

Dynamic Thermal Management Considering Accurate Temperature-Leakage Interdependency

Bing Shi, Ankur Srivastava

The
Institute for
Systems
Research



A. JAMES CLARK
SCHOOL OF ENGINEERING

ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the A. James Clark School of Engineering. It is a graduated National Science Foundation Engineering Research Center.

www.isr.umd.edu

Dynamic Thermal Management Considering Accurate Temperature-Leakage Interdependency

Bing Shi and Ankur Srivastava
Department of Electrical and Computer Engineering
University of Maryland, College Park, USA

Abstract

In this paper, we develop an accurate dynamic thermal management DTM approach considering the interdependency of temperature and leakage. By modeling the leakage-thermal interdependence as a quadratic polynomial, we develop accurate analytical equations that capture the thermal transient. We also identify *all* the situations in which thermal runaway would occur, which should be avoided by DTM. We then present a discrete dynamic programming algorithm that performs thermal-aware task and speed scheduling using the model we derived. Compared to a linear leakage-thermal model [1], owing to our more accurate model, our scheme resulted in 18.2% better performance while maintaining the temperature below constraint.

1 INTRODUCTION

Dynamic thermal management is the process of runtime control of the processor control knobs such as V_{dd} , frequency, task scheduling etc for curtailing excessive chip temperature. With growing transistor densities dynamic thermal management DTM has become a very active topic of research [1], [2], [3], [4], [5].

With continued scaling, the impact of leakage power is growing as well. It has been shown that leakage and temperature are highly interdependent: higher temperature increases the leakage power which in-turn further increases the temperature. This interdependence has been known for years and several attempts have been made during design time to better estimate/control the leakage and temperature through various design decisions. For example [6] estimates the chip thermal and leakage profile while accounting for their interdependence. [7] estimates the chip leakage profile while accounting for thermal variability. Such approaches are mostly interested in analyzing the steady state behavior and use a combination of numerical and analytical techniques to capture this thermal-leakage relationship.

DTM on the other hand is a dynamic approach where we need to be able to predict the fashion in which temperature changes in time for a set of processor control knob options. For example, for a given choice of frequency and initial temperature, we would like to know the time at which the temperature will violate the thermal constraint. The complex relationship between leakage and temperature can result in two different dynamic behavior of temperature. Either the temperature increases or decreases and eventually settles to a steady state

value. Or, the temperature increases uncontrollably, thereby leading to thermal runaway. We would like to capture both transient behaviors. A steady state analysis approach will not suffice in such scenario. Recent work presented in [1], [8] captures the leakage, thermal dependence using a linear model. Although their approach is interesting, the linear model has limited accuracy and may mispredict thermal runaway. The approach in [3] uses a piecewise linear approximation which is an improvement over the pure linear model. But, they use approximation schemes to solve the thermal differential equations thereby leading to limited accuracy estimates.

In this paper, we develop an accurate approach by modeling the leakage-thermal interdependence as a quadratic polynomial. Existing work presented in [7] [9] has verified the accuracy of the quadratic model. We develop accurate *analytical* equations that capture the thermal transient. We also identify the situations in which thermal runaway would occur. Using our approach, any DTM scheme can estimate the temperature dynamics over time as a function of processor frequency and also if a specific choice of frequency will lead to thermal runaway. Our model is generic and can be applied in any DTM scheme regardless of whether a discrete or continuous mathematical optimization framework is used. We demonstrate the applicability of our model in a dynamic programming based DTM scheme for performing task speed assignment and scheduling. Specifically, the following contributions have been made. 1) An analytical model for the temperature dynamics in time parameterized w.r.t the initial temperature and the processor speed. 2) The model can predict the thermal behavior in time for any choice of initial temperature and frequency. 3) The model enumerates *all cases* where thermal runaway will definitely occur. These cases have been tabulated and could be used by any DTM scheme as system states that must be avoided. It is important to note that we have identified all the cases of thermal runaway. 4) Incorporated the model in a dynamic programming based DTM scheme for allocating task schedules and speed. Experimental results indicate that compared to a linear leakage-thermal model [1], owing to our more accurate model, our scheme resulted in 18.2% better performance while maintaining the temperature below constraint. We also show that using the model which does not consider the leakage-thermal interdependence in DTM results in underestimate of the real temperature and the solution will considerably violate the thermal constraint.

Although, the quadratic polynomial model is an approximation as well, using more accurate models (such as [10]) would force us to use numerical schemes for estimating the thermal dynamics. Such an approach would not be amenable to effi-

cient optimization during DTM. Also existing work presented in [7] [9] has already verified the accuracy of the quadratic model.

Rest of the paper is organized as follows. Section 2 develops the thermal model considering leakage-thermal interdependence. We then explore a dynamic programming based thermal-aware task speed scheduling problem using our model in section 3. The experimental result is given in section 4.

2 RC THERMAL MODEL CONSIDERING LEAKAGE

2.1 Single core RC thermal model

The thermal behavior of a processor can be modeled by an RC pair. Here we adopt the lumped RC thermal model similar to the one used in [4] [11]. As shown in figure 1, voltage represents the temperature, and current represents the power consumption in the processor. The resistance represents a potential heat transfer path through the packaging, while the capacitance indicates the ability of the processor to store heat.

