IRWIN AND JOAN JACOBS
CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES

An Integer Linear Programming Approach for the Analysis of DTM Strategies

Tomer M. London, Uri C. Weiser and Aviad Cohen

CCIT Report #745
September 2009

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL

CCIT REPORT #745 September 2009

An Integer Linear Programming Approach for the Analysis of DTM Strategies

Tomer M. London, Uri C. Weiser, Aviad Cohen

Abstract— As the number of cores in multicore processors and their operating frequency continue to grow, processor power consumption increases and leads to an escalation in chip temperatures. This escalation causes today's multicore processors' performance to be limited by chip thermal constraints rather than process technology or circuit design. As a result, Dynamic Thermal Management (DTM) has an increasingly significant role in the design of new microprocessors.

We propose a new way to design and quantitatively analyze DTM strategies. Using Integer Linear Programming, we compute optimal offline DTM strategies that achieve maximal performance and meet thermal constraints. By analyzing the optimal strategies, we are able to calculate the upper bound on DTM performance, to identify optimal strategy patterns and to compare the thermal limitations of several microprocessor layout designs.

We employ offline Optimal DTM analysis on the case of Multicore Task Scheduling DTM. We find that this analysis suggests that a layout of many small cores is more thermally efficient than a layout of several large cores. The analysis further suggests specific task scheduling algorithm guidelines that maximize performance under thermal constraints. In addition, we compute and analyze the optimal multicore task scheduling strategies for DVS/DFS-based mechanism and Stop&Go DTMs

1. Introduction

While multicore processors leverage the parallelism of multi-threaded applications to achieve higher performance, the increased on-chip parallelism results in greater power densities that may lead to a dangerous increase in chip temperatures [1]. As high temperatures can cause functional failures, reliability reduction and changes in circuit timing [5], today's multicore processors' performance is most often limited by thermal constraints rather than process technology or circuit design constraints [1].

In order to mitigate the thermal problem, computer architects design Dynamic Thermal Management (DTM) [3] mechanisms that monitor and dynamically reduce the chip's temperatures via regulation of the on-chip power distribution. Several DTM techniques had been proposed such as Stop&Go (global clock toggling) [3], local toggling [5], activity migration [4], thread migration, dynamic voltage/frequency scaling and multicore Temperature Aware Task Scheduling [11]. In most cases, the DTM design process is rather empirical and involves using 'trial and error' with various simulators to fine-tune algorithm parameters. In general, this design process does not achieve the optimum performance.

In this paper, we propose a new approach to design and quantitatively analyze DTM strategies and employ this approach on multicore task scheduling DTM. We use use Integer Linear Programming [23] to compute offline optimal DTM strategies that achieve maximal performance within chip temperature limits. We then analyze the behaviors of the optimal offline strategies and find specific strategy patterns that are desirable for DTM behavior. Finally, we use these patterns to formulize the on-line behavior of the DTM mechanism to be designed.

In addition to formulating the desired behavior of online task scheduling, we compute its upper bounds on performance and devise a method to compare and identify the thermal limitations of microprocessor layout designs. We find that the multicore layout has a crucial effect on the thermal limitation of the microprocessor, and by comparing several multicore layouts we are able to identify several simple layout design guidelines in the case of multicore task scheduling. Besides its application to multicore task scheduling DTM, we show that our approach also suggests important design guidelines for other DTMs such as multicore DVS/DFS-based mechanism and Stop&Go (global clock toggling) DTMs.

The rest of this paper is organized as follows: In section 2, we discuss previous work on DTM design and analysis, section 3 contains the formulation of the Thermal Multicore Task Scheduling DTM optimization problem and the model we use. In section 4, we describe the methods we use to compute the optimal solution while in section 5 our results for Multicore task scheduling, multicore DVS/DFS-based mechanism and Stop&Go DTM mechanisms are presented. Finally, in section 6, we discuss our conclusions and future research directions.

- *T.M. London is with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel. E-mail: tomerlondon@gmail.com*
- *U.C. Weiser is with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel.*
- *A. Cohen is Intel Israel (84), Haifa 31015, Israel.*

2. Related Work

Brooks et al. introduced Dynamic Thermal Management (DTM) as a solution for microprocessor thermal problems [3]. Since then, several different DTM mechan-

isms were proposed, such as Stop&Go (global clock toggling) [3], local toggling [5], activity migration [4], thread migration, dynamic voltage/frequency scaling and multicore Temperature Aware Task Scheduling [11]. These mechanisms were shown to be applicable by Donald and Martonosi [17] to both single core systems and multicore systems.

In most of the above cases, the DTM design process includes using architectural and thermal simulators and 'trial and error' to fine-tune algorithm parameters. An alternative, more precise approach is to design the DTM by using optimization methods to compute the offline optimal algorithm that reaches maximal performance and meets thermal constraints. This approach was used on Dynamic Voltage Scaling (DVS) DTM [2] and on Dynamic Frequency (DFS) Scaling [15]. There, it is proposed to solve the DVS/DFS optimization problem using optimal control methods. In addition, an analysis of the optimal strategy of a simple single-block thermal model is given, but more complex scenarios remain unexplored. A similar approach to solve the multicore DFS offline optimization problem, which is based on convex optimization rather than optimal control, was shown in [20]. Another alternative approach to design DTM is the analytical approach that seeks to optimize online algorithm parameters and formulate closed analytical solution for the DTM optimization problem. This approach was used on single-core DFS in [16] and on multicore DFS in [19]. Mathematical optimization of single-core voltage scheduling was also used for purposes other than thermal controllers, such as energy minimization in [13] and [14].

Unlike previous work, our analysis of the optimal strategy relies on a multi-block layout rather then a single-block layout. This approach is more realistic and thermally accurate. Our optimization method relies on Integer Linear Programming as oppose to Convex Optimization or Optimal Control. Because our problem is discrete, the other optimization methods can not be applied. Finally, we also innovate by investigating the effects of optimal DTM on chip layout design.

