The Journal of Systems and Software 102 (2015) 123–133

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Stochastic thermal-aware real-time task scheduling with considerations of soft errors

Junlong Zhou, Tongquan Wei[1],*

Shanghai Key Laboratory of Multidimensional Information Processing, and the Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China

ARTICLE INFO

Article history:
Received 7 August 2014
Revised 19 October 2014
Accepted 4 December 2014
Available online 8 January 2015

Keywords:
Fault-tolerance
Real-time systems
Thermal-aware

ABSTRACT

With the continued scaling of the CMOS devices, the exponential increase in power density has strikingly elevated the temperature of on-chip systems. Dynamic voltage/frequency scaling is a widely utilized system level power management technique to reduce the energy consumption and lower the on-chip temperature. However, scaling the voltage or frequency for thermal management leads to an increase in soft error rates, thus has adverse impact on system reliability. In this paper, the authors propose a stochastic thermal-aware task scheduling algorithm that considers soft errors in real-time embedded systems. For the given customer-defined soft error related target reliability and the maximum peak temperature, the proposed scheduling algorithm generates an energy-efficient task schedule by selecting the energy efficient operating frequency for each task and alternating the execution of hot tasks and cool tasks at the scaled operating frequency. The proposed stochastic scheduling algorithm features the consideration of uncertainty in transient fault occurrences. To handle the uncertainty, a fault adaptation variable α is introduced to adapt task execution to the stochastic property of fault occurrences. An energy efficiency factor δ is also introduced to facilitate the enhancement of energy efficiency by maximizing the energy saved per unit slack. Extensive simulations of synthetic real-time tasks and real-life benchmarking tasks were performed to validate the effectiveness of the proposed algorithm. Experimental results show that the proposed algorithm consumes up to 17.8% less energy as compared to the benchmarking schemes, and the peak temperature of the proposed algorithm is always below the maximum temperature limit and can be up to 9.6 °C lower than that of the benchmarking schemes.

1. Introduction

As technology advances toward the deep submicron region, the chip power density has increased exponentially, which in turn leads to huge energy consumption and elevated chip temperature. A system will fall into the predicament of functional incorrectness, low reliability and even hardware failures if the operating temperature exceeds a certain threshold. Thus, energy and thermal management has been a significant and pressing research issue in computing systems, especially for real-time embedded systems with limited cooling techniques.

The dynamic voltage and frequency scaling (DVFS) is a widely used system level power management technique that exploits technological advances in power supply circuits to reduce processor power consumption and lower chip temperature by dynamically scaling down the processor speed. Considerable research effort has been devoted to the design of energy-efficient thermal-aware real-time systems by using DVFS in the past decades (Bao et al., 2009; Chen et al., 2011, 2012; Coronel and Simo, 2012; Terzopoulos and Karatza, 2012; Wang et al., 2009). These works are originally motivated to address the issues of high energy consumption and high temperature due to high power density. However, they ignore the fact that the transient fault rate increases dramatically when a system is running in the low power state. It has been shown that scaling down the processor speed increases the transient fault rates, especially for those induced by cosmic ray radiations, and thus degrades system reliability (Zhu et al., 2004). Thus soft errors need to be tolerated to provide temporal and logical correctness for real-time tasks when DVFS is operated to reduce energy consumption and temperature. In this paper, the checkpointing technique is utilized to tolerate soft errors while a DVFS-based scheme is designed to minimize energy consumption under the thermal constraint. Checkpointing with rollback-recovery is a popular fault-tolerant technique deployed in real-time embedded

* Corresponding author. Tel.: +86 2154345184; fax: +86 2154345184.
E-mail addresses: joe@ecnu.cn (J. Zhou), tqwei@cs.ecnu.edu.cn (T. Wei).
[1] Member, IEEE.

http://dx.doi.org/10.1016/j.jss.2014.12.009
0164-1212/© 2015 Elsevier Inc. All rights reserved.

systems to maintain the system reliability (Zhang and Chakrabarty, 2006). In checkpointing, the state of the system is saved on a stable storage at each checkpoint. When a transient fault occurs, the system rolls back to the most recent checkpoint and resumes the execution.

Most existing studies (Li et al., 2013; Terzopoulos and Karatza, 2014; Wei et al., 2011, 2006, 2008, 2012; Xu et al., 2014; Zhu and Aydin, 2006) on the fault-tolerance real-time scheduling exploit all the slack generated due to early completion of real-time tasks to save energy under the reliability constraint. The slack available is utilized for both energy savings and fault tolerance, in which temperature optimization is ignored. However, this is not suitable for real-time systems deployed certain safety-critical applications (e.g. implantable cardioverters in medical applications) where energy, reliability and thermal control are all crucial for the correct operation of systems. Thus, in this paper the generated slack is utilized to provide fault tolerance, save energy, and reduce the temperature. The existing works (Li et al., 2013; Wei et al., 2011, 2006, 2008, 2012; Xu et al., 2014; Zhu and Aydin, 2006) also adopt an imprecise power model which considers dynamic power the main source of power dissipation, or assumes that static/leakage power is a constant. These claims are inaccurate since it is known that leakage power can contribute as much as 50% of the total power dissipation of modern microprocessor circuits (Chandrakasan et al., 2000). In addition, temperature-dependent leakage power is ignored in these works. The proposed thermal-aware fault-tolerance task scheduling scheme takes into account the temperature's dependency of leakage power. It trades energy for thermal control by maximizing the energy efficiency with respect to slack and exploiting the remaining slack for thermal management.

Little research has been conducted on the management of reliability for thermal-aware real-time systems. In Ukhov et al. (2012), the authors proposed a fast and accurate technique for the steady state dynamic temperature profile (SSDTP) calculation, and provided an analytical solution for the steady state dynamic temperature analysis (SSDTA) of embedded multiprocessor systems. Based on the temperature-aware reliability model and the thermal cycling failure mechanism, the presented approach efficiently performs a temperature-aware reliability optimization for embedded multiprocessor systems. The lifetime of an embedded system can be improved significantly by considering the steady state dynamic temperature profile of the system during the design stage. However, instead of transient faults, the technique focuses on the thermal cycling failure. The authors are interested in transient faults in this study. Furthermore, the technique does not consider energy as an optimization constraint.

Recently, a novel thermal management technique, referred to as the thermal-aware task sequencing combined with dynamic voltage/frequency scaling, has attracted considerable research attention. It utilizes temperature characteristics of tasks to reduce the peak temperature of processors at a scaled operating frequency without incurring extra monetary expenses (Jayaseelan and Mitra, 2008; Yang et al., 2008; Zhang and Chatha, 2010). The slack available is allocated for voltage/frequency scaling and tasks are paired and alternatively executed at the scaled frequency in the order of being cool–hot. Unlike Jayaseelan and Mitra (2008) and Zhang and Chatha (2010) that alternate the execution of tasks in the order of being cool–hot, the proposed algorithm alternates the execution of a hot task and a cool task in the order of being hot–cool. This is motivated by the observation that the final temperature of tasks executing in the hot–cool order is lower than that of tasks executing in the cool–hot order (Yang et al., 2008). Compared to the online scheduler presented in Yang et al. (2008), the proposed method does not necessitate the deployment of a runtime temperature monitor and thermal sensors. Moreover, the proposed DVFS-based scheme is more practical since it takes into account the time overhead of frequency switching, which is not considered in Jayaseelan and Mitra (2008), Zhang and Chatha (2010), and Yang et al. (2008).

In this paper, the authors propose a stochastic thermal-aware real-time task scheduling algorithm under the constraint of system reliability requirement. The algorithm takes as input a given set of real-time tasks, the customer-defined target reliability, and the maximum peak temperature. It generates a task schedule that meets the design requirements by selecting the energy efficient operating frequency for each task and alternating the execution of a *hot* task and a *cool* task at the scaled operating frequency. The energy efficient operating frequency for a task is the one that achieves the maximum energy efficiency. The major contributions of this paper are summarized as follows.

- The proposed task scheduling algorithm jointly tackles the energy efficiency, fault tolerance, and thermal control for real-time systems. A fault adaption variable α is also introduced to adapt task execution time including fault recovery overhead to the Poisson probability distribution of fault occurrences, which enables the designing of stochastic real-time systems based on status quo of fault occurrences.
- The proposed algorithm enhances the efficiency of reducing the energy consumption. An energy efficiency factor δ that indicates the energy saved per unit slack is introduced and maximized for each task. In addition, practical issues such as temperature's dependency of leakage power and time overhead of frequency switching are also taken into account.
- Task sequencing combined with dynamic voltage/frequency scaling is adopted to meet the thermal requirement of the system. The proposed algorithm alternates the execution of a *hot* task and a *cool* task at the scaled operating frequency for better thermal control.
- Two sets of simulation experiments have been implemented to validate the effectiveness of the proposed algorithm in energy efficiency and thermal management. Simulation results have demonstrated that the proposed algorithm achieves better performance when compared to the benchmarking schemes.

The rest of the paper is organized as follows. Section 2 introduces the system architecture and models. Section 3 formulates the problem definition. Section 4 describes the proposed stochastic thermal-aware task scheduling algorithm under the target reliability goal constraint. Section 5 presents the experimental results and Section 6 concludes the paper.

