Thermal-Aware Correlated Two-Level Scheduling of Real-Time Tasks with Reduced Processor Energy on Heterogeneous MPSoCs

Junlong Zhou, Jianming Yan, Kun Cao, Yanchao Tan, Tongquan Wei, Mingsong Chen, Gongxuan Zhang, Xiaodao Chen, and Shiyan Hu

***Abstract*—With the exponential increase in power density and the relentless scaling of transistors in VLSI circuits over the past decades, modern high-performance processors fall into a predicament of high energy consumption and elevated chip temperature. Such increased energy consumption and chip temperature could induce significant economic, ecological, and technical problems. Thus, energy-efficient task scheduling with thermal consideration has become a pressing research issue in sustainable computing systems, especially for battery-powered real-time embedded systems with limited cooling techniques.**

This paper tackles the above challenge through scheduling tasks leveraging correlated optimizations at two different scales. Precisely, a two-level thermal-aware energy-efficient scheduling algorithm for real-time tasks on DVFS-enabled heterogeneous MPSoC systems is developed considering the constraints of task deadlines, task precedences, and chip peak temperature limit. At the processor level, a multi-processor model supporting dynamic voltage/frequency scaling is transformed to a virtual multi-processor model supporting only one fixed frequency level. At the core level, real-time tasks are assigned to individual cores of the virtual processor under the constraints of task precedence and peak temperature limit. Through nicely interleaving optimizations at both levels, high quality task scheduling solutions can be computed efficiently. Extensive simulations of synthetic real-time tasks and real-life benchmarks are performed to validate the proposed algorithm. Experimental results demonstrate the effectiveness of the proposed algorithm as compared to the benchmarking schemes.

***Index Terms*—Energy efficiency, thermal-aware, task precedence, real-time MPSoC heterogeneous systems, co-scheduling.**

I. Introduction

As needs for high performance computing in sustainable systems continue to increase, the energy consumption of VLSI systems explodes, which poses adverse impact on the lifespan of portable devices with limited battery capacity. Meanwhile,

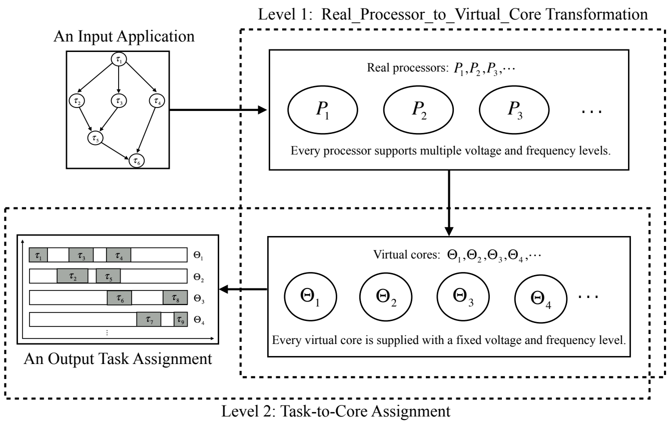

Level 1: Real_Processor_to_Virtual_Core Transformation
An Input Application
Real processors: $P_1, P_2, P_3, \cdots$
P_1
P_2
P_3
Every processor supports multiple voltage and frequency levels.
Virtual cores: $\Theta_1, \Theta_2, \Theta_3, \Theta_4, \cdots$
Θ_1
Θ_2
Θ_3
Θ_4
Every virtual core is supplied with a fixed voltage and frequency level.
An Output Task Assignment
Level 2: Task-to-Core Assignment

Fig. 1: Design flow of the proposed two-level scheduling.

the soaring integration level of transistors in VLSI circuits strikingly increases chip power density and thus elevates chip temperature. Such high temperature degrades system reliability by accelerating the device wearout mechanisms through electro-migration, dielectric breakdown, thermal cycling, or stress migration [1]–[3]. It also results in high leakage power due to strong temperature/leakage dependency, which in turn increases the chip temperature and consequently incurs significant packaging and cooling costs. Hence, temperature-aware energy minimization is a pressing research issue in the design of battery-powered sustainable computing systems.

Heterogeneous multiprocessors have been extensively adopted in various real-time embedded applications due to their relatively better performance and lower energy consumption when compared to homogeneous processors [4]–[6]. A multiprocessor system on chip (MPSoC) is naturally heterogeneous in the sense that its processing units such as customized hardware modules, programmable microprocessors, and embedded FPGAs have distinctive functionalities and demonstrate varying computing capability [7]. In this paper, we focus on temperature-aware energy-efficient task scheduling issues for heterogeneous real-time MPSoC systems.

Specifically, we proposed a task allocation and frequency selection method that minimizes the energy consumed by MPSoC systems supporting discrete voltage/frequency levels. The algorithm takes as input a given set of precedence constrained real-time tasks and peak temperature limit. It generates an energy-efficient schedule that meets the design requirements by wisely determining the tasks assigned to processors and the operating frequency of assigned tasks. Energy savings

This work was partially supported by Natural Science Foundation of Shanghai under the grant 16ZR1409000, and National Natural Science Foundation of China under the grant 61672230. T. Wei is the corresponding author. Email: tqwei@cs.ecnu.edu.cn.

J. Zhou, J. Yan, K. Cao, C. Tan and T. Wei are with the Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China. J. Zhou is also with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.

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

G. Zhang is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.

X. Chen is with the School of Computer Science, China University of Geosciences, Wuhan 430074, China.

S. Hu is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931, USA.

are achieved by utilizing heterogeneities of both MPSoC systems and precedence-constrained real-time tasks. As shown in Fig. 1, the proposed two-level scheme is specifically tailored for heterogeneous multi-scale computing system. The scheme operates as follows. Given an input application that consists of a set of precedence constrained tasks, it first transforms the real multiprocessor system MPSoC to a virtual multicore system at the processor level, then conducts task to core mapping to generate an output task assignment at the core level. The generated task assignment can minimize the system energy consumption under all the constraints. High quality task scheduling solutions can be computed efficiently via optimization at both levels. This paper makes the following major contributions:

- We presented a transformation method for the MPSoC system that converts the processor model with multiple voltage and frequency levels to multiple virtual cores each of which has a fixed supply voltage and frequency level. This method can effectively decrease one dimension for optimization of system energy consumption by reducing task-to-(real) processor assignment and frequency selection problem to task-to-(virtual) core assignment problem.
- We analyzed the energy optimality of assigning tasks to multiple virtual cores of an MPSoC system, and proposed a theorem on optimum task assignment. Based on the theorem, we developed a task-to-(virtual) core assignment heuristic algorithm to reduce the energy consumption.
- We conducted extensive simulation experiments to validate the effectiveness of the proposed algorithm in energy efficiency. Simulation results have demonstrated that the proposed algorithm achieves better performance when compared to the benchmarking schemes.

In this paper, we explore the energy minimization of precedence constrained real-time tasks on DVFS-enabled MPSoCs by utilizing heterogeneities of both MPSoCs and real-time tasks. The remainder of this paper is organized as follows. Section II discusses the related works, Section III presents the system models, Section IV defines and analyzes the concerned energy minimization problem, and Section V describes the proposed energy-efficient task assignment and frequency selection scheme. The effectiveness of the proposed scheme is verified in Section VI and concluding remarks are given in Section VII. For the sake of easy presentation and better comprehension, we summarize the definition of main notations used in the whole paper in TABLE I-III.

II. Related Work

The study on energy efficiency of a heterogeneous MPSoC platform aims to maximize energy savings by allocating limited computational resources of the platform to individual real-time tasks under various design constraints. A. Colin et al. showed in the literature [8] that neither balancing the load nor assigning all load to a particular processor is the best strategy for energy optimization. Hence, researchers often start from an analytically justified target task-to-processor assignment for energy optimization and then derive an energy-efficient task assignment heuristic that approximates it. For example,

TABLE I: Definition of main notations in Section III and IV.

Notation	Definition
P	The MPSoC containing M processors $P_1, P_2, \cdots, P_M$
P_m	The mth processor in system P
$v_{m,k}, f_{m,k}$	The kth voltage and frequency level of processor P_m
x_m	The number of voltage/frequency levels of processor P_m
$G = (\mathcal{V}, \mathcal{E})$	The directed acyclic graph (DAG)
$\mathcal{V}$	The set of $\lvert\mathcal{V}\rvert$ tasks
$\mathcal{E}$	The set of edges representing task precedency
τ_i, τ_j	The ith and jth task in set $\mathcal{V}$
μ_i	The active factor of task τ_i
c_i	The worst case execution cycles of task τ_i
D	The common deadline (frame size) of tasks in set $\mathcal{V}$
$ET_{i,m,k}$	The execution time of τ_i on processor P_m at frequency $f_{m,k}$
$Pow^{leak}_{m,k}$	The leakage power consumption of processor P_m at $f_{m,k}$
$\alpha_m, \gamma_m, \delta_m$	Non-negative architecture-dependent constants of processor P_m
$T(t), T_0, T_e$	The processor operating temperature at time instance t, t_0, t_e
$T_m(t)$	The instantaneous temperature of processor P_m at time instance t
$Pow^{dyn}_{i,m,k}$	The dynamic power consumed by executing τ_i on P_m at $f_{m,k}$
$Pow(t)$	The processor power consumption at time instance t
T^{amb}_m	The ambient temperature of processor P_m
T^{amb}	The processor ambient temperature
R, C	The processor thermal resistance and capacitance
R_m, C_m	The thermal resistance and capacitance of processor P_m
EC^{tot}_D	The total energy consumed by system P during frame D
EC^{leak}_D	The leakage energy consumed by system P during frame D
EC^{dyn}_D	The dynamic energy consumed by system P during frame D
$\mathcal{V}_{m,k}$	The subset of tasks allocated to P_m and executed at $f_{m,k}$
T^{peak}_D	The peak temperature of system P during frame D
$t_s(\tau_i)$	The start execution time of task τ_i
$t_f(\tau_j)$	The finish execution time of task τ_j
$\Lambda_{i,j}$	The binary variable used to represent if task τ_j precedes task τ_i