3. MULTICORE TASK SCHEDULING UNDER THERMAL CONSTRAINTS

Multicore Task Scheduling is a DTM mechanism for multicore systems that schedules tasks to cores at given time intervals, according to the temperatures present in different regions of the chip [11]. Our goal is to compute the multicore task scheduling strategy that achieves maximal performance while meeting thermal constraints for a given workload. In order to compute an optimal offline strategy, we use Integer Linear Programming (ILP) optimization which requires the definition of a search space, a set of constraints and an objective function to maximize.

In this section, we present the model we used to formulate the search space, constraints and objective function. The model is approximate and is derived from analytic formulas and simulation results. The model is simple and can be further expanded.

3.1 Task Energy Model

We define multicore task scheduling as the problem of determining the assignments of unit-tasks to specific cores in specific times, where each unit-task is identical in duration and energy consumption to the others. This model is derived from the idea that longer tasks can be broken into several shorter unit-tasks. We used a simplified task energy model:

The tasks: Tasks can be broken into different amounts of unit-tasks. We assume the system contains:

$$\begin{aligned} &n_T \to \infty \text{ identical unit-tasks} \\ &\text{Each unit-task consumes } E \text{ [J]} \end{aligned} \quad (1)$$

The scheduler: Each core can handle only one unit-task at a time period. The assignments are done on a discrete time basis: each time period, assignments are being made. There are n_P time periods, each with Δt length. This is the *scheduling partition*:

$$t_0 = 0,\ t_1 = \Delta t,\ \ldots,\ t_{n_P} = n_P \cdot \Delta t \quad (2)$$

The multicore system: The system contains n_C cores that are able to complete one unit-task in one time period Δt.

Performance metrics: Since unit-tasks are identical in duration, we define the objective function of our optimization to be the amount of unit-tasks completed in time throughout the scheduling period $[0..t_{n_P}]$.

Thermal constraints: The system is able to successfully complete tasks when its maximum junction temperature is below $T_{\max}$. Thus, our constraints are that temperatures do not cross $T_{\max}$ in any scheduling time period $t_0, t_1, \ldots, t_{n_P}$. There is no need to calculate the chip temperatures within the time periods because, for a given task assignment strategy and a time interval, the temperature function in respect to time is monotone within each time interval.

3.2 Thermal Model

In order to express the thermal constraints as analytic formulas of the task scheduling strategy, we use the RC-network thermal model [5] [10]. Our thermal model captures the following thermal scene:

Heat is generated at the cores, in the junction area (the lowest vertical layer). Cooling is accomplished by isotropic 3D heat conduction towards a heat sink through a stack of thermal blocks. The thermal blocks represent the silicon chip, the Thermal Interface Material layer and a heat sink.

The RC-network thermal model requires the system to be divided into a number of thermal blocks where each block is assigned a temperature in each time period. As an example, several divisions of a 4 cores system to 9 thermal blocks are illustrated in Figure 6. The thermal model is based on the duality between heat-flow and electrical current.

In our thermal model, we use a 3-dimentional RC network and connect each thermal block through a capacitor C_i to the ground modeling its heat capacitance. We

only assign power sources to blocks of cores in order to model core activity. Between neighboring nodes, a thermal resistance R_{ij} is added to model the lateral and vertical heat conduction path. We connected all the blocks in the top layer to the heat sink node. An example of a thermal RC model is illustrated in Figure 1. A comprehensive analysis of RC thermal model is found in [5].

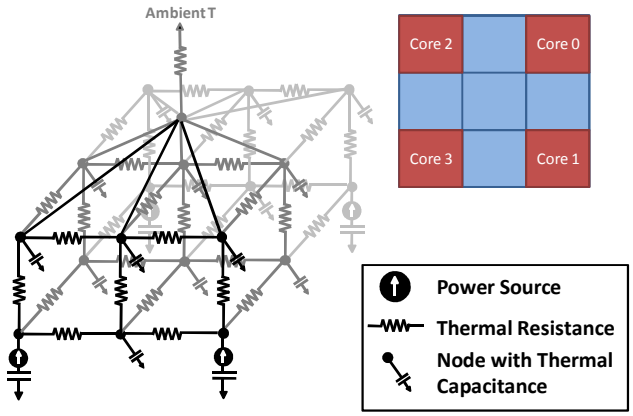

Figure 1. Thermal RC-network model for a 4-cores microprocessor. The cores are located in the corner nodes of the lower layer.

3.3 Calculating Chip Temperatures

By applying Ohm and Kirchoff laws to the network, we are able to express the chip temperatures as a function of the power assignments. The application of the laws yields a first order differential equation in the following form:

$$\dot{\vec{T}}(t) = Q \cdot \vec{T}(t) + S \cdot \vec{P}(t)$$

To prove this equation, we define *capacitance matrix* $C = diag(C_1, C_2, ..., C_{n_B})$ and *conductance matrix:*

$$G_{ij} = \begin{cases} \frac{1}{R_{ij}}, & \text{if } i \neq j \\ -\sum_{k=1}^{n_B} \frac{1}{R_{ik}}, & \text{if } i = j \end{cases}$$

Then the application of the Ohm and Kirchoff laws yields:

$$\dot{\vec{T}} = C^{-1} G \cdot \vec{T} + C^{-1} \cdot \vec{P}$$

This means that for our purposes, we define:

$$Q = C^{-1} G \text{ and } S = C^{-1} \quad (3)$$

And we finally get:

$$\dot{\vec{T}}(t) = Q \cdot \vec{T}(t) + S \cdot \vec{P}(t) \quad (4)$$

This differential equation will later help us to express the thermal constraints as analytical formulas of the task scheduling strategy.

4. Offline Optimization of Multicore Task Scheduling

In this section, we will use our model to formulate the optimization problem as an instance of Integer Linear Programming and then solve it using Integer Linear programming solvers.

