J Sign Process Syst (2016) 84:111–121
DOI 10.1007/s11265-015-0994-4

Peak Temperature Minimization via Task Allocation and Splitting for Heterogeneous MPSoC Real-Time Systems

Junlong Zhou · Jianming Yan · Jing Chen · Tongquan Wei

Received: 30 November 2014 / Revised: 27 February 2015 / Accepted: 6 March 2015 / Published online: 9 April 2015

Abstract With the continued scaling of the CMOS devices, the exponential increase in power density has strikingly elevated the temperature of on-chip systems. Thus, thermal-aware design has become a pressing research issue in computing system, especially for real-time embedded systems with limited cooling techniques. In this paper, the authors formulate the thermal-aware real-time multiprocessor system-on-chip (MPSoC) task allocation and scheduling problem, present a task-to-processor assignment heuristics that improves the thermal profiles of tasks, and propose a task splitting policy that reduces the on-chip peak temperature. The thermal profiles of tasks are improved via task mapping by minimizing task steady state temperatures, and the task splitting technique is applied to reduce the peak temperature by enabling the alternation of hot task execution and slack time. The proposed algorithms explicitly exploits thermal characteristics of both tasks and processors to minimize the peak temperature without incurring significant overheads. Extensive simulations of benchmarking tasks were performed to validate the effectiveness of the proposed algorithms. Experimental results have shown that the task steady state temperature achieved by the proposed algorithm is 3.57 °C lower on average as compared to the benchmarking schemes, and the peak temperature of the proposed algorithm can be up to 11.5 % lower than that of the benchmarking schemes

Keywords Thermal-aware · Task allocation and scheduling · Task splitting · MPSoC real-time systems

J. Zhou · J. Yan · T. Wei (✉)
Shanghai Key Laboratory of Multidimensional Information Processing, Shanghai, 200241, China
e-mail: tqwei@cs.ecnu.edu.cn

J. Zhou
e-mail: joe@ecnu.cn

J. Yan
e-mail: yanjianming2005@ecnu.cn

J. Zhou · J. Yan · T. Wei
Computer Science and Technology Department of East China Normal University, Shanghai, 200241, China

J. Chen
Baidu Corporation, Shanghai, 200241, China
e-mail: chenjing11@baidu.com

1 Introduction

As technology advances towards the deep submicron region, the shrinking device sizes and growing transistor counts result in exponential increase of chip power density, which in turn leads to the elevated chip temperature. High temperatures are responsible for transient faults caused by timing errors since every 10 °C temperature increase can cause about 5 % interconnect delay [1]. In addition, elevated temperatures directly impact electromigration and hence reduce the mean time to failure (MTTF) of the chip. In a word, a system will fall into the predicament of functional incorrectness, low reliability and even hardware failures if the operating temperature exceeds a certain threshold. Therefore, thermal management has been a significant and pressing research issue in computing systems, especially for the embedded system with limited cooling techniques.

Considerable research effort has been devoted to the design of temperature-aware task allocation and scheduling in real-time MPSoC systems [2–10]. Research works on temperature-aware task allocation and scheduling can be classified into two categories. One category of research works aims to maximize the system performance or minimize the energy consumption under thermal constraints [2–6]. For instance, Chantem et al. [2] investigated the impacts of temperature and thermal cycling on system lifetime reliability, and presented an online task assignment and scheduling algorithm to maximize MTTF. Static and dynamic temperature-aware scheduling techniques were studied in [3] for MPSoCs to minimize energy, balance energy, and minimize hot spots. An integer linear programming (ILP) solution was first proposed to statically solve the temperature-aware task scheduling problem of MPSoC, then an adaptive dynamic policy that modifies the workload allocation policy in the runtime was designed based on the temperature history. Saha et al. [4] developed a genetic algorithm-based approach to solve the thermal constrained energy-aware real-time task mapping for heterogeneous multiprocessor systems. The proposed task mapping strategy minimizes the energy consumption with considerations of real-time constraint and maximum temperature limit. The widely utilized system-level power management technique, dynamic voltage scaling (DVS), has also been exploited in [5, 6] to minimize the energy consumption under thermal constraints. However, scaling down the operating frequency of a processor degrades the performance of the processor [11]. In addition, all the above works focus on improving lifetime reliability or minimizing energy for MPSoCs under the thermal constraint, however, the thermal optimization is not taken into account. This is not suitable for real-time systems deployed certain safety-critical applications (e.g. implantable cardioverters in medical applications) where temperature optimization is crucial for the correct and safe operation of systems.

Another category of existing research works concentrates on reducing the peak temperature under the constraint of a given threshold temperature [7–10]. An effective thermal management technique, that is load balancing through activity/task migration, has been utilized in [7, 8] to optimize the peak temperature of MPSoCs. Ebi et al. [7] presented an extremum-seeking control-based method to minimize on-chip peak temperature and optimize thermal balancing by finding an optimal scheme of activity migration based on observed temperatures. In the literature [8], a task migration algorithm that balances the loads on different cores was proposed to reduce hotspots. Ghahfarokhi and Ejlali [9] developed a schedule swapping scheme to mitigate the peak temperature of multiple processors in a distributed system. The peak temperature is reduced by swapping tasks of overheated processors with the coldest processor, and deadline constraints of tasks are guaranteed using a feasibility checking technique at static time. However, the schemes proposed in [7–9] may incur significant time and energy overheads due to frequent task migration or swapping. An optimal phased steady-state mixed integer linear programming (MILP) based solution was designed in [10] to solve the problem of temperature-aware real-time MPSoC mapping and scheduling for minimizing the chip peak temperature. However, the thermal characteristics of tasks and processors are not considered in the above works for temperature optimization.

In this paper, we proposed a thermal-ware task allocation and scheduling scheme for heterogeneous MPSoC systems to reduce peak temperature. The proposed scheme first attempts to improve the thermal profiles of tasks by minimizing the task steady state temperatures through task mapping, then further reduces the peak temperature of tasks on the processors by using the task splitting technique. The thermal characteristics of tasks and processors are exploited in the proposed task mapping strategy, and two scenarios (ideal and realistic scenarios) are considered in the proposed task splitting policy.

The rest of the paper is organized as follows. Section 2 introduces the system model and defines the thermal-aware MPSoC task mapping and scheduling problem. Section 3 presents the proposed temperature-aware task allocation strategy. Section 4 describes the proposed temperature-aware task splitting policy. The benefits and efficiency of the proposed algorithms are experimentally determined in Section 5. Section 6 concludes the paper.

2 System Model and Problem Definition

This section describes the system model and the thermal-aware task-to-processor mapping and scheduling problem.