2. System architecture and models

In this paper, real-time tasks in a task set are assumed to execute on a DVFS-enabled processor. The processor is equipped with L discrete normalized frequencies $\{f_1, f_2, \ldots, f_L\}$, and $0 < f_{\min} = f_1 < f_2 < \cdots < f_L = f_{\max} = 1.0$ holds for the sake of easy presentation, where $f_{\min}$ indicates the minimum operating frequency and $f_{\max}$ denotes the maximum operating frequency. The task set Γ consisting of N real-time tasks $\{\tau_1, \tau_2, \ldots, \tau_N\}$ is a frame-based task set, in which all tasks share a common deadline D that is also the frame size (Yang et al., 2009). There are many embedded real-time systems (Berten et al., 2008; Li et al., 2013; Yang et al., 2009; Zhu et al., 2004) operating on a cyclic basis and having a set of tasks that must execute in a frame. When all task executions have finished, the whole frame is repeated. These real-time systems find their applications in multimedia data transmissions, delay and jitter control in ATM networks, chemical process control, and air traffic control (Han et al., 1996; Nguyen and Cheng, 1996).

It is assumed that tasks in the task set are independent and non-preemptive. As a result, the frequency transition occurs before a task starts its execution or after the task finishes its execution. The proposed method hence produces exactly one frequency level for each task and at most one frequency transition occurs for a task in a frame. The incurred frequency switching overhead can be incorporated in task execution time for simplicity. Let O_{sw} denote the time overhead of frequency switching, and e_i represent the worst case execution time of task τ_i $(1 \le i \le N)$ at the maximum operating frequency $f_{\max}$, then the execution time of task τ_i at its operating frequency $f(\tau_i)$ is given by $\frac{e_i}{f(\tau_i)} + O_{\text{sw}}$.

2.1. Fault and recovery model

Due to the ever increasing level of integration and continued reducing size of transistor features, the number of transient faults in circuits has been rising rapidly, especially for embedded systems deployed in harsh operating environment. Moreover, blindly applying DVFS for energy savings without fault-tolerant techniques will result in drastic increase in transient fault rates. Therefore, fault-tolerant techniques need to be deployed in energy-efficient design. In this paper, a widely used technique for fault-tolerance, equidistant checkpointing, is adopted for improving system reliability. In equidistant checkpointing, checkpoint intervals for a given task are set to be equal. At each checkpoint, the state of the system is saved in a secure device to prepare for rolling back to the most recent checkpoint and re-execution. Given the worst case number of transient fault occurrences during the execution of task τ_i at the operating frequency $f(\tau_i)$, which is denoted as k_i, the optimal number of checkpoints for task τ_i at $f(\tau_i)$ is then represented by $N_{\text{opt},i}$, as is given by

$$N_{\text{opt},i} = \left\| \sqrt{\frac{k_i}{O_{\text{ck}}}\left(\frac{e_i}{f(\tau_i)} + O_{\text{sw}}\right)} - 1 \right\|, \tag{1}$$

where O_{ck} is the time overhead of checkpointing (Zhang and Chakrabarty, 2006), e_i is the task execution time at the maximum frequency $f_{\max}$, and O_{sw} is the time overhead of frequency switching.

Let $\Psi_{\text{best},i}$ be the execution time of task τ_i including checkpointing overhead in the case of no fault occurrences, then it is given by

$$\Psi_{\text{best},i} = \frac{e_i}{f(\tau_i)} + O_{\text{sw}} + N_{\text{opt},i} \times O_{\text{ck}}, \tag{2}$$

where $\frac{e_i}{f(\tau_i)} + O_{\text{sw}}$ is the execution time of task τ_i at the frequency $f(\tau_i)$.

Let $\Psi_{\text{worst},i}$ be the execution time of task τ_i including checkpointing and fault recovery overhead in the presence of k_i transient fault occurrences. In other words, the $\Psi_{\text{worst},i}$ consists of two terms. The first term is the $\Psi_{\text{best},i}$ and the second term is the fault recovery overhead in the presence of k_i transient fault occurrences. The $\Psi_{\text{worst},i}$ is then given by

$$\Psi_{\text{worst},i} = \Psi_{\text{best},i} + \frac{k_i}{(N_{\text{opt},i}+1)}\left(\frac{e_i}{f(\tau_i)} + O_{\text{sw}}\right) + 2k_i O_{\text{ck}}, \tag{3}$$

where $\frac{k_i}{(N_{\text{opt},i}+1)}\left(\frac{e_i}{f(\tau_i)} + O_{\text{sw}}\right)$ represents the recovery overheads to tolerate k_i faults and $2k_iO_{\text{ck}}$ denotes the overheads of k_i checkpoint savings and system state retrievals (Zhang and Chakrabarty, 2006).

The $\Psi_{\text{best},i}$ and $\Psi_{\text{worst},i}$ are constant if the variables $N_{\text{opt},i}$, $f(\tau_i)$ and k_i are fixed. However, due to the stochastic property of transient fault occurrences, the execution time of a task is not determined. A fault adaptation variable, which is denoted as $\alpha \in [0, 1]$, is introduced in this work to model the uncertainty in task execution caused by transient fault occurrences. As a consequence, the execution time Ψ_i of task τ_i can be formulated as a function of $\Psi_{\text{best},i}$ and $\Psi_{\text{worst},i}$, as is given by

$$\Psi_i = \alpha \times \Psi_{\text{worst},i} + (1-\alpha) \times \Psi_{\text{best},i}. \tag{4}$$

The execution time of task τ_i is $\Psi_{\text{best},i}$ when $\alpha = 0$ and is $\Psi_{\text{worst},i}$ when $\alpha = 1$.

The average fault arrival rate at the operating frequency level ℓ for $1 \le \ell \le L$ is denoted by ν_ℓ, where L is the number of frequency levels supported by the processor. Assuming the operating frequency is f_ℓ, then the ν_ℓ can be obtained using the equation $\nu_\ell = \lambda \times e^{-\xi f_\ell}$, where λ and ξ are constant parameters (Zhu et al., 2004). Since the Poisson distribution is typically utilized to model transient faults, the probability of x transient fault occurrences during the execution of task τ_i at frequency f_ℓ is then given by

$$P(x) = \frac{e^{-\nu_\ell \times \Psi_i} \times (\nu_\ell \times \Psi_i)^x}{x!}. \tag{5}$$

The reliability of a real-time task is defined as the probability that the task executes successfully till its deadline in the presence of soft errors. For a task τ_i that is supposed to tolerate k_i errors during its execution at the frequency f_ℓ, the task is deemed to be able to finish its execution successfully in the presence of at most k_i errors. When the number of transient fault occurrences exceeds k_i, the execution of the task fails. Hence, the reliability of task τ_i is the probability that at most k_i errors occur during its execution, which is written as

$$R_i = P(0 \le x \le k_i) = \sum_{x=0}^{k_i} \frac{e^{-\nu_\ell \times \Psi_i} \times (\nu_\ell \times \Psi_i)^x}{x!}. \tag{6}$$

The correct operation of a system depends on the successful execution of all tasks in a task set. The target system contains N tasks, thus, the current reliability of the system, which is denoted by R_{curr}, is given by the product of reliabilities of individual tasks in the task set, that is,

$$R_{\text{curr}} = \prod_{i=1}^{N} R_i. \tag{7}$$

In a reliability-constrained real-time system, the target reliability is denoted by R_{goal}, and the reliability of the system is maintained if the inequality $R_{\text{goal}} \le R_{\text{curr}}$ holds for $\alpha = 1$.

2.2. Power model

The power consumption of a CMOS device can be modeled as the sum of dynamic power dissipation and leakage power dissipation. The dynamic power consumption is independent of the temperature and can be formulated as $P_{\text{dyn}} \propto V_{\text{dd}}^2 f$ (Weste and Eshraghian, 1992), where V_{dd} is the supply voltage, and f is the operating frequency. Assuming processors use voltage/frequency scaling technique to scale frequency, the operating frequency is then approximately linear with the supply voltage (Weste and Eshraghian, 1992). As a result, the average dynamic power consumption can be estimated by a strictly increasing and convex function of the operating frequency, that is, $P_{\text{dyn}} \propto f^3$. Thus, the dynamic power consumption of task τ_i on the processor at frequency $f(\tau_i)$ is given by $P_{\text{dyn}} = C_{\text{ef}} f(\tau_i)^3$, where C_{ef} is the effective switching capacitance.

The leakage power consumption is temperature dependent and can be expressed as $P_{\text{leak}} = N_{\text{gate}} V_{\text{dd}} I_{\text{leak}}$, where N_{gate} is the number of gates and I_{leak} is the leakage current. The leakage current I_{leak} can be formulated by a nonlinear exponential equation (Liao et al., 2005) as

$$I_{\text{leak}} = I_s\left(AT^2 e^{(\vartheta_1 V_{\text{dd}} + \vartheta_2)/T} + Be^{(\vartheta_3 V_{\text{dd}} + \vartheta_4)}\right), \tag{8}$$

where I_s is the leakage current at a certain reference temperature and supply voltage, T is the operating temperature, V_{dd} is the supply voltage, and A, B, ϑ_1, ϑ_2, ϑ_3, and ϑ_4 are empirically determined technology constants. Since the operating frequency is nearly linear with the supply voltage (Weste and Eshraghian, 1992) and the leakage current changes super linearly with temperature (Liu et al., 2007), the leakage power consumption of task τ_i on the processor at frequency $f(\tau_i)$ can be effectively estimated to be $P_{\text{leak}} = C_1 f(\tau_i) + C_2 T f(\tau_i)$ (Huang et al., 2011), where C_1 and C_2 are curve fitting constants, and T is the operating temperature. Therefore, the power consumption of task τ_i on the processor at frequency $f(\tau_i)$ is given by

$$P_i = C_{\text{ef}} f(\tau_i)^3 + C_1 f(\tau_i) + C_2 T f(\tau_i), \tag{9}$$

where C_{ef} is the effective switching capacitance.

