Thermal-Aware Task Scheduling for Energy Minimization in Heterogeneous Real-Time MPSoC Systems

Junlong Zhou, *Student Member, IEEE*, Tongquan Wei, *Member, IEEE*, Mingsong Chen, *Member, IEEE*, Jianming Yan, Xiaobo Sharon Hu, *Senior Member, IEEE*, and Yue Ma

***Abstract*—With the continuous scaling of CMOS devices, the increase in power density and system integration level have not only resulted in huge energy consumption but also led to elevated chip temperature. Thus, energy efficient task scheduling with thermal consideration has become a pressing research issue in computing systems, especially for real-time embedded systems with limited cooling techniques. In this paper, we design a two-stage energy-efficient temperature-aware task scheduling scheme for heterogeneous real-time multiprocessor system-on-chip (MPSoC) systems. In the first stage, we analyze the energy optimality of assigning real-time tasks to multiple processors of an MPSoC system, and design a task assignment heuristic that minimizes the system dynamic energy consumption under the constraint of task deadlines. In the second stage, the optimality of minimizing the peak temperature of a processor is investigated, and a slack distribution heuristic is proposed to improve the temperature profile of each processor under the thermal constraint, thus the temperature-dependent system leakage energy consumption is reduced. Through the extensive efforts made in two stages, the system overall energy consumption is minimized. Experimental results have demonstrated the effectiveness of our scheme.**

***Index Terms*—Energy-efficient, real-time MPSoC systems, task allocation and scheduling, thermal-aware.**

I. Introduction

THE ADVANCE of technology scaling enables the integration of multiple processing elements, memory hierarchies, and dedicated hardware and I/O components on a single silicon die to form a multiprocessor system-on-chip (MPSoC) system. An MPSoC system is naturally heterogeneous in the sense that its processing elements such as customized hardware modules, programmable microprocessors, and embedded field-programmable gate arrays have distinctive functionalities and demonstrate varying computing capability [1]. Due to their powerful parallel processing capability, higher computing density, and lower clock frequencies, MPSoCs have replaced uniprocessors to become the main design paradigms for current and future embedded microprocessors in various application domains [2]. The distinct features of different types of processors of an MPSoC system can be exploited to meet the stringent design requirements of emerging real-time applications. In this paper, we focus on task scheduling issues for heterogeneous real-time MPSoC systems.

The conventional research on MPSoC systems concentrates on trading off the performance with resource requirements. Recently, increasing system integration level and decreasing feature sizes of very large-scale integration circuits have led to a striking rise in power density [3], which not only results in huge energy consumption but also leads to elevated chip temperatures. Increase in energy consumption causes serious technical, economic, and ecological problems, such that energy management has become a critical issue in computing systems, especially for battery-powered systems that operate in harsh environments [4]–[6]. High chip temperature has adverse impact on system reliability, performance, and cost. A system will fall into the predicament of functional incorrectness, low reliability, and even permanent damage if operating temperature exceeds a certain threshold [7]. Industrial studies have shown that a difference in operating temperature (10 °C–15 °C) can make a 2× difference in device lifespan [8]. Thus, energy and thermal management has become a significant and pressing research issue in computing systems.

Considerable research efforts have been devoted to the investigation of task allocation for energy minimization in heterogeneous MPSoC systems. The heterogeneities of MPSoC systems are manifested by the varying core types, different operating frequencies and power consumptions, and distinctive state switching overheads of processors. Colin *et al.* [9] addressed the problem of allocating real-time tasks onto heterogeneous cores for energy minimization under timing constraints. The presented allocation heuristics are designed as approximations to a target load distribution derived analytically. Awan and Petters [10] explored the energy efficient task

Manuscript received April 25, 2015; revised August 21, 2015; accepted October 23, 2015. Date of publication November 16, 2015; date of current version July 15, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 91418203 and Grant 61202103, and in part by the East China Normal University Outstanding Doctoral Dissertation Cultivation Plan of Action under Grant PY2015047. This paper was recommended by Associate Editor D. Chen. *(Corresponding author: Tongquan Wei.)*

J. Zhou, T. Wei, and J. Yan are with the Shanghai Key Laboratory of Multidimensional Information Processing, and the Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China (e-mail: tqwei@cs.ecnu.edu.cn).

M. Chen is with the Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200241, China.

X. S. Hu and Y. Ma are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2501286

mapping on heterogeneous multicore platform to reduce overall energy consumption of a real-time system. The developed heuristic first assigns tasks to processors to minimize the system active energy consumption. It then trades off higher active energy consumption for increased ability to use more efficient sleep states to reduce the system static power consumption. Quan and Pimentel [11] designed a hybrid task mapping algorithm for heterogeneous MPSoCs to improve system efficiency. The hybrid method aims to maximize the throughput via static task mappings under a predefined energy budget, and further improve the performance of the mappings and reduce the energy consumption by considering the dynamic behavior of applications at runtime. All the above works attempt to fully exploit the energy saving potentials of heterogeneous processors. However, the effectiveness of utilizing the heterogeneities of an MPSoC system to reduce the chip temperature is not investigated.

Considering the temperature design constraint, Yu *et al*. [12] leveraged the task-level adaptability and designed a thermal-aware frequency scaling-based scheduling algorithm for maximizing the execution quality-of-service of applications on heterogeneous MPSoC platforms. The presented method converts the temperature threshold into timing constraints, then optimizes the total workload cycle over all processors by judicious frequency selection. Wang *et al*. [13] studied the problem of reducing the peak temperature of real-time applications in MPSoC systems by utilizing system heterogeneities caused by manufacturing variations. Although it is effective to reduce the peak temperature by exploiting the heterogeneities of MPSoC systems, the energy design constraint is not discussed in these works. In addition, the heterogeneities of real-time tasks are not utilized to enhance system temperature and energy profiles.

Real-time tasks are deemed to be heterogenous when they consume different power at the same operating frequency and temperature on the same processor [14]. In [15], the heterogeneities of both system architecture and real-time tasks are used to minimize the energy consumption. A relaxation-based algorithm for three types of heterogeneous platforms are designed to achieve the task partition that is closest to the optimal solution of the relaxed problems. However, temperature is not considered as a design constraint. Saha *et al*. [16] developed a genetic algorithm-based task allocation that minimizes the energy consumption under the constraints of temperature limit and task deadlines. Although both energy and temperature are taken into account for optimization, the heterogeneity of real-time tasks is not considered.

In this paper, we present a static two-stage energy-efficient temperature-aware task allocation and scheduling scheme for heterogeneous real-time MPSoC systems under the constraints of task deadlines and temperature limit. The first stage of the proposed approach aims to minimize the system dynamic energy consumption by assigning the subset having a larger power dissipation factor to the processor having a smaller power dissipation factor. The second stage of the proposed approach aims to minimize the system leakage energy consumption by reducing the peak temperature of processors through slack distribution. In the two stages, feasibility analysis techniques are also designed to ensure that the target system meets its timing and thermal constraints. The major contributions of this paper are summarized as follows.

1) We analyze the energy optimality of assigning tasks to multiple processors of an MPSoC system. Based on this analysis, we design a task assignment heuristic that minimizes the system dynamic energy consumption.
2) We prove that the peak temperature of tasks in the thermal steady state is minimal if tasks on the same processor assume a uniform steady state temperature. Using a slack distribution policy that is developed based on this observation, the temperature profiles of processors are improved, and the temperature-dependent system leakage energy consumption is hence reduced.
3) We exploit the heterogeneities of both system architecture and real-time tasks to reduce the system energy consumption. We also utilize feasibility analysis techniques to ensure real-time and temperature constraints are satisfied.

The rest of this paper is organized as follows. Section II introduces the system architecture and models and Section III shows the overview of energy minimization. Section IV presents the proposed task assignment strategy for minimizing the system dynamic energy consumption and Section V describes the proposed slack distribution policy that reduces the temperature for minimizing the system leakage energy consumption. The effectiveness of the proposed approach is verified in Section VI and concluding remarks are given in Section VII.

II. System Architecture and Models

Consider an MPSoC system $\mathcal{P}$ consisting of M processors $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k, \ldots, \mathcal{P}_M\}$, where every processor $\mathcal{P}_k$ $(1 \leq k \leq M)$ operates at a given supply voltage and processing speed pair (v_k, s_k). Dynamic voltage scaling (DVS) is not considered in this paper since it would add another dimension for optimization [10]. In addition, the benefit of using DVS to reduce temperature is partially offset by the adverse impact of DVS on system performance.

A. Task Model

We consider real-time periodic tasks to be executed on the concerned MPSoC platform. Tasks are assumed to be heterogeneous in the sense that different tasks exhibit different power consumptions on the same processor, even executing at the same operating speed and temperature. This is due to the fact that power consumptions of tasks strongly rely on circuit activities and usage patterns of different functional units [14]. Thus, the activity factor of a task, denoted by μ (ranging in (0, 1]), is introduced to capture how intensively functional units are being utilized by the task [17].

The timing characteristics of a periodic real-time task is in general described by three parameters, that is, the deadline, the period, and the worst-case execution time in cycles. A real-time task must guarantee response within a specified time constraint, which is referred to as the deadline. In a periodic real-time system, each task requires repeated execution, and the time duration between the time point of one task ready to be executed and that of the next is referred to as the period. Associating each real-time task with a worst-case execution time and a period is widely accepted in the real-time system community and is commonly adopted in the literature.

Assuming that a set Γ contains N real-time periodic tasks, denoted by $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_i, \ldots, \tau_N\}$, and considering the task activity factor, the characteristics of τ_i $(1 \leq i \leq N)$ is described by a quadruplet $\tau_i : \{D_i, p_i, c_i, \mu_i\}$, where D_i is the deadline, p_i is the period, c_i is the worst-case execution time in cycles, and μ_i is the task activity factor. The hyper-period of set Γ, denoted by H, is the least common multiple of periods $\{p_1, p_2, \ldots, p_N\}$. Let $\mathrm{ET}(i, k)$ be the execution time of task τ_i on processor $\mathcal{P}_k$ at supply voltage/speed (v_k, s_k), that is

$$\mathrm{ET}(i, k) = \frac{c_i}{s_k}. \tag{1}$$

B. Power Model

The power consumption P of a CMOS device can be modeled as the sum of dynamic power consumption P_{dyn} and leakage (or static) power consumption P_{leak}, that is

$$P = \hbar \cdot P_{\mathrm{dyn}} + P_{\mathrm{leak}}. \tag{2}$$

Here $\hbar$ is employed to represent system states and indicate whether the system is currently consuming dynamic power. Specifically, $\hbar = 1$ when the processor is in the active state and $\hbar = 0$ when the processor is in the idle state.

