Journal of Circuits, Systems, and Computers
Vol. 28, No. 6 (2019) 1950102 (20 pages)

DOI: 10.1142/S0218126619501020

Peak Temperature Minimization for Hard Real-Time Systems Using DVS and DPM*

Mingchuan Zhou[†,§], Long Cheng[‡,¶], Manuel Dell'Antonio[†,‖], Xiebing Wang[†,**], Zhenshan Bing[†,††], M. Ali Nasseri[†,‡‡], Kai Huang[‡,§§] and Alois Knoll[†,¶¶]

[†]*Chair of Robotics, Artificial Intelligence and Real-Time Systems, Department of Computer Science, Technische Universität München, Boltzmannstraße 3, 85748 Garching bei München, Germany*

[‡]*School of Data and Computer Science, Sun Yat-Sen University, 135 Xingang Xi Road, Guangzhou 510275, P. R. China*
[§]*zhoum@in.tum.de*
[¶]*chenglong3@mail.sysu.edu.cn*
[‖]*dellanto@in.tum.de*
[**]*wangx@in.tum.de*
[††]*bing@in.tum.de*
[‡‡]*nasseri@in.tum.de*
[§§]*huangk36@mail.sysu.edu.cn*
[¶¶]*knoll@in.tum.de*

Received 11 March 2018
Accepted 9 July 2018
Published 23 August 2018

With the increasing power densities, managing the on-chip temperature has become an important design challenge, especially for hard real-time systems. This paper addresses the problem of minimizing the peak temperature under hard real-time constraints using a combination of dynamic voltage scaling and dynamic power management. We derive a closed-form formulation for the peak temperature and provide a genetic-algorithm-based approach to solve the problem. Our approach is evaluated with both simulations and real measurements with an Intel i5 processor. The evaluation results demonstrate the effectiveness of the proposed approach compared to related works in the literature.

Keywords: Peak temperature; hard real-time system; DVS–DPM.

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

1. Introduction

A hard real-time system is commonly used in critically safety-associated applications, such as automatic driving systems and artificial hearts. It is a crucial priority to ensure that the system latency is bounded by a specific deadline constraint.[1–3] To achieve this goal, real-time systems require processors to work in a stable and predictable manner. However, in recent years, the semiconductor technology continues to scale down the package size and increase on-chip power density, leading to a rapid increase of the peak temperature. The high temperature seriously threatens the reliability and performance of the systems,[4] where a 10–15°C difference in operating temperature can result in a two-fold difference in the life span of a system.[4] High temperatures are responsible for transient faults caused by timing errors since every 10°C temperature increase can cause about 5% interconnect delay.[5] In addition, elevated temperatures directly impact electromigration and hence reduce the mean time to failure (MTTF) of the chip.[6] As a consequence, temperature management has become a key system design issue.

In order to control the on-chip temperature, two approaches are normally used, i.e., hardware cooling devices and dynamic thermal management (DTM).[7] Cooling devices are a costly method with 1–3 $/W.[8] DTM reduces the temperature by lowering the performance and thus incurs less cost, thus is more attractive for price-sensitive applications. In principle, DTM has two mechanisms, dynamic power management (DPM) and dynamic voltage scaling (DVS). DPM is designed to switch the processor between sleep and active states.[9,10] DVS controls the temperature by dynamically scaling the supply voltage.[11,12] The disadvantage of DVS is the lack of means to reduce static power consumption, while DPM is limited by the relatively high switching overheads and single active speed. Therefore, the combination of both can compensate for each other and is more preferable for real-life applications.

The DVS–DPM combined approach has already showed superiorities for power management in real-time systems.[13–16] However, those methods for power management cannot be directly applied for the temperature optimization, due to the fact that although temperature is a strong function of power, power management techniques that are effective for energy saving may not be suitable for temperature management, which has already been theoretically proved in Ref. 7. Furthermore, as a DVS–DPM combination scheme will provide multiple power modes, how to formulate the peak temperature of such multi-mode scenario is nontrivial. Without a proper representation of the peak temperature, the late-on minimization will be even more complicated.

In this paper, we explore how to derive a DVS–DPM scheme to minimize the peak temperature for general arrival workloads while the worst-case deadline of the system is guaranteed. We adopt periodic thermal management (PTM) and within a hyper-period the processor can operate on arbitrary numbers of power

modes. For the given processor model and input workloads, we derive a multi-mode hyper-period to control the processor. By transiting the processor into the derived power modes accordingly within a hyper-period, the peak temperature of the system is managed. We employed real-time calculus (RTC)[17] to model the irregular job arrivals in time interval domain. The constraints are formulated to guarantee that the system provides such service that all workloads complete within their deadlines. Although we focus on single core in this paper, the proposed method can be applied to multiple cores in principle. The detailed contributions of the paper are as follows:

- We derive a mathematical representation to calculate the peak temperature for our DVS–DPM scheme, which can cope with arbitrary numbers of power modes. The correctness of this representation is formally proved.
- A genetic-algorithm-based approach is designed to tackle the peak temperature minimization problem.
- We evaluated our approach with both simulations and real measurements with an Intel i5 processor. For the measurements, we extended an open-source thermal prototyping framework[18] to integrate our DVS–DPM scheme and conducted real measurements on a laptop equipped with an Intel Core i5-4210U CPU.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 3 represents the system model and problem statement. In Sec. 4, we analyze the peak temperature and derive the mathematical formulation. Section 5 presents an algorithm to find a DVS–DPM scheme for the peak temperature minimization. Several cases are studied in Secs. 6 and 7 concludes this paper.

2. Related Works

Temperature control has been insensitively investigated recently. These researches can be basically divided into two categories, online and offline methods. The online temperature control method is always defined as an online feedback control problem where the temperature is considered as the constraint maximizing the computing ability. Quan and Zhang[19] studied the feasibility issue for a hard real-time periodic event set. The authors focused on the peak temperature checking within the safety bound, but without providing scheduling policies. What is more, the thermal optimization is not taken into account when only checking the peak temperature in the safety bound. Jayaseelan and Mitra[20] proposed an event sequencing mechanism with DVS to minimize the peak temperature for periodic events. Some researchers are concerned about the problem of peak temperature minimization in the real-time system. Huang *et al.*[21] introduced an M-oscillating real-time scheduling algorithm based on DVS to minimize the peak temperature, by considering the leakage/temperature dependency as well as transition overhead for the period event model.

