The Journal of Systems and Software 112 (2016) 1–10

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Reliability guaranteed energy minimization on mixed-criticality systems

Zheng Li [a,*], Chunhui Guo [b], Xiayu Hua [b], Shangping Ren [b]

[a] Western Illinois University, Macomb, IL 61455, USA
[b] Illinois Institute of Technology, Chicago, IL 60616, USA

ARTICLE INFO

Article history:
Received 4 February 2015
Revised 16 October 2015
Accepted 17 October 2015
Available online 27 October 2015

Keywords:
Mixed-criticality
Reliability
Energy

ABSTRACT

This paper studies the energy minimization problem in mixed-criticality systems that have stringent reliability and deadline constraints. We first analyze the resource demand of a mixed-criticality task set that has both reliability and deadline requirements. Based on the analysis, we present a heuristic task scheduling algorithm that minimizes system's energy consumption and at the same time also guarantees system's reliability and deadline constraints. Extensive experiments are conducted to evaluate and validate the performance of the proposed algorithm. The empirical results show that the algorithm further improves energy saving by up to 10% compared with the approaches proposed in our earlier work.

1. Introduction

For real-time embedded systems, to further reduce system cost, more and more tasks with different functionality and of different levels of criticality are integrated on the same hardware platform (Burns and Davis, 2013; Ekberg and Yi, 2014). Examples include recent unmanned aerial vehicles (UAV) which integrate the HI-criticality tasks such as flight-control tasks and LO-criticality tasks such as photo capturing tasks on the same platform (Barhorst et al., 2009). Though the mixed-criticality design paradigm reduces system cost, tasks of different criticalities compete for the shared resource and cause system's timing behaviors become less predictable.

For a safety-critical system, high-criticality tasks are more crucial to the entire system than low-criticality tasks. To ensure that HI-criticality tasks always meet their deadlines, two worst case execution times are set for each HI-criticality task, i.e., *worst case execution time by design* and a more pessimistic one, *worst case execution time by certification*. When tasks' actual execution time is no more than their designed worst case execution time, the system is considered operating under the LO-mode. However, if any task executes beyond this limit, the system changes to the HI-mode immediately to signal that a situation beyond designed behaviors has occurred and some actions need to take place. A mixed-criticality system is schedulable if the following two conditions are satisfied (Ekberg and Yi, 2012): 1) both LO-criticality and HI-criticality tasks are guaranteed to meet their deadlines under the LO-mode; and 2) HI-criticality tasks are also guaranteed to meet their deadlines under the HI-mode. Determining whether a given mixed-criticality system is schedulable has been proven to be NP-hard (Baruah et al., 2012a) and different heuristic approaches are developed to addressing the schedulability issue.

In addition to guarantee task deadlines, power/energy efficiency and reliability issues are also critical for real-time embedded systems. As more and more transistors are integrated into a single chip, operation power/energy consumption of the chip has increased exponentially. Dynamic Voltage and Frequency Scaling (DVFS) technique, which dynamically lowers down the supply voltage and working frequency, is widely used for power/energy management. However, existing work (Zhu et al., 2004; Niu and Xu, 2015) has shown that transient fault rate increases when the supply voltage on the chip scales down. In other words, lowering down system's supply voltage potentially degrades the system's reliability. Hence, minimizing system's energy consumption without sacrificing the reliability requirement is another design challenge.

In this paper, we study how to schedule a mixed-criticality task set to minimize system's energy consumption under the following constraints:

1. schedulability constraint: both HI-criticality and LO-criticality tasks are guaranteed to meet their deadlines under LO-mode, and HI-criticality tasks are guaranteed to meet their deadlines under HI-mode;
2. reliability constraint: both HI-criticality and LO-criticality tasks are guaranteed to meet their reliability constraints under LO-mode, and HI-criticality tasks are guaranteed to meet their reliability constraints under HI-mode.

* Corresponding author. Tel. +1312-6509688.
E-mail addresses: z-li2@wiu.edu (Z. Li), cguo13@iit.edu (C. Guo), xhua@iit.edu (X. Hua), ren@iit.edu (S. Ren).

http://dx.doi.org/10.1016/j.jss.2015.10.029
0164-1212/

The main contributions of this paper are three-fold:

1. theoretically analyze the resource demand of mixed-criticality task set under both reliability and schedulability constraints;
2. develop a heuristic search based frequency assignment (HSFA) algorithm that decides the lowest task execution frequency which guarantees both deadline and reliability constraints;
3. empirically evaluate and compare the energy saving performance of the HSFA algorithm with the state-of-the-art approaches in the literature.

The rest of the paper is organized as follows. We first discuss related work in Section 2 and then present the models and definitions the paper is based upon in Section 3. The research problem to be addressed in the paper is formally defined in Section 4. The theoretical foundations are established in Section 5. Based on the theoretical analysis, we present a heuristic search based frequency assignment algorithm in Section 6. Experimental results are discussed in Section 7 and finally we conclude our work in Section 8.

2. Related work

The study of mixed-criticality task set scheduling issue started in the recent years. Baruah and Vestal (2008) first proposed to apply earliest deadline first (EDF) scheduling theory in mixed-criticality task set scheduling. To ensure the schedulability of a mixed-criticality task set, the earliest deadline first with virtual deadline (EDF-VD) scheduling algorithm was proposed (Baruah et al., 2012b). The EDF-VD algorithm assigns HI-criticality tasks reduced deadlines to ensure that HI-criticality tasks are schedulable even if they overun their normal worst case execution time. Ekberg and Yi (2012); 2014) proposed a greedy approach by using the demand-bound function analysis to determine a task set's schedulability under EDF algorithm. Ekberg's greedy algorithm has a significant improvement over the EDF-VD, but it has higher time complexity. However, both Baruah's EDD-VD and Ekberg's greedy algorithm take the approach of terminating all LO-criticality tasks if any instance of HI-criticality task overruns its normal worst case execution time, i.e., when system enters into the HI-mode.