According to the RC thermal model in figure 1, the thermal behavior of the processor can be described as the following differential equation:

$$\frac{dT}{dt} = -\frac{T - T_{amb}}{RC} + \frac{P_{total}}{C} \quad (1)$$

where $T(K)$ is the chip temperature, $T_{amb}(K)$ is the ambient temperature, $P(W)$ is the total power consumption of the chip, $R(K/W)$ is the thermal resistance, and $C(J/K)$ is the thermal capacitance. By assuming that the power does not change in time (just for the time being) and solving differential equation 1, the chip temperature T is formulated as equation 2, assuming the initial temperature is T_0 .

$$T = (T_0 - T_{amb} - RP_{total})e^{-t/RC} + RP_{total} + T_{amb} \quad (2)$$

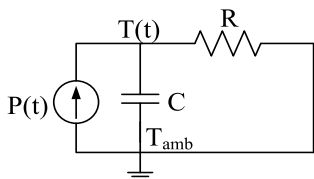


Figure 1: RC Thermal Model

2.2 Leakage and Temperature Dependency

As the IC technology scales down to deep sub-micron domain, leakage becomes a large component of power consumption. Today, up to 50% (or even more) of the total power consumption is leakage power [12]. Therefore, simply approximating the total power consumption by dynamic power [13] [14] will lead to an underestimation of the real temperature.

There are three components of leakage power: Band-To-Band-Tunneling (BTBT) leakage, sub-threshold leakage and

gate leakage. The BTBT leakage and sub-threshold leakage increases with temperature while the gate leakage is rather insensitive to temperature.

The works in [7] [9] model the thermal dependence of leakage as a polynomial function of temperature:

$$P_l = a \cdot T^2 - b \cdot T + d \quad (3)$$

where a , b and d are technology dependent parameters and could be generated by polynomial approximation of the accurate leakage model in [10].

From equation 2 and 3, we can see that temperature and leakage power influence each other. That is, higher leakage power leads to higher temperature while higher temperature will further increase the leakage power. Because of this dynamic cross coupling of temperature and leakage power, simply assuming that leakage power P_l does not change in time and substituting $P_{total} = P_d + P_l$ into equation 2 cannot capture the dynamic nature of the interdependence between temperature and leakage.

In this paper, we first develop a thermal model that take leakage power and its dynamic interdependence with temperature into consideration. Here we assume that dynamic power is linearly dependent on frequency, that is $P_d = kf$, where f is the frequency and k is a constant (although our method is trivially generalizable to more accurate models where $P_d = kf^\alpha$). We get the expression of total power consumption as shown in equation 4.

$$P_{total} = P_l + P_d = a \cdot T^2 - b \cdot T + d + kf \quad (4)$$

By substituting the power model in equation 4 into the thermal differential equation 1, we get the new thermal differential equation as shown in equation 5.

$$\begin{aligned} \frac{dT}{dt} &= -\frac{T - T_{amb}}{RC} + \frac{P_l + P_d}{C} \\ &= -\frac{T - T_{amb}}{RC} + \frac{a \cdot T^2 - b \cdot T + d + kf}{C} \\ &= \frac{a}{C}T^2 - \left(\frac{b}{C} + \frac{1}{RC}\right)T + \frac{Rkf + Rd + T_{amb}}{RC} \\ &= A \cdot T^2 + B \cdot T + D(f) \end{aligned} \quad (5)$$

where

$$A = \frac{a}{C} \quad B = -\left(\frac{b}{C} + \frac{1}{RC}\right) \quad D(f) = \frac{Rkf + Rd + T_{amb}}{RC} \quad (6)$$

Assuming the frequency f does not change in time (for the time being), we would like to solve the differential equation to derive the underlying equation governing the thermal transient. The temperature profile described in differential equation 5 can be derived by solving the following integration:

$$\int_{T_0}^T \frac{dT}{AT^2 + BT + D(f)} = \int_0^t dt \quad (7)$$

The solution of equation 7 depends on the values of parameters A , B , $D(f)$ and initial temperature T_0 .

2.3 Thermal Model with Leakage Power

In this section we present our analytical model that captures the temperature dynamic w.r.t time parameterized using initial temperature T_0 and processor frequency f . This model is created by solving differential equation 5 and is a complete model that captures all combinations of input parameters T_0 and f . Using our model, we also enumerate *all* the cases for which thermal runaway occurs. Our analytical model could be used by any DTM scheme. We first define the following:

$$f_m = \frac{1}{k} \left(\frac{(Rb+1)^2}{4aR^2} - \frac{T_{amb}}{R} - d \right)$$

$$T_{s1} = \frac{-B + \sqrt{B^2 - 4AD(f)}}{2A}, \quad T_{s2} = \frac{-B - \sqrt{B^2 - 4AD(f)}}{2A} \quad (8)$$

Since the model is rather complex, we first enumerate the model and then prove its correctness by presenting the derivation. The different input parameter combinations T_0, f have been partitioned into the following cases.

