# Performance Optimal Online DVFS and Task Migration Techniques for Thermally Constrained Multi-core Processors

Vinay Hanumaiah, *Student Member, IEEE,* Sarma Vrudhula, *Senior Member, IEEE,* and Karam S. Chatha, *Member, IEEE*

***Abstract*—Extracting high performance from multi-core processors requires increased use of thermal management techniques. In contrast to offline thermal management techniques, online techniques are capable of sensing changes in the workload distribution and setting the processor controls accordingly. Hence online solutions are more accurate and are able to extract higher performance than the offline techniques. This paper presents performance optimal online thermal management techniques for multicore processors. The techniques include dynamic voltage and frequency scaling and task-to-core allocation or task migration. The problem formulation includes accurate power and thermal models as well as leakage dependence on temperature. The paper provides a theoretical basis for deriving the optimal policies and computationally efficient implementations. The effectiveness of our DVFS and task-to-core allocation techniques are demonstrated by numerical simulations. The proposed task-to-core allocation method showed a 20.2% improvement in performance over a power-based thread migration approach. The techniques have been incorporated in a thermal-aware architectural-level simulator called MAGMA that allows for design space exploration, offline and online dynamic thermal management. The simulator is capable of handling simulations of hundreds of cores within reasonable time.**

***Index Terms*—Multi-core, online thermal management, makespan minimization, performance optimization, task migration, convex optimization, dynamic voltage and frequency scaling, optimal control, leakage dependence on temperature.**

## I. Introduction

The transition to multi-core processors enabled the microelectronics industry to circumvent the *power wall* - the inability to improve performance of single core processors by simply relying on miniaturization and increasing clock speed, because of the resulting unsustainable growth in power consumption. Increasing the number of cores has become the new *scaling strategy*, with the expectation that processors with hundreds of cores will be possible in the not so distant future [1].

V. Hanumaiah is with the Department of Electrical Engineering, Arizona State University, Tempe, AZ, 85281, USA (e-mail: vinayh@asu.edu).

S. Vrudhula and K. S. Chatha are with the Faculty of Computer Science Engineering, Arizona State University, Tempe, AZ, 85281, USA (e-mail: vrudhula@asu.edu; kchatha@asu.edu).

Manuscript received Nov. 30, 2010; revised April 19, 2011. Current version published MMM, DD, 2011. This work was supported in part by NSF grant CSR-EHS 0509540, by Consortium for Embedded Systems grant DWS-0086, by the Science Foundation Arizona (SFAz) grant SRG 0211-07 and by the Stardust Foundation.



In comparison with single cores, many-core processors will once again exhibit increased power consumption as well as greater temporal and spatial variation in power consumption among the cores. This is due to variations in the number of threads that can be deployed at any time, and the increased intra-core and inter-core process variations due to miniaturization. Designing a package of a future many-core processor to dissipate the maximum possible power consumption will not only be uneconomical but may not be feasible. A more practical solution will be to design the package that is capable of dissipating the average power, and rely on dynamic thermal management (DTM) techniques (dynamic voltage and speed control or DVFS, and task migration) to ensure that the thermal constraints will not be violated when the power dissipation exceeds the capability of the package. Therefore, with many-core processors, DTM techniques are expected to be activated more often (i.e. not just at thermal emergencies) than in single core processors, and consequently, will have a greater negative impact on the performance.

DTM techniques typically control the power and the thermal behavior of a processor by varying three main controls; core speed, core voltage and allocation of tasks to cores. Speed control is the easiest among them and incurs the least performance penalty but offers less power reduction, whereas voltage and task migration offer the greatest power reduction, but suffer from higher performance penalties.

### A. Summary of Contributions

In this paper, we present online control algorithms for optimizing the performance of a heterogeneous multi-core processor subject to thermal constraints, when executing a set of tasks. By heterogeneous, we mean that each core can be operated at independent speed and voltage, and the power-thermal characteristics of cores can vary. The objective is to minimize the latest completion times of all tasks (*makespan*). The decision variables are the task-to-core mapping, and the individual core speeds and voltages. The relation between temperature of each core and its power dissipation is expressed through a set of linear differential equations.

The above optimization is solved in two steps to reflect the fact that the dynamic control takes place on two different time scales (see Fig. 1). The allocation of tasks are determined at the beginning of the migration interval which is typically between 50 ms to 100 ms (much higher than the die thermal

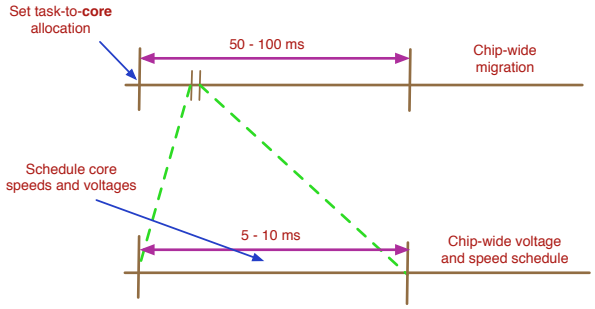


Fig. 1. Time scales for optimization

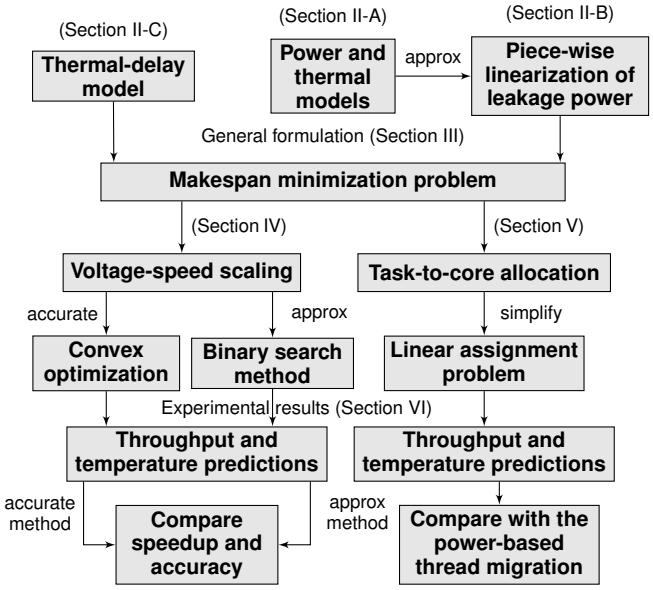


Fig. 2. Key contributions and paper outline.

time constant $\tau_{die}$ – 10 ms). Once the tasks are assigned to the cores at the beginning of the migration interval, then every 5 to 10 ms ($\approx \tau_{die}$) within the migration interval, the core speeds and voltages are adjusted. The core speeds and voltages are held fixed within a scheduling interval.

The problem of optimal *thermal aware* task-to-core allocation turns out to be a computationally difficult non-linear optimization problem. We present a simplification that reduces the problem to a linear assignment problem, which is polynomially solvable, and shows that the results yield significant improvements over power based thread migration method [2].

We show that the thermally constrained DVFS policy (also called the *voltage-frequency* or the *voltage-speed* policy) over a given interval that minimizes the makespan of a set of tasks is equivalent to maximizing their *instantaneous* throughput. *Throughput* is defined as the aggregate of speeds of all cores over a duration of time. Further, this is achieved through the *zero-slack* policy, which either sets the speed of a core to its maximum value if the temperature of its hottest block is below the temperature upper bound $T_{max}$, or sets the speed to a value that keeps the temperature of the core at $T_{max}$.

The *zero-slack* policy is just that - a policy. It expresses the form of the globally optimal speed function. We first show that implementation of that policy requires repeatedly solving a convex optimization problem over short intervals. Next we exploit the structure of certain matrices that appear in constraints of the convex optimization formulation to develop a fast computational procedure which when compared to the convex optimization approach is **20** times faster, and with about **0.4**% error in the predicted throughput.

The proposed solutions on the task-to-core allocation and thermally constrained DVFS are demonstrated by numerical results for a many-core processor. The proposed method for the task-to-core allocation achieves **20.2**% improvement over corresponding power based thread migration scheme [2]. Finally, the potential of real-time executions of both online DVFS and task-to-core allocation are also demonstrated through experiments.

Fig. 2 shows the organization of the paper.

### B. Related Work

Table I summarizes some of the related work in the field of processor level DTM. Until recently, the focus of the research on DTM has been on single-core processors [3]–[6]. Multi-core processors require speed and voltage control of each core, and assignment of tasks to the cores under thermal constraints. This makes the process of determining the optimal configurations significantly more challenging. Some of the existing works [7], [8] address the power management for multi-cores. Although power is a major limiting factor in the design of processors, especially portable devices, with the shrinkage of devices, thermal related issues are gaining increasing consideration. In general, power reduction techniques are not necessarily optimal under thermal constraints as power constraints are of aggregate nature, while thermal constraints are required to be satisfied at every instant of time.

Owing to the difficulty of incorporating time-dependent thermal constraints, much of the existing work on DTM have resorted to the use of simpler thermal models [3], [4], [9]–[11]. Some of these approximations include: (**i**) the use of simple lumped thermal RC model [3], [4], [12], which ignores the spatial thermal distribution and ignores the differences between the die and the package thermal time constants; (**ii**) neglecting the effect of leakage dependence on temperature [3], [4], [9] (at high temperatures, leakage power can increase power consumption by ten-fold); (**iii**) undermining the importance of voltage scaling [9], [10], [13] (DVFS provides cubic power reduction). These assumptions may severely underestimate the throughput of processors.

This work advances the existing work in DTM of multicore processors in three ways: (**1**) it gives a precise formulation of the problem of optimal speed and voltage control with task-to-core allocation, that includes accurate power and thermal models; (**2**) it presents an optimal solution in the form of an optimal *policy*; and (**3**) it introduces approximations that are based on minimal realistic simplifications, which lead to efficient computational procedures for solving the DTM online.

TABLE I
OVERVIEW OF PREVIOUS WORK IN THE AREA OF DYNAMIC THERMAL MANAGEMENT FOR PROCESSORS.

| Subject | References and description |
|---|---|
| Thermal modeling | HotSpot [14], ATMI [15] |
| Power modeling | Wattch [16], PTScalar [17] |
| Processor DVFS control techniques | DFS: [9], [13], [18] (MC), [3] (SC)<br>DVFS: [19] (power), [11], [20] (thermal) |
| Thread migration | Thermal-aware [21]–[25] |
| Task sequencing | [26]–[28] |
| Analytical approaches | Thermal: [4] (SC) , [10] (MC)<br>Power: [7], [8] (MC) |
| Closed loop implementations | Thermal: [5] (SC) [29] (MC)<br>Power: [7], [8] (MC) |
| Comparison of DTM techniques | [6] (SC), [30] (MC)<br>Thread migration heuristics [2], [12], [31] |
| Survey/overview | [1], [32] |

* SC = single-core, MC = multi-core

## II. MODEL DESCRIPTION

### A. Power and Thermal Models

The task model consists of $q$ tasks which are scheduled to run on an $n$ core processor. Each core is capable of executing a single task and each task is assumed to execute independently of other tasks, which means that there is no inter-task communication. The speed and voltage of a core $c$ are denoted by $s_c$ and $v_c$, and are assumed to be continuous functions of time normalized over [0,1]. A task $j$ consists of $I_j$ number of instructions and its instruction per cycle is denoted by $IPC_j$.