To provide a guaranteed minimum level of service to LO-criticality tasks when system enters into the HI-mode, Su and Zhu (2013) considered using elastic task models (Buttazzo et al., 1998) to increase LO-criticality tasks' period and hence reduce their competition against HI-criticality tasks but allow LO-criticality tasks to execute when possible. By noticing that postponing HI-criticality tasks' execution can promote earlier execution of LO-criticality tasks, Park and Kim (2011) developed a scheme called criticality based EDF which delays the execution of HI-criticality tasks as late as possible but without causing deadline violations.

Energy saving and reliability are two other major concerns for real-time embedded systems. For single criticality real-time systems, i.e. all tasks in the system have the same criticality, Zhu (2011) proposed a reliability-aware power management scheme which aims to minimize energy consumption while at the same time maintain system's reliability at the same level as if all tasks were executed with the highest processing frequency. Zhao et al. later improved the approach and developed a shared recovery technique that allows all tasks to share the same reserved recovery block, but only allows a single fault recovery during the entire task set execution (Zhao et al., 2009). Zhao et al. (2011) further extended the work and developed a generalized shared recovery technique which removed single recovery constraint and allow multiple fault recoveries by reserving multiple recovery blocks.

Recently, the study of energy saving and reliability issues in the context of mixed-criticality task sets has drawn increased attention. With the objective to minimize system's energy consumption and at the same time guarantee task deadlines, Huang et al. (2014) presented an approach by utilizing DVFS and EDF-VD techniques (Baruah et al., 2012b) to solve the problem. To satisfy system's reliability requirements, Axer et al. (2011) developed an approach to tolerating transient faults by duplicating high criticality tasks. Pathan (2014) also proposed a fixed-priority scheduling algorithm to tolerate transient faults in mixed-criticality systems.

As we can see, the aforementioned work treats reliability and energy consumption issues independently. However, system reliability and energy minimization are correlated when DVFS is used. In particular, reduced processing frequency reduces system energy consumption, but at the same time, reduced processing frequency also increases transient failure rate. Hence, they need to be addressed conjointly. In this paper, different with Huang et al. (2014) work that reliability constraint is not taken into consideration, we are to address the problem about how to schedule mixed-criticality task sets to minimize system's energy consumption while at the same time satisfying *both* deadline and reliability constraints.

3. Models and definitions

In this section, we introduce the models and definitions the research is based upon.

3.1. Models

3.1.1. Processor model

The processor is DVFS enabled with a finite set of available frequencies, i.e. $F = \{f_1, \ldots, f_q\}$. The frequency values in F are in a descending order with $f_1 = f_{\max}$ and $f_q = f_{\min}$. These frequencies are normalized with respected to $f_{\max}$, i.e., $f_{\max} = 1$.

3.1.2. Task model

In this paper, we make the same assumptions as in Ekberg and Yi (2014), Baruah et al. (2012b), i.e. there are two different criticality levels in a task set. In particular, for a given mixed-criticality task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$, each task τ_i is defined by a quadruple as $\tau_i = (L_i, C_i, T_i, D_i)$, where

- T_i is the task's period,
- D_i is the task's relative deadline and we assume $D_i \leq T_i$,
- $L_i \in \{\text{LO}, \text{HI}\}$ is the task's criticality level,
- $C_i = \{C_i(\text{LO}), C_i(\text{HI})\}$ is task's worst-case execution time and $C_i(\chi)$ is the worst case execution time at criticality level χ under maximum processing frequency $f_{\max}$. If τ_i is a HI-criticality task, $C_i(\text{LO}) \leq C_i(\text{HI})$; while if τ_i is a LO-criticality task, $C_i(\text{LO}) = C_i(\text{HI})$.

The LO-criticality subset and the HI-criticality subset are denoted as Γ_L and Γ_H, respectively.

3.1.3. Transient fault model

Although both permanent and transient faults may occur during task execution, transient faults are found more frequent than permanent faults (Niu and Xu, 2015; Guo et al., 2013). Hence, in this paper, we focus on transient faults. We adapt the same assumption as in the literature that the transient fault rate follows Poisson distribution with an average fault rate λ (Zhu, 2011; Zhao et al., 2011; Li et al., 2013). When a system is running under frequency f_i, the average transient fault rate is expressed as

$$\lambda(f_i) = \hat{\lambda}_0 10^{-\hat{d} f_i}, \tag{1}$$

where $\hat{\lambda}_0 = \lambda_0 10^{\frac{d}{1-f_{\min}}}$, $\hat{d} = \frac{d}{1-f_{\min}}$, and λ_0 is the average fault arrival rate when system running under the maximum frequency $f_{\max}$. The value $d(>0)$ is a system-dependent constant, which indicates the sensitivity of the system's fault arrival rate to system voltage and frequency scaling, the larger the d value is, the more sensitive the fault arrival rate to voltage and frequency scaling.

3.1.4. Fault recovery model

Backward fault recovery (Zhao et al., 2012) is used in our system to recover failed task instances. We assume fault detection is taken at the end of a task instance's execution. If a fault is detected, the failed task instance is re-executed at the maximum processing speed ($f_{\max}$) and the re-execution must be completed within the same instance period. The time cost of fault detection is considered to be part of task's worst case execution time.

3.1.5. System execution model

The system has two execution modes, i.e. the LO-mode and the HI-mode. Initially, the system executes at the LO-mode. In the LO-mode, each task τ_i can execute up to $\frac{C_i(\text{LO})}{f(\tau_i)}$ time units within each period under frequency $f(\tau_i)$. When any task τ_i executes beyond $\frac{C_i(\text{LO})}{f(\tau_i)}$ time within its period, the system switches to the HI-mode immediately. In the HI-mode, all LO-criticality tasks are removed from further execution and every HI-criticality task τ_i's maximum execution time changes to $\frac{C_i(\text{HI})}{f'(\tau_i)}$ if $f'(\tau_i)$ is task τ_i's processing frequency under the HI-mode.

It is worth pointing out that fault recovery time is *not* counted as task execution time. More specifically, when a task τ_i executes under frequency $f(\tau_i)$ for t_1 time units but fails, and it takes additional t_2 time to do the recovery, if $t_1 + t_2 \geq \frac{C_i(\text{LO})}{f(\tau_i)}$ but $t_1 < \frac{C_i(\text{LO})}{f(\tau_i)}$, the system still operates under the LO-mode and does not change to the HI-mode. In other words, system mode change is triggered only when $t_1 \geq \frac{C_i(\text{LO})}{f(\tau_i)}$.