By adopting an online two-power DVS schedule method, they reported a better effectiveness than the reactive two-power method. An approach called cool shaper (CS) was studied in Ref. 8 to minimize the peak temperature by dynamically and selectively inserting idle time during the execution of hard real-time jobs. This approach is offline/online-combined, i.e., the parameters of the shaper are offline computed and the workload is runtime gathered with pre-computed shaper. Zhou *et al.*[22] developed an algorithm to optimize the peak temperature in heterogeneous real-time system with two modes while the ideal and real computation overheads are considered for the multi-processors. A further work was performed by Zhou *et al.*[23] with energy optimization. The benefit of online method is that sometimes it could take the real temperature as feedback to adjust the model. However, the online adjustment will also introduce a certain amount of running overhead. Some of the proposed methods switched the processor only with two-power modes which has a limitation that two speeds normally would not lead to optimal results of the system. Furthermore, most of the researches here model the coming events into the periodic event set with the missing of nondeterminism of the event arrivals which actually is not matching with the real situation.

Some other works minimize the peak temperature using offline methods which consequently introduce much less running overhead. Schor *et al.*[24] proposed an analytic method to offline calculate the worst-case peak temperature of a real-time application with a nondeterministic workload that is running on a multi-core system, where the worst-case peak temperature can be guaranteed by a DTM scheme algorithm. Ahmed *et al.*[25] presented an offline algorithm for sporadic tasks to minimize the peak temperature in embedded real-time systems by utilizing thermal-aware periodic resources. Cheng *et al.*[9] developed an offline DTM scheme based on the arrival curve model. In a later work, they developed a two-mode scheme for multi-core peak temperature optimization with pipelined hard real-time systems.[26] Compared to the online method, the offline has the benefit of no online adjustment overhead. The drawback would be that it does not include the online feedback which makes it critical to correctly estimate the system thermal model and coming event pattern. The methods mentioned above in the offline method either consider two-power modes scenario or a simple event without nondeterminism characteristic, which cannot provide an optimal active speed to meet the requirements of dealing with the coming nondeterministic events and controlling the peak temperature simultaneously. The nondeterminism workload brings the difficulty to model the coming event and the multi-power modes bring the challenges of optimization for adding another dimension.[27] In order to cope with these challenges, in what follows, we develop an offline DVS–DPM scheduling approach that is suitable for arbitrary numbers of power modes and the arrival curve is introduced to model general task arrivals, thus to reduce the peak temperature more efficiently. The scheduling approach is also verified on the real processor to prove the small gap between simulations and real measurements.

3. System Model and Problem Statement

3.1. *Thermal model*

The processor has a discrete set of running speeds $S = \{s_0, s_1, \ldots, s_n\}$, each of which has a corresponding supply voltage $V = \{v_0, v_1, \ldots, v_n\}$, where n is the number of nonzero speed levels. The supply voltage value in V is ranked from low to high as $v_0 < v_1 < \cdots < v_n$. Here, two states of operation are identified; (1) *sleep state*: the processor cannot handle the coming job, and has the lowest power consumption with a supply voltage v_0; (2) *active state*: the processor executes some events in the system at speed $s_k \in S$, $k \in \{1, \ldots, n\}$, with the corresponding supply voltage $v_m \in V$, $m \in \{1, \ldots, n\}$. We use t_{sw} to denote all kinds of switch overhead, as shown in Fig. 1:

$$t_{\text{sw}} = \begin{cases} t_{\text{swon}} & \text{from sleep to active state}, \\ t_{\text{swoff}} & \text{from active to sleep state}, \\ t_{\text{swact}} & \text{two active states}. \end{cases} \tag{1}$$

During the switch between two modes, the processor cannot tackle any coming event, but still has a certain power consumption. For t_{swoff} and t_{swon}, the processor has power consumption of the coming active mode, while for t_{swact} it has power consumption of the next mode, see Fig. 1. The temperature of the processor is described based on the well-known RC thermal model in Refs. 21 and 28. Let the temperature at the initial time be $T(0)$. During the time interval $[t_0, t]$, if the processor remains in the same mode with voltage v_m, the temperature at time point t can be obtained through solving the RC thermal differential as follows[28]:

$$T(t) = T_m^{\infty} + (T(t_0) - T_m^{\infty}) e^{-B_m(t-t_0)}, \tag{2}$$

where $T_m^{\infty} = \frac{A_m}{B_m}$, with A_m and B_m being the processor-specific constants corresponding to each running speed. If the system keeps one mode with $t \to \infty$, $T(t) = T_m^{\infty}$. We call T_m^{∞} the steady temperature. Moreover, the thermal model is

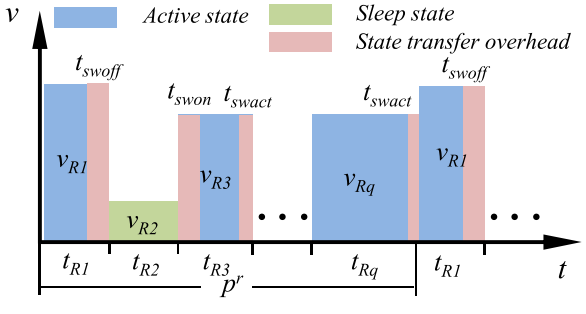


Fig. 1. DVS–DPM scheme with q intervals.

regulated under the following conditions:

- The sleep steady temperature is defined as T_0^∞, and we define the initial temperature $T(0) = T_0^\infty$.
- Also $T_m^\infty > 0$, $A_m > 0$ and $B_m > 0$.

3.2. *Event model*

The arrival curve model,[17] which can preserve more information like the nondeterminism of the event arrivals, is used to model the task. The arrive curve is bounded by the lower arrival curve $\overline{\alpha}^l(\Delta)$ and the upper arrival curve $\overline{\alpha}^u(\Delta)$, where Δ is the length of an arbitrary time interval. $\overline{\alpha}^l(\Delta)$ and $\overline{\alpha}^u(\Delta)$ can be calculated as

$$\overline{\alpha}^l(\Delta) \leq R(t) - R(t') \leq \overline{\alpha}^u(\Delta)\,, \quad \Delta = (t - t'),\ t - t' \geq 0\,, \tag{3}$$

where $R(t)$ is the cumulative workload, which represents the number of events arriving at the processor in time interval $[0, t]$. With the concept of arrival curve, we can unify many other common timing models of event stream. For an event stream which can be specified by the (p, j, d) model,[8] where p denotes the period, j denotes jitter and d denotes minimal inter-arrival distance, the upper arrival curve $\overline{\alpha}^u(\Delta) = \min\{\lfloor\frac{\Delta+j}{p}\rfloor, \lfloor\frac{\Delta}{d}\rfloor\}$ is modeled by RTC Toolbox developed by Thiele *et al.*[17] Similar to the arrival curve, the service curve also provides the upper and lower bounds of cumulative function $C(t)$, which is the total time for the processor to handle the coming events in the time interval $[0, t]$. The worst-case execution time of one event arrival stream is defined as c, thus the event-based arrival curve can be transferred into time-based arrival curve, which can be described as $\alpha^l(\Delta) = c \cdot \overline{\alpha}^l(\Delta)$ and $\alpha^u(\Delta) = c \cdot \overline{\alpha}^u(\Delta)$. For multi-event scenarios, assume N as the number of tasks, D_i and $\alpha_i^u(\Delta)$ as the deadline and upper arrival curve for one event, respectively. With the schedule policy of earliest deadline first (EDF), the lower service bound $\beta_B(\Delta)$ is

