2016 International Symposium on System and Software Reliability

# Static Thermal-Aware Task Assignment and Scheduling for Makespan Minimization in Heterogeneous Real-time MPSoCs

Kun Cao, Junlong Zhou, Min Yin, Tongquan Wei, and Mingsong Chen

***Abstract*—In this paper, the authors address the problem of allocating and scheduling tasks of bag-of-tasks applications (BoTs) to multiprocessors for achieving makespan minimization under the thermal and timing constraints. The proposed scheme first selects the processor with highest allocation probability for every task. The allocation probability is calculated under the consideration of processor workload and temperature profiles. In addition, the higher allocation probability of a processor indicates the better performance in terms of makespan and temperature can be achieved by executing the task on this processor. Then, the operating frequencies of tasks are determined and tasks on the processor are executed in the alternate order of being hot-cool to reduce the on-chip peak temperature. Task splitting, that is, splitting a hot task into multiple sections and executing the hot subtasks with idle time alternatively, is also utilized to ensure the peak temperature constraint. Extensive simulations were performed to validate the effectiveness of the proposed approach. The proposed scheme achieves $15.31\%$ and $19.56\%$ reduction in makespan as compared to benchmarking scheme RATM and $\alpha$-VSTM, respectively. The peak temperature of the proposed algorithms can be up to $4.38\%$ and $4.49\%$ lower than that of benchmarking schemes, respectively.**

***Index Terms*—Makespan-Aware, Thermal Management, Task-to-Processor Allocation and Scheduling, Real-Time Tasks**

## I. Introduction

Due to the advance of technology scaling and ever increasing demand for performance, multiprocessors have replaced uniprocessors to become the main design paradigm for current and future processors. In the meantime, parallel processing approach that has the ability to carry out multiple operations or tasks simultaneously, are hence proposed to meet the rising computational requirements. However, parallel processing has to cope with a lot of problems not encountered in traditional sequential processing, especially for the issue of the utmost importance, the multiprocessor scheduling [1]. For multiprocessor scheduling, a typical target is to minimize the overall length of time required to execute the applications on processors, namely makespan, which has attracted considerable attention in recent years [2]–[5].

Assayad et al. [2] presented an off-line scheduling heuristic to jointly optimize task schedule length, system reliability, and power consumption. The proposed scheduling heuristic utilizes active replication to minimize the makespan and ensures the system reliability, and employs dynamic voltage and frequency scaling (DVFS) to lower the power consumption. Aupy et al. [3] addressed the problem of deciding which task to re-execute, and at which speed each execution of a task should be operated, in order to realize energy minimization under the constraints of a prescribed bound on the execution time (makespan) and a reliability threshold. Hanumaiah et al. [4] developed a temperature-aware DVFS-based approach for multi-core processors to optimize the makespan while satisfying timing and thermal constraints. Moreover, a theoretical basis and analytical relations between speed, voltage, power and temperature are provided. Li et al. [5] proposed a heuristic energy-aware stochastic task scheduling (ESTS) algorithm to jointly optimize the schedule length and energy consumption of heterogeneous computing systems.

As technology advances towards the deep submicron region, the chip power density has increased exponentially, which in turn leads to the elevated chip temperature. A system will fall into the predicament of functional incorrectness, low reliability and even permanent damage if the operating temperature exceeds a certain threshold. Industrial studies have shown that a small difference in operating temperature (10-15°C) can make two times difference in the device lifespan [6]. Thus, thermal management has been a significant and pressing research issue in computing systems, especially for embedded systems with limited cooling techniques. This necessitates the use of dynamic thermal management (DTM) techniques such as fetch toggling [7], DVFS [8], scheduling priority adjustments [9], and task migration [10], to achieve high-performance computing while maintaining the peak temperature of the chip below a specified temperature limit.

Unlike traditional thermal management methods, a new calibration-based approach [11] is proposed to estimate accurate temperature traces by building a compact thermal model without requiring any power-trace information, or the hard-to-obtain details about hardware, such as the detailed floorplan. Recently, a novel thermal management technique, that is, thermal-aware task sequencing combined with the dynamic voltage scaling, has attracted considerable research attention. It utilizes temperature characteristics of tasks to reduce the peak temperature of processors at a scaled operating frequency without incurring extra monetary expenses [12]–[14].

To the best of our knowledge, few existing scheduling techniques focusing on both homogeneous and heterogeneous multiprocessor systems jointly consider makespan minimization and thermal management. In other words, some work [15]–[18] only optimizes makespan and other goals without considering the thermal management while some work [19]–[21] considers the thermal management without optimizing makespan. Therefore, in this paper, we concentrate on designing a makespan- and thermal-aware static task scheduling scheme under the thermal and timing constraints. In our

K. Cao, J. Zhou, M. Yin, and T. Wei are with the Department of Computer Science and Technology, East China Normal University, Shanghai China. T. Wei is the corresponding author, email: tqwei@cs.ecnu.edu.cn.

M. Chen is with the Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai China.

978-1-5090-5563-0/16 $31.00 © 2016 IEEE
DOI 10.1109/ISSSR.2016.18

scheme, tasks are inclined to be assigned to processors with lighter workload and better temperature profiles. Real-time constraints of tasks are also checked during the task assignment. To ensure the system peak temperature is below a safe threshold, Several techniques such as frequency scaling, task sequencing and splitting are exploited in task scheduling to further improve the temperature profiles of processors.

The rest of the paper is organized as follows. Section II introduces the system architecture and models. Section III formulates the problem definition. Section IV describes the proposed task assignment and scheduling scheme. The effectiveness of the proposed scheme is verified by simulation in Section V and concluding remarks are given in Section VI.