3.1.6. Task-instance-level reliability model

Task-instance-level reliability is defined as the probability of completing a task instance without incurring errors caused by transient faults (Zhao et al., 2012). It is calculated as

$$r_i(f(\tau_i)) = e^{-\lambda(f(\tau_i))\cdot\frac{C_i}{f(\tau_i)}}, \tag{2}$$

where C_i and $f(\tau_i)$ are the task instance's execution time and system's working frequency, respectively.

3.1.7. Task-level reliability model

Task-level reliability is defined as the probability of completing all instances of task τ_i successfully within a hyperperiod of a given task set. It is denoted as R_i. If τ_i has k_i instances within a hyperperiod H, all these k_i instances are executed under $f(\tau_i)$, and at most δ_i among them are allowed to be recovered, then the reliability (R_i) of task τ_i can be calculated as Zhao et al. (2009)

$$\begin{aligned} &R_i(f(\tau_i), \delta_i, k_i) \\ &= (r_i(f(\tau_i)))^k + \sum_{j=1}^{\delta_i} \binom{k_i}{j} (1 - r_i(f(\tau_i)))^j (r_i(f(\tau_i)))^{k_i - j}. \end{aligned} \tag{3}$$

3.1.8. Energy model

We use the same power/energy model given in Zhu (2011), Zhao et al. (2011), Li et al. (2013). In particular, under the processing frequency $f(\tau_i)$, system's power consumption is represented as

$$P(f(\tau_i)) = P_s + hP_a = P_s + h(P_{\text{ind}} + C_{\text{ef}}(f(\tau_i))^\theta), \tag{4}$$

where P_s is the static power used to maintain the system in a standby state. The static power cannot be saved unless the system is turned off. P_a is the active power used when the system is in the working state. P_a has two components, frequency independent power (P_{ind}), such as the power used for memory and I/O operations, and frequency dependent power ($C_{\text{ef}} f^\theta$). Parameters C_{ef} and θ are system dependent constants and $\theta \geq 2$ (Aydin et al., 2006; Burd and Brodersen, 1995). Boolean parameter $h = 1$ indicates the system is in the working state, and $h = 0$ indicates the system is in the standby state.

We assume the system is always on and hence focus only on active power saving. As the energy consumption due to voltage and frequency scaling is independent of P_s, without loss of generality, we set $P_s = 0$. In addition, we also make the same assumption as in Aydin and Zhu (2009) that task execution time is linearly related to working frequency. Therefore, under the working frequency $f(\tau_i)$, task τ_i's execution time is $\frac{C_i}{f(\tau_i)}$ and the energy consumption within one period can be represented as

$$\begin{aligned} E(f(\tau_i), c_i) &= (P_{\text{ind}} + C_{\text{ef}} f(\tau_i)^\theta) \frac{c_i}{f(\tau_i)} \\ &= P_{\text{ind}} \frac{c_i}{f(\tau_i)} + C_{\text{ef}} C_i f(\tau_i)^{\theta-1}. \end{aligned} \tag{5}$$

From Eq. (5), it is clear that scaling down the processing frequency reduces frequency dependent energy, i.e. $C_{\text{ef}} C_i f(\tau_i)^{\theta-1}$. Hence, there is a balanced frequency, i.e. the *energy-efficient frequency* (f_{ee})—further scaling down the processing frequency below f_{ee} will increase total energy consumption. Early study (Zhu, 2011) concludes that

$$f_{\text{ee}} = \sqrt[\theta]{\frac{P_{\text{ind}}}{C_{\text{ef}}(\theta - 1)}}. \tag{6}$$

To simplify the discussion, we further assume $f_{\min} \geq f_{\text{ee}}$.

3.2. Definitions

We introduce the following definitions which will be used in the paper.

Task τ_i LO-mode utilization $u_L(\tau_i)$: $u_L(\tau_i) = \frac{C_i(\text{LO})}{T_i}$.

Task τ_i HI-mode utilization $u_H(\tau_i)$: $u_H(\tau_i) = \frac{C_i(\text{HO})}{T_i}$.

LO-criticality task set χ-mode utilization $U_\chi(\Gamma_L)$: $U_\chi(\Gamma_L) = \sum_{\tau_i \in \Gamma_L} u_\chi(\tau_i)$, where $\chi \in \{\text{HI}, \text{LO}\}$.

HI-criticality task set χ-mode utilization $U_\chi(\Gamma_H)$: $U_\chi(\Gamma_H) = \sum_{\tau_i \in \Gamma_H} u_\chi(\tau_i)$, where $\chi \in \{\text{HI}, \text{LO}\}$.

4. Problem formulation

Based on the defined models, we define the problem this paper is to address as below:

Problem 1. Given a DVFS enabled processor with q different processing frequencies, i.e. $F = \{f_1, \ldots, f_q\}$, where $f_1 = f_{\max}$, $f_q = f_{\min}$ and $f_i > f_j$ if $i < j$, and a mixed-criticality task set $\Gamma = \{\Gamma_L, \Gamma_H\}$, develop a scheduling algorithm that minimizes system's energy consumption under the following two constraints:

1. schedulability constraint: HI-criticality and LO-criticality tasks are both guaranteed to meet their deadlines under the LO-mode, and HI-criticality tasks are guaranteed to meet their deadlines under the HI-mode;
2. reliability constraint: HI-criticality and LO-criticality tasks are both guaranteed to meet their reliability constraint under the LO-mode, and HI-criticality tasks are guaranteed to meet their reliability constraints under the HI-mode.