1. Case 1:
 - (a) Case 1a: $f < f_m$ and $T_0 \leq T_{s1}$
 - (b) Case 1b: $f < f_m$ and $T_0 > T_{s1}$
2. Case 2:
 - (a) Case 2a: $f = f_m$ and $T_0 \leq -B/2A$
 - (b) Case 2b: $f = f_m$ and $T_0 > -B/2A$
3. Case 3: $f > f_m$

Note that these cases enumerate all possible input combinations. Now we highlight the model for each of these cases.

Case 1a:

$$T(t) = \frac{\sqrt{B^2 - 4AD(f)}}{2A} \frac{x_0 + 1 + (x_0 - 1)e^{\sqrt{B^2 - 4AD(f)}t}}{x_0 + 1 - (x_0 - 1)e^{\sqrt{B^2 - 4AD(f)}t}} - \frac{B}{2A} \quad (9)$$

where

$$x_0 = \frac{2AT_0 + B}{\sqrt{B^2 - 4AD(f)}} \quad (10)$$

Case 1b: The temperature will subject to thermal runaway.

Case 2a:

$$T(t) = \frac{2AT_0 + B}{-A(2AT_0 + B)t + 2A} - \frac{B}{2A} \quad (11)$$

Case 2b: The temperature will subject to thermal runaway.

Case 3: The temperature will subject to thermal runaway.

We will derive all these cases subsequently. It is important to note that our analytical model has the following benefits.

- a. Based on the initial temperature, one can choose the frequency such that thermal runaway will not occur;
- b. One can choose the frequency such that the desired temperature profile can be achieved;

c. The equations of temperature profile are purely analytical;

d. These are optimal rules that consider all possible cases for the temperature profile. If we develop thermal-aware frequency policies based on these rules, we are guaranteed that the temperature will never violate the constraints and/or the runaway condition will never occur.

This model can be easily incorporated in many DTM schemes. For example a DTM scheme can use it to evaluate a set of possible frequency choices for a given initial temperature. This model can be used to estimate the amount of time it will take to reach a desired temperature for a given f, T_0 assignment. It can also be incorporated in a non-linear programming based optimization framework. Next we present the derivation.

2.3.1 Case 1:

Note that in this case $f < f_m$, that means $B^2 - 4AD(f) > 0$. So function $AT^2 + BT + D(f) = 0$ has two real roots T_{s1} and T_{s2} (described in equation 8) under this scenario. The integration in equation 7 can be solved as follows:

$$\frac{1}{A} \int_{T_0}^T \frac{dT}{(T - T_{s1})(T - T_{s2})} = \int_0^t dt \quad (12)$$

We define:

$$x(t) = \frac{2AT(t) + B}{\sqrt{B^2 - 4AD(f)}} \quad x_0 = x(0) \quad (13)$$

we get

$$\frac{2}{\sqrt{B^2 - 4AD(f)}} \int_{x_0}^x \frac{1}{(x-1)(x+1)} dx = \int_0^t dt$$

$$\Rightarrow \ln \frac{x-1}{x+1} \Big|_{x_0}^x = \sqrt{B^2 - 4AD(f)} t \Big|_0^t \quad (14)$$

$$\Rightarrow x(t) = \frac{x_0 + 1 + (x_0 - 1)e^{\sqrt{B^2 - 4AD(f)}t}}{x_0 + 1 - (x_0 - 1)e^{\sqrt{B^2 - 4AD(f)}t}}$$

By substituting equation 14 into the expression of $x(t)$ in equation 13, we can get the expression for the temperature as shown in equation 9. Here x_0 is an initial condition which is a function of both the initial temperature T_0 and frequency f . It will decide the temperature profile.

There are two subcases associated with different initial condition T_0 . We describe Case 1b first and then Case 1a.

Case 1b $f < f_m$ and $T_0 > T_{s1}$: It can be shown that for this case $x_0 > 1$. If we plot equation 9 with this as x_0 , we find that the temperature $T(t)$ quickly diverges to infinity. This case is shown in figure 2(c). As we can see from the figure, the temperature will diverge and increase to infinity in a very short time, which results in thermal runaway. So we should avoid this case.

Case 1a $f < f_m$ and $T_0 \leq T_{s1}$: It can be shown that for this case $x_0 \leq 1$. When x_0 equals exactly to 1, the temperature will stay at $T = T_{s1}$ (this can be illustrated by simply substituting the appropriate values in equation 9). But the problem with this case is, as long as there is a tiny fluctuation which

make the temperature increase a very small value, the processor will jump out of steady state and enter case 1b, which is thermal runaway. So in practical implementation, we should avoid this case as well.

When $x_0 < 1$ (which happens when $T_0 < T_{s1}$), equation 9 governs the thermal dynamics. In fact, it can be proved that temperature must always converge to T_{s2} as $t \rightarrow \infty$. For the sake of illustration, this case is plotted in figure 2(a) and 2(b). For the same thermal equation 9, the actual thermal behavior is different for $T_{s2} < T_0 < T_{s1}$ (plotted in figure 2(a)) than for $T_0 \leq T_{s2}$ (plotted in figure 2(b)). Both cases converge to T_{s2} .