The thermal model used in this work is based on HotSpot-4 [33]. Using an analogy from electrical networks, HotSpot-4 represents the thermal characteristics and interactions between various thermal blocks as an RC network, with power inputs modeled as current sources, while heat spreading and storing capacities modeled through resistors and capacitances respectively. Fig. 3 shows the HotSpot thermal model for a typical four core processor. Each core is divided into $m$ thermal blocks on the die and the thermal interface material (TIM) layers. The package, which includes the heat spreader and the heat sink, is modeled with 5 and 9 thermal blocks, respectively. Together, the total number of thermal blocks is $N = 2nm + 14$.

The thermal model can be expressed using state-space models [34] as:

$$\frac{d\mathbf{T}(t)}{dt} = \mathbf{AT}(t) + \mathbf{BP}(\mathbf{s}, \mathbf{v}, \mathbf{T}, t), \tag{1}$$

where $\mathbf{T}$ and $\mathbf{P}$ are temperature and power vectors[1] of dimension $N \times 1$ respectively, $N$ being the total number of thermal blocks. Since only the die units of chip generates heat, only the first $nm$ units of $\mathbf{P}$ are non-zero. The dimension of $\mathbf{s}$ and $\mathbf{v}$ are $n \times 1$, where $n$ is the number of cores. $\mathbf{A}$ and $\mathbf{B}$ are constant matrices of size $N \times N$ and this makes the thermal system a time-invariant system. Note the **cyclical dependency** between $\mathbf{P}$ and $\mathbf{T}$ in (1).

$\mathbf{P}$ represents the total power, which is sum of the dynamic power $\mathbf{P}_d$ and the leakage power $\mathbf{P}_l$. The dynamic power varies linearly with the speed and quadratically with the voltage. The components of the dynamic power vector are expressed as

$$P_{d,c,b}(t) = P^{max}_{d,c,b}(t) s_c(t) v_c^2(t), \tag{2}$$

where $P^{max}_{d,c,b}$ is the dynamic power dissipated by block $b$ of core $c$ when the core is at the maximum speed and voltage. $P^{max}_{d,c,b}$ is obtained by profiling the time-varying power consumption of the task to be run on core $c$.

The leakage power is given by the following empirical model [17].

$$P_{l,c,b}(t) = k_1 v_c(t) T^2_{c,b}(t) e^{\frac{\alpha v_c(t) + \beta}{T_{c,b}(t)}} + k_2 e^{(\gamma v_c(t) + \delta)}. \tag{3}$$

$k_1, k_2, \alpha, \beta, \gamma$ and $\delta$ are parameters that depend on circuit topology, size, technology and design. The non-linear leakage power dependence on temperature and voltage (LDTV), as well as the cyclic dependency between the leakage power and the temperature, complicate the analysis and the solution to various optimization problems. Without any further simplification, one can only resort to numerical solutions for general non-linear optimization problems. To make any further progress and to develop computationally efficient solutions, this relation needs to be approximated by linear models.

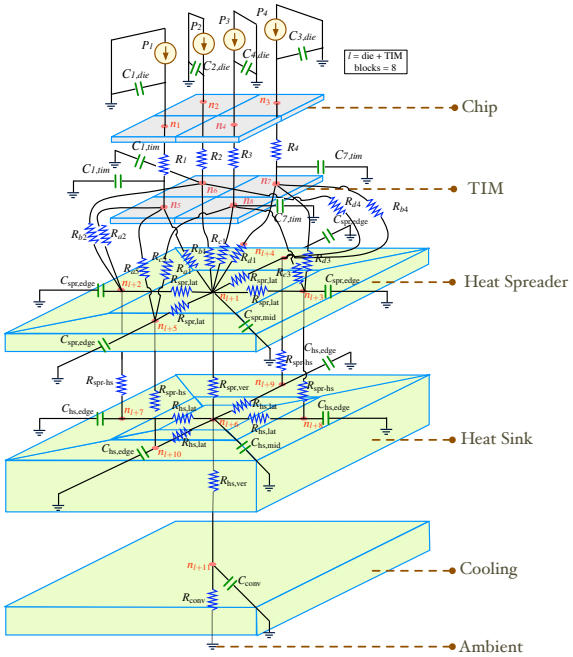


Fig. 3. HotSpot-4 thermal model for a four core processor.

### B. Piece-wise Linear Approximation (PWL) to LDTV

The leakage power described in (3) can be represented as a three dimensional surface with voltage and temperature axis as shown in Fig. 4. This surface can be linearized w.r.t temperature and voltage to any desired accuracy. The leakage power for an approximated linear section is expressed by the following equation:

$$P_{l,c,b}(t) = P_{l0,c,b} + g_{T,c,b} T_{c,b}(t) + p_{v,c} v_c(t), \ \forall c, b, t. \tag{4}$$

[1] All vectors are considered as column vectors, unless mentioned explicitly. Matrices and vectors are represented in bold.

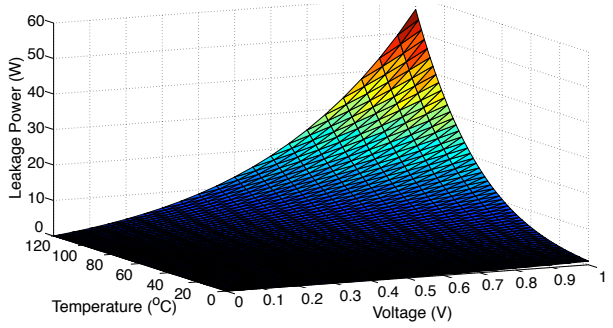


Fig. 4. Piecewise-linear approximation of leakage power

$g_{T,c,b}$ and $p_{v,c}$ represent the temperature and the voltage coefficients. $P_{l0,c,b}$ represents the leakage power for block $b$ in core $c$ corresponding to $T_{c,b} = 0$ and $v_c = 0$. Note that $T_{c,b} = 0$ and $v_c = 0$ refer to the ambient temperature and the minimum voltage.

Let $\mathbf{Y}$ be defined as

$$Y(i,j) = \begin{cases} 1, & \text{if block } i \text{ belongs to core } j \text{ and } i \leq nm \\ & \text{(total no. of blocks in the die)}, \\ 0, & \text{otherwise.} \end{cases}$$

Then the dynamic and the leakage power vectors are given by

$$\mathbf{P}_d(\mathbf{s},\mathbf{v},t) = diag(\mathbf{P}_d^{max}(t))\mathbf{Y}diag(\mathbf{v}(t))^2\mathbf{s}(t), \quad (5)$$
$$\mathbf{P}_l(\mathbf{v},\mathbf{T},t) = \mathbf{P}_{l0} + \mathbf{G}_T\mathbf{T}(t) + \mathbf{P}_v(\mathbf{v},t), \quad (6)$$

where $\mathbf{P}_v(\mathbf{v},t) = diag(\mathbf{P}_v)\mathbf{Y}\mathbf{v}(t)$. $\mathbf{G}_T$ and $\mathbf{P}_v$ are vectors of $g_{T,c,b}$ and $p_{v,c}$ respectively. $diag$ creates a diagonal matrix of a vector.

Substituting (5) and (6) in (1),

$$\frac{d\mathbf{T}(t)}{dt} = \hat{\mathbf{A}}\mathbf{T}(t) + \mathbf{B}[\mathbf{P}_d(\mathbf{s},\mathbf{v},t) + \mathbf{P}_v(\mathbf{v},t) + \mathbf{P}_{l0}], \quad (7)$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{B}\mathbf{G}_T$. Thus we removed the cyclical dependency between the temperature and the power in (1) with the use of PWL.

### C. Thermal-delay Models

It is well known that temperature, supply and threshold voltages affect the delay of a circuit. The following model [17], incorporates the above factors to express the maximum frequency of operation $s_c^{max}$ as a function of the core voltage and its temperature for 65 nm technology.

$$s_c^{max}(t) = k_v \frac{(v_c(t) - v_{th})^{1.2}}{v_c(t)\max_b(\mathbf{T}_c(t))^{1.19}}, \quad (8)$$

where $v_{th}$ is the threshold voltage and $k_v$ is the constant of proportionality. The max in (8) is taken over all the blocks in core $c$.

We have compiled the commonly used notation in this paper in Table II for easy reference.

TABLE II
COMMONLY USED NOTATION USED IN THE PAPER.

| Symbol | Meaning |
|---|---|
| $n$ | No. of cores |
| $q$ | No. of tasks |
| $m$ | No. of functional units in a core |
| $N$ | Total no. of functional units in a processor |
| $s_c$ | Normalized clock speed of core $c$ |
| $v_c$ | Normalized voltage of core $c$ |
| $v_t$ | Threshold voltage of circuit |
| $I_j$ | No. of instructions in task $j$ |
| $IPC_j$ | Instruction per cycle of task $j$ |
| $x_c(t)$ | No. of instructions completed by task in core $c$ by time $t$ |
| $P_{c,b}$ | Total power consumed by block $b$ in core $c$ |
| $P_{d,c,b}$ | Dynamic power of block $b$ in core $c$ |
| $P_{d,c,b}^{max}$ | Profile of maximum $P_{d,c,b}$ |
| $P_{l,c,b}$ | Leakage power of block $b$ in core $c$ |
| $T_{c,b}$ | Temperature of block $b$ in core $c$ |
| $T_{max}$ | Maximum allowed temperature |
| $\mathbf{G}$ | Conductance matrix of entire processor |
| $g_{T,c,b}$ | Temperature co-eff. of lkg. power of block $b$ in core $c$ |
| $P_{v,c}$ | Voltage co-eff. of leakage power of core $c$ |
| $\mathbf{M}$ | Task-to-core allocation matrix |
| $\mathbf{G}_{\text{die},c}$ | Conductance matrix of the die layer of core $c$ |
| $\mathbf{G}_{\text{tim},c}$ | Conductance matrix of the TIM layer of core $c$ |
| $\mathbf{T}_{pkg}$ | Package temperature vector |
| $T_{\text{spr}}$ | Temperature of the spreader center |

## III. MAKESPAN MINIMIZATION

### A. Problem Definition

Consider $q$ tasks (not necessarily identical). The problem is to determine the assignment of tasks to cores and the transient voltages and speeds of cores, such that the latest task completion time or the makespan of all tasks is minimized, subject to constraints on temperature, speeds and voltages.