The EDF-VD algorithm (Baruah et al., 2012b; Zhang et al., 2014; Ekberg and Yi, 2012) is a widely used scheduling algorithm for mixed-criticality task sets. Under the EDF-VD algorithm, when system operates under the LO-mode, tasks are scheduled using EDF algorithm with reduced deadlines which is also called virtual deadlines (VD). However, when the system changes to the HI-mode, all LO-criticality tasks are removed from further execution, and each HI-criticality task's deadline is reset to its original deadline to guarantee that even in the overrun situation tasks can still meet their deadlines. In this paper, we assume that the EDF-VD algorithm is used to schedule the mixed-criticality task sets. Due to the improbability of system switching to the HI-mode (Huang et al., 2014), for deadline guarantee and

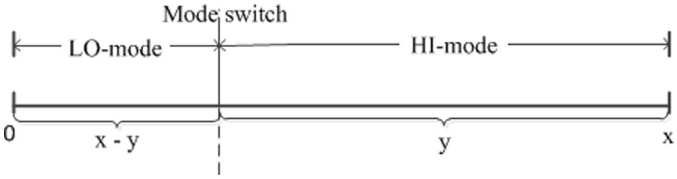

Fig. 1. Illustration of mixed-criticality task execution.

energy saving purposes, we run tasks at $f_{\max}$ under the HI-mode and use lower frequencies when the system executes under the LO-mode.

However, in order to actually implement the above task execution strategy, we have to answer under *what frequency* $f(\tau_i)$ and with *what virtual deadline* VD_i each task τ_i should be executed when the system is in the LO-mode so that the energy consumption is minimized and system schedulability and reliability constraints are guaranteed. We address the question in three steps: first, determine whether the given task set working frequency $f(\tau_i)$ and virtual deadline VD_i satisfy system schedulability and reliability constraints; second, decide tasks' virtual deadline VD_i under given tasks' working frequency $f(\tau_i)$ that guarantees system schedulability and reliability constraints; and third, determine tasks' best working frequency $f(\tau_i)$ that minimizes energy consumption without violating system's schedulability and reliability constraints.

5. Theoretical foundations

In this section, we establish the theoretical foundation that determines whether a mixed-criticality system's reliability and schedulability constraints are satisfied when each task τ_i's processing frequency $f(\tau_i) \le f_{\max}$ and its virtual deadline $\text{VD}_i \le D_i$ are given.

5.1. Satisfying reliability constraint

According to the reliability model, i.e. Eq. (3), scaling down task's working frequency increases transient fault rate and hence more recoveries are required to meet the reliability constraint. For a periodic task τ_i, if the task-level reliability constraint is R_i^c and the processing frequency under LO-mode is $f(\tau_i)$, according to Eq. (3), the number of recoveries (δ_i^L) needed under the LO-mode to guarantee the system reliability constraint must satisfy the following Inequation:

$$r_i(f(\tau_i))^k + \sum_{j=1}^{\delta_i^L} \binom{k_i}{j} (1 - r_i(f(\tau_i)))^j r_i(f(\tau_i))^{k_i - j} \ge R_i^c. \tag{7}$$

where

$$r_i(f(\tau_i)) = e^{-\lambda(f(\tau_i)) \cdot \frac{C_i(\text{LO})}{f(\tau_i)}}.$$

The value of δ_i^L can be obtained by solving Inequation (7). Similarly, the number of recoveries (δ_i^H) needed under HI-mode can be obtained by replacing $C_i(\text{LO})$ with $C_i(\text{HI})$ and $f(\tau_i)$ with $f_{\max}$ in Inequation (7).

The reliability constraint of task τ_i is satisfied if at least δ_i^L and δ_i^H number of task instances are allowed to be recovered under the LO-mode and the HI-mode, respectively. The next question is, with the amount of recoveries, whether the schedulability constraint can also be satisfied?

5.2. Satisfying schedulability constraint

Based on the resource demand and supply analysis, a mixed-criticality task set is schedulable if the resource demand is no larger than the resource supply when system operates under both the LO-mode and HI-mode. In particularly, as illustrated in Fig. 1, if the total execution duration is x and system changes from the LO-mode to the HI-mode at time $x - y$, then the task set is schedulable if (Zhang et al., 2014)

$$\forall x, y \in [0, H] \wedge x \ge y : \sum_{\tau_i \in \Gamma} dbf_i^{\text{LO}}(x - y) \le x - y, \tag{8}$$

and

$$\forall x, y \in [0, H] \wedge x \ge y : \sum_{\tau_i \in \Gamma_H} dbf_i^{\text{HI}}(x, y) \le y. \tag{9}$$

Among which, $dbf_i^{\text{LO}}(x - y)$ is the resource demand of the task τ_i within the LO-mode execution duration $x - y$ and $dbf_i^{\text{HI}}(x, y)$ is the resource demand within the HI-mode duration y among the total execution duration x.

In the following, we discuss how to derive task's LO-mode and HI-mode resource demand to meet the schedulability constraint.

5.2.1. LO-mode resource demand analysis

By definition, when a mixed-criticality system operates under the LO-mode, both LO-criticality and HI-criticality tasks should be guaranteed to meet their deadlines. In addition, to satisfy the reliability constraint, at least δ_i^L recoveries are required for each task τ_i within a given task set's hyperperiod. Therefore, to execute a mixed-criticality task set that has the reliability constraint, both the resource demand for task execution and fault recovery need to be taken into consideration.

Before providing task's resource demand under LO-mode, we first state a lemma (Koren and Shasha, 1995) based on which to calculate the resource demand for task recoveries as follows:

Lemma 1. *Given a task set* $\Gamma = \{\tau_1, \ldots, \tau_n\}$ *and their maximal allowed number of recoveries* $\{\delta_1, ..., \delta_n\}$, *the task set is schedulable if and only if all the deadlines are satisfied under the worst case scenario, i.e., when the first* δ_i *instances of all task* $\tau_i \in \Gamma$ *need to be recovered (Koren and Shasha, 1995).*

Based on Lemma 1, we obtain the task set resource demand when the system operates under the LO-mode. It is given by Theorem 1.

