TA-MCF: Thermal-Aware Fluid Scheduling for Mixed Criticality System

Article *in* Journal of Circuits, Systems and Computers · May 2018

DOI: 10.1142/S0218126619500294

5 authors, including:

Tiantian Li
Northeastern University (Shenyang, China)
14 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Jie Song
Northeastern University (Shenyang, China)
24 PUBLICATIONS **109** CITATIONS
SEE PROFILE

Journal of Circuits, Systems, and Computers
Vol. 28, No. 2 (2019) 1950029 (19 pages)

DOI: 10.1142/S0218126619500294

World Scientific
www.worldscientific.com

TA-MCF: Thermal-Aware Fluid Scheduling for Mixed-Criticality System*

Tiantian Li[†,§], Tianyu Zhang[†,¶], Ge Yu[†,‖], Yichuan Zhang[‡,**] and Jie Song[‡,††]

[†]*College of Computer Science and Engineering, Northeastern University, P. R. China*

[‡]*Software College, Northeastern University, P. R. China*
[§]*litiantian_neu@163.com; litiantian@stumail.neu.edu.cn*
[¶]*tyzhang@stumail.neu.edu.cn*
[‖]*yuge@mail.neu.edu.cn*
[**]*zhangyc@mail.neu.edu.cn*
[††]*songjie@mail.neu.edu.cn*

Received 3 December 2017
Accepted 20 April 2018
Published

Fluid scheduling allows tasks to be allocated with fractional processing capacity, which significantly improves the schedulability performance. For dual-criticality systems (DCS), dual-rate fluid-based scheduling has been widely studied, e.g., the state-of-the-art approaches mixed-criticality fluid scheduling (MCF) and MC-Sort. However, most of the existing works on DCS either only focus on the schedulability analysis or minimize the energy consumption treating leakage power as a constant. To this end, this paper considers the effect of temperature on leakage power and proposes a thermal and power aware fluid scheduling strategy, referred to as thermal and energy aware (TA)-MCF which minimizes both the energy consumption and temperature, while ensuring a comparable schedulability ratio compared with the MCF and MC-Sort. Extensive experiments validate the efficiency of TA-MCF.

Keywords: Thermal-aware; temperature minimization; energy minimization; mixed-criticality system; fluid scheduling.

1. Introduction

Energy minimization has become the one prime goal to achieve in the field of embedded system design due to both cost reduction reasons and related thermal issues.[1] Meanwhile, with the drastically increasing power density of processors caused by the aggressive shrinking of chip size, temperature has also become another

*This paper was recommended by Regional Editor Tongquan Wei.
§Corresponding author.

urgent issue to cope with. Further, since a vicious circle exists between the temperature and the temperature-related leakage power consumption, it is necessary to explore scheduling methods with taking both power and temperature into consideration.

An increasing trend in the embedded system design is to integrate applications with different criticality levels onto a shared platform for better cost and power efficiency. Such *mixed-criticality* systems (MCS), first proposed by Vestal ~~in~~,[2] have been widely studied in the last decade (e.g., Refs. 1, 3–15). However, studies on minimizing both energy and temperature for MCS, the focus of this paper, are relatively few. On the other hand, techniques of energy and temperature minimization for conventional real-time systems cannot be straightforwardly applied to MCS due to the property of multiple execution modes of the system which aggravates the problem of schedulability analysis. All above reasons motivate us to explore an efficient and effective scheduling strategy considering both energy consumption and processor temperature for MCS in this work.

Task scheduling for MCS has been well studied in the literature and several distinguished algorithms have been proposed, e.g., period transformation (PT),[4] dynamic ~~EPF~~ with ~~visual~~ deadline (EDF-VD),[5–8] MC-Fluid[9,10] and MC-Sort.[11] An important observation achieved from these works is that fluid scheduling not only obtains a higher schedulability compared with other approaches (e.g., EDF-VD,[10]) but also enables us to eliminate the oscillating feature of the processor energy and temperature exhibited in other scheduling strategies. Therefore, fluid scheduling maintains a higher possibility to achieve the optimal condition of both energy and temperature for MCS. However, existing works studying fluid scheduling mostly focus on the schedulability analysis without considering both energy and temperature. Thus, this work aims to explore a thermal and energy aware-mixed-criticality fluid scheduling (TA-MCF).

AQ: Please check the edit.

In this paper, we consider a generalized power model by taking into account both the temperature-dependent leakage power and the task switching activity factor which can result in different power values. By proving the thermal and energy optimal conditions for the nonmixed-criticality fluid scheduling, we deduce the optimal condition for MCF. In spite of the advantages of fluid scheduling, it is not realistic to directly implement fluid-based scheduling on real systems due to the stringent assumption that one processor can execute multiple tasks at the same time. To tackle this challenge, we present an approximate fluid scheduling to reduce the high task switching overhead caused by the original algorithm. Our extensive experimental evaluation validates that the proposed TA-MCF significantly reduce the energy consumption and temperature with an acceptable loss of schedulability compared with the state-of-the-art MCF[10] and MC-Sort.[11]

The rest of this paper is organized as follows: Section 2 introduces the related work, while Sec. 3 describes the system models we use, including the task model, power and thermal model. Section 4 first conducts the schedulability analysis of

MCS, then deduces the thermal and energy optimal condition for dual-criticality system (DCS) scheduling and finally proposes our thermal and energy aware fluid scheduling algorithm for DCS based on the optimal condition deduced. Section 5 designs extensive experiments to validate the efficiency of the proposed algorithm while Sec. 6 summarizes this work.

2. Related Work

In the last decade, MCS have been widely studied and several distinguished scheduling approaches have been proposed. Reference 4 evaluates several static scheduling approaches for sporadic tasks in MCS, such as PT. Reference 5 first proposes the dynamic EDF with virtual deadlines (EDF-VD) scheduling for MCS, and proves its effectiveness by a speedup factor of 1.619 for a DCS on uniprocessor. Aiming at an imprecise mixed-criticality task model, Ref. 8 presents a utilization-based schedulability test for EDF-VD scheduling and proves its sub-optimality in terms of speedup factors. Reference 7 improves EDF-VD enabling it to deal with a more general mixed-criticality model. Further, to fully exploit the schedulability of EDF-VD, authors in Ref. 6 have developed new schedulability analysis methods for EDF-VD. The new method analyzes the system behavior across multiple criticality levels so as to obtain a more precise analysis, which results in a higher schedulability with the cost of higher complexity compared with traditional approaches. Reference 16 also proposes a schedulability analysis method for arbitrarily activated tasks in MCS. These more general schedulability analysis methods can help us explore much more efficient scheduling approaches.