Li et al. [9] formulated the energy optimization problems as binary integer programming problems, relaxed them as convex optimization problems, and proposed a relaxation-based iterative rounding algorithm to approximate the optimal solution of the relaxed problems. Similarly, Wang et al. [10] first presented an integer linear programming-based method to solve the problem of task and data allocation for energy minimization. Then two heuristic algorithms were designed to generate near-optimal solutions for real-time applications in polynomial time. Zhang et al. [11] proposed a novel genetic algorithm based approach to improve energy savings and system reliability for scheduling workflow with precedence constraints in heterogeneous multicore systems. Zahaf et al. [12] presented a general methodology to model the energy consumption of sporadic tasks on heterogeneous multicore architectures such as ARM big/LITTLE. They also developed a heuristic for parallelizing and allocating threads on the multicore, and setting the frequency and power state of the cores to reduce the total energy consumption without missing deadlines. Although the above work can effectively reduce the energy consumption of MPSoC systems, the temperature design constraint is not taken into account, which may result in degraded system performance.

Energy and temperature are two design constraints that interplay. Energy-aware schemes alone have insufficient impact on chip temperature and may lead to unnecessarily high temperature such that system temperature constraint is violated [13]. As a result, extensive investigation has been conducted on temperature-constrained task scheduling for energy minimization. Liu et al. [14] first introduced dynamic voltage/frequency scaling (DVFS) in design-time thermal optimization for uniprocessor systems. They compared the energy and thermal-optimal solutions, and proposed a thermal-constrained energy optimization procedure to minimize system energy consumption under a constraint on peak temperature. Motivated by this work, Deogun et al. [15], [16] investigated thermal-constrained partitioning of periodic real-time tasks for energy minimization in heterogeneous multiprocessor systems. They designed genetic algorithm and branch-and-bound based methods to assign real-time tasks to individual processors, adopted a power model that captures the impact of temperature and voltage on the leakage current, and utilized heterogeneities of MPSoCs to save energy and improve system performance.

In addition to heterogeneities of MPSoCs, the heterogeneous characteristics of real-time tasks also can be utilized for temperature-constrained energy minimization. Tasks are deemed to be heterogeneous when different tasks exhibit distinctive power consumptions on the same processor with exactly the same configuration [14]. We exploited heterogeneities of both MPSoCs and real-time tasks for the first time in the literature [17] to minimize energy consumption of a system under the temperature constraint. Heterogenous real-time tasks are initially allocated to individual processors to minimize dynamic energy consumption, then slack time is distributed among allocated tasks to minimize the chip temperature, which in turn reduces the leakage power. However, this work assumes a heterogenous MPSoC which supports fixed voltage levels and operating frequencies. Moreover, real-time tasks are assumed to be independent, which limits the application of the presented schemes in scenarios like streams where real-time tasks have precedence or data dependencies.

Unlike the above works that concentrate on independent real-time tasks, He et al. [18] developed a graph-based scheduling algorithm to find the optimal co-scheduling solution for serial and parallel jobs on multicores. However, energy and timing are not considered as a design constraint. Energy and timing are taken into account in the literature [19]–[21] when scheduling precedence-constrained real-time tasks on multiprocessor computers using DVFS. However, heterogeneities of neither multiprocessors nor real-time tasks are employed for energy savings. In addition, temperature is not considered as a design constraint in the above works.

III. System Models

In this section, we briefly introduce our system models, including the processor and application model, power model, and temperature model.

A. Processor and Application Model

The MPSoC system considered in this paper consists of M processors, denoted by $P = \{P_1, P_2, \cdots, P_M\}$. Each processor P_m $(1 \leq m \leq M)$ is a typical DVFS-enabled processor that can operate with a set of discrete supply voltage and frequency pairs $(v_{m,k}, f_{m,k})$ $(1 \leq k \leq x_m)$, where $v_{m,1} < v_{m,2} < \cdots < v_{m,k} < \cdots < v_{m,x_m}$, $f_{m,1} < f_{m,2} < \cdots < f_{m,k} < \cdots < f_{m,x_m}$, and x_m is the voltage/frequency level of processor P_m. Real-time applications are supposed to execute on the MPSoC system. Such an application is a set of precedence constrained tasks and can be represented by a directed acyclic graph (DAG) $G = (\mathcal{V}, \mathcal{E})$. The set of vertices $\mathcal{V}$ represents the set of tasks τ_i $(1 \leq i \leq |\mathcal{V}|)$, and the set of edges represents a partial order corresponding to the precedence constraints among tasks. For example, edge $(\tau_i, \tau_j) \in \mathcal{E}$ $(1 \leq i, j \leq |\mathcal{V}|)$ indicates the task τ_i cannot start to execute until task τ_j has been completed. A task without any parent is called an entry task and a task without any child is called an end task. Same as the literature [22], we assume that a DAG G has only one entry node and one end node. Fig. 2 presents an example application with six tasks and precedence constraints. As described in the figure, τ_1 is the entry task since it has no parent and τ_6 is the end task since it has no child. The execution of tasks needs to meet the precedence constraints that tasks $\tau_2 - \tau_4$ cannot start to execute until task τ_1 is finished, task τ_5 cannot start to execute until tasks $\tau_2 - \tau_3$ are finished, and task τ_6 cannot start to execute until tasks $\tau_4 - \tau_5$ are finished. Since tasks $\tau_2 - \tau_4$ are independent, they could be either executed serially in an arbitrary order or executed in parallel. An execution order of tasks $\tau_1 - \tau_6$ that meets the precedence constraints can be obtained through topological sorting algorithm [23].

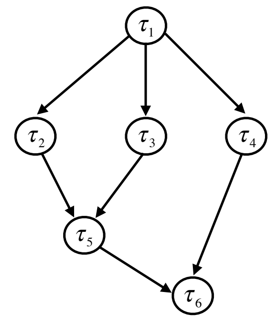


Fig. 2: An example application with precedence constraints.

Each task τ_i in an application is characterized using a triplet $\tau_i : \{\mu_i, c_i, D\}$, where μ_i is the task active factor, c_i is the worst case execution cycles, and D is the common deadline that is also the frame size of tasks in the application. Besides the operating frequency of the MPSoC system, the power consumption of a task highly depends upon the circuit activity and the usage frequency of different functional units when executing the task [24]. Hence, the task activity factor μ (ranging in (0,1]) that defines how intensively functional units are being utilized by the task is adopted to capture the different switching factors of different tasks [25]. Let $ET_{i,m,k}$ denote the execution time of task τ_i on processor P_m at the frequency $f_{m,k}$, then it is calculated by

$$ET_{i,m,k} = \frac{c_i}{f_{m,k}}. \tag{1}$$

B. Power Model

The power consumption of an CMOS device can be modeled as the sum of leakage power dissipation and dynamic power dissipation. The leakage power is temperature dependent and consumed by the leakage current required to maintain basic state of circuits [26]. As leakage current changes super linearly with temperature, the leakage power consumption of

$$\frac{dT_m(t)}{dt} = \frac{T_m^{amb} + \alpha_m \cdot R_m \cdot v_{m,k} + \gamma_m \cdot R_m \cdot v_{m,k} \cdot T_m(t) + \mu_i \cdot \delta_m \cdot R_m \cdot v_{m,k}^2 \cdot f_{m,k}}{R_m \cdot C_m} - \frac{T_m(t)}{R_m \cdot C_m} \tag{5}$$

$$\int_{t_0}^{t_e} dt = \int_{T_0}^{T_e} \frac{dT_m(t)}{\frac{T_m^{amb}+\alpha_m \cdot R_m \cdot v_{m,k}+\mu_i \cdot \delta_m \cdot R_m \cdot v_{m,k}^2 \cdot f_{m,k}}{R_m \cdot C_m} - (\frac{1-\gamma_m \cdot R_m \cdot v_{m,k}}{R_m \cdot C_m}) \cdot T_m(t)} \tag{6}$$

$$\begin{aligned} T_e =& (T_0 - \frac{T_m^{amb} + \alpha_m \cdot R_m \cdot v_{m,k} + \mu_i \cdot \delta_m \cdot R_m \cdot v_{m,k}^2 \cdot f_{m,k}}{1-\gamma_m \cdot R_m \cdot v_{m,k}}) \cdot e^{-(\frac{1-\gamma_m \cdot R_m \cdot v_{m,k}}{R_m \cdot C_m})(t_e-t_0)} \\ &+ \frac{T_m^{amb} + \alpha_m \cdot R_m \cdot v_{m,k} + \mu_i \cdot \delta_m \cdot R_m \cdot v_{m,k}^2 \cdot f_{m,k}}{1-\gamma_m \cdot R_m \cdot v_{m,k}} \end{aligned} \tag{7}$$

processor P_m at the kth supply voltage and frequency can be effectively estimated as [27]

$$Pow_{m,k}^{leak} = \alpha_m \cdot v_{m,k} + \gamma_m \cdot v_{m,k} \cdot T_m(t), \tag{2}$$

where α_m and γ_m are both non-negative architecture-dependent constants of processor P_m, and $T_m(t)$ is the operating temperature of processor P_m at time instance t. Clearly, the leakage power varies with the instantaneous temperature.