Theorem 1. *For a given periodic task* $\tau_i = (L_i, C_i, T_i, D_i)$, *when system operates under the LO-mode, the task's working frequency is* $f(\tau_i)$, *the recovery allowance within one hyper-period H is* δ_i^L *and the virtual deadline is set as* VD_i, *then the resource demand of task* τ_i *within LO-mode duration* $x - y (x \ge y)$ *can be calculated as*

$$dbf_i^{\text{LO}}(x - y) = \begin{cases} N_S \cdot \left(1 + \frac{f_{\max}}{f(\tau_i)}\right) \cdot C_i(\text{LO}), & \text{if } x - y \le \delta_i^L \cdot T_i; \\ \left(\delta_i^L + (\delta_i^L + N_L) \cdot \frac{f_{\max}}{f(\tau_i)}\right) \cdot C_i(\text{LO}), & \text{otherwise.} \end{cases} \tag{10}$$

where

$$N_S = \left\lfloor \frac{x - y - \text{VD}_i}{T_i} \right\rfloor + 1,$$

and

$$N_L = \left\lfloor \frac{x - y - \delta_i^L \cdot T_i - \text{VD}_i}{T_i} \right\rfloor + 1.$$

Proof. According to Lemma 1, the case that all the first δ_i^L instances of each task $\tau_i \in \Gamma$ needs to be recovered has the highest resource demand and we focus on the resource demand analysis under this case.

When $x - y \le \delta_i^L \cdot T_i$, as illustrated in Fig. 2, each instance needs one recovery running under $f_{\max}$ which takes $C_i(\text{LO})$ time units. In addition, finishing each instance at frequency $f(\tau_i)$ also costs $\frac{f_{\max}}{f(\tau_i)} \cdot C_i(\text{LO})$ time units. Sum them together, each instance of task τ_i demands $(1 + \frac{f_{\max}}{f(\tau_i)}) \cdot C_i(\text{LO})$ time units. For a given LO-mode duration $x - y$, at most $\lfloor \frac{x - y - \text{VD}_i}{T_i} \rfloor + 1$ task instances must be finished within

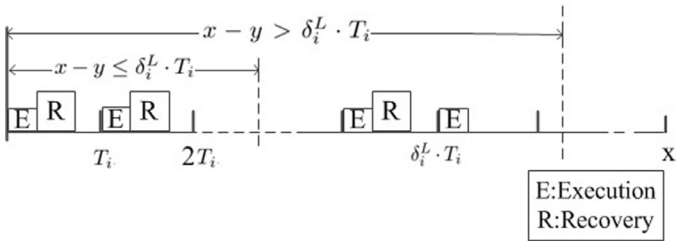

Fig. 2. Illustration of LO-mode task execution and recovery.

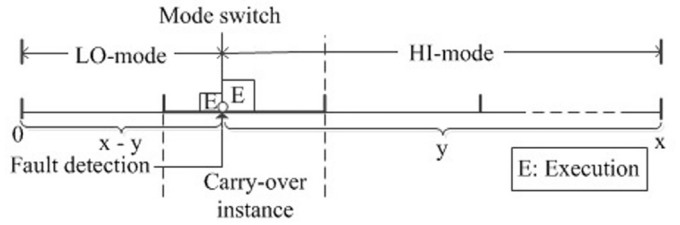

Fig. 3. Illusration of carry-over task instance.

the LO-mode duration $x - y$. Therefore, if $x - y \le \delta_i^L \cdot T_i$, the resource demand of task τ_i is at most

$$\left(\left\lfloor \frac{x - y - \mathrm{VD}_i}{T_i} \right\rfloor + 1\right) \cdot \left(1 + \frac{f_{\max}}{f(\tau_i)}\right) \cdot C_i(\mathrm{LO}).$$

When $x - y > \delta_i^L \cdot T_i$, each of the first δ_i^L instances needs to recover once and hence the total resource demand of the first δ_i^L instances is

$$\delta_i^L \cdot \left(1 + \frac{f_{\max}}{f(\tau_i)}\right) \cdot C_i(\mathrm{LO}). \tag{11}$$

Since total $\lfloor \frac{x-y-\mathrm{VD}_i}{T_i} \rfloor + 1$ task instances must finish within the LO-mode duration $x - y$, each of the remaining $\lfloor \frac{x-y-\mathrm{VD}_i}{T_i} \rfloor + 1 - \delta_i^L$ instances only need to execute once without recovery. Therefore, the resource demand of these remaining instances is

$$\left(\left\lfloor \frac{x - y - \delta_i^L \cdot T_i - \mathrm{VD}_i}{T_i} \right\rfloor + 1\right) \cdot \frac{f_{\max}}{f(\tau_i)} \cdot C_i(\mathrm{LO}). \tag{12}$$

Let $N_L = \lfloor \frac{x-y-\delta_i^L \cdot T_i - \mathrm{VD}_i}{T_i} \rfloor + 1$, and then the total resource demand can be represented as

$$\left(\delta_i^L + (\delta_i^L + N_L) \cdot \frac{f_{\max}}{f(\tau_i)}\right) \cdot C_i(\mathrm{LO}).$$

□

5.2.2. HI-mode resource demand analysis

When system changes to the HI-mode, all the LO-criticality tasks are removed from further execution and all the HI-criticality tasks run under $f_{\max}$. Therefore, the resource demand comes only from HI-criticality tasks. Within the HI-mode execution duration of y, there are total $\lfloor \frac{y-D_i}{T_i} \rfloor$ task instances that are yet to be started and completed.

In addition to the yet to be started task instances, it is possible that at the mode switch point, there is a carry-over task instance of which part of its execution is completed in the LO-mode under a scaled down frequency $f(\tau_i) < f_{\max}$ and it may already encounter a transient fault. The scenario is depicted in Fig. 3.

Initially, fault detection is only taken at the end of each task instance's execution, which means the fault of this carry-over task instance can only be detected after its remaining part is finished under the HI-mode. Resource would be wasted if a failed task is further executed. To avoid such resource waste, for carry-over task instances, in addition to taking fault detection at its end, we also assume the execution correctness of carry-over tasks is checked at the mode switch point. If a task instance is detected failed, we re-execute the task instance from the beginning in the HI-mode; otherwise, continue to finish its remaining part.