Let $\mathbf{M}$ be an $n \times q$ matrix that represents the assignment of tasks to cores. It is defined by

$$M(i,j) = \begin{cases} 1, & \text{if task } j \text{ is assigned to core } i, \\ 0, & \text{otherwise.} \end{cases}$$

The following equations express the bijective task-to-core mapping.

$$\sum_{i=1}^{n} \mathbf{M}(i,j) = 1, \ \sum_{j=1}^{q} \mathbf{M}(i,j) = 1, \ \forall i,j. \quad (9)$$

Then the general problem can be stated as follows.

$$\min_{\mathbf{s}(t),\mathbf{v}(t),\mathbf{M}(t)} \quad t_f = \max_{1\leq c\leq n} t_{f,c}, \quad (10)$$
$$s.t. \quad \frac{d\mathbf{x}(t)}{dt} = \mathbf{s}(t)\mathbf{M}(t)\mathbf{IPC}(t), \quad (11)$$
$$\mathbf{x}(0) = 0, \ \mathbf{x}(t_f) = \mathbf{I}, \quad (12)$$
$$\frac{d\mathbf{T}(t)}{dt} = \hat{\mathbf{A}}\mathbf{T}(t) + \mathbf{B}[\mathbf{P}_d(\mathbf{M},\mathbf{s},\mathbf{v},t) + \mathbf{P}_v(\mathbf{v},t) + \mathbf{P}_{l0}], \ \forall t, \quad (13)$$
$$\mathbf{P}_d(\mathbf{M},\mathbf{s},\mathbf{v},t) = \mathbf{P}_{dq}^{max}(t)\mathbf{M}'diag(\mathbf{v}(t))^2\mathbf{s}(t), \ \forall t, \quad (14)$$
$$\mathbf{T}(0) = \mathbf{T_0}, \quad \mathbf{T}(t) \leq T_{max}, \ \forall t, \quad (15)$$
$$0 \leq s_c(t) \leq k_v \frac{(v_c(t) - v_{th})^{1.2}}{v_c(t)\max_b(\mathbf{T}_c(t))^{1.19}}, \ \forall t,c, \quad (16)$$
$$\mathbf{0}_{n\times 1} \leq \mathbf{v}(t) \leq \mathbf{1}_{n\times 1}, \ \forall t. \quad (17)$$

$t_{f,c}$ refers to the task completion time of task on core $c$ and $t_f$ denotes the final completion time or the makespan of all tasks. Here $\mathbf{x}(t)$ is the number of instructions that have been completed by tasks on respective cores as dictated by the allocation matrix $\mathbf{M}$ at time $t$. Equation (11) relates the speed of a core to its execution rate. Each task starts at time 0 and finishes by time $t_f$ as described in (12).

$\mathbf{P}_d(\mathbf{M}, \mathbf{s}, \mathbf{v}, t)$ maps the power consumption of tasks to cores as described in (14). $\mathbf{P}_{dq}^{max}$ is a matrix of size $N \times q$, where column $j$ contains the $\mathbf{P}_d^{max}$ of a task $j$. In (14), by multiplying $\mathbf{P}_{dq}^{max}$ by $\mathbf{M}'$, we are choosing the tasks according to the allocation specified in $\mathbf{M}$ and combining with the cores characteristics ($\mathbf{s}$, $\mathbf{v}$) to derive the total dynamic power consumption of each core.

Task allocation or migration has high performance penalty compared with the voltage-speed scaling. Hence the migration interval is typically chosen around 50–100 ms much larger than the die thermal time constant (5–10 ms). Once the tasks are allocated, the voltage-speed scaling for each core is performed on the order of the die thermal time constant. These two problems are addressed separately in the following sections.

## IV. Voltage-speed Scaling

### A. Problem Definition and the Optimal Policy

The formulation of the transient voltage-speed control to minimize the makespan of tasks within a migration interval is presented here. Since the task allocation is known at the beginning of the migration interval, without loss of generality, the task and the core number are the same and used interchangeably. $I_c$ denotes the number of instructions of the task assigned to core $c$. The control variables, viz., the speed and the voltage are continuous, hence the problem is formulated as an optimal control problem.

$$\min_{\mathbf{s}(t),\mathbf{v}(t)} \quad t_f = \max_{1\le c\le n} \; t_{f,c}, \tag{18}$$

$$s.t. \quad \frac{dx_c(t)}{dt} = IPC_c s_c(t), \ \forall c, \tag{19}$$

$$\mathbf{x}(0) = 0, \ \mathbf{x}(t_f) = \mathbf{I}, \tag{20}$$

$$\frac{d\mathbf{T}(t)}{dt} = \hat{\mathbf{A}}\mathbf{T}(t) + \mathbf{B}[\mathbf{P}_d(\mathbf{s}, \mathbf{v}, t) + \mathbf{P}_v(\mathbf{v}, t) + \mathbf{P}_{l0}], \ \forall t, \tag{21}$$

$$\mathbf{T}(0) = \mathbf{T}_0, \quad \mathbf{T}(t) \le T_{max}, \ \forall t, \tag{22}$$

$$0 \le s_c(t) \le k_v \frac{(v_c(t) - v_{th})^{1.2}}{v_c(t) \max_b(\mathbf{T}_c(t))^{1.19}}, \ \forall t, c, \tag{23}$$

$$\mathbf{0}_{n\times 1} \le \mathbf{v}(t) \le \mathbf{1}_{n\times 1}, \ \forall t. \tag{24}$$

The above formulation is a *minimum time problem* in optimal control theory [35]. Since the task allocation is fixed for the duration of the voltage-speed scaling, the formulation does not contain $\mathbf{M}$. $\mathbf{x}$ and $\mathbf{T}$ are two state variables with fixed initial conditions. $\mathbf{x}$ has a variable endpoint $t_f$. The *mixed control-state point-wise inequalities* (22), (23) complicate the solution process. The solution is obtained through the use of *direct adjoining approach* [36]. Only the final solution is presented below and the details of the derivation can be found in Appendix A.

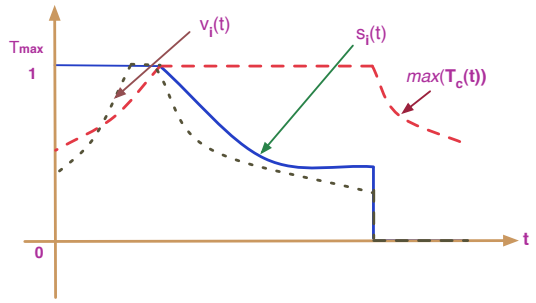


Fig. 5. Figure showing the optimal voltage-speed policy of a core as described in (25) and (26).

Let $s_{max,c}$ be the speed which sets the temperature of the hottest block of core $c$ at the maximum. Then the optimal speed policy for the core $c$ is given by:

$$s_c^*(t) = \begin{cases} 1, & \max(\mathbf{T_c}(t)) < T_{max}, \\ 0, & \max(\mathbf{T_c}(t)) > T_{max}, \\ s_{max,c}(t), & \max(\mathbf{T_c}(t)) = T_{max}. \end{cases} \tag{25}$$

The corresponding voltage is calculated by solving the following equation numerically.

$$s_c(t) = k_v \frac{(v_c(t) - v_{th})^{1.2}}{v_c(t) \max_b(\mathbf{T}_c(t))^{1.19}}, \ \forall t, c. \tag{26}$$

The above policy, which we refer to as the *zero-slack* policy suggests that in order to minimize the overall makespan, either the speed of a core is set to the maximum when the temperatures of all thermal blocks in that core are less than the maximum or the speed should be set such that atleast one of the thermal blocks in that core is at the maximum specified temperature.

Fig. 5 shows the typical optimal voltage-speed profile for a core. The speed is deduced from (25) and the corresponding voltage from (26). Initially the temperature of the hottest block is assumed to be less than the maximum, hence the speed is set to the maximum, which corresponds to the first condition in (25). As the core temperature begins to increase, the voltage also increases in order to satisfy the delay constraint in (26). Once the temperature of the hottest block has reached the maximum, the speed is throttled such that the temperature is maintained at the maximum, which is the speed $s_{r,c}(t)$. The corresponding voltage is derived from (26).

The above *zero-slack* policy is just a policy or a guideline to set core speeds and voltages to minimize the makespan. It does not provide a mechanism to implement the policy. In order to find an implementation, we note that the core speeds and voltages determined through the zero-slack policy depends only on the current temperature of the cores and is independent of the core speeds, voltages and temperatures at any other time instants. Thus *global minimization of makespan can be achieved through local maximization of instantaneous throughput*. For an online implementation of the zero-slack

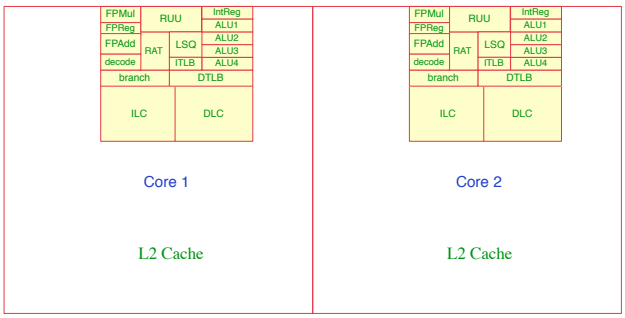


Fig. 6. Dual-core floorplan of Alpha 21264 processor [37].

policy, we discretize the execution time into short time intervals referred to as the *scheduling intervals*, where the core speeds and voltages are held fixed. With this setup, the zero-slack policy can be implemented as the solution of a convex optimization problem discussed in the following section.

### B. Convex Optimization Formulation and its Solution

In order to implement the zero-slack policy at every scheduling interval, the temperature of all thermal blocks at any scheduling interval should be computable. Let $t_s$ be the length of the scheduling interval, then the temperature at the $k^{th}$ interval can be computed as (obtained by discretizing (7)) [38],

$$\mathbf{T}(kt_s) = e^{\hat{\mathbf{A}}t_s}\mathbf{T}((k-1)t_s) + \hat{\mathbf{A}}^{-1}(e^{\hat{\mathbf{A}}t_s} - \mathbb{1}_{N\times N})\mathbf{B}\hat{\mathbf{P}}(\mathbf{s},\mathbf{v},kt_s), \tag{27}$$

where

$$\hat{\mathbf{P}}(\mathbf{s},\mathbf{v},kt_s) = \mathbf{P}_d(\mathbf{s},\mathbf{v},kt_s) + \mathbf{P}_v(\mathbf{v},kt_s) + \mathbf{P}_{l0} \tag{28}$$

and $\mathbb{1}$ is the identity matrix. For simplicity of notation, the use of $t_s$ is omitted and $\mathbf{T}(kt_s)$ is simply written as $\mathbf{T}(k)$. Rewriting the above equation with $\mathbf{E} = e^{\hat{\mathbf{A}}t_s}$ and $\mathbf{R} = \hat{\mathbf{A}}^{-1}(e^{\hat{\mathbf{A}}t_s} - \mathbb{1}_{N\times N})\mathbf{B}$,

$$\mathbf{T}(k) = \mathbf{E}\mathbf{T}(k-1) + \mathbf{R}\hat{\mathbf{P}}(\mathbf{s},\mathbf{v},k). \tag{29}$$

With the above discretization, the problem of instantaneous throughput maximization for every scheduling interval is formulated as a convex optimization problem.

$$\max_{\mathbf{s},\mathbf{v}} \quad \sum_c s_c(k), \ \forall c, \tag{30}$$

$$s.t. \quad \mathbf{T}(k) = \mathbf{E}\mathbf{T}(k-1) + \mathbf{R}\hat{\mathbf{P}}(\mathbf{s},\mathbf{v},k), \tag{31}$$

$$\mathbf{T}(k) \le T_{max}, \tag{32}$$

$$0 \le s_c(k) \le k_v \frac{(v_c(k) - v_{th})^{1.2}}{v_c(k)\max_b(\mathbf{T}_c(k))^{1.19}}, \ \forall c, \tag{33}$$

$$\mathbf{0}_{n\times 1} \le \mathbf{v}(k) \le \mathbf{1}_{n\times 1}. \tag{34}$$

The proof of convexity of the above optimization problem is derived in Appendix B.