The dynamic power consumption of a processor is independent of the temperature and can be estimated by a strictly increasing and convex function of the supply voltage and operating frequency, that is, $Pow^{dyn} \propto v^2 f$ [28], where v is the supply voltage and f is the operating frequency. The dynamic power is only consumed when executing tasks. Thus the dynamic power consumption of executing task τ_i on processor P_m at the kth supply voltage and frequency is

$$Pow_{i,m,k}^{dyn} = \mu_i \cdot \delta_m \cdot v_{m,k}^2 \cdot f_{m,k}, \tag{3}$$

where μ_i is the active factor of task τ_i and δ_m is a non-negative constant depending on the architecture of processor P_m.

C. Temperature Model

An accurate and practical dynamic model of temperature is needed to accurately characterize the thermal behavior of an application. In this paper, it is assumed that there is negligible or no heat transfer among processor units and among other different units [27], [29], [30]. Hence, a heat-independent thermal model proposed by Skadron et al. [31] that is widely used in the literature is adopted to predict the temperature of the core. Let $T(t)$ be the temperature at time instance t, then the temperature model is given by

$$R \cdot C \cdot \frac{dT(t)}{dt} + T(t) - R \cdot Pow(t) = T^{amb}, \tag{4}$$

where $Pow(t)$ is the power consumption at time instance t, and it can be obtained using Eqs. (2)-(3). R and C are thermal resistance and capacitance, and are hardware dependent constants. T^{amb} is the ambient temperature.

Let T_m^{amb}, $T_m(t)$, R_m, and C_m be the ambient temperature, instantaneous temperature, thermal resistance, and capacitance of processor P_m, respectively. Then, for a given time interval $[t_0, t_e]$, if the initial temperature is T_0, the ending temperature of executing task τ_i on processor P_m at the supply voltage and frequency $(v_{m,k}, f_{m,k})$, denoted as T_e, can be derived by solving Eq. (4). The derivation of ending temperature T_e is described in Eqs. (5)-(7). (See the top of this page).

IV. Energy Minimization Problem Definition and Analysis

The focus of this work is to minimize the energy consumption of precedence-dependent real-time tasks on the target MPSoC system under the constraints of task deadline and maximum temperature limit. We solve the energy minimization problem by designing thermal-aware energy-efficient task assignment and frequency selection algorithm that determines the tasks assigned to every processor and the operating frequency of every assigned task. Before presenting our algorithm, we first give the formulation and the analysis of our energy minimization problem below.

The energy consumption is calculated as the product of power consumption and execution time. Unlike the traditional works [19], [23], [32] that ignore the temperature dependency in leakage power when calculating energy consumption, this paper adopts a more precise power model that takes into account the dependency. Specifically, based on the power model given in Section III-B and the temperature model given in Section III-C, the total energy consumption of executing the tasks in application $\mathcal{V}$ on the MPSoC system P during the frame D is formulated as

$$EC_D^{tot} = EC_D^{leak} + EC_D^{dyn}. \tag{8}$$

EC_D^{leak} is the energy consumption due to leakage power, which is always consumed unless the processor is turned off. Thus the leakage energy consumption is expressed as

$$\begin{aligned} EC_D^{leak} &= \sum_{m=1}^{M} Pow_m^{leak} \cdot D = \sum_{m=1}^{M} \alpha_m \cdot v_{m,1} \cdot D \\ &+ \sum_{m=1}^{M} \int_0^D \gamma_m \cdot v_{m,1} \cdot T_m(t) dt. \end{aligned} \tag{9}$$

EC_D^{dyn} is the energy consumption due to dynamic power, which is only consumed when executing tasks. Thus the dynamic energy consumption is calculated as

$$\begin{aligned} EC_D^{dyn} &= \sum_{m=1}^{M} \sum_{k=1}^{x_m} \sum_{\tau_i \in \mathcal{V}_{m,k}} Pow_{i,m,k}^{dyn} \cdot ET_{i,m,k} \\ &= \sum_{m=1}^{M} \sum_{k=1}^{x_m} \sum_{\tau_i \in \mathcal{V}_{m,k}} \mu_i \cdot \delta_m \cdot v_{m,k}^2 \cdot f_{m,k} \cdot \frac{c_i}{f_{m,k}} \\ &= \sum_{m=1}^{M} \sum_{k=1}^{x_m} (\delta_m \cdot v_{m,k}^2 \cdot \sum_{\tau_i \in \mathcal{V}_{m,k}} \mu_i \cdot c_i), \end{aligned} \tag{10}$$

where $\mathcal{V}_{m,k}$ denotes the subset of tasks allocated to processor P_m and executed at frequency $f_{m,k}$.

To avoid temperature-induced failures and hence enable the system in a safe and reliable mode, the processor temperature should be below a temperature limit (threshold) T^{max}. The threshold T^{max} is typically specified based on the system design requirements. The peak temperature of the MPSoC system P when executing the application $\mathcal{V}$ during the frame D is given by

$$T_D^{peak} = \max\{T(t) | \forall t \in [0, D]\}, \tag{11}$$

where $T(t)$ is the instantaneous temperature of processors in P at time instance t and can be calculated using Eq. (7).

Considering the above design constraints, the task assignment and frequency selection problem for energy minimization can be formulated as follows. Given an application $\mathcal{V}$ of precedence constrained real-time tasks, and an MPSoC system P of heterogeneous processors, it is expected to derive a task assignment and frequency selection strategy to minimize the system energy consumption while satisfying the constraint of peak temperature, task deadlines, and task precedence. In other words, the problem can be formulated into the below form.

$$\text{Minimize: } EC_D^{tot}$$

$$\text{Subject to: } T_D^{peak} \leq T^{max} \tag{12}$$

$$\forall m = 1, 2, \cdots, M, \ \sum_{k=1}^{x_m} \sum_{\tau_i \in \mathcal{V}_{m,k}} \frac{c_i}{f_{m,k}} \leq D \tag{13}$$

$$\forall \tau_i, \tau_j \in \mathcal{V}, \ t_s(\tau_i) \geq t_f(\tau_j) \cdot \Lambda_{i,j} \tag{14}$$

where EC_D^{tot} and T_D^{peak} are given in Eqs. (8) and (11), respectively. Eq. (12) indicates that the processor peak temperature cannot exceed the threshold, and Eq. (13) indicates that the execution of tasks on processors should be finished before the common deadline. $t_s(\tau_i)$ is the start time of execution of task τ_i, and $t_f(\tau_j)$ is the finish time of execution of task τ_j. $\Lambda_{i,j}$ is a binary variable that takes the value of 1 if task τ_j precedes task τ_i, otherwise takes the value of 0. Eq. (14) indicates the precedence constraint among tasks.

It has been shown in Eqs. (8)-(10) that the system energy consumption is the sum of leakage and dynamic part, and the leakage energy is dependent on temperature while the dynamic energy is not. Since the leakage power varies with temperature and temperature is changing with time, it is challenging to rapidly and accurately estimate the leakage energy consumption. To be specific, either using Eq. (7) to compute the temperature at every time instance is computationally expensive or using thermal modeling tool (e.g., HotSpot [33]) to obtain the temperature profiles is time consuming. Some early literatures such as [19], [23], [32], [34], [35] either simply assume a constant leakage power or totally ignore it since leakage energy consumption used to be a small part of overall energy consumption. However, the portion of leakage part in overall power dissipation is ever-increasing with the continuous scaling of integrated circuits, thus, these energy models are not practical any more.

Temperature-aware leakage energy consumption can be reduced by using a task splitting method proposed in the literature [17]. In this method, task scheduling horizon is divided into sufficiently short intervals of equal length such that the peak temperature in the interval is constant. The leakage dominated static energy consumption during the interval is calculated under the assumption of a fixed peak temperature [14], and the static energy consumed in the whole scheduling horizon is derived by summing up the static energy consumed in all intervals [17]. Since the leakage power positively depends upon chip peak temperature [14], the leakage energy consumption is then minimized by deriving the lowest peak temperature for all intervals of the scheduling horizon.

It can be deduced from the above description that the total energy consumption of the system is mainly determined by the dynamic energy for the scenario where peak temperature reaches temperature limit of the system. This is because the peak temperature dependent leakage energy consumption can not be further reduced in this case. In this work, we concentrate on energy minimization for this case.

TABLE II: Definition of main notations used in Section V.