4.1 Integer Linear Programming

Integer linear programming (ILP) [23] is a class of optimization problems with a linear objective function, subject to linear equality and inequality constraints where the optimization variable is integer. Any ILP problem can be described by:

$$\begin{aligned} & p^T \cdot x \rightarrow \max \\ & Ax \leq b \\ & x \in \mathbb{Z}^n,\ p \in R^n,\ b \in R^m,\ A \in M_{m \times n} \end{aligned} \quad (5)$$

Where $p^T x$ is the *objective function* which is linear in respect to the *optimization variable* x. The inequality $Ax \leq b$ defines a set of m linear constraints that specify a convex polyhedron search space where the solution is to be sought.

4.2 The Optimization Problem

Our offline optimization problem is to find a multicore task scheduling strategy $\hat{f}$ that maximizes performance for a given workload and complies with thermal constraints over time intervals $t_0, t_1, ..., t_{n_P}$. For a multicore system with n_C cores and n_B thermal blocks, and given $n_T \rightarrow \infty$ unit-tasks and a time partition made from n_P time periods, we present the optimization problem as follows:

Search space: The search space consists of all task scheduling strategies $\hat{f}$. $\hat{f}$ is a binary vector of size $n_C \cdot n_P$, where a '1' on the $i \cdot n_C + j$ element means that a unit-task is assigned at time period i, on core j and '0' means that no unit-task was assigned as illustrated in Figure 2.

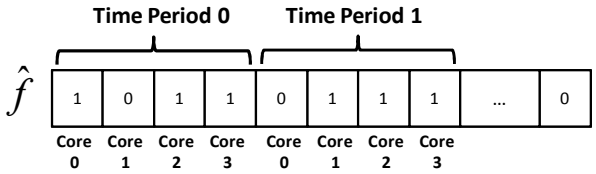

Figure 2. Binary strategy vector for the case of a system with 4 cores.

Objective function: The performance function $Perf(\hat{f})$ gives the number of unit-tasks completed in the system after n_P time periods.

$$Perf(\hat{f}) = \sum_{i=0}^{n_P \cdot n_C - 1} \hat{f}_i$$

Constraints: $Temp_{B_i, P_j}(\hat{f})$ is the temperature calculated for block i at time period j. The temperatures are to be kept below T_{Max} at all times.

It is now possible to define the above optimization problem in a more compact form:

$$\sum_{i=0}^{n_P \cdot n_C - 1} \hat{f}_i \rightarrow \max \quad (6)$$

s.t.

$$Temp_{B_i,P_j}\left(\hat{f}\right) \le T_{Max} \quad (7)$$

$$\hat{f} \in \{0,1\}^{N_P \cdot N_C} \quad (8)$$

4.3 Describing the Constraints in Linear Form

In order to complete the formulation of our optimization problem to ILP form, we must also formulate the constraints $Temp_{B_i,P_j}\left(\hat{f}\right) \le T_{Max}$ in a linear form such as: $A \cdot \hat{f} \le b$.

Previously in equations 3, 4 we showed that the RC-model gives us the following linear differential equation for the case of constant power in each interval:

$$\dot{\vec{T}}(t) = Q \cdot \vec{T}(t) + S \cdot \vec{P}(t)$$

Where:

$$Q = C^{-1}G \text{ and } S = C^{-1}$$

After solving the differential equation for a constant power $\vec{P}$ and start condition $\vec{T}(0) = \vec{T}_0$ we get:

$$\vec{T}(t) = \Gamma(t) \cdot \vec{T}_0 + \Phi(t) \cdot \vec{P}$$

Where:

$$\Gamma(t) = e^{Qt}$$

$$\Phi(t) = (e^{Qt} - \mathbf{I})Q^{-1}S$$

If we apply the constant power solution to each scheduling partition interval, we will get that for interval i:

$$\vec{T}(i) = \Gamma(t) \cdot \vec{T}(i-1) + \Phi(t) \cdot \vec{P}(i),\ i = 1 \ldots n_P$$

This is a recurrence relation that can be extracted to the following formula:

$$\vec{T}(i) = \Gamma^i(t) \cdot \vec{T}(0) + \sum_{k=1}^{i} \left(\Gamma^{i-k}(t) \cdot \Phi(t) \cdot \vec{P}(k)\right),\ i = 1 \ldots n_P$$

This formula expresses the temperatures of each block at each time as a linear combination of the initial temperatures and the power assignments at all times. This is exactly what we wanted.

To express the coefficients in a single matrix rather than the sum in the formula, we define recursively a sequence of matrices $\Lambda(i)$:

- $\Lambda(0)$ is a $n_C \times n_C n_P$ zero matrix.
- $\Lambda(i)$, $0 \le i \le n_P$, is $\Gamma \cdot \Lambda(i-1)$ with adding the n_C columns of matrix Φ to columns $(i-1) \cdot n_C + 1, \ldots, (i-1) \cdot n_C + n_C$.

Now we describe $\vec{T}(i)$ in this matrix form:

$$\vec{T}(i) = \Gamma^i \cdot \vec{T}(0) + \Lambda(i) \cdot \vec{P}$$

This is a linear connection between the temperatures in scheduling interval i and the power assignments vector $\vec{P}$. As said in section 3.1, within each interval of the scheduling partition, the task assignments are constant and according to the definition of the unit-task in equation 1, the power assignments are also constant in each interval and consume E Energy units. Thus the last equation is equivalent to:

$$\vec{T}(i) = \Gamma^i \cdot \vec{T}(0) + \Lambda(i) \cdot \hat{f} \cdot E$$

And if all blocks' temperatures are constrained to be below $\vec{T}_{\max}$, we formulate the linear constraints to be:

$$\vec{T}(i) = \Gamma^i \cdot \vec{T}(0) + \Lambda(i) \cdot \hat{f} \cdot E \le T_{\max}$$

or:

$$E \cdot \Lambda(i) \cdot \hat{f} \le T_{\max} - \Gamma^i \cdot \vec{T}(0)$$

And by defining $A = \begin{pmatrix} E \cdot \Lambda(1) \\ \ldots \\ E \cdot \Lambda(n_P) \end{pmatrix}$ and $b = \begin{pmatrix} \vec{T}_{\max} - \begin{pmatrix} \Gamma^1(1) \cdot \vec{T}(0) \\ \ldots \\ \Gamma^{n_P}(n_P) \cdot \vec{T}(0) \end{pmatrix} \end{pmatrix}$, we finally get: $A \cdot \hat{f} \le b$