Checkpointing operations including checkpoint saving and retrieval incurs power consumption. It has been shown in Zhang and Chakrabarty (2006) that checkpointing power consumption depends on the checkpoint size, access speed of the checkpoint storage, and power consumption of the checkpoint storage. The checkpoint size strongly depends on the task set. The embedded applications may produce relatively small checkpoint with sizes in the order of a few kilobytes. It is clear that the checkpointing power consumption is independent of the processor operating frequency and temperature. On the contrary, the processor power consumption is a function of processor operating frequency and temperature, as is given in Eq. (9). It also has been shown in Meneses et al. (2012) that processor power consumption drops to the power draw of idle state during checkpointing. Owing to this analysis, we adopt processor power consumption in the DVFS-based frequency selection scheme described in Algorithm 2 and consider both processor power consumption and checkpointing power consumption when energy is derived in the experimental section.

2.3. Temperature model

An accurate and practical dynamic model of temperature is needed to accurately characterize the thermal behavior of an application. In this paper, a dynamic thermal model proposed by Skadron et al. (2004) that is widely used in the literature is adopted as the temperature model to predict the temperature of the processor. The model is based on a lumped RC model, and the temperature $T(t)$ at the time instance t is given by

$$T(t) = T_{\text{std}} - (T_{\text{std}} - T_{\text{init}})e^{\frac{-t}{RC}}, \tag{10}$$

where T_{std} is the steady state temperature of a task, T_{init} is the initial temperature, R is the thermal resistance, and C is the thermal capacitance. The thermal resistance R and capacitance C are processor architecture dependent constants.

The steady state temperature of a task is the temperature that will be reached if infinite number of instances of the task execute continuously on the processor. It is associated with a certain average input power, and is given by

$$T_{\text{std}} = PR + T_{\text{amb}}. \tag{11}$$

where P is the power consumption, R is the thermal resistance and T_{amb} is the die's ambient temperature. Since the power consumptions of different tasks vary significantly, the steady state temperatures of tasks in the task set are different.

3. Problem definition

The focus of the study is to minimize the energy consumed by tasks in a task set using dynamic voltage/frequency scaling under the thermal constraint and system reliability constraint. Let E_{tot} denote the energy consumed by all tasks in a task set, then it is given by

$$E_{\text{tot}} = \sum_{i=1}^{N} E_i, \tag{12}$$

where E_i is the energy consumed by task τ_i, including the energy cost of task execution and checkpointing operations. The energy consumption is calculated as the product of power consumption and execution time.

For thermal-aware task scheduling, the maximum temperature limit, which is denoted by T_{max}, is in general specified based on system design requirements. Let T_{peak} denote the peak temperature of tasks in a task set, then it is given by

$$T_{\text{peak}} = \max\{T(t)| \quad \forall t \in [0, D]\}, \tag{13}$$

where $T(t)$ is the temperature at the time instance t and can be derived by using Eq. (10). The system is deemed to be in a safe mode when the peak temperature T_{peak} does not exceed the threshold temperature T_{max}. With regards to the reliability, the system operates dependably in its expected environment if the current reliability R_{curr} is no less than the system target reliability R_{goal}. In addition to the thermal constraint and the reliability constraint, all task in a task set must finish their execution before the deadline D. Considering the above design constraints, the proposed stochastic scheduling problem can be formulated as follows. For a given task set of real-time tasks that share a common deadline and a DVFS-capable processor, it is expected to derive the operating frequency for each task and arrange the execution sequence for all tasks in the task set such that the energy consumption of the system is minimized under the reliability and thermal design constraints. In other words, the problem can be formulated into the below form.

$$\text{Minimize: } E_{\text{tot}} = \sum_{i=1}^{N} E_i$$

$$\text{Subject to: } T_{\text{peak}} \le T_{\text{max}}$$

$$R_{\text{goal}} \le R_{\text{curr}} = \prod_{i=1}^{N} R_i$$

$$\sum_{i=1}^{N} \Psi_i \le D$$

where E_{tot}, T_{peak}, R_{curr}, and Ψ_i are given by Eqs. (12), (13), (7), and (4), respectively.

4. Stochastic task scheduling with reliability and thermal considerations

The objective of the proposed stochastic task scheduling scheme is to generate an energy optimum schedule without violating the reliability and thermal design constraints. The uncertainty in transient fault occurrences is considered and modeled using the Poisson probability distribution in the proposed scheme. Algorithm 1 is developed in this section to present the overview of the proposed stochastic task scheduling scheme. Inputs to the algorithm are a given task set Γ, the target reliability R_{goal}, and the thermal constraint T_{max}. Line 1 of the algorithm randomly assigns a value for the fault adaptation variable α, and the execution time of each task is updated based on the selected α (line 3). Energy optimization is achieved when each task executes at the energy efficient operating frequency (line 4) and the method to obtain the energy efficient frequency is described in Algorithm 2. To avoid the violation of the maximum temperature limit T_{max}, a thermal-aware task sequencing is utilized to reduce the peak temperature T_{peak} of tasks (line 5), as is given in Algorithm 3. The reliability of the obtained task schedule is produced using the Monte Carlo simulation method (line 6), and if it does not satisfy the stop-condition of Algorithm 1, the value of fault adaptation variable α is adjusted and the above procedures is repeated. In contrast, if $(R_{\text{curr}} - R_{\text{goal}}) \ge \epsilon > 0$ holds for an arbitrarily small positive number ϵ, the reliability of the system satisfies the stop-condition of Algorithm 1 and the output is the desired task schedule. The time complexity of Algorithm 1 is $O(NL + N^2)$, where N is the number of tasks in the task set, and L is the processor-supported frequency levels.

Slack is generated due to early completion of real-time tasks. In this paper the generated slack is utilized to provide fault tolerance, save energy, and reduce the temperature, which necessitates a trade-off among fault tolerance, energy efficiency, and thermal management. The major contributions of this paper are three-fold. First, the slack is utilized to guarantee a certain level of reliability requirement in the presence of stochastic soft errors, second, the energy efficiency of the proposed scheme is achieved by utilizing the slack under the reliability constraint, and finally, the temperature is reduced by utilizing the task sequencing technique. The temperature

Algorithm 1 Generate the energy optimum task schedule under the reliability and thermal constraints.

```
Require: task set $\Gamma$ & the target reliability goal $R_{\text{goal}}$ & the thermal constraint $T_{\max}$;
1: randomly pick a value of $\alpha$ in $0 \le \alpha \le 1$;
2: repeat
3:     update the execution time of each task based on $\alpha$, then update $\alpha$;
4:     calculate the energy efficient operating frequency of each task using Algorithm 2;
5:     derive the thermal-aware task sequence under $T_{\max}$ using Algorithm 3;
6:     obtain $R_{\text{curr}}$ of the task set $\Gamma$ using Monte Carlo simulation;
7: until $(R_{\text{curr}} - R_{\text{goal}}) \ge \epsilon > 0$;
```

is further reduced by using the remaining slack. Unlike the previous work presented by Zhu and Aydin (2006) that utilizes all slack for energy savings when the reliability constraint is satisfied, the proposed scheme trades energy for thermal control by maximizing the energy efficiency with respect to slack and utilizing the remaining slack for thermal management, which is demonstrated in Algorithms 2 and 3. In other words, Algorithm 2 aims at maximizing the energy efficiency with respect to slack instead of maximizing energy savings, and consequently Algorithm 3 reduces temperature by using both the remaining slack and task sequencing technique. The details of each step in the proposed scheme including Algorithms 2 and 3 are described in the following subsections.

4.1. Parallel schedule generation based on the fault adaptation variable α

In this paper, the authors introduce a fault adaptation variable α which adapts the task execution behavior to the Poisson distribution of probabilistic transient fault occurrences. In comparison with the traditional method of designing for corner cases, the proposed scheme features the consideration of the uncertainty in fault occurrences. The variable α ranges from 0 to 1, indicating the fault-free scenario to the scenario of the worst case fault occurrences. The α is a deterministic variable. Due to the statistical property of transient fault occurrences, there is no existence of deterministic relationship between the deterministic fault adaptation variable α and the reliability of tasks in the task set. Thus, if the current system reliability R_{curr} does not satisfy the stop-condition of Algorithm 1, it cannot be utilized to direct the value assignment of α to be employed in the next iteration of Algorithm 1. In other words, the selection of the α is independent of the current value of the system reliability. The α that models the status quo of fault occurrences could be any value in the range of $[0, 1]$. As a result, a binary search-based approach is adopted to search for the target fault adaptation variable.

The binary search is a simple yet efficient search algorithm to find the position of a specific key value in an ordered value sequence. In contrast to other search algorithms, it needs less comparisons since the search interval is halved in each step. The binary search-based approach operates as follows. For a given initial range of $\alpha \in [0, 1]$, the initial value of α is denoted by α_0, then the next values of α could be varied in the range of both $[0, \alpha_0]$ and $[\alpha_0, 1]$. If the current system reliability R_{curr} does not reach the target system reliability R_{goal}, the next values of α, which will be exploited to generate the task schedule, are updated to $\alpha_1 = \alpha_0/2$ and $\alpha_2 = (\alpha_0 + 1)/2$, as is shown in Fig. 1. The search is stopped when the R_{curr} is greater than yet close enough to the R_{goal}; otherwise, repeat this process until the stop-condition is met. This approach can strikingly accelerate the generation of the desired task schedule by running the proposed algorithm with different α on multiple computing nodes.