Notation	Definition
$\mathbf{a}$	The vector that captures (real) processor dependent parameters
$\mathbf{b}$	The vector that captures task related parameters
$a_{m,k} \in \mathbf{a}$	The power dissipation factor of processor P_m at frequency $f_{m,k}$
$b_{m,k} \in \mathbf{b}$	The power dissipation factor of subset $\mathcal{V}_{m,k}$
$\mu_i c_i$	The power dissipation factor of task τ_i
$\mathcal{Y}$	The sum of power dissipation factor of tasks in the application $\mathcal{V}$
$\mathcal{X}$	The total number of virtual cores
Θ	The set of virtual cores $\Theta_1, \Theta_2, \cdots, \Theta_{\mathcal{X}}$
Θ_ℓ	The ℓth virtual core in set Θ
$\mathcal{V}_\ell$	The subset of tasks assigned to virtual core Θ_ℓ
η	The vector that captures virtual core dependent parameters
ζ	The vector that captures task related parameters
$\eta_\ell, \eta_\kappa \in \eta$	The power dissipation factor of virtual core Θ_ℓ, Θ_κ
$\zeta_\ell, \zeta_\kappa \in \zeta$	The power dissipation factor of subset assigned to core Θ_ℓ, Θ_κ
ζ^*	The vector that characterizes the optimum task assignment solution
$\zeta_\ell^*, \zeta_\kappa^* \in \zeta^*$	The ℓth and κth elements in vector ζ^*

V. The Proposed Processor Model Transformation and Task Scheduling Heuristics

As introduced above, the target of this work is to design a two-level correlated optimization process for energy minimization. At the first level of the optimization, a real processor model supporting multiple voltage and frequency levels is transformed to a virtual processor consisting of multiple cores of fixed voltage and frequency levels. At the second level of the optimization, real-time tasks are assigned to individual virtual cores in such a way that the system energy consumption is minimized.

We first design a **R**eal_**P**rocessor_to_**V**irtual_**C**ore (**RPVC**) transformation for the MPSoC system. This transformation maps a real processor with multiple voltage and frequency levels to a virtual processor with multiple virtual cores. Each virtual core has a fixed voltage and frequency level corresponding to one level of the real processor model. Through the transformation, task-to-(real) processor assignment and frequency selection can be translated into task-to-(virtual) core assignment, which reduces one dimension for optimization and hence simplifies our problem. We then analyze the energy optimality of assigning tasks to virtual cores and present a theorem on optimum task assignment. We finally develop a task-to-(virtual) core assignment heuristic algorithm based on the theorem.

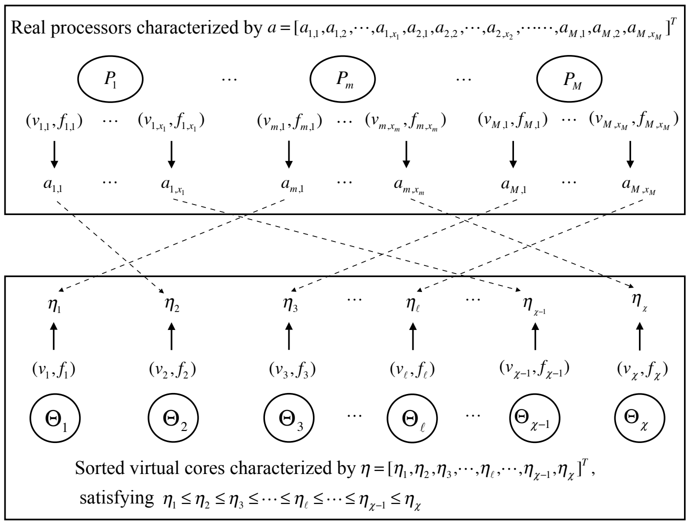
A. ***R**eal_**P**rocessor_to_**V**irtual_**C**ore Transformation*

The dynamic energy consumption given in Eq. (10) can be written as a product of two vectors, that is,

$$\begin{aligned} EC_D^{dyn} &= \sum_{m=1}^{M}\sum_{k=1}^{x_m}(\delta_m \cdot v_{m,k}^2 \cdot \sum_{\tau_i \in \mathcal{V}_{m,k}} \mu_i \cdot c_i) \\ &= \sum_{m=1}^{M}\sum_{k=1}^{x_m} a_{m,k} \cdot b_{m,k} = \mathbf{a} \cdot \mathbf{b}, \end{aligned} \tag{15}$$

where $a_{m,k} = \delta_m \cdot v_{m,k}^2$ and $b_{m,k} = \sum_{\tau_i \in \mathcal{V}_{m,k}} \mu_i \cdot c_i$. Vector $\mathbf{a} = [a_{1,1}, a_{1,2}, \cdots, a_{1,x_1}, \cdots\cdots, a_{M,1}, a_{M,2}, \cdots, a_{M,x_M}]^T$ captures processor dependent parameters and vector $\mathbf{b} = [b_{1,1}, b_{1,2}, \cdots, b_{1,x_1}, \cdots\cdots, b_{M,1}, b_{M,2}, \cdots, b_{M,x_M}]$ captures task related parameters. Here, $a_{m,k} \in \mathbf{a}$ is referred to as the power dissipation factor of processor P_m, $b_{m,k} \in \mathbf{b}$ is referred to as the power dissipation factor of subset $\mathcal{V}_{m,k}$, and $\mu_i c_i$ is referred to as the power dissipation factor of task τ_i. It is clear that **a** is constant since δ_m and $v_{m,k}$ are known for the given MPSoC system P while **b** is not since it depends on the task assignment and frequency selection (e.g., $\mathcal{V}_{m,k}$). In addition, for a given application of $|\mathcal{V}|$ tasks, the sum of power dissipation factor of tasks in the application is a constant (e.g., denoted by $\mathcal{Y}$) and can be calculated as $\sum_{m=1}^{M}\sum_{k=1}^{x_m} b_{m,k} = \sum_{m=1}^{M}\sum_{k=1}^{x_m}\sum_{\tau_i \in \mathcal{V}_{m,k}} \mu_i \cdot c_i = \sum_{i=1}^{|\mathcal{V}|} \mu_i \cdot c_i = \mathcal{Y}$.

The power dissipation of the concerned MPSoC system can be characterized by using the vector $\mathbf{a} = [a_{1,1}, a_{1,2}, \cdots, a_{1,x_1}, \cdots\cdots, a_{M,1}, a_{M,2}, \cdots, a_{M,x_M}]^T$ defined in Eq. (15). However, the current form of **a** doesnot support a one-to-one correspondence between the number of processors (i.e. M) and the number of power dissipation factors (i.e. $\sum_{m=1}^{M} x_m$) sine every processor could have multiple supply voltage levels. Thereby, for the sake of easy presentation and to simplify the optimization, we perform an **RPVC** transformation that takes as input the MPSoC system and outputs a sorted virtual core system. In the transformation, a one-to-one correlation between cores and core power dissipation factors is created and the virtual cores are arranged in the increasing order of their power dissipation factors. This can be realized by transforming a real processor model supporting multiple supply voltage and frequency levels into a virtual processor of multiple virtual cores each of which supports a fixed voltage and frequency pair, and sorting the virtual cores in the increasing order of their power dissipation factors. The obtained virtual cores are assumed to share the same characteristics as the real processor except for the supply voltage and frequency.



Real processors characterized by $a = [a_{1,1}, a_{1,2}, \cdots, a_{1,x_1}, a_{2,1}, a_{2,2}, \cdots, a_{2,x_2}, \cdots\cdots, a_{M,1}, a_{M,2}, a_{M,x_M}]^T$
P_1
P_m
P_M
$(v_{1,1}, f_{1,1})$
(v_{1,x_1}, f_{1,x_1})
$(v_{m,1}, f_{m,1})$
(v_{m,x_m}, f_{m,x_m})
$(v_{M,1}, f_{M,1})$
(v_{M,x_M}, f_{M,x_M})
$a_{1,1}$
a_{1,x_1}
$a_{m,1}$
a_{m,x_m}
$a_{M,1}$
a_{M,x_M}
η_1 η_2 η_3 η_ℓ $\eta_{\chi-1}$ η_χ
(v_1, f_1) (v_2, f_2) (v_3, f_3) (v_ℓ, f_ℓ) $(v_{\chi-1}, f_{\chi-1})$ (v_χ, f_χ)
Θ_1 Θ_2 Θ_3 Θ_ℓ $\Theta_{\chi-1}$ Θ_χ
Sorted virtual cores characterized by $\eta = [\eta_1, \eta_2, \eta_3, \cdots, \eta_\ell, \cdots, \eta_{\chi-1}, \eta_\chi]^T$,
satisfying $\eta_1 \le \eta_2 \le \eta_3 \le \cdots \le \eta_\ell \le \cdots \le \eta_{\chi-1} \le \eta_\chi$

Fig. 3: An illustration of the **RPVC** transformation.

An illustration of the **RPVC** transformation is shown in Fig. 3. As demonstrated in the figure, the real MPSoC system P is transformed into a virtual processor $\Theta = \{\Theta_1, \Theta_2, \cdots, \Theta_\ell, \cdots, \Theta_{\mathcal{X}-1}, \Theta_{\mathcal{X}}\}$, where $\mathcal{X} = \sum_{m=1}^{M} x_m$ and every core $\Theta_\ell \in \Theta$ ($1 \le \ell \le \mathcal{X}$) is supplied with a fixed voltage and frequency pair (v_ℓ, f_ℓ). The virtual core system Θ is then characterized by vector $\eta = [\eta_1, \eta_2, \cdots, \eta_\ell, \cdots, \eta_{\mathcal{X}-1}, \eta_{\mathcal{X}}]^T$ and $\eta_1 \le \eta_2 \le \cdots \le \eta_\ell \le \cdots \le \eta_{\mathcal{X}-1} \le \eta_{\mathcal{X}}$ holds, where $\eta_\ell = \delta_\ell \cdot v_\ell^2$ is referred to as the power dissipation factor of virtual core Θ_ℓ. Accordingly, the power dissipation of subsets assigned to virtual cores can be represented by vector $\zeta = [\zeta_1, \zeta_2, \cdots, \zeta_\ell, \cdots, \zeta_{\mathcal{X}-1}, \zeta_{\mathcal{X}}]$, where $\zeta_\ell = \sum_{\tau_i \in \mathcal{V}_\ell} \mu_i \cdot c_i$ is referred to as the power dissipation factor of subset assigned to core Θ_ℓ and $\zeta_1 + \zeta_2 + \cdots + \zeta_\ell + \cdots + \zeta_{\mathcal{X}} = \mathcal{Y}$ holds. Thus the dynamic energy consumption given in Eq. (15) can be expressed as the product of η and ζ, that is,

$$EC_D^{dyn} = \eta \cdot \zeta. \tag{16}$$

Through the **RPVC** transformation, our problem of assigning tasks to real processors and determining operating frequency of assigned tasks for energy minimization can be converted into the problem of designing an energy-optimum task-to-(virtual) core assignment.