$$\beta_B(\Delta) = \sum_{i=1}^{N} \alpha_i^u(\Delta - D_i)\,. \tag{4}$$

3.3. *Problem statement*

The purpose of this paper is to design an approach to minimize the peak temperature by using DVS–DPM scheme. As shown in Fig. 1, DVS–DPM scheme adopts a sequence scheme with a total number of q intervals, and each interval has a pair of (v_{Ri}, t_{Ri}), where v_{Ri} is the interval mode randomly selected from V, $Ri \in \{0, \ldots, n\}$, and t_{Ri} is the corresponding *interval running time*. The processor will keep or change the mode based on the order in the sequence periodically which is the so-called

PTM. We define this kind of scheme as PTM(q),

$$\mathrm{PTM}(q) = \{(v_{R1}, t_{R1}), \ldots, (v_{Ri}, t_{Ri}), \ldots, (v_{Rq}, t_{Rq})\}\,. \tag{5}$$

Without loss of generality, we define that any two adjacent modes differ from each other, if there exists more than one mode in PTM(q). The running period of PTM(q), p^r, is calculated as $\sum_{i=1}^{q} t_{Ri}$. Assuming s_{Ri} as the corresponding processor running speed with v_{Ri}, the cumulative function of service ability $C^{\mathrm{PTM}(q)}(t)$ generated by PTM(q) schedule is

$$C^{\mathrm{PTM}(q)}(t) = \begin{cases} 0\,, & t \le t_{\mathrm{sw}}\,, \\ s_{R1}(t - t_{\mathrm{sw}})\,, & t_{\mathrm{sw}} < t \le t_{R1}\,, \\ \cdots & \\ s_{Rq}\left(t - \sum_{i=1}^{q-1} t_{Ri}\right) + \sum_{i=1}^{q-1} s_{Ri}(t_{Ri} - t_{\mathrm{sw}})\,, & \\ \sum_{i=1}^{q-1} t_{Ri} < t \le p^r\,. & \end{cases} \tag{6}$$

Since PTM(q) is executed periodically, $C^{\mathrm{PTM}(q)}(t)$ is also accumulated periodically. The lower service curve $\beta_l^{\mathrm{PTM}(q)}(\Delta)$ under PTM(q) schedule in time interval Δ is

$$\beta_l^{\mathrm{PTM}(q)}(\Delta) = \inf\{C^{\mathrm{PTM}(q)}(t) - C^{\mathrm{PTM}(q)}(t')\}\,, \quad t - t' \ge 0\,. \tag{7}$$

By designating the peak temperature under PTM(q) as $T_{\mathrm{peak}}^{\mathrm{PTM}(q)}$, our problem is stated as follows:

Given a system characterized by the thermal model and event model described above, our goal is to derive a DVS–DPM scheme PTM(q) by which $T_{\mathrm{peak}}^{\mathrm{PTM}(q)}$ is minimal and PTM(q) can handle all events within their deadlines, i.e., the lower service curve $\beta_l^{\mathrm{PTM}(q)}$ satisfies the service bound of event streams $\beta_B(\Delta)$:

$$\min\, T_{\mathrm{peak}}^{\mathrm{PTM}(q)} \quad \text{s.t.} \quad \beta_l^{\mathrm{PTM}(q)}(\Delta) \ge \beta_B(\Delta)\,. \tag{8}$$

4. Peak Temperature Analysis

Theoretically, temperature of the processor at any time instant can be derived by Eq. (2), with the thermal model and PTM(q) schedule. However, it is time-consuming to evolve a convincing peak temperature. In this section, we present a mathematical representation of the peak temperature, which is a function of parameters from the thermal model and PTM(q) schedule. First, we present some properties about the trend of temperature evolution under PTM(q) schedule.

Lemma 1. *The temperature at the end of the first period under* PTM(q) *schedule*, T_q^1, *is higher than the initial temperature* $T(0)$, *i.e.*,

$$T_q^1 > T(0)\,.$$

Proof. We prove this lemma by discussing any PTM(q) schedule which is included in the following three situations:

- The PTM(q) only has sleep mode inside. In this situation, the processor cannot handle any coming event and the temperature will remain the initial temperature $T(0)$.
- There is only one active mode $v_{Ri} > v_0$ inside. Let $K_i = e^{B_{Ri}t_{Ri}}$, $T_i^\infty = \frac{A_{Ri}}{B_{Ri}}$. At the end of this *mode running time interval*, based on Eq. (2), we have $T_i^1 = T_i^\infty + \frac{(T(0)-T_i^\infty)}{K_i}$. Since $K_i > 1$ and $T(0) = T_0^\infty < T_i^\infty$, we can obtain

$$\begin{aligned} T_i^1 - T(0) &= T_i^\infty + \frac{(T(0) - T_i^\infty)}{K_i} - T(0) \\ &= (1 - K_i^{-1})T_i^\infty - (1 - K_i^{-1})T(0)\,, \end{aligned} \tag{9}$$

  where $i = q$ or $i = q - 1$. When $i = q$, we have $T_q^1 > T(0)$ from Eq. (9). When $i = q - 1$, we have $T_q^1 = T_0^\infty + (T_i^1 - T_0^\infty)e^{-B_0 t_{Rq}} > T_0^\infty = T(0)$.
- There are more than one active mode inside. Similar to the second situation, we can obtain $T_q^1 > T(0)$.

Since PTM(q) is generated to handle events, the first situation is not considered in our design. Therefore, in this paper only the second and third situations are discussed, thus we have $T_q^1 > T(0)$. The lemma is proved. □

The tendency of temperature under PTM(q) schedule is analyzed to determine when the peak temperature will be reached. From Ref. 19, we can obtain,

$$T_i^n = T_i^1 + \frac{(T_q^1 - T(0))(1 - U^{-(n-1)})}{1 - U^{-1}} U_i^{-1}\,, \tag{10}$$

where $U_i = \prod_{j=1}^{i} K_j > 1$, $U = \prod_{j=1}^{q} K_j > 1$, $K_j = e^{B_{Rj}t_{Rj}}$ and T_i^n is the temperature at the end of ith *interval running time* in the nth period under PTM(q) scheme.