2.1 Processor and Task Model

Processing element (PE) is the fundamental computational block integrated in the MPSoC system. For heterogeneous MPSoC platforms, the PEs can be instruction set processors (ISP), digital signal processors (DSP), general-purpose CPUs or FPGA fabric tiles. In addition, each PE can support a wide range of voltages and operating speeds (e.g. [3, 12, 13]), or equipped with a fixed voltage and operating frequency (e.g. [14, 15]). This work considers the latter system architecture (no DVS) for separate concerns since DVS would add another dimension for optimization. Specifically, the MPSoC system *PE* is composed of M heterogeneous processing elements $\{PE_1, PE_2, \cdots, PE_M\}$, where each processing element PE_m $(1 \leq m \leq M)$ is assumed to support one active mode and one sleep mode. The active mode of PE_m is characterized by a fixed supply voltage/frequency pair (v_m, f_m). Tasks are only executed when the processor is in active mode while the processor is idle in sleep mode. It is assumed that a processor can be switched from one mode to another mode at any time with the timing overhead of t^{sw}, during which no computation is allowed.

Consider a task set Γ comprising N independent periodic real-time tasks $\{\tau_1, \tau_2, \cdots, \tau_N\}$. The characteristics of a task τ_i $(1 \leq i \leq N)$ is modeled using a quadruplet $\tau_i : \{c_i, \mu_i, p_i, d_i\}$, where c_i is the worst case execution cycles of task τ_i, and μ_i (ranging in (0, 1]) is an active factor that defines how intensively functional units have been utilized by the task τ_i [16]. p_i and d_i are the period and relative deadline of the task, respectively. For a given task set (e.g. Γ), the hyper-period of the task set, denoted by L, is the lowest common multiple (LCM) of all task periods $\{p_1, p_2, \cdots, p_N\}$. Assume the task τ_i is running on the processing element PE_m, then the corresponding execution time of task τ_i is given by $\frac{c_i}{f_m}$.

2.2 Power Model

The power consumption of a CMOS device can be modeled as the sum of leakage power dissipation and dynamic power dissipation. The leakage power is temperature dependent and can be expressed as $P^{leak} = N^{gate} V^{dd} I^{leak}$, where N^{gate} is the number of gates, V^{dd} is the supply voltage, and I^{leak} is the leakage current. The leakage current I^{leak} can be formulated by a nonlinear exponential equation [17] as

$$I^{leak} = I^s (AT^2 e^{(\vartheta_1 V^{dd} + \vartheta_2)/T} + Be^{(\vartheta_3 V^{dd} + \vartheta_4)}), \tag{1}$$

where I^s is the leakage current at a certain temperature and supply voltage, T is the operating temperature, and $A, B, \vartheta_1, \vartheta_2, \vartheta_3$, and ϑ_4 are all empirical technology constants. Since the leakage current changes super linearly with temperature [18], the leakage power consumption of processing element PE_m can be closely approximated by $P_m^{leak} = \alpha_m v_m + \delta_m T v_m$ [18], where α_m and δ_m are both curve fitting constants and dependent on the architecture of PE_m. The dynamic power P_m^{dyn} is independent of the temperature and can be estimated by a strictly increasing and convex function of the operating frequency, that is, $P_m^{dyn} \propto f_m^3$ [19]. As the operating frequency is nearly linear with the supply voltage [19], the power consumption $P_{m,i}$ of task τ_i on the processing element PE_m is formulated as

$$P_{m,i} = \mu_i (C_m^{ef} v_m^3 + \alpha_m v_m + \delta_m T v_m), \tag{2}$$

where μ_i is the active factor of task τ_i, and C_m^{ef} is the effective switching capacitance of processor PE_m.

2.3 Temperature Model

An accurate and practical dynamic model of temperature is needed to accurately characterize the thermal behavior of an application. In this paper, a lumped RC thermal model proposed by Skadron et al. [20] that is widely used in the literature is adopted as the temperature model to predict the temperature of the processor. Let $T(t)$ be the temperature at time instance t, as is given by

$$RC \frac{dT(t)}{dt} + T(t) - RP(t) = T^a, \tag{3}$$

where $P(t)$ is the power consumption at time instance t. R and C are thermal resistance and capacitance, respectively. They are both processor architecture dependent constants. T^a is the ambient temperature of the die.

Consider a task τ_i executing on the processing element PE_m during the time interval $[t_0, t_0 + \Delta t]$. Let T_0 be the initial temperature at time instance t_0, the ending temperature of the task τ_i at time instance $t_0 + \Delta t$ is derived by solving (3) and then formulated as

$$T_{m,i}(t_0 + \Delta t) = T_0 e^{-K_{m,i} \Delta t} + T_{m,i}^{std} (1 - e^{-K_{m,i} \Delta t}), \tag{4}$$

where

$$T_{m,i}^{std} = \frac{R_m(\alpha_m v_m + C_m^{ef} v_m^3)\mu_i + T^a}{1 - R_m \delta_m v_m \mu_i} \tag{5}$$

is the steady state temperature of task τ_i on the processing element PE_m, and $K_{m,i} = \frac{1 - R_m \delta_m v_m \mu_i}{R_m C_m}$ is a time constant that depends on the task and processing element.

2.4 Problem Definition

Given an input task set Γ of N tasks $\{\tau_1, \tau_2, \cdots, \tau_N\}$ on a MPSoC system PE of M heterogeneous processing elements $\{PE_1, PE_2, \cdots, PE_M\}$, a task partition $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$ is feasible if $\Gamma_m \bigcap \Gamma_k = \emptyset \ \forall m \neq k$ $(1 \leq m, k \leq M)$ and $\Gamma_1 \bigcup \Gamma_2 \bigcup \cdots \bigcup \Gamma_M = \Gamma$, where Γ_m and Γ_k indicate the subset of tasks allocated to processing element PE_m and PE_k, respectively. In addition, all tasks should be finished before their individual deadlines, that is, $RT_{m,i} \leq d_i$ holds for $\forall 1 \leq m \leq M$ and $\forall 1 \leq i \leq N$, where $RT_{m,i}$ is the worst case response time of task τ_i executing on the processing element PE_m, and d_i is the relative deadline of task τ_i. The worst case response time of task τ_i is formulated as

$$RT_{m,i} = \frac{c_i}{f_m} + \sum_{\tau_j \in \Gamma_m, p_j < p_i} \left\lceil \frac{RT_{m,i}}{p_j} \right\rceil \times \frac{c_j}{f_m}, \tag{6}$$

where $\frac{c_i}{f_m}$ and $\frac{c_j}{f_m}$ are the execution time of task τ_i and τ_j on the processing element PE_m, respectively. Task τ_j has a higher priority than task τ_i for $j < i$, and $\left\lceil \frac{RT_{m,i}}{p_j} \right\rceil$ indicates the number of instances of task τ_j during the time interval of $RT_{m,i}$. The objective of the studied problem is to find an optimal task partition and scheduling strategy, which generates the minimum on-chip peak temperature. Let T^{Peak} denote the on-chip peak temperature, then it is given by

$$T^{peak} = \max \sum_{m \in [1,M]} \sum_{\tau_i \in \Gamma_m} T_{m,i}(t), \tag{7}$$

where $T_{m,i}(t)$ is the temperature of task τ_i on the processing element PE_m at time instance t and can be obtained using Eq. 4.