B. *Optimality Analysis of Task-to-(Virtual) Core Assignment*

As we have pointed out earlier, the dynamic energy consumption can be minimized by optimally assigning $|\mathcal{V}|$ tasks to $\mathcal{X}$ virtual cores. Let $\zeta^* = [\zeta_1^*, \zeta_2^*, \cdots, \zeta_\ell^*, \cdots, \zeta_{\mathcal{X}-1}^*, \zeta_{\mathcal{X}}^*]$ denote the power dissipation factor of the optimum task assignment solution, where ζ_ℓ^* is the power dissipation factor of task subset assigned to processor ℓ. The power dissipation factor vector $\eta = [\eta_1, \eta_2, \cdots, \eta_\ell, \cdots, \eta_{\mathcal{X}-1}, \eta_{\mathcal{X}}]^T$ of the virtual core system Θ is also given. We then present a theorem below which shows that the dynamic energy consumption is minimized when the virtual core with smaller power dissipation factor ends up with the subset of its allocated tasks having a larger power dissipation factor.

Theorem 1: *If the virtual core power dissipation factor $\eta_1 \leq \eta_2 \leq \cdots \leq \eta_\ell \leq \cdots \leq \eta_{\mathcal{X}}$ holds for $\eta = [\eta_1, \eta_2, \cdots, \eta_\ell, \cdots, \eta_{\mathcal{X}}]^T$, and the sum of the corresponding task subset power dissipation factor $\zeta_1^* + \zeta_2^* + \cdots + \zeta_\ell^* + \cdots + \zeta_{\mathcal{X}}^*$ is fixed for $\zeta^* = [\zeta_1^*, \zeta_2^*, \cdots, \zeta_\ell^*, \cdots, \zeta_{\mathcal{X}}^*]$, then the dynamic energy consumption EC_D^{dyn} is minimized to $EC_{D,*}^{dyn}$ if $\zeta_1^* \geq \zeta_2^* \geq \cdots \geq \zeta_\ell^* \geq \cdots \geq \zeta_{\mathcal{X}}^*$ holds.*

Proof: Let $EC_D'^{dyn}$ denote the dynamic energy consumption where positions of exactly two elements in the energy optimal assignment ζ^* are exchanged. Assume that the position of ζ_κ^* and ζ_ℓ^* $(1 \leq \ell < \kappa \leq \mathcal{X})$ in ζ^* is exchanged for $EC_D'^{dyn}$, then ζ^* becomes $[\zeta_1^*, \zeta_2^*, \cdots, \zeta_{\ell-1}^*, \zeta_\kappa^*, \zeta_{\ell+1}^*, \cdots, \zeta_{\kappa-1}^*, \zeta_\ell^*, \zeta_{\kappa+1}^*, \cdots, \zeta_{\mathcal{X}}^*]$ in this case. According to the definition of dynamic energy consumption given in Eq. (16), we have $EC_{D,*}^{dyn} = \eta_1\zeta_1^* + \eta_2\zeta_2^* + \cdots + \eta_\ell\zeta_\ell^* + \cdots + \eta_\kappa\zeta_\kappa^* + \cdots + \eta_{\mathcal{X}}\zeta_{\mathcal{X}}^*$ and $EC_D'^{dyn} = \eta_1\zeta_1^* + \eta_2\zeta_2^* + \cdots + \eta_\ell\zeta_\kappa^* + \cdots + \eta_\kappa\zeta_\ell^* + \cdots + \eta_{\mathcal{X}}\zeta_{\mathcal{X}}^*$. Since $EC_{D,*}^{dyn}$ is optimum, we have $EC_D'^{dyn} - EC_{D,*}^{dyn} = (\eta_\ell - \eta_\kappa)(\zeta_\kappa^* - \zeta_\ell^*) \geq 0$. It is known that $\eta_\ell \leq \eta_\kappa$ holds for $\ell < \kappa$, then $\zeta_\ell^* \geq \zeta_\kappa^*$ is derived. It can be observed from the above proof that for any two virtual cores of an optimum task assignment solution, the virtual core with smaller core power dissipation factor has a subset of tasks with larger task power dissipation.

Given the optimum task assignment solution $\zeta^* = [\zeta_1^*, \zeta_2^*, \cdots, \zeta_{\ell-1}^*, \zeta_\ell^*, \zeta_{\ell+1}^*, \cdots, \zeta_{\kappa-1}^*, \zeta_\kappa^*, \zeta_{\kappa+1}^*, \cdots, \zeta_{\mathcal{X}}^*]$ that minimizes the dynamic energy consumption, any feasible solution in the solution space[1] can be obtained by exchanging elements in ζ^* multiple times. In each iteration of the exchange, it can be deduced that $\zeta_\ell^* \geq \zeta_\kappa^*$ holds for $\ell < \kappa$. In other words, the dynamic energy consumption is minimized when the virtual core with smaller power dissipation factor ends up with the subset of its assigned tasks having a larger power dissipation factor. The theorem is proved. ■

C. Task-to-(Virtual) Core Assignment Heuristic

Assigning multiple tasks to individual cores is well known as an NP-hard problem, which necessitates the designing of a sub-optimal task assignment heuristics. In this work, we propose a sub-optimal task assignment heuristics which is motivated by the theorem presented in Section V-B, that is, assigning the subset having a larger power dissipation factor to the virtual core having a smaller power dissipation factor can minimize the dynamic energy consumption. The heuristics operates as follows. Tasks in the subset with the maximum power dissipation factor is assigned to the virtual core with the minimum power dissipation factor, and tasks in the subset with the next maximum power dissipation factor is assigned to the virtual core with the next minimum power dissipation factor. This process repeats until all subsets of tasks are assigned to individual cores. In addition, the constraints of task deadline and precedence, and system peak temperature limit are examined during the task assignment.

The details of the task assignment heuristics are described in Algorithm 1. It partitions the tasks in task set (i.e., application $\mathcal{V}$) into subsets, then assigns subsets of selected tasks to individual virtual cores. Since the $\mathcal{X}$ virtual cores in set Θ are sorted in the non-decreasing order of core power dissipation factors, the focus of the algorithm becomes to derive a task-to-(virtual) core assignment that partitions tasks into $\mathcal{X}$ subsets, arranged in the non-increasing order of subset power dissipation factors, then assigns them to corresponding virtual cores. This can be achieved by assigning tasks with larger task power dissipation factors to virtual cores with smaller core power dissipation factors.

TABLE III: Definition of some notations used in Algorithm 1.

Notation	Definition
$U(\tau_i, \Theta_\ell)$	The utilization of task τ_i when assigned to virtual core Θ_ℓ
$U(\Theta_\ell)$	The utilization of virtual core Θ_ℓ
$\overline{\Theta_\ell}$	Virtual cores transformed from the same real processor as Θ_ℓ is
$U(\overline{\Theta_\ell})$	The utilization of virtual cores denoted by $\overline{\Theta_\ell}$
$\mathcal{V}_{temp}$	A temporary task subset $\mathcal{V}_{temp}$ prepared for $\mathcal{V}_\ell$
$ET_{i,\ell}$	The execution time of task τ_i on virtual core Θ_ℓ
$ET(\mathcal{V}_\ell)$	The execution time of tasks assigned to virtual core Θ_ℓ
$ET(\mathcal{V}_{temp})$	The execution time of tasks in temporary subset $\mathcal{V}_{temp}$

For the sake of better understanding of Algorithm 1, the definition of some notations used in the algorithm are listed in TABLE III. As shown in the table, $U(\tau_i, \Theta_\ell)$ denotes the utilization of task τ_i if it is assigned to virtual core Θ_ℓ, which is calculated as the quotient of task execution time and frame size. $U(\Theta_\ell)$ denotes the utilization of virtual core Θ_ℓ, which is calculated as the sum of utilization of tasks assigned to the core. Let $\overline{\Theta_\ell}$ represent the virtual cores that are transformed from the same real processor as Θ_ℓ is, then the utilization of such cores is represented by $U(\overline{\Theta_\ell})$. Due to the processor capacity constraint, the sum of utilization of all the virtual cores transformed from the same real processor cannot be greater than 1, which is expressed as $U(\Theta_\ell) + U(\tau_i, \Theta_\ell) + U(\overline{\Theta_\ell}) \leq 1$. The algorithm needs to take into account this constraint when assigning tasks to virtual cores. Let $ET_{i,\ell}$, $ET(\mathcal{V}_\ell)$, and $ET(\mathcal{V}_{temp})$ denote the execution time of task τ_i on virtual core Θ_ℓ, tasks assigned to virtual core Θ_ℓ, and temporary task subset $\mathcal{V}_{temp}$, respectively. All of $ET_{i,\ell}$, $ET(\mathcal{V}_\ell)$, and $ET(\mathcal{V}_{temp})$ can be calculated based on Eq. (1).