Proof. From Eq. (10), T_i^n is a nondecreasing function of n. When $n \to \infty$, T_i^n has a certain value. □

Now based on the discussion above, we present the first main result of our work as the below theorem.

Theorem 1. *Denote* T_i^{st} *as the steady temperature of* T_i^n *under* PTM(q) *schedule. The peak temperature of the system* $T_{\text{peak}}^{\text{PTM}(q)}$ *can be presented as follows*:

$$T_{\text{peak}}^{\text{PTM}(q)} = \max\{T_1^{\text{st}}, \ldots, T_i^{\text{st}}, \ldots, T_q^{\text{st}}\}\,. \tag{11}$$

Proof. From the thermal model, we can easily prove that temperature of the system is a monotonic function of time t in any constant speed mode, i.e., between mode switches. Therefore, the peak temperature must be the maximal temperature when mode switches occur. As previously proved, T_i^n is increasing with n from $T(0)$ and finally reaches a state-state steady value. Therefore, it is clear that Theorem 1 is proved. □

In Theorem 1, computing T_i^{st} is not straightforward. We present the second main result of our work in Theorem 2, a closed-form representation for T_i^{st}. Based on Theorem 1, we assume that the temperature of the processor is already in the steady state. Therefore, we can get the following equation about the steady temperature at the end of each period based on Eq. (2):

$$\begin{cases} T_1^{\text{st}} = T_1^{\infty} + ((T_q^{\text{st}}) - T_1^{\infty})K_1^{-1}\,, \\ \cdots \\ T_q^{\text{st}} = T_q^{\infty} + ((T_{q-1}^{\text{st}}) - T_q^{\infty})K_q^{-1}\,. \end{cases} \tag{12}$$

We solve these equations from $q = 2, 3, \ldots$ and discover the pattern of solver, which is presented in following theorem.

Theorem 2. *With* PTM(q) *schedule, the steady temperature* T_i^{st} *can be calculated as*

$$T_i^{\text{st}} = \left(T_i^{\infty} - \sum_{j=0}^{q-2} \frac{T_{i-j}^{\infty} - T_{i-j-1}^{\infty}}{\prod_{k=i-j}^{i} K_k} - \frac{T_{i+1}^{\infty}}{U} \right) \frac{U}{U-1}\,. \tag{13}$$

Proof. Assuming the ith case as true, we prove the $(i+1)$th case by contradiction. For brevity, let $f(i,q)$ denote $(T_i^{\infty} - \sum_{j=0}^{q-2} \frac{T_{i-j}^{\infty} - T_{i-j-1}^{\infty}}{\prod_{k=i-j}^{i} K_k} - \frac{T_{i+1}^{\infty}}{U}) \frac{U}{U-1}$. Assume that the steady-state temperature T_i^{st} is

$$T_i^{st} = f(i,q) + \epsilon\,, \tag{14}$$

where ϵ is a real number, thus

$$\begin{aligned} T_{i+1}^{\text{st}} &= T_{i+1}^{\infty} + (T_i^{\text{st}} - T_{i+1}^{\infty})K_{i+1}^{-1} \\ &= (1 - K_{i+1}^{-1})T_{i+1}^{\infty} + T_i^{\text{st}} K_{i+1}^{-1} \\ &= (1 - K_{i+1}^{-1})\left(1 - \frac{1}{U}\right)T_{i+1}^{\infty} K_{i+1}^{-1} \\ &\quad + \left(T_i^{\infty} K_{i+1}^{-1} - \sum_{j=0}^{q-2} \frac{T_{i-j}^{\infty} - T_{i-j-1}^{\infty}}{K_{i+1} \prod_{k=i-j}^{i} K_k} - \frac{T_{i+1}^{\infty}}{K_{i+1} U} \right) \frac{U}{U-1} + \epsilon K_{i+1}^{-1} \end{aligned}$$

$$= \left(T_{i+1}^{\infty} - \frac{T_{i+1}^{\infty}}{U} - T_{i+1}^{\infty} K_{i+1}^{-1} + \frac{T_{i+1}^{\infty}}{K_{i+1} U} + T_i^{\infty} K_{i+1}^{-1} \right.$$

$$\left. - \sum_{j=0}^{q-2} \frac{T_{i-j}^{\infty} - T_{i-j-1}^{\infty}}{K_{i+1} \prod_{k=i-j}^{i} K_k} - \frac{T_{i+1}^{\infty}}{K_{i+1} U} \right) \frac{U}{U-1} + \epsilon K_{i+1}^{-1}$$

$$= \left(T_{i+1}^{\infty} - \sum_{j=0}^{q-2} \frac{T_{i-j+1}^{\infty} - T_{i-j}^{\infty}}{\prod_{k=i-j+1}^{i+1} K_k} - \frac{T_{i-q+2}^{\infty} - T_{i-q+1}^{\infty}}{U} - \frac{T_{i+1}^{\infty}}{U} \right) \frac{U}{U-1} + \epsilon K_{i+1}^{-1} .$$

Since T_i^n evolves into a steady-state value, we obtain that $T_{i-q+1}^{\infty} = T_{i+1}^{\infty}$ and $T_{i-q+2}^{\infty} = T_{i+2}^{\infty}$. Hence, we have the following equation:

$$T_{i+1}^{\mathrm{st}} = \left(T_{i+1}^{\infty} - \sum_{j=0}^{q-2} \frac{T_{i-j+1}^{\infty} - T_{i-j}^{\infty}}{\prod_{k=i-j+1}^{i+1} K_k} - \frac{T_{i+2}^{\infty}}{U} \right) \frac{U}{U-1} + \epsilon K_{i+1}^{-1}$$

$$= f(i+1, q) + \epsilon K_{i+1}^{-1} . \tag{15}$$

Based on Eqs. (14) and (15), we have the steady temperature $T_{i+q}^{\mathrm{st}} = f(i+q, q) + \frac{\epsilon}{U}$. Then we can obtain the equation as

$$\begin{aligned} f(i,q) + \epsilon &= f(i+q, q) + \frac{\epsilon}{U} , \\ \epsilon &= \frac{\epsilon}{U} . \end{aligned} \tag{16}$$

Since $U > 1$, Eq. (16) is valid only when $\epsilon = 0$. Therefore Eq. (13) is proved. □

5. GMPT Peak Temperature Minimization

Genetic algorithm has the advantages in solving constrained combinatorial optimization problems by relying on bio-inspired operators such as mutation, crossover and selection. The characteristic of PTM–DVS schedule is very suitable to be coded as individuals for evolution, meanwhile, the peak temperature is treated as the fitness value. In this section, we adopt a particular genetic algorithm, GMPT, to find an optimized PTM(q) for the peak temperature reduction.