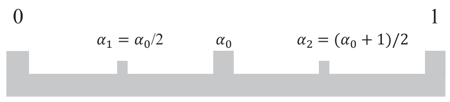

Fig. 1. An example of the binary search-based approach.

4.2. Efficient frequency selection based on the energy efficiency factor δ

It is evident that the energy efficiency is maximized if slack is allocated in a way that a unit slack generates the maximum energy saving. Let ΔS_i denote the slack allocated to task τ_i and ΔE_i denote the energy saved when the operating frequency is scaled from $f_{\max}$ down to $f(\tau_i)$, then $\delta_i = \frac{\Delta E_i}{\Delta S_i}$, referred to as the energy efficiency factor for task τ_i, is introduced to indicate the energy saving of a unit slack for task τ_i. The ΔS_i and ΔE_i are given by

$$\Delta S_i = \Psi_i(f(\tau_i)) - e_i, \tag{14}$$

and

$$\Delta E_i = E_i(f_{\max}) - E_i(f(\tau_i)), \tag{15}$$

respectively, where $\Psi_i(f(\tau_i))$ given in Eq. (4) is the task execution time at $f(\tau_i)$ considering fault recovery overhead and e_i is the task execution time at $f_{\max}$. The $E_i(f_{\max})$ and $E_i(f(\tau_i))$ are the energy consumptions of task τ_i at $f_{\max}$ and $f(\tau_i)$, respectively. The energy efficiency factor δ_i is in fact a function of the operating frequency $f(\tau_i)$, that is,

$$\delta_i(f(\tau_i)) = \frac{\Delta E_i(f(\tau_i))}{\Delta S_i(f(\tau_i))} = \frac{E_i(f_{\max}) - E_i(f(\tau_i))}{\Psi_i(f(\tau_i)) - e_i}. \tag{16}$$

It is clear that the energy efficiency factor $\delta_i(f(\tau_i))$ is optimized when the derivative of $\delta_i(f(\tau_i))$ with respect to $f(\tau_i)$, $\delta'_i(f(\tau_i))$, is set equal to zero. In this case, the expression

$$\Delta E'_i(f(\tau_i))\Delta S_i(f(\tau_i)) - \Delta E_i(f(\tau_i))\Delta S'_i(f(\tau_i)) = 0 \tag{17}$$

holds, where $\Delta E'_i(f(\tau_i))$ and $\Delta S'_i(f(\tau_i))$ are the derivatives of $\Delta E_i(f(\tau_i))$ and $\Delta S_i(f(\tau_i))$ respectively, which are given in Eqs. (15) and (14). The aforementioned optimization approach is well suited for continuous variable while general embedded processors only support discrete frequencies. In addition, the above higher order equation of complicated structure is difficult to solve. Thus, a heuristic that iteratively derives the sub-optimal operating frequency for each task is proposed, as is described in Algorithm 2.

Algorithm 2 iteratively derives the operating frequency for the N tasks in the task set. It takes as input the execution times of N tasks updated by the fault adaptation variable α. For each task, it calculates the optimal number of check points for the task at the current frequency using Eq. (1), computes the energy efficiency factor of the task at the current frequency using Eq. (16), and checks if the current energy efficiency factor is maximal. The frequency that results in the maximum energy efficiency factor is selected as the operating frequency of the current task.

Let $\delta_i(f_\ell)$ be the energy efficiency factor of task τ_i at the frequency f_ℓ, $\delta_{\max}$ be the maximum of $\delta_i(f_\ell)$ $(1 \le \ell \le L)$, *flag* be the index of the frequency in $\{f_1, f_2, \ldots, f_L\}$ that generates the maximum energy efficiency factor of the task, and S_{rem} be the remaining slack time of the system. Algorithm 2 operates as follows. Line 1 of the algorithm initializes the remaining slack time S_{rem} to $D - \sum_{i=1}^{N} e_i$. Lines 3–5 calculate the energy efficiency factor $\delta_i(f_\ell)$ of task τ_i at the frequency f_1 and initialize the $\delta_{\max}$ to $\delta_i(f_\ell)$. In lines 6–12, the energy efficiency factor $\delta_i(f_\ell)$ at the operating frequency f_ℓ for $2 \le \ell \le L$ is derived, and the frequency f_{flag} that generates the maximum energy efficiency factor $\delta_{\max}$ is picked. If the slack demand $\Delta S_i(f_{\text{flag}})$ required to scale down the frequency to f_{flag} is not greater than the available system slack S_{rem}, the required slack $\Delta S_i(f_{\text{flag}})$ is allocated to task τ_i, and the remaining system slack S_{rem} is updated to $S_{\text{rem}} - \Delta S_i(f_{\text{flag}})$. Otherwise, the task τ_i is set to run at the highest operating frequency $f_{\max}$. Repeat this process until all N tasks in the task set are examined once.

Algorithm 2 Select the energy efficient operating frequency for tasks in the task set Γ.

```
Require: the N tasks with execution times updated by $\alpha$
1:  initialize the remaining slack $S_{\text{rem}}$ to $D - \sum_{i=1}^{N} e_i$;
2:  for $i = 1$ to $N$ do
3:      calculate the optimal checkpoint number $N_{\text{opt},i}$ of task $\tau_i$ at the frequency $f_1$ using Eq. (1);
4:      derive the energy efficiency factor $\delta_i(f_1)$ of task $\tau_i$ at the frequency $f_1$ using Eq. (16);
5:      $\delta_{\max} = \delta_i(f_1), flag = 1$;
6:      for $\ell = 2$ to $L$ do
7:          calculate the optimal checkpoint number $N_{\text{opt},i}$ of task $\tau_i$ at the frequency $f_\ell$ using Eq. (1);
8:          derive the energy efficiency factor $\delta_i(f_\ell)$ of task $\tau_i$ at the frequency $f_\ell$ using Eq. (16);
9:          if $\delta_i(f_\ell) > \delta_{\max}$ then
10:             $\delta_{\max} = \delta_i(f_\ell)$, flag $= \ell$;
11:         end if
12:     end for
13:     if $\Delta S_i(f_{\text{flag}}) \le S_{\text{rem}}$ then
14:         $f(\tau_i) = f_{\text{flag}}$;
15:         allocate $\Delta S_i(f(\tau_i))$ for task $\tau_i$;
16:         update the remaining slack: $S_{\text{rem}} -= \Delta S_i(f(\tau_i))$;
17:     else
18:         $f(\tau_i) = f_{\max}$;
19:     end if
20: end for
```

The remaining system slack S_{rem} can further be utilized for thermal management of tasks in the task set after application of the frequency selection procedure, as is discussed in the next subsection. The time complexity of Algorithms 2 is $O(NL)$, where N is the number of tasks in the task set, and L is the processor-supported frequency levels.

4.3. Thermal-aware task sequencing based on the hot–cool task pairing

Unlike traditional cooling solutions, thermal-aware task sequencing utilizes temperature characteristics of tasks to reduce peak temperature of the processor without degrading system performance and incurring extra monetary expenses. The proposed thermal-aware task sequencing heuristic is based on the observation that the execution order of a hot task and a cool task has non-negligible impact on the peak temperature. It has been proven that the final temperature of tasks executing in the hot–cool order is lower than that of tasks executing in the cool–hot order (Yang et al., 2008). In other words, sequencing tasks in the order of hot–cool is more effective for thermal management. In this subsection, a static sequencing scheme based on the hot–cool task order is proposed to generate a thermal-aware task sequence for tasks in the task set. When compared to the online regulation approach presented in Yang et al. (2008), the proposed scheme does not necessitate the deployment of a runtime temperature monitor and thermal sensors. In addition, it achieves better thermal performance when combined with the frequency selection scheme presented in Subsection 4.2.

The proposed scheme iteratively derives the thermal-aware task sequence for task set Γ under the maximum temperature limit $T_{\max}$ by exploiting thermal characteristics of tasks. It maintains a hot queue with the hottest task at the head, a cool queue with the coolest task at the head, a target queue with all tasks ready for execution, and the remaining slack S_{rem} used for thermal management.

In each iteration, tasks in the target queue are dequeued from the queue, classified into hot or cool tasks, and enqueued onto the respective hot or cool queue. The target queue becomes empty. The hottest task in the hot queue and the coolest task in the cool queue are then dequeued and paired in the order of hot–cool, which is in turn pushed into the target queue as a newly formed task. If either the hot queue or the cool queue is empty, the tasks in the non-empty queue are appended to the target queue. The tasks in the target queue becomes ready for the next iteration of task pairing.

Since the task paring technique postulates opposite thermal characteristics of tasks, the iteration stops when the tasks in the target queue Q_{tgt} are of the same type (cool or hot). After thermal-aware task pairing and sequencing is finished, the remaining slack S_{rem} is partitioned and assigned to tasks in the target queue to further improve temperature profiles in the case where all tasks in Q_{tgt} are cool or to avoid thermal emergency in the case where all tasks in Q_{tgt} are hot. The peak temperature of all tasks in the target queue is then derived using Eq. (13). If the peak temperature is below the maximum temperature limit, the algorithm returns the task sequence in the target queue; otherwise, the algorithm exits and it cannot find a thermally feasible task schedule.