## II. System Architecture and Models

### A. Architecture and Application Model

Consider a MPSoC system composed of $N$ heterogeneous processing elements $\boldsymbol{PE} = \{PE_1, PE_2, \cdots, PE_N\}$, where every processor $PE_k \in PE$ $(1 \leq k \leq N)$ is DVS-enabled and equipped with a set of discrete frequencies. The frequency set supported by processor $PE_k$ is denoted by $\{F_{k,1}, F_{k,2}, \cdots, F_{k,\iota}, \cdots, F_{k,\ell}\}$ $(1 \leq \iota \leq \ell)$, where $F_{k,min} = F_{k,1} \leq F_{k,2} \leq \cdots \leq F_{k,\ell} = F_{k,max}$ holds for the sake of easy presentation. It is assumed that every processor $PE_k$ supports one sleep mode and $\ell$ active modes that are characterized by their operating frequencies $F_{k,\iota}$. Tasks can be only executed in the active mode and the processor is idle when it is in sleep mode.

Bag-of-tasks applications (BoTs) are supposed to execute on the DVS-enabled MPSoC system. Such applications are considered to be independent and atomic, indicating no communication or precedence constraints among tasks. Due to this nature, BoTs can be highly parallelized and thus widely used in variety scenarios. Typical BoTs contain Monte Carlo simulations, tomographic reconstructions, parameter sweeps, computational biology and computer imaging [5], [22]. Consider a BoT application that consists of $\gamma$ independent tasks, which is denoted by $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_\gamma\}$. In such an application, tasks share a common deadline. The characteristic of every task $\tau_i$ $(1 \leq i \leq \gamma)$ is described by a triple $\tau_i : \{\mu_i, WC_i, D\}$, where $\mu_i$, $WC_i$, and $D$ are the activity factor, worst case execution cycles, and common deadline of $\tau_i$, respectively. Let $f_i$ be the operating frequency of task $\tau_i$, the worst case execution time of $\tau_i$ at $f_i$ is then given by $\frac{WC_i}{f_i}$.

### B. Power and Energy Model

The power consumption of a CMOS device can be modeled into two parts: (1) dynamic power dissipation, which is related to the operating frequency and switching activity, and (2) static power dissipation, which is independent of switching activity. Based on the system-level power model introduced in [23], the power consumed by processor $PE_k$ when executing task $\tau_i$ at frequency $F_{k,\iota}$ is given by

$$P_{i,k,\iota} = P_{k,\iota,s} + \delta \cdot P_{i,k,\iota,d}, \tag{1}$$

where $P_{k,\iota,s}$ and $P_{i,k,\iota,d}$ are the static and dynamic power consumption, respectively. Here a variable $\delta$, taking the value of either 0 or 1, is employed to represent processor modes and indicates whether the system is currently consuming active power. Specifically, $\delta = 1$ when the processor is in active mode and $\delta = 0$ when the processor is in sleep mode.

The static power is utilized to maintain the basic circuits of the system. It is temperature dependent and can only be eliminated by turning off the whole system. The static power consumption of processor $PE_k$ at supply voltage $V_{k,\iota}$ and operating frequency $F_{k,\iota}$ is

$$P_{k,\iota,s} = (\alpha_k + \beta_k T_{k,\iota}) V_{k,\iota}, \tag{2}$$

where $\alpha_k$ and $\beta_k$ are hardware constants of processor $PE_k$, and $T_{k,\iota}$ is the operating temperature.

The dynamic power is the dominant component of power consumption in a well designed circuit and mainly results from charging/discharging of gates in the circuits. It is temperature-independent and can be formulated as a function of supply voltage and operating frequency. The dynamic power consumption of processor $PE_k$ when executing task $\tau_i$ at operating frequency $F_{k,\iota}$ is formulated as

$$P_{i,k,\iota,d} = C_k^{eff} u_i V_{k,\iota}^2 F_{k,\iota}, \tag{3}$$

where $C_k^{eff}$ is the effective capacitance of processor $PE_k$.

### C. Temperature Model

An accurate and practical dynamic model of temperature is needed to accurately characterize the thermal behavior of a task. In this work, we assume that there is negligible or no heat transfer among processor units and among other different units [24]–[26]. Hence, a heat-independent thermal model proposed by Skadron et al. [27] that is widely used in the literature is adopted to predict the temperature of the processor. The model is based on a lumped $RC$ model, and the instantaneous temperature of task $\tau_i$ executing at $F_{k,\iota}$ on processor $PE_k$ is given by

$$T_{i,k,\iota}(t) = T_{std,i,k,\iota} - (T_{std,i,k,\iota} - T_{init,i}) e^{\frac{-t}{R_k C_k}}, \tag{4}$$

where $T_{std,i,k,\iota}$ is the steady state temperature of task $\tau_i$ and $T_{init,i}$ is the initial temperature. $R_k$ and $C_k$ are thermal resistance and capacitance of processor $PE_k$ respectively, which are both processor architecture dependent constants.

The steady state temperature of task $\tau_i$ is the temperature that will be reached if infinite number of instances of task $\tau_i$ execute continuously on the processor. It is associated with a certain input power, and is formulated as

$$T_{std,i,k,\iota} = P_{i,k,\iota} \times R_k + T_{amb}, \tag{5}$$

where $T_{amb}$ is the die's ambient temperature. Typically, a task $\tau_i$ is classified into hot or cool task category based on the steady state temperature.