Inputs to Algorithm 1 are the application $G = (\mathcal{V}, \mathcal{E})$, MPSoC system P, and peak temperature limit T^{max}. In addition to these inputs, the algorithm maintains a task queue Q_{task}, in which tasks are classified into two categories. That is, tasks without precedence constraints are regarded as independent tasks while tasks with precedence constraints are regarded as dependent tasks. The independent tasks are sorted in the non-increasing order of power dissipation factors for following the idea of our heuristics, in order to construct the energy-optimum task assignment. The dependent tasks are sorted to satisfy task precedence constraints in $\mathcal{E}$, which can be achieved

[1]The feasible solution space includes all task assignments that meet system timing and temperature constraints. The optimum task assignment consumes minimum energy in the solution space.

Algorithm 1 Energy-efficient task-to-(virtual) core assignment under system constraints

```
Input: Application represented by $G = (\mathcal{V}, \mathcal{E})$, MPSoC system represented by $P$, and peak temperature limit $T^{max}$
Require: Maintain a task queue $Q_{task}$ where independent tasks are in the non-increasing order of their power dissipation factors and dependent tasks are in sequence to satisfy their precedence constraints in $\mathcal{E}$
Output: Task-to-(virtual) core assignment represented by subsets $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_\ell, \cdots, \mathcal{V}_\mathcal{X}$
 1: transform the real processor system $P$ to the virtual core system $\Theta$ by $\Theta = \mathbf{RPVC}(P)$, where the $\mathcal{X}$ cores of $\Theta$ are in the non-decreasing order of power dissipation factors;
 2: for $\ell = 1$ to $\mathcal{X}$ do
 3:   initialize the utilization and subset of virtual core $\Theta_\ell$ by $U(\Theta_\ell) = 0$ and $\mathcal{V}_\ell = \emptyset$, respectively;
 4: end for
 5: move all tasks in application $\mathcal{V}$ to the queue $Q_{task}$;
 6: $\ell = 1$;
 7: for $i = 1$ to $len(Q_{task})$ do
 8:   $flag[i] = $ true;
 9: end for
10: create the queue $Q_{assign}$ to copy the assigned tasks and initialize the queue to $\emptyset$;
11: while $Q_{task} \neq$ NULL && $\ell \leq \mathcal{X}$ do
12:   create a temporary subset $\mathcal{V}_{temp}$;
13:   $ET(\mathcal{V}_\ell) = 0$;
14:   for $i = 1$ to $len(Q_{task})$ do
15:     $\mathcal{V}_{temp} = \mathcal{V}_\ell + \tau_i$;
16:     $ET(\mathcal{V}_{temp}) = ET(\mathcal{V}_\ell) + ET_{i,\ell}$;
17:     if $ET(\mathcal{V}_{temp}) \leq D$ && $U(\Theta_\ell) + U(\tau_i, \Theta_\ell) + U(\overline{\Theta_\ell}) \leq 1$ && $T_D^{peak}(\mathcal{V}_{temp}) \leq T^{max}$ then
18:       for $j = 1$ to $len(Q_{assign})$ do
19:         if $t_s(\tau_i) < t_f(\tau_j) \cdot \Lambda_{i,j}$ then
20:           $flag[i] = $ false; // Precedence is violated
21:           break;
22:         end if
23:       end for
24:       if $flag[i] ==$ true then
25:         $\mathcal{V}_\ell = \mathcal{V}_\ell + \tau_i$;
26:         $U(\Theta_\ell) = U(\Theta_\ell) + U(\tau_i, \Theta_\ell)$;
27:         $ET(\mathcal{V}_\ell) = ET(\mathcal{V}_\ell) + ET_{i,\ell}$;
28:         $Q_{task} = Q_{task} - \tau_i$;
29:         $Q_{assign} = Q_{assign} + \tau_i$;
30:       end if
31:     end if
32:   end for  // Use first-fit to group tasks into subsets
33:   $\ell = \ell + 1$;
34: end while
35: if $Q_{task} \neq$ NULL && $\ell > \mathcal{X}$ then
36:   exit(1);  // Exit when infeasible
37: end if
38: return the subsets $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_\ell, \cdots, \mathcal{V}_\mathcal{X}$;
```

by topological sorting algorithm [23]. The head and tail of task queue Q_{task} are the entry task and end task, respectively.

The algorithm first transforms the MPSoC system to a virtual core system using $\Theta = \mathbf{RPVC}(P)$ where the cores are arranged in the non-decreasing order of power dissipation factors, and initializes the utilization and subset of virtual cores using $U(\Theta_\ell) = 0$ and $\mathcal{V}_\ell = \varnothing$ (lines 1-4). All the tasks in application $\mathcal{V}$ are moved to queue Q_{task} and the core index ℓ is set to 1 (lines 5-6). Lines 7-9 initialize the value of $flag[i]$ for every task in the queue Q_{task}, which is utilized to judge if the precedence constraint of task τ_i is satisfied. The queue Q_{assign} is created to copy the assigned tasks and is initialized to empty (line 10). The algorithm then iteratively implements the process of task-to-(virtual) core assignment if the queue Q_{task} is not empty and not all the cores in Θ have been considered (lines 11-34).

In each round of iteration, lines 12-13 create a temporary subset $\mathcal{V}_{temp}$ for feasibility test of assigning tasks to virtual core Θ_ℓ and initialize the execution time $ET(\mathcal{V}_\ell)$ of tasks in subset $\mathcal{V}_\ell$, and lines 14-32 iteratively assign tasks in queue Q_{task} to core Θ_ℓ and construct subset $\mathcal{V}_\ell$ of tasks in a first-fit manner according to the schedulability requirement. During the assignment of task in queue Q_{task} to core Θ_ℓ, a temporary subset $\mathcal{V}_{temp}$ is constructed by copying the subset $\mathcal{V}_\ell$ and task τ_i (line 15) and used to facilitate the feasibility analysis of assigning task τ_i to core Θ_ℓ under the constraints of task deadline (i.e., $ET(\mathcal{V}_{temp}) \leq D$), processor capacity (i.e., $U(\Theta_\ell) + U(\tau_i, \Theta_\ell) + U(\overline{\Theta_\ell}) \leq 1$), and peak temperature (i.e., $T_D^{peak}(\mathcal{V}_{temp}) \leq T^{max}$) (lines 16-17). Lines 18-23 check whether the task precedence constraint (i.e., $t_s(\tau_i) < t_f(\tau_j) \cdot \Lambda_{i,j}$) is satisfied or not. If the assignment can satisfy these constraints, the task is assigned to the core, then task τ_i is added to subset $\mathcal{V}_\ell$ (line 25), the utilization $U(\Theta_\ell)$ of core Θ_ℓ is increased by $U(\Theta_\ell) = U(\Theta_\ell) + U(\tau_i, \Theta_\ell)$ (line 26), the execution time $ET(\mathcal{V}_\ell)$ of subset $\mathcal{V}_\ell$ is updated to $ET(\mathcal{V}_\ell) + ET_{i,\ell}$ (line 27). Since task τ_i is assigned to core Θ_ℓ, the queue Q_{task} of unassigned tasks and the queue Q_{assign} of assigned tasks both need to be updated by $Q_{task} = Q_{task} - \tau_i$ (line 28) and $Q_{assign} = Q_{assign} + \tau_i$ (line 29), respectively. The procedure then moves to the next iteration and considers the allocation of the next task in queue Q_{task}. Otherwise, the task is not assigned and the procedure directly moves to the next iteration. If there is no feasible schedule for the system under the constraints, the algorithm exits (lines 35-37). The target task-to-(virtual) core assignment, represented by subsets $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_\ell, \cdots, \mathcal{V}_\mathcal{X}$, are returned in line 38.

Algorithm 1 is developed to generate an energy-efficient task-to-(virtual) core assignment under system requirements. Once the task-to-(virtual) core assignment is generated, the task-to-(real) processor assignment can be obtained accordingly. The execution of tasks on each real processor follows the order that they are assigned to the virtual cores of the processor. Since tasks in the frame-based application are executed consecutively [2], [23], [34], the schedule of tasks on processors can be readily derived when the assignment and execution order of tasks are determined. The time complexity of Algorithm 1 is $O(|\mathcal{V}|^2 \cdot \mathcal{X})$, where $|\mathcal{V}|$ is the number of tasks in set $\mathcal{V}$ and $\mathcal{X}$ is the number of virtual cores.

VI. Experimental Results

Two sets of simulation experiments have been carried out to validate our task assignment heuristics in energy efficiency under the design constraints. In the first set of simulations, synthetic real-time tasks were generated to verify our heuristics while in the second set of simulations, real-life benchmarks were utilized to validate our heuristics. In the two sets of simulations, we compare the energy consumption of our task assignment heuristics with that of hybrid worst-fit genetic algorithm (HWGA) [29] and A*-search [36]. HWGA integrates a worst-fit based partition heuristics with the genetic algorithm to generate a task assignment that reduces the energy consumption while satisfying all the design constraints [29]. The worst-fit based partition heuristics assigns the task with the highest priority to the core with maximum remaining capacity. A*-search [36] is an optimum path finding algorithm that combines heuristic approaches like greedy best-first-search and formal approaches like Dijsktras algorithm. It is widely used in pathfinding and graph traversal. In this experiment, we take the A*-search algorithm as a benchmarking method to find the optimal solution of task assignment in terms of energy efficiency by treating the assignment of a task to a core as a path and the energy consumption of the task on its assigned core as the weight of the path. Finding the optimal task assignment is then equivalent to finding the path with minimal overall weight from the entry task to the end task.