3 The Proposed Temperature-aware Task Allocation Strategy

The steady state temperature of a task is defined as the temperature that will be reached if infinite number of instances of the task execute continuously on the processor, which is in general utilized to represent the thermal characteristics of a task. Specifically, the lower the task steady state temperature, the better thermal characteristics the task [21]. For instance, a task is assumed to be a hot task if its steady state temperature exceeds the system maximum temperature limit, otherwise it is assumed to be a cool task [21, 22]. As is shown in Eq. 5, the steady state temperature of a given task is explicitly determined by parameters of the processor where the task executes. In other words, the steady state temperature of tasks depends on the task-to-processor assignment. Therefore, this paper aims to minimize the steady state temperature of all tasks by finding an optimal task mapping strategy.

Given a set Γ of N real-time tasks, it can be partitioned into M subsets $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$, where Γ_m $(1 \leq m \leq M)$ indicates the subset of tasks assigned to processing element PE_m. It is clear that there are M^N partitioning instances. In other words, assigning tasks to processors is essentially a NP hard problem. Thus, this work focuses on proposing a suboptimal task-to-processor assignment heuristic, in which the preference of a task to any processor is set according to its ascending order of steady state temperature. The most preferred processor for a task is the one that makes the task has a minimal steady state temperature. On the contrary, the processor that produces a maximal steady state temperature of the task is least preferred.

The proposed thermal-aware task allocation heuristic attempts to assign tasks to their most preferred processor to minimize the steady state temperature of individual tasks in the system. However, due to the real-time constraint of tasks and the limited capacity of processors, not all tasks can be assigned to their respective most preferred processor. As a result, a ranking mechanism needs to be designed in the proposed algorithm to handle the competition for a processor between multiple tasks.

Let $\Delta T_{m,i}^{std}$ be the minimum steady state temperature increment of task τ_i from its current processing element PE_m to other processing elements, as is given by

$$\Delta T_{m,i}^{std} = \min_{k \neq m, 1 \leq k, m \leq M} \{T_{k,i}^{std} | T_{k,i}^{std} \geq T_{m,i}^{std}\} - T_{m,i}^{std}, \tag{8}$$

where $T_{k,i}^{std}$ and $T_{m,i}^{std}$ are the steady state temperature of task τ_i on processing element PE_k and PE_m, respectively. They can be obtained using Eq. 5. The minimum steady state temperature increment of a task reflects the degradation of task thermal profiles when the task is allocated to its less preferred processor. This metric can be utilized to rank the tasks that are competing for the same processor. More specifically, for a given processor, tasks are sorted by the values of ΔT^{std} in decreasing order and the tasks with larger values have higher priorities to occupy the processor. Based on the ranking mechanism, a thermal-aware task allocation heuristic is designed. The task-to-processor assignment can be started from any processor, for instance, it starts from the processor with index 1 in Algorithm 1 (line 18). The principle of the heuristic is that a task assigned to a processor is not considered for an assignment to any other processor where the task has a higher steady state temperature than its currently assigned processor. The same process is repeated for all the processors, and is stopped when all the tasks are assigned to exactly one processor.

The proposed thermal-aware task allocation heuristic given in Algorithm 1 operates as follows. It takes as input a task set Γ, a processor set PE, M subsets $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$, and the utilization $\{U_1, U_2, \cdots, U_M\}$ of M processors. Lines 1-2 of the algorithm initialize the subsets $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$ and the utilizations $\{U_1, U_2, \cdots, U_M\}$ to $\{\varnothing, \varnothing, \cdots, \varnothing\}$ and $\{0, 0, \cdots, 0\}$, respectively. The minimum steady state temperature increment of tasks on each processor is iteratively derived in lines 3-15. Lines 5-6 compute the steady state temperatures of task τ_i on M processors, and obtain the maximum using $\delta_i = \max\{T_{1,i}^{std}, T_{2,i}^{std}, \cdots, T_{m,i}^{std}, \cdots, T_{M,i}^{std}\}$. If the task τ_i is not on its least suitable processor, that is, $T_{m,i}^{std} \neq \delta_i$, the minimum steady state temperature increment $\Delta T_{m,i}^{std}$ of the task is calculated using Eq. 8 (lines 9-10); otherwise, it is set to 0 (lines 11-13). The iteration stops when the minimum steady state temperature increment of all tasks are derived. Then a $M \times N$ matrix $\mathbf{A}$ is constructed to store these increments by $a_{mi} = \Delta T_{m,i}^{std}$, where a_{mi} is the element of matrix $\mathbf{A}$ (line 16).

Lines 17-35 describe the procedure of task allocation to M processors. For each processor PE_m, tasks in the task set Γ are sorted in decreasing order of a_{mi} (line 19), then the algorithm iteratively selects the top (τ_i) of ordered task list and attempts to assign it to processor PE_m. If the processor PE_m whose utilization is U_m has the capacity to accommodate task τ_i whose utilization is $U_{m,i}$ (lines 20-21), the task is assigned to the processor (line 22), and not considered for allocation to other processors where the task has a higher steady state temperature as compared to the current processor PE_m. If the task τ_i was previously assigned