Let Q_{hot}, Q_{cool}, and Q_{tgt} denote the hot queue, cool queue, and target queue, respectively, and let T_{peak} denote the peak temperature of the target queue and len(Q_{tgt}) indicate the length of the target queue. The proposed task sequencing scheme given in Algorithm 3 operates as follows. Line 1 of the algorithm initializes the target queue by pushing all tasks into the queue. Lines 3–8 classify a task τ_i or a newly formed task τ_i^* into hot or cool task category based on the steady state temperature $T_{\text{std}}(\tau_i^*)$ of the task and insert the task into the corresponding queue. If $T_{\text{std}}(\tau_i^*) \ge T_{\max}$, the task τ_i^* is deemed to be a hot task and inserted into the hot queue Q_{hot}. Otherwise, it is a cool task and inserted into the cool queue Q_{cool}.

Algorithm 3 Derive the thermal-aware task sequence for task set Γ under $T_{\max}$ by exploiting thermal characteristics of tasks.

```
Require: maintain a hot queue $Q_{\text{hot}}$ with the hottest task at the head, a cool queue $Q_{\text{cool}}$ with the coolest task at the head, a target queue $Q_{\text{tgt}}$, and the remaining slack $S_{\text{rem}}$
1:  move all tasks in task set $\Gamma$ into the target queue $Q_{\text{tgt}}$;
2:  repeat
3:      for $i = 1$ to len($Q_{\text{tgt}}$) do
4:          calculate the steady state temperature $T_{\text{std}}$ for newly formed task $\tau_i^*$ using Eq. (11);{$\tau_i^* = \tau_i$ in the 1st iteration of repeat-until}
5:          classify $\tau_i^*$ into hot (cool) task based on $T_{\text{std}}(\tau_i^*)$;
6:          derive $T_{\text{start}}(\tau_i^*)$ if hot and $T_{\text{end}}(\tau_i^*)$ if cool;
7:          insert $\tau_i^*$ into hot queue $Q_{\text{hot}}$ (cool queue $Q_{\text{cool}}$);
8:      end for
9:      while ($Q_{\text{hot}} \ne$ NULL) or ($Q_{\text{cool}} \ne$ NULL) do
10:         if ($Q_{\text{hot}} \ne$ NULL) and ($Q_{\text{cool}} \ne$ NULL) then
11:             pair tasks at the head of $Q_{\text{hot}}$ and $Q_{\text{cool}}$ to form new task $\tau^*$;
12:             sequence the $\tau^*$ in the order of hot–cool;
13:             push the $\tau^*$ into the target queue $Q_{\text{tgt}}$;
14:             update the $Q_{\text{hot}}$ and $Q_{\text{cool}}$;
15:         else
16:             append tasks in the non-empty queue to $Q_{\text{tgt}}$;
17:         end if
18:     end while
19: until tasks in $Q_{\text{tgt}}$ are of the same type (cool or hot);
20: partition and allocate the remaining slack $S_{\text{rem}}$ to tasks in $Q_{\text{tgt}}$;
21: set the initial temperature $T_{\text{init}}$ of $Q_{\text{tgt}}$ to $T_{\text{amb}}$;
22: derive the peak temperature $T_{\text{peak}}$ for $\tau_i \in Q_{\text{tgt}}$ using Eq. (13);
23: if $T_{\text{peak}} \le T_{\max}$ then
24:     return the task sequence in the target queue $Q_{\text{tgt}}$;
25: else
26:     exit(1);{Exit when infeasible}
27: end if
```

The T_{start} is defined to be the start temperature of a hot task assuming the task ends its execution at the maximum temperature limit T_{max}, and the T_{end} is defined to be the end temperature of a cool task assuming the task starts its execution at the ambient temperature T_{amb}. The T_{start} and T_{end} are key characteristics of hot tasks and cool tasks, respectively. A lower T_{start} of a task indicates that the task is hotter. Similarly, a lower T_{end} of a task shows that the task is cooler. Tasks in the hot queue Q_{hot} are sorted in the increasing order of T_{start} and tasks in the cool queue Q_{cool} are sorted in the increasing order of T_{end}. In other words, the hot queue has the hottest task at its head while the cool queue has the coolest task at its head.

Lines 9–18 describe the procedure of task paring and sequencing. In each round of the iteration, if neither Q_{hot} nor Q_{cool} is empty, the task at the head of Q_{hot} and the task at the head of Q_{cool} are paired in the order of hot–cool, both tasks are removed from the Q_{hot} and Q_{cool}, and the pair is pushed into the target queue Q_{tgt}. Otherwise, the tasks in the non-empty queue are appended to the tail of the target queue. After task paring and sequencing is finished, the algorithm partitions and allocates the remaining slack S_{rem} to tasks in Q_{tgt} to further improve thermal profiles or avoid thermal emergency in line 20. Line 21 sets the initial temperature of the task sequence to the environmental temperature T_{amb}, that is, the initial temperature T_{init} of the task at the head of Q_{tgt} is set to T_{amb}. The initial temperature of the current task is the final temperature of its predecessor since tasks in the sequence execute continuously. The initial temperature of tasks in the sequence is thus derived. Considering the derived initial temperature and given steady state temperature of tasks in the sequence, the current temperature $T(t)$ of a task at the time instance t can be derived using Eq. (10). Given the $T(t)$, the peak temperature T_{peak} for $\tau_i \in Q_{tgt}$ can be derived using Eq. (13) (line 22). If the peak temperature T_{peak} is not greater than the maximum temperature limit T_{max}, the algorithm generates a task sequence that meets system thermal requirements (lines 23–24); otherwise, Algorithm 3 exits (lines 25–27). The time complexity of Algorithm 3 is $O(N^2)$, where N is the number of tasks in the task set.

4.4. Using the Monte Carlo simulation to obtain the reliability

In this section, the Monte Carlo simulation method is utilized to evaluate the reliability of the generated task schedule under transient fault occurrences of Poisson probability distribution. In general, one sample of the Monte Carlo simulation is obtained in two major steps. In the first step, based on the probability distribution of transient fault occurrences, the number of transient fault occurrences during the execution of the current task schedule is produced, and the execution time of each task is hence updated to include the overhead of fault recovery. In the second step, the feasibility of the current task schedule is verified. The current task schedule is feasible if all tasks in the task schedule complete their executions before the deadline D. Repeat the two steps to generate more than 10 000 Monte Carlo samples. The reliability R_{curr} of the current task schedule is calculated as the ratio of the number of feasible schedules to the total number of schedules. If R_{curr} is greater than but close enough to R_{goal}, the current task schedule is the desired schedule and the execution of Algorithm 1 exits. Otherwise, Algorithm 1 jumps from line 6 to line 3 and continues its execution.

5. Numerical results

Two sets of simulation experiments have been carried out to validate the proposed algorithms in energy efficiency and effectiveness of thermal management. In the first set of simulations, synthetic real-time tasks were generated to verify the proposed schemes while in the second set of simulations, the characteristics of real-life benchmarking tasks were utilized to validate the proposed algorithms. The proposed algorithms were implemented in C++, and the simulation was performed on a machine with Intel Dual-Core 3.0 GHz processor and 4 GB memory.

5.1. Simulations for synthetic real-time tasks

This subsection first describes experimental settings for the simulation, then compares the energy consumptions of the proposed scheme under three different scenarios of transient fault occurrences, and compares the peak temperature of the proposed scheme with that of the benchmarking schemes.

5.1.1. Experimental settings

The simulated processor is assumed to support five normalized discrete frequency levels $\{0.5, 0.65, 0.8, 0.9, 1.0\}$. The analytical formula, Eq. (8) developed by Liao et al. (2005), is utilized to compute leakage currents for temperatures ranging from 20 °C to 100 °C with the step size of 10 °C. The currents are then utilized to determine the curve-fitting constants C_1 and C_2 in Eq. (9). The effective switching capacitance C_{ef} is set equal to 1.0 (Zhu and Aydin, 2006).

The time overhead of frequency switching is typically on the order of tens of microseconds to tens of milliseconds. For instance, the worst case frequency switching time of ARM microprocessor core is in a range from 10 to 520 μs (Zhang and Chakrabarty, 2006). Hence, the time overhead of frequency switching O_{sw} is set to 100 μs. The worst case execution time of each task in a task set at the maximum frequency is randomly generated in the range of 10–40 ms and task sets of varying number of tasks (5, 10, 20, 40) are utilized to validate the proposed scheme. The common deadline D of tasks in a task set is set to $1.4\sum_{i=1}^{N} e_i$, where N is the number of tasks in the task set and e_i is the execution time of task τ_i at the maximum frequency f_{max}.

Checkpoint data are saved in DRAM, and the size of checkpoint is assumed to be 5KB so that the time overhead O_{ck} and power consumption of checkpointing are set to 0.4 ms and 400 mW, respectively (Zhang and Chakrabarty, 2006). Transient fault occurrences are assumed to follow the Poisson probability distribution. For the lumped RC model, the thermal resistance R and the thermal capacitance C are set equal to 1.83 °C/W and 0.0084 J/°C, respectively (Wang et al., 2012).

5.1.2. Comparison of the energy consumption

Three designing approaches are implemented and compared to evaluate the energy efficiency of the proposed scheme, which include the best case scenario, the worst case scenario, and the proposed stochastic approach. Task sets with varying sizes are executed under different levels of reliability requirements. Let E_{best}, E_{worst} and $E_{proposed}$ denote the energy consumed by a task set under the best case scenario of transient fault occurrences ($\alpha = 0$), the worst case scenario of transient fault occurrences ($\alpha = 1$) and the stochastic transient fault occurrences ($0 < \alpha < 1$), respectively. $E_{proposed}$ in fact indicates the energy consumed by the proposed algorithm. Then $E_{wp} = \frac{E_{worst}-E_{proposed}}{E_{worst}} \times 100\%$ denotes energy savings of the proposed algorithm ($0 < \alpha < 1$) as compared to the approach designing for the worst case scenario of transient fault occurrences (α = 1).