Hence, in the worst-case scenario, all carry-over task instances need to be re-executed from the beginning and hence the resource demand of a HI-criticality task τ_i with the HI-mode duration y is

$$\left(\left\lfloor \frac{y - D_i}{T_i} \right\rfloor + 1\right) \cdot C_i(\mathrm{HI}),$$

and the total resource demand of all HI-criticality tasks within the HI-mode execution duration of y is

$$\sum_{\tau_i \in \Gamma_H} \left(\left\lfloor \frac{y - D_i}{T_i} \right\rfloor + 1\right) \cdot C_i(\mathrm{HI}). \tag{13}$$

However, using (13) to calculate the HI-mode resource demand is too pessimistic as it assumes every HI-criticality task has a carry-over instance which may not be the case. For instance, if the mode switch happens after a task instance's virtual deadline VD_i but before its deadline D_i, i.e. $(x - y) \mod T_i > \mathrm{VD}_i$, according to virtual deadline definition, this task does not have a carry-over instance.

The following theorem provides more accurate calculation of task resource demand under the HI-mode.

Theorem 2. *For a given periodic task $\tau_i = (L_i, C_i, T_i, D_i)$, the virtual deadline is VD_i, the working frequency and recovery allowance under HI-mode are $f_{\max}$ and δ_i^H within one hyper-period H, respectively, then the resource demand of task τ_i within HI-mode execution duration y among the total execution duration x can be represented as*

$$dbf_i^{\mathrm{HI}}(x, y) = (n_i(x, y) + \min\{n_i(x, y), \delta_i^H\}) \cdot C_i(\mathrm{HI}), \tag{14}$$

where

$$n_i(x, y) = \begin{cases} \max\left\{\left\lfloor \frac{y-D_i+\varepsilon_i}{T_i} \right\rfloor, 0\right\}, & \text{if } \varepsilon_i > \mathrm{VD}_i; \\ \left\lfloor \frac{y-D_i+\varepsilon_i}{T_i} \right\rfloor + 1, & \text{if } \varepsilon_i \le \mathrm{VD}_i. \end{cases} \tag{15}$$

and $\varepsilon_i = (x - y) \mod T_i$.

Proof. If $\varepsilon_i > \mathrm{VD}_i$, at the mode switch point, no carry-over task instance exists and all the released task τ_i instances are finished, and the next task instance will be released at $T_i - \varepsilon_i$ time units after the mode switch point. Hence, within the HI-mode duration y, the number of task instances that must be finished is

$$n_i(x, y) = \max\left\{\left\lfloor \frac{y - (T_i - \varepsilon_i) - D_i}{T_i} \right\rfloor + 1, 0\right\},$$

which can be re-written as

$$n_i(x, y) = \max\left\{\left\lfloor \frac{y - D_i + \varepsilon_i}{T_i} \right\rfloor, 0\right\}.$$

When $\varepsilon_i \le \mathrm{VD}_i$, task τ_i may have a carry-over instance. Under this scenario, there are three possible cases:

Case (1) $y < D_i - \varepsilon_i$: since under the HI-mode, deadline of a HI-criticality task is D_i, $y < D_i - \varepsilon_i$ means this carry-over instance does not reach its deadline, hence, $n_i(x, y) = 0$;

Case (2) $D_i - \varepsilon_i \le y \le T_i - \varepsilon_i$: the carry-over instance has reached its deadline but no new task instance is released, hence, we only have the carry-over task instance to be finished. Therefore, $n_i(x, y) = 1$;

Case (3) $y > T_i - \varepsilon_i$: the carry-over instance has reached its deadline and new released task instances also need to be finished. In this case,

$$n_i(x, y) = 1 + \left\lfloor \frac{y - (T_i - \varepsilon_i) - D_i}{T_i} \right\rfloor + 1 = \left\lfloor \frac{y - D_i + \varepsilon_i}{T_i} \right\rfloor + 1. \tag{16}$$

In fact, Eq. (16) generalizes all the aforementioned three cases.

To satisfy the reliability constraint, each task τ_i is allowed δ_i^H number of instances to be recovered within one hyperperiod. According to Lemma 1, the worst case from resource demand perspective is when the first δ_i^H instances of each task τ_i need to be recovered. In addition, under the HI-mode, task recovery under $f_{\max}$ may take up to $C_i(\text{HI})$ time. Hence, the resource demand of task τ_i under the HI-mode for the duration of y is

$$(n_i(x,y) + \min\{n_i(x,y), \delta_i^H\}) \cdot C_i(\text{HI}).$$

□

Based on resource demands under LO-mode, i.e. Eq. (10) and HI-mode, i.e. Eq. (14), we have the following lemma:

Lemma 2. *For a given periodic task τ_i with period as T_i, when system operates under the LO-mode, the task's working frequency is $f(\tau_i)$ and the recovery allowance within one hyper-period is δ_i^L, the task's LO-mode resource demand will be monotonically nondecreasing when decreasing the assigned virtual deadline VD_i.*

Proof. Task's LO-mode resource demand can be calculated by Eq. (10), which can be re-written as

$$dbf_i^{\text{LO}}(x-y) = N_S \cdot \frac{f_{\max}}{f(\tau_i)} \cdot C_i(\text{LO}) + \min\{N_S, \delta_i^L\} \cdot C_i(\text{LO}),$$

where x is the total execution duration and $x-y$ is the LO-mode execution duration. It is not hard to find that, under given $x-y>0$, reducing VD_i may increase but never decrease the value of $N_S = \lfloor \frac{x-y-\text{VD}_i}{T_i} \rfloor$, and hence $dbf_i^{\text{LO}}(x-y)$ will never be decreased. □

Lemma 3. For a given periodic task τ_i with period as T_i and deadline as D_i, the recovery allowance within one hyper-period under HI-mode is δ_i^H, then the task's HI-mode resource demand will be *monotonically nonincreasing* when decreasing the assigned virtual deadline VD_i.

Proof. The HI-mode resource demand can be calculated using Eq. (14), i.e.

$$dbf_i^{\text{HI}}(x,y) = (n_i(x,y) + \min\{n_i(x,y), \delta_i^H\}) \cdot C_i(\text{HI}),$$

where x is the total execution duration and y is the HI-mode execution duration. $n_i(x, y)$ can be calculated based on Eq. (15). From Eq. (15), it is clearly that $n_i(x, y)$ may be decreased but never be increased by reducing the value of VD_i. Therefore, $dbf_i^{\text{HI}}(x,y)$ is monotonically nonincreasing when decreasing VD_i. □

6. Heuristic search based approach to minimizing energy consumption

In Section 5, Theorem 1 and Theorem 2 are established to check whether a mixed-criticality task set with the reliability constraint is schedulable under given task's virtual deadline and processing frequency assignments. In this section, we first present a virtual deadline assignment strategy and based on the virtual deadline assignment, we then give a processing frequency to task assignment algorithm that satisfies both schedulability and reliability constraints and also results in the least energy consumption.