5.1. *Population initialization*

The first step of GMPT is to generate the population pop with a suitable number of individuals N_{pop}. In order to obtain a better divergence of initial population, the process for population initialization is designed as follows: we define that the maximum available period of PTM(q) is $p_{\max}$, the maximum available mode of PTM(q) is $q_{\max}$, set the lower bound value of *mode running time interval* $t_{\min}$ for each mode and discrete the $p_{\max}$ with a step of t_{step}. The population initialization algorithm (PIA) is shown in Algorithm 1. To guarantee that the PTM(q) schedule is feasible under deadline constraints, we obtain the lower service curve of PTM(q) based on Eq. (7). The detailed calculation is presented in Algorithm 2. Meanwhile, speed change points and their corresponding times are calculated to meet the input requirements of RTC Toolbox.

Algorithm 1. Population initialization algorithm

INPUT: N, $p_{\max}$, $q_{\max}$, $t_{\min}$
OUTPUT: PTM(q)

procedure PopulationInitialization(C)
  Randomly select the mode v_i repeatable from mode set N for $q_{\max}$ times and obtain the supply voltage array $\{v_1^r, \ldots, v_i^r, \ldots, v_q^r\}$
  for $i = 1$ to $q_{\max}$ **do**
    $t_{\text{upper}}^{\text{available}} = p_{\max} - (q_{\max} + i - 1)t_{\min}$
    $t_i^r \leftarrow$ randomly select from the interval $[t_{\min}, t_{\text{upper}}^{\text{available}}]$
  end for
  Random permutation of the time array $\{t_1^r, \ldots, t_i^r, \ldots, t_q^r\}$
  Pair the time array and mode array then obtain the PTM(q) as Eq. (5)
end procedure

Algorithm 2. Feasibility check of PTM(q) algorithm (isFeasible)

INPUT: PTM(q), Event Model
OUTPUT: True or False

procedure PopulationInitialization(C)
  Generate $\beta_l^{\text{PTM}(q)}(\Delta)$ of PTM(q) from Eq. (7)
  Generate $\beta_B(\Delta)$ from Eq. (4)
  if $\beta^l(\Delta) \geq \beta_B(\Delta)$ **then return** True
  else return False
  end if
end procedure

5.2. *Population operation*

According to the standard genetic algorithm, the process for population operation includes evaluation, selection, crossover and mutation. For evaluating fitness value of an individual, the *fitness* function is set as the reciprocal of $T_{\text{peak}}^{\text{PTM}(q)}$. Selection is to extract a certain proportion C of the population with the evaluation of fitness value. Afterwards, the crossover and mutation processes are continued. r_{cross} and r_{mute} are defined as the occurrence probabilities of crossover and mutation, respectively. The new individual will be selected by taking the place of the previous one only when it can pass through the feasibility check. The crossover and mutation are single-point operations which can be seen from Figs. 2(a) and 2(b), respectively. As soon as the crossover procedure happens, two random chromosomes will be selected. A random integer $i \in \{1, \ldots, qm\}$ will be generated where i is either an odd integer or an even integer. Another random $j \in \{1, \ldots, qn\}$ will be generated and it keeps the same odevity with i. The two chromosomes will exchange the values in i bits and j bits. Therefore, the operation of two chromosomes can be ensured either on the voltage parameter or on the time parameter. Afterwards, the new chromosomes will be checked by Algorithm 2 to make sure that they are feasible under deadline constraints. The new chromosome is reinserted into the population only if it passes the feasibility check, otherwise the population remains the original one. As the mutation procedure, the basic flow is similar to the crossover procedure. A random integer $i \in \{1, \ldots, qm\}$ will be generated to decide which bit will be mutated. If i is an odd integer, then v_i can be replaced by a

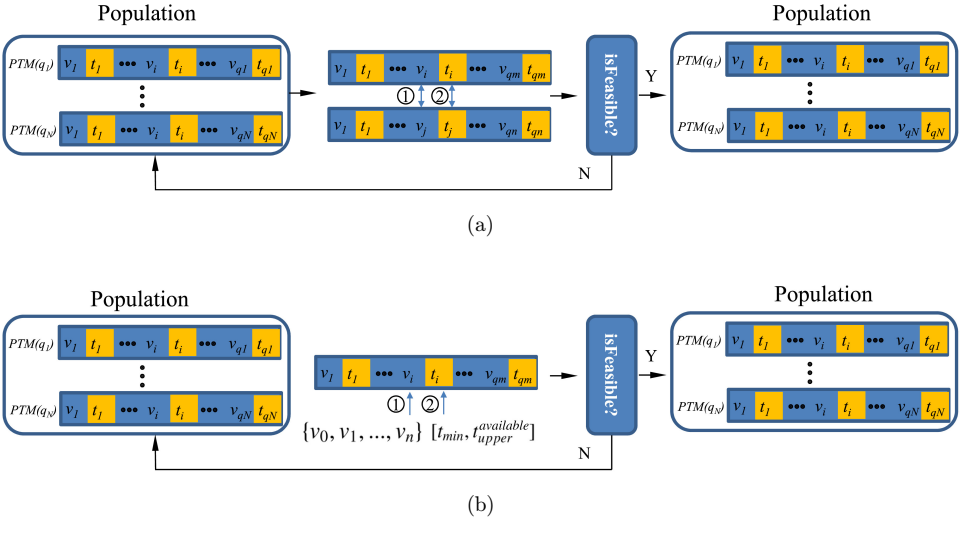


Fig. 2. The (a) crossover operation and (b) mutation operation. The ① and ② mean the two possible operations in the odd number bit or even number bit of the chromosome.

Algorithm 3. GMPT algorithm

INPUT: PTM(q), G, N_{pop}, Event Model, Thermal Model
OUTPUT: The highest fitness value of PTM(q) in pop

```
while i < N_pop do
    Generate PTM(q) using PIA
    if isFeasible(PTM(q),Event Model) then
        pop(i) = PTM(q), i ← i + 1
    end if
end while
for g = 1 to G do
    Calculate the population fitness value and rank
    Select the best certain proportion C from pop with the roulette selection method
    Crossover(pop), Mutation(pop)
    Reinsert to maintain the number of pop
end for
```

random value from the array $V = \{v_0, v_1, \ldots, v_n\}$. Otherwise, v_i will be replaced by a random integer from the time interval $[t_{\min}, t_{\text{upper}}^{\text{available}}]$. The new chromosome will be only reinserted into the population if it passes the feasibility check. Iterate these population operations for G times, then stop the calculation and output the best individual. The overall GMPT algorithm in pseudo-code is described in Algorithm 3.

6. Case Studies

The performance of our approach is studied and compared with the approach in Ref. 9 for both simulations and real-life measurements.