The energy consumptions of task sets with varying sizes under different levels of reliability requirements are shown in Table 1. Tasks in a task set are executed under three levels of reliability requirements ($R_{goal} = 0.7$, $R_{goal} = 0.8$, and $R_{goal} = 0.9$) for $\alpha = 0$, $\alpha = 1$, and $0 < \alpha < 1$, respectively. The proposed algorithm ($0 < \alpha < 1$) achieves energy savings of up to 17% when compared to the approach of designing for the worst case fault occurrences ($\alpha = 1$). For instance, for tasks of a given task set the size of which is 20 and the target reliability $R_{goal} = 0.7$, the proposed algorithm ($0 < \alpha < 1$) consumes 17.8% less energy when compared to the designing approach for the worst case fault occurrences ($\alpha = 1$). The better energy efficiency of the proposed algorithm benefits from the consideration of stochastic characteristics of transient fault occurrences. As compared to the designing approach

Table 1
The energy consumptions of task sets with varying sizes and different target reliability R_{goal}.

Task set size	$R_{goal} = 0.7$				$R_{goal} = 0.8$				$R_{goal} = 0.9$			
	$\alpha = 0$	$0 < \alpha < 1$		$\alpha = 1$	$\alpha = 0$	$0 < \alpha < 1$		$\alpha = 1$	$\alpha = 0$	$0 < \alpha < 1$		$\alpha = 1$
	E_{best}	$E_{proposed}$	E_{wp} (%)	E_{worst}	E_{best}	$E_{proposed}$	E_{wp} (%)	E_{worst}	E_{best}	$E_{proposed}$	E_{wp} (%)	E_{worst}
5	168.3	179.2	14.8	210.4	182.4	198.6	11.6	224.7	183.4	212.3	9.2	233.8
10	385.4	410.1	7.1	441.3	415.2	486.3	12.2	553.8	428.1	475	16.5	569.2
20	841.7	889.6	17.8	1082.5	866.1	1001.2	8.3	1092.1	859.7	962.3	15.2	1135.4
40	1567.2	1832.5	15.1	2159.2	1621.3	1815.3	17.4	2198.5	1672	1940.7	14.8	2278.9

for the worst case fault occurrences ($\alpha = 1$), the proposed algorithm with a fault adaptation variable ($0 < \alpha < 1$) has more slack to reduce energy consumption while satisfying the reliability constraint R_{goal}. In addition, Table 1 shows that task sets with higher target reliability consume more energy than task sets with lower target reliability. This increase in energy consumption is primarily due to the fault recovery overhead for higher reliability.

5.1.3. Comparison of the effectiveness of thermal management scheme

In the proposed scheme, the temperature profiles of tasks in task sets are improved by selecting an energy efficient operating frequency for each task and alternating the execution of a hot task and a cool task at the scaled operating frequency. Firstly, the proposed algorithm is compared with three benchmarking methods in terms of peak temperature to demonstrate its effectiveness of thermal management. The first benchmarking scheme, referred to as NOTM, does not utilize any thermal management techniques while the second benchmarking scheme, referred to as VSTM, utilizes greedy dynamic voltage/frequency scaling for thermal management. The third benchmarking scheme, referred to as SEQ (Zhang and Chatha, 2010), calculates an optimal initial temperature for a task sequence and utilizes thermal characteristics of cool tasks and sleep tasks in the task sequence to reduce temperatures.

Then two scenarios, that is, the worst case task sequence (Worst Sequence) and the best case task sequence (Best Sequence), and a state-of-the-art scheme, referred to as TSVS (Jayaseelan and Mitra, 2008), are implemented and compared to demonstrate the effectiveness of the proposed task sequencing algorithm in thermal management. To obtain the two task sequences, an exhaustive search of peak temperatures of all possible task sequences is conducted. The sequence with the lowest peak temperature is the best case task sequence, whereas the one that generates the highest peak temperature is the worst case task sequence. The TSVS (Jayaseelan and Mitra, 2008) scheme utilizes thermal characteristics of tasks to derive a task sequence in the alternate order of being cool–hot and allocates available slack time in a greedy way for voltage/frequency scaling. On the contrary, in the proposed algorithm, the task sequence is formed in the alternate order of being hot–cool to obtain a lower temperature, and slack time is allocated based on the tradeoff between energy efficiency and thermal management. The reliability requirement for the proposed algorithm is set to 0.99 in the comparison study.

Fig. 2 shows the peak temperatures of four different task sets with the size of 40. The maximum temperature limit T_{max} is assumed to be 70 °C. The initial temperature of each task set is assumed to equal the ambient temperature T_{amb}, which is set to 40 °C in this study. It has been demonstrated in Fig. 2 that the peak temperature of the proposed algorithm is much lower than that of the NOTM, VSTM, and SEQ (Zhang and Chatha, 2010) method. For example, for the task set Γ_4, when the system reliability requirement is 0.99, the proposed algorithm reduces the peak temperature by 14.8% as compared to the NOTM method, 7.5% as compared to the VSTM method, and 3.4% as compared to the SEQ (Zhang and Chatha, 2010) method. The NOTM method does not employ any thermal control techniques and is utilized as a baseline to show the highest efficiency

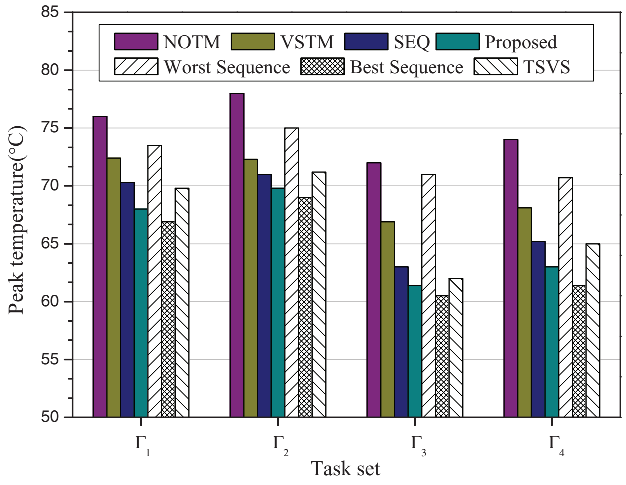

Fig. 2. Peak temperatures of four synthetic task sets under the thermal benchmarking methods NOTM, VSTM, SEQ (Zhang and Chatha, 2010), Worst Sequence, Best Sequence, TSVS (Jayaseelan and Mitra, 2008), and the proposed scheme ($R_{goal} = 0.99$).

of the proposed algorithm in reducing temperature. The proposed algorithm also outperforms the benchmarking method VSTM and SEQ (Zhang and Chatha, 2010) in thermal management since it not only executes the tasks at a scaled operating frequency, but also exploits thermal characteristics of cool tasks and slack time together to cool down hot tasks. Especially, the proposed algorithm alternates the execution of tasks in the order of being hot–cool to achieve a lower temperature.

Fig. 2 also plots the peak temperatures of the four task sets under the benchmarking algorithms Worst Sequence, Best Sequence, TSVS (Jayaseelan and Mitra, 2008), and the proposed scheme. When the system reliability requirement is 0.99, the peak temperature of the proposed scheme is very close to that of the Best Sequence within a small margin which varies from 0.8 °C to 1.6 °C, and can be up to 9.6 °C lower than that of the Worst Sequence. For example, for tasks in task set Γ_2, the peak temperature of the proposed scheme is 0.8 °C higher than that of the Best Sequence. For tasks in task set Γ_3, the peak temperature of the proposed scheme is 9.6 °C lower than that of the Worst Sequence. Moreover, the proposed scheme achieves better thermal profiles than that of the state-of-the-art scheme TSVS (Jayaseelan and Mitra, 2008). On average, the proposed scheme can reduce peak temperatures of tasks by 1.5 °C. This experiment has further demonstrated the effectiveness of the proposed thermal-ware task sequencing scheme in reducing peak temperature.

5.2. Simulations for real-life benchmarking tasks

This subsection validates the proposed scheme in terms of the energy consumption and effectiveness of the thermal management for real-life tasks. The TSVS (Jayaseelan and Mitra, 2008) and NOEM algorithms are utilized to benchmark the energy consumption of the proposed scheme, as is described in Subsection 5.1.2. The thermal effectiveness of the proposed scheme is compared with that of the

Table 2
Configuration of the simulated processor.

Issue width	2-width, partial, in-order issue
Pipeline	An 8-stages pipeline
L1 cache	32KB instruction and data caches
L2	Configurable cache size of 128KB to 1MB,
Cache	8-way set-associative cache structure
Memory	MMU and fully-associative I/D TLBs of 10 entries each
Branch	Dynamic branch prediction, 8-entry branch target address cache,
Predictor	Global history buffer, and 8-entry return stack

Table 3
The four constructed sets of real-life benchmarking tasks.

Task set	Tasks that comprise the task set
Γ_1	lame, djpeg, sha, ghostscript, blowfish, epic, gsm, dijkstra
Γ_2	epic, gsm, strsearch, adpcm, lame, mp3, sha, pegwit
Γ_3	susan, gsm, ghostscript, mp3, pegwit, dijkstra, epic, crc
Γ_4	crc, djpeg, dijkstra, lame, patricia, strsearch, sha, blowfish

Table 4
The energy consumptions of four task sets for the common target reliability goal $R_{goal} = 0.99$.