Dynamic power consumption mainly results from charging and discharging of gates in the circuits. It is independent of the temperature, and can be formulated as a function of supply voltage V_{dd} and operating frequency f [18], that is

$$P_{\mathrm{dyn}} = C^{\mathrm{eff}} V_{\mathrm{dd}}^2 f \tag{3}$$

where C^{eff} is the effective capacitance. Since $s \propto f$, where s is the processor speed, the power consumption of task τ_i on processor $\mathcal{P}_k$ at the supply voltage/speed (v_k, s_k) is

$$P_{\mathrm{dyn}}(i, k) = C_k^{\mathrm{eff}} \mu_i v_k^2 s_k \tag{4}$$

where μ_i is the activity factor of task τ_i.

Leakage power consumption mainly results from the leakage current and is expressed as

$$P_{\mathrm{leak}} = N_{\mathrm{gate}} \cdot V_{\mathrm{dd}} \cdot I_{\mathrm{leak}} \tag{5}$$

where N_{gate} is the number of gates, V_{dd} is the supply voltage, and I_{leak} is the leakage current. I_{leak} can be captured by a nonlinear exponential equation [19] as

$$I_{\mathrm{leak}} = I_s\Big(\mathcal{A}T^2 e^{(\vartheta_1 V_{\mathrm{dd}} + \vartheta_2)/T} + \mathcal{B} e^{(\vartheta_3 V_{\mathrm{dd}} + \vartheta_4)}\Big) \tag{6}$$

where I_s is the leakage current at a certain reference temperature and supply voltage, T is the operating temperature, and $\mathcal{A}$, $\mathcal{B}$, ϑ_1, ϑ_2, ϑ_3, and ϑ_4 are empirically determined, technology-dependent constants. (6) clearly demonstrates the complex relationship between the leakage power and temperature. However, the high-order and nonlinear terms make (6) prohibitive to perform real-time feasibility analysis (RTFA). As reported in [20], the leakage current changes super linearly with the temperature and using linear approximation to model the leakage-temperature dependence can significantly simplify the leakage model while maintaining an acceptable accuracy. Therefore, as in [21], we model the leakage power of processor $\mathcal{P}_k$ at the supply voltage/speed (v_k, s_k) as

$$P_{\mathrm{leak}}(k) = (\alpha_k + \beta_k T) \cdot v_k \tag{7}$$

where α_k and β_k are constants depending on processor $\mathcal{P}_k$.

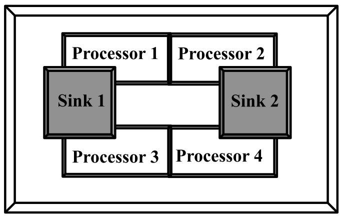


Fig. 1. Example layout of four processors with two heat sinks [22].

C. Thermal Model

In an MPSoC system, each processor is assumed to be a discrete thermal element, and there is a set of heat sinks on top of the processors. These heat sinks are only used for heat dissipation and generate no power. An example layout of four processors with two heat sinks is given in Fig. 1. Heat transfer among the processors and heat sinks is a complicated dynamic process depending on the physical system. This dynamic heat transfer process can be closely approximated by Fourier's law [16], [22]–[24], where the thermal coefficients can be obtained by using the RC models [16], [20]–[25].

Let $G_{k,m}$ represent the thermal conductance between processor $\mathcal{P}_k$ and $\mathcal{P}_m$ in set $\mathcal{P}$ and $G_{k,m} = G_{m,k}$ holds for any $1 \leq k \neq m \leq M$. If there is no heat transfer between processor $\mathcal{P}_k$ and $\mathcal{P}_m$, then $G_{k,m} = 0$. $G_{k,k} = 0$ holds for any processor in processor set $\mathcal{P}$, and the thermal capacitance of processor $\mathcal{P}_k$ is C_k. Let $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_h, \ldots, \Theta_{\mathcal{H}}\}$ denote the set of $\mathcal{H}$ heat sinks on top of the processors. The vertical thermal conductance between processor $\mathcal{P}_k$ and heat sink Θ_h is $G_{k,h}$, which depends on the interface material and the thickness. If there is no heat dissipation from processor $\mathcal{P}_k$ to heat sink Θ_h, then $G_{k,h} = 0$. The lateral thermal conductance between heat sink Θ_h and Θ_ℓ in set Θ is $G_{h,\ell}$ and $G_{h,\ell} = G_{\ell,h}$ holds for any $1 \leq h \neq \ell \leq \mathcal{H}$. The thermal conductance of a heat sink that dissipates heat to the ambient is G_{amb}. The thermal capacitance of sink Θ_h in set Θ is C_h.

Let $T_k(t)$ and $T_h(t)$ be the temperature of processor $\mathcal{P}_k$ and heat sink Θ_h at time instance t, respectively. Let T_{amb} and $P_k(t)$ be the ambient temperature of the chip and the power consumption of processor $\mathcal{P}_k$ at time instance t, respectively. Then according to Fourier's law, the heat transfer process can be described as [16], [22], [23]

$$\begin{aligned} C_k \frac{\mathrm{d}T_k(t)}{\mathrm{d}t} = P_k(t) &- \sum_{\Theta_h \in \Theta} G_{k,h}(T_k(t) - T_h(t)) \\ &- \sum_{\mathcal{P}_m \in \mathcal{P}} G_{k,m}(T_k(t) - T_m(t)) \end{aligned} \tag{8}$$

$$C_h \frac{\mathrm{d}T_h(t)}{\mathrm{d}t} = -G_{\mathrm{amb}}(T_h(t) - T_{\mathrm{amb}}) - \sum_{\mathcal{P}_k \in \mathcal{P}} G_{k,h}(T_k(t) - T_h(t)) - \sum_{\Theta_\ell \in \Theta} G_{h,\ell}(T_h(t) - T_\ell(t)) \quad (9)$$

where $(\mathrm{d}T_k(t)/\mathrm{d}t)$ and $(\mathrm{d}T_h(t)/\mathrm{d}t)$ are derivatives of the temperature of processor $\mathcal{P}_k$ and heat sink Θ_h, respectively. As shown in [16], [22], and [23], all these thermal parameters can be derived using the RC thermal model for a given platform.

III. Overview of Energy Minimization

The focus of this paper is to minimize the energy consumption of the concerned MPSoC system in a schedule duration (SD) under the constraints of real-time task deadlines and temperature limit. In this section, we first show the preliminary for estimating leakage energy consumption, then present the calculation of system overall energy consumption and define the energy minimization problem. Finally, we present the framework of our solution to solve the problem.

A. Preliminary for Leakage Energy Estimation

As the focus of this paper is to minimize the overall energy consumption of the concerned MPSoC system, developing a method that can rapidly and accurately estimate the system energy consumption is of the top priority. However, this is challenging since derivation of leakage energy consumption is difficult. As introduced in Section II-B, leakage power varies with temperature and temperature is changing with time. Here the temperature refers to the operating temperature of processors since heat sinks generate no power. Either using (8) to compute the temperature at every time instance is computationally expensive or using thermal modeling tool (e.g., HotSpot [26]) to obtain the temperature profiles is time consuming. Some early works such as [27]–[30] either simply assume leakage power as a constant or totally ignore it since leakage energy consumption used to be a small part of overall energy consumption. However, with the continuous scaling of integrated circuits, the proportion of leakage in overall power dissipation is ever-increasing such that these simplistic energy models can lead to large estimation errors.

Therefore, to take into account both accuracy and computational cost of leakage energy estimation, we adopt a compromised method that divides an SD into multiple small intervals with equal length. During every interval, the operating temperature is assumed to be constant such that the leakage power consumed in the interval can be readily derived. Specifically, let [0, SD] be the schedule duration and L be the length of every interval. Then [0, SD] can be discretized into R intervals $[0, L], [L, 2L], \ldots, [(r-1)L, rL], \ldots, [(R-1)L, RL]$, where $1 \le r \le R$ and $R = (\mathrm{SD}/L)$. Let $T_{k,r}^{\mathrm{Const}}$ denote the constant operating temperature of processor $\mathcal{P}_k$ during the interval $[(r-1)L, rL]$, then based on (7) and our assumption,

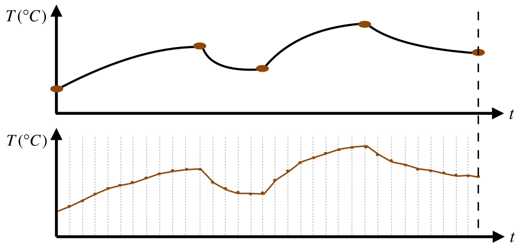


Fig. 2. Example of temperature curve with small variation in each interval.

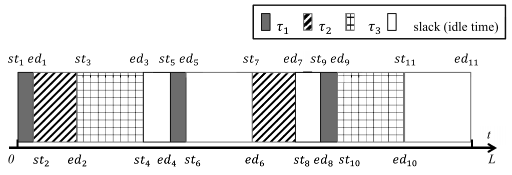


Fig. 3. Example of execution subintervals.

the leakage power of processor $\mathcal{P}_k$ during $[(r-1)L, rL]$ is

$$P_{\mathrm{leak}}(k, r) = \left(\alpha_k + \beta_k T_{k,r}^{\mathrm{Const}}\right) \cdot v_k. \quad (10)$$

This method is similar to that in [14] and is motivated by an observation illustrated in Fig. 2, that is, the temperature variation is small during each interval. Obviously, as long as the length of such interval is sufficiently small, the accuracy of this method can be very high.

When we assume the operating temperature during an interval is constant, one immediate question is what temperature should be selected for the leakage energy calculation. Since leakage is becoming the dominant source of power dissipation as the semiconductor technology advances toward the deep submicrometer era, we select the peak temperature occurring in the interval as the operating temperature. Now we only need to focus on how to obtain the peak temperature of a single interval. We take the first interval $[0, L]$ as an example. The execution of tasks on a processor during the interval $[0, L]$ can be depicted using a sequence of execution subintervals, where the start time and end time of the qth subinterval is denoted by $[\mathrm{st}_q, \mathrm{ed}_q]$. Fig. 3 shows an example of execution subintervals, where three tasks are arranged to execute on a processor and 11 execution subintervals are produced.