In recent years, due to the performance in terms of schedulability in multiprocessor systems, fluid scheduling for MCS attracts more attention in the literature. Reference 9 first analyzes the exact schedulability condition of MC-Fluid scheduling, and then proposes an optimal assignment algorithm for criticality-dependent task execution rates with polynomial complexity. Based on the results in Ref. 9, authors in Ref. 10 propose a simplified version called MCF with linear complexity with an acceptable loss of schedulability. Further, Ref. 11 proposes two new algorithms, referred to as MC-Sort and MC-Slope, both with linearithmic complexity ($n\log n$). The schedulability ratio of the proposed algorithms is significantly improved compared with MCF,[10] and is very close to the optimal algorithm MC-Fluid with a reduced runtime complexity. Considering the difficulty of implementing fluid scheduling in real systems, Ref. 9 proposes the MC-DP-Fair which transforms a fluid schedule into a nonfluid one while preserving the same schedulability properties. However, all the above discussed approaches fail to consider minimizing both the energy consumption and temperature.

Existing thermal and energy aware studies on MCS usually consider temperature as a constraint to minimize system energy consumption, rather than minimize both the temperature and energy consumption which can fully explore the optimization

space. In terms of energy consumption minimization, techniques like dynamic voltage and frequency scaling (DVFS) and dynamic power management (DPM) have been adopted by a lot of researches. By DPM, Ref. 12 aims to find an appropriate trade-off between the deadline miss ratio and energy consumption for low-criticality tasks while ensuring the deadlines of high-criticality tasks. Reference 14 advocates to adopt energy utilization as the optimization goal rather than energy when the power supply is not sufficient. However, all the works ignore the effect of temperature on the leakage power by considering it as a constant. By DVFS, Ref. 17 aims at makespan minimization under reliability and temperature constraints, and Ref. 18 aims at fault-tolerant task scheduling for MCS, both of which do not focus on energy consumption minimization. Reference 19 proposes a hybrid approach to find the energy-efficient speeds for traditional real-time tasks which cannot be directly applied to MCS and it also treats the leakage power as a constant rather than a variant changing with temperature. Reference 15 proposes an energy-efficient solution to find the optimal speed setup for tasks with different criticalities as well as the virtual deadline scaling factor for high criticality tasks in a unicore MCS under EDF-VD scheduling. However, this work only considers LO-mode dynamic power consumption while temperature-related leakage power consumption is neglected. Compared with Ref. 15, Ref. 1 further considers the total power consumption (including the leakage power) in both LO and HI modes. However, they treat the leakage power as a constant in order to get the critical speed without considering its changeability with temperature. Considering dynamic leakage power, Ref. 20 proposes a two-stage energy-efficient task scheduling, and Refs. 21 and 22 study the thermal utilization and energy minimization of traditional real-time systems under fluid scheduling. However, these studies cannot be straightforwardly applied to MCS due to the property of multiple execution modes of the system which aggravates the problem of schedulability analysis.

3. System Model

In this work, we follow the mixed-criticality literature to use a DCS as the system model.

3.1. *Dual-criticality task model*

Compared with traditional task, an MC task is characterized by an additional feature called criticality level. In a DCS with task set $\Gamma = \{\tau_1, \ldots, \tau_n\}$, each task τ_i has either high (HI) or low (LO) criticality and can be denoted as $\tau_i = \langle A_i, T_i, C_i^L, C_i^H, \chi_i \rangle$, where A_i is the switching activity factor, T_i is the minimum inter-arrival time (deadline D_i is omitted here, since it is assumed to be equal to T_i), C_i^L and C_i^H, respectively, correspond to a less and a more conservative estimates of the worst case execution time (WCET) and $\chi_i \in \{\text{HI}, \text{LO}\}$ denotes the criticality level.

A DCS can be in two modes: LO and HI. The system starts with the LO mode and remains there as long as each job released by task τ_i completes execution within C_i^L time. Otherwise, if any job released by task τ_i executes for more than the C_i^L time, the system will immediately switch into the HI mode, and all of the LO tasks will be dropped to guarantee the schedulability of HI tasks. Furthermore, the system could also switch back from HI mode to LO mode, providing that all tasks can meet their deadlines after transiting back.[23,24]

3.2. *Power and thermal model*

The state-of-the-art power model adopted in this paper is described as follows:

$$P_d(t) = A(t) \cdot s(t)^3, \quad P_s(t) = \alpha\Theta(t) + \beta, \quad P(t) = P_d(t) + P_s(t)\,, \tag{1}$$

where $P_d(t)$ refers to the dynamic power depending on the task-related switching factor $A(t)$ and the processor speed $s(t)$, while $P_s(t)$ refers to the static power due to the leakage current depending on the processor temperature $\Theta(t)$ (α and β are positive linear fitting constants). Here, instead of using the accurate but complex leakage model,[25] we adopt a linear leakage model which has been proved to be accurate enough in Ref. 26.

In addition, according to the widely used RC thermal model,[27] temperature can be characterized as the following equation:

$$\frac{d\Theta(t)}{dt} = \frac{P(t)}{C_{\text{th}}} - \frac{\Theta(t) - \Theta_a}{R_{\text{th}}C_{\text{th}}}\,, \tag{2}$$

where R_{th} refers to the thermal resistance, C_{th} refers to the thermal capacitance and Θ_a refers to the ambient temperature. Through mathematical computing, the temperature function can be derived as follows:

$$\begin{aligned} &\theta(t) = C_{\text{th}}\left(\Theta(t) - \frac{R_{\text{th}}\beta + \Theta_{\text{a}}}{1 - R_{\text{th}}\alpha}\right), \qquad \theta'(t) = P_{\text{d}}(t) - \lambda\theta(t)\,, \\ &\theta(t) = \int_0^t P_{\text{d}}(u)e^{\lambda(u-t)}du + \theta(0)e^{-\lambda t}, \quad \lambda = \frac{1}{R_{\text{th}}C_{\text{th}}} - \frac{\alpha}{C_{\text{th}}}\,. \end{aligned} \tag{3}$$

When dynamic power $P_d(t)$ is a constant, P_d, $\theta(t)$ can be simplified as

$$\theta(t) = (P_d/\lambda)(1 - e^{-\lambda t}) + \theta(0)e^{-\lambda t}\,. \tag{4}$$

According to Eq. (4), we can see that $\theta(t)$ is monotonically increasing with P_d. By Eq. (1), $P_s(t)$ is shown to be monotonically increasing with $\Theta(t)$, which can be expressed as a linear function of $\theta(t)$ with a positive coefficient as $1/C_{\text{th}}$ (according to Eq. (3)). That is, $P_{\text{s}}(t)$ can be minimized if $\theta(t)$ is minimized through minimizing P_d. Based on such an analysis, we have the following lemma.