Task set	k	NOEM E_1	TSVS E_2	Proposed E_3	E_{13} (%)	E_{23} (%)
Γ_1	1	25.79	22.64	21.8	15.5	3.7
	2	25.79	23.45	22.43	13	4.4
	3	25.79	NF	23.02	10.7	–
Γ_2	1	21.84	20.07	18.7	14.4	6.7
	2	21.84	19.59	18.91	13.4	3.4
	3	21.84	20.32	19.33	11.5	4.9
Γ_3	1	23.52	20.48	19.4	17.5	5.2
	2	23.52	21.05	19.57	16.7	7
	3	23.52	NF	20.4	13.3	–
Γ_4	1	26.63	23.72	22.59	15.2	4.9
	2	26.63	22.89	22.46	15.7	1.9
	3	26.63	24.45	23.08	13.3	5.6

benchmarking algorithms NOTM, VSTM, SEQ (Zhang and Chatha, 2010), Worst Sequence, Best Sequence, and TSVS (Jayaseelan and Mitra, 2008), as is described in Subsection 5.2.3.

5.2.1. Experimental settings

The simulated processor is modeled based on the ARM Cortex A7 processor (ARM Cortex A7 Processor), which can assemble 1–4 cores. The number of core is set to 1 in the simulation. The configuration of the simulated processor is shown in Table 2. The maximal frequency supported by the processor is 1.6 GHz, and five different frequencies between 1.6 GHz and 800 MHz are exploited for dynamic voltage/frequency scaling. The thermal resistance and thermal capacitance are derived as 1.85 °C/W and 0.12 J/°C, respectively, by using the default HotSpot configuration and equations (Huang et al., 2007). The floorplan (ARM Cortex A7 Processor) of Cortex A7 processor (1-core) is taken by the thermal simulator HotSpot (Skadron et al., 2004) as input. In this floorplan, the entire die is divided into three sections. The data processing unit (dpu) dominates one section, which is located in one edge of the die. The central area of the die that is another section, has store buffer (stb), bus interface unit (biu), data cache unit (dcu), instruction cache unit (icu), main translation lookaside buffer (tlb), and prefetch unit (pfu). The L2 cache occupies the last section, which is located in the opposite edge to make the entire die as a square.

Fifteen benchmarking tasks from Mibench (Guthaus et al., 2001) and Mediabench (Lee et al., 1997), including the lame, djpeg, sha, ghostscript, blowfish, epic, gsm, dijkstra, strsearch, adpcm, mp3, pegwit, susan, crc, and patricia, are utilized to construct four task sets, as is shown in Table 3. The numbers of clock cycles of these tasks are in the range of $[4 \times 10^7, 6 \times 10^8]$. The architecture-level power simulator McPAT (Li et al., 2009) is utilized to obtain power consumptions of tasks in the table. It can model all three types of power dissipation, including dynamic, leakage, and short-circuit power, and provide a complete view of power consumptions.

5.2.2. Comparison of the energy consumption

Two benchmarking algorithms are implemented and compared with the proposed scheme in energy efficiency. The first one is the TSVS (Jayaseelan and Mitra, 2008) algorithm that combines task sequencing and voltage/frequency scaling for energy saving and temperature control. The second one, referred to as NOEM, is the algorithm that does not utilize any techniques for energy management. In other words, the algorithm operates at the highest processor frequency. Let E_1, E_2 and E_3 denote the energy consumption of a task set using the NOEM algorithm, the TSVS (Jayaseelan and Mitra, 2008) algorithm and the proposed algorithm, respectively. Then $E_{13} = \frac{E_1 - E_3}{E_1} \times 100\%$ denotes energy savings of the proposed scheme when compared to the NOEM algorithm, and $E_{23} = \frac{E_2 - E_3}{E_2} \times 100\%$ denotes energy savings of the proposed scheme when compared to the algorithm TSVS (Jayaseelan and Mitra, 2008).

In general, the reliability requirements of a system are given at requirements analysis stage. For instance, the reliability of a system may be no less than 0.99. Table 4 shows the energy consumptions of four benchmarking task sets for a common target reliability of 0.99. Tasks in a task set are executed under three levels of transient fault occurrences ($k = 1$, $k = 2$, and $k = 3$), where k is the worst case number of fault occurrences during the execution of a task. NF indicates that tasks in a task set cannot be feasibly scheduled under the given requirement of fault tolerance.

It has been shown in Table 4 that the proposed scheme outperforms the benchmarking algorithms NOEM and TSVS (Jayaseelan and Mitra, 2008) in terms of energy efficiency. The proposed scheme achieves energy savings of up to 17% and 7% when compared to the algorithms NOEM and TSVS (Jayaseelan and Mitra, 2008), respectively. For example, for tasks in task set Γ_3 with $k = 1$, the proposed scheme consumes 17.5% less energy when compared to the NOEM algorithm. For tasks in task set Γ_3 with $k = 2$, the energy consumption of the proposed scheme is 7% less than that of the TSVS (Jayaseelan and Mitra, 2008) algorithm. NOEM plays the role of a baseline to exhibit the maximum energy efficiency achieved by the proposed approach since it does not adopt any energy management technique. As compared to the TSVS (Jayaseelan and Mitra, 2008) that allocates the available slack in a greedy way for voltage/frequency scaling, the proposed approach consumes less energy since it selects the energy efficient operating frequency for each task. In addition, the proposed scheme is more resilient to fault occurrences. When the requirement of fault tolerance becomes high ($k = 3$), the TSVS (Jayaseelan and Mitra, 2008) algorithm may be infeasible, whereas the proposed scheme can feasibly schedule all tasks in the task set. The enhanced reliability of the proposed scheme is due to the consideration of stochastic property of transient fault occurrences and the reasonable allocation of slack. As expected, the proposed scheme consumes more energy when the number of faults to be tolerated is large. For instance, for tasks in task set Γ_1, the proposed scheme saves 15.5% energy over the NOEM algorithm with $k = 1$, but it saves only 10.7% energy for the case where $k = 3$. The increased energy consumption is mainly caused by the recovery overhead of growing faults.

5.2.3. Comparison of the effectiveness of thermal management scheme

The same set of benchmarking schemes adopted in Subsection 5.1.3, including NOTM, VSTM, SEQ (Zhang and Chatha, 2010), Worst Sequence, Best Sequence, and TSVS (Jayaseelan and Mitra, 2008), is implemented and compared to demonstrate the effectiveness of the proposed algorithm in thermal management.

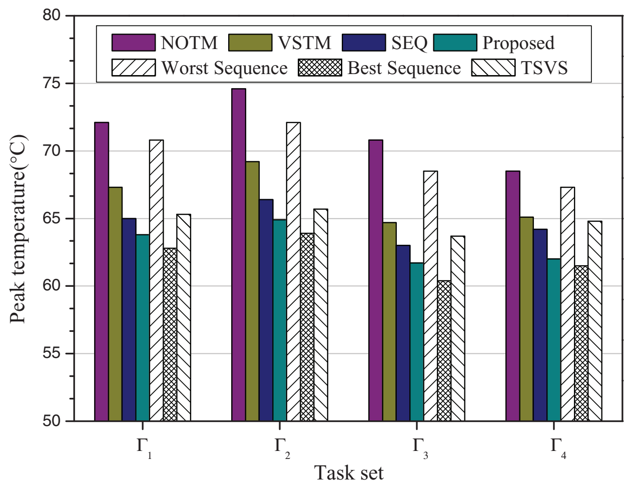

Fig. 3. Peak temperatures of four benchmarking task sets under the thermal benchmarking methods NOTM, VSTM, SEQ (Zhang and Chatha, 2010), Worst Sequence, Best Sequence, TSVS (Jayaseelan and Mitra, 2008), and the proposed scheme ($R_{\text{goal}} = 0.99$).

The peak temperatures of four benchmarking task sets under the aforementioned benchmarking methods and the proposed scheme are shown in Fig. 3. It has been demonstrated in Fig. 3 that the peak temperature of the proposed algorithm is much lower than that of the NOTM, VSTM, and SEQ (Zhang and Chatha, 2010) method. For example, for the task set Γ_2, when the system reliability requirement is 0.99, the proposed algorithm reduces the peak temperature by 13% as compared to the NOTM method, 6.2% as compared to the VSTM method, and 2.3% as compared to the SEQ (Zhang and Chatha, 2010) method.

Fig. 3 also plots the peak temperatures of the four benchmarking task sets under the benchmarking algorithms Worst Sequence, Best Sequence, TSVS (Jayaseelan and Mitra, 2008), and the proposed scheme. When the system reliability requirement is 0.99, the peak temperature of the proposed scheme is very close to that of the Best Sequence within a small margin which varies from 0.5 °C to 1.3 °C, and can be up to 7.2 °C lower than that of the Worst Sequence. For example, for tasks in task set Γ_4, the peak temperature of the proposed scheme is 0.5 °C higher than that of the Best Sequence. For tasks in task set Γ_2, the peak temperature of the proposed scheme is 7.2 °C lower than that of the Worst Sequence. In addition, when compared to the state-of-the-art scheme TSVS (Jayaseelan and Mitra, 2008), the proposed scheme exhibits better performance in thermal management. Specifically, the peak temperature of tasks is reduced by 1.8 °C on average using the proposed scheme.

Synthetic real-time tasks have been produced to validate the proposed thermal management scheme in Subsection 5.1.3. In this subsection, from the simulation results of real-life tasks shown above, the same conclusion can be drawn that the proposed scheme outperforms the benchmarking methods in thermal control. The analysis has been given in Subsection 5.1.3 such that it is omitted here for the sake of brevity.