Algorithm 1 Assign tasks in the task set Γ to processors in the processor set *PE*

```
Require: The task set $\Gamma$ & The processor set $PE$ & The subsets $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$ & The utilizations $\{U_1, U_2, \cdots, U_M\}$
1:  initialize $M$ subsets $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$ to $\{\varnothing, \varnothing, \cdots, \varnothing\}$;
2:  initialize $M$ utilizations $\{U_1, U_2, \cdots, U_M\}$ to $\{0, 0, \cdots, 0\}$;
3:  for $i = 1$ to $N$ do
4:      for $m = 1$ to $M$ do
5:          compute the steady state temperature $T^{std}_{m,i}$ of task $\tau_i$ on the processor $PE_m$ using Eq. 5;
6:      end for
7:      $\delta_i = \max\{T^{std}_{1,i}, T^{std}_{2,i}, \cdots, T^{std}_{m,i}, \cdots, T^{std}_{M,i}\}$;
8:      for $m = 1$ to $M$ do
9:          if $T^{std}_{m,i} \neq \delta_i$ then
10:             calculate the minimal steady state temperature increment $\Delta T^{std}_{m,i}$ of task $\tau_i$ using Eq. 8;
11:         else
12:             $\Delta T^{std}_{m,i} = 0$;
13:         end if;
14:     end for
15: end for
16: construct the matrix $\mathbf{A}$ by $a_{mi} = \Delta T^{std}_{m,i}$, where $a_{mi} \in \mathbf{A}$;
17: for all tasks $\tau_i \in \Gamma$ do
18:     for $m = 1$ to $M$ do
19:         sort all tasks $\tau_i \in \Gamma$ in decreasing order of $a_{mi}$;
20:         for all the ordered tasks $\tau_i \in \Gamma$ do
21:             if $U_m + U_{m,i} \leq 1$ then
22:                 assign task $\tau_i$ to processor $PE_m$;
23:                 for $k = 1$ to $M$ do
24:                     if $k \neq m$ && $T^{std}_{k,i} \geq T^{std}_{m,i}$ then
25:                         delete the $a_{ki}$ in the matrix $\mathbf{A}$;
26:                         update the utilization $U_k$ by $U_k = U_k - U_{k,i}$;
27:                     end if
28:                 end for
29:             else
30:                 break;
31:             end if
32:         end for
33:     end for
34: end for
35: return $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$;
```

to other processors that generate higher steady state temperatures (lines 23-24), such task assignments need to be removed, and the matrix $\mathbf{A}$, the processor utilization U_k need to be updated accordingly (lines 25-26). Then the same procedure repeats for next processor. When the first iteration is completed, the algorithm starts the next iteration again from the first processor, and repeats the iterations until all the tasks in the task set are assigned to exactly one processor. Finally, the algorithm outputs the M subsets $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$, which is allocated to M processors (line 35).

4 The Proposed Temperature-aware Task Splitting Policy

Task splitting refers to the technique that equally partitions a task into multiple sections and executes them with slack time alternatively. Unlike the previous work presented in [16] that utilizes task splitting technique to maximize throughput, the proposed thermal-aware task scheduling algorithm adopts the task splitting technique and exploits the slack that is generated due to the early completion of real-time tasks to reduce the peak temperature. A motivational example given in Fig. 1 shows the effectiveness of task splitting in reducing peak temperature. Consider a task τ with execution time of 600 ms, deadline of 1000 ms, and steady state temperature of 105 °C. The initial temperature is set to 45 °C. As is shown in Fig. 1a, the task τ has a slack time of 400 ms. When the task is executing in active mode, its temperature increases from 45 °C to 96.9 °C, then drops to 55 °C when the processor is idle. The peak temperature of 96.9 °C is reached at the end of task execution. Figure 1b demonstrates the temperature profiles of task τ when the task is equally split into 5 sections and executed alternately with the slack time. The peak temperature of 82.8 °C is reached at the end of the 5th section. Thus, in this example, the peak temperature can be reduced up to 14.1 °C by using the task splitting technique.

The proposed scheduling algorithm first identifies hot tasks from the task set based on the task steady state temperature,then splits each hot task into multiple sections to enable the alternation of hot task execution and slack time, such that the thermal profile is improved and the peak temperature is hence reduced. Two scenarios, that is, one ideal scenario that does not consider the mode switching overhead and another realistic scenario that covers the

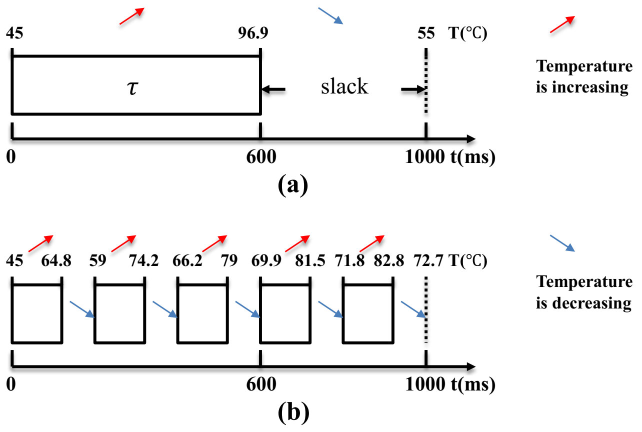


Figure 1 The motivational example of task splitting.

mode switching overhead are both taken into account in the algorithm.

4.1 The Ideal Scenario without Considering the Mode Switching Overhead

In the ideal scenario, all the slack on the processor can be exploited to cool down the execution of hot tasks and each hot task can be split into endless sections when mode switching overhead is negligible. This is motivated by the observation in [16] that the more the task splitting operation, the lower the task peak temperature. Additionally, all the hot tasks on the processor are assumed to have a uniform ideal peak temperature after thermal optimization, which is realized by endless task splitting and slack distribution. The focus of the ideal scenario is to calculate the uniform ideal peak temperature of tasks on the processor and determine the corresponding ideal slack allocated to each task.

Let Γ_m be the subset of tasks allocated to processing element PE_m, then the total slack SL_m^{tot} generated on the processor during a hyper-period L_m is given by

$$SL_m^{tot} = L_m - \sum_{\tau_i \in \Gamma_m} \frac{c_i}{f_m} \times \frac{L_m}{p_i}, \tag{9}$$

where L_m is the hyper-period of tasks in the subset Γ_m, $\frac{c_i}{f_m}$ is the execution time of task τ_i at the frequency f_m, and $\frac{L_m}{p_i}$ is the number of instances of task τ_i during the time interval of L_m.

A task can be either a hot task or a cool task based on its steady state temperature. Specifically, if the steady state temperature of task τ_i is larger than the maximum temperature limit T^{max}, the task τ_i is deemed to be a hot task and is inserted into the hot subset Γ_m^h; otherwise it is a cool task. According to the definition of ideal scenario, all the available slack time on the processor are assigned to hot tasks for reducing temperature, that is

$$\sum_{\tau_i \in \Gamma_m^h} sl_i^{ideal} \times \frac{L_m}{p_i} = SL_m^{tot}, \tag{10}$$

where sl_i^{ideal} is the ideal slack allocated to hot task τ_i. As is mentioned above, all the hot tasks on the processor will assume a uniform ideal peak temperature T_m^{ideal} through endless task splitting and slack distribution, that is,

$$\lim_{\forall \tau_i \in \Gamma_m^h, S_i \to \infty} T_{m,i}^{std} = T_m^{ideal}, \tag{11}$$

where S_i is the split number of task τ_i, and the steady state temperature $T_{m,i}^{std}$ of task τ_i is given in Eq. 5.