The computational time to solve the above convex optimization problem does not scale well with the problem size and thus it is less attractive for an online implementation. With the aid of realistic assumptions, a fast computational procedure is derived in the following section.

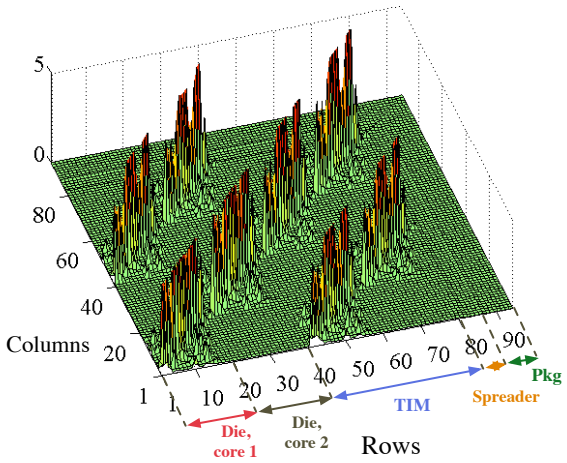


Fig. 7. Plot of values of matrix $\mathbf{R}$ for the dual-core Alpha processor with scheduling interval $t_s = 10\ ms$ [11].

### C. Fast Computational Procedure

In this section, a fast computational method, which avoids the complexity of convex optimization is proposed. By making use of certain useful properties that matrix $\mathbf{R}$ exhibits for scheduling intervals on the order of the die thermal time constant, an efficient computational procedure is developed, whose computational time complexity is *linear in the number of cores and logarithmic in the number of discrete states of speeds and voltages*.

In order to understand the structure of $\mathbf{R}$, we need to know the typical multi-core floorplan. Fig. 6 shows the floorplan of a dual core Alpha 21264 processor as an example of a typical multi-core floorplan. This floorplan is constructed by replicating the single-core floorplan, such that the processing units of cores are separated by L2 caches. This floorplan is shown to be beneficial as it produces fewer thermal hotspots compared to other floorplans due to reduced lateral heat flow [39].

Next, the properties of $\mathbf{R}$ are studied for scheduling intervals on the order of the die thermal time constant. Fig. 7 shows a typical plot representing values of $\mathbf{R}$ for the dual-core Alpha 21264 processor with the scheduling interval set to the die thermal time constant, which is typically 10 ms for the Alpha processor. The arrangement of die, TIM, spreader and package blocks of the matrix are shown in the figure. Note that the Alpha 21264 processor has 20 functional blocks in both the die and the TIM layers.

From (29), we see that the temperature $\mathbf{T}$ is obtained by multiplying $\mathbf{R}$ with the power vector $\hat{\mathbf{P}}$. From Fig. 7, we see that the power dissipation of a die section of a certain core affects mainly the temperature of the die section of the same core. For example, we see that rows 1–20 are populated mostly between columns 1–20 (die) and 41–60 (TIM). Noting that the power vector is non-zero only for the die blocks, we see that the power dissipation of the die section of core 1 is mostly responsible for the temperature of the die section of core 1. This observation can be explained due to the presence of large and cooler inter-core caches (see Fig. 6) which reduce the inter-core lateral heat transfer. Since the scheduling intervals

**Input** : Power profile, instruction length of tasks.
Initial temperature.
Available speed and voltage states.
**Output**: Speed and voltage profile
**for** *each scheduling interval* $kt_s$ **do**
  **for** *each core* $c$ **do**
    Run binary search on $s_c$ such that
    $\mathbf{T}_{die,c}(kt_s) = T_{max}$ (see (35));
    Let $s_c(kt_s)$ be the corresponding speed;
    $s_c(kt_s) = \min(s_c(kt_s), 1)$;
    Solve (26) to find the corresponding $v_c(kt_s)$;
  **end**
**end**

**Algorithm 1**: Fast computational procedure to determine the voltage-speed scaling to minimize makespan.

are chosen on the order of the die thermal time constant, which is orders of magnitude lesser than the package thermal time constant, the heat generation due to the power dissipation of a core gets localized within the core for these short intervals of time. Thus the temperature of a core is mainly derived from the power of that core for time durations on the order of the die thermal time constant.

The above observation helps in the local determination of speeds and voltages of cores based on their own power dissipation and independent of the speeds and voltages of other cores. The simplified temperature computation of the die section of core $c$ is given by

$$\mathbf{T}_{die,c}(k) = \mathbf{E}\mathbf{T}_{die,c}(k-1) + \mathbf{R}_{die,c}\hat{\mathbf{P}}_{die,c}(k), \qquad (35)$$

The matrix $\mathbf{R}_{die,c}$ corresponds to the die section of core $c$ in $\mathbf{R}$. For the dual-core processor shown in Fig. 6, $\mathbf{R}_{die,1}$ corresponds to rows 1–20 and columns 1–20 of $\mathbf{R}$. Note that the matrix $\mathbf{R}_{die,c}$ is of constant size (number of functional units) irrespective of the number of cores. Thus the temperature computation of an $n$ core processor is linear in the number of cores.

This isolation of temperature determination enables the use of binary search technique to determine the speed and the voltage of each core such that they satisfy the optimal policy (25). The time complexity of the binary search method is logarithmic in the number of discrete states of speeds and voltages. The above procedure is summarized in Algorithm 1.

## V. Task-to-core Allocation

### A. Problem Definition

The goal of the task-to-core allocation is to determine an optimal allocation for every migration interval, which when combined with the optimal voltage-speed scaling policy (derived in Section IV) within the migration interval minimizes the overall makespan.

Recall that the optimal speed policy is given by (25). In [10], the authors have shown for speed-only DTM (neglecting the die capacitances) that the optimal speed curve is an exponential function, whose time constant is that of the package thermal time constant. It can be shown that the optimal speed policy derived in Section IV-A follows this exponential curve in the general sense with variations along the path.

The above inference is useful in comparing two speed profiles. It can be deduced that a speed function with higher steady-state speed has higher overall throughput (integral of speed over the duration of execution) than a speed function with lower steady-state speed. Hence the steady-state speed can be used as a metric for throughput comparison of different speed functions. We previously mentioned that the tasks are allocated at intervals much longer than the die thermal time constant. At these intervals, the die and the TIM capacitances saturate. Thus we can conduct the steady-state thermal analysis for the die and the TIM. Note that the package temperature can be assumed to remain constant during this interval as its thermal time constant (1–2 min) is orders of magnitude higher than the migration interval.

From the zero-slack policy stated in Section IV-A, we know that minimizing makespan can be achieved by maximizing the instantaneous throughput. Hence we replace the objective in the formulation in Section III with maximizing throughput ($\mathbf{s}_{ss}$) for the purpose of determining the task-to-core allocation as shown below.

$$\max_{\mathbf{s}_{ss},\mathbf{v}_{ss},\mathbf{M}} \quad \sum_c s_{c,ss}, \ \forall c, \qquad (36)$$

$$s.t. \quad \mathbf{T} = -\hat{\mathbf{A}^{-1}}\mathbf{B}[\mathbf{P}_d(\mathbf{M},\mathbf{s}_{ss},\mathbf{v}_{ss}) + \mathbf{P}_v(\mathbf{v}_{ss}) + \mathbf{P}_{l0}], \qquad (37)$$

$$\mathbf{P}_d(\mathbf{M},\mathbf{s}_{ss},\mathbf{v}_{ss}) = \mathbf{P}_{dq}^{max}\mathbf{M}' diag(\mathbf{v}_{ss})^2 \mathbf{s}_{ss}, \qquad (38)$$

$$\mathbf{T} \le T_{max}, \quad \mathbf{T}_{pkg} = \mathbf{T}_{pkg,0}, \qquad (39)$$

$$0 \le s_{c,ss} \le k_v \frac{(v_{c,ss} - v_{th})^{1.2}}{v_{c,ss} \max_b (\mathbf{T}_{c0})^{1.19}}, \ \forall c, \qquad (40)$$

$$\mathbf{0}_{n\times 1} \le \mathbf{v}_{ss} \le \mathbf{1}_{n\times 1}. \qquad (41)$$

$s_{c,ss}$ and $v_{c,ss}$ are the steady-state speed and voltage of core $c$. $\mathbf{T}_{pkg}$ is the package temperature vector. Equation (37) is the steady-state temperature derived by setting $\frac{d\mathbf{T}}{dt} = 0$ in (7). Equation (38) is same as (14), but for steady-state conditions.

The solution to the above optimization problem is complicated by the fact that it contains a mixed-integer non-linear constraint in (38) involving $\mathbf{s}_{ss}$, $\mathbf{v}_{ss}$ and $\mathbf{M}$. Performing a mixed-integer non-linear optimization in real time is computationally hard. We remove this non-linearity by making the observation that the lateral temperature contribution from neighboring cores to a core is less due to the presence of large caches. This simplification is detailed in the following section.

### B. Structure of the HotSpot Conductance Matrix

Given a floorplan, it is easy to construct the conductance matrix $\mathbf{G}$ using the HotSpot thermal circuit model [33]. Fig. 8 shows the sparsity plot of the conductance matrix for the dual-core Alpha processor shown in Fig. 6. A non-zero entry in the matrix is represented by a dot in the plot. It is seen from the figure that most of the entries are concentrated along the main diagonal and the off-diagonal elements of the die and the TIM sections; the package and the spreader sections. The square diagonal blocks represent the lateral resistances of the die and the TIM of the cores. The vertical resistances connecting

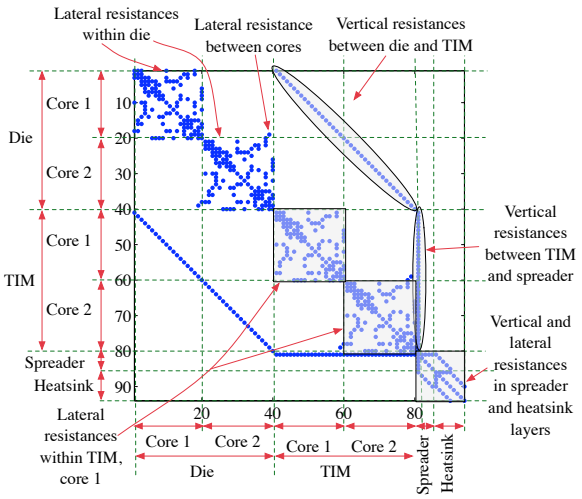


Fig. 8. Plot of the conductance matrix showing the sparsity of the dual-core Alpha processor floor plan shown in Fig. 6 [37].