It has been shown in [31] that the peak temperature can be reached at the start/end times of execution subintervals since the temperature during each subinterval is monotonically increasing or decreasing. Thus, the peak temperature of tasks on processor $\mathcal{P}_k$ in the interval $[0, L]$ can be given as

$$T_{k,1}^{\mathrm{peak}} = \max\{T_k(t) | t = \mathrm{st}_1, \mathrm{ed}_1, \mathrm{st}_2, \mathrm{ed}_2, \ldots, L\}$$

where $\mathrm{st}_1 = 0$. Since the start time of a subinterval is the end time of its previous subinterval, that is, $\mathrm{st}_q = \mathrm{ed}_{q-1}$, the peak temperature in the interval $[0, L]$ is updated to

$$T_{k,1}^{\mathrm{peak}} = \max\{T(t) | t = \mathrm{st}_1, \mathrm{st}_2, \ldots, L\}. \quad (11)$$

Applying this method to the following intervals, the peak temperature of R intervals are derived, and the operating temperature of these intervals are hence obtained using that

$T_{k,r}^{\text{Const}} = T_{k,r}^{\text{peak}}$ for $1 \le r \le R$. Then the calculation of leakage power is updated to

$$P_{\text{leak}}(k,r) = \left(\alpha_k + \beta_k T_{k,r}^{\text{peak}}\right) \cdot v_k. \tag{12}$$

Using (12), we can compute the leakage energy consumption.

B. Calculation of System Overall Energy Consumption

The N real-time tasks in set Γ are assigned to M processors in set $\mathcal{P}$. In other words, a given task set Γ is partitioned into M subsets $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_k, \ldots, \Gamma_M\}$, where Γ_k is the subset of tasks assigned to processor $\mathcal{P}_k$. The leakage power is always consumed to maintain basic circuits and can be only eliminated by turning off the system, and the dynamic power is only consumed when executing tasks. Let $E_{\text{SD}}^{\text{tot}}$ represent the total energy consumption of M processors in an SD, then based on (1), (4), and (12), it can be computed as

$$\begin{aligned} E_{\text{SD}}^{\text{tot}} &= \sum_{k=1}^{M} \sum_{\tau_i \in \Gamma_k} C_k^{\text{eff}} \mu_i v_k^2 s_k \cdot \frac{c_i}{s_k} \cdot \frac{\text{SD}}{p_i} \\ &\quad + \sum_{k=1}^{M} \left(\alpha_k v_k \text{SD} + \beta_k v_k L \sum_{r=1}^{R} T_{k,r}^{\text{peak}} \right) \\ &= \sum_{k=1}^{M} \left(C_k^{\text{eff}} v_k^2 \sum_{\tau_i \in \Gamma_k} \frac{\mu_i c_i}{p_i} \right) \text{SD} \\ &\quad + \sum_{k=1}^{M} \left(\alpha_k v_k \text{SD} + \beta_k v_k L \sum_{r=1}^{R} T_{k,r}^{\text{peak}} \right) \end{aligned} \tag{13}$$

where the first term is the dynamic energy consumption and the second term is the static energy consumption.

The expression $\sum_{k=1}^{M}(C_k^{\text{eff}} v_k^2 \sum_{\tau_i \in \Gamma_k}(\mu_i c_i / p_i))$ in the first term of (13) is essentially the overall dynamic power consumption. Clearly, the dynamic energy consumption is minimal if the overall dynamic power consumption $\sum_{k=1}^{M}(C_k^{\text{eff}} v_k^2 \sum_{\tau_i \in \Gamma_k}(\mu_i c_i / p_i))$, denoted by $\mho_{\text{metric}}$, is minimized. The $\mho_{\text{metric}}$ is in fact an energy metric to estimate the dynamic energy consumption of the MPSoC system. It can be formulated into the product of vectors, that is

$$\begin{aligned} \mho_{\text{metric}}\left(A^M, B^M\right) &= A^M \times B^M \\ &= A_1 b_1 + A_2 b_2 + \cdots + A_M b_M \end{aligned} \tag{14}$$

where $A^M = [A_1, A_2, \ldots, A_k, \ldots, A_M]$ and $B^M = [b_1, b_2, \ldots, b_k, \ldots, b_M]^T$. A^M captures processor dependent parameters, where $A_k = C_k^{\text{eff}} v_k^2$ is referred to as the power dissipation factor of processor $\mathcal{P}_k$. B^M captures task related parameters, where $b_k = \sum_{\tau_i \in \Gamma_k} \delta_i = \sum_{\tau_i \in \Gamma_k}(\mu_i c_i / p_i)$ is referred to as the power dissipation factor of subset Γ_k, and δ_i is the power dissipation factor of task τ_i. A^M is determined since C_k^{eff} and v_k are known for a given MPSoC system, while B^M is not determined and depends on task assignment. For a given set Γ of N real-time tasks, the sum of power dissipation factors of all tasks, denoted by $Y(\Gamma)$, can be calculated as

$$Y(\Gamma) = \sum_{i=1}^{N} \delta_i = \sum_{k=1}^{M} \sum_{\tau_i \in \Gamma_k} \frac{\mu_i c_i}{p_i} = \sum_{k=1}^{M} b_k = Y_0 \tag{15}$$

where $Y(\Gamma)$ is constant for a given set Γ and is denoted by Y_0.

The expression $\sum_{k=1}^{M}(\alpha_k v_k \text{SD} + \beta_k v_k L \sum_{r=1}^{R} T_{k,r}^{\text{peak}})$ in the second term of (13) is essentially the overall leakage energy consumption in the duration SD. In this expression, α_k, β_k, and v_k are constants for a given processor $\mathcal{P}_k$, and SD and L are parameters decided by the scheduler. Thus, the leakage energy consumption only depends on $T_{k,r}^{\text{peak}}$, which is the peak temperature of processor $\mathcal{P}_k$ during $[(r-1)L, rL]$. Obviously, the overall leakage energy consumption is minimal if the peak temperature of processors in every interval are minimized.

C. Energy Minimization Problem

As analyzed above, it is clear that the system dynamic energy consumption depends on the task assignment, and the system leakage energy consumption depends on the peak temperature of intervals. Thus, both energy-efficient task assignment and temperature-aware task scheduling are helpful to minimize the system overall energy consumption. In this paper, we propose a task assignment and scheduling scheme to address the problem of minimizing the system overall energy consumption under the real-time and thermal constraints.

1) Real-Time Constraint: In a real-time system, each task should be finished before its deadline. Suppose that the execution of real-time tasks in the system is preemptable, and the task with a smaller period has a higher priority. Let $\text{RT}(i,k)$ denote the worst case response time of task τ_i at the supply voltage/speed (v_k, s_k), then it can be formulated as

$$\text{RT}(i,k) = \text{ET}(i,k) + \sum_{\tau_j \in \Gamma_k, p_j < p_i} \left\lceil \frac{\text{RT}(i,k)}{p_j} \right\rceil \times \text{ET}(j,k) \tag{16}$$

where $\text{ET}(i,k)$ and $\text{ET}(j,k)$ are the execution time of task τ_i and τ_j, respectively. They both can be obtained using (1). τ_j has a higher priority than τ_i for $j < i$, and $\lceil(\text{RT}(i,k)/p_j)\rceil$ indicates the number of instances of τ_j during time interval $\text{RT}(i,k)$.

2) Thermal Constraint: The temperature of the chip 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} denote the peak temperature at any position on the chip during the SD, that is

$$T_{\text{peak}} = \max\{T(t) | \forall t \in [0, \text{SD}]\}. \tag{17}$$

Here $T(t)$ can be the temperature of processors and heat sinks at time instance t. The system is deemed to be in a safe mode when the T_{peak} is below the threshold temperature $T_{\max}$.

3) Problem Definition: Considering the above design constraints, task allocation and scheduling problem of concerned MPSoC systems is defined as the following. Given a set Γ of N periodic real-time tasks and a set $\mathcal{P}$ of M heterogeneous processors, derive a task allocation and scheduling scheme to minimize the system overall energy consumption in an SD while satisfying the timing and thermal constraints. In other words, the problem can be formulated as

$$\begin{aligned} \text{Minimize:} \quad & E_{\text{SD}}^{\text{tot}} \\ \text{subject to:} \quad & \text{RT}(i,k) \le D_i \\ & T_{\text{peak}} \le T_{\max}. \end{aligned}$$

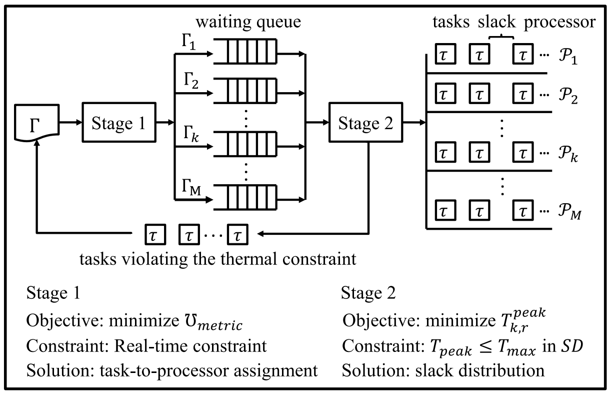

waiting queue
tasks slack processor
Γ
Stage 1
Γ_1
Γ_2
Γ_k
Γ_M
Stage 2
$\mathcal{P}_1$
$\mathcal{P}_2$
$\mathcal{P}_k$
$\mathcal{P}_M$
τ
tasks violating the thermal constraint
Stage 1
Objective: minimize $\mho_{metric}$
Constraint: Real-time constraint
Solution: task-to-processor assignment
Stage 2
Objective: minimize $T_{k,r}^{peak}$
Constraint: $T_{peak} \le T_{max}$ in SD
Solution: slack distribution

Fig. 4. Framework of the proposed two-stage solution.

D. Framework of Our Two-Stage Solution

We propose a static two-stage task allocation and scheduling scheme to solve the above problem. As shown in Fig. 4, the proposed scheme is implemented in two stages. In the first stage, for a given task set Γ, the proposed scheme partitions tasks into M subsets and assigns them to corresponding processors, in order to minimize the system dynamic energy consumption (characterized by $\mho_{\text{metric}}$). In the second stage, for the subset assigned to each processor, the proposed scheme distributes available slack on the processor to local tasks for reducing the peak temperature $T_{k,r}^{\text{peak}}$ of every interval, in order to minimize the system leakage energy consumption. Through the efforts made in two stages, the system overall energy consumption is minimized.