2.3.2 Case 2:

Here $f = f_m$, therefore $B^2 - 4AD(f) = 0$. So the function $AT^2 + BT + D(f) = 0$ has one real root $-B/2A$. By solving equation 7, we obtain the following:

$$\begin{aligned} \frac{1}{A} \int_{T_0}^T \frac{dT}{(T + \frac{B}{2A})^2} &= \int_0^t dt \\ \Rightarrow -\frac{1}{A} \cdot \frac{1}{T + \frac{B}{2A}} \Big|_{T_0}^T &= t \Big|_0^t \\ \Rightarrow T(t) &= \frac{2AT_0 + B}{-A(2AT_0 + B)t + 2A} - \frac{B}{2A} \end{aligned} \quad (15)$$

There are two subcases in this situation depending on the initial temperature T_0 .

Case 2a $T_0 \leq -B/2A$: 1) If the initial temperature T_0 is lower than the value $-B/2A$, the temperature will slowly increase to $-B/2A$ as in figure 3(a). 2) If the initial temperature equals exactly to $-B/2A$, the temperature will stay at the initial temperature $-B/2A$. So $-B/2A$ is the steady state temperature when $f = f_m$. Hence in this situation the thermal dynamics is governed by equation 15 (or equation 11).

Case 2b $T_0 > -B/2A$: When we incorporate this case in equation 15, we find that the temperature quickly diverges to infinity (as shown in figure 3(b)), which results in thermal runaway. We should also avoid this case.

2.3.3 Case 3:

In this case, $B^2 - 4AD(f) < 0$ since $f > f_m$. We can solve the integration in equation 7 and get the temperature profile by following steps:

$$\begin{aligned} \int_{T_0}^T \frac{dT}{AT^2 + BT + D(f)} &= \int_0^t dt \\ \Rightarrow \frac{4A}{4AD(f) - B^2} \int_{T_0}^T \frac{dT}{(\frac{2A}{\sqrt{4AD(f) - B^2}}(T + \frac{B}{2A}))^2 + 1} &= \int_0^t dt \\ \Rightarrow \frac{2}{\sqrt{4AD(f) - B^2}} \arctan \frac{2A}{\sqrt{4AD(f) - B^2}} (T + \frac{B}{2A}) \Big|_{T_0}^T &= t \Big|_0^t \\ \Rightarrow T(t) &= \frac{\sqrt{4AD(f) - B^2}}{2A} \tan(y(t)) - \frac{B}{2A} \end{aligned} \quad (16)$$

$$y(t) = \frac{\sqrt{4AD(f) - B^2}}{2} t + \arctan\left(\frac{2A}{\sqrt{4AD(f) - B^2}}(T_0 + \frac{B}{2A})\right) \quad (17)$$

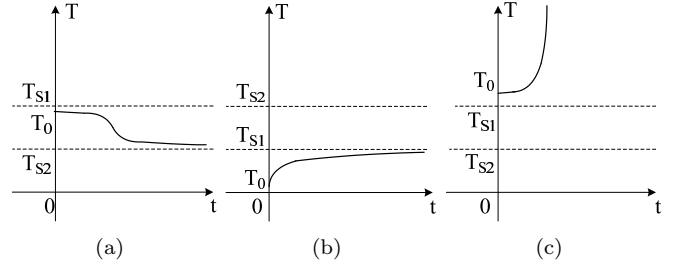


Figure 2: (a)(b) represent case 1a, (c) represents case 1b

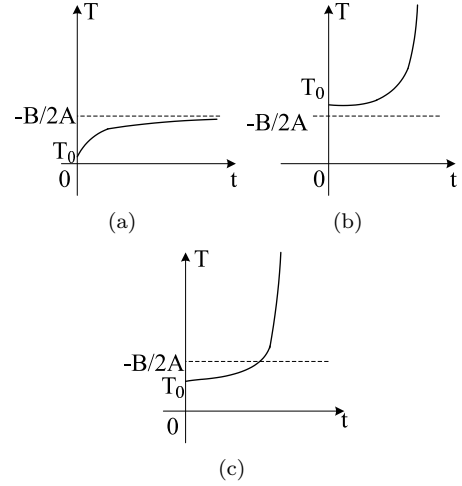


Figure 3: (a): Case 2a, (b): Case 2b, (c): Case 3

As we can see from equation 16 and figure 3(c), the temperature profile in this case follows a tangent function $\tan(y(t))$ which increases very rapidly towards infinity when $y(t)$ approaches $\pi/2$. This happens regardless of the initial temperature T_0 . So we should avoid this situation as well. As a result, we set the maximum frequency $f_{max} = f_m$, and the frequency should never exceed f_{max} .

3 THERMAL AWARE SPEED AND TASK SCHEDULING

Following the discussion of previous section, we can use the rules derived there in any dynamic thermal management scheme to constraint the frequency so that thermal runaway condition does not occur. Also we can create temperature profiles such that thermal constraints are never violated. Our rules can be used in both continuous (mathematical) and discrete optimization frameworks.