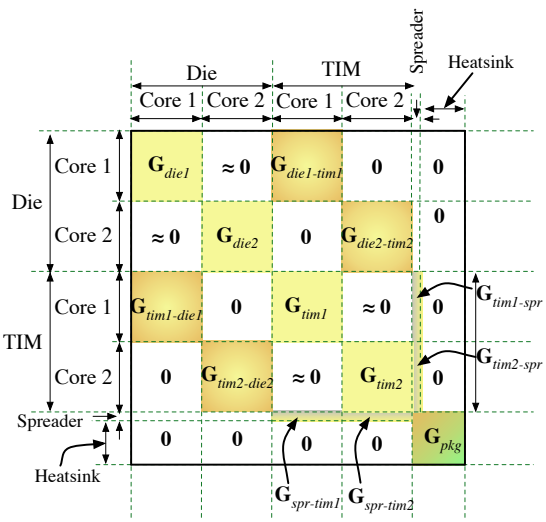


Fig. 9. Components of the thermal conductance matrix $\mathbf{G}$ of the dual-core Alpha processor floor plan shown in Fig. 6 [37].

the die and the TIM layers of the chip are shown by the off diagonal elements. The vertical and the horizontal strip of dots in the figure represent the vertical resistances between the TIM and the spreader.

As seen from Fig. 6, each core is surrounded by a large cache. These caches act as cooler regions of the chip and help in reducing the lateral heat flow between the cores. Hence fewer lateral resistances between the cores are seen in Fig. 8. Based on this inference, the sparse blocks of the matrix corresponding to the inter-core die and TIM resistances can be neglected and the resultant conductance matrix is labeled as shown in Fig. 9.

The diagonal components of the conductance matrix in Fig. 9 are named $\mathbf{G}_{\text{die}}$ and $\mathbf{G}_{\text{tim}}$ corresponding to the die and the TIM layers of the chip respectively. $\mathbf{G}_{\text{die-tim}}$ and $\mathbf{G}_{\text{tim-die}}$ denote the same vertical conductances connecting the die and the TIM layers. Similarly, $\mathbf{G}_{\text{tim-spr}}$ and $\mathbf{G}_{\text{spr-tim}}$ denote the conductances connecting the spreader layer to the TIM layer. Finally, the package conductances are denoted by $\mathbf{G}_{\text{pkg}}$.

### C. Simplified Temperature Computation

The total power dissipation of a processor for a given task allocation is given by:

$$\mathbf{GT} = \mathbf{P}_d(\mathbf{s}, \mathbf{v}, t) + \mathbf{G}_T\mathbf{T} + \mathbf{P}_v(\mathbf{v}, t) + \mathbf{P}_{l0}. \tag{42}$$

Applying Kirchhoff's current law for the die and the TIM layers, the following equations are derived for task $j$ executing on core $i$:

$$\mathbf{G}_{\text{die,ij}}\mathbf{T}_{\text{die,i}} + \mathbf{G}_{\text{die-tim,i}}\mathbf{T}_{\text{tim,i}} = s_{ij}v_{ij}^2\mathbf{P}_{d,j}^{max} + \mathbf{G}_{T,\text{i}}\mathbf{T}_{\text{die,ij}} + \mathbf{P}_{v,i}v_{ij} + \mathbf{P}_{l0,i}, \tag{43}$$

$$\mathbf{G}_{\text{die-tim,i}}\mathbf{T}_{\text{die,ij}} + \mathbf{G}_{\text{tim,i}}\mathbf{T}_{\text{tim,i}} + \mathbf{G}_{\text{tim-spr,i}}T_{\text{spr}} = 0. \tag{44}$$

$\mathbf{T}_{\text{die,i}}$ and $\mathbf{T}_{\text{tim,i}}$ denote the temperature vectors of the die and the TIM units of core $i$ respectively. $T_{\text{spr}}$ is the scalar temperature of the spreader center. $T_{\text{spr}}$ can be computed uniquely for a given $\mathbf{T}_{\text{pkg}}$ [37]. $s_{ij}$, $v_{ij}$ and $\mathbf{T}_{\text{die,ij}}$ are the steady-state speed, voltage and temperature of die units of core $i$ when executing task $j$. Note that the subscript "ss" is dropped to keep the notations clean.

To simplify the notations, the following are defined.

$$\mathbf{G}_{app,i} \triangleq \mathbf{G}_{\text{die,i}} - \mathbf{G}_{\text{die-tim,i}}\mathbf{G}_{\text{tim,i}}^{-1}\mathbf{G}_{\text{die-tim,i}} - \mathbf{G}_{T,\text{i}}, \tag{45}$$

$$\mathbf{P}_{app,i} \triangleq \mathbf{G}_{\text{die-tim,i}}\mathbf{G}_{\text{tim,i}}^{-1}\mathbf{G}_{\text{tim-spr,i}}T_{\text{spr}} + \mathbf{P}_{l0,i}. \tag{46}$$

With the above definitions, $\mathbf{T}_{\text{die,ij}}$ is calculated from (43) and (44) as

$$\mathbf{T}_{\text{die,ij}} = \mathbf{G}_{app,i}^{-1}[\mathbf{P}_{app,i} + s_{ij}v_{ij}^2\mathbf{P}_{d,j}^{max} + \mathbf{P}_{v,i}v_{ij}]. \tag{47}$$

The problem stated in Section V-A is re-formulated using the above simplifications.

$$\max_{\mathbf{s},\mathbf{M}} \quad \mathbf{M's}, \tag{48}$$

$$s.t.\ \mathbf{T}_{\text{die,i}} = \mathbf{G}_{app,i}^{-1}[\mathbf{P}_{app,i} + s_iv_i^2\mathbf{P}_{dq}^{max}\mathbf{M}_i^T + \mathbf{P}_{v,i}v_i],\ \forall i, j, \tag{49}$$

$$\mathbf{T} \le T_{max}, \quad T_{\text{spr}} = T_{\text{spr0}}, \tag{50}$$

$$0 \le s_i \le k_v\frac{(v_i - v_{th})^{1.2}}{v_i \max_b(\mathbf{T}_c)^{1.19}},\ \forall i, \tag{51}$$

$$\mathbf{0}_{n\times 1} \le \mathbf{v} \le \mathbf{1}_{n\times 1}. \tag{52}$$

$\mathbf{M}_i$ refers to the $i^{th}$ row of matrix $\mathbf{M}$. This formulation is converted to a linear assignment problem as illustrated in the next section. Note that the initial condition on $\mathbf{T}_{pkg}$ in (39) has been replaced with the initial condition on $T_{spr}$ in (50) as the $\mathbf{T}_{pkg}$ can be derived exactly knowing $T_{spr}$ [37].

### D. Linear Assignment Problem

The optimal speeds and voltages of cores for a given allocation can be obtained by setting $\mathbf{T}_{\text{die,ij}} = T_{max}$ and solving (47) and (26) simultaneously (the corresponding $s_{ij} \le 1$). Thus the computation of speed for a core executing a given task is independent of the speed computations of other cores. Let $\mathbf{S}$ be a matrix, whose elements are $s_{ij}$, where $i$, $j$ refer

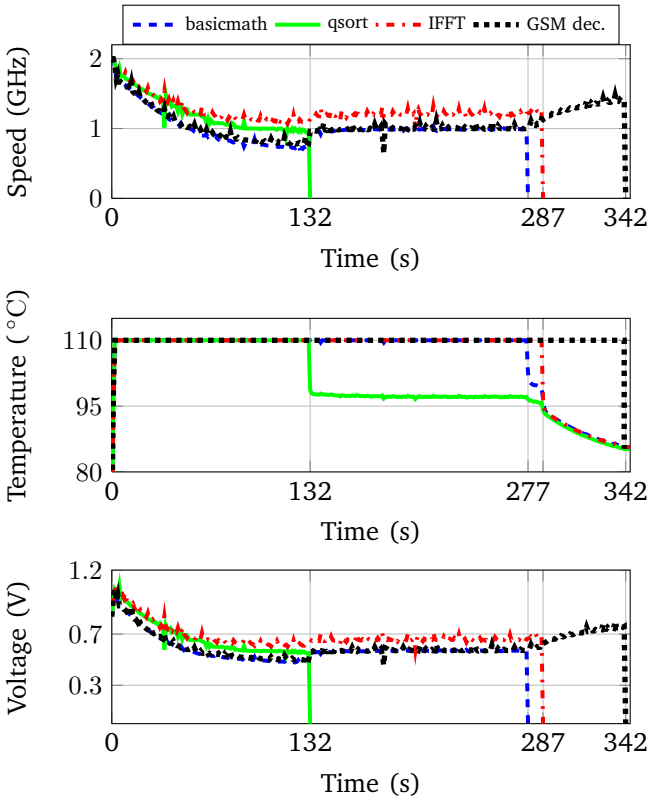


Fig. 10. Performance of the optimal DVFS scheme in minimizing the overall makespan

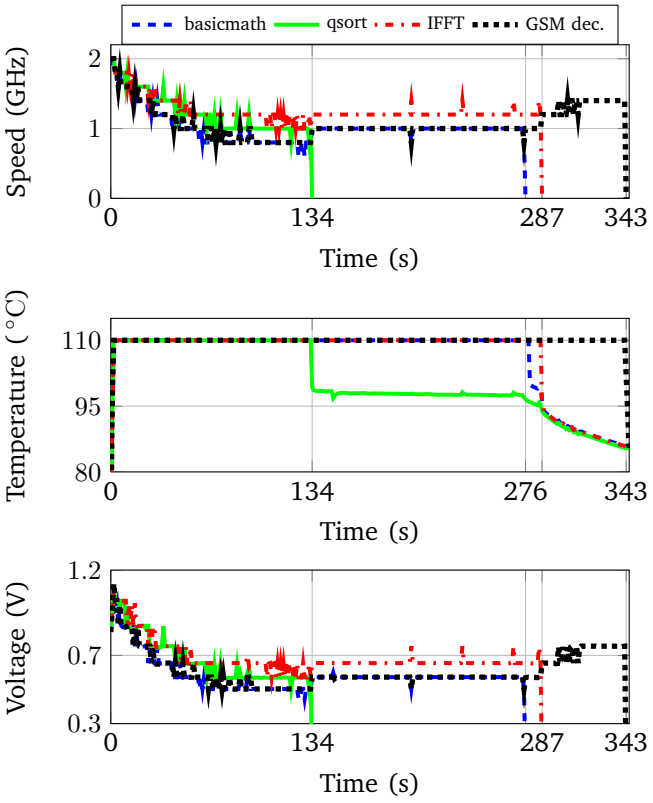


Fig. 11. Discretization of the optimal makespan algorithm with ten speed states

to the core and the task number respectively. This matrix is referred to as the *speed matrix*.