By this, we have completed the formulation of our optimization problem as an instance of ILP as shown in equation 6:

$$\sum_{i=0}^{n_P \cdot n_C - 1} \hat{f}_i = [111 \ldots 1]^T \cdot \hat{f} \to \max \quad (9)$$

s.t.

$$A \cdot \hat{f} \le b \quad (10)$$

$$\hat{f} \in \{0,1\}^{N_P \cdot N_C} \quad (11)$$

4.4 Solving Integer Linear Programming Problems

There is no general efficient algorithm for solving ILP and it is an *NP-hard* problem. Nevertheless, there are several useful heuristics for solving ILP. It should be noted that all of these approaches are still of non-polynomial time complexity in the worst case, but they provide an efficient way to solve ILP in numerous cases.

In this research we used the *Branch and Bound* algorithm [24] as implemented in the commercial Tomlab Optimization Toolbox for Matlab v.7.1 [25]. This approach was successful in solving our ILP problems.

5. DTM Optimization Results and Analysis

In this section we compute and analyze the optimal DTM strategies using ILP. By analyzing the performance and behavior of the optimal strategies we not only calculate the upper bound on performance for the DTM, but also specify the desired behavior of the online DTM. In addition, by comparing the performance of optimal strategies in several different chip layouts, we conclude which layouts are more thermally efficient.

5.1 Task scheduling in a single-core microprocessor

To exemplify the concept of offline DTM optimization, we first introduce a simple scenario. In this DTM scenario, described in [3] the DTM controller is able to stall all computation (e.g. by global clock gating) in the

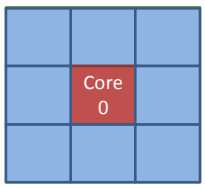

Figure 3. The thermal model for the single-core Stop&Go scenario. The junction is divided into 3x3 blocks. The core resides in the central block and it is the only active block in the thermal system.

microprocessor in order to let it cool. Essentially, single-core Stop & Go is the single-core version of task scheduling DTM. The layout of the system is shown in Figure 3

We use a simplified thermal model where we divide the junction area to a 3x3 grid of thermal blocks. The core resides in the active central thermal block while all the other blocks are non-active and model the much less power intensive cache and other subsystems.

The optimal strategy and the resulting temperature evolution are depicted in Figure 4. Observing the optimal solution over time, we divide it into two distinct parts. In the first part no stalling occurs for numerous time intervals (a task is assigned to core 0 in each scheduling time interval) and the temperatures of the chip rise steadily, reaching close to $T_{\max}$. In the second part, stalling occurs periodically (there are intervals where no task is assigned to the core) and the temperatures oscillate closely to but never crossing $T_{\max}$. In Figure 4, the first part takes place in time intervals 1-3 and the second part takes place in intervals 4-40. The optimal solution tries to keep the temperatures as high as possible without crossing $T_{\max}$. The optimal solution does not succeed in keeping the temperatures steady at $T_{\max}$.

We can further formulate this optimal behavior as an online strategy: Define $T_{control}$ as the minimum chip temperature that when present in the system, loading the processor for one more scheduling interval will lead to crossing $T_{\max}$ in the next interval. We see that the optimal strategy follows the following intuitive rule: "Keep the microprocessor running until it reaches $T_{control}$, than halt the microprocessor until it cools down to below $T_{control}$ and continue running again". $T_{control}$ is easily computable and should be computed to match the thermal model and the task energy model of the optimized system when designing the online Stop&Go DTM.

5.2 Multicore task scheduling in a 4-cores microprocessor

In this DTM scenario, the controller may schedule tasks or stall scheduling for each core separately in every scheduling time interval. We present the optimal strategy for multicore task scheduling in three different 4-cores layouts illustrated in Figure 6.

The optimal strategy and the resulting temperature evolution for Mid-Side layout are depicted in Figure 7 Observing the optimal solution over time, we divide it to two distinct parts. In the first part no stalling occurs for numerous time intervals (tasks are assigned to all cores in each scheduling time interval) and the temperatures of the chip rise steadily, reaching close to $T_{\max}$. In the second part, a periodic task scheduling pattern emerges in the cores. In this part, the temperatures oscillate closely to but never crossing $T_{\max}$. In Figure 7, the first part takes place in time intervals 1-3 and the second takes place in time intervals 4-20.

Several observations can be made:

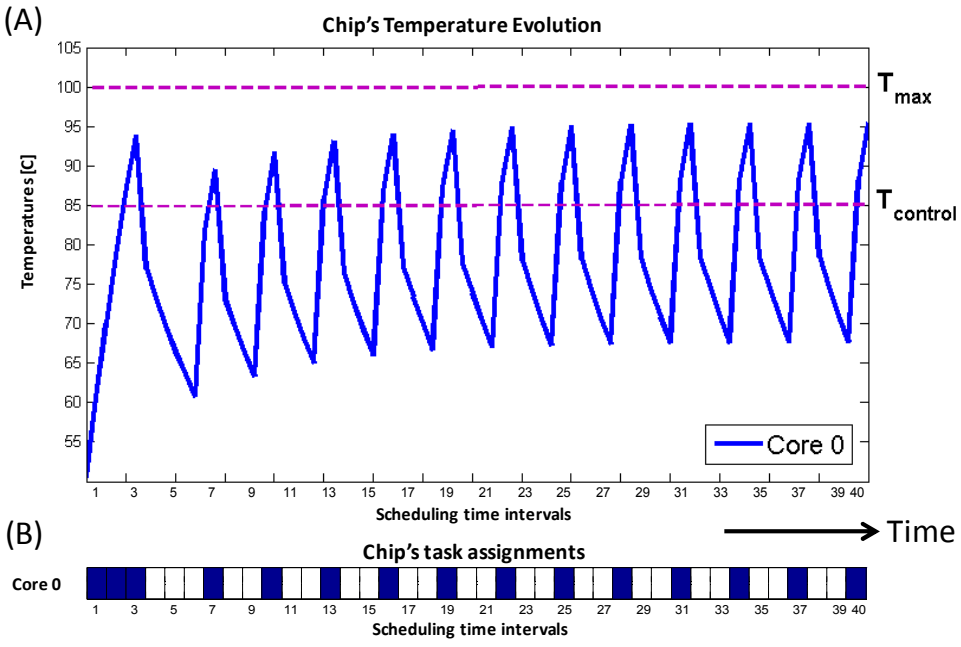

Figure 4. The optimal single-core Stop&Go strategy and temperatures evolution over time. Graph A shows the temperature curve of the core block. Graph B shows the core activation times

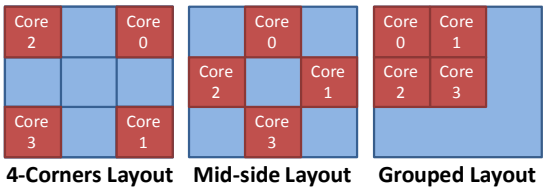

Figure 6. The thermal models for the 4-cores task scheduling scenario. The junction is divided into 3x3 blocks. The cores' corresponding thermal blocks are the only power-active blocks in the model.

- It is natural for distant cores to work in parallel because it is more thermally efficient than having close cores to work in parallel. In our case, cores 0 and 3 are distant across the chip (see Figure 6) and therefore form an *assignment group*. The second assignment group contains cores 1 and 2.
- Switching periodically between assignment groups lowers the overall temperature of the chip. A scheduling strategy should use all the cores available.
- There can arise a thermal opportunity to break the pattern and assign more tasks (see schedule intervals 5, 6). These opportunities can be identified by carefully inspecting the temperature curves.
- The optimal solution does not succeed in keeping the temperatures steady at $T_{\max}$.

When designing the online task scheduling DTM, one should consider the optimal assignment groups and the optimal periodic patterns that match the thermal

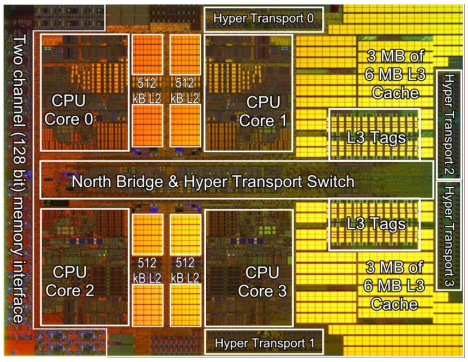

Figure 5. Chip layout of AMD Quad Core Shanghai. The 4 cores are positioned close together. [22]

model and the task energy model of the optimized system. We have also calculated the optimal strategy for 4-Corners layout and Grouped layout and found that the above analysis applies.

The performance results of the optimal strategies give the upper bound on DTM performance. When comparing the performance of the optimal solution in each of the above layouts, we discover that Mid-Side layout gives the best results, 4-Corners layout comes second, and Grouped layout gives the worst results (see Table 1). This means that in terms of upper bound on DTM performance, the Mid-Side layout is more thermally efficient than the other layouts. This result is consistent with the fact that in Mid-Side layout, the heat from the cores can be conducted through 3 surrounding thermal blocks, as opposed to just 2 in the case of 4-Corners layout. In the