Hence sl_i^{ideal} and $T_{m,i}^{std}$, which are given as follows, can be derived by solving the system of the Eqs. 5, 9, 10, and 11.

$$sl_i^{ideal} = \left(\frac{T_{m,i}^{std}}{T_m^{ideal}} - 1\right)\frac{c_i B_{m,i}}{f_m}, \tag{12}$$

$$T_m^{ideal} = \frac{\sum_{\tau_i \in \Gamma_m^h} \frac{L_m}{p_i} \frac{c_i}{f_m} T_{m,i}^{std} B_{m,i}}{L_m - \sum_{\tau_i \in \Gamma_m} \frac{L_m}{p_i} \frac{c_i}{f_m} + \sum_{\tau_i \in \Gamma_m^h} \frac{L_m}{p_i} \frac{c_i}{f_m} B_{m,i}}, \tag{13}$$

where $B_{m,i} = (1 - \mu_i R_m \delta_m v_m)$.

4.2 The Realistic Scenario with Considering the Mode Switching Overhead

In the realistic scenario that considers the mode switching overhead, the key point of minimizing the peak temperature is find the optimal split number for each hot task. This is due to the fact that given a hot task, increasing the number of task splitting operation can further reduce the task peak temperature, but also cause a larger switching overhead. It has been proved in [16] that the effectiveness of task splitting technique in reducing peak temperature is maximized when the switching overhead of a task is equal to its allocated slack time. In addition, considering that the processing element needs to change its mode twice for each task splitting operation. Therefore, the optimal split number S_i of task τ_i is given by

$$S_i = \lfloor \frac{sl_i}{2 \times t^{sw}} \rfloor, \tag{14}$$

where S_i is the optimal split number of task τ_i, sl_i is the slack time allocated to task τ_i, and $2 \times t^{sw}$ is the mode switching overhead of one task splitting operation.

4.3 The Proposed Temperature-aware Task Splitting Algorithm

Both the ideal scenario and realistic scenario are discussed in the proposed temperature-aware task splitting algorithm, as is described in Algorithm 2. The algorithm takes as input the tasks of subset Γ_m on the processing element PE_m, the maximum temperature limit T^{max}, and an arbitrarily small positive number ϵ. Line 1 identifies the hot tasks in the subset Γ_m based on their steady state temperatures. Specifically, if $T_{m,i}^{std} \geq T^{max}$, the task τ_i is a hot task and inserted into the hot subset Γ_m^h. Lines 2-9 show the ideal scenario that does not consider the mode switching overhead t^{sw}. Line 3 calculates the uniform ideal peak temperature T_m^{ideal} using Eq. 13, and lines 4-5 derive the ideal slack sl_i^{ideal} of each hot task using Eq. 12. Then the available slack allocated to task τ_i under the real-time constraint is obtained using the procedure $sl_i^{rt} = \mathrm{SLK}(\tau_i, \Gamma_m)$ in line 6. The slack $sl_i = \min\{sl_i^{ideal}, sl_i^{rt}\}$ is finally assigned to task τ_i with endless task splitting for thermal optimization, where the operator $\lhd$ indicates the operation of task splitting (lines 7-8).

Algorithm 2 Thermal-aware task splitting on the processor PE_m

```
Require: The task subset $\Gamma_m$ & The maximum temperature limit $T^{max}$ & An arbitrarily small positive number $\epsilon$
1: identify hot tasks from $\tau_i \in \Gamma_m$ based on $T^{std}_{m,i}$ and insert hot tasks into $\Gamma^h_m$ by $\Gamma^h_m = \{\tau_i | \tau_i \in \Gamma_m \wedge T^{std}_{m,i} \geq T^{max}\}$;
2: if mode switching overhead $t^{sw}$ is not considered then
3:     calculate the uniform ideal peak temperature $T^{ideal}_m$ using Eq. 13;
4:     for all hot tasks $\tau_i \in \Gamma^h_m$ do
5:         derive the ideal slack $sl^{ideal}_i$ of $\tau_i$ using Eq. 12;
6:         obtain the available slack allocated to $\tau_i$ under real-time constraint using $sl^{rt}_i = \text{SLK}(\tau_i, \Gamma_m)$;
7:         $sl_i = \min\{sl^{ideal}_i, sl^{rt}_i\}$;
8:         $\tau_i \lhd (sl_i, \infty)$;   // $\lhd$ is the operator of task splitting
9:     end for
10: else
11:     for all hot tasks $\tau_i \in \Gamma^h_m$ do
12:         $sl_i = \text{SLK}(\tau_i, \Gamma_m)$;
13:         compute the optimal split number $S_i$ using Eq. 14;
14:         $\tau_i \lhd (sl_i, S_i)$;
15:     end for
16: end if

        Procedure SLK($\tau_i, \Gamma_m$)
17: $[sl_{low}, sl_{high}] = [0, d_i - RT_{m,i}]$;
18: $\rho = sl_{high} - sl_{low}$;
19: while ($\epsilon < \rho$) do
20:     $sl_{mid} = (sl_{low} + sl_{high})/2$;
21:     $\tau_{tem} = \tau_i + sl_{mid}$, $\Gamma_{tem} = \Gamma_m + \tau_{tem}$;
22:     if (RTFA($\Gamma_{tem}$) == true) then
23:         $[sl_{low}, sl_{high}] = [sl_{mid}, sl_{high}]$;
24:     else
25:         $[sl_{low}, sl_{high}] = [sl_{low}, sl_{mid}]$;
26:     end if
27:     $\rho = sl_{high} - sl_{low}$;
28: end while
29: return $sl_{low}$;

        Procedure RTFA($\Gamma_{tem}$)
30: for all tasks $\tau_i \in \Gamma_{tem}$ do
31:     obtain the worst case response time $RT_{m,i}$ of task $\tau_i$ using Eq. 6;
32:     if $RT_{m,i} > d_i$ then
33:         break;
34:     end if
35: end for
36: if $i == sizeof(\Gamma_{tem}) + 1$ then
37:     return true;
38: else
39:     return false;
40: end if
```

The realistic scenario is described in lines 10-16. The slack assigned to task τ_i is $sl_i = \text{SLK}(\tau_i, \Gamma_m)$ and the optimal split number S_i of the task is computed using Eq. 14 (lines 12-13). Then line 14 splits task τ_i into S_i sections and alternates their execution with slack time to reduce peak temperature.