All the algorithms were implemented in C++, and the simulations were performed on a machine with Intel Dual-Core 3.0 GHz processor and 8GB memory. For the sake of fair comparison, the same simulation settings are adopted for our heuristic algorithm and benchmarking algorithms HWGA [29] and A*-search [36].

TABLE IV: Parameters of the simulated platform [29].

P	f_m^{max}	α_m	γ_m	δ_m	R_m	C_m
P_1	3.3	20.5060	0.1666	3.656	0.282	340
P_2	3.4	5.0187	0.1942	2.138	0.487	295
P_3	3.3	12.7880	0.2043	3.645	0.288	320
P_4	3.0	15.6262	0.1942	4.556	0.238	320
P_5	3.2	20.6393	0.1574	3.204	0.278	295
P_6	3.1	11.9759	0.1586	2.719	0.480	255
P_7	3.0	10.3490	0.1124	2.074	0.661	335
P_8	2.6	13.1568	0.1754	2.332	0.680	380

A. Simulation for Synthetic Real-Time Tasks

The simulated MPSoC system P is assumed to consist of 8 interconnected heterogeneous processing units (i.e., $M = 8$), and each processing unit is assumed to support multiple discrete supply voltage and frequency pairs. The parameters of the simulated platform [29], including the maximal frequency f_m^{max}, hardware-dependent constants α_m, γ_m, δ_m, thermal resistance R_m, and thermal capacitance C_m of processor P_m, are given in TABLE IV. The number of frequency levels supported by processor P_m, denoted by x_m, is assumed to be varied from 3 to 5 for $1 \leq m \leq M$. In other words, each processor could have three levels of frequency at least and five levels of frequency at most. Given the maximal frequency f_m^{max} of processor P_m, the other $x_m - 1$ frequency levels are derived by the decrement of f_m^{max} with a step size of 0.2. The corresponding parameters such as hardware-dependent constants of the $x_m - 1$ frequency levels can be obtained using curve fitting techniques [27]. 30 synthetic real-time applications are utilized in the simulation, where the number of real-time tasks in an application is set to 100. The real-time tasks in an application are generated by assuming a common deadline D and the worst case execution cycles of tasks in the application are assumed to be in the range of $[4 \times 10^7, 6 \times 10^8]$. Precedence constraint is applied to randomly selected tasks in the application. The task activity factors μ are uniformly distributed in the interval $[0.4, 1]$, which demonstrates the heterogeneous nature of tasks [25].

TABLE V shows the energy consumed by the system when executing 30 synthetic real-time applications under four thermal constraints using our proposed algorithm and two benchmarking algorithms HWGA [29] and A*-search [36]. In addition to the energy consumption, the schedule feasibility of these applications using the three algorithms under the four thermal constraints are also given in the table. The thermal constraint takes the values of $T^{max} = 65°\text{C}$, 70°C, 75°C, and 80°C. NF indicates that tasks in an application cannot be feasibly scheduled under the constraints of task deadline, task precedence, and peak temperature limit. The schedule feasibility, denoted by $Feasibility$, is calculated as the ratio of the number of applications that can be successfully scheduled to the total number of applications adopted in the test.

As can be seen in the table, our proposed algorithm consumes less energy for a given thermal constraint when compared to HWGA [29], but more energy for a given thermal constraint when compared to A*-search [36]. For example, the average energy consumption E_{avg} of 30 applications using our proposed algorithm, HWGA [29], and A*-search [36] under the thermal constraint of $T^{max} = 80°\text{C}$ are 395.464, 408.568, and 388.960, respectively. The results in the table demonstrate that our algorithm is more efficient in saving energy than HWGA [29], which is due to the energy-optimality of our task assignment method, as analyzed in Section V-B.

A*-search [36] consumes the least energy among the three algorithms, but incurs the lowest schedule feasibility. This is because it attempts to find an energy-optimal solution without considering the design constraints. Unlike A*-search [36], the constraints of task deadline, task precedence and peak temperature limit are all examined in our task assignments. As a result, a higher schedule feasibility can be achieved by using our algorithm, as is shown in TABLE V.

B. Simulation for Real-Life Benchmarks

A heterogeneous MPSoC system [32] that consists of an AMD Athlon processor with three supply voltage/frequency levels and an TI DSP processor with two supply voltage/frequency levels is adopted in this simulation. The three supply voltage and frequency pairs of AMD Athlon are $(0.89\text{V}, 1.8\text{GHz})$, $(1.12\text{V}, 2.4\text{GHz})$, $(1.34\text{V}, 3\text{GHz})$, and the two supply voltage and frequency pairs of TI DSP are $(0.98\text{V}, 2.0\text{GHz})$, $(1.42\text{V}, 3.0\text{GHz})$ [32], respectively. Four

TABLE V: Energy consumption of 30 applications and schedule feasibility using our proposed algorithm and two benchmarking algorithms under four system thermal constraints.

	$T^{max} = 65°\mathrm{C}$			$T^{max} = 70°\mathrm{C}$			$T^{max} = 75°\mathrm{C}$			$T^{max} = 80°\mathrm{C}$		
Application	Proposed	HWGA	A*-search	Proposed	HWGA	A*-search	Proposed	HWGA	A*-search	Proposed	HWGA	A*-search
1	482.228	506.427	476.568	452.089	467.988	446.725	421.950	429.491	415.918	391.811	406.044	385.826
2	490.698	497.805	485.312	460.029	472.000	453.982	429.361	436.778	424.346	398.692	413.229	393.632
3	507.584	512.605	498.251	475.860	481.568	470.584	444.136	451.924	436.904	412.412	419.549	406.288
4	453.556	460.159	444.299	425.208	446.328	419.479	396.861	412.671	390.915	368.514	388.233	363.365
5	457.452	470.486	449.825	428.862	441.862	423.350	400.271	413.017	394.670	371.680	390.136	366.284
6	477.664	489.138	NF	447.810	466.089	440.326	417.956	424.602	411.798	388.102	400.758	382.957
7	531.941	NF	526.061	498.695	512.196	491.594	465.448	471.996	458.844	432.202	444.893	425.498
8	446.369	452.795	432.817	418.471	431.285	412.138	390.573	402.029	382.654	362.675	383.410	355.570
9	505.489	NF	NF	473.896	NF	NF	442.303	453.935	436.803	410.710	418.109	405.396
10	485.435	494.051	478.776	455.095	461.686	449.022	424.755	439.897	418.345	394.416	409.431	389.357
11	488.123	498.365	481.883	457.615	481.264	450.403	427.107	443.397	419.262	396.600	405.428	391.330
12	500.842	513.972	494.739	469.539	479.676	462.303	438.237	445.092	429.675	406.934	427.327	401.831
13	487.772	506.865	480.653	457.286	471.735	451.516	426.800	442.913	421.195	396.315	410.895	389.049
14	480.649	495.976	473.142	450.609	479.333	444.715	420.568	432.750	408.831	390.528	404.564	385.509
15	504.477	516.137	492.976	472.947	481.738	466.518	441.417	451.412	428.322	409.888	416.215	404.142
16	476.658	488.782	467.039	446.867	458.826	437.785	417.076	428.075	412.060	387.285	402.364	377.406
17	469.505	474.798	464.176	440.161	446.242	432.030	410.817	422.121	405.650	381.472	390.985	373.809
18	498.723	507.706	NF	467.553	476.240	NF	436.383	455.402	NF	405.213	416.184	398.115
19	427.506	439.832	421.298	400.787	426.220	392.632	374.068	395.864	366.580	347.349	365.681	342.165
20	519.028	533.134	513.742	486.589	498.886	479.880	454.149	464.580	446.207	421.710	430.893	413.132
21	503.147	512.793	495.077	471.700	483.977	464.183	440.253	445.522	432.895	408.807	425.162	402.179
22	486.329	501.361	477.218	455.933	463.495	450.749	425.538	440.754	419.941	395.142	407.663	389.943
23	502.422	518.660	496.205	471.021	478.478	461.182	439.619	450.448	428.530	408.218	423.009	400.609
24	508.554	520.061	502.602	476.769	484.221	471.418	444.985	450.425	439.078	413.200	418.294	405.728
25	483.440	490.729	NF	453.225	464.227	NF	423.010	436.123	NF	392.795	400.772	384.852
26	492.310	498.886	485.066	461.540	479.612	456.422	430.771	441.679	422.911	400.002	417.019	392.488
27	514.671	520.028	508.552	482.504	496.335	476.282	450.337	460.938	442.237	418.170	423.311	412.986
28	487.877	500.144	480.047	457.385	465.145	452.210	426.893	435.066	416.592	396.400	411.564	390.579
29	455.779	485.258	NF	427.293	461.245	419.694	398.807	417.818	392.252	370.320	393.833	360.958
30	475.506	485.741	468.291	445.787	465.433	437.745	416.068	434.566	408.832	386.349	392.088	377.816
E_{avg}	**486.724**	**496.168**	**479.985**	**456.304**	**469.77**	**448.699**	**425.884**	**437.710**	**418.295**	**395.464**	**408.568**	**388.960**
$Feasibility$	**100%**	**93.3%**	**83.3%**	**100%**	**96.7%**	**90%**	**100%**	**100%**	**93.3%**	**100%**	**100%**	**100%**

TABLE VI: Parameters of the real-life benchmarks [32].