Feasibility analysis techniques are introduced in two stages to ensure timing and thermal constraints are met. Specifically, an RTFA technique is adopted in task allocation to check if the task deadlines are satisfied. A temperature feasibility analysis (TFA) technique is used in task scheduling to verify the thermal constraint in the SD. If the peak temperature limit is violated, the tasks that violates the thermal constraint are moved to task set for reallocation. The proposed task assignment strategy and slack distribution policy are detailed in Sections IV and V, respectively.

IV. Our Task Assignment Strategy

This section analyzes the dynamic energy optimality of assigning tasks to multiple processors, presents a proposition on optimum task assignment, and develops a task-to-processor assignment heuristic based on the proposition.

A. Analysis of the Optimality of Task-to-Processor Assignment

The system dynamic energy consumption is in fact estimated by the dynamic energy metric $\mho_{\text{metric}}(A^M, B^M)$, which can be minimized by optimally assigning N real-time tasks to M processors. Since the vector $A^M = [A_1, A_2, \dots, A_M]$ is constant and independent of task-to-processor partition strategies, the dynamic energy metric $\mho_{\text{metric}}(A^M, B^M)$ is determined by the vector $B^M = [b_1, b_2, \dots, b_M]^T$, which varies with different task-to-processor partition strategies. Specifically, a given set Γ of N real-time tasks can be partitioned into M subsets $\{\Gamma_1, \Gamma_2, \dots, \Gamma_M\}$, where $\Gamma_k (1 \le k \le M)$ indicates the subset of tasks assigned to processor $\mathcal{P}_k$. It is clear that there are M^N instances of partitioning. In other words, assigning tasks to processors is essentially a combinatorial optimization problem. The target of combinatorial optimization problem is to find the optimum solution from all feasible solutions. Let $\Upsilon = \{\gamma_1, \gamma_2, \dots, \gamma_n\}$ be a solution space and $f(\gamma_i)$ be the value of the objective function corresponding to the solution γ_i, then the combinatorial optimization problem involves finding the optimum solution γ^* such that $f(\gamma^*) = \min f(\gamma_i)$ holds for $\forall \gamma_i \in \Upsilon$. Since the combinatorial optimization problem is known to be NP-hard [32], the concerned task assignment problem is also NP-hard, which motivates the proposed suboptimal task-to-processor assignment heuristic.

For a given MPSoC system, the dynamic power dissipation of processor set $\mathcal{P}$ is characterized by a vector $A^M = [A_1, A_2, \dots, A_M]$. For the sake of easy presentation, it is assumed that $A_1 \le A_2 \le \cdots \le A_M$ holds. Similarly, the optimum power dissipation of subsets assigned to individual processors can be characterized by a vector $B^M = [B_1, B_2, \dots, B_M]^T$, indicating that the optimum task assignment solution can minimize the objective $\mho_{\text{metric}}(A^M, B^M)$. The sum of power dissipation factors of all assigned tasks is $Y_0 = B_1 + B_2 + \cdots + B_M$, as shown in (15). This optimal task assignment solution minimizes the dynamic energy metric $\mho_{\text{metric}}(A^M, B^M)$ by correlating the task assignment with processor power dissipation factors, as described below.

Proposition 1: If dynamic energy metric $\mho_{\text{metric}}(A^M, B^M)$ is minimized when $A^M = [A_1, A_2, \dots, A_M]$ $(A_1 \le A_2 \le \cdots \le A_M)$, $B^M = [B_1, B_2, \dots, B_M]^T$, and $B_1 + B_2 + \cdots + B_M = Y_0$, then the inequality $B_1 \ge B_2 \ge \cdots \ge B_M$ holds.

Proof: The proposition states that for an optimum task assignment solution, the processor with smaller power dissipation factor ends up with the subset of its assigned tasks having a larger power dissipation factor. As given in the proposition, the dynamic energy metric $\mho_{\text{metric}}(A^M, B^M)$ is minimized when $A^M = [A_1, A_2, \dots, A_i, \dots, A_j, \dots, A_M]$ $(A_1 \le A_2 \le \cdots \le A_i \le \cdots \le A_j \le \cdots \le A_M)$, $B^M = [B_1, B_2, \dots, B_i, \dots, B_j \dots, B_M]^T$, and $B_1 + B_2 + \cdots + B_i + \cdots + B_j + \cdots + B_M = Y_0$, then the inequality $B_1 \ge B_2 \ge \cdots \ge B_i \ge \cdots \ge B_j \ge \cdots \ge B_M$ holds.

Let $\mho_{\text{metric}}(A^M, B^M)'$ be the dynamic energy metric where the position of exactly two elements in B^M is exchanged. Assume that the position of B_i and B_j $(i < j)$ is exchanged for $\mho_{\text{metric}}(A^M, B^M)'$, then B^M becomes $[B_1, B_2, \dots, B_{i-1}, B_j, B_{i+1}, \dots, B_{j-1}, B_i, B_{j+1}, \dots, B_M]^T$ in this case. According to the definition of dynamic energy metric in (14), $\mho_{\text{metric}}(A^M, B^M) = A_1B_1 + A_2B_2 + \cdots + A_iB_i + \cdots + A_jB_j + \cdots + A_MB_M$ and $\mho_{\text{metric}}(A^M, B^M)' = A_1B_1 + A_2B_2 + \cdots + A_iB_j + \cdots + A_jB_i + \cdots + A_MB_M$. Since $\mho_{\text{metric}}(A^M, B^M)$ is the optimum, $\mho_{\text{metric}}(A^M, B^M)' - \mho_{\text{metric}}(A^M, B^M) = (A_i - A_j)(B_j - B_i) \ge 0$. It is known that $A_i \le A_j$ for $i < j$, then $B_i \ge B_j$ for $i < j$ is derived.

Given the optimum task assignment solution $B^M = [B_1, B_2, \dots, B_M]^T$ that minimizes the dynamic energy metric

$\mho_{\text{metric}}(A^M, B^M)$, any feasible solution in the solution space can be obtained by exchanging elements in $B^M = [B_1, B_2, \ldots, B_M]^T$ multiple times. In each iteration of the exchange, it can be deduced that $B_i \geq B_j$ holds for $i < j$. In other words, the dynamic energy metric $\mho_{\text{metric}}(A^M, B^M)$ is minimized when the processor with smaller power dissipation factor ends up with the subset of its assigned tasks having a larger power dissipation factor. The proposition is proved. ■

B. Task-to-Processor Assignment Heuristic

As described in Section IV-A, assigning tasks to individual processors is an NP-hard problem, which necessitates a task assignment scheme that observes the proposition presented in Section IV-A. Specifically, tasks in the subset with the maximum power dissipation factor is assigned to the processor with the minimum power dissipation factor, and tasks in the subset with the next maximum power dissipation factor is assigned to the processor with the next minimum power dissipation factor. This process repeats until all subsets of tasks are assigned to individual processors. Once a task-to-processor assignment is generated, the slack available on individual processors is distributed among local tasks. The details of the task assignment heuristic are given in Algorithm 1.

Algorithm 1 essentially partitions the tasks in the given set Γ into subsets, then assigns subsets of selected tasks to individual processors in set $\mathcal{P}$. The algorithm aims at minimizing the system dynamic energy consumption under the timing constraint. It is motivated by the proposition presented in Section IV-A, that is, assigning the subset having a larger power dissipation factor to the processor having a smaller power dissipation factor can minimize the system dynamic energy consumption. Since the M processors in set $\mathcal{P}$ are sorted in the nondecreasing order of processor power dissipation factors, the focus of the algorithm becomes to derive a task-to-processor assignment that partitions tasks into M subsets, arranged in the nonincreasing order of subset power dissipation factors, then assigns them to corresponding processors. This can be achieved by assigning tasks with larger task power dissipation factors to processors with smaller processor power dissipation factors. In addition, task deadlines are examined to meet the real-time constraint for each task assignment.

Algorithm 1: Energy-Efficient Task-to-Processor Assignment Under the Real-Time Constraint

```
Input: task set $\Gamma$, processor set $\mathcal{P}$, ambient temperature $T_{\text{amb}}$,
       and temperature limit $T_{max}$
1  initialization: $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_M\} \leftarrow \{\varnothing, \varnothing, \ldots, \varnothing\}$,
   $T_{init} \leftarrow T_{amb}$, and $k \leftarrow 1$;
2  while $\Gamma \neq \varnothing$ and $k \leq M$ do
3      calculate task power dissipation factor $\delta_i$ of every task $\tau_i$ in
       $\Gamma$ according to $\delta_i = \frac{\mu_i c_i}{p_i}$;
4      sort $\tau_i \in \Gamma$ in the non-increasing order of $\delta_i$;
5      create a temporary subset $\Gamma_{tem}$;
6      for $i = 1$ to $sizeof(\Gamma)$ do      /* use First-Fit to
       group tasks into subsets */
7          $\Gamma_{tem} = \Gamma_k + \tau_i$;
8          if (RTFA($\Gamma_{tem}, k$) == true) then
9              $\Gamma_k = \Gamma_k + \tau_i$;
10             $\Gamma = \Gamma - \tau_i$;
11     assign the slack to tasks in subset $\Gamma_k$ and check the
       thermal constraint using Algorithm 2;
12     $k \leftarrow k + 1$;
13 if $\Gamma \neq \varnothing$ and $k > M$ then
14     exit(1);      /* the tasks in set $\Gamma$ cannot be
       feasibly scheduled under the timing and
       thermal constraints */
15 else if $\Gamma = \varnothing$ and $k < M$ then
16     power off the vacant processors in $\mathcal{P}$ to save energy;
17 return the target schedule $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_M\}$;
   Procedure RTFA($\Gamma_{tem}, k$)
18 flag = true;
19 for $i = 1$ to $sizeof(\Gamma_{tem})$ do
20     calculate the worst case response time $RT(i, k)$ using (16);
21     if $RT(i, k) > D_i$ then
22         flag = false;
23         break;
24     return flag;
```