The procedure SLAK that derives the maximum available slack for a task is a binary search-based approach. Inputs to the procedure are the task τ_i, the subset Γ_m, and the arbitrarily small positive number ϵ. A search space $[sl_{low}, sl_{high}]$ is defined and initialized to $[0, d_i - RT_{m,i}]$, where sl_{low} and sl_{high} denote the lower and upper bound of the space, respectively, and d_i and $RT_{m,i}$ are the deadline and response time of task τ_i, respectively (line 17). The search length, which is denoted by ρ, is set to $\rho = sl_{high} - sl_{low}$ (line 18). Lines 19-28 describe the searching process. In each round of iteration, a dummy task τ_{tem} is created and initialized to τ_i, the median sl_{mid} of the search space $[sl_{low}, sl_{high}]$ is calculated and taken as the slack assigned to the dummy task τ_{tem}, and a dummy subset Γ_{tem} is created and set to $\Gamma_m + \tau_{tem}$. The procedure RTFA presented in Algorithm 2 is called to check if the timing constraint of tasks in the subset Γ_{tem} is met. The search space $[sl_{low}, sl_{high}]$ and the search length ρ are updated in each iteration, and the process stops when the ρ is less than yet close enough to the arbitrarily small positive number ϵ. The lower bound sl_{low} of the search space is returned as the maximum available slack that could be assigned to the task τ_i (line 29). The feasibility analysis technique, referred to as real-time feasibility analysis (RTFA), is utilized to check if the timing constraint is satisfied. If the response time $RT_{m,i}$ of the task τ_i exceeds the deadline d_i of the task, the task τ_i cannot be feasibly assigned to the processing element PE_m (lines 30-40).

There is a possibility that the available slack time during the system is insufficient to control the peak temperature below the maximum temperature limit. In this scenario, the thermal characteristics of cool tasks can be exploited to further reduce the peak temperature by splitting cool tasks into multiple sections and alternating the execution of cool subtasks and hot subtasks. If cool tasks still cannot help to limit the peak temperature under the safe threshold, in which case the task set is deemed infeasible.

5 Experimental Results And Discussions

Extensive simulation experiments have been conducted to validate the effectiveness of the proposed task allocation and scheduling scheme in thermal management. The proposed algorithms were implemented in C++, and the simulation was performed on a machine with Intel Dual-Core 2.3 GHz processor and 8GB memory. This section first describes experimental settings for the simulation, then compares the proposed algorithms with benchmarking schemes.

Moreover, the same simulation settings are adopted for the proposed algorithms and benchmarking schemes for the sake of fair comparison.

5.1 Experimental Settings

A simulated platform of multiple processing elements is constructed. The number of processing elements is assumed to be six, and the corresponding supply voltages of six processing elements PE_1, PE_2, PE_3, PE_4, PE_5, PE_6 are set to 0.85 V, 0.9 V, 0.95 V, 1.0 V, 1.05 V, 1.1 V [23]. The parameters of the operating point of each processing element are constant, that is, the supply voltage v, curve fitting constants α and δ, effective switching capacitance C^{ef} and operating frequency f of each processing element are fixed, as is listed in Table 1.

25 benchmarking tasks are selected from Mibench [24] and Mediabench [25] to form the task set Γ for validating the proposed task allocation and scheduling algorithm. The numbers of clock cycles of these tasks are in the range of $[4 \times 10^7, 6 \times 10^8]$. The architecture-level power simulator McPAT [26] is utilized to obtain power consumptions of tasks. It can model all three types of power dissipation, including dynamic, leakage, and short-circuit power, and provide a complete view of power consumptions. The task active factor μ evenly takes values between [0.4, 1] to manifest the heterogeneous nature of tasks [16]. The average of thermal resistance R and thermal capacitance C of processing elements are assumed to be 0.80 K/W and 340 J/K, respectively [23], and the variance of the R and C of a processing element is deemed to be 0.05 and 10.0, respectively. The overhead of mode switching is assumed to be 5 ms [16], and the ambient temperature T^a is set to 35 °C. The temperature is obtained using the thermal simulator HotSpot [20], which is an accurate yet fast and practical tool to capture thermal profiles.

Table 1 Parameters of the simulated platform [23].

v	α	δ	C^{ef}	f
0.85	7.3249	0.1666	15.0	0.8010
0.90	8.6126	0.1754	15.0	0.8291
0.95	10.238	0.1846	15.0	0.8553
1.00	12.315	0.1942	15.0	0.8797
1.05	14.998	0.2043	15.0	0.9027
1.10	18.497	0.2149	15.0	1.0

5.2 Experimental Results

5.2.1 Comparison of the Task Steady State Temperature

Two benchmarking methods are implemented and compared to evaluate the effectiveness of the proposed task allocation heuristics in minimizing task steady state temperature. The first one is the Random algorithm that randomly assigns tasks to processors without utilizing any techniques. The second one, referred to as RMBF [27], is the partitioning heuristic that assigns the task with the highest priority to the processor with smallest unused capacity among those processors on which it fits. The proposed thermal-aware task allocation heuristic attempts to select the most suitable processing element in terms of task steady state temperature for each task in the system.

Figure 2 shows the steady state temperature of 25 tasks under the proposed method, and benchmarking schemes Random and RMBF [27]. It has been demonstrated in Fig. 2 that the steady state temperature of the proposed algorithm is almost much lower than that of the Random and RMBF [27] method. For example, the average steady state temperature of 25 tasks achieved by the proposed algorithm, benchmarking schemes Random and RMBF are 68.86 °C, 72.4 °C, and 71.32 °C, respectively. Thus, the average steady state temperature achieved by the proposed algorithm is 3.57 °C lower than that of scheme Random, and 2.46 °C lower than that of scheme RMBF [27]. The proposed algorithm outperforms the benchmarking methods Random and RMBF [27] in thermal management since it fully takes the thermal characteristics of tasks and processing elements into account during the task allocation.

5.2.2 Comparison of the Peak Temperature

In the proposed scheme, the thermal profiles of tasks in the task set are improved by selecting the most suitable processing element for each task, then the peak temperature is reduced by splitting hot tasks into multiple sections and enabling the alternation of hot task execution and slack time. The proposed algorithm is compared with two benchmarking methods in terms of peak temperature to demonstrate its effectiveness of thermal management. The first benchmarking scheme, referred to as NOTM, does not utilize any thermal management techniques while the second benchmarking scheme, referred to as task sequencing (TS) [21], utilizes thermal characteristics of tasks to derive a task sequence in the alternate order of being cool-hot.

