiki/index.php?title=linear_systems.
2. http://www.comsol.com/products/multiphysics.

3. http://www.flomerics.com/flotherm/.
4. http://www.spec.org.
5. IA-32 intel architecture software developer's manuals.
6. Model order reduction site at MIT, http://web.mit.edu/mor/.
7. D. Brooks and M. Martonosi, Dynamic thermal management for high-performance microprocessors. *Proceedings of the Seventh International Symposium on High Performance Computer Architecture (HPCA-7)* **(2001)**, pp. 171–182.
8. D. Brooks, V. Tiwari, and M. Martonosi, Wattch: a framework for architectural-level power analysis and optimizations. *Proceedings of the 27th annual international symposium on Computer architecture (ISCA)* **(2000)**, pp. 83–94.
9. W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and, S. Velusam, Compact thermal modeling for temperature-aware design. *Proceedings of the 41st Annual Conference on Design Automation (DAC)* **(2004)**, pp. 878–883.
10. C. Isci and M. Martonosi, Runtime power monitoring in high-end processors: Methodology and empirical data. In *36th ACM/IEEE International Symposium on Microarchitecture (MICRO-36)* **(2003)**, pp. 93–104.
11. K.-J. Lee and K. Skadron, Using performance counters for run-time temperature sensing in high-performance processors. In *IPDPS* **(2005)**.
12. H. Li, P. Liu, Z. Qi, L. Jin, W. Wu, S. X.-D. Tan, and J. Yang, Efficient thermal simulation for run-time temperature tracking and management. In *International Conference on Computer Design (ICCD)* **(2005)**, pp. 130–136.
13. P. Liu, Z. Qi, H. Li, L. Jin, W. Wu, S. X.-D. Tan, and J. Yang, Fast thermal simulation for architecture level dynamic thermal management. In *International Conference on Computer-Aided Design (ICCAD)* **(2005)**, pp. 639–644.
14. A. V. Oppenheim, A. S. Willsky, and N. S. Hamid, Signals and systems **(1996)**.
15. K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, Temperature-aware microarchitecture. *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA)* 2003, pp. 2–13.
16. Y. Yang, Z. Gu, C. Zhu, R. P. Dick, and L. Shang, ISAC: Integrated space-and-time-adaptive chip-package thermal analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **(2007)**, Vol. 26, pp. 86–99.

Yongkui Han

Yongkui Han *received the BS and MS degrees from the Department of Electronic Engineering, Tsinghua University in 1999 and 2002, respectively. Since 2002, he has been a PhD student at the University of Massachusetts, Amherst. The topic of the PhD program is power, energy, and temperature aware computer systems.*

Israel Koren

Israel Koren *received the DSc degree from the Technion-Israel Institute of Technology, Haifa, in 1975 in electrical engineering. He is currently a professor of electrical and computer engineering at the University of Massachusetts, Amherst. Previously he was with the Technion-Israel Institute of Technology. He also held visiting positions with the University of California at Berkeley, University of Southern California, Los Angeles, and University of California, Santa Barbara. He has been a consultant to several companies, including IBM, Intel, Analog Devices, AMD, Digital Equipment Corp., National Semiconductor, and Tolerant Systems. Dr. Korens current research interests include techniques for yield and reliability enhancement, fault-tolerant architectures, real-time systems, and computer arithmetic. He has published extensively in several IEEE Transactions and has more than 200 publications in refereed journals and conferences. He currently serves on the editorial board of the IEEE Transactions on VLSI Systems and of IEEE Computer Architecture Letters. He was a co-guest editor for the IEEE Transactions on Computers, special issue on high yield VLSI systems, April 1989, and the special issue on computer arithmetic, July 2000, and served on the editorial board of these transactions between 1992 and 1997. He also served as general chair, program chair, and program committee member for numerous conferences. He is the author of the textbook Computer Arithmetic Algorithms (A. K. Peters, Ltd., 2002), and a co-author of the textbook Fault Tolerant Systems (Morgan-Kaufmann, 2007). He is a fellow of the IEEE.*

C. M. Krishna

C. M. Krishna *received his Ph.D. from the University of Michigan in 1984, and has been on the faculty of the University of Massachusetts ever since. His research interests include real-time systems, computer networks, and power-aware computing.*