The pseudo code of our task assignment heuristic is given in Algorithm 1. Inputs to the algorithm are task set Γ, processor set $\mathcal{P}$, ambient temperature T_{amb}, and temperature limit $T_{\max}$. Line 1 of the algorithm initializes subsets $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_M\}$ to $\{\varnothing, \varnothing, \ldots, \varnothing\}$, chip initial temperature T_{init} to ambient temperature T_{amb}, and index k to 1. Lines 2–12 iteratively implement the process of task assignment and scheduling if the task set Γ is not empty and not all processors in $\mathcal{P}$ have been considered. In each round of iteration, the tasks in subset Γ_k assigned to processor $\mathcal{P}_k$ are determined in lines 3–10. More specifically, lines 3 and 4 calculate the task power dissipation factor δ_i of every task τ_i in set Γ and sort the tasks in the decreasing order of δ_i. Line 5 creates a temporary subset Γ_{tem}. Lines 6–10 iteratively assign tasks in set Γ to processor $\mathcal{P}_k$ and construct subset Γ_k of tasks in a first-fit manner according to the schedulability requirement. The task with larger task power dissipation factor has higher priority when assigned to the processor. The temporary subset Γ_{tem} is used to facilitate the timing feasibility analysis of assigning task τ_i to processor $\mathcal{P}_k$ (line 7). If the assignment can satisfy the real-time constraint, the task is assigned to the processor, and both subset Γ_k and set Γ are updated (lines 8–10). The procedure then moves to the next iteration and considers the allocation of the next task in set Γ. Otherwise, the task is not assigned and the procedure directly moves to the next iteration. The slack available on processor $\mathcal{P}_k$ is assigned to tasks in subset Γ_k under the thermal constraint using Algorithm 2 (line 11). The process continues until a feasible schedule is generated for the system. If there is no feasible schedule for the system under the constraints, the algorithm exits (lines 13 and 14). When the task assignment is finished, if the system still has some vacant processors, these vacant processors are powered off for energy savings (lines 15 and 16). The target schedule is returned in line 17. RTFA is called in line 8 to check if the timing design constraint is satisfied. If the response time $\text{RT}(i, k)$ of task τ_i exceeds the deadline D_i, τ_i cannot be feasibly assigned to processor $\mathcal{P}_k$ (lines 21–23).

V. Our Slack Distribution Policy

Real-time tasks in a given set Γ are assigned to individual processors using Algorithm 1 for reducing the dynamic energy consumption. RTFA is conducted for the task assignment. Slack is the time when the processor is in the idle state, which is due to that tasks may not always take the worst-case execution time to finish and can complete earlier before the deadline. Using slack distribution can reduce processor peak temperature without increasing system dynamic or leakage energy consumption since slack distribution is in fact a rearrangement of the available slack time on the processor rather than introducing additional slacks. On the contrary, the temperature-dependent leakage energy consumption can be reduced due to the improved temperature profiles achieved by slack distribution. In this section, we focus on the design of temperature-aware slack distribution policy for minimizing the system leakage energy consumption. Thermal feasibility analysis is conducted for the slack distribution.

A. Slack Assignment to Reduce Peak Temperature

From the thermal model introduced in Section II-C and the leakage energy calculation analysis given in Section III-A, it is advantageous to do slack distribution under thermal steady state. This is because even if we discretize the SD into a large number of small intervals, transient thermal analysis may still be too costly [23]. It has been shown in [23] that steady state thermal analysis can rapidly and accurately predict the temperature when task execution times are long compared to the thermal time constant of the processors; otherwise, it may lead to overestimated peak temperature when task execution times are short relative to the processor thermal time constants. Under the thermal steady state, we prove that the temperature profiles can be improved if all tasks on the processor assume a uniform steady state temperature. We then discuss the policy for slack distribution which can effectively reduce the processor peak temperature.

Proposition 2: Under the thermal steady state, the peak temperature of tasks on a processor is minimal if all tasks assume a uniform steady state temperature.

Proof: Suppose that $T_{\text{std}}(i,k)$ is the steady state temperature of task τ_i on processor $\mathcal{P}_k$, which is formulated as [31]

$$T_{\text{std}}(i,k) = P_{\text{std}}(i,k) \times R_k + T_{\text{amb}} \tag{18}$$

where $P_{\text{std}}(i,k)$ is the power consumption in the steady state and can be treated as a constant since temperature becomes steady. R_k is the thermal resistance of $\mathcal{P}_k$ and T_{amb} is the ambient temperature. Both of them are known. Thus, for the subset Γ_k, we can conclude that $\sum_{i=1}^{z} T_{\text{std}}(i,k)$ is a constant, where $z = \text{sizeof}(\Gamma_k)$. In the thermal steady state, the peak temperature of tasks are no more than their steady state temperature [23] so that the peak temperature of tasks on processor $\mathcal{P}_k$ is $\max\{T_{\text{std}}(1,k), T_{\text{std}}(2,k), \ldots, T_{\text{std}}(z,k)\}$. Since $\sum_{i=1}^{z} T_{\text{std}}(i,k)$ is a constant, it is easy to see that $\max\{T_{\text{std}}(1,k), T_{\text{std}}(2,k), \ldots, T_{\text{std}}(z,k)\}$ is minimal iff $T_{\text{std}}(1,k) = T_{\text{std}}(2,k) = \cdots = T_{\text{std}}(z,k)$ holds. The proposition is proved. ∎

The discussion above states that the peak temperature of a processor in the steady state is minimal if all tasks on the processor assume a uniform steady state temperature, which motivates the proposed slack assignment heuristic that balances steady state temperatures of tasks on the processor through slack distribution. With the improved temperature profiles, the temperature-dependent leakage energy consumption of the system is then reduced, as analyzed in Section III-B.

Let sl_i^* be the optimal slack allocated to task τ_i on processor $\mathcal{P}_k$ for temperature balance, then the average steady power consumption $\bar{P}_{\text{std}}(i,k)$ of task τ_i during its execution time and slack time is given by

$$\bar{P}_{\text{std}}(i,k) = \frac{P_{\text{leak}}^{\text{std}}(k) \times \left(\frac{c_i}{s_k} + \text{sl}_i^*\right) + P_{\text{dyn}}(i,k) \times \frac{c_i}{s_k}}{\frac{c_i}{s_k} + \text{sl}_i^*}$$

where $P_{\text{leak}}^{\text{std}}(k)$ is the leakage power in steady state, $P_{\text{dyn}}(i,k)$ is the dynamic power, and (c_i/s_k) is the task execution time.

Let $T_{\text{std},k}$ be the uniform steady state temperature of tasks on processor $\mathcal{P}_k$, i.e., $T_{\text{std}}(i,k) = T_{\text{std},k}$ holds for $\forall \tau_i \in \Gamma_k$. Given the steady state temperature and power consumption of task τ_i, the optimal slack assigned to the task can be derived by substituting $T_{\text{std}}(i,k) = T_{\text{std},k}$ and $P_{\text{std}}(i,k) = \bar{P}_{\text{std}}(i,k)$ into (18), and is written as

$$\text{sl}_i^* = \frac{P_{\text{dyn}}(i,k) \times R_k \times c_i}{\left(T_{\text{std},k} - T_{\text{amb}} - P_{\text{leak}}^{\text{std}}(k) \times R_k\right) \times s_k} - \frac{c_i}{s_k}. \tag{19}$$

In (19), $P_{\text{leak}}^{\text{std}}(k)$ is dependent upon $T_{\text{std},k}$, and other terms are constants either dependent upon the processor or the task, which indicates that sl_i^* is determined by $T_{\text{std},k}$. Thus, the key of solving (19) is to derive the uniform steady state temperature of tasks on processor $\mathcal{P}_k$.

It has been shown in [16], [22], and [23] that the uniform steady state temperature of tasks on the processor is derived when the processor reaches the steady state condition $((\text{d}T_k(t)/\text{d}t) = 0)$. Hence we can obtain the uniform steady state temperature $T_{\text{std},k}$ of each processor $\mathcal{P}_k$ by substituting the condition $(\text{d}T_k(t)/\text{d}t) = 0$ into (8), which are given as [16], [22], [23]

$$\begin{bmatrix} \Omega_{1,1} & \cdots & \Omega_{1,M} \\ \Omega_{2,1} & \cdots & \Omega_{2,M} \\ \vdots & \vdots & \vdots \\ \Omega_{M,1} & \cdots & \Omega_{M,M} \end{bmatrix} \begin{bmatrix} T_{\text{std},1} \\ T_{\text{std},2} \\ \vdots \\ T_{\text{std},M} \end{bmatrix} = -\begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_M \end{bmatrix}. \tag{20}$$

For any $1 \le k \ne m \le M$, $\Omega_{k,k} = \beta_k v_k - \sum_{h=1}^{\mathcal{H}} G_{k,h} - \sum_{m=1}^{M} G_{k,m}$, $\Omega_{k,m} = G_{k,m}$, and $\Psi_k = \alpha_k v_k + C_k^{\text{eff}} v_k^2 s_k$.

The optimal slack sl_i^* given in (19) is derived under the assumption that all the slack available on processor $\mathcal{P}_k$ is assigned to tasks in subset Γ_k. Note that the slack assigned to a task is used to cool down the processor. Similar to the scenario that assigning slack to a task to slow down the processor increases the response time of successive tasks, assigning slack to a task to cool down the processor will lead to an increase in the response time of successive tasks. Therefore, there exists a slack bound for a task beyond which the task will miss its deadline. Let $\text{sl}_{i,\max}$ be the maximum amount of slack that can be assigned to task τ_i without violating the timing constraint, then the slack actually assigned to task τ_i is given as $\mathrm{sl}_i = \min\{\mathrm{sl}_i^*, \mathrm{sl}_{i,\max}\}$. The next section describes the slack assignment heuristic in detail.

B. Temperature-Aware Slack Assignment Heuristic

Based on the proposed slack distribution policy, the slack assignment heuristic is developed to reduce the peak temperature of processors. The details of the heuristic are given in Algorithm 2. The algorithm iteratively assigns slack to tasks in subset Γ_k, and moves the tasks that violate thermal constraint to set Γ for reallocation. It takes as input subset Γ_k that is generated by Algorithm 1, and an arbitrarily small positive number ϵ. In each round of iteration, the optimal slack sl_i^* of task τ_i used for temperature minimization is first calculated using (19) (line 2). Then the maximum slack $\mathrm{sl}_{i,\max}$ that could be assigned to τ_i is derived using procedure SLAK($\Gamma_k, \tau_i, \epsilon$) (line 3). The slack $\mathrm{sl}_i = \min\{\mathrm{sl}_i^*, \mathrm{sl}_{i,\max}\}$ is assigned to τ_i, and the task execution time is hence updated (line 4). This process repeats until all tasks in Γ_k are examined. After the slack assignment is finished, a TFA procedure is used to evaluate the thermal feasibility of the resultant task schedule, and those tasks that violate the thermal constraint are sent back to set Γ and considered to be allocated to the next processors as well as unassigned tasks.