In this section, we present a discrete dynamic programming algorithm that performs thermal-aware task and speed scheduling using the model we derived in the previous section, which accounts for the non-linear leakage and temperature dependence. Our primary objective is not to approach the DTM problem with a new algorithmic technique, but to empower a large class of DTM algorithms with a better way of predicting the thermal impact of their decisions.

3.1 Problem Formulation

Given:

- (1) a periodic sequence of n tasks $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$, whose corresponding workload is $\mathcal{W} = \{w_1, w_2, \dots, w_n\}$, respectively;
- (2) m distinct frequency levels $\mathcal{F} = \{f_1, f_2, \dots, f_m\}$, so the execution time of the i 'th task using frequency f_j is w_i/f_j ;
- (3) l sleep state where the frequency $f = 0$, and the length of l sleep states are l_1, l_2, \dots, l_l ;
- (4) the initial temperature is T_0 ,

We would like to assign a frequency for each task and a potential sleep state after each task so that the total execution time of the n tasks is minimized. During the execution of each task, the frequency does not change. Also the temperature should never exceed the maximum temperature constraint T_{max} and after the execution of all tasks, the final temperature decreases below T_0 .

This problem is NP-complete [11]. But we present a dynamic programming method to find solutions reasonably close to optimal. Many versions of this problem have been studied before [4] [11]. Note that this problem is a demonstrator of the practicality of the model we developed in the previous section.

3.2 Dynamic Programming with Fixed Task Order

We first explore the optimization problem assuming that the task order is already given. In section 3.3, we will incorporate task order rescheduling to this problem. The steps to solve this speed scheduling optimization problem with dynamic programming is as follows:

- (1) Sort the frequency in ascending order so that $f_1 < f_2 < \dots < f_m$.
- (2) Traverse the tasks from first to the last.
- (3) For the first task, the initial processor temperature is T_0 . For each of the frequency states, use the model to predict the thermal profile and/or potential runaway state. Note that the task will take time w_0/f_i to execute for each frequency. Prune out frequencies that result in runaway or violation of thermal constraint. It is noteworthy that if frequency f_j violates the constraints then frequencies $f_{j+1} \dots f_m$ will violate as well and do not need to be evaluated. This will give us a set of solutions with varying latency and final temperature. Each solution k is characterized by a latency and temperature tuple (d_k, T_k) . Here latency d_k is the time it takes to finish task 1 and final temperature T_k is the temperature upon completion of task 1 for this specific frequency assignment. Among these solutions we can prune out those which are clearly sub-optimal than others (higher latency and higher final temperature). This pruning process results in a left-over set of co-optimal solutions $\mathcal{C} = \{(d_1, T_1), (d_2, T_2), \dots, (d_K, T_K)\}$ as illustrated in figure 4.

(4) Now evaluate the possibility of adding a potential sleep state. For each co-optimal solution from the previous set \mathcal{C} , incorporate the possibility of adding each of the l sleep states. For a solution $(d_k, T_k) \in \mathcal{C}$, adding a sleep state l_i would make the overall latency become $d_k + l_i$. The final temperature at the end of the sleep state (l_i time units) could be computed using our model with T_k as the initial temperature and $f = 0$. Combining each co-optimal solution in \mathcal{C} with each sleep state in l will give us a set of new solutions with total latency and final temperature as parameters. Once again we can generate

a new co-optimal set \mathcal{C} of solutions using the pruning criteria in figure 4.

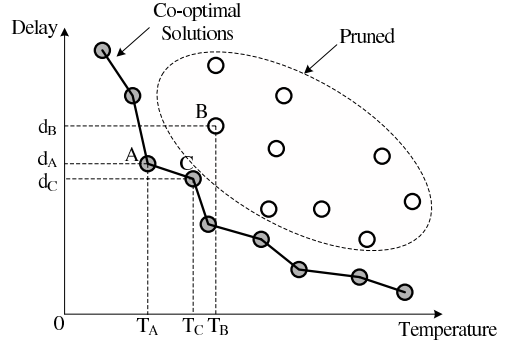


Figure 4: Multi-objective optimization

(5) Now the set \mathcal{C} represents the total latency vs final temperature tradeoff for finishing task 1. Note that none of these solutions violate the thermal constraint or result in thermal runaway.

(6) Now pick the next task J_2 . For each solution $(d_k, T_k) \in \mathcal{C}$, use T_k as the initial temperature and compute the total latency of running J_2 at each of the available frequencies, plus the latency of previous task (and sleep states). Also compute the thermal profile using our model. Prune out solution that result in thermal runaway and also those that violate the thermal constraint. Among the remaining set of solutions with total latency and final temperature, prune out sub-optimal solutions using the criteria of figure 4. This will give us a new set of co-optimal solutions $\mathcal{C} = \{(d_1, T_1), (d_2, T_2), \dots, (d_K, T_K)\}$ where d_k is the total latency of executing J_1 and J_2 with potential sleep states in between and T_k is the final temperature. Now repeat step (4) for adding new sleep states.