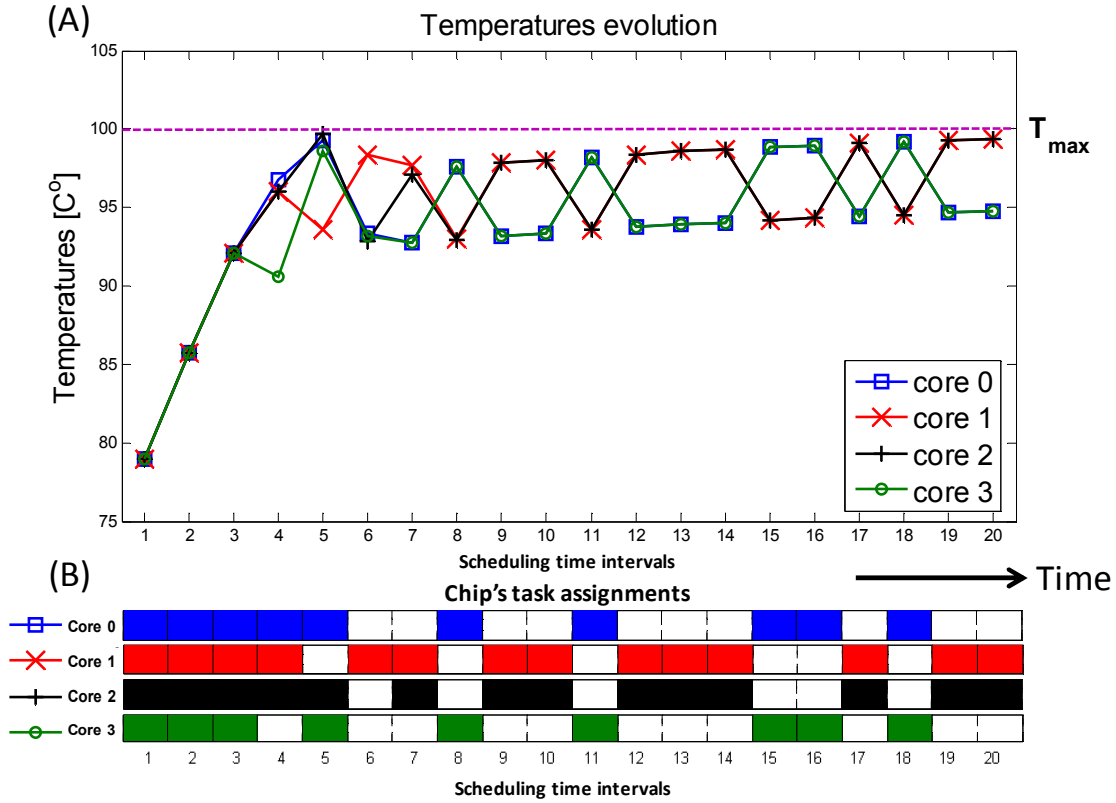

Figure 7. Mid-Side results. The optimal 4-cores task scheduling strategy and corresponding temperatures evolution for Mid-Side layout. Graph A - temperature curves of the cores' thermal blocks, Graph B- optimal task assignments for each core (maximizes the total amount of completed tasks).

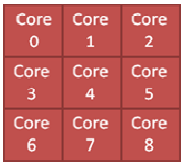

Figure 8. The thermal model of the 9-cores task scheduling scenario. The junction area is divided into 3x3 blocks. Each core resides in an active block.

case of Grouped layout, the cores are positioned close together to form a typical hot spot and thus give the poorer thermal results. Note that many current multiprocessors' floor plans (see Figure 5) are similar to Grouped layout in the fact that the cores are positioned close together, forming a natural hot spot (see Figure 5). This further stresses the fact that the performance of a system is heavily dependent on its geometrical layout because of thermal considerations.

5.3 Multicore task scheduling in a 9-cores microprocessor

The layout of the system is illustrated in Figure 9, the optimal strategy and the resulting temperature evolution are depicted in Figure 9. As in previous scenarios, the optimal solution can be divided over time to two distinct parts. The first "heating up" part takes place in scheduling time intervals 1-4 and the second "periodic scheduling patterns" part takes place in scheduling time intervals 5-20.

The conclusions of the 4-cores scenario also apply here with the following differences:

- In this case, the 9 cores divide into two assignment groups that work in parallel: the even-numbered cores and the odd numbered cores.
- The thermal opportunity to break the assignment pattern is depicted in schedule interval 17.
- The optimal solution does not succeed in keeping the temperatures steady at T_{max}. However, the solution manages to oscillate closer to T_{max} in respect to the 4-cores scenario. This implies that the 9-cores task scheduling DTM is more performance efficient than the 4-cores task scheduling DTM.

5.4 DVS/DFS-based power assignment mechanism results

In this DTM scenario, the controller is able to change the voltage or the operating frequency of the microprocessor in order to change its power consumption and thus cool it down. In this analysis, we assume that the controller is able to use the voltage and frequency scaling in order to control the exact power consumption of the chip over time (e.g. by accurately predicting the capacitance of the tasks). Thus, in every scheduling time interval, the controller is able to assign exact power consumptions to each core. Essentially, this DTM scenario is the "frictional task" version of multicore task scheduling, where a friction of the task's energy can be assigned to cores. The fact that the power assignments are frictional

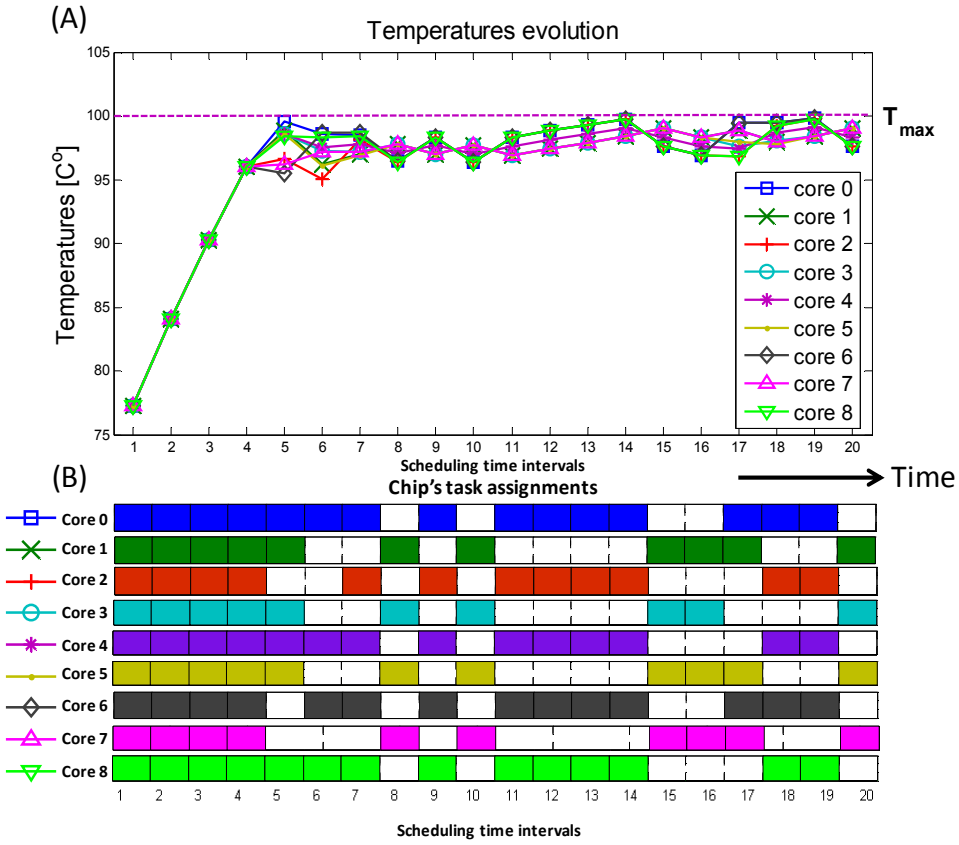

Figure 9. 9-cores results. The optimal 9-cores task scheduling strategy in terms of the amount of completed tasks and corresponding temperatures evolution. Graph A - temperature curves of the cores' thermal blocks, Graph B- task assignments times for each core.