Procedure SLAK derives the maximal slack for a task in a binary search-based manner. Inputs to the procedure are task τ_i, subset Γ_k, and the arbitrarily small positive number ϵ. A search space $[\mathrm{sl}_l, \mathrm{sl}_h]$ is defined and initialized to $[0, D_i - \mathrm{RT}(i,k)]$, where sl_l and sl_h are the lower and upper bound of the space, and D_i and $\mathrm{RT}(i,k)$ are the deadline and response time of τ_i, respectively (line 7). The search length, denoted by ρ, is set to $\mathrm{sl}_h - \mathrm{sl}_l$ (line 8). Lines 9–16 describe the searching process. In each round of iteration, a dummy task τ_{tem} is created and initialized to τ_i, the median $\mathrm{sl}_{\mathrm{tem}}$ of search space 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 $\Gamma_k + \tau_{\mathrm{tem}}$. Procedure RTFA presented in Algorithm 1 is called to check if timing constraint of tasks in Γ_{tem} is met. The search space $[\mathrm{sl}_l, \mathrm{sl}_h]$ and length ρ are updated in each iteration, and the process stops when ρ is less than yet close enough to ϵ. The lower bound sl_l of the search space is returned as the maximum slack that could be assigned to task τ_i (line 17).

The chip temperature should be below a temperature limit, as described in (17) to avoid the temperature-induced failures. To check if the thermal constraint is satisfied, we need to know the temperature of processors and heat sinks. As discussed in Section V-A, transient thermal analysis is prohibitive due to its extremely expensive computation cost, and steady state thermal analysis is less costly but may result in overestimated peak temperature. Fortunately, it is safe to use steady state thermal analysis to check the thermal constraint since if the overestimated peak temperature is below the temperature limit, the actual peak temperature must be as well. Thus, obtaining the steady state temperature of processors and heat sinks becomes the focus. At the end of Section V-A, we show the derivation of processor steady state temperatures, as in (20). Similarly, we can obtain the steady state temperature $T'_{\mathrm{std},h}$ of each heat sink

Algorithm 2: Temperature-Aware Slack Assignment for Subset Γ_k Under the Thermal Constraint

```
   Input: subset $\Gamma_k$, an arbitrarily small positive number $\epsilon$
1  for $i = 1$ to $sizeof(\Gamma_k)$ do
2      calculate the optimal slack $sl_i^*$ of task $\tau_i$ using (19);
3      derive the maximum slack $sl_{i,max}$ for task $\tau_i$ using
       SLAK($\Gamma_k, \tau_i, \epsilon$);
4      allocate slack $sl_i = \min\{sl_i^*, sl_{i,max}\}$ to $\tau_i$ and update
       execution time $\mathrm{ET}(i,k) = \mathrm{ET}(i,k) + sl_i$;
5  if (TFA($\Gamma_k$) == false) then
6      move the tasks in $\Gamma_k$ that violate the thermal constraint to
       $\Gamma$ for re-allocation using Algorithm 1;

   Procedure SLAK($\Gamma_k, \tau_i, \epsilon$)        /* SLAK is a binary
   search-based method */
7  $[sl_l, sl_h] = [0, D_i - RT(i,k)]$;
8  $\rho = sl_h - sl_l$;
9  while ($\epsilon < \rho$) do
10     $sl_{tem} = (sl_l + sl_h)/2$;
11     $\tau_{tem} = \tau_i + sl_{tem}$, $\Gamma_{tem} = \Gamma_k + \tau_{tem}$;
12     if (RTFA($\Gamma_{tem}, k$) == true) then
13         $[sl_l, sl_h] = [sl_{tem}, sl_h]$;
14     else
15         $[sl_l, sl_h] = [sl_l, sl_{tem}]$;
16     $\rho = sl_h - sl_l$;
17 return $sl_l$;

   Procedure TFA($\Gamma_k$)
18 calculate the steady state temperature $T_{std,k}$ of $\mathcal{P}_k$ using (20);
19 calculate the steady state temperature $T'_{std,h}$ of $\Theta_h$ using (21);
20 if $T_{std,k} \le T_{max}$ and $T'_{std,h} \le T_{max}$ then
21     return true;
22 else
23     return false;
```

Θ_h by substituting the steady state condition $((\mathrm{d}T_h(t)/\mathrm{d}t) = 0)$ into (9), which are given as [16], [22], [23]

$$\begin{bmatrix} \Omega'_{1,1} & \cdots & \Omega'_{1,\mathcal{H}} \\ \Omega'_{2,1} & \cdots & \Omega'_{2,\mathcal{H}} \\ \vdots & \vdots & \vdots \\ \Omega'_{\mathcal{H},1} & \cdots & \Omega'_{\mathcal{H},\mathcal{H}} \end{bmatrix} \begin{bmatrix} T'_{\mathrm{std},1} \\ T'_{\mathrm{std},2} \\ \vdots \\ T'_{\mathrm{std},\mathcal{H}} \end{bmatrix} = - \begin{bmatrix} \Psi'_1 \\ \Psi'_2 \\ \vdots \\ \Psi'_{\mathcal{H}} \end{bmatrix}. \quad (21)$$

For any $1 \le h \ne \ell \le \mathcal{H}$, $\Omega'_{h,h} = -G_{\mathrm{amb}} - \sum_{k=1}^{M} G_{k,h} - \sum_{\ell=1}^{\mathcal{H}} G_{h,\ell}$, $\Omega'_{h,\ell} = G_{h,\ell}$, and $\Psi'_h = G_{\mathrm{amb}} T_{\mathrm{amb}}$. Based on the steady state thermal analysis, we can utilize the steady state temperature to verify the thermal feasibility of the resultant task schedule, as given in Procedure TFA (lines 18–23).

VI. Evaluation

Extensive simulation experiments have been conducted to validate the proposed scheme. The proposed scheme is compared with two benchmarking algorithms rate monotonic first fit (RMFF), rate monotonic best fit (RMBF) [33], and a state-of-the-art approach hybrid worst-fit genetic algorithm (HWGA) [16]. Benchmarking algorithms RMFF and RMBF [33] are taken as baseline schemes to exhibit the energy efficiency of the proposed algorithms. The two algorithms assign priority to tasks based on task periods. A task with shorter period has higher priority than a task with

TABLE I
PROCESSOR PARAMETERS AND CONSTANTS [21]

v (V)	s (GHz)	α	β	C^{eff}
0.85	0.8010	7.3249	0.1666	13.0
0.90	0.8291	8.6126	0.1754	12.0
0.95	0.8553	10.238	0.1846	14.0
1.00	0.8797	12.315	0.1942	15.0
1.05	0.9027	14.998	0.2043	17.0
1.10	1.0000	18.497	0.2149	16.0

longer period. RMFF is a partition algorithm that assigns the task with the highest priority to the first processor that can accommodate the task, while RMBF is a partition heuristic that assigns the task with the highest priority to the processor with smallest unused capacity among those processors on which it fits [33]. HWGA integrates a worst-fit based partition heuristic with the genetic algorithm to generate a task allocation that reduces the energy consumption while satisfying all system constraints [16]. The worst-fit based partition scheme assigns the task with the highest priority to the processor with maximum remaining capacity. For the sake of fair comparison, the same simulation settings are adopted for the proposed method and benchmarking algorithms RMFF, RMBF, and HWGA.

A. Experimental Settings

We perform our experimental simulations based on a 2×3 MPSoC system ($M = 6$). Our processor model is built on 65 nm technology [19], [21]. The supply voltage v, processing speed s, and constants α, β, and C^{eff} of six processors are listed in Table I. Four real-life benchmarks (task sets) from the Embedded System Synthesis Benchmark Suite [34] are utilized to validate the proposed scheme. The benchmarks are automotive-industrial, consumer-networking, telecom, and mpeg, which consist of 16, 20, 17, and 15 tasks, respectively. As its name indicates, each benchmark represents an application. The periods of tasks in applications are assumed to equal their deadlines. The task activity factors μ are uniformly distributed in the interval [0.4, 1], which demonstrates the heterogeneous nature of tasks [17]. We use HotSpot [26] to obtain the RC thermal model for the above platform. The floorplan and HotSpot parameters are given as follows. The number of processors is six, the area per processor is 4 mm^2, the die thickness is 0.15 mm, the heat spreader side is 20 mm, and the heat sink side is 30 mm. The average of thermal resistance and capacitance of processors are selected as 0.8 K/W and 340 J/K, respectively. The ambient temperature is set to 45 °C. Simulation experiments have been carried out under varying thermal constraints ($T_{\text{max}} = 60$ °C, 65 °C, 70 °C, and 75 °C) to verify the effectiveness of the proposed algorithms.

B. Simulation Results

1) Evaluation of the Accuracy and Efficiency of the Proposed Energy Estimation Method: We evaluate the accuracy and efficiency of the proposed energy estimation method, which is given in (13). Specifically, we test the performance of the proposed energy estimation method when the processor runs at different supply voltages and in SDs with varying lengths. The proposed energy estimation method is compared with the baseline approach presented in [14] from the aspects of energy consumption and computation cost. The baseline approach splits an SD into a series of small intervals and assumes the temperature (and hence the leakage power) in every interval is close to a constant. To achieve an accurate energy estimation, we let the length of every interval be small, that is, 0.01 s.

TABLE II
ACCURACY AND EFFICIENCY EVALUATION OF THE PROPOSED ENERGY ESTIMATION METHOD

SD (s)	v (V)	E_{Pro} (J)	E_{Bas} (J)	Err (%)	ACT_{Pro} (s)	ACT_{Bas} (s)	Spe ($\times$)
5	0.9	7.45	7.18	3.6	3.8	34	8.9
	1.0	13.78	13.29	3.5			
	1.1	24.01	23.25	3.2			
10	0.9	15.16	14.57	3.9	4.1	46	11.2
	1.0	29.74	28.84	3.0			
	1.1	52.48	50.93	2.9			
20	0.9	31.22	30.16	3.4	4.0	58	14.5
	1.0	60.57	58.94	2.7			
	1.1	99.85	96.91	2.9			

Let E_{Pro} and E_{Bas} denote the system energy consumption calculated by using the proposed method and baseline approach [14], respectively. $\text{Err} = (E_{\text{Pro}} - E_{\text{Bas}}/E_{\text{Pro}}) \times 100\%$ denotes the relative error of the proposed method when compared to the baseline approach in terms of system energy estimation. Let ACT_{Pro} and ACT_{Bas} denote the average CPU time consumed by the proposed method and baseline approach [14], respectively. $\text{Spe} = (\text{ACT}_{\text{Bas}}/\text{ACT}_{\text{Pro}})$ denotes the speedup achieved by the proposed method when compared to the baseline approach in terms of average CPU time. The simulation results given in Table II clearly show that the proposed estimation method is accurate and efficient. As can be seen in the table, the system energy consumption estimated by the proposed method is close to that of the baseline approach. The maximal relative error is no more than 3.9%. On the other hand, the proposed method can reduce the computational cost and achieves up to $14.5\times$ of speedup in terms of average CPU time.