Given this *speed matrix* $\mathbf{S}$, the problem of optimal task-to-core allocation is formulated as

$$\max_{\mathbf{M}} \quad \sum \mathbf{M}'\mathbf{S}, \tag{53}$$

$$s.t. \quad \sum_{i=1}^{n} M(i,j) = 1, \ j \in \{1,\ldots,q\}, \tag{54}$$

$$\sum_{j=1}^{q} M(i,j) = 1, \ i \in \{1,\ldots,n\}, \tag{55}$$

$$M(i,j) \in \{0,1\}, \ i \in \{1,\ldots,n\}, \ j \in \{1,\ldots,q\}. \tag{56}$$

This is a linear assignment problem and has an efficient polynomial time solution ($O((nq)^3)$) using *Munkres* algorithm [40]. Once the allocation is determined using this algorithm, the speeds and the voltages within the migration interval are determined for every scheduling interval as described in Section IV.

## VI. Experimental Results

### A. Experimental Setup

A multi-core version of Alpha 21264 processor described in Section IV-C was used to experimentally validate the proposed techniques. We chose to use the Alpha processor for our experiments as the details of power and thermal models of other processors were not available. The thermal behavior of the processor was modeled using HotSpot-4 thermal-circuit model [33]. The tasks were obtained from MiBench benchmark [41]; their power values were extracted by simulating them using PTScalar [17]. The maximum temperature was limited to 110 °C and the maximum frequency of operation was set at 2 GHz. The supply voltages of cores were allowed to vary from 0.3 V to 1.2 V. The total dynamic power of the processor was restricted to 230 W, while the leakage power contribution was limited to 60 W. The convectional thermal resistance in HotSpot thermal model was set at 0.35 °C/W. The scheduling interval for the voltage-speed scaling method was fixed at 10 ms. The migration interval was fixed at 100 ms to minimize the computation overhead.

TABLE III
Characteristics of benchmark tasks used in the voltage-speed scaling experiment and the resultant task completion times.

| Benchmark | | basicmath | qsort | IFFT | GSM dec. |
|---|---|---|---|---|---|
| Avg. dyn. power (W) | | 60.13 | 56.75 | 55.41 | 57.54 |
| IPC | | 1.8 | 1.85 | 1.95 | 2.45 |
| Instructions (billion) | | 500 | 300 | 700 | 900 |
| Completion times (sec) | Optimal | 277 | 132 | 287 | 342 |
| | Discrete | 276 | 134 | 287 | 343 |

We have developed a thermal management simulator called *MAGMA* [42], which is capable of design space exploration, performing several offline and online thermal management techniques including the algorithms developed in this paper. The simulator includes the leakage dependence on temperature and allows user to trade-off model accuracy with the simulation time. The simulator is open source and is available for public download.

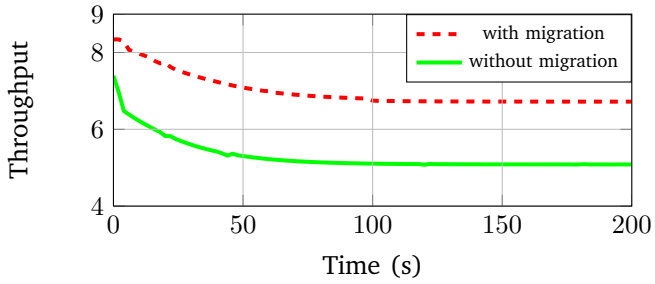


Fig. 12. Comparison of throughput with and without task migration.

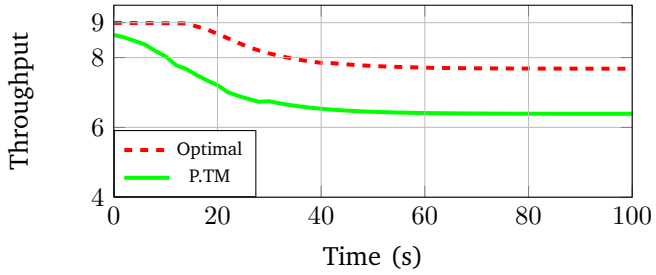


Fig. 13. Comparison of temporal performance of the optimal allocation scheme with the P.TM scheme.

TABLE IV
COMPARISON OF THE PROPOSED APPROXIMATE METHOD WITH THE ACCURATE CONVEX OPTIMIZATION METHOD.

| No. of cores | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|
| Speed up | 3.27 | 4.06 | 5.59 | 10.52 | 24.13 |
| Error(%) | 0.023 | 0.05 | 0.07 | 0.15 | 0.35 |

### *B. The Optimal Makespan Minimization Policy*

Fig. 10 shows the results of execution of the optimal makespan policy (Section IV) for four tasks executing on four cores. The temperature plots are for the hottest block in each core. The characteristics of the tasks are listed in Table III. It also lists the task completion times.

Since we did not find an equivalent work which considers both voltage and frequency scaling, with time-varying power profiles to minimize the makespan, we could not provide a comparison of our optimal makespan policy with other works.

In Fig. 10, the temperature of all cores are initially below the maximum specified temperature of 110 °C. Hence the speeds are set to the maximum. As the temperatures increase, the speeds of the cores are throttled exponentially to maintain the core temperatures at the maximum. We also see that once a task is completed, it enables other tasks to execute at faster pace as there is less power consumption. For example, after task *qsort* (132 s) and *basicmath* (277 s) finish their executions, we see an increase in the speeds of other tasks. The optimal DVFS scheduling resulted in a makespan of **342** s.

### *C. Comparison of the Approximate Solution with the Convex Optimization Solution for Makespan Minimization*

The results of using the faster but approximate method described in Section IV-C in comparison with the convex optimization method (Section IV-B) are shown in Table IV. The error reported is simply the relative difference between the speeds obtained from the convex optimization approach and the approximate method. The approximate procedure improves the computation time significantly (**3X – 24X**) with increasing number of cores, while the error in accuracy is less than **0.4**%.

### *D. Discrete Voltage-speed Implementation*

To demonstrate that our proposed procedure is suitable for practical implementation, we computed an approximate discrete voltage-speed policy using ten speed and voltage states (current Intel processors support upto ten voltage-speed states) as shown in Fig. 11. The tasks conditions are the same as in Table III. The speeds are chosen such that the highest discrete speed which satisfies the thermal constraint is chosen. This may lead to frequent toggling between neighboring speed states as seen in the figure to maintain the temperature at the maximum according to the zero-slack policy. Since the discrete implementation is an approximation to the continuous optimal policy, it results in a slightly higher makespan (**343** s) and longer completion times of tasks compared with the optimal policy as seen from Table III.

### *E. Performance Improvement through Task Migration*

This section demonstrates the advantages of task migration in improving the overall performance by comparing the throughput improvement by executing a set of tasks with and without task migration. Fig. 12 shows the plot of throughput of a four core processor executing four tasks chosen out of a set of eight tasks for a duration of 200 s with the initial package temperature set at 35 °C. Throughput is measured as the sum of core speeds weighted by their respective IPCs.

First, we execute tasks according to the optimal voltage-speed scaling alone as derived in Section IV for a fixed allocation of tasks to cores. Next, we execute the same set of tasks using both the task migration (according to Section V) and the optimal voltage-speed scaling. We see **32.3**% improvement in throughput with the task migration. Hence demonstrating the need for task migration to improve performance.

### *F. Performance Comparison of the Optimal Task-to-core Allocation with the Power-based Thread Migration*

The comparison of performance of the proposed optimal task-to-core allocation with the power based thread migration (P.TM) [2] method is presented here. The mapping algorithms were required to choose four tasks among eight tasks to execute on a four-core processor for every migration interval. Note that each core still executes only one task.

In P.TM, the cores are sorted by their current temperatures (increasing) and tasks are sorted by their power dissipation numbers (decreasing). At the beginning of every migration interval, task $i$ is mapped to core $i$ according to their respective lists, i.e. the highest power dissipating task is assigned to the coldest core and the least power dissipating task to the hottest core.

Fig. 13 shows the plots of the throughputs (sum of core speeds weighted by their IPCs) of the proposed optimal

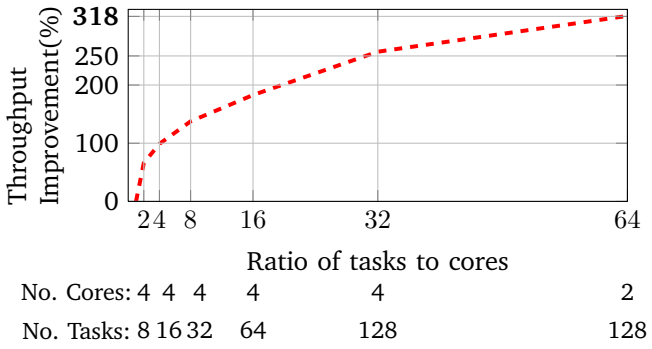


Fig. 14. Plot of throughput improvement of the proposed optimal algorithm against the P.TM technique for various ratios of tasks and cores

allocation with the P.TM technique for a four core processor with eight available tasks. The simulation is performed for a duration of 100 s with the initial package temperature set at 35 °C. The throughput within a migration interval is obtained using the optimal voltage-speed scaling as described in Section IV.

Since the P.TM method does not predict the temperatures for all combinations of tasks and cores, and considers only the current temperature, it cannot take into account the heterogeneity of cores and the spatial variation of leakage power. Also, if some cores have the same temperature, the P.TM cannot decide on the mapping of tasks to those cores. Hence P.TM results in lower throughput compared with the proposed allocation technique. The spatial variation of leakage power was set at 30% of the maximum leakage power. The plots demonstrate that the proposed optimal allocation scheme has a throughput improvement of **20.2**% over the P.TM technique at the steady-state conditions, which is equivalent of losing the throughput of one core among five cores.

### *G. Effect of Cores and Tasks on the Performance of Task-to-core Allocation*

Fig. 14 shows the performance of the proposed optimal task allocation for various combinations of number of tasks and cores against the P.TM scheme for similar configuration. The plot shows the percentage improvement in throughput. Higher task-to-core ratio provides more flexibility for the proposed algorithm to extract the maximum throughput in comparison with the P.TM scheme. This can be observed from the figure with improvements as high as **3.2X**.

### *H. Computation Times*

Table V shows the computation time for both the proposed task-to-core allocation and voltage-speed scaling methods for various number of cores under DVFS. The table also includes the simulation time for the P.TM method. The simulation was conducted in Matlab, run on Intel Core 2 duo processor at 1.67 GHz with a 2 GB RAM. The computation time for voltage-speed scaling method is the time taken to compute the speed and voltage by every core for one scheduling interval.

Although the P.TM method consumes negligible time to compute the mapping of tasks to cores, it is worth considering the trade-off between the loss in throughput and the simulation time.