6.1. Virtual deadline assignment

Given a task τ_i's virtual deadline as VD_i, Inequation (8) and (9) together are able to check whether the task set is schedulable. The question is how to decide the virtual deadline VD_i for each task that satisfiesEq. (8) and (9).

A closer look reveals that satisfying Inequation (8) and (9) is equivalent to satisfy the following Inequation (17) and (18) for $\forall x \in [0, H]$, respectively, i.e.

$$\sum_{\tau_i \in \Gamma} dbf_i^{\text{LO}}(x) \le x, \tag{17}$$

and

$$\forall y \in [0,x] : \sum_{\tau_i \in \Gamma_H} dbf_i^{\text{HI}}(x,y) \le y. \tag{18}$$

Furthermore, based on the Lemma 3, decreasing VD_i potentially decreases task's resource demand under the HI-mode and hence Inequation (18) is more likely to be satisfied. However, Lemma 2 indicates that the task's resource demand under the LO-mode increases when the task's VD_i decreases, which causes Inequation (18) is more likely to be violated. Based on these observations, we propose a greedy virtual deadline assignment algorithm. The major steps of the algorithm are highlighted as follows:

1. Initialization step: we set $x := 0$ and each HI-criticality task τ_i's virtual deadline $\text{VD}_i := D_i$. In addition, we define a new task set which contains tasks whose deadlines can be reduced. We call the new task set a *reducible* task set Ω, and it is initialized to be Γ_H, i.e. $\Omega := \Gamma_H$. In other words, initially every HI-criticality task's deadline is allowed to be reduced.
2. If Inequation (18) is violated: if $\Omega \neq \Phi$, choose a task $\tau \in \Omega$ whose $dbf^{\text{HI}}(x,y)$ decreases the most when its virtual deadline is reduced by 1 time unit and reduce the task's virtual deadline by 1 time unit and reset $x := 0$; if $\Omega = \Phi$, return FAILURE.
3. If Inequation (17) is violated: undo the last task τ_j's virtual deadline change, update $\Omega := \Omega - \{\tau_j\}$ and reset $x := 0$; if there is no change to undo, return FAILURE.
4. Otherwise, increase x by one and repeat step 2)–3) until $x = H$ where H is the hyperperiod of the task set and return SUCCESS.

It is worth pointing out that the algorithm returns SUCCESS only if both Inequation (18) and Inequation (17) are satisfied for every x within range [0, H]. The details of our greedy virtual deadline assignment (GVDA) algorithm are given in Algorithm 1 . Lines 1–4 are for initialization, lines 8–16 are the Step 2) and lines 17–25 are the Step 3).

6.2. Task execution frequency assignment

For a given frequency to task assignment, the GVDA algorithm can determine if there exists a valid virtual deadline assignment which guarantees both reliability and schedulability constraints. However, there may be multiple task execution frequencies satisfying the schedulability and reliability constraints. Our next step is to decide a valid task execution frequency assignment that minimizes energy consumption of executing the task set.

The proposed heuristic search based approach has the following major steps:

1. Initialization step: all the tasks are assigned $f_{\max}$ as their execution frequencies;
2. Choose one task to scale down its frequency by one level but without violating the reliability and schedulability constraints based on Inequation (8) and (9);
3. Repeat step 2) until no task's execution frequency can be scaled down further.

In step 2), there may be more than one task's processing frequency can be scaled down without violating the reliability and schedulability constraints, which task is chosen to scale down its execution frequency impacts total energy consumption.

Executing a task under a lower frequency reduces energy consumption, but the task takes longer time to complete under a reduced execution frequency. Furthermore, lowering down task's execution frequency increases system transient fault rate which in turn

Algorithm 1: GVDA(Γ, $\{f(\tau_1), \ldots, f(\tau_n)\}$, $\{R_1^c, \ldots, R_n^c\}$).

```
1  ∀τ_i ∈ Γ_H : set VD_i := D_i ;
2  set Ω := Γ_H, k := −1 and final := FALSE;
3  calculate task set's hyper-period as H;
4  calculate δ_i^L and δ_i^H based on Inequation (7);
5  while final = FALSE do
6      final := TRUE;
7      for x := 0; x ≤ H; x++ do
8          if Inequation (18) is not satisfied then
9              if Ω is empty then
10                 return FAILURE;
11             end
12             find the τ_j ∈ Ω that decreases dbf_i^HI(x, y) the most if
               changing VD_j := VD_j − 1;
13             set VD_j := VD_j − 1 and k := j;
14             set final := FALSE;
15             break ;
16         end
17         if Inequation (17) is not satisfied then
18             if k = −1 then
19                 return FAILURE;
20             end
21             set VD_k := VD_k + 1 and k := −1;
22             update Ω := Ω − {τ_k};
23             set final := FALSE;
24             break;
25         end
26     end
27 end
28 return SUCCESS.
```

increases task recovery time. We define a metric to measure the energy saving gain over a unit resource demand increase caused by reducing the task's execution frequency from $f(\tau_i)$ to a lower one $f'(\tau_i)$, i.e., energy saving efficiency $\mathrm{ESE}(\tau_i, f(\tau_i), f'(\tau_i))$ is defined as

$$\mathrm{ESE}(\tau_i, f(\tau_i), f'(\tau_i)) = \frac{\mathcal{E}(f(\tau_i)) - \mathcal{E}(f'(\tau_i))}{\mathrm{Gap}(F) - \mathrm{Gap}(F')}. \tag{19}$$