6.1. *Experimental setup*

For simulations, we implement a discrete-event simulation kernel in MATLAB by using the RTC/RTS Toolbox. For measurements, we extended the McFTP,[18,29] a fast thermal prototyping framework, to support our DVS–DPM scheduling scheme and deploy the generated schemes on a laptop equipped with an Intel Core i5-4210U CPU. The extended framework is shown in Fig. 3. The temperature is obtained from the on-chip temperature sensor with a measurement frequency of 200 ms and resolution of 1°C. The CPU fan is disabled in the BIOS of the laptop, so that the temperature will not be influenced by the variation of fan speed. The reported results are the means from 10 runs of every scheduling schemes.

The thermal and mode parameters of the processor are listed in Table 1. We adopt five runtime speed levels, i.e., 0, 0.4, 0.6, 0.8 and 1. A_m and B_m are calculated

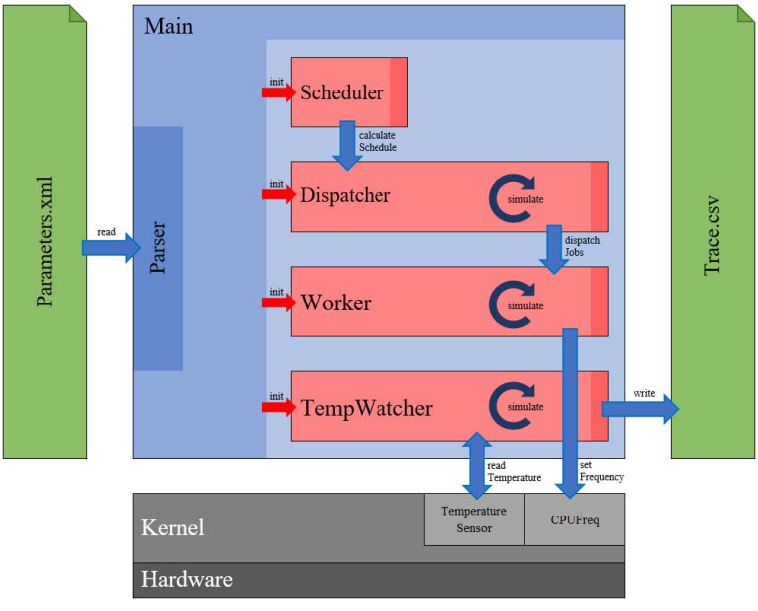


Fig. 3. The extended McFTP framework. Main: This component offers all the basic functionalities such as initialization, reading input parameters as well as the creation of all other threads. Once all threads have started, the simulation of the given schedule begins and this component sleeps throughout the entire experiment to minimize its performance impact. It is therefore completely inactive for the duration of the simulation. Scheduler: This component is responsible for scheduling decisions. As it currently only supports static scheduling approaches, all scheduling decisions can be determined beforehand. This results in a minimal overhead as the thread is able to exit almost instantly, finishing much earlier than the actual simulation. Dispatcher: This component is in charge of triggering all the events at the appropriate moment in time. Unlike in the case of the scheduler, the dispatcher needs to be active throughout the simulation as it needs to release the events by communicating with the worker. Even though all event parameters are known prior to execution, it needs to be ensured that these parameters are actually enforced, otherwise there would be no guarantee that constraints such as the period length apply. Worker: This component is used to manage all job-related functions and represents CPU core. Its primary purpose is to execute the events released by the dispatcher and to do so by forcing the highest possible workload on a CPU core. It is furthermore responsible for switching between idle and active states (when idle states are part of the schedule) and for changing to the different CPU frequencies within one period. TempWatcher: This component is used to monitor the CPU temperature throughout the entire simulation. The functionality has its own thread assigned to it to ensure that these measurements can be performed regularly and independently of the current state of the other components.

by linear approximation of Eq. (2) to the temperature and time datasets recorded by McFTP in different frequency/voltage levels.[25] The switch overhead t_{swact} is treated as 0.1 ms and t_{swon} and t_{swoff} are both 1 ms.[30] We use the event streams set as in Refs. 8 and 31 including a video codec, an audio codec and a network processor for communication management, as shown in Table 2. The video codec operation is considered with the task invocation range from 20 ms to 90 ms, which corresponds to 12–50 fps (frames per second). The (p, j, d, c) event model is adopted to model an input task. The relative deadline D_i is defined as $D_i = p_i$.

Table 1. Thermal and hardware model parameters.

m	s_m	f (GHz)	A_m ($\frac{\mathrm{W}}{\mathrm{V}}$)	B_m ($\frac{\mathrm{W}}{\mathrm{V}^\circ\mathrm{C}}$)	T_m^∞ ($^\circ$C)
0	0	0	1.695	0.03859	43.9
1	0.4	0.8	2.057	0.04358	47.2
2	0.6	1.1	3.299	0.06758	48.8
3	0.8	1.4	3.844	0.07531	51.0
4	1	1.7	5.157	0.07868	65.6

Table 2. Event stream setting.

	Video	Audio	Network
p (ms)	[20,90]	20	50
j (ms)	50	10	10
d (ms)	1	1	1
c (ms)	6	3	2

For the GMPT setting, the occurrence probabilities of crossover r_{cross} and mutation r_{mute} are set as 0.8 and 0.1, respectively. G is set as 30 and N_{pop} is set as 100 based on the convergence situation of the algorithm. The maximum available number of modes for PTM(q), $q_{\max}$, is set as 5. The maximum available period of PTM(q) is set as 50 ms and the lower bound for each mode $t_{\min}$ is set as 1 ms.

For comparisons, the PMPT approach[9] is chosen, which provides the exact minimum peak temperature for two-mode cases, i.e., sleep and full-speed modes. Both algorithms are verified and compared under simulation and real measurement.

6.2. *Results*

As the initial population is randomly generated, we first test the repeatability and convergence of GMPT algorithm. Audio codec task is selected as an example to show the detailed information about the initial individual distribution. The lower service curve generated by the optimum PTM(q) and the evolution process are shown in Fig. 4. Figure 4(a) presents the divergence of schedules which cover different levels of service. There is no service curve going below the service bound, since we have selected the service curves with Algorithm 2 to satisfy the deadline requirement. From Fig. 4(b), the performances of repeatability and convergence of GMPT with 10 different runs are reported. Figure 4(c) shows the lower service curve of the schedule optimized by GMPT and PMPT. Both generated lower service curves are very close to the lower service bound. Even though GMPT has a longer time span on the active mode, it can evolve to select an optimal mode with lower running speed with less heat. From these figures, we can also observe that the final PTM(q) schedule does not generate directly from the initial schedules, but evolves during the iterations.