Lemma 1. *When dynamic power $P_d(t)$ is constant, say P_d, both the processor temperature and the system power consumption can be minimized when P_d is minimized.*

When switching onto multiprocessor platforms, RC model should be modified as

$$\frac{d\Theta_i(t)}{dt} = \frac{P_i(t)}{C^i_{\text{th}}} - \frac{\Theta_i(t) - \Theta_{\text{a}}}{R^{ii}_{\text{th}} C^i_{\text{th}}} - \sum_{j \neq i} \frac{\Theta_i(t) - \Theta_j(t)}{R^{ij}_{\text{th}} C^i_{\text{th}}}\,, \tag{5}$$

where R^{ij}_{th} refers to the thermal resistance between processors i and j, C^i_{th} refers to the thermal capacitance of processor i. It is trivial to prove that Lemma 1 still holds when analyzing the temperature of each core under multiprocessor setting.

4. TA-MCF Scheduling

AQ: Kindly check edit.

Different from the existing research, TA-MCF is a thermal-aware enforced version of the existing MCF, which considers minimization of both temperature and energy consumption due to the close inner relationship between them. Fluid scheduling has been studied by quite a lot of researches like Refs. 9–11 for MCS, in which each job of a task τ_i will run with a fixed execution rate ω_i^χ during any time interval in the system mode χ $(0 < \omega_i^\chi \leqslant 1)$. For a DCS, TA-MCF seeks the optimal values of ω_i^L and ω_i^H in the LO and HI modes to minimize both temperature and energy consumption.

4.1. *MCF schedulability analysis*

Schedulability analysis for fluid scheduling consists of two aspects: task schedulability and rate feasibility. For a DCS upon m processors, task schedulability can be analyzed in the LO and HI modes, respectively. LO-mode schedulability, which is quite intuitive, can be guaranteed if $\omega_i^L \geqslant \mu_i^L$ $(\mu_i^L = C_i^L/T_i)$. For HI-mode schedulability, it is determined by the correct transition from LO mode to HI mode, which can be guaranteed by the following condition[9]:

$$\omega_i{}^L \geqslant \frac{\mu_i{}^L \omega_i{}^H}{\omega_i{}^H - (\mu_i{}^H - \mu_i{}^L)}, \quad \chi_i = \text{HI}\,. \tag{6}$$

As to the rate feasibility, it guarantees the correct execution of all active jobs, requiring that task execution rates ω_i^L and ω_i^H satisfy the following condition, where $\Gamma_H = \{\tau_i | \tau_i \in \Gamma, \chi_i = \text{HI}\}$:

$$\sum_{\Gamma} \omega_i^L \leqslant m, \omega_i^L \in (0,1]; \quad \sum_{\Gamma_H} \omega_i^H \leqslant m,\ \omega_i^H \in (0,1]\,. \tag{7}$$

4.2. *TA-MCF optimal condition*

TA-MCF is a thermal-aware version of MCF,[10] which integrates DVFS into the minimization of not only energy consumption but also temperature. It also considers the activity factor differences of tasks which are usually neglected by other works,

and further explores the optimal space without sacrificing much the schedulability of MCF. This section derives the optimal condition of TA-MCF to achieve its optimization goal.

As we know, whatever the goal is, system optimization should be conducted with the precondition of schedulability. Once speed scaling is considered, the schedulability condition presented in Sec. 4.1 should be correspondingly modified. Denote s_i^L and s_i^H as the processor speeds executing task τ_i in the LO mode and HI mode. Then, LO-mode schedulability condition is simply modified as $\omega_i^L \geqslant \mu_i^L / s_i^L$, while the HI-mode schedulability condition is re-deduced as inequation (9). Let t_o denote the first switching time instant from the LO mode to HI mode. For a job of HI task τ_i that is active at t_o, suppose it has arrived at $(t_o - t_i)$, where $0 \leqslant t_i \leqslant T_i$. Then, during $[t_o - t_i, t_o)$, the following inequation must hold:

$$t_i \cdot \omega_i^L \leqslant {C_i}^L / {s_i}^L \iff t_i \leqslant {C_i}^L / (\omega_i^L {s_i}^L)\,. \tag{8}$$

During interval $[t_o, t_o - t_i + T_i]$, to meet the job's deadline, the following inequation must hold (among which $\omega_i^H \geqslant \omega_i^L$, see Corollary 4 in Ref. 9, which proves that we only need to analyze this case for the task schedulability of a HI task in HI mode):

$$\begin{aligned} & t_i \cdot {\omega_i}^L + (T_i - t_i) \cdot {\omega_i}^H \geqslant \frac{{C_i}^L}{{s_i}^L} + \frac{{C_i}^H - {C_i}^L}{{s_i}^H} \\ \Leftarrow\ & T_i {\omega_i}^H - \frac{{C_i}^L({\omega_i}^H - {\omega_i}^L)}{{\omega_i}^L {s_i}^L} \geqslant \frac{{C_i}^L}{{s_i}^L} + \frac{{C_i}^H - {C_i}^L}{{s_i}^H} \\ \iff\ & {\omega_i}^L \geqslant \frac{{\mu_i}^H}{{s_i}^L} = \frac{R \cdot {\mu_i}^L}{{s_i}^L}, \quad R = \frac{{\mu_i}^H}{{\mu_i}^L} = \frac{{C_i}^H}{{C_i}^L}\,. \end{aligned} \tag{9}$$

Here, R is assumed as a constant and it is true in some real-life applications like fight management system.[1,15,23]

Before the description of the TA-MCF optimal condition, we first introduce the following theorem.

Theorem 1 (NonMCF Optimal Condition). *Under fluid scheduling, both processor temperature and dynamic power consumption are minimized when the speeds of tasks are assigned as*

$$\begin{aligned} & s_i = \frac{{A_i}^{-1/3} \sum_{\Gamma} \mu_i {A_i}^{1/3}}{U_{\max}}, \quad \mu_i = C_i / T_i; \\ & P_i = A_i s_i^3 = \left(\sum_{\Gamma} \frac{\mu_i {A_i}^{1/3}}{U_{\max}} \right)^3 = A_j s_j^3 = P_j, \quad \forall i, j\,, \end{aligned} \tag{10}$$

where $U_{\max}$ is the maximal possible system utilization.

Proof. Under fluid scheduling, the dynamic power consumption of the processor (P_d) keeps unchanged since each task will be executed with a fixed rate at any time. P_d can be expressed as $\sum_{\Gamma}(C_i/T_i s_i)A_i s_i^3$, with a constraint of $\sum_{\Gamma}(C_i/T_i s_i) \leqslant U_{\max}$ ($U_{\max}$ is the maximal possible system utilization, say 1 in a homogeneous system). Then, by the Lagrange Multiplier method and the famous Karush–Kuhn–Tucker (KKT) condition, P_d is minimized when the condition shown in Eq. (10) is satisfied. Combining with Lemma 1, Theorem 1 can be proved. □

When considering each criticality mode separately, an MCF system can be treated as a nonMCF system and thus the above conclusion can be readily applied. In particular, the DCS considered in this paper can be analyzed, respectively, in the LO mode and HI mode.