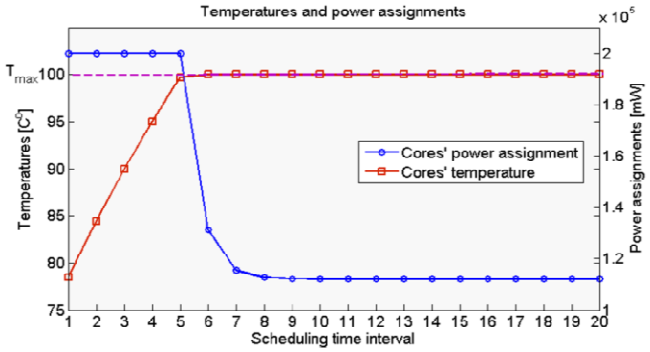

Figure 10. The optimal 9-cores DVS/DFS based assignment strategy and resulting temperatures evolution.

also means that the optimization variable is not obliged to be integer so the corresponding optimization problem is Linear Programming rather than Integer linear programming.

We use the 9-cores layout depicted in Figure 9 and compute the optimal strategy and the resulting temperature evolution as depicted in Figure 10. Examining the results, we see that optimal strategy assigns the same amount of power to all the cores. This can be explained by the fact that each individual core is heated by its neighboring active cores whether the core is active or not so it's best for it to just as well be active under the power consumption that correspond to its neighboring temperatures.

We divide the optimal strategy into three distinct parts. First, maximal power assignments are made to all the cores and the temperatures of the chip rise steadily, reaching close to $T_{\max}$. Secondly, the power assignments magnitude decline exponentially until they reach a certain constant assignment and the temperatures reduce their rising rate and settle exactly on $T_{\max}$. In the third part, the power assignments and the resulting temperatures remain constant. The first part takes place in Figure 10in scheduling time intervals 1-4, while the second part takes place in intervals 5-10 and the third part takes place in time intervals 11-20. This optimal behavior is consistent with the analytical analysis done in [2].

Several observations can be made:

- All the cores are assigned the same power assignments in all times. All cores share the same temperature.
- As seen in previous scenarios, the optimal solution seeks to reach and stay at $T_{\max}$.
- In contrast to previous DTM mechanisms, here the optimal solution succeeds in keeping the temperatures steady at $T_{\max}$. The processor uses the entire thermal potential of the system and this will have the best performance among the DTMs we've observed.

5.5 Performance Comparison

We computed the performance of the above systems in terms of unit-tasks per interval. To make the comparison valid, we normalized the unit-task according to the number of cores n_C in each scenario in the following way: Each time a core is activated, it completes n_C^{-1} unit-task and consumes E^{-1} energy units. The maximum performance measurement of any system is 1. The results are shown in Table 1.

Table 1. DTM performance results

Task Scheduling Scenario	Normalized Performance	DVS Scenario	Normalized Performance
Single Core TS	0.5	Single core DVS	0.62
4-cores TS, 4-Corners layout	0.63	4-cores DVS, 4-Corners layout	0.68
4-cores TS, Mid-Side layout	0.65	4-cores DVS, Mid-Side layout	0.68
4-cores TS, Grouped layout	0.61	4-cores DVS, Grouped layout	0.66
9-cores TS	0.68	9-cores DVS	0.70

The results indicate that the potential performance of task scheduling and DVS rises with the number of cores in the system. In addition, the performance of the optimal continuous DVS-based mechanisms is superior to the performance of optimal task scheduling mechanisms. As said previously, the most thermally efficient 4-cores layout is Mid-Side layout and the worst is Grouped layout. Note that many current multiprocessors' floor plans are similar to Grouped layout [10] in the fact that the cores are positioned close together, forming a natural hot spot (see Figure 5).

5.6 The Integrality Gap

We have seen that the best performing optimal DTM mechanism is the DVS/DFS-based power assignment mechanism. Thus, it is interesting to calculate the performance gap between a given optimal multicore task scheduling mechanism and the DVS/DFS-based power assignment mechanism. This gap, in percentage, is called the *Integrality Gap* and essentially expresses how much performance a multicore task scheduling mechanism loses because it is discrete rather than continuous in nature.

We compute the integrality gap in several multicore layouts and core power consumption values in order to test which core consumption is thermally more efficient in terms of optimal DTM solutions. Figure 11 illustrates the results.

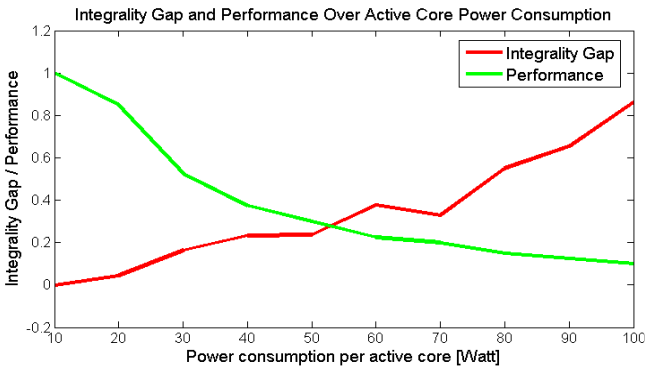

Figure 11. The integrality gap and performance over core power consumption.

Our results show that systems with less power consumption per core (e.g. with many small cores) have lower integrality gaps and higher optimal strategy performance. These low integrality gaps indicates that the optimal task scheduling strategy in these layouts achieves performance that is closer to the optimal DVS/DFS-based power assignment mechanism. This result is also consistent with the fact that a multicore task scheduling mechanism with many small cores has more granularities in its control and is thus more flexible in nature. This conclusion is consistent with the work of Huang et al. in [18].