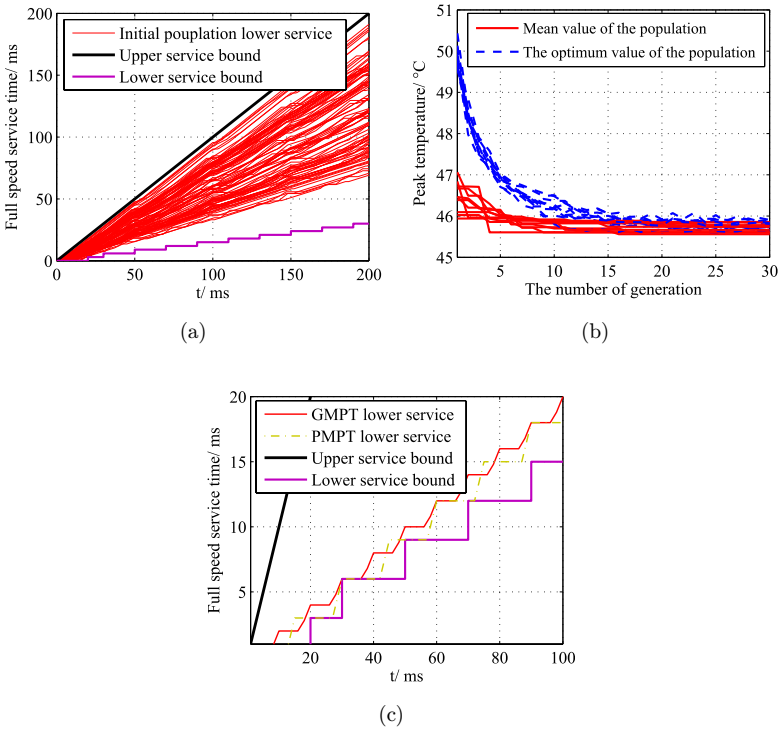
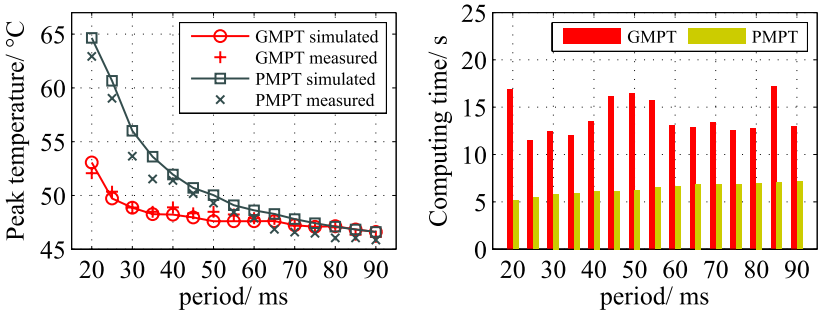


Fig. 4. The process of generating an optimized PTM(q) schedule by GMPT for the audio codec task: (a) The lower service curves of 100 randomly generated PTM(q) schedules by population initialization, (b) the evolution process of schedules for the peak temperature minimization for 10 repeats and (c) the lower service curve of optimized PTM(q) schedule by GMPT and PMPT.

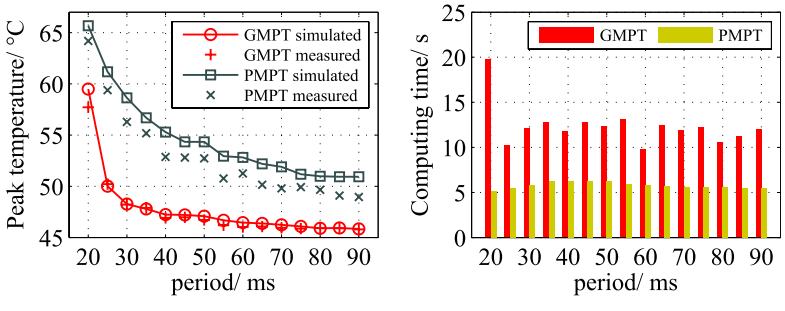
In order to verify the effectiveness of our GMPT under different workloads, we compare the simulated and measured peak temperatures generated by the two methods for single-event scenario (Fig. 5) and multi-event scenario (Fig. 6). From these figures we can see that, with the increase of invocation period, the peak temperature obtained by both the methods decreases. The reason is that the incremental invocation period reduces the time of active modes. The performance of GMPT is better than PMPT with average measured peak temperature reductions of 1.9°C (maximum 11.5°C) and 5.4°C (maximum 11.2°C) for single-event scenario and multi-event scenario, respectively. The average differences of peak temperature (0.4°C for both scenarios) between simulated GMPT and measured GMPT show a good accuracy performance of our approach. The computing time of GPMT is higher than PMPT, which does not show a clear tendency with the increase of the invocation period. However, computing time is not considered as a crucial criteria, as computing the optimal scheduling schemes of both methods is performed offline.

The robustness of our GMPT is also analyzed with the variations of the crossover rate r_{cross} and mutation rate r_{mute}. As shown in Fig. 7, the peak temperature obtained



(a) (b)

Fig. 5. The comparison of (a) simulated and measured peak temperatures and (b) computing times by the two methods under single-event scenario (video task) with different invocation periods of video codec.



(a) (b)

Fig. 6. The comparison of (a) simulated and measured peak temperatures and (b) computing times by the two methods under multi-event scenario (video, audio and network tasks) with different invocation periods of video codec.

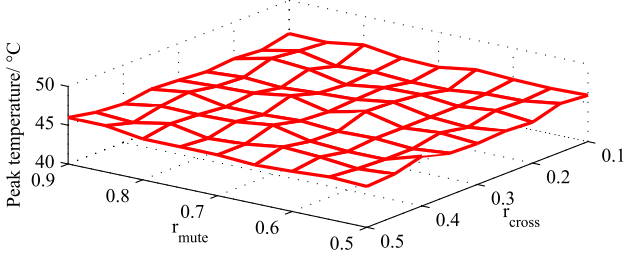


Fig. 7. Peak temperatures under different mutation and crossover rates.

by GMPT ranges from 45.2°C to 46.4°C with a standard deviation of 0.1°C. This small fluctuation is caused by the random generation of initial population, which can be ignored. Even though r_{cross} and r_{mute} have a large range, GMPT can still have a robustness ability of peak temperature reduction.

7. Conclusions

In this paper, we present a new approach based on DVS–PDM scheme to minimize the peak temperature for hard real-time system. The peak temperature minimization problem is defined as a combinatorial optimization problem with deadline constraints. Meanwhile, the formula of peak temperature is derived directly with the known thermal model of processor. Afterwards, an algorithm named GMPT is designed and we verified the effectiveness and accuracy of this algorithm by comparing it with a two-mode switch method on the simulation and real-life measurement platforms. Experimental results show that our method can reduce the peak temperature by 1.9°C (single-event scenario) and 5.4°C (multi-event scenario) compared to PMPT methods.