TABLE V
COMPUTATION TIMES PER CORE (IN MILLISECONDS) FOR BOTH VOLTAGE-SPEED SCALING AND TASK-TO-CORE ALLOCATION SCHEMES.

| No. of cores | | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|
| Voltage-speed scaling | | 17.4 | 18.3 | 16.7 | 19.1 | 18.7 |
| Task-to-core allocation | Optimal | 138 | 167 | 126 | 157 | 142 |
| | P.TM | 3.44 | 3.41 | 3.51 | 3.47 | 3.43 |

As mentioned previously, each core can compute its own voltage and speed. Although solving the linear assignment problem to compute the task allocation cannot be completely parallelized, the *speed matrix* can be computed in parallel (as each core computes its own speed and voltage). Moreover, computational time for solving linear assignment problem through Munkres algorithm [40] takes negligible time compared to the time to compute the speed matrix. Hence we have shown only the computation time per core in the table. Note that the computation time per core in both the voltage-speed scaling and the task-to-core allocation methods are almost the same across the number of cores. Thus proving that the complexity of our algorithms is linear in the number of cores.

If the algorithms are compiled into executables, we expect a modest 30X improvement (verified through execution of few matrix operations implemented in Matlab and C). With this it can be verified that the computation times shown in the table for both the voltage-speed scaling and the task-to-core allocation schemes have small overhead (less than 6%) compared to the scheduling and the migration intervals used in the experiments respectively.

## VII. CONCLUSION

Developing efficient online techniques for multi-core processors is necessary for maximum processor utilization under changing workload conditions. In this paper, the optimal DVFS scheduling problem is addressed as separate problems of task-to-core allocation over migration intervals and voltage-speed scaling within migration intervals. Practically implementable techniques for both the problems are developed and shown to be optimal. Simulations demonstrate that our techniques achieve significant performance improvements over the existing techniques. The proposed approximate techniques are shown to be accurate within 0.4% of the optimal solution, but have orders of magnitude speedup.

## APPENDIX A
## PROOF OF OPTIMAL CONTROL POLICY

### *A. Hamiltonian Setup*

Consider the optimal control formulation in Section IV-A. First we construct the Hamiltonian [35] from the performance index (18), which can be written as $\int_0^{t_f} 1 \; dt$, and the state equality constraints (19) and (21) as shown below:

$$\mathcal{H}(\mathbf{x}, \mathbf{T}, \mathbf{s}, \mathbf{v}, \boldsymbol{\lambda}_x, \boldsymbol{\lambda}_T, \boldsymbol{\mu}, t) = 1 + \sum_c \lambda_{x,c}(t) IPC_c(t) s_c(t) + \boldsymbol{\lambda}_T' [\hat{\mathbf{A}}\mathbf{T}(t) + \mathbf{B}\{diag(\mathbf{P}_d^{max})\mathbf{X} diag(\mathbf{v}(t))^2 \mathbf{s}(t) + diag(\mathbf{P}_v)\mathbf{X}\mathbf{v}(t) + P_{l0}\}]. \quad (57)$$

$\boldsymbol{\lambda}_x$ and $\boldsymbol{\lambda_T}$ are the co-state variables that act as Lagrange multipliers for (19) and (21) respectively.

The states ($\mathbf{s}$, $\mathbf{v}$, $\mathbf{T}$) and the co-states ($\boldsymbol{\lambda}_x$, $\boldsymbol{\lambda}_T$) in the Hamiltonian (57) are required to satisfy the following:

$$\frac{d\mathbf{x}^*(t)}{dt} = \mathcal{H}^*_{\boldsymbol{\lambda}_x}(t), \qquad \frac{d\mathbf{T}^*(t)}{dt} = \mathcal{H}^*_{\boldsymbol{\lambda}_T}(t), \tag{58}$$

$$\frac{d\boldsymbol{\lambda}^*_x(t)}{dt} = -\mathcal{H}^*_{\mathbf{x}}(t), \qquad \frac{d\boldsymbol{\lambda}^*_T(t)}{dt} = -\mathcal{H}^*_{\mathbf{T}}(t). \tag{59}$$

Solving for the co-state variables,

$$\boldsymbol{\lambda}^*_x(t) = \boldsymbol{\lambda} \text{ (a constant)}, \tag{60}$$

$$\frac{d\boldsymbol{\lambda}^*_T(t)}{dt} = -\hat{\mathbf{A}}'\boldsymbol{\lambda}^*_T(t). \tag{61}$$

From state space methods [38], $\boldsymbol{\lambda}^*_T$ can be calculated as

$$\boldsymbol{\lambda}^*_T(t) = e^{-\hat{\mathbf{A}}t}\boldsymbol{\lambda}^*_T(0). \tag{62}$$

Let $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ be the vectors of constraint qualifiers [36] for (23) and (22) respectively, such that

$$\mu_c(t)g_c(t) = 0, \qquad \mu_c(t) \geq 0, \quad \forall t, c. \tag{63}$$

$$\boldsymbol{\nu}'(t)\mathbf{h}(t) = 0, \qquad \boldsymbol{\nu}(t) \geq 0, \quad \forall t. \tag{64}$$

where $g_c(t) = s_c(t) - k_v \frac{(v_c(t)-v_{th})^{1.2}}{v_c(t)\max\limits_b(\mathbf{T}_c(t))^{1.19}}$ and $\mathbf{h}(t) = (\mathbf{T}(t) - T_{max})$.

### *B. Optimal Voltage-Speed Profile*

Pontryagin minimum principle [35] states that the optimal controls $\mathbf{s}^*$, $\mathbf{v}^*$ need to satisfy the following inequality:

$$\mathcal{H}(\mathbf{x}^*, \mathbf{T}, \mathbf{s}^*, \mathbf{v}^*, \boldsymbol{\lambda}^*_x, \boldsymbol{\lambda}^*_T, \boldsymbol{\mu}^*, t) \leq \mathcal{H}(\mathbf{x}^*, \mathbf{T}^*, \mathbf{s}, \mathbf{v}, \boldsymbol{\lambda}^*_x, \boldsymbol{\lambda}^*_T, \boldsymbol{\mu}^*, t). \tag{65}$$

From the *direct adjoining approach* [36],

$$\mathcal{L}^*_{s_c}(t) = \mathcal{H}^*_{s_c}(t) + \mu_c g_{s_c}(t) = 0, \quad \forall t, c, \tag{66}$$

$$\mathcal{L}^*_{v_c}(t) = \mathcal{H}^*_{v_c}(t) + \mu_c g_{v_c}(t) = 0, \quad \forall t, c. \tag{67}$$

Replacing the Hamiltonian in (65), (66) and (67) with (57) and separating the controls for every core results in,

$$\min_{s_c(t), v_c(t)} \left[ \sum_{j=1}^{m} (P^{max}_{d,j}(t)v_c^2(t) \sum_{i=1}^{N} B_{i,j}\lambda_{T,i}(t) + \lambda_{x,c} IPC_c(t))\right] s_c(t) + \sum_{j=1}^{m} P^{max}_{v,j}(t)v_c(t), \tag{68}$$

$$\left[\lambda_{x,c}IPC_c(t) + \sum_{j=1}^{m}\left(P^{max}_{d,j}(t)v_c^2(t)\sum_{i=1}^{N}B_{i,j}\lambda_{T,i}(t)\right)\right] = -\mu_c \leq 0, \ \forall t, c, \text{ and} \tag{69}$$

$$\sum_{j=1}^{m}\left(P^{max}_{v,j}(t)v_c(t) + P^{max}_{d,j}(t)v_c^2(t)\sum_{i=1}^{N}B_{i,j}\lambda_{T,i}(t)\right) = \frac{k_v}{v_c^2(t)\max\limits_b(\mathbf{T}_c(t))^{1.19}}(v_c(t)-v_t)^{0.2}(0.2v_c(t)+v_t), \ \forall t, c. \tag{70}$$

respectively. Note that all the terms on the L.H.S and the R.H.S in (70) except $\lambda_{T,i}$ are non-negative by definition. This forces $\lambda_{T,i} \geq 0$. On a similar argument, on examining (69), $\lambda_{x,c} \leq 0$.

Hence to minimize (68), $s_c(t) = 1$ (see (69)). Since $\lambda_{T,i} \geq 0$ and $\lambda_{x,c} \leq 0$, $v_c(t) = 0$ to minimize (68). However, $v_c(t)$ is constrained below by (23). This transforms the voltage-speed inequality (23) to a equality (26). Similarly, $s_c(t)$ has an upper bound constraint (22). Thus the resulting optimal $s_c(t)$ is given by (25).

## APPENDIX B
## PROOF OF CONVEXITY OF VOLTAGE-SPEED SCALING

In order for the formulation in Section IV-B to be a convex optimization problem, the objective and all the inequality constraints need to satisfy either the first or the second order necessary and sufficient convexity conditions [43]. Here we have used the second order necessary and sufficient conditions to prove the convexity of the problem.

The second order necessary and sufficient conditions state that a function $f$, differentiable in the domain ($\mathbf{dom}$) of $f$ is convex iff the domain of $f$ is convex and its *Hessian* or second derivative is positive semi-definite, i.e.

$$\nabla^2 f(x) \succeq 0, \ \forall x \in \mathbf{dom} f. \tag{71}$$

The above second-order conditions can be easily verified for (30), (32) and (34). The proof of convexity for rest of the constraints viz. (31) and (33) are shown below.

Since $\mathbf{T}$ is linearly dependent on $\hat{\mathbf{P}}$ (as $\mathbf{T}(k-1)$ is a constant in $k^{\text{th}}$ interval) according to (35), proving the convexity of $\hat{\mathbf{P}}$ is sufficient to prove the convexity of $\mathbf{T}$. Note that the time indicator $k$ or $kt_s$ is dropped in the subsequent derivations as the convex formulation (30) – (34) remains the same for every scheduling interval.

$\hat{\mathbf{P}}$ is defined in (28). The first and second order derivatives of $\hat{\mathbf{P}}$ w.r.t $\mathbf{s}$ are given by

$$\nabla\hat{\mathbf{P}} = [2diag(\mathbf{P}^{max}_d)diag(\mathbf{v}) + diag(\mathbf{P}_v)]\mathbf{X}\nabla\mathbf{v}, \tag{72}$$

$$\nabla^2\hat{\mathbf{P}}(t) = 2diag(\mathbf{P}^{max}_d)\mathbf{X}[diag(\nabla\mathbf{v}) + \nabla^2\mathbf{v}] + diag(\mathbf{P}_v)\mathbf{X}\nabla^2\mathbf{v}. \tag{73}$$

Note that $v_c$ depends only on $s_c$ from (26). Hence finding the derivative of $v_c$ w.r.t $s_c$,

$$\frac{dv_c}{ds_c} = \frac{v_c^2\max\limits_b(\mathbf{T}_c)^{1.19}}{k_v}\frac{1}{(v_c-v_t)^{0.2}(0.2v_c+v_t)}. \tag{74}$$

Similarly, the second derivative is computed as

$$\frac{d^2v_c}{ds_c^2} = 2\frac{(\frac{dv_c}{ds_c})^2}{v_c}\left[\frac{0.2v_c^2 + 0.8v_tv_c - v_t^2}{(0.2v_c+v_t)(v_c-v_t)}\right]. \tag{75}$$

For $v_c \geq v_t$, it can be easily shown that $\frac{d^2v_c}{ds_c^2} \geq 0$. Hence voltage $v_c$ is convex w.r.t speed $s_c$ and thereby $\hat{\mathbf{P}}$ and $\mathbf{T}$ are also convex w.r.t $\mathbf{s}$ from (73).