Figure 3 plots the peak temperature of six processing elements under the proposed method, and benchmarking

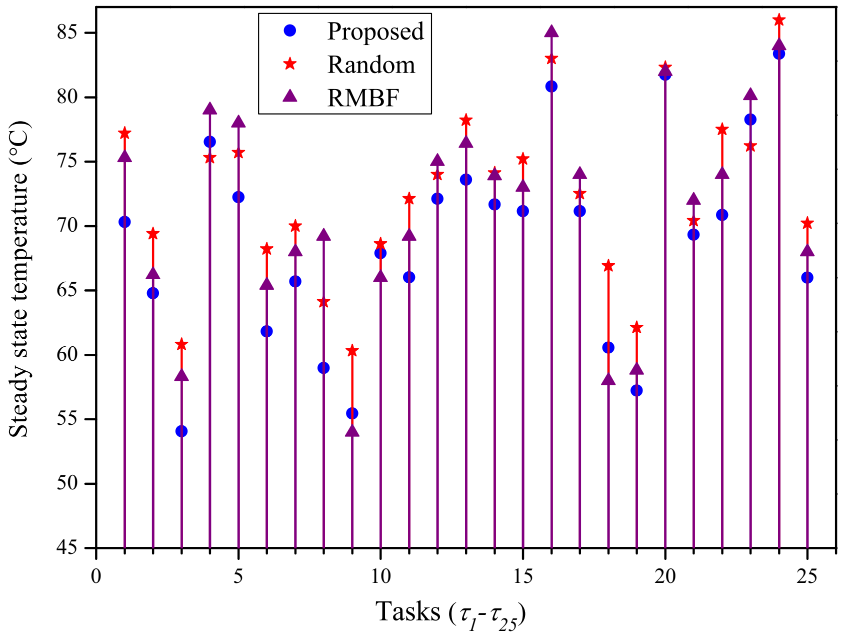


Figure 2 The steady state temperature of 25 tasks under the proposed method, and benchmarking schemes Random and RMBF [27].

schemes TS [21] and NOTM. The initial temperature of each processing element is assumed to equal the ambient temperature T^a, which is set to 35 °C in this study. It has been shown in Fig. 3 that the peak temperature of six processing elements achieved by the proposed method is much lower than that of benchmarking schemes TS [21] and NOTM. For instance, the proposed method reduces the peak temperature of tasks on the processing element PE_2 by 5.8% as compared to the TS [21] scheme, and 11.5% as compared to the NOTM scheme. The NOTM method does not employ any thermal control techniques and is utilized as a baseline to show the highest efficiency of the proposed algorithm in reducing peak temperature. The proposed algorithm also outperforms the state-of-the-art scheme TS [21] in thermal management since it first improves the thermal profiles of each task through a thermal-ware task mapping heuristics, then exploits the task splitting technique and slack time to cool down the execution of hot tasks on the processor for a lower peak temperature.

In addition, Fig. 4 compares the instantaneous temperature of benchmarking scheme TS [21] and the proposed task splitting algorithm. Two tasks are running on the processor. One is a cool task with execution time of 520 s and another

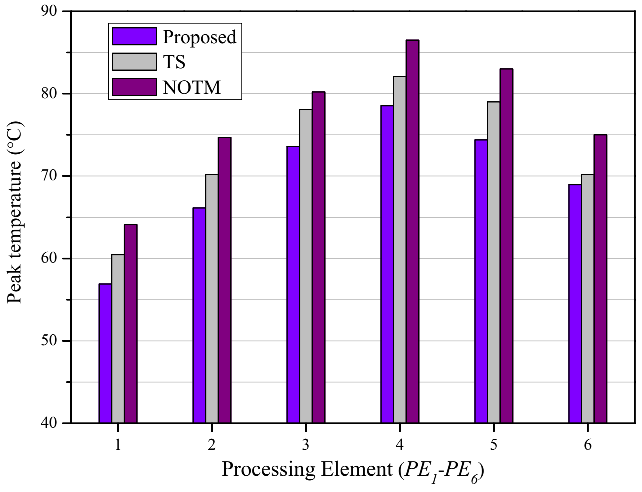


Figure 3 The peak temperature of six processing elements under the proposed method, and benchmarking schemes TS [21] and NOTM.

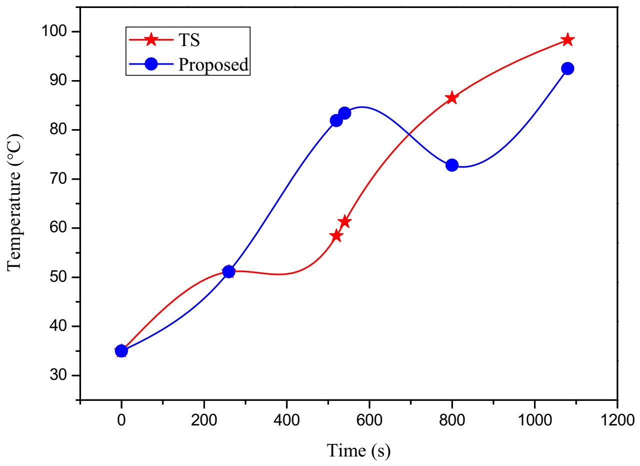


Figure 4 Compare the instantaneous temperature of benchmarking scheme TS [21] and the proposed task splitting algorithm.

is a hot task with execution time of 560 s. The TS [21] method executes the two tasks in the order of being cool-hot. The proposed task splitting algorithm first partitions the cool task and hot task into two cool subtasks and two hot subtasks respectively, then interleaves the execution of cool subtasks and hot subtasks. The peak temperature of the proposed task splitting algorithm is 92.59 °C, which is 5.75 °C lower than that of the benchmarking method TS [21].

6 Conclusions

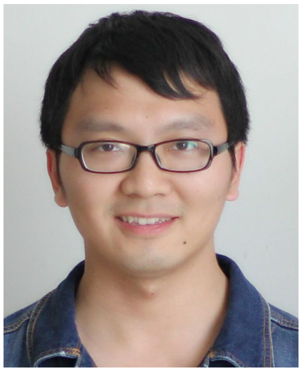
This paper explores the thermal-aware task mapping and scheduling for heterogeneous MPSoC systems to reduce the on-chip peak temperature under the real-time constraint. The proposed algorithms explicitly exploit thermal characteristics of both tasks and processors to minimize the peak temperature without incurring significant overheads. The proposed algorithms are implemented in two steps. In the first step, tasks are assigned to individual processors to minimize their respective steady state temperature, while in the second steps, tasks with undesired thermal characteristics (hot tasks) are selected and split into multiple sections to enable the alternation of hot task execution and slack time, such that the peak temperature of tasks is reduced. Experimental results show that a 11.5% reduction of peak temperature can be achieved by the proposed algorithms as compared to the benchmarking schemes.

Acknowledgments This work was supported in part by the Natural Science Foundation of Shanghai City under the grant 12ZR1409200 and by the Scientific Research Foundation for Returned Scholars, Ministry of Education of China, under the grant 44420340.