2) Comparison of the Energy Consumption: We compare the proposed scheme with the methods RMFF, RMBF [33], and HWGA [16] in energy efficiency. The benchmarking methods RMFF and RMBF first arrange tasks in the order of increasing task periods, then allocate tasks to individual processors using the first fit and best fit heuristics. The state-of-the-art approach HWGA allocates tasks to individual processors using the genetic-algorithm worst-fit based heuristics. In the proposed scheme, processors are arranged in the order of increasing processor power dissipation factor while tasks are organized in the order of decreasing task power dissipation factor. Tasks with large power dissipation factor are then assigned to processors with small power dissipation factor, which has been proved to be able to minimize system dynamic energy consumption in Section IV-A. The available slack on the processor is allocated to tasks for achieving a uniform steady state temperature. Through this slack distribution, the peak temperature of tasks on the processor is minimized, as proved in Section V-A, then the temperature-dependent system leakage energy savings is hence maximized.

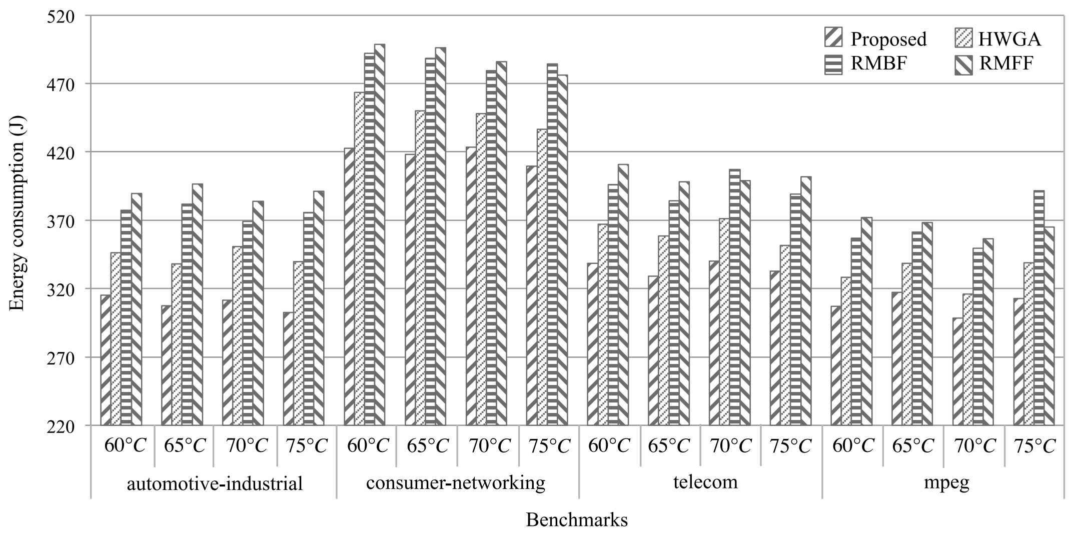


Fig. 5. Average energy consumption of benchmarks under four system thermal constraints using the proposed algorithm and three benchmarking schemes.

Fig. 5 shows the average energy consumed by the system when executing four benchmarks (automotive-industrial, consumer-networking, telecom, and mpeg) under four system thermal constraints using the proposed algorithm and three benchmarking schemes HWGA [16], RMBF, and RMFF [33]. The system thermal constraint takes the values of $T_{\max} = 60$ °C, 65 °C, 70 °C, and 75 °C. The energy consumption given in the figure is averaged over 1000 test instances. As can be seen in the figure, the proposed algorithm consumes the least energy for a given thermal constraint among the four algorithms. Specifically, the proposed algorithm achieves energy savings of up to 11.1%, 20.1%, and 23.3% as compared to benchmarking methods HWGA, RMBF, and RMFF, respectively. For example, for the scenario of benchmark automotive-industrial under the constraint $T_{\max} = 70$ °C, the energy consumption ($E_{\text{Pro}} = 311.7$ J) of the proposed scheme is 11.1% lower than that ($E_{\text{HWGA}} = 350.6$ J) of HWGA. For the scenario of benchmark mpeg under the constraint $T_{\max} =$ 75 °C, the energy consumption ($E_{\text{Pro}} = 312.7$ J) of the proposed algorithm is 20.1% lower than that ($E_{\text{RMBF}} = 391.6$ J) of RMBF. For the scenario of benchmark automotive-industrial under the constraint $T_{\max} = 75$ °C, the energy consumption ($E_{\text{Pro}} = 301.5$ J) of the proposed algorithm is 23.3% lower than that ($E_{\text{RMFF}} = 393.2$ J) of RMFF.

3) Comparison of the Schedule Feasibility: We also compare the proposed scheme with benchmarking algorithms HWGA [16], RMBF, and RMFF [33] from the aspects of schedule feasibility under different thermal constraints. In addition to the algorithm adopted, the schedule feasibility is affected by two factors, that is, the input benchmark (task set) assigned to processor set and the thermal constraint set by system designer. In the simulation, we adopt four benchmarks and the temperature limits $T_{\max}$ are set to 60 °C, 65 °C, 70 °C, and 75 °C. The feasibility is calculated as the ratio of the number of benchmark instances that can be feasibly scheduled to the total number of benchmark instances. The total number of benchmark instances employed in feasibility test is 1000.

The feasibility test results are given in Fig. 6. As shown in the figure, when thermal constraint is loose ($T_{\max} = 75$ °C), the feasibility of the proposed algorithm, HWGA, RMBF, and RMFF are nearly 100%. As expected, the feasibility of the four algorithms are decreasing with the increase in benchmark size for a given thermal constraint. For instance, for the scenario of $T_{\max} = 65$ °C, the feasibility rates achieved by HWGA in the case of benchmarks automotive-industrial ($N = 16$), consumer-networking ($N = 20$), telecom ($N = 17$), and mpeg ($N = 15$) are 95.2%, 92%, 94.5%, and 96%, respectively. This is because increasing benchmark size leads to heavier workload on processors, which may incur violation of thermal and timing constraints. It also has been demonstrated in Fig. 6 that for a given benchmark, the feasibility of the four algorithms decreases when a rigorous thermal constraint is applied.

The proposed algorithm outperforms benchmarking algorithms HWGA, RMBF, and RMFF in feasibility by up to 15%. For example, for the scenario of benchmark consumer-networking under the constraint $T_{\max} = 60$ °C, the feasibility of the proposed algorithm exceeds that of HWGA, RMBF, and RMFF by 6%, 11%, and 15%, respectively. This is primarily due to that the proposed algorithm allocates tasks to individual processors with considerations of timing and thermal constraints, while RMBF and RMFF performs task allocation without considering the thermal constraint. As compared to the state-of-the-art method HWGA, the proposed algorithm utilizes a thermal-aware slack assignment heuristic to improve processor temperature profiles and exploits feasibility analysis techniques to ensure the timeliness of the system.

4) Comparison of the Time Complexity: Due to the differences in the hardware platforms, it is difficult to directly compare running time with the three benchmarking algorithms. Therefore, we provide a time complexity analysis in this section. The time complexity of the proposed scheme and benchmarking methods HWGA [16], RMBF, and RMFF [33] are $O(M^2N^2)$, $O(\text{Max}^{\text{gen}} \cdot M^2N\log M)$, $O(MN\log N)$, and $O(MN\log N)$, respectively, where M is the

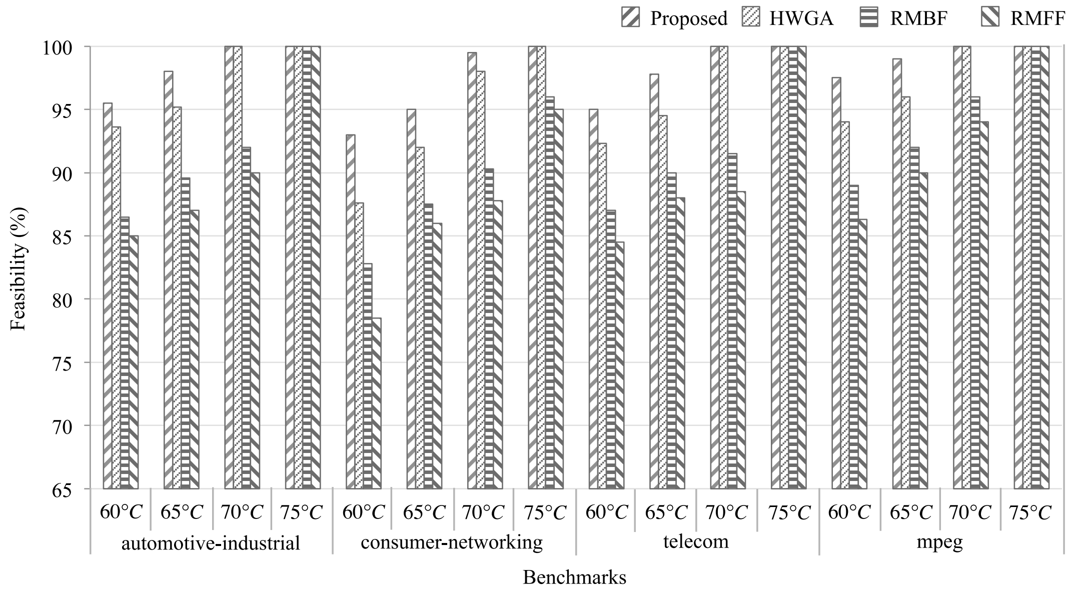


Fig. 6. Compare the proposed algorithm with benchmarking schemes HWGA [16], RMBF, and RMFF [33] in schedule feasibility.