(7) Repeat step (6) for all subsequent tasks.

(8) After the last task pick the solution with smallest latency whose final temperature is less than T_0 .

With these steps, we are guaranteed to find the optimal solution since during pruning, we only prune those solutions that violate the temperature constraints and the suboptimal solutions, and we did not prune any potential optimal solutions. Although this method is not proved mathematically with polynomial runtime, in practical implementation, as will be shown in the simulation result, it does prune many solutions and the runtime is fast. We can also use more strict pruning criteria to prune more co-optimal solutions to gain runtime. However, we will not explore it in this paper since it is not the focus of this paper.

3.3 Considering Task Rescheduling

Rescheduling the task order may further decrease the task execution time. So here we combine task rescheduling with speed scheduling in this problem to find better task order and the corresponding frequency assignment for each task. Once again our primary objective is to demonstrate the applicability of our model in different DTM paradigms.

3.3.1 Dynamic Programming with Task Scheduling

To consider task rescheduling, we also use the dynamic programming method, but add one more dimension of choice to the dynamic programming: task to be executed.

(1) Just as before, we start with initial temperature T_0 . Instead of evaluating just the first task J_1 , we evaluate the possibility of starting from any of the n tasks. For each of the tasks as the first task, we get a set of co-optimal solutions with (d_k, T_k) as parameters. Following this we can add potential sleep states similar to the previous algorithm. This set of solutions (for all tasks) represent that one task has been completed (although the specific task may be different for each solution).

(2) This set of co-optimal solutions may be too many. We can use several heuristic based pruning techniques to reduce the potential solution space without potentially losing out on optimality. The pruning mechanism we used in this paper quantifies each solution by the total workload that has been completed by it. Hence now each solution (d_k, T_k) also has an associated *total workload executed* parameter W_k . Now we can perform pruning of these solutions based on a criteria similar to the one in figure 4. Note that in this case higher workload with small latency and temperature are desirable properties.

(3) Now we have $n - 1$ tasks left. Repeat the above steps using the current set of co-optimal solutions and choosing the next task from the left-over set of $n - 1$ tasks. Note that the set of leftover tasks depends upon which tasks were scheduled in the current solution being considered from the co-optimal set.

(4) Repeat step (3) till all tasks are considered.

(5) Among all solutions pick the one with smallest latency and final temperature less than T_0 .

However, we found that even after our heuristic pruning, there are still many co-optimal solutions left. So in the next section, we develop a heuristic method to further reduce the number of co-optimal solutions.

3.3.2 Heuristic to Improve the Computation Speed

The heuristic works as follows.

(1) Sort the tasks in ascending order of their workload.

(2) Divide the tasks from the previous sorted list $\mathcal{G} = \{g_1, g_2, \dots, g_z\}$ into z contiguous groups. Each group has n/z tasks.

(3) When choosing the next task to be executed (see step (4) of the previous algorithm), instead of considering all the tasks that has not been executed, here we only consider one task from each group. That is, we pick one task from each group that has not been scheduled yet as candidates for the next task to be executed (the task in each group is selected in ascending order of workload). This guarantees that both tasks with smaller and larger workload are in the candidate tasks each time.

With this grouping, the number of choices can be greatly reduced since the number of groups z is much smaller than the number of tasks n . Besides, we can flexibility select the number of groups z to make a tradeoff between the optimality, i.e., total task execution time and the computation complexity, i.e., total number of co-optimal solutions.

4 EXPERIMENTAL RESULTS

We obtained the thermal model parameters R, C from [5]. The initial temperature $T_0 = 50^\circ\text{C}$ and the maximum temperature constraint $T_{max} = 80^\circ\text{C}$. In our simulation, we use three discrete frequency $f_1 = 1 \times 10^9\text{Hz}$, $f_2 = 3 \times 10^9\text{Hz}$, $f_3 = 5.5 \times 10^9\text{Hz}$ and two sleep states whose length are 20sec and 50sec , respectively.

4.1 Comparison 1: With existing leakage-temperature models

We first compare the frequency schemes and corresponding temperature profile generated by the dynamic programming based algorithm when used in conjunction with 1) our leakage aware thermal model, 2) the model which does not consider the interdependence of leakage and temperature (assumes leakage as a constant value calculated at room temperature) and 3) linear leakage-temperature model [1], [8]. For this comparison, we only focus on the fixed task order case.

We took a set of 20 tasks with workloads ranging from 1×10^{11} to 2×10^{12} units. The tasks are scheduled in ascending order of their workload. The optimal frequency scheme obtained by using our model is shown in figure 5(a). The first three tasks run at the highest frequency f_3 so that the temperature heats up to as high as possible (without violating the temperature constraint), and then the frequency reduce to f_2 since continue running at f_3 at the fourth task will make the temperature exceed T_{max} . From the fourth task, the processor runs at f_2 continuously until at the last task, the frequency reduces to f_1 so that the final temperature goes back to T_0 . Besides, no sleep state is used between any two tasks. **The total task execution time is 8242sec.** By applying this frequency scheme to the accurate thermal-leakage model in [10], the temperature profile is shown in figure 5(b). As we expect, the temperature will never violate the temperature constraint for this scheme (figure 5(b)). The total runtime is 0.2sec.