References

1. Narayanan, V., & Xie, Y. (2006). Reliability concerns in embedded system designs. *Computer*, *39*(1), 118–120.
2. Chantem, T., Xiang, Y., Hu, X., & Dick, R. (2013). Enhancing multicore reliability through wear compensation in online assignment and scheduling. In *Proceedings of the international conference on design, automation and test in Europe* (pp. 1373–1378).
3. Coskun, A., Rosing, T., Whisnant, K., & Gross, K. (2008). Static and dynamic temperature-aware scheduling for multiprocessor SoCs. *IEEE Transactions on Very Large Scale Integration Systems*, *16*(9), 1127–1140.
4. Saha, S., Lu, Y., & Deogun, J. (2012). Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems. In *Proceedings of the international conference on embedded and real-time computing systems and applications* (pp. 41–50).
5. Bao, M., Andrei, A., Eles, P., & Peng, Z. (2008). Temperature-aware voltage selection for energy optimization. In *Proceedings of the international conference on design, automation and test in Europe* (pp. 1083–1086).
6. Gupta, N., & Mahapatra, R. (2011). Temperature aware energy management for real-time scheduling. In *Proceedings of the international symposium on quality electronic design* (pp. 1–6).
7. Ebi, T., Amrouch, H., & Henkel, J. (2012). Cool: control-based optimization of load-balancing for thermal behavior. In *Proceedings of the international conference on hardware/software codesign and system synthesis* (pp. 255–264).
8. Mulas, F., Pittau, M., Buttu, M., Carta, S., Acquaviva, A., Benini, L., & Atienza, D. (2008). Thermal balancing policy for streaming computing on multiprocessor architectures. In *Proceedings of the international conference on design, automation and test in Europe* (pp. 734–739).
9. Ghahfarokhi, F., & Ejlali, A. (2010). Schedule swapping: a technique for temperature management of distributed embedded systems. In *Proceedings of the international conference on embedded and ubiquitous computing* (pp. 1–6).
10. Chantem, T., Hu, X., & Dick, R. (2011). Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs. *IEEE Transactions on Very Large Scale Interation Systems*, *19*(10), 1884–1897.
11. Wei, T., Mishra, P., Wu, K., & Zhou, J. (2012). Quasi-static fault-tolerant scheduling schemes for energy-efficient hard real-time systems. *Journal of Systems and Software*, *85*(6), 1386–1399.
12. Singh, A., Das, A., & Kumar, A. (2013). Energy optimization by exploiting execution slacks in streaming applications on multiprocessor systems. In *Proceedings of the international conference on design automation*.
13. Murali, S., Mutapcic, A., Atienza, D., Gupta, R., Boyd, S., & Micheli, G. (2007). Temperature-aware processor frequency assignment for MPSoCs using convex optimization. In *Proceedings of the international conference on hardware/software codesign and system synthesis* (pp. 111–116).
14. Chen, G., Huang, K., Huang, J., & Knoll, A. (2013). Cache partitioning and scheduling for energy optimization of real-time MPSoCs. In *Proceedings of the international conference on application-specific systems, architectures and processors* (pp. 35–41).
15. Intel Corporation, Single-chip cloud computer (SCC). [Online]. Available: http://www.intel.com/content/www/us/en/research/intel-labs-single-chip-cloud-overview-paper.html.
16. Huang, H., Chaturvedi, V., Quan, G., Fan, J., & Qiu, M. (2014). Throughput maximization for periodic real-time systems under the maximal temperature constraint. *ACM Transactions on Embedded Computing Systems*, *13*(2s).
17. Liao, W., He, L., & Lepak, K. (2005). Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *24*(7), 1042–1053.
18. Liu, Y., Dick, R., Shang, L., & Yang, H. (2007). Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *Proceedings of the international conference on design, automation and test in Europe* (pp. 1526–1531).
19. Weste, N., & Eshraghian, K. (1992). *Principles of CMOS VLSI design: a system perspective.* Addison-Wesley Publishing Company.
20. Skadron, K., Stan, M., Sankaranarayanan, K., Huang, W., Velusamy, S., & Tarjan, D. (2004). Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, *1*(1), 94–125.
21. Jayaseelan, R., & Mitra, T. (2008). Temperature aware task sequencing and voltage scaling. In *Proceedings of the international conference on computer-aided design* (pp. 618–623).
22. Zhou, J., & Wei, T. (2015). Stochastic thermal-aware real-time task scheduling with considerations of soft errors. *Journal of Systems and Software*, *102*, 123–133.

23. Quan, G., & Chaturvedi, V. (2010). Feasibility analysis for temperature constraint hard real-time periodic tasks. *IEEE Transactions on Industrial Informatics*, *6*(3), 329–339.
24. Guthaus, M., Ringenberg, J., Ernst, D., Austin, T., Mudge, T., & Brown, R. (2001). Mibench: a free, commercially representative embedded benchmark suite. In *Proceedings of the international workshop on workload characterization* (pp. 3–14).
25. Lee, C., Potkonjak, M., & Mangione-Smith, W. (1997). Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. In *Proceedings of the international symposium on microarchitecture* (pp. 330–335).
26. Li, S., Ahn, J., Strong, R., Brockman, J., Tullsen, D., & Jouppi, N. (2009). Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the international symposium on microarchitecture* (pp. 469–480).
27. Zapata, O., & Alvarez, P. (2005). EDF and RM multiprocessor scheduling algorithms: survey and performance evaluation. *Seccion de Computacion Av. IPN.*



Junlong Zhou is currently working towards his Ph.D. degree in Computer Science and Technology Department at East China Normal University. Now he is also a research visitor at the University of Notre Dame. His research interests are in the areas of real-time systems, energy efficient and reliable embedded system design, and thermal-aware scheduling techniques. He is an active reviewer of many international journals, including IEEE transactions on Industrial Informatics, Journal of Systems and Software, Journal of Scheduling, Journal of Circuits, Systems, and Computers (World Scientific), Journal of Modeling and Simulation (ACTA Press). He is a student member of IEEE.

Jianming Yan is currently pursuing the master's degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interests include task allocation and scheduling techniques in heterogeneous real-time MPSoC systems.

Jing Chen received her M.S. degree in Computer Science and Technology from East China Normal University, Shanghai, China, in 2013. She is currently with Baidu Corporation, Shanghai, China. Her current research interests include the management of energy and reliability for real-time embedded systems.

Tongquan Wei received his Ph.D. degree in Electrical Engineering from Michigan Technological University in 2009. He is currently an Associate Professor in the Department of Computer Science and Technology at the East China Normal University.

His research interests are in the areas of real-time systems, green and reliable computing, and parallel and distributed systems. He serves as a Regional Editor for Journal of Circuits, Systems, and Computers (World Scientific) since 2012. He also served as the Guest Editor of the IEEE Transactions on Industrial Informatics Special Section on Building Automation, Smart Homes, and Communities, and the ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems for Energy-Efficient, Reliable, and Secure Smart Homes. He is a member of the IEEE.