References

1. H. Chai, G. Zhang, J. Zhou, J. Sun, L. Huang and T. Wang, A short review of security-aware techniques in real-time embedded systems, *J. Circuits Syst. Comput.* (2018), doi: 10.1142/S0218126619300022.
2. 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) 1750159.
3. B. Hu, K. Huang, G. Chen, L. Cheng and A. Knoll, Evaluation and improvements of runtime monitoring methods for real-time event streams, *ACM Trans. Embed. Comput. Syst.* **15** (2016) 56.
4. T. Chantem, X. S. Hu and R. P. Dick, Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **19** (2011) 1884–1897.
5. V. Narayanan and Y. Xie, Reliability concerns in embedded system designs, *Computer* (2006) 118–120.
6. J. Zhou and T. Wei, Stochastic thermal-aware real-time task scheduling with considerations of soft errors, *J. Syst. Softw.* **102** (2015) 123–133.
7. N. Bansal and K. Pruhs, Speed scaling to manage temperature, *Proc. Annu. Symp. Theoretical Aspects of Computer Science* (Springer, 2005), pp. 460–471.
8. P. Kumar and L. Thiele, Cool shapers: Shaping real-time tasks for improved thermal guarantees, *Proc. Design Automation Conf. (DATE)* (ACM, 2011), pp. 468–473.
9. L. Cheng, K. Huang, G. Chen, B. Hu and A. Knoll, Periodic thermal management for hard real-time systems, *Proc. Int. Symp. Industrial Embedded Systems (SIES)* (IEEE, 2015), pp. 1–10.
10. P. Kumar and L. Thiele, Thermally optimal stop-go scheduling of task graphs with real-time constraints, *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)* (IEEE, 2011), pp. 123–128.

11. T. Chantem, X. S. Hu and R. P. Dick, Online work maximization under a peak temperature constraint, *Proc. Int. Symp. Low Power Electronics and Design (ISLPED)* (ACM, 2009), pp. 105–110.
12. S. Zhang and K. S. Chatha, Approximation algorithm for the temperature-aware scheduling problem, *Proc. Int. Conf. Computer-Aided Design (ICCAD)* (IEEE, 2007), pp. 281–288.
13. G. Chen, K. Huang, C. Buckl and A. Knoll, Applying pay-burst-only-once principle for periodic power management in hard real-time pipelined multiprocessor systems, *ACM Trans. Des. Autom. Electron. Syst.* **20** (2015) 26.
14. G. Chen, K. Huang, J. Huang, C. Buckl and A. Knoll, Effective online power management with adaptive interplay of DVS and DPM for embedded real-time system, *Proc. Euromicro Conf. Digital System Design (DSD)* (IEEE, 2013), pp. 881–889.
15. G. Chen, K. Huang and A. Knoll, Energy optimization for real-time multiprocessor system-on-chip with optimal DVFs and DPM combination, *ACM Trans. Embed. Comput. Syst.* **13** (2014) 111.
16. J. Zhuo, C. Chakrabarti and N. Chang, Energy management of DVS-DPM enabled embedded systems powered by fuel cell-battery hybrid source, *Proc. Int. Symp. Low Power Electronics and Design (ISLPED)* (ACM, 2007), pp. 322–327.
17. L. Thiele, S. Chakraborty and M. Naedele, Real-time calculus for scheduling hard real-time systems, *Proc. Int. Symp. Circuits and Systems (ISCAS)*, Vol. 4 (IEEE, 2000), pp. 101–104.
18. L. Cheng, Z. Zhao, K. Huang, G. Chen and A. Knoll, McFTP: A framework to explore and prototype multi-core thermal managements on real processors, *Proc. Int. Conf. Embedded Software and Systems (ICESS)* (2017).
19. G. Quan and Y. Zhang, Leakage aware feasibility analysis for temperature-constrained hard real-time periodic tasks, *Proc. Euromicro Conf. Real-Time Systems (ECRTS)* (IEEE, 2009), pp. 207–216.
20. R. Jayaseelan and T. Mitra, Temperature aware task sequencing and voltage scaling, *Proc. Int. Conf. Computer-Aided Design (ICCAD)* (IEEE Press, 2008), pp. 618–623.
21. H. Huang, V. Chaturvedi, G. Liu and G. Quan, Leakage aware scheduling on maximum temperature minimization for periodic hard real-time systems, *J. Low Power Electron.* **8** (2012) 378–393.
22. J. Zhou, J. Yan, J. Chen and T. Wei, Peak temperature minimization via task allocation and splitting for heterogeneous MPSoC real-time systems, *J. Signal Process. Syst.* **84** (2016) 111–121.
23. 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.
24. L. Schor, I. Bacivarov, H. Yang and L. Thiele, Worst-case temperature guarantees for real-time applications on multi-core systems, *Proc. Real-Time and Embedded Technology and Applications Symp. (RTAS)* (IEEE, 2012), pp. 87–96.
25. M. Ahmed, N. Fisher, S. Wang and P. Hettiarachchi, Minimizing peak temperature in embedded real-time systems via thermal-aware periodic resources, *Sustain. Comput. Inf. Syst.* **1** (2011) 226–240.
26. L. Cheng, K. Huang, G. Chen, B. Hu and A. Knoll, Minimizing peak temperature for pipelined hard real-time systems, *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE)* (IEEE, 2016), pp. 1090–1095.

27. M. A. Awan and S. M. Petters, Energy-aware partitioning of tasks onto a heterogeneous multi-core platform, *Proc. Real-Time and Embedded Technology and Applications Symp. (RTAS)* (IEEE, 2013), pp. 205–214.
28. M. Mohaqeqi, M. Kargahi and A. Movaghar, Analytical leakage-aware thermal modeling of a real-time system, *IEEE Trans. Comput.* **63** (2014) 1378–1392.
29. L. Cheng, K. Huang, G. Chen, B. Hu, Z. Jiang and A. Knoll, Applying periodic thermal management on hard real-time systems to minimize peak temperature, *J. Circuits Syst. Comput.* (2018), doi: 10.1142/S0218126618502080.
30. Y. Liu, S. C. Draper and N. S. Kim, SleepScale: Runtime joint speed scaling and sleep states management for power efficient data centers, *ACM SIGARCH Comput. Archit. News* **42** (2014) 313–324.
31. D. Rai, H. Yang, I. Bacivarov, J.-J. Chen and L. Thiele, Worst-case temperature analysis for real-time systems, *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE)* (IEEE, 2011), pp. 1–6.