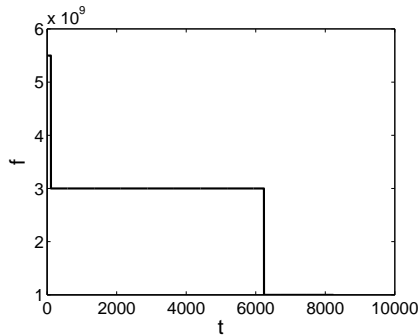
For comparison we use the model which does not consider the interdependence between leakage and temperature (leakage is assumed to be a constant value computed at room temperature). This model under-estimate the actual temperature for a given frequency assignment. Hence it over-clocks the system thereby leading to higher temperature. Figure 6(a) shows the frequency scheme and figure 6(b) highlights the real temperature profile of this speed assignment scheme when applied to the accurate leakage model in [10]. It can be seen that the solution considerably violates the thermal constraint.

Finally we also compare the results obtained by using the linear leakage-thermal interdependency model [1] [8]. This linear model was generated by performing linear regression on the real thermal-leakage model [10]. The frequency policy and the associated real thermal profile is shown in figure 7. Although this model did not violate the thermal constraint, the overall execution latency of the tasks was 9742.4 seconds. **This is 18.2% more than our model.**

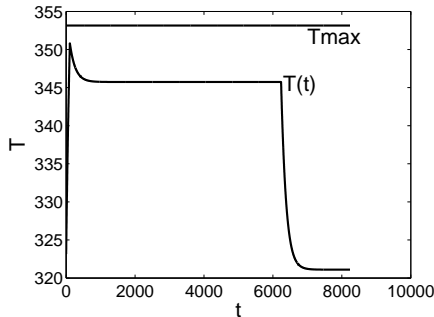
Hence it can be seen that our model captures the thermal transient more accurately thereby leading to improved solutions.

Table 1: Task latency and computational complexity comparison

	Fix order		Fully reschedule			Group ($z=5$)			Group ($z=10$)		
	Latency	Runtime(s)	Latency	Runtime(s)	Gain(%)	Latency	Runtime(s)	Gain(%)	Latency	Runtime(s)	Gain(%)
Benchmark 1	1	0.20	0.8504	1055.22	14.96	0.8638	24.91	13.62	0.8561	121.86	14.39
Benchmark 2	1	0.07	0.8459	574.65	15.05	0.8713	11.15	12.87	0.8534	69.87	14.66
Benchmark 3	1	0.02	0.8495	150.41	15.05	0.8580	5.57	14.20	0.8504	16.42	14.96
Benchmark 4	1	0.01	0.8480	60.30	15.20	0.8448	1.97	15.52	0.8528	6.90	14.72
Benchmark 5	1	0.01	0.8480	42.77	15.20	0.8510	1.73	14.90	0.8510	6.03	14.90
Benchmark 6	1	<0.01	0.8480	15.86	15.20	0.8569	0.65	14.31	0.8412	2.99	15.88
Benchmark 7	1	<0.01	0.8480	6.75	15.20	0.8644	0.29	13.56	0.8406	1.30	13.94
Benchmark 8	1	<0.01	0.8480	6.63	15.20	0.8402	0.24	15.98	0.8375	1.29	15.95

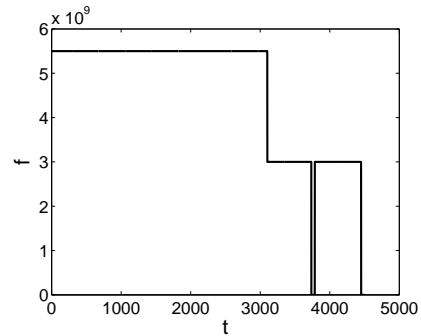


(a)

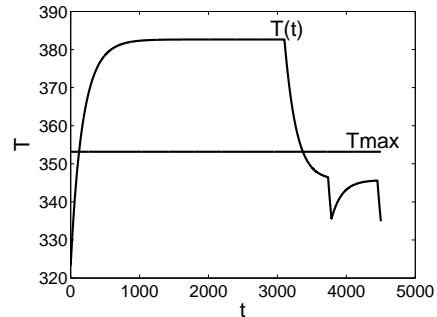


(b)

Figure 5: (a) Optimal frequency scheme for fixed order tasks using our model, (b) Real temperature profile for this scheme



(a)



(b)

Figure 6: (a) Optimal frequency scheme for fixed order tasks using model without leakage, (b) Real temperature profile for this scheme

4.2 Comparison 2: Fixed order, full rescheduling and grouping

Now we compare the results obtained with and without task rescheduling. We generated a set of benchmarks with total task workloads ranging from 2×10^{13} to 1×10^{15} . We compared the results obtained by 1) fixed task order assuming the task schedule was increasing w.r.t. workloads, 2) full task rescheduling in which no task grouping was done (see section 3.3) and 3) task rescheduling when they are grouped into 5 and 10 clusters (see section 3.3).