Application	Description	Expected Task Execution Time	Standard Deviation	Number of Tasks in the Application
mpegplay	MPEG video decoder	113.4	38.9	30
madplay	MP3 audio decoder	43.1	34.8	40
tmndec	H.263 video decoder	89.5	34.7	20
toast	GSM speech encoder	5.6	4.7	30

TABLE VII: Energy consumption, schedule feasibility, and peak temperature of four benchmarks using our proposed algorithm and two benchmarking algorithms under four system thermal constraints.

	$T^{max} = 65°\mathrm{C}$			$T^{max} = 70°\mathrm{C}$			$T^{max} = 75°\mathrm{C}$			$T^{max} = 80°\mathrm{C}$		
Application	Proposed	HWGA	A*-search	Proposed	HWGA	A*-search	Proposed	HWGA	A*-search	Proposed	HWGA	A*-search
mpegplay	275.932	284.663	266.606	258.686	264.093	231.949	241.441	251.042	230.639	224.195	234.021	213.645
madplay	153.966	159.139	138.501	144.343	151.029	128.777	134.720	142.019	122.580	125.098	131.144	116.070
tmndec	119.256	125.494	106.916	111.802	124.371	104.443	104.349	109.790	88.681	96.895	101.948	91.080
toast	13.017	14.303	11.716	12.204	13.363	10.630	11.390	13.152	10.179	10.577	11.723	9.463
E_{avg}	**140.543**	**145.900**	**130.935**	**131.759**	**138.214**	**118.950**	**122.975**	**129.001**	**113.020**	**114.191**	**119.709**	**107.565**
$Feasibility$	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
T_{avg}^{peak}	**61.051**	**61.752**	**64.130**	**66.328**	**67.494**	**69.822**	**71.682**	**72.363**	**74.643**	**75.154**	**77.602**	**79.349**

real-life multimedia applications, that is, mpegplay, madplay, tmndec, and toast, are adopted for evaluations. The parameters of tasks in the four practical applications, including the expected value and standard deviation of task execution time, and the number of tasks in each application, are listed in TABLE VI.

TABLE VII shows the energy consumption, schedule feasibility, and peak temperature of four benchmarks using our proposed algorithm and two benchmarking algorithms under four system thermal constraints (i.e., $T^{max} = 65°\mathrm{C}$, $70°\mathrm{C}$, $75°\mathrm{C}$, and $80°\mathrm{C}$). From the simulation results of real-life benchmarks, we can draw the same conclusion as in TABLE V that our algorithm consumes less energy when compared to HWGA [29] but higher energy when compared to A*-search [36] for a given thermal constraint. For instance, the average energy consumption E_{avg} of four benchmarks using our algorithm, HWGA [29], and A*-search [36] under the thermal constraint of $T^{max} = 75°\mathrm{C}$ are 122.975, 129.001, and 113.020, respectively.

In this set of simulations, all the four benchmarks can be feasibly scheduled under the design constraints, thus we further compare the peak temperature of the four benchmarks using our algorithm, HWGA [29], and A*-search [36]. As demonstrated in TABLE VI, the peak temperature T_{avg}^{peak} achieved by our algorithm is lower than that of benchmarking algorithms HWGA [29] and A*-search [36] on average. Taking the case of $T^{max} = 65°\text{C}$ as an example, the average peak temperature of four benchmarks achieved by our algorithm, HWGA [29], and A*-search [36] are $61.051°\text{C}$, $61.752°\text{C}$, and $64.130°\text{C}$, respectively.

VII. Conclusion

In this paper, we propose a two-level scheduling approach to reduce energy consumption of DVFS-enabled heterogeneous MPSoCs under constraints of task deadline, task precedence, and peak temperature limit. The proposed algorithm computes high quality scheduling solutions in two correlated optimization steps of different scales. At the processor level, a multi-processor model supporting DVFS is transformed to a virtual multi-processor model supporting only one fixed frequency level. At the core level, real-time tasks are assigned to individual cores of the virtual processor under the constraints of task precedence and peak temperature limit. Two sets of simulation experiments have been conducted to validate the effectiveness of the proposed algorithm in saving energy and improving schedule feasibility. Simulation results have demonstrated that the proposed algorithm achieves better performance when compared to the benchmarking schemes.

References

[1] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "Exploiting structural duplication for lifetime reliability enhancement," *The Proceedings of the International Symposium on Computer Architecture (ISCA)*, pp. 520-531, 2005.

[2] J. Zhou, X. Hu, Y. Ma, and T. Wei, "Balancing lifetime and soft-error reliability to improve system availability," *The Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 685-690, 2016.

[3] J. Zhou, J. Yan, J. Chen, and T. Wei, "Peak temperature minimization via task allocation and splitting for heterogeneous MPSoC real-time systems," *Journal of Signal Processing Systems*, vol. 84, no. 1, pp. 111-121, 2016.

[4] R. Kumar and D. M. Tullsen, "Core architecture optimization for heterogeneous chip multiprocessors," *The Proceedings of International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 23-32, 2006.

[5] J. Zhou, J. Yan, T. Wei, M. Chen, and X. Hu, "Energy-adaptive scheduling of imprecise computation tasks for QoS optimization in real-time MPSoC systems," *The Proceedings of Design, Automation & Test in Europe (DATE)*, pp. 1402-1407, 2017.

[6] M. Chen, S. Huang, X. Fu, X. Liu, and J. He, Statistical Model Checking-Based Evaluation and Optimization for Cloud Workflow Resource Allocation, *IEEE Transactions on Cloud Computing*, 2016.

[7] F. Wang, C. Nicopoulos, X. Wu, X. Xie, and N. Vijaykrishnan, "Variation-aware task allocation and scheduling for MPSoC," *The Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, pp. 598-603, 2007.

[8] A. Colin, A. Kandhalu and R. Rajkumar, "Energy-efficient allocation of real-time applications onto heterogeneous processors," *The Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 1-10, 2014.

[9] D. Li and J. Wu, "Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 810-823, 2015.

[10] Y. Wang, K. Li, H. Chen, L. He and K. Li, "Energy-aware data allocation and task scheduling on heterogeneous multiprocessor systems with time constraints," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 134-148, 2014.

[11] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Information Sciences*, vol. 379, pp. 241-256, 2017.

[12] H. Zahaf, A. Benyaminab, R. Olejnika, and G. Lipari, "Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms," *Journal of Systems Architecture*, vol. 74, pp. 46-60, 2017.

[13] W. Hung, Y. Xie, N. Vijaykrishnan, and M. Kandemir, "Thermal-aware task allocation and scheduling for embedded systems," *The Proceedings of Design, Automation, & Test in Europe (DATE)*, pp. 898-899, 2005.

[14] Y. Liu, R. Dick, L. Shang, and H. Yang, "Thermal vs energy optimization for DVFS-enabled processors in embedded systems," *The Proceedings of the International Symposium on Quality Electronic Design (ISQED)*, pp. 204-209, 2007.

[15] S. Saha, Y. Lu, and J. Deogun, "Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems," *The Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 41-50, 2012.

[16] B. Barrefors, Y. Lu, S. Saha and J. Deogun, "A novel thermal-constrained energy-aware partitioning algorithm for heterogeneous multiprocessor real-time systems," *The Proceedings of the International Conference on Performance Computing and Communications (IPCCC)*, pp. 1-8, 2014.

[17] J. Zhou, T. Wei, M. Chen, Y. Ma and X. Hu, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 8, pp. 1269-1282, 2016.

[18] L. He, H. Zhu and S. Jarvis, "Developing Graph-based Co-scheduling Algorithms on Multicore Computers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1617-1632, 2016.

[19] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1668-1681, 2012.

[20] M. Gerards, J. Hurink, and J. Kuper, "On the interplay between global DVFS and scheduling tasks with precedence constraints," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1742-1754, 2015.

[21] Y. Lee and A. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," *The Proceedings of the International Symposium on Cluster Computing and the Grid (CCGRID)*, pp. 92-99, 2009.

[22] J. Liu, K. Li, D. Zhu, J. Han, and K. Li, "Minimizing cost of scheduling tasks on heterogeneous multicore embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 2, pp. 1-25, 2016.

[23] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 316-328, 2010.

[24] Y. Liu, R. Dick, L. Shang, and H. Yang, "Thermal vs energy optimization for DVFS-enabled processors in embedded systems," *The Proceedings of the International Symposium on Quality Electronic Design (ISQED)*, pp. 204-209, 2007.

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

[26] W. Liao, L. He, and K. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1042-1053, 2006.

[27] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature constraint hard real-time periodic tasks," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 329-339, 2010.

[28] N. Weste and K. Eshraghian, "Principles of CMOS VLSI design: A system perspective," *Addison-Wesley Publishing Company*, 1992.

[29] S. Saha, Y. Lu, and J. Deogun, "Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems," *The Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 41-50, 2012.

[30] J. Zhou and T. Wei, "Stochastic thermal-aware real-time task scheduling with considerations of soft errors," *Journal of Systems and Software*, vol. 102, pp. 123-133, 2015.

[31] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94-125, 2004.

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

[33] HotSpot. University of Virginia. [Online]. Available: http://lava.cs.virginia.edu/HotSpot.

[34] D. Zhu, R. Melhem, and B. Childers, "Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems," *The Proceedings of the International Symposium on Real-Time Systems (RTSS)*, pp. 84-94, 2001.

[35] J. Chen, H. Hsu, and T. Kuo, "Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems," *The Proceedings of the International Symposium on Real-Time and Embedded Technology and Applications (RTAS)*, pp. 408-417, 2006.

[36] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," *Prentice Hall, 3rd edition*, 2009.

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