As stated before, TA-MCF aims at minimizing both the energy consumption and temperature. For a sporadic task system $\Gamma = \{\tau_1, \ldots, \tau_n\}$, the energy consumed in a hyperperiod (the least common multiplier of T_i) consists of two parts: $P_d(t)$ and $P_s(t)$. Due to the executing characteristic of fluid scheduling, it is trivial to find that $P_d(t)$ is a constant, which we denote it as P_d. Therefore, according to Lemma 1, both processor temperature and system energy consumption can be minimized when P_d is minimized. Furthermore, by Theorem 1, condition (10) constitutes the nonMCF optimal condition of both temperature and dynamic power consumption. Therefore, for the DCS studied in this work, TA-MCF optimal condition will be explored separately in the LO mode and HI mode.

As per the above discussion, system optimization should be conducted with the precondition of schedulability summarized as: (1) $\mu_i^L/s_i^L \leqslant \omega_i^L \leqslant 1$, $\mu_i^H/s_i^H \leqslant \omega_i^H \leqslant 1$; $\mu_i^L \leqslant s_i^L \leqslant 1$; $\mu_i^H \leqslant s_i^H \leqslant 1$; (2) $\sum_{\Gamma}\omega_i^L \leqslant m$, $\sum_{\Gamma}\omega_i^H \leqslant m$; (3) For any HI task $\tau_i \in \Gamma_H$, $\omega_i^L \geqslant R \cdot \mu_i^L/s_i^L$.

In LO mode, taking speed scaling into consideration, P_d can be expressed as

$$P_d = \frac{1}{\prod\limits_{\tau_i\in\Gamma} T_i} \sum_{\tau_i\in\Gamma} \frac{\prod\limits_{\tau_i\in\Gamma} T_i}{T_i} \cdot \frac{{C_i}^L}{{s_i}^L} \cdot A_i({s_i}^L)^3 = \sum_{\tau_i\in\Gamma} \frac{{\mu_i}^L}{{s_i}^L} \cdot A_i \cdot ({s_i}^L)^3 , \tag{11}$$

Due to the different constraints imposed on the HI and LO tasks caused by the system mode switch, the speed assignments must be considered separately. For LO tasks, it is trivial to guarantee their deadlines by setting $\omega_i^L = \mu_i^L/s_i^L$. For HI tasks, by Theorem 1, the optimal speed values of HI tasks can be achieved when $U_{\mathrm{H}}^{L\prime} = \sum_{\Gamma_{\mathrm{H}}} \mu_i^L/s_i^L$ is maximized. Under the constraints stated above, we can get that

$$U_H^{L\prime} = \sum_{\Gamma_H} \frac{{\mu_i}^L}{{s_i}^L} \leqslant \frac{1}{R}\left(m - \sum_{\Gamma_L} \frac{{\mu_i}^L}{{s_i}^L}\right) = \frac{1}{R}(m - U_L^{L\prime}) . \tag{12}$$

Then, by Theorem 1, LO mode dynamic power P_d is minimized under given $U_L^{L\prime}$ when the speeds of tasks are assigned as

$$s_i^L = \begin{cases} \dfrac{A_i^{-1/3} \sum\limits_{\Gamma_H} \mu_i^L A_i^{1/3}}{(m - U_L^{L\prime})/R}, & \tau_i \in \Gamma_H (\Gamma_H = \{\tau_i | \tau_i \in \Gamma, \chi_i = \mathrm{HI}\}), \\ \dfrac{A_i^{-1/3} \sum\limits_{\Gamma_L} \mu_i^L A_i^{1/3}}{U_L^{L\prime}}, & \tau_i \in \Gamma_L (\Gamma_L = \{\tau_i | \tau_i \in \Gamma, \chi_i = \mathrm{LO}\}). \end{cases} \tag{13}$$

With such speeds assignment, P_d can be calculated as

$$\begin{aligned} P_d = P_L^L + P_H^L &= \sum_{\tau_i \in \Gamma_L} \frac{\mu_i^L}{s_i^L} \cdot A_i \cdot (s_i^L)^3 + \sum_{\tau_i \in \Gamma_H} \frac{\mu_i^L}{s_i^L} \cdot A_i \cdot (s_i^L)^3 \\ &= \frac{A}{(U_L^{L\prime})^2} + \frac{B}{(m - U_L^{L\prime})^2}, \quad A = \left(\sum_{\Gamma_L} \mu_i^L {A_i}^{1/3} \right)^3, \\ B &= R^2 \times \left(\sum_{\Gamma_H} \mu_i^L A_i^{1/3} \right)^3. \end{aligned} \tag{14}$$

By mathematical differential computing, we find that P_d is further minimized when the following equation holds:

$$U_L^{L\prime} = \sum_{\Gamma_L} \frac{\mu_i}{{s_i}^L} = \frac{m}{1 + \sqrt[3]{B/A}}. \tag{15}$$

Combining Eq. (13), we can derive the optimal speeds assignment to minimize both temperature and energy in the LO mode. Under this optimal condition, the assignment of ω_i^L can be summarized as

$$\omega_i^L = \frac{R \cdot {\mu_i}^L}{{s_i}^L}, \quad \tau_i \in \Gamma_H; \quad \omega_i^L = \frac{{\mu_i}^L}{{s_i}^L}, \quad \tau_i \in \Gamma_L. \tag{16}$$

In the HI mode, since only HI tasks can be executed, their deadlines can be readily guaranteed by setting $\omega_i^H = \mu_i^H / s_i^H$. According to Theorem 1, dynamic power P_d is minimized when the speeds of HI tasks are assigned as

$$s_i^H = \frac{1}{m} {A_i}^{-1/3} \sum_{\Gamma_H} \mu_i^H {A_i}^{1/3}, \quad \tau_i \in \Gamma_H. \tag{17}$$

According to Lemma 1, condition (17) also results in the minimal processor temperature and power consumption. Since energy consumption in a hyperperiod is the integral of $P(t)$ upon time t, condition (17) also minimizes the energy consumption.

So far, the optimal conditions for both minimizing the temperature and energy consumption in the LO mode and HI mode, separately, have been found. However, the system usually executes in both two modes alternately, then how to consider them together is another problem to be addressed. Ref. 1 defines the weighted sum of the LO-mode power consumption and that of the HI-mode as the objective function, i.e., $E = \omega_{LO} E_{LO} + \omega_{HI} E_{HI}$, where $\omega_{LO} + \omega_{HI} = 1$, and the values of them can be set as required. In this work, since $U_H^{H\prime}$ is independent with $U_H^{L\prime}$ and $U_L^{L\prime}$, unlike that in VD-EDF,[1] the optimal conditions deducted above will not change under different settings of ω_{LO} and ω_{HI}. To sum up, we have the following theorem.