In table 1, we compare the total task execution time (*la-*

tency), which is normalized w.r.t. the latency of fixed task order scheme, and the program running time for these schemes. None of these schemes violate the thermal constraint. From the table, we can see that when we fully consider task rescheduling the task latency can reduce about 15.13%, however, the runtime is about 239 sec on average which is 5000 more than the fixed order method. But the grouping works quite well. For example, when the number of group $z = 5$, computing the best schedule needs only 5.8sec on average, while the average task latency can still improve 14.37%.

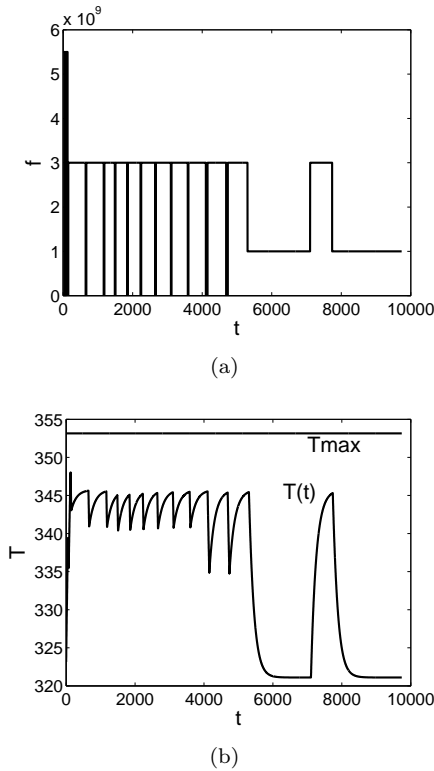


Figure 7: (a) Frequency scheme for fixed order tasks using linear leakage model, (b) Real temperature profile for this scheme

5 CONCLUSION

In this paper, we developed an accurate dynamic thermal management approach considering the interdependency of temperature and leakage power. We also present a discrete dynamic programming algorithm that performs thermal-aware task and speed scheduling using the model we derived. The experimental result shows that compared to a linear leakage-thermal model, owing to our more accurate model, our scheme resulted in 18.2% better performance while maintaining the temperature below constraint.

6 Acknowledgments

We would like to thank NSF grant CCF 0937865 for supporting part of this research.

References

[1] G. Quan and Y. Zhang, “Leakage aware feasibility analysis for temperature-constrained hard real-time periodic tasks,” in *21st Euromicro Conference on Real-Time Systems (ECRTS '09)*, pp. 207–216, 2009.

[2] R. Rao, S. Vrudhula, and N. Chang, “An optimal analytical solution for processor speed control with thermal

constraints,” in *Proc. of Intl. Symp. on Low Power Electronics and Design (ISLPED'06)*.

- [3] R. Rao and S. Vrudhula, “Performance optimal processor throttling under thermal constraints,” in *Proc. of Intl. Conf. on Compilers Architectures and Synthesis for Embedded Systems (CASES'07)*.
- [4] T. Chantem, X. S. Hu, and R. P. Dick, “Online work maximization under a peak temperature constraint,” in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design (ISLPED 09)*, pp. 105–110, 2009.
- [5] K. Skadron, T. Abdelzahery, and M. R. Stan, “Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management,” in *Proc. of Intl. Symp. on High-Performance Computer Architecture (HPCA '02)*.
- [6] T. Sato, J. Ichimiya, N. Ono, and M. Hashimoto, “On-chip thermal gradient analysis considering interdependence between leakage power and temperature,” in *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, pp. 3491–3499, 2006.
- [7] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, “Full chip leakage estimation considering power supply and temperature variations,” in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED'03)*, pp. 78 – 83, 2003.
- [8] Y. Liu, R. Dick, L. Shang, and H. Yang, “Accurate temperature-dependent integrated circuit leakage power estimation is easy,” in *Proceedings of the conference on Design, automation and test in Europe (DATE'07)*, pp. 1–6, 2007.
- [9] R. Ayoub and T. Rosing, “Predict and act: Dynamic thermal management for multi-core processors,” in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED'09)*, pp. 99–104, 2009.
- [10] L. He, W. Liao, and M. R. Stan, “System level leakage reduction considering the interdependence of temperature and leakage,” in *Design Automation Conference (DAC'04)*.
- [11] S. Zhang and K. S. Chatha, “Approximation algorithm for the temperature-aware scheduling problem,” in *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design (ICCAD 07)*, pp. 281–288, 2007.
- [12] N. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, and V. Narayanan, “Leakage current: Moores law meets static power,” *IEEE Computer Society*, vol. 36, no. 12, pp. 68–75.
- [13] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D. Micheli, “Temperature-aware processor frequency assignment for mpsoes using convex optimization,” in *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis (CODES'07)*, pp. 111–116, 2007.
- [14] S. Wang and R. Bettati, “Delay analysis in temperature-constrained hard real-time systems with general task arrivals,” in *Proceedings of the 27th IEEE International Real-Time Systems Symposium*, pp. 323–334, 2006.