## III. Problem Definition

In the concerned MPSoC system, processing elements are assumed to be equally distributed and tightly coupled, and the interunit communication is achieved via shared memories, such that the communication cost is negligible. Under this

assumption, the makespan can be defined as the overall length of time required to complete the tasks on processors. Let $t_f(PE_k)$ be the final completion time of tasks executing on processor $PE_k$, the makespan of the whole system is then the maximum of schedule length of processors, that is,

$$t_f = \max\{t_f(PE_k)|k \in [1, 2, \cdots, N]\}. \quad (6)$$

For the thermal-aware task scheduling, the peak temperature of tasks should be below a temperature limit (threshold) $T_{max}$ to avoid temperature-induced failures. The value of $T_{max}$ is in general specified based on system design requirements. Let $T_{peak}(PE_k)$ denote the peak temperature of processor $PE_k$, which is given by $T_{peak}(PE_k) = \max\{T(t)|\forall t \in [0, t_f(PE_k)]\}$. Here $T(t)$ is the instantaneous temperature during time interval $[0, t_f(PE_k)]$ and can be obtained by (4). Then let $T_{peak}$ denote the on-chip peak temperature, which can be calculated as

$$T_{peak} = \max\{T_{peak}(PE_k)|k \in [1, 2, \cdots, N]\}. \quad (7)$$

The system is deemed to be safe when the peak temperature $T_{peak}$ does not exceed the threshold temperature $T_{max}$. In addition to the temperature constraint, all tasks in an application should finish their execution before the common deadline $D$. Considering the above design constraints, the task allocation and scheduling problem to maximize the makespan of a N-processor system can be formulated into the below form.

$$\begin{aligned} \text{Minimize: } & t_f = \max_{1 \le k \le N} t_f(PE_k) \\ \text{Subject to: } & T_{peak} \le T_{max} \\ & t_f \le D, \end{aligned}$$

where $t_f$ and $T_{peak}$ are given in (6) and (7), respectively.

## IV. Task Allocation and Scheduling Scheme

The proposed task allocation and scheduling scheme aims to optimize the makespan under the peak temperature and deadline constraints for the target N-processor system. The workflow of the proposed scheme is shown in Fig. 1. As shown in the figure, a global scheduler accepts as input the BoT-based real-time tasks, and generates a makespan- and thermal-aware task-to-processor assignment as output. The assigned tasks are stored in the local queue of individual processors. The local scheduler then takes as input the tasks in local queues and determines the scheduling of tasks on the processor. More specifically, during the task assignment, the global scheduler takes into account the schedule length and temperature profiles of processors and checks the deadline constraints of tasks, such that the makespan of the system could be reduced under the thermal and timing constraints. During the task scheduling, the local scheduler utilizes multiple techniques such as frequency scaling, task sequencing and splitting to further improve the system temperature profiles without increasing system makespan. The details of our methods are presented in the following subsections below.

### A. Task-to-Processor Assignment

The $\gamma$ tasks of a BoT are accepted by the global queue $Q_{global}$ and would be assigned to processors by the global

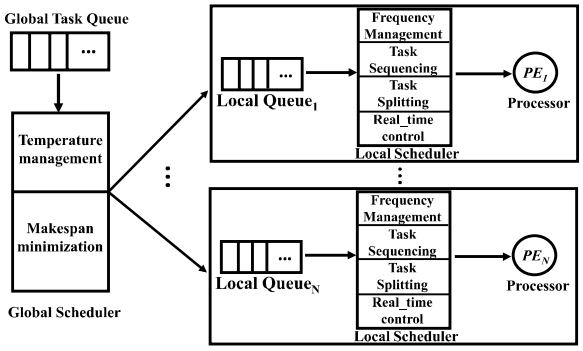



Fig. 1: The workflow of the proposed scheme.

scheduler. During the assignment, schedule length and temperature profiles of processors are taken into account. The assignment of every task is carried out under the principle that the higher the peak temperature and the larger the schedule length of the processor, the lower probability of assigning the task to the processor. To be specific, it operates as follows. For every task in the global queue, the probabilities of assigning the task to processors are first calculated based on the schedule length and thermal profiles of processors. After the assignment probabilities of all processors are derived, the processor with highest probability is chose to execute the task. The calculation of assignment probability is described below in detail.

Since the probability of assigning task $\tau_i$ to processor $PE_k$ depends on both the thermal profiles and schedule length of the processor, a variable $\alpha$ is introduced to balance the weights of two factors. Let $\lambda_k$ represent the probability of assigning task $\tau_i$ to processor $PE_k$, and it is given by

$$\lambda_k = \alpha \times \lambda_{k,t} + (1 - \alpha) \times \lambda_{k,w}, \quad (8)$$

where $\lambda_{k,t}$ is the probability decided by thermal profiles, whereas $\lambda_{k,w}$ is the probability determined by schedule length.

We select the ending temperature of a task to characterize its thermal profiles since the peak temperature of a task execution is reached at its start/ending time instant [21] and the ending time instant is also the start time instant of its successor task. Therefore, in our task assignment scheme, task $\tau_i$ is inclined to be assigned to the processor with lower ending temperature. In addition, the task is not permitted to be assigned to the processor violating the temperature constraint $T_{max}$. Given these, the probability $\lambda_{k,t}$ decided by thermal profiles of processor $PE_k$ is derived as

$$\lambda_{k,t} = \begin{cases} 1 - \frac{T_e}{T_{max}} & T_e < T_{max} \\ 0 & otherwise \end{cases}, \quad (9)$$

where $T_e$ is ending temperature of task $\tau_i$. It is clear from (9) that the higher the ending temperature $T_e$, the smaller the assignment probability $\lambda_{k,t}$.

Let $\mathcal{L}_k$ denote the schedule length (in cycles) of local queue $Q_{local}$ of processor $PE_k$, which is given by

$$\mathcal{L}_k = \begin{cases} \mathcal{L}_k + WC_i & \tau_i \in PE_k \\ \mathcal{L}_k & otherwise \end{cases}. \quad (10)$$

At the begin of task assignment, the value of $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_N$

are all initialized to 0 and would be updated when tasks are assigned to processors by $\mathcal{L}_k = \mathcal{L}_k + WC_i$. Moreover, a higher $\mathcal{L}_k$ indicates that more tasks are on the processor and need to be executed, leading to a higher makespan of the system. Therefore, considering the goal of minimizing the system makespan, the probability $\lambda_{k,w}$ decided by schedule length of processor $PE_k$ is defined as

$$\lambda_{k,w} = 1 - \frac{\frac{\mathcal{L}_k}{F_{k,max}}}{\sum_{k=1}^{N} \frac{\mathcal{L}_k}{F_{k,max}}}. \quad (11)$$

As shown above, the higher the schedule length $\mathcal{L}_k$ the smaller the assignment probability $\lambda_{k,w}$.