where $F = \{f(\tau_1), \ldots, f(\tau_i), \ldots, f(\tau_n)\}$ and $F' = \{f(\tau_1), \ldots, f'(\tau_i), \ldots, f(\tau_n)\}$. $\mathcal{E}(f(\tau_i))$ is the expected total energy consumption of task τ_i under the assigned frequency $f(\tau_i)$ within one hyper-period H, which can be expressed as

$$\mathcal{E}_i(f(\tau_i)) = \frac{H}{T_i} \cdot E_i(f(\tau_i)). \tag{20}$$

$\mathrm{Gap}(F)$ is defined as

$$\min_{x \in [0,H]} \left\{ x - \sum_{\tau_i \in \Gamma} dbf_i^{\mathrm{LO}}(f(\tau_i), x) \,|\, \sum_{\tau_i \in \Gamma} dbf_i^{\mathrm{LO}}(f(\tau_i), x) \neq 0 \right\}, \tag{21}$$

where $dbf_i^{\mathrm{LO}}(f(\tau_i), x)$ is the resource demand of task τ_i under frequency $f(\tau_i)$, which can be calculated using (10) [1].

It is worth pointing out that fault recoveries also consume energy, but as the probability of fault occurrence is relatively small, the expected energy consumption for fault recoveries is negligible compared to the energy cost of executing all tasks in the task set.

The function $\mathrm{Gap}(F)$ indicates the gap between the resource demand and resource supply. It can be interpreted as the remaining time to be used for energy saving purposes. Hence, $\mathrm{Gap}(F)$ - $\mathrm{Gap}(F')$ is the extra execution time demand needed to lower down task τ_i's processing from $f(\tau_i)$ to $f'(\tau_i)$.

[1] $dbf_i^{\mathrm{LO}}(f(\tau_i), x)$ is abbreviated as $dbf_i^{\mathrm{LO}}(x)$ in Section 5 to simplify the representation.

Algorithm 2: HSFA(Γ, F, $R^C := \{R_1^c, \ldots, R_{|\Gamma|}^c\}$).

```
1  int[|Γ|] I := {0, ..., 0};
2  while ∃i : I[i] < |F| do
3      find Ψ := {τ_i | CHECK(Γ, F, R^C, I, τ_i) = TRUE ∧ τ_i ∈ Γ}
4      if Ψ is not empty then
5          find τ_k with ESE(τ_k, F[I[k]], F[I[k] + 1]) =
           max_{τ_i ∈ Ψ}{ESE(τ_i, F[I[i]], F[I[i] + 1])}
6          I[k] := I[k] + 1;
7      end
8      else
9          break;
10     end
11 end
12 return I;
```

Algorithm 3: CHECK(Γ, F, R^C, I, τ_i).

```
1  I[i] := I[i] + 1
2  set F_a := {F[I[1]], ..., F[I[i]]}
3  if GVDA(Γ, F_a, R^C) = SUCCESS then
4      return TRUE;
5  end
6  return FALSE;
```

With the ESE metric, we given a heuristic search based frequency assignment (HSFA) algorithm that minimizes energy consumption in Algorithm 2 .

In Algorithm 2, I is an array representing each task's execution frequency. Line 1 initializes each task's working frequency to $f_{\max}$. Function CHECK checks whether task τ_i's execution frequency can be scaled down by one level without violating the schedulability and reliability constraints. Line 3 finds the task set whose task's execution frequencies can be scaled down. Line 5 selects the one in Ψ that has the largest ESE value.

The pseudo-code code of the function CHECK is given in Algorithm 3 . It calls the function GVDA to determine if task τ_i's execution frequency can be reduced from $I[i]$ to $I[i] + 1$.

7. Evaluation

In this section, we evaluate the energy saving performance of our proposed approach. In our proposed approach, when the system operate under the HI-mode, we run all the tasks under the maximum frequency $f_{\max}$ and no energy saving can be achieved. Therefore, in the following evaluations, we focus on energy saving comparisons when the system operates under the LO-mode.

7.1. Experimental settings

The mixed-criticality task sets are generated by UUniForm algorithm (Bini and Buttazzo, 2005), which gives an unbiased distribution of task utilization. In particular, the task sets are generated in the following steps:

- use the UUniForm algorithm to generate task utilization, i.e. $u_H(\tau_i)$ and $u_L(\tau_i)$;
- randomly select task period T_i within [20, 100];
- set task execution time $C_i(HI)$ as $T_i \cdot u_H(\tau_i)$, where $u_H(\tau_i)$ is task τ_i's HI-mode utilization. For LO-criticality tasks, set $C_i(LO) = C_i(HI)$; while for HI-criticality tasks, set $C_i(LO) = \mu \cdot C_i(HI)$, where μ is a random value within the range of [0.3, 0.5].

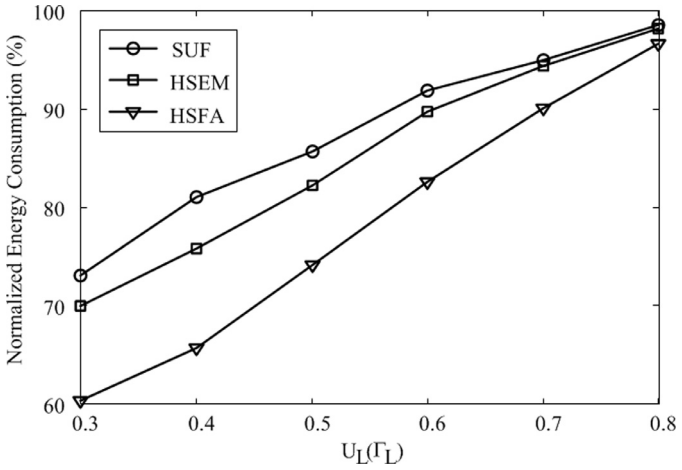

Fig. 4. Energy consumption under varied $U_L(\Gamma_L)$.

For energy model parameters, we assume $P_{\text{ind}} = 0.1$, $C_{\text{ef}} = 1$, and $\theta = 3$. The available frequencies are set as $F = \{0.4, 0.6, 0.8, 1\}$. For transient fault model parameters, we set $\lambda_0 = 10^{-6}$ and $d = 3$.

We evaluate our proposed approach by comparing it with the two heuristics, i.e. the smallest