6. Summary and Conclusions

In this paper, we propose a new way to design and quantitatively analyze DTM strategies. First, we use Integer Linear Programming to compute offline optimal DTM strategies that achieve maximal performance and meet thermal constraints for a given workload. We then analyze the behaviors of the optimal offline strategies and find specific strategy patterns that are desirable for DTM behavior. Finally, we use these patterns to formulize the behavior of the online DTM mechanism to be designed.

We apply offline Optimal DTM analysis on the case of multicore task scheduling, Stop&Go and DVS/DFS-based power assignment DTMs. Applying the analysis, we formulate the desired behavior of online task scheduling mechanisms and compute upper bounds on performance for it. We also devise a method to compare and identify the thermal limitations of microprocessor layout designs. We find that the multicore layout has a crucial effect on the thermal limitation of the microprocessor, and by comparing several multicore layouts we are able to identify layout design guidelines in the case of multicore task scheduling DTM.

We identify that optimal DTMs try to reach and keep chip's temperatures at T_{Max}. We also identify that in multicore systems, the optimal strategy uses *assignment groups* to have distant cores work in parallel and that pseudo-periodic multicore scheduling patterns occur. We identify that direct DVS/DFS-based power assignment mechanism DTM is the most efficient DTM and that it controls all of its cores identically. Using the definition of the integrality gap, we also indentify that layouts with many smaller cores are thermally more efficient than a layout with few larger cores.

The application of our methodology improves the design process of DTMs and multicore layout designs. Further research directions include improving the accuracy of the thermal model, computing optimal *online* DTMs and computing thermally optimal layouts.

Acknowledgments

The authors would like to thank Zvika Guz and Ronny Ronen for their kind support and information sharing.

References

[1] S. Borkar, "Design challenges of technology scaling," in IEEE Micro, vol. 19, no. 4, Jul.–Aug. 1999, pp. 23–29.

[2] A. Cohen, L.Finkelstein, R.Ronen, A.Mendelson, "On Estimating Optimal Performance of CPU Dynamic Thermal Management", in Computer Architecture Letters, October 2003.

[3] D. Brooks and M.Martonosi, "Dynamic thermal management for high-performance microprocessors", In Proceedings of the Seventh International Symposium on High-Performance Computer Architecture, pages 171–82, Jan. 2001

[4] S. Heo et al., "Reducing Power Density Through Activity Migration", In Intl. Symp. on Low Power Elec. Design, 2003

[5] K. Skadron, M. R. Stan,W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture", in Proc. ISCA-30, June 2003

[6] J.H. Anderson, J. M. Calandrino, and U. C. Devi, "Real-Time Scheduling on multiprocessor platform," in RTAS, pp. 199-207, Apr 2006

[7] S. Baruah, N. Cohen, C.G. Plaxton, and D. Varvel, "Proportionate progress: A notion of fairness in resource allocation", in Algorithmica, 15:600–625, 1996

[8] Y. Han, I. Koren, and C. A. Moritz, "Temperature aware floorplanning," in Second Workshop on Temperature-Aware Computer Systems(TACS-2), held in conjunction with ISCA-32, June 2005.

[9] E. Kursun, C. Y. Cher, A. Buyuktosunoglu, and P. Bose, "Investigating the Effects of Task Scheduling on Thermal Behavior", In Third Workshop on Temperature-Aware Computer Systems (TACS'06), June 2006

[10] M. Pedram, S. Nazarian, "Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods", in Proceedings of the IEEE, Vol. 94, No. 8, pp. 1487-1501, 2006.

[11] A. K. Coskun, T. S. Rosing, and K. Whisnant. "Temperature aware task schedulingin MPSoCs", In Proc. DATE, pp. 1659–1664, 2007.

[12] A. Manzak and C. Chakrabarti. "Variable voltage task scheduling algorithms for minimizing energy," in Proc. Int. Symp. Low Power Electronics and Design, Aug. 2001.

[13] F. Yao, A. Demers and S. Shenker, "A scheduling model for reduced CPU energy", in IEEE Annual Foundations of Computer Science, pp. 374-82, 1995.

[14] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling for minimizing energy", in Proc. Int. Symp. Low Power Design, 2001, pp. 279-282.

[15] Ravishankar Rao and Sarma Vrudhula, "Performance Optimal Processor Throttling Under Thermal Constraints", in CASES '07

[16] R. Rao, S. Vrudhula, C. Chakrabarti and N. Chang, "An optimal analytical solution for processor speed control with thermal constraints", in Proc. Intl' Symp. Low Power Electronics and Design (ISLPED), October 2006, pp. 292–297.

[17] J. Donald and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration", in Proceedings of the 33th International Symposium on Computer Architecture (ISCA-33), 2006.

[18] Wei Huang, Mircea R. Stan, Karthik Sankaranarayanan, Robert J. Ribando, Kevin Skadron, "Many-core design from a thermal perspective", in DAC 2008: 746-749.

[19] Ravishankar Rao and Sarma Vrudhula, "Efficient online computation of core speeds to maximize the throughput of thermally constrained multi-core processors", In Proceedings of the

IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2008.

[20] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. De Micheli, "Temperature-aware processor frequency assignment for MPSoCs using convex Optimization", in Proc. of CODES/ISSS '07.

[21] A. Leon, L. Jinuk, K. Tam, W. Bryg, F. Schumacher, P. Kongetira, D. Weisner, and A. Strong, "A power-efficient high-throughput 32-thread SPARC processor", in ISSCC 200.

[22] www.chip-architect.com, Rev.2, March 17, 2008.

[23] Schrijver, A. "Theory of Linear and Integer Programming", Wiley-Interscience, New York, 1986.

[24] Wolsey, Laurence A., "Integer Programming", John Wiley & Sons, 1998.

[25] TOMLAB optimization environment for Matlab. http://tomopt.com.