Since all the constraints in the formulation are proved to be convex, the formulation turns out to be a convex optimization problem.

## REFERENCES


[1] S. Borkar, "Thousand Core Chips – A Technology Perspective," in *Proc. DAC*, 2007, pp. 746–749.

[2] P. Chaparro, J. González, G. Magklis, Q. Cai, and A. González, "Understanding the Thermal Implications of Multicore Architectures," *IEEE Trans. Parallel and Distributed Sys.*, vol. 18, pp. 1055–1065, 2007.

[3] S. Zhang and K. S. Chatha, "Approximation Algorithm for the Temperature-Aware Scheduling Problem," in *Proc. ICCAD*, 2007, pp. 281–288.

[4] A. Cohen, F. Finkelstein, A. Mendelson, R. Ronen, and D. Rudoy, "On Estimating Optimal Performance of CPU Dynamic Thermal Management," *IEEE Computer Architecture Letters*, vol. 2, pp. 6–6, 2003.

[5] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware Microarchitecture: Modeling and Implementation," *ACM Trans. Arch. Code Opt.*, vol. 1, pp. 94–125, 2004.

[6] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-performance Microprocessors," in *Proc. HPCA*, 2001, pp. 171–182.

[7] S. Cho and R. Melhem, "Corollaries to Amdahl's Law for Energy," *IEEE Computer Architecture Letters*, vol. 7, pp. 25–28, 2008.

[8] J. Li and J. F. Martínez, "Power-performance Considerations of Parallel Computing on Chip Multiprocessors," *ACM Trans. Archit. Code Optim.*, vol. 2, pp. 397–422, 2005.

[9] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D. Micheli, "Temperature-aware Processor Frequency Assignment for MP-SoCs using Convex Optimization," in *Proc. CODES*, 2008, pp. 111–116.

[10] V. Hanumaiah, S. Vrudhula, and K. S. Chatha, "Performance Optimal Speed Control of Multi-Core Processors under Thermal Constraints," in *Proc. DATE*, 2009, pp. 288–293.

[11] ——, "Maximizing Performance of Thermally Constrained Multi-core Processors by Dynamic Voltage and Frequency Control," in *Proc. ICCAD*, 2009, pp. 310–313.

[12] S. Heo, K. Barr, and K. Asanovic, "Reducing Power Density through Activity Migration," in *Proc. ISLPED*, 2003, pp. 217–222.

[13] R. Rao and S. Vrudhula, "Efficient Online Computation of Core Speeds to Maximize the Throughput of Thermally Constrained Multi-core Processors," in *Proc. ICCAD*, 2008, pp. 537–542.

[14] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, and S. Ghosh, "HotSpot: A Compact Thermal Modeling Method for CMOS VLSI Systems," *IEEE Trans. VLSI Sys.*, vol. 14, pp. 501–513, 2006.

[15] P. Michaud and Y. Sazeides, "ATMI: An Analytical Model of Temperature in Microprocessors," in *Proc. Workshop on Modeling, Benchmarking and Simulation (MOBS)*, 2007.

[16] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-level Power Analysis and Optimizations," in *Proc. Intl' Symp. Comp. Arch. (ISCA)*, 2000, pp. 83–94.

[17] W. Liao, L. He, and K. M. Lepak, "Temperature and Supply Voltage Aware Performance and Power Modeling at Microarchitecture Level," *IEEE Trans. Computer-Aided Design*, vol. 24, pp. 1042–1053, 2005.

[18] R. Mukherjee and S. O. Memik, "Physical Aware Frequency Selection for Dynamic Thermal Management in Multi-Core Systems," in *Proc. ICCAD*, 2006, pp. 547–552.

[19] T. Chantem, X. S. Hu, and R. P. Dick, "Online Work Maximization under a Peak Temperature Constraint," in *Proc. ISLPED*, 2009, pp. 105–110.

[20] H. Jung and M. Pedram, "Stochastic Dynamic Thermal Management: A Markovian Decision-based Approach," in *Proc. ICCD*, 2006, pp. 452–457.

[21] K. Stavrou and P. Trancoso, "Thermal-aware Scheduling for Future Chip Multiprocessors," *EURASIP J. Embedded Syst.*, vol. 2007, pp. 40–40, 2007.

[22] A. Coskun, T. Rosing, and K. Gross, "Proactive Temperature Balancing for Low Cost Thermal Management in MPSoCs," in *Proc. ICCAD*, 2008, pp. 250–257.

[23] A. Coskun, T. Rosing, K. Whisnant, and K. Gross, "Static and Dynamic Temperature-Aware Scheduling for Multiprocessor SoCs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, pp. 1127–1140, 2008.

[24] M. D. Powell, M. Gomaa, and T. N. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to Manage Power Density through the Operating System," in *Proc. ASPLOS*, 2004, pp. 260–270.

[25] F. Mulas, M. Pittau, M. Buttu, S. Carta, A. Acquaviva, L. Benini, and D. Atienza, "Thermal Balancing Policy for Streaming Computing on Multiprocessor Architectures," in *Proc. DATE*, 2008, pp. 734–739.

[26] X. Zhou, J. Yang, M. Chrobak, and Y. Zhang, "Performance-aware Thermal Management via Task Scheduling," *ACM Trans. Archit. Code Optim.*, vol. 7, pp. 5:1–5:31, 2010.

[27] S. Zhang and K. S. Chatha, "Thermal Aware Task Sequencing on Embedded Processors," in *Proc. DAC*, 2010, pp. 585–590.

[28] R. Jayaseelan and T. Mitra, "Temperature Aware Task Sequencing and Voltage Scaling," in *Proc. ICCAD*, 2008, pp. 618–623.

[29] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained Power Control for chip Multiprocessors with Online Model Estimation," *SIGARCH Comput. Archit. News*, vol. 37, pp. 314–324, 2009.

[30] J. Donald and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration," in *Proc. ISCA*, 2006.

[31] P. Michaud, A. Seznec, D. Fetis, Y. Sazeides, and T. Constantinou, "A Study of Thread Migration in Temperature-constrained Multicores," *ACM Trans. Arch. Code Opt.*, vol. 4, pp. 9–1–9–28, 2007.

[32] M. Hill and M. Marty, "Amdahl's Law in the Multicore Era," *Computer*, vol. 41, pp. 33–38, 2008.

[33] W. Huang, K. Sankaranarayanan, R. J. Ribando, M. R. Stan, and K. Skadron, "An Improved Block-based Thermal Model in Hotspot 4.0 with Granularity Considerations," in *Proc. WDDD*, 2007.

[34] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management," in *Proc. HPCA*, 2002, pp. 17–28.

[35] D. E. Kirk, *Optimal Control Theory*. Prentice-Hall, 1970.

[36] R. F. Hartl, S. P. Sethi, and R. G. Vickson, "A Survey of the Maximum Principles for Optimal Control Problems with State Constraints," *SIAM Rev.*, vol. 37, pp. 181–218, 1995.

[37] R. Rao and S. Vrudhula, "Fast and Accurate Prediction of the Steady State Throughput of Multi-core Processors under Thermal Constraints," *IEEE Trans. Computer-Aided Design*, vol. 28, pp. 1559–1572, 2009.

[38] P. M. DeRusso, R. J. Roy, C. M. Close, and A. A. Desrochers, *State Variables for Engineers*. Wiley-Interscience, 1997.

[39] M. Monchiero, R. Canal, and A. González, "Power/performance/thermal Design Space Exploration for Multicore Architectures," *IEEE Trans. Parallel and Distributed Sys.*, 2008.

[40] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, pp. 32–38, 1957.

[41] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. WWC*, 2001, pp. 3–14.

[42] V. Hanumaiah, R. Rao, and S. Vrudhula, "The MAGMA Thermal Simulator," http://vrudhula.lab.asu.edu/magma, Arizona State University.

[43] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.


**Vinay Hanumaiah** (S'08–S'11) received the B.Eng. degree in Electronics and Communication Engineering from M.S. Ramaiah Institute of Engineering, Bangalore, India, in 2005, the M.Tech. degree in Microelectronics from National Institute of Engineering, Karnataka, India, in 2007, and currently pursuing the Ph.D. degree in Electrical Engineering in Arizona State University.

His interests are in the thermal modeling and thermal-aware, performance and energy efficient operation of multi-core processors, 3-D ICs and data centers.

**Sarma Vrudhula** Sarma Vrudhula (M'85-SM'02) is a Professor in the department of Computer Science and Engineering at the Arizona State University, Tempe AZ, and the Director of the NSF Industry/Univeristy Coop-erative Research Center in Embedded Systems. He received the B.Math (Honors) from the University of Waterloo, Ontario, Canada, in 1976 and his M.S. and Ph.D degrees in electrical engineering from the University of Southern California in 1980 and 1985, respectively.

During 1985-1992 he was on the faculty of the EE-Systems department of the University of Southern California. From 1992 to 2005 he was a professor in the Electrical and Computer Engineering department at the University of Arizona, in Tucson, AZ., and the Director of the NSF UA/ASU Center for Low Power Electronics. His work spans several areas in design automation and computer aided design for digital integrated circuit and systems. His current research work covers statistical methods in the analysis of process variations, logic synthesis, threshold logic based design, low power circuit and system design, power, thermal and energy management for multicore processors, energy efficient multicore architectures, and novel nanoscale devices.

**Karam Chatha** (M'01) received the B.E. degree (with honors) in computer technology from Bombay University, Mumbai, India, in 1993 and the M.S and Ph.D. degrees in computer science and engineering from the University of Cincinnati, Cincinnati, OH, in 1997 and 2001, respectively.

He is currently an Associate Professor with the Faculty of Computer Science and Engineering at Arizona State University, Tempe. He was a visiting faculty at Qualcomm Corporate R&D in 2011. His research interests are in embedded systems with emphasis on system-level design of hardware and software. Specifically, he has focused on network-on-chip (NoC) design, parallel programming and compilation on embedded multi-core processors, low power and thermal aware design, multiprocessor system-on-chip (MPSoC) design, hardware-software co-design, and reconfigurable and adaptive computing. Dr. Chatha has served on the technical program committees of several international conferences, and has been a referee for national and international funding agencies.

Dr. Chatha is a recipient of the National Science Foundation CAREER Award in 2006. His publications were recognized by the Best Paper Awards at the International Conference on Computer Aided Design (ICCAD) 2007, and International Workshop on Field Programmable Logic (FPL) 1999. He is a member of the Association of Computing Machinery (ACM).