Algorithm 1 describes pseudo-code of our task-to-processor assignment scheme under the peak temperature and deadline constraints. It takes as input the weighting parameter $\alpha$, peak temperature limit $T_{max}$, and $\gamma$ tasks $\tau_1, \tau_2, \cdots, \tau_\gamma$ in global queue $Q_{global}$. The algorithm operates as follows. Based on the weighting parameter $\alpha$, the most suitable processor for tasks are selected using procedure *Select* (line 1). Line 2 determines the operating frequencies of tasks assigned to processors using Algorithm 2 and line 3 improves the thermal profiles of tasks using Algorithm 3. If the ending temperature $T_e$ of task $\tau_i$ exceeds the temperature limit $T_{max}$, Algorithm 4 is called to further reduce temperature by splitting the task that violates thermal constraint into multiple sections and executing them alternatively (lines 6-8). Lines 10-13 of the algorithm check whether the makespan $t_f$ exceeds the common deadline $D$.

The selection of the most suitable processor in terms of minimizing makespan and improving temperature profiles for each task is implemented by procedure *Select* (lines 14-28). The input of the procedure is the weighting parameter $\alpha$. In each round of the iteration, lines 15-16 calculate the probability $\lambda_1(\tau_j)$ of allocating task $\tau_j$ to processor $PE_1$ using (8), and initialize $\lambda_{max}$ and $flag$ to $\lambda_1(\tau_j)$ and 1, respectively. In lines 17-24, the allocation probability $\lambda_k(\tau_j)$ of task $\tau_j$ to processor $PE_k$ for $2 \leq k \leq N$ is derived, and the processor $PE_{flag}$ that generates the maximal allocation probability $\lambda_{max}$ is selected. Then task $\tau_j$ is assigned to the selected processor $PE_{flag}$, and the global queue $Q_{global}$, local queue $Q_{local,flag}$, and schedule length $\mathcal{L}_{flag}$ of processor $PE_{flag}$ are hence updated (lines 25-27). Repeat this process until all the $\gamma$ tasks are all assigned to processors.

**Algorithm 1** The proposed tasks allocation scheme

**Require:** Weighting parameter $\alpha$ & peak temperature limit $T_{max}$ & tasks $\tau_1, \tau_2, \cdots, \tau_\gamma$ in global queue $Q_{global}$
1: assign the $\gamma$ tasks in $Q_{global}$ to $N$ processors using *Select*;
2: determine the operating frequencies of tasks in $N$ store queues using Algorithm 2;
3: improve the task thermal profiles using Algorithm 3;
4: **for** $i \leftarrow 1$ to $\gamma$ **do**
5: derive the ending temperature $T_e$ of $\tau_i$ using (4);
6: **if** $T_e > T_{max}$ **then**
7: call Algorithm 4 to reduce temperature of task $\tau_i$; // Split a task into multiple subtasks and execute them alternatively.
8: **end if**
9: **end for**
10: derive the makespan $t_f$ of the system using (6);
11: **if** $t_f \geq D$ **then**
12: **exit(1);** // The task set can't be feasibly scheduled.
13: **end if**

**Procedure** *Select*
**Require:** $\alpha$
14: **for** $j \leftarrow 1$ to $\gamma$ **do**
15: computes the probability $\lambda_1(\tau_j)$ of allocating task $\tau_j$ to processor $PE_1$ using (8);
16: $\lambda_{max} \leftarrow \lambda_1(\tau_j)$, $flag \leftarrow 1$; // $flag$ indicates the index of processor that generates the maximal probability $\lambda_{max}$.
17: **for** $k \leftarrow 2$ to $N$ **do**
18: derive the probability $\lambda_{k,t}(\tau_j)$ decided by processor thermal profiles using (9);
19: obtain the probability $\lambda_{k,w}(\tau_j)$ decided by processor schedule length using (11);
20: calculate the probability $\lambda_k(\tau_j)$ using (8) based on probability $\lambda_{k,t}(\tau_j)$ and $\lambda_{k,w}(\tau_j)$;
21: **if** $\lambda_{max} \leq \lambda_k(\tau_j)$ **then**
22: $\lambda_{max} \leftarrow \lambda_k(\tau_j)$, $flag \leftarrow k$;
23: **end if**
24: **end for**
25: assign the task $\tau_j$ to processor $PE_{flag}$;
26: update the global queue by $Q_{global} \leftarrow Q_{global} - \tau_j$;
27: update the local queue and schedule length by $Q_{local,flag} \leftarrow Q_{local,flag} + \tau_j$ and $\mathcal{L}_{flag} \leftarrow \mathcal{L}_{flag} + WC_j$, respectively;
28: **end for**

### B. Determine the Operating Frequencies of Tasks

Due to early completion of tasks, slack is generated during the execution of tasks and it can be used to reduce the on-chip peak temperature by scaling the operating frequencies. The tasks can be classified into hot or cool task category based on their steady state temperatures. If the steady state temperature of a task is more than the maximal temperature limit, the task is deemed to be a hot task; otherwise, it is a cool task. Considering the bad temperature characteristics of hot tasks, Algorithm 2 scales the operating frequencies of hot tasks to achieve a lower on-chip peak temperature without increasing the system makespan.