Theorem 2 (TA-MCF Optimal Condition). *With the DVFS technique, processor temperature and energy consumption can be minimized separately in the LO mode and HI mode when the speeds of tasks are assigned as Eq.* (13), (15) *and* (17), *and the execution rates are assigned as*

$$\omega_i^L = \begin{cases} R \cdot \dfrac{{\mu_i}^L}{{s_i}^L}, & \tau_i \in \Gamma_H, \\ \dfrac{{\mu_i}^L}{{s_i}^L}, & \tau_i \in \Gamma_L; \end{cases} \quad \omega_i^H = \begin{cases} \dfrac{{\mu_i}^H}{{s_i}^H}, & \tau_i \in \Gamma_H, \\ 0, & \tau_i \in \Gamma_L. \end{cases} \tag{18}$$

4.3. *TA-MCF algorithm*

The TA-MCF proposed in this work can be implemented based on the optimal speed and execution rate assignments according to Theorem 2. However, in spite of the advantages of fluid scheduling, it is not realistic to directly implement fluid-based scheduling on real systems due to the stringent assumption that one processor can execute multiple tasks at the same time. To tackle this challenge, we implement an approximate fluid scheduling. In addition, considering that the optimal task speed assignment proposed in Theorem 2 may cannot always be achieved due to reasons like time constraints of tasks (task speed should be no less than its utilization to meet its deadline) and processor speed limitation (some processors have a speed range, so the task speed has to satisfy this constraint), we also add a module to modify the task speeds shown in the optimal condition. The details of TA-MCF is shown in Algorithm 1.

Algorithm 1 can be divided into two main parts. Lines 1–5 constitate the first part, which aims to assign the optimal LO and HI speeds together with the LO and HI execution rates for tasks according to the deduction in Sec. 4.2. Line 1 calculates the optimal total utilization of LO tasks in the LO mode by Eq. (15), while lines 2 and 3 assign the optimal LO and HI speeds for tasks by Eqs. (13) and (17). Line 4 adjusts the optimal task speeds according to the constraints of task deadlines and the processor speed range. When the optimal task speed is lower than the minimal speed that meets its deadline, it will be adjusted as the minimal speed; when the optimal

Algorithm 1. TA-MCF (Thermal-Aware Mixed Criticality Fluid scheduling)

Input: A task set $\Gamma_n = \{\tau_1, \tau_2, \ldots, \tau_n\}$ with each task $\tau_\tau =< A_\tau, T_\tau, C_\tau^{\mathrm{L}}, C_\tau^{\mathrm{H}}, \chi_\tau >$, U_{tot} and thermal related parameters.

Output: The processor power and temperature values during the execution of the tasks.

1: Calculate the optimal $U_L^{L\prime}$ by Eq. (15), and set $U_L^{L\prime}$ as the maximal one of U_L^L and the optimal value;
2: Set the LO speeds of tasks as Eq. (13);
3: Set the HI speeds of tasks as Eq. (17);
4: speedModification(); //Modify the LO and HI speeds according to the range and schedulability constraints.
5: Set the LO and HI execution rates as Eq. (18);
6: **repeat**
7: $releaseTasks()$;
8: **repeat**
9: $runningTask = getNextTask()$;
10: $runTask()$; //run tasks for $\mathrm{IL} \cdot \omega_\tau(C_i/T_i s_i)$ time units;
11: $updateTemperature()$;
12: $updatePower()$;
13: **if** $runningTask.finish()$ **then**
14: $removeTask(runningTask)$;
15: **else**
16: $runningTask.setVisibility(False)$;
17: **end if**
18: **until** $\Delta t < \mathrm{IL}$
19: $resetTasks()$;//set all tasks visible;
20: **until** $k \cdot \mathrm{IL} < \mathrm{TI}$

task speed is larger than the maximal processor speed, say 1, it will be adjusted as the maximal processor speed. Once the speed of one task is adjusted, the speeds of the remained tasks also need to be adjusted so as to get the best sub-optimal solution under the constraint of schedulability condition. This procedure will be recursively executed until all of the tasks' speeds satisfy the constraints. Line 5 assigns the LO and HI execution rates according to Eq. (18).

Lines 6–20 constitute the second part, which aims to execute the tasks in an approximate way of fluid scheduling with a fixed scheduling length under the speeds and execution rates determined in the first part. Line 7 releases tasks according to their periods, while lines 8–18 execute the tasks and update the temperature and power values of the processors based on the models shown in Sec. 3. In detail, lines 9 and 10 select a task from the queue of tasks that have been released to execute with

the execution rate assigned; lines 11 and 12 update the temperature and power values of the processors due to the task execution; when the task finishes, it will be removed from the task queue (lines 13 and 14), otherwise, it will be suspended until the next scheduling interval (lines 15–17). Once all of tasks have been executed once in one scheduling interval, they will be set ready for the next scheduling interval (line 19). In this part, IL refers to the scheduling interval length, and TI is the time interval we set to observe the processor power and temperature. In the approximate fluid scheduling algorithm, the shorter the interval length is, the closer the approximate algorithm is to the fluid scheduling, while the longer the interval length is, the lower the overhead caused by task switching. When applied in practice, a trade-off has to be made between the two factors to determine an appropriate interval length. Further, to meet the timing constraints, the interval length is upper bounded to the greatest common divisor of the tasks' periods.

About the runtime complexity of the proposed algorithm, it should be evident that the TA-MCF shown in Algorithm 1 has runtime that is linear in the number of tasks in Γ. In one straightforward implementation strategy, the optimal value of $U_L^{L'}$ can be determined in one pass through the task system, the high and low speeds of tasks in a second pass and the high and low execution rates together with the test, to ensure that their values sum to no more than m in a third pass.

5. Experimental Evaluation

This section compares TA-MCF with the state-of-the-art strategies, MCF[10] and MC-Sort,[11] upon three metrics of schedulability ratio, energy consumption and processor temperature. The three groups of experiments validate that, compared with the MCF and MC-Sort, the TA-MCF proposed in this paper can better minimize the energy and temperature while ensuring a comparable schedulability ratio.

5.1. *Experiment setup*

Our experiments are carried out on a dual-criticality implicit-deadline task system upon an m-core platform. We adopt a task generation approach similar to the one used in Refs. 10 and 11. The task generation is controlled by the following parameters: (1) U_{tot}, the normalized system utilization bound, which is drawn from the range of [0.45, 1.0] with an increment length of 0.05; (2) m, the number of cores, which is set as 2, 4, 8 and 16, respectively; (3) $\mu_{\max}$, the maximum individual task utilization, which is set as 0.9 here; (4) T_i, the period of task τ_i, which is an integer drawn from the range of [100, 1000] with millisecond order; (5) A_i, the activity factor of task τ_i, which is drawn from the range of [0.1, 1]; (6) p^H, the probability of a task with high criticality level, which is set as 0.5 here; (7) R, the ratio of C_i^H to C_i^L for any HI task, which is set as 1.4 as that adopted in previous studies.[1,15,23] The parameters and their values shown above are extracted from the real applications,

and the task generation approach we adopt based on these parameters has been widely used by many recent studies like Refs. 9–11, with which our proposed approach will be compared. The task set with these parameters is generated through the following steps:

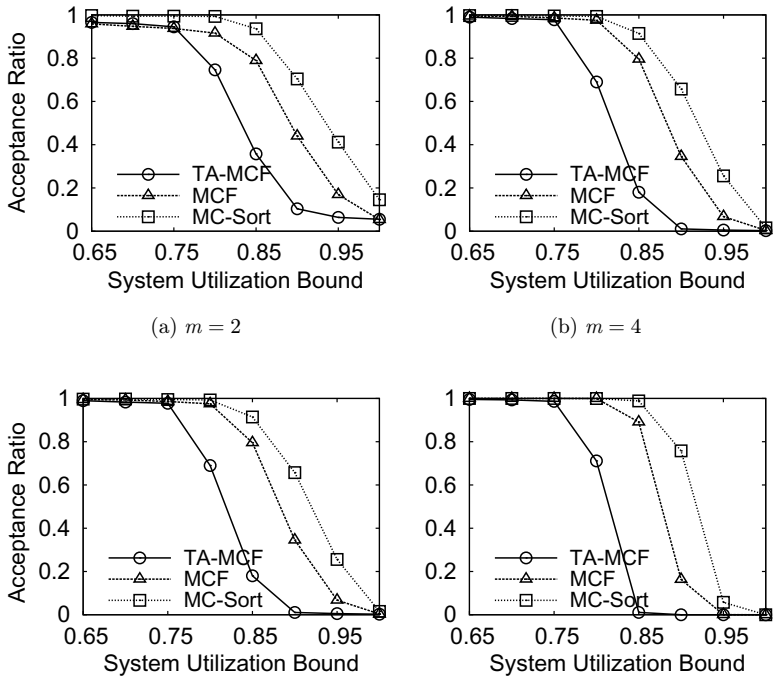
(1) Select T_i and A_i for task τ_i from the given ranges shown above.
(2) With given p^H, a real number p_i is randomly drawn from the range [0, 1]. If $p_i < p^H$, then $\chi_i = \text{HI}$. Otherwise, $\chi_i = \text{LO}$.
(3) Task utilization μ_i is drawn from the range $[0.02, \mu_{\max}]$. If $\chi_i = \text{LO}$, then $\mu_i^L = \mu_i$. Otherwise, $\mu_i^H = \mu_i$ and $\mu_i^L = \mu_i^H/R$. In both cases, C_i^L is set to $\lceil \mu_i^L \times T_i \rceil$ and C_i^H is set to $\lceil \mu_i^H \times T_i \rceil$.
(4) Repeat the above steps as long as $\max\{(U_L^L + U_H^L)/m, U_H^H/m\} \leqslant U_{\text{tot}}$. Once this condition is violated, discard the last generated task.
(5) If the resulting taskset satisfies the condition $\max\{(U_L^L + U_H^L)/m, U_H^H/m\} > U_{\text{tot}} - 0.05$, we accept the taskset and exit the procedure. Otherwise, discard the taskset and repeat the above steps.

In addition, the system thermal parameters of the processors are set as Refs. 21, 22: $R_{\text{th}} = 0.36$, $C_{\text{th}} = 0.8$, $\alpha = 0.001$, $\beta = 0.1$, $\Theta_a = 25°\text{C}$.

5.2. *Evaluation results*

To compare the schedulability ratios and energy consumptions of TA-MCF, MCF and MC-Sort, we generate 10,000 random tasksets at each system utilization point ($U_{\text{tot}} \in [0.45, 1]$ with a step length of 0.05 with $p^H = 0.5$. The effect of the number of cores on the two metrics is also considered, and m is, respectively, set to be 2, 4, 8, 16.

Figures 1(a)–1(d) show the schedulability ratios of the three strategies, which denotes the ratio between the number of schedulable task sets and that of the total generated task sets. The results show that: (i) TA-MCF has the lowest but comparable schedulability compared with MCF and MC-Sort; (ii) all three strategies perform well until the system utilization is beyond 0.8, in which situation the schedulability drops dramatically; (iii) the effect of the number of cores becomes obvious only when the system is busy enough. Figures 2(a)–2(d) show the energy consumptions of the three strategies which is normalized to the maximal one in each parameter setup, so as to provide a much more intuitive presentation of the comparative results. Since the system executes in both HI and LO modes, and the time length that the system stays in HI mode is different depending on the actual execution of tasks, here we model the total energy as the weighted sum of complete LO-mode energy and complete HI-mode energy (here the two weights are both set to 0.5, and they can be adjusted as required in practice). In addition, considering the sharply decreasing schedulability of all three strategies when U_{tot} is beyond 0.8, we only analyze the energy consumptions of them with U_{tot} in range [0.45, 0.8] and the

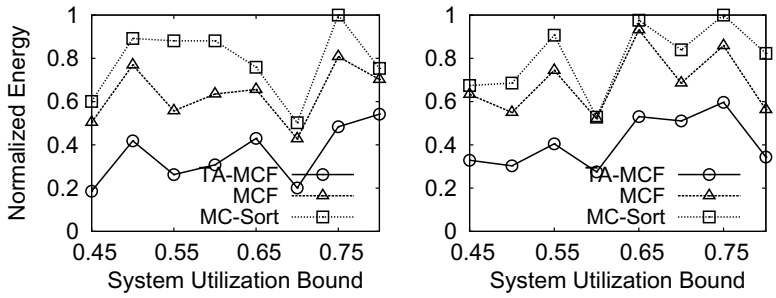


(a) $m = 2$ (b) $m = 4$

(c) $m = 8$ (d) $m = 16$

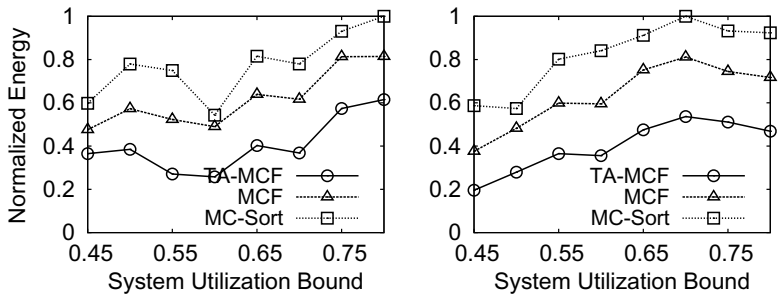
Fig. 1. Effect of utilization bound on acceptance ratio under different number of cores.

task set that is schedulable in all three strategies will be selected. The results show that although in terms of schedulability ratio, MC-Sort has the best behavior which is followed by MCF and TA-MCF, the order is reversed when it comes to the energy consumption. This means that, we cannot have both schedulability ratio and energy consumption optimized when exploring MCF strategy. A trade-off between them has to be cautiously made in practice.