6. Summary and future work

In this paper, the authors propose a stochastic real-time task scheduling algorithm to generate an energy efficient task schedule under constraints of the customer-defined target reliability and maximum peak temperature. The proposed algorithm first employs dynamic voltage/frequency scaling to reduce the energy consumption of tasks in the task set and selects the operating frequency that maximizes the energy saved per unit slack. It then classifies tasks into *hot* and *cool* tasks, sorts *hot* tasks and *cool* tasks in the *increasing* order of T_{start} and T_{end} respectively, and alternates the execution of *hot* tasks and *cool* tasks. The stochastic property of transient fault occurrences is handled by modeling the uncertainty using a fault adaptation variable α and accommodating task execution to transient fault occurrences by varying the α iteratively. Two sets of simulation results have demonstrated the effectiveness of the proposed algorithm in energy conservation and thermal management. In the simulations for synthetic tasks, the proposed algorithm achieves energy savings of up to 17.8% as compared to the approach of designing for the worst case transient fault occurrence. In the simulations for real-life benchmarking tasks, the peak temperature of the proposed algorithm is very close to that of the best case task sequence within a small margin which varies from 0.5 °C to 1.3 °C, and can be up to 7.2 °C lower than that of the worst case task sequence. Moreover, the peak temperature of the proposed algorithm is 1.8 °C lower than that of the benchmarking scheme on average.

The proposed task sequencing approach is designed for independent real-time tasks. If the target tasks are real-time tasks with data dependency or precedence constraints, task sequencing should be judiciously employed. In addition, due to the differences in actual task execution time, real-time systems can experience great variations of temperature and fault occurrence at run-time. The authors intend to extend their framework and address the two problems in future work.

Acknowledgments

The authors would like to thank the reviewers for their helpful suggestions. This work was supported in part by the Natural Science Foundation of Shanghai, China under the grant no. 12ZR1409200 and by the Scientific Research Foundation for Returned Scholars, Ministry of Education of China, under the grant no. 44420340.

References

ARM Cortex A7 Processor. http://www.arm.com/zh/products/processors/cortex-a/cortex-a7.php.
Bao, M., Andrei, A., Eles, P., Peng, Z., 2009. On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration. In: The Proceedings of the International Conference on Design Automation, pp. 490–495.
Berten, V., Chang, C., Kuo, T., 2008. Discrete frequency selection of frame-based stochastic real-time tasks. In: The Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 269–278.
Chandrakasan, A., Bowhill, W., Fox, F., 2000. Design of High-Performance Microprocessor Circuits. Wiley-IEEE Press.
Chen, J., Heusse, M., Urvoy-Keller, G., 2011. EFD: an efficient low-overhead scheduler. In: The Proceedings of the International Conference on Networking 2011, pp. 150–163.
Chen, J., Huang, K., Thiele, L., 2012. Dynamic frequency scaling schemes for heterogeneous clusters under quality of service requirements. Int. J. Inf. Sci. Eng. 28 (6), 1073–1090.
Coronel, J., Simo, J., 2012. High performance dynamic voltage/frequency scaling algorithm for real-time dynamic load management. J. Syst. Softw. 85 (4), 906–919.
Guthaus, M., Ringenberg, J., Ernst, D., Austin, T., Mudge, T., Brown, R., 2001. Mibench: a free, commercially representative embedded benchmark suite. In: The Proceedings of the International Workshop on Workload Characterization, pp. 3–14.
Han, C., Lin, K., Hou, C., 1996. Distance-constrained scheduling and its applications to real-time systems. IEEE Trans. Comput. 45 (7), 814–826.
Huang, H., Quan, G., Fan, J., Qiu, M., 2011. Throughput maximization for periodic real-time systems under the maximal temperature constraint. In: The Proceedings of the International Conference on Design Automation, pp. 363–368.
Huang, W., Sankaranarayanan, K., Ribando, R., Stan, M., Skadron, K., 2007. An Improved Block-Based Thermal Model in Hotspot 4.0 with Granularity Considerations. University of Virginia.
Jayaseelan, R., Mitra, T., 2008. Temperature aware task sequencing and voltage scaling. In: The Proceedings of the International Conference on Computer-Aided Design, pp. 618–623.
Lee, C., Potkonjak, M., Mangione-Smith, W., 1997. Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. In: The Proceedings of the International Symposium on Microarchitecture, pp. 330–335.
Li, S., Ahn, J., Strong, R., Brockman, J., Tullsen, D., Jouppi, N., 2009. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In: The Proceedings of the International Symposium on Microarchitecture, pp. 469–480.

Li, Z., Wang, L., Li, S., Ren, S., Quan, G., 2013. Reliability guaranteed energy-aware frame-based task set execution strategy for hard real-time systems. J. Syst. Softw. 86 (12), 3060–3070.
Liao, W., He, L., Lepak, K., 2005. Temperature and supply voltage aware performance and power modeling at microarchitecture level. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 24 (7), 1042–1053.
Liu, Y., Dick, R., Shang, L., Yang, H., 2007. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In: The Proceedings of the International Conference on Design, Automation and Test in Europe, pp. 1526–1531.
Meneses, E., Sarood, O., Kale, L., 2012. Assessing energy efficiency of fault tolerance protocols for hpc systems. In: The Proceedings of the International Symposium on Computer Architecture and High Performance Computing, pp. 35–42.
Nguyen, L., Cheng, A., 1996. An imprecise real-time image magnification algorithm. In: The Proceedings of the International Conference on Multimedia Systems.
Skadron, K., Stan, M., Sankaranarayanan, K., Huang, W., Velusamy, S., Tarjan, D., 2004. Temperature-aware microarchitecture: modeling and implementation. ACM Trans. Arch. Code Optimization 1 (1), 94–125.
Terzopoulos, G., Karatza, H., 2012. Maximizing performance and energy efficiency of a real-time heterogeneous 2-level grid system using dvs. In: The Proceedings of the International Conference on Distributed Simulation and Real Time Applications, pp. 185–191.
Terzopoulos, G., Karatza, H., 2014. Energy-efficient real-time heterogeneous cluster scheduling with node replacement due to failures. J. Supercomput. 68 (2), 867–889.
Ukhov, I., Bao, M., Eles, P., Peng, Z., 2012. Steady-state dynamic temperature analysis and reliability optimization for embedded multiprocessor systems. In: The Proceedings of the International Conference on Design Automation, pp. 197–204.
Wang, S., Chen, J., Shi, Z., Thiele, L., 2009. Energy-efficient speed scheduling for real-time tasks under thermal constraints. In: The Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 201–209.
Wang, Z., Ranka, S., Mishra, P., 2012. Temperature-aware task partitioning for real-time scheduling in embedded systems. In: The Proceedings of the International Conference on VLSI Design, pp. 161–166.
Wei, T., Chen, X., Hu, S., 2011. Reliability-driven energy efficient task scheduling for multiprocessor real-time systems. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 30 (10), 1569–1573.
Wei, T., Mishra, P., Wu, K., Liang, H., 2006. Online task-scheduling for fault-tolerant low-energy real-time systems. In: The Proceedings of the International Conference on Computer-Aided Design, pp. 522–527.
Wei, T., Mishra, P., Wu, K., Liang, H., 2008. Fixed-priority allocation and scheduling for energy-efficient fault-tolerance in hard real-time multiprocessor systems. IEEE Trans. Parallel Distrib. Syst. 19 (11), 1511–1526.
Wei, T., Mishra, P., Wu, K., Zhou, J., 2012. Quasi-static fault-tolerant scheduling schemes for energy-efficient hard real-time systems. J. Syst. Softw. 85 (6), 1386–1399.
Weste, N., Eshraghian, K., 1992. Principles of CMOS VLSI Design: A System Perspective. Addison-Wesley Publishing Company.
Xu, F., Liu, F., Jin, H., Vasilakos, A., 2014. Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions. Proc. IEEE 102 (1), 11–31.
Yang, C., Chen, J., Kuo, T., Thiele, L., 2009. An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems. In: The Proceedings of the International Conference on Design, Automation and Test in Europe, pp. 694–699.
Yang, J., Zhou, X., Chrobak, M., Zhang, Y., Jin, L., 2008. Dynamic thermal management through task scheduling. In: The Proceedings of the International Symposium on Performance Analysis of Systems and Software, pp. 191–201.
Zhang, S., Chatha, K., 2010. Thermal aware task sequencing on embedded processors. In: The Proceedings of the International Conference on Design Automation, pp. 585–590.
Zhang, Y., Chakrabarty, K., 2006. A unified approach for fault tolerance and dynamic power management in fixed-priority real-time embedded systems. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 25 (1), 111–125.
Zhu, D., Aydin, H., 2006. Energy management for real-time embedded systems with reliability requirements. In: The Proceedings of the International Conference on Computer-Aided Design, pp. 528–534.
Zhu, D., Melhem, R., Mosse, D., 2004. The effects of energy management on reliability in real-time embedded systems. In: The Proceedings of the International Conference on Computer-Aided Design, pp. 35–40.

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

Tongquan Wei received his Ph.D. degree in Electrical Engineering from Michigan Technological University in 2009. He is currently an Associate Professor in the Department of Computer Science and Technology at the East China Normal University. His research interests are in the areas of real-time systems, green and reliable computing, and parallel and distributed systems. He serves as a Regional Editor for *Journal of Circuits, Systems, and Computers* (World Scientific) since 2012. He also served as the Guest Editor of the *IEEE Transactions on Industrial Informatics Special Section on Building Automation, Smart Homes, and Communities*, and the *ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems for Energy-Efficient, Reliable, and Secure Smart Homes*. He is a member of the IEEE.