Inputs to Algorithm 2 are the $N$ local queues $\{Q_{local,1}, Q_{local,2}, \cdots, Q_{local,N}\}$ and maximal temperature limit $T_{max}$. Lines 1-2 of the algorithm calculate the final completion time $t_f(PE_k)$ of $N$ processors when supposing all the tasks in the $N$ local queues are executed at the maximal frequency $F_{k,\ell}$, and derive the system makespan $t_f$ using (6). Lines 3-22 determine the operating frequency of all tasks. In each round of the iteration, the available slack time $sl_k$ of processor $PE_k$ is obtained using $sl_k \leftarrow t_f - t_f(PE_k)$ (lines 4). Lines 5-12 classify the task $\tau_j$ into hot or cool task based on its steady state temperature $T_{std}(\tau_j)$. If $\tau_j$ is a hot task, it will be inserted into hot task queue $Q_{k,h}$; otherwise, it will be inserted into cool task queue $Q_{k,c}$. Line 13 sets the operating frequencies of cool tasks in $Q_{k,c}$ to $F_{k,\ell}$. For the non-empty hot queue $Q_{k,h}$, all tasks in $Q_{k,h}$ are sorted in descending order of $T_{tsd}$ and their operating frequencies are initialized to $F_{k,\ell}$ (lines 14-16). For each hot task in $Q_{k,h}$, its final operating frequency is determined by procedure *Scale* (lines 17-19). Line 21 updates the two queues $Q_{k,h}$ and $Q_{k,c}$.

Lines 23-35 describe the scaling of operating frequency of hot tasks in $Q_{k,h}$. The inputs of procedure *Scale* are hot task $\tau_i$ and the available slack $sl_k$ of processor $PE_k$. $\iota$ is used

**Algorithm 2** Decide the operating frequencies of tasks

**Require:** Local queues $\{Q_{local,1}, Q_{local,2}, \cdots, Q_{local,N}\}$ & the temperature limit $T_{max}$;
1: derive the final completion time $t_f(PE_k)$ for $1 \leq k \leq N$ when assuming tasks in local queues are executed at $F_{k,\ell}$;
2: obtain the makespan $t_f$ using (6);
3: **for** $k \leftarrow 1$ to $N$ **do**
4: calculate the available slack $sl_k$ of processor $PE_k$ using $sl_k \leftarrow t_f - t_f(PE_k)$;
5: **for** $j \leftarrow 1$ to $sizeof(Q_{local,k})$ **do**
6: compute the $T_{std}(\tau_j)$ using (5);
7: **if** $T_{std}(\tau_j) \geq T_{max}$ **then**
8: insert $\tau_j$ into $Q_{k,h}$; // Insert hot task into $Q_{k,h}$.
9: **else**
10: insert $\tau_j$ into $Q_{k,c}$; // Insert cool task into $Q_{k,c}$.
11: **end if**
12: **end for**
13: set the operating frequency of tasks in $Q_{k,c}$ to $F_{k,\ell}$;
14: **if** ($Q_{k,h} \neq$NULL) **then**
15: sort tasks in $Q_{k,h}$ in descending order of $T_{std}$;
16: set the operating frequencies of hot tasks in $Q_{k,h}$ to $F_{k,\ell}$;
17: **for** $i \leftarrow 1$ to $sizeof(Q_{k,h})$ **do**
18: call *Scale*;
19: **end for**
20: **end if**
21: update the $Q_{k,h}$ and $Q_{k,c}$;
22: **end for**

**Procedure** *Scale*
**Require:** the hot task $\tau_i$ & the available slack time $sl_k$
23: $\iota = \ell$; // The highest frequency of processor $PE_k$ is $F_{k,\ell}$.
24: **while** $sl_k > 0$ **do**
25: **if** $\frac{WC_i}{F_{k,\iota-1}} - \frac{WC_i}{F_{k,\iota}} \leq sl_k$ **then**
26: $f_i \leftarrow F_{k,\iota-1}$;
27: $sl_k \leftarrow sl_k - (\frac{WC_i}{F_{k,\iota-1}} - \frac{WC_i}{F_{k,\iota}})$, $\iota \leftarrow \iota - 1$;
28: **if** $\iota == 1$ **then**
29: **break**;
30: **end if**
31: **else**
32: insert the residual slack as an idle task into $Q_{k,c}$ and set its operating frequency to $F_{k,\ell}$;
33: **break**;
34: **end if**
35: **end while**

to represent the $\iota th$ active model of processor $PE_k$ and is initialized to $\ell$ (line 23). Then, the frequency $F_{k,\iota}$ is lowered in steps by exploiting the available slack until $sl_k = 0$ (lines 24-35). In each round of the iteration, line 25 checks whether the $sl_k$ is sufficient to lower the task operating frequency from $F_{k,\iota}$ to $F_{k,\iota-1}$. If the answer is yes, the task operating frequency $f_i$ is scaled to $F_{k,\iota-1}$ (line 26). $sl_k$ and $\iota$ are also updated (line 27). Note that, when the frequency is scaled to the lowest frequency, the iteration exits (lines 28-30). However, if the slack is not enough, the iteration terminates, and the residual slack is treated as an idle task and inserted into the $Q_{k,c}$ with operating frequency $F_{k,\ell}$ (lines 31-34).

### C. Improve the Thermal Profiles of Hot Tasks

Unlike traditional cooling solutions, temperature-aware task sequencing utilizes thermal characteristics of tasks to reduce their peak temperature without degrading system performance and incurring extra monetary expenses. The task sequencing heuristic is based on the observation that the execution order of a hot task and a cool task has non-negligible impact on the peak temperature. Moreover, it has been proven in [28] that the final temperature of tasks executing in the hot-cool order is lower than that of tasks executing in the cool-hot order. Therefore, we utilize task sequencing in Algorithm 3 to promote the thermal profiles of hot tasks by executing their tasks in the alternate order of being hot-cool. Compared to the regulation approach presented in [28], the proposed scheme doesnot necessitate the deployment of a runtime temperature monitor and thermal sensors.