(a) $m = 2$ (b) $m = 4$

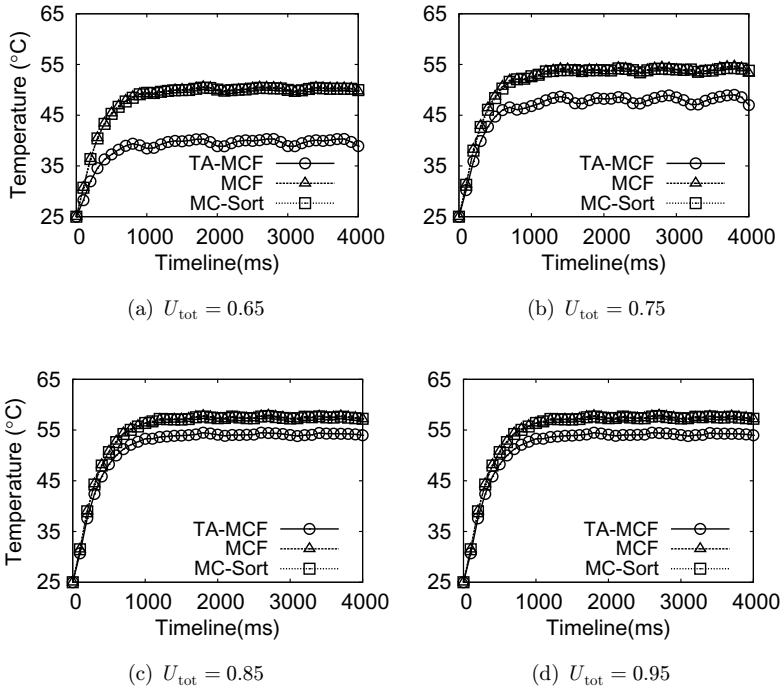
Fig. 2. Effect of utilization bound on system total energy under different number of cores.



(c) $m = 8$ (d) $m = 16$

Fig. 2. (*Continued*)

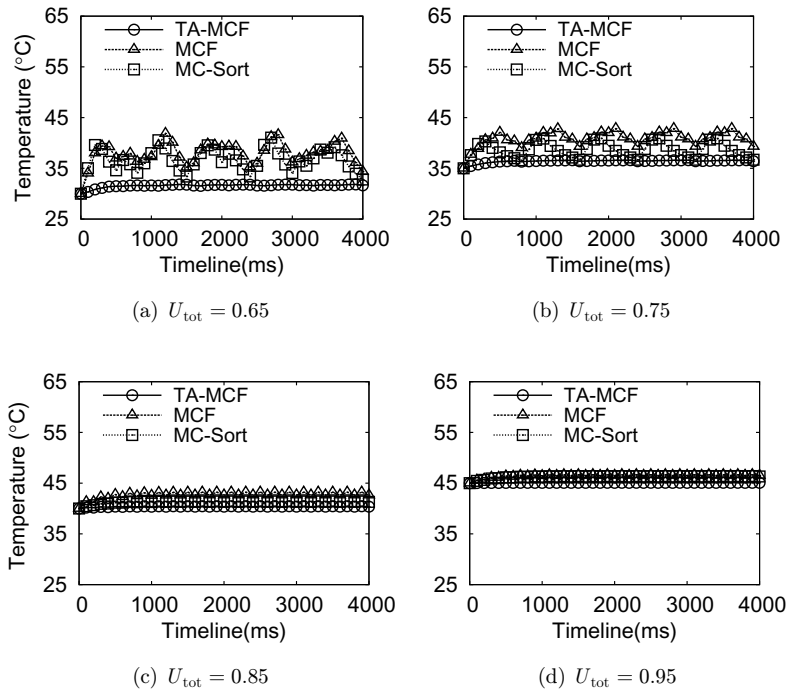
In terms of processor temperature, we cannot treat it like energy consumption since it is a nonaccumulative variable. Therefore, we only analyze the temperature separately in the LO mode and in the HI mode of one core, and the task set that is schedulable in all three strategies will be selected. Figures 3(a)–3(d) show the temperature in the LO mode. The results show that TA-MCF has the lowest temperature along with time, while MCF and MC-Sort have almost the same higher



(a) $U_{\text{tot}} = 0.65$ (b) $U_{\text{tot}} = 0.75$

(c) $U_{\text{tot}} = 0.85$ (d) $U_{\text{tot}} = 0.95$

Fig. 3. Effect of utilization bound on system temperature under LO mode.



(a) $U_{\text{tot}} = 0.65$ (b) $U_{\text{tot}} = 0.75$

(c) $U_{\text{tot}} = 0.85$ (d) $U_{\text{tot}} = 0.95$

Fig. 4. Effect of utilization bound on system temperature under HI mode.

temperature in all parameter setups. Meanwhile, with the increasing of U_{tot}, the gap between TA-MCF and the others becomes narrow, and this is because the system load is too high for us to conduct any optimization. Figures 4(a)–4(d) show the temperature in HI mode. It demonstrates a similar behavior with that of the LO mode, except that the temperature curves of MCF and MC-Sort in Figs. 4(a) and 4(b) exhibit an obvious oscillation feature, which is caused by the following reasons: (1) U_{tot} refers to the system total utilization, while in HI mode only HI tasks are executed, which will make the system under-loaded; (2) MCF and MC-Sort do not conduct speed scaling to stretch the system utilization. Actually, HI mode scheduling here is equal to the nonMC system scheduling whose optimal condition has been shown in Theorem 1.

Besides, for a multi-core system with a complex thermal model considering the thermal effect among multiple processor elements, the first thing to do is to determine the task-to-core assignment which is beyond the scope of this paper. Nevertheless, for a given task-to-core assignment, the presented TA-MCF can be applied on each core to minimize the temperature of the core. When the temperature of each core is minimized compared with the MCF and MC-Sort, the thermal flow among different cores and heat sinks can also be minimized. Hence, the whole system's

temperature will be decreased. In fact, by TA-MCF, all of the tasks will be assigned speeds that can result in the same dynamic power consumption. In this situation, even in multi-core system, each core will have the similar temperature curve as long as we average the utilization of each core through appropriate task-to-core mapping approach. On the contrary, MCF and MC-Sort do not consider the thermal factor, so the temperature of each core may have a great differences, thus causing vast thermal flow among different processor elements and resulting in a relatively higher peak temperature. That is to say, even for the same task-to-core mapping, TA-MCF results in a lower temperature compared with the MCF and MC-Sort, let alone that the thermal-aware feature of TA-MCF can further average the temperature of different cores through averaging the utilization of each core when assigning tasks to cores. We will further study this part in our near future works in detail.

6. Conclusions

This work proposes a thermal and energy aware fluid scheduling strategy for MCS on multi-processors, which minimizes both the energy consumption and temperature while providing comparable schedulability ratio compared with the state-of-the-art approaches MCF and MC-Sort. The optimal condition of TA-MCF, determining the task speed and execution rate assignments, is deduced. Further, considering the unrealistic assumption made by fluid scheduling that one processor can execute multiple tasks at the same time, an approximate TA-MCF algorithm is implemented. Extensive experiments validate the efficiency of the proposed algorithm.