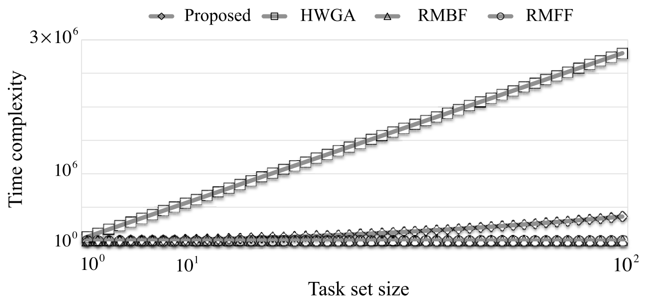


Fig. 7. Log–log plot of the time complexity of the proposed scheme and benchmarking methods HWGA [16], RMBF, and RMFF [33].

number of processors in the processor set, N is the number of tasks in the task set, and $\mathrm{Max}^{\mathrm{gen}}$ is the maximum number of generations for the genetic algorithm used in method HWGA.

Fig. 7 shows the log–log plot of time complexity for the proposed scheme and benchmarking methods HWGA, RMBF, and RMFF. The plot is generated based on the setting of $M = 6$ and $\mathrm{Max}^{\mathrm{gen}} = 1000$. It has been demonstrated in the figure that the time complexity of the proposed scheme is much lower when compared to that of the state-of-the-art method HWGA, and is close to that of benchmarking methods RMBF and RMFF. The reason why RMBF and RMFF have the lowest time complexity is that they do not take into account the thermal constraint and thermal control, which adversely impacts their performance in schedule feasibility, as the results in Section VI-B3. However, the proposed scheme can not only achieve a similar low time complexity, but also has a high schedule feasibility, as the results in Sections VI-B3 and VI-B4.

VII. Conclusion

This paper proposes a task allocation scheme and a slack assignment policy for heterogeneous real-time MPSoC systems. The proposed task allocation scheme minimizes the system dynamic energy consumption by assigning tasks to individual processors in the way that the processor with a small power dissipation factor ends up with allocated tasks in a subset having a large power dissipation factor. The proposed slack assignment policy that reduces the system leakage energy consumption by improving processor temperature profiles, is motivated by the observation that the peak temperature of tasks on a processor is minimal if tasks assume a uniform steady state temperature. Feasibility analysis techniques are utilized to ensure the timing and thermal constraints can be satisfied.

The proposed energy estimation method is evaluated with aspects to accuracy and efficiency. The simulation results show that as compared to the baseline approach, the proposed method can not only accurately estimate the energy consumption within 3.9% relative error, but also achieve up to 14.5× speedup in terms of CPU time. The proposed algorithms are compared with benchmarking schemes RMFF, RMBF, and HWGA in terms of energy efficiency and schedule feasibility. The simulation results show that the proposed algorithms consume up to 23.3% less energy and achieve up to 15% higher feasibility as compared to benchmarking schemes.

References

[1] F. Wang, C. Nicopoulos, X. Wu, Y. Xie, and N. Vijaykrishnan, "Variation-aware task allocation and scheduling for MPSoC," in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2007, pp. 598–603.

[2] H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "Energy-efficient adaptive pipelined MPSoCs for multimedia applications," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 5, pp. 663–676, May 2014.

[3] S. Pagani, J.-J. Chen, and J. Henkel, "Energy and peak power efficiency analysis for the single voltage approximation (SVA) scheme," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 9, pp. 1415–1428, Sep. 2015.

[4] A. Ejlali, B. M. Al-Hashimi, and P. Eles, "Low-energy standby-sparing for hard real-time systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 3, pp. 329–342, Mar. 2012.

[5] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 3s, 2014, Art. ID 111.

[6] M. Shafique, L. Bauer, and J. Henkel, "Adaptive energy management for dynamically reconfigurable processors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 1, pp. 50–63, Jan. 2014.

[7] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Exploiting structural duplication for lifetime reliability enhancement," in *Proc. Int. Symp. Comput. Archit.*, Madison, WI, USA, 2005, pp. 520–531.

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

[9] A. Colin, A. Kandhalu, and R. Rajkumari, "Energy-efficient allocation of real-time applications onto heterogeneous processors," in *Proc. Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Chongqing, China, 2014, pp. 1–10.

[10] M. A. Awan and S. M. Petters, "Energy-aware partitioning of tasks onto a heterogeneous multi-core platform," in *Proc. Int. Symp. Real Time Embedded Technol. Appl.*, Philadelphia, PA, USA, 2013, pp. 205–214.

[11] W. Quan and A. Pimentel, "A hybrid task mapping algorithm for heterogeneous MPSoCs," *ACM Trans. Embedded Comput. Syst.*, vol. 14, no. 1, 2015, Art. ID 14.

[12] H. Yu, R. Syed, and Y. Ha, "Thermal-aware frequency scaling for adaptive workloads on heterogeneous MPSoCs," in *Proc. Int. Conf. Design Autom. Test Europe*, Dresden, Germany, 2014, pp. 1–6.

[13] T. Wang, M. Fan, G. Quan, and S. Ren, "Heterogeneity exploration for peak temperature reduction on multi-core platforms," in *Proc. Int. Symp. Qual. Electron. Design*, Santa Clara, CA, USA, 2014, pp. 107–114.

[14] Y. Liu, R. P. Dick, L. Shang, H. Wang, and H. Yang, "Thermal vs energy optimization for DVFS-enabled processors in embedded systems," in *Proc. Int. Symp. Qual. Electron. Design*, San Jose, CA, USA, 2007, pp. 204–209.

[15] D. Li and J. Wu, "Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 810–823, Mar. 2015.

[16] S. Saha, Y. Lu, and J. S. Deogun, "Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems," in *Proc. Int. Conf. Embedded Real Time Comput. Syst. Appl.*, Seoul, Korea, 2012, pp. 41–50.

[17] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2s, 2014, Art. ID 70.

[18] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A System Perspective*. Reading, MA, USA: Addison-Wesley, 1992.

[19] W. Liao, L. He, and K. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 7, pp. 1042–1053, Jul. 2005.

[20] Y. Liu, R. P. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *Proc. Int. Conf. Design Autom. Test Europe*, Nice, France, 2007, pp. 1–6.

[21] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature constraint hard real-time periodic tasks," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 329–339, Aug. 2010.

[22] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time on multicore systems," in *Proc. Int. Symp. Real-Time Embedded Technol. Appl.*, San Francisco, CA, USA, 2009, pp. 131–140.

[23] T. Chantem, X. S. Hu, and R. P. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 10, pp. 1884–1897, Oct. 2011.

[24] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *Proc. Int. Conf. Comput.-Aided Design*, San, Jose, CA, USA, 2007, pp. 281–288.

[25] K. Skadron *et al.*, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, 2004.

[26] HotSpot. (2009). *University of Virginia*. [Online]. Available: http://lava.cs.virginia.edu/HotSpot

[27] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 316–328, Aug. 2010.

[28] D. Zhu, R. Melhem, and B. R. Childers, "Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems," in *Proc. Int. Symp. Real-Time Syst.*, London, U.K., 2001, pp. 84–94.

[29] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo, "Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems," in *Proc. Int. Symp. Real Time Embedded Technol. Appl.*, San Jose, CA, USA, 2006, pp. 408–417.

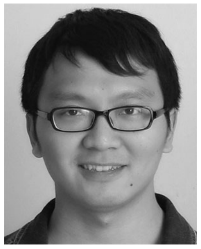
[30] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers," *IEEE Trans. Comput.*, vol. 61, no. 12, pp. 1668–1681, Dec. 2012.

[31] R. Jayaseelan and T. Mitra, "Temperature aware task sequencing and voltage scaling," in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2008, pp. 618–623.

[32] B. Korte, J. Vygen, B. Korye, and J. Vygen, *Combinatorial Optimization*. Berlin, Germany: Springer, 2002.

[33] O. U. Zapata and P. M. Alvarez, "EDF and RM multiprocessor scheduling algorithms: Survey and performance evaluation," Dept. Seccion de Computacion, CINVESTAV-IPN, Mexico City, Mexico, Tech. Rep. CINVESTAV-CS-RTG-02, 2005.

[34] *E3S*. (2013). [Online]. Available: http://ziyang.eecs.umich.edu/~dickrp/e3s/



Junlong Zhou (S'15) is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China.

He was a Research Visitor with the University of Notre Dame, Notre Dame, IN, USA. His current research interests include real-time embedded systems and cloud computing.

Mr. Zhou is an Active Reviewer of several international journals, including IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, *Journal of Systems and Software*, and *Journal of Circuits, Systems, and Computers*.

Tongquan Wei (M'11) received the Ph.D. degree in electrical engineering from Michigan Technological University, Houghton, MI, USA, in 2009.

He is currently an Associate Professor with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interests include real-time embedded systems, green and reliable computing, parallel and distributed systems, and cloud computing.

Dr. Wei has been a Regional Editor for the *Journal of Circuits, Systems, and Computers* since 2012. He served as a Guest Editor for several special sections of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS and *ACM Transactions on Embedded Computing Systems*.

Mingsong Chen (S'08–M'11) received the B.S. and M.E. degrees from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer engineering from the University of Florida, Gainesville, FL, USA, in 2010.

He is currently an Associate Professor with the Department of Embedded Software and Systems, East China Normal University, Shanghai, China. His current research interests include design automation of cyber-physical systems, formal verification techniques, and mobile cloud computing.

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 multiprocessor system-on-chip systems.

Xiaobo Sharon Hu (S'85–M'89–SM'02) received the B.S. degree from Tianjin University, Tianjin, China, in 1982, the M.S. degree from the Polytechnic Institute of New York, Brooklyn, NY, USA, in 1984, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 1989.

She is currently a Professor with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA. She has published over 200 papers in the related areas.

Prof. Hu was a recipient of the National Science Foundation CAREER Award in 1997, and the Best Paper Award from Design Automation Conference in 2001 and the IEEE Symposium on Nanoscale Architectures in 2009. She served as an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, and *ACM Transactions on Embedded Computing*. She has been on the Program Committee of a number of conferences, such as Design Automation Conference, the International Conference on Computer-Aided Design, the Design, Automation and Test in Europe Conference, and the IEEE Real-Time Systems Symposium.

Yue Ma received the B.S. degree from Chengdu University of Technology, Chengdu, China, in 2010, and the M.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013. He is currently pursuing the Ph.D. degree with the University of Notre Dame, Notre Dame, IN, USA.

His current research interests include real-time scheduling in cyber-physical systems and computer architecture.