**Algorithm 3** Promote the temperature profiles of hot tasks using task sequencing

**Require:** Hot/cool task queue $Q_{k,h}/Q_{k,c}$ of processors;
1: **for** $k \leftarrow 1$ to $N$ **do**
2: sort tasks in $Q_{k,h}$ in descending order of $T_{std}$;
3: sort tasks in $Q_{k,c}$ in ascending order of $T_{std}$;
4: $size_{min} \leftarrow min(sizeof(Q_{h,k}), sizeof(Q_{c,k}))$;
5: **for** $i \leftarrow 1$ to $size_{min}$ **do**
6: push $Q_{k,h}[i]$ into $Q_k[2 * i - 1]$;
7: delete $Q_{k,c}[i]$ in $Q_{k,h}$;
8: push $Q_{k,c}[i]$ into $Q_k[2 * i]$;
9: delete $Q_{k,c}[i]$ in $Q_{k,c}$;
10: **end for**
11: push residual tasks in $Q_{k,h}$ or $Q_{k,c}$ into $Q_k$;
12: **end for**

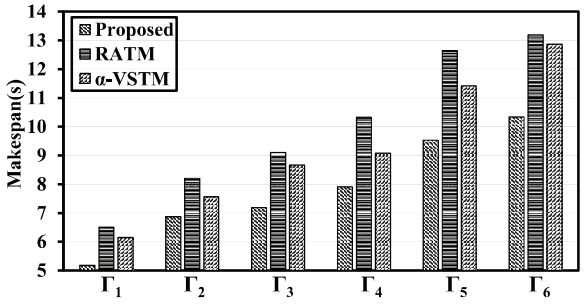
Algorithm 3 operates as follows. It takes as input the hot task queue $Q_{k,h}$ and cool tasks $Q_{k,c}$ of $N$ processors. For each processor $PE_k$, lines 1-3 of the algorithm sort the hot tasks of $Q_{k,h}$ in descending order of $T_{std}$ and the cool tasks of $Q_{k,c}$ in ascending order of $T_{std}$. $size_{min}$ is utilized to represent the minimum size of hot queue and cool queue (line 4). Lines 5-10 alternately push the hot tasks in $Q_{k,h}$ and cool tasks in $Q_{k,c}$ into the target queue $Q_k$. In each round of the iteration, when a task is pushed into $Q_k$, it is deleted from its original queue $Q_{k,h}$ or $Q_{k,c}$. Obviously, the task sequence formed in the target queue $Q_k$ is hot-cool. The tasks in the non-empty queue are appended to the tail of the target queue (line 11).

**Algorithm 4** Reduce task peak temperature by task splitting

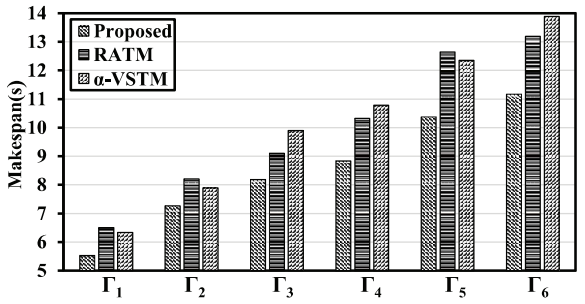
**Require:** Task $\tau_i$ & Context switching overhead $t_{sw,i}$ of $\tau_i$
1: $t_{idle,i}^{opt} \leftarrow t_{sw,i}$; // Set the initial value of optimal idle time equal to context switching overhead $t_{sw,i}$.
2: $T_e(t_{idle,i}) \leftarrow T_{amb} - (T_{amb} - T_{init,i})e^{-\frac{t_{sw}}{R_k C_k}}$; // Calculate the ending temperature of the inserted idle time interval.
3: $t_{active,i} \leftarrow -\frac{1}{B_{k,i}} ln(\frac{T_{init,i} - T_{std,i}}{T_e(t_{idle,i}) - Tstd,i})$, where $B_{k,i} \leftarrow \frac{1}{R_k C_k} - \frac{1}{C_k}\beta_{k,\iota} V_{k,\iota}$ and $\beta_{k,\iota}$ is a constant depending on processor $PE_k$; // Compute the execution time of subtasks in active mode.
4: $m_i^{opt} \leftarrow \lfloor \frac{WC_i}{f_i t_{active,i}} \rfloor$; // Derive the optimal split number of $\tau_i$.
5: $t_{idle,i}^{opt} \leftarrow -R_k C_k ln(\frac{T_e(t_{sw,i})}{T_{init,i}})$; // Obtain the optimal idle time.

### D. Reduce the Task Peak Temperature by Task Splitting

Although Algorithm 2 scales the operating frequencies of hot tasks and Algorithm 3 pairs the task execution in the order of hot-cool, the temperature constraint may be still violated due to the bad thermal profiles. To overcome the predicament,




(a) $\alpha = 0.3$




(b) $\alpha = 0.8$

Fig. 2: The makespan of six task sets achieved by benchmarking schemes RATM, $\alpha-$VSTM, and the proposed algorithm.

two effective thermal management techniques, that is, sleep mode distribution and task splitting, can be adopted. The former is to put the processor into sleep mode for cooling the task execution, while the latter is to split a hot task into multiple sections and execute the hot subtasks with idle time alternatively. As shown in [13], under the same peak temperature constraint, the extra time cost incurred by task splitting is less then that of sleep mode distribution. Therefore, task splitting is more suitable for our problem of minimizing makespan under the peak temperature and real-time constraint, and is adopted in our algorithm.

Algorithm 4 describes how to split a task into multiple subtasks and insert idle time. The algorithm takes as input the task $\tau_i$ that needs to be split and the context switching overhead $t_{sw,i}$. Line 1 of the algorithm initializes the optimal idle time $t^{opt}_{idle,i}$ that follows the execution of hot subtasks to the context switching time $t_{sw,i}$. Line 2 calculates the ending temperature $T_e(t_{idle,i})$ of the inserted idle time and line 3 computes the execution time $t_{active,i}$ of subtasks in active mode. Finally, after obtaining $T_e(t_{idle,i})$ and $t_{active,i}$, the optimal split number $m^{opt}_i$ of task $\tau_i$ and the optimal idle time $t^{opt}_{idle,i}$ for each hot subtask are derived (lines 4-5). More details of task splitting can be found in [13].