In theory, the proposed technique can be extended to the generalized multi-criticality systems. We only need to conduct the schedulability analysis and the optimal speed assignment in each criticality mode just like the low and high modes we analyze in this work.

References

1. S. Narayana, P. Huang, G. Giannopoulou, L. Thiele and R. V. Prasad, Exploring energy saving for mixed-criticality systems on multi-cores, *IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS)*, Vienna, Austria, April 11–14, 2016, pp. 135–146.
2. S. Vestal, Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance, *Proc. 28th IEEE Real-Time Systems Symposium (RTSS 2007)* Tucson, Arizona, USA, 2007, pp. 239–243.
3. A. Burns and R. I. Davis, Mixed criticality systems — A review, Computing Science Technical Report (University of York, UK, 2017).
4. H. Huang, C. D. Gill and C. Lu, Implementation and evaluation of mixed-criticality scheduling approaches for sporadic tasks, *ACM Trans. Embedded Comput. Syst.* **13** (2014) 126:1–126:25.

AQ: Kindly provide location details for symposium.

5. S. K. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. van der Ster and L. Stougie, Mixed-criticality scheduling of sporadic task systems, *Proc. 19th Annual European Symp.* Saarbrücken Germany, 2011, pp. 555–566.
6. T. Zhang, N. Guan, Q. Deng and W. Yi, On the analysis of edf-vd scheduled mixed-criticality real-time systems, *IEEE International Symp. Industrial Embedded Systems*, 2014, pp. 179–188.
7. S. Baruah, Schedulability analysis for a general model of mixed-criticality recurrent real-time tasks, *2016 IEEE Real-Time Systems Symp. RTSS 2016*, Porto, Portugal, November 29–December 2, 2016, pp. 25–34.
8. D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov and W. Yi, EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees, *2016 IEEE Real-Time Systems Symp. RTSS 2016*, Porto, Portugal, November 29–December 2, 2016, pp. 35–46.
9. J. Lee, K. Phan, X. Gu, J. Lee, A. Easwaran, I. Shin and I. Lee, Mc-fluid: Fluid model-based mixed-criticality scheduling on multiprocessors, *Proc. IEEE 35th IEEE Real-Time Systems Symp., RTSS 2014*, Rome, Italy, December 2–5, 2014, pp. 41–52.
10. S. K. Baruah, A. Easwaran and Z. Guo, Mc-fluid: Simplified and optimally quantified, *2015 IEEE Real-Time Systems Symp., RTSS 2015*, San Antonio, Texas, USA, December 1–4, 2015, pp. 327–337.
11. S. Ramanathan and A. Easwaran, Mc-fluid: Rate assignment strategies, *Proc. 3rd Workshop on Mixed Criticality Systems (WMC)*, San Antonio, Texas, USA, 1st December 2015, pp. 6–11.
12. V. Legout, M. Jan and L. Pautet, Mixed-criticality multiprocessor real-time systems: Energy consumption vs deadline misses, *Proc. First Workshop on Real-Time Mixed Criticality Systems (ReTiMiCS)*, Taipei, Taiwan, August 2013, pp. 1–6.
13. Z. Li, X. Hua, C. Guo and S. Ren, Empirical study of energy minimization issues for mixed-criticality systems with reliability constraint, *Proc. 1st Workshop on Low-Power Dependable Computing (LPDC)*, Dallas, TX USA, November 3–5, 2014, pp. 1–6.
14. M. Völp, M. Hähnel and A. Lackorzynski, Has energy surpassed timeliness? Scheduling energy-constrained mixed-criticality systems, *20th IEEE Real-Time and Embedded Technology and Applications Symp., RTAS 2014*, Berlin, Germany, April 15–17, 2014, pp. 275–284.
15. P. Huang, P. Kumar, G. Giannopoulou and L. Thiele, Energy efficient DVFS scheduling for mixed-criticality systems, *2014 Int. Conf. Embedded Software, EMSOFT 2014*, New Delhi, India, October 12–17, 2014, pp. 11:1–11:10.
16. B. Hu, K. Huang, G. Chen, L. Cheng, D. Han and A. Knoll, Schedulability analysis towards arbitrarily activated tasks in mixed-criticality systems, *J. Circuits Syst. Comput.* **26** (2017) 1–31.
17. J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan and Y. Ma, Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms, *J. Syst. Softw.* **133** (2017) 1–16.
18. J. Zhou, M. Yin, Z. Li, K. Cao, J. Yan, T. Wei, M. Chen and X. Fu, Fault-tolerant task scheduling for mixed-criticality real-time systems, *J. Circuits Syst. Comput.* **26** (2017) 1–17.
19. L. Niu and W. Li, Energy-efficient scheduling for embedded real-time systems using threshold work-demand analysis, *J. Circuits Syst. Comput.* **26** (2017) 1–36.
20. J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu and Y. Ma, Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems, *IEEE Trans. Comput. -Aided Des. Integr. Circuits Syst.* **35** (2016) 1269–1282.

21. R. Ahmed, P. Ramanathan and K. K. Saluja, On thermal utilization of periodic task sets in uni-processor systems, *2013 IEEE 19th Int. Conf. Embedded and Real-Time Computing Systems and Applications, RTCSA 2013*, Taipei, Taiwan, August 19–21, 2013, pp. 267–276.
22. R. Ahmed, P. Ramanathan and K. K. Saluja, Necessary and sufficient conditions for thermal schedulability of periodic real-time tasks under fluid scheduling model, *ACM Trans. Embedded Comput. Syst.* **15** (2016) 49:1–49:26.
23. P. Huang, G. Giannopoulou, N. Stoimenov and L. Thiele, Service adaptions for mixed-criticality systems, *Proc. of the 19th Asia and South Pacific Design Automation Conf. ASP-DAC 2014* (Singapore, 2014), pp. 125–130.
24. S. K. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster and L. Stougie, The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems, *24th Euromicro Conf. Real-Time Systems, ECRTS 2012*, Pisa, Italy, July 11–13, 2012, pp. 145–154.
25. W. Liao, L. He and K. M. Lepak, Temperature and supply voltage aware performance and power modeling at microarchitecture level, *IEEE Trans. Comput. -Aided Des. Integr. Circuits Syst.* **24** (2005) 1042–1053.
26. Y. Liu, R. P. Dick, L. Shang and H. Yang, Accurate temperature-dependent integrated circuit leakage power estimation is easy, *2007 Design, Automation and Test in Europe Conf. and Exposition, DATE 2007*, Nice, France, April 16–20, 2007, pp. 1526–1531.
27. K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy and D. Tarjan, Temperature-aware microarchitecture: Modeling and implementation, *ACM Trans. Archit. Code Option.* **1** (2004) 94–125.