## V. Numerical Results

### A. Experimental Settings

A simulated platform is modeled based on the embedded multicore architecture resembling ARM Cortex A15 [29], comprising of 4 cores, 40-bit large physical address extensions, a 15 stage pipeline, 32KB instruction and data caches. In the simulations, benchmarking tasks of BoTs are generated by a task generator. The worst case execution cycles of every task are randomly generated in the range of $[4 \times 10^7, 6 \times 10^8]$ [12]. The task activity factors are uniformly distributed in the interval $[0.4, 1]$, which demonstrates the heterogeneous nature of tasks [13]. Six task sets $\Gamma_1 - \Gamma_6$ are created in this way and the size of every task set is increased from 100 to 200 with a step size of 20. The ambient temperature $T_{amb}$ is set to be $25°\mathrm{C}$ and the peak temperature limit $T_{max}$ is set to $50°\mathrm{C}$. The proposed algorithms were implemented in C++, and the simulation was performed on a machine with Intel Dual-Core 3.0 GHz processor and 4GB memory.

### B. Comparison of the Makespan

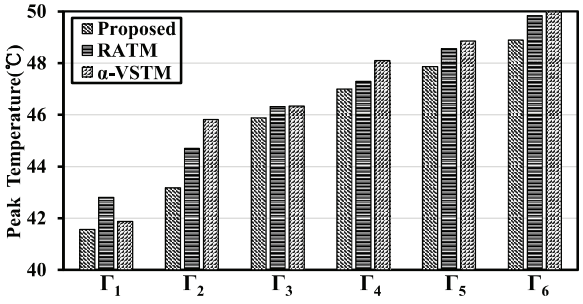
Two sets of simulation experiments have been carried out to validate the proposed algorithm. In the first set of simulations, two benchmarking schemes, referred to as random assignment with thermal management (RATM) and $\alpha-$ voltage scaling with thermal management ($\alpha-$VSTM), are compared with the proposed algorithm to verify its effectiveness in terms of minimizing makespan when $\alpha = 0.3$. RATM randomly assigns the tasks to processors so that it ignores the workload and thermal characteristics of processors during the assignment. After task-to-processor assignment, RATM adopts the techniques of frequency scaling, task sequencing and splitting to improve the system temperature profiles as the proposed algorithm. $\alpha-$VSTM adopts the proposed task-to-processor assignment, but applies the dynamic voltage scaling to thermal management rather than the combination of frequency scaling, task sequencing and splitting. In the second set of simulations, the proposed algorithm are compared with RATM and $\alpha-$VSTM under the setting of $\alpha = 0.8$.

Fig. 2 demonstrates the makespan of six task sets achieved by the proposed algorithm, RATM, and $\alpha$-VSTM under the thermal constraint of $T_{max} = 50°\mathrm{C}$. As shown in the figure, the makespan achieved by the proposed algorithm is much lower than that of the benchmarking scheme RATM and $\alpha$-VSTM for the six task sets under test. The conclusion can be drawn in the both case of $\alpha = 0.3$ and $\alpha = 0.8$. For instance, compared to RATM and $\alpha$-VSTM, the proposed algorithm reduces the makespan by $21.62\%$ and $19.63\%$ for task set $\Gamma_6$ when $\alpha = 0.3$, respectively. When $\alpha = 0.8$, the proposed algorithm achieves $15.31\%$ and $19.56\%$ makespan reduction as compared to RATM and $\alpha$-VSTM, respectively.

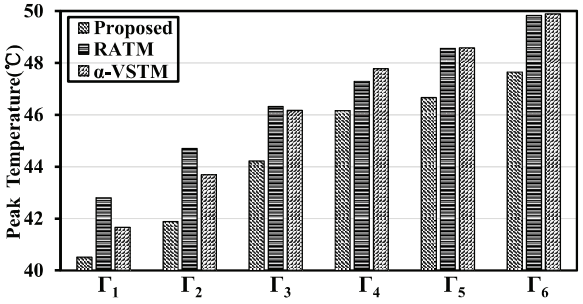
In addition, as expected, the makespan of the proposed scheme is larger as $\alpha$ becomes higher. For example, the makespan of task set $\Gamma_3$ is 7.1808s when $\alpha = 0.3$ but is 8.1875s when $\alpha = 0.8$. This is because the proposed scheme puts less weights on the processor workload as the increase in $\alpha$. Hence, the scheduler can adjust the $\alpha$ to meet varying system design requirements.

### C. Comparison of the Peak Temperature

The proposed algorithm is also compared with two benchmarking approaches RATM and $\alpha$-VSTM in terms of peak




(a) $\alpha = 0.3$




(b) $\alpha = 0.8$

Fig. 3: The peak temperature of six task sets achieved by benchmarking schemes RATM, $\alpha-$VSTM, and the proposed algorithm.

temperature to demonstrate its effectiveness of thermal management. The proposed scheme improves the temperature profiles of tasks by scaling the operating frequency for hot tasks and alternating the execution of hot tasks and cool tasks. Moreover, task splitting is exploited to prevent the peak temperature from exceeding the temperature limit $T_{max}$.

Fig. 3 plots the peak temperature of six task sets achieved by two benchmarking scheme RATM, $\alpha$-VSTM, and the proposed algorithm when $\alpha = 0.3$ and $\alpha = 0.8$. It has been shown in the figure that the proposed algorithm outperforms the two benchmarking schemes in terms of reducing peak temperature for different task sets under test. For instance, in the case of $\alpha = 0.3$ for task set $\Gamma_6$, the proposed algorithm reduces the peak temperature by $1.89\%$ and $2.19\%$ as compared to RATM and $\alpha$-VSTM, respectively. In addition, when $\alpha = 0.8$, the proposed algorithm achieves $4.38\%$ and $4.49\%$ reduction in peak temperature as compared to RATM and $\alpha$-VSTM, respectively. As expected, the proposed algorithm has a lower peak temperature as $\alpha$ becomes higher. This is because that the proposed scheme puts more weights on the processor thermal profiles as the increase in $\alpha$. Again, the scheduler can adjust the $\alpha$ to meet varying system design requirements.

## VI. Conclusion

In this paper, the authors propose a task assignment and scheduling scheme that minimizes the execution time and smoothes the chip temperature. The proposed scheme first assigns the tasks to processors in a way that leads to reduced processors' schedule length and improved processor temperature profiles. It classifies the tasks into hot and cool task category. The cool tasks are executed at the maximal frequencies supported by processors while the hot tasks are executed at scaled frequencies. It then executes tasks on the processor in the alternate order of being hot-cool for achieving better thermal profiles. It finally utilizes task splitting to ensure the peak temperature constraint is met. Extensive simulations have been performed to validate the effectiveness of the proposed scheme. Simulation results show that the proposed scheme can achieve $15.31\%$ and $19.56\%$ reduction in makespan when compared to benchmarking scheme RATM and $\alpha$-VSTM, respectively. Moreover, the peak temperature of the proposed scheme can be up to $4.38\%$ and $4.49\%$ lower than that of benchmarking schemes, respectively.

## Acknowledgments

This work was partially supported by Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000), Natural Science Foundation of China (Grant No. 91418203 and 61672230), and East China Normal University Outstanding Doctoral Dissertation Cultivation Plan of Action (Grant No. PY2015047).

## References

[1] D. Culler, J. Singh, and A. Gupta, "Parallel computer architecture: A hardware/software approach," 1999.

[2] I. Assayad, A. Girault, and H. Kalla, "Tradeoff exploration between reliability power consumption and execution time," *The Proceedings of the International Conference on Computer Safety, Reliability and Security*, pp. 437–451, 2011.

[3] G. Aupy, A. Benoit, and Y. Robert, "Energy-aware scheduling under reliability and makespan constraints," *The Proceedings of the International Conference on High Performance Computing*, pp. 1–10, 2012.

[4] V. Hanumaiah and S. Vrudhula, "Temperature-aware dvfs for hard real-time applications on multicore processors," *IEEE Transactions on Computers*, vol. 61, no. 10, pp. 1484–1494, 2012.

[5] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogenous computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2867–2876, 2014.

[6] R. Viswanath, W. Vijay, A. Watwe, and V. Lebonheur, "Thermal performance challenges from silicon to systems," *Intel Technology Journal*, vol. 4, no. 3, pp. 1–16, 2000.

[7] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," *The Proceedings of the International Symposium on High-Performance Computer Architecture*, pp. 171–182, 2001.

[8] K. Skadron, "Hybrid architectural dynamic thermal management," *The Proceedings of the International Conference on Design, Automation and Test in Europe*, pp. 10–15, 2004.

[9] A. Kumar, S. Li, L. Peh, and N. Jha, "Hybdtm: a coordinated hardware-software approach for dynamic thermal management," *The Proceedings of the International Conference on Design Automation*, pp. 548–553, 2006.

[10] Y. Ge, P. Malani, and Q. Qiu, "Distributed task migration for thermal management in many-core systems," *The Proceedings of the International Conference on Design Automation*, pp. 579–584, 2010.

[11] D. Rai, H. Yang, I. Bacivarov, and L. Thiele, "Power agnostic technique for efficient temperature estimation of multicore embedded systems," *The Proceedings of the International Conference on Compliers, Architectures and Synthesis for Embedded Systems*, pp. 61–70, 2012.

[12] R. Jayaseelan and T. Mitra, "Temperature aware task sequencing and voltage scaling," *The Proceedings of the International Conference on Computer-Aided Design*, pp. 618–623, 2008.

[13] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint," *ACM transactions on Embedded Computing Systems*, vol. 13, no. 2s, pp. 1–22, 2014.

[14] S. Zhang and K. Chatha, "Thermal aware task sequencing on embedded processors," *The Proceedings of the International Conference on Design Automation*, pp. 585–590, 2010.
[15] J. Dongarra, E. Jeannot, E. Saule, and Z. Shi, "Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems," *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pp. 280–288, 2007.
[16] X. Wang, C. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1124–1134, 2011.
[17] C. Rajendran and H. Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs," *European Journal of Operational Research*, vol. 155, no. 2, pp. 426–438, 2004.
[18] S. Shakya and U. Prajapati, "Task scheduling in grid computing using genetic algorithm," *Proceedings of the 2015 International Conference on Green Computing and Internet of Things*, pp. 1245–1248, 2015.
[19] N. Fisher, J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling and analysis on multicore systemse," *Journal of Systems Architecture*, vol. 57, no. 5, pp. 547–560, 2011.
[20] S. Sharifi, Y. Wu, and T. Rosing, "Temperature-aware scheduling for embedded heterogeneous mpsocs with special purpose ip cores," *Proceedings of 20th International Conference on Computer Communications and Networks*, pp. 1–6, 2011.
[21] J. Zhou, T. Wei, M. Chen, J. Yan, S. Hu, and Y. Ma, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015.
[22] W. Cirne, F. Brasileiro, J. Sauve, N. Andrade, D. Paranhos, E. Santos-Neto, and R. Medeiros, "Grid computing for bag of tasks applications," *Proceedings of the 3rd IFIP Conference on E-Commerce, E-Business and E-Government*, 2003.
[23] B. Zhao, H. Aydin, and D. Zhu, "Enhanced reliability-aware power management through shared recovery technique," *The Proceedings of the International Conference on Computer-Aided Design*, pp. 63–70, 2009.
[24] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature constraint hard real-time periodic tasks," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 329–339, 2010.
[25] S. Saha, Y. Lu, and J. Deogun, "Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems," *In Proc. Int. Conf. Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 41–50, 2012.
[26] J. Zhou and T. Wei, "Stochastic thermal-aware real-time task scheduling with considerations of soft errors," *Journal of Systems and Software*, vol. 102, pp. 123–133, 2015.
[27] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, 2004.
[28] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, "Dynamic thermal management through task scheduling," *The Proceedings of the International Symposium on Performance Analysis of Systems and Software*, pp. 191–201, 2008.
[29] ARM Cortex A15 Processor. [Online]. Available: http://www.arm.com/zh/products/processors/cortex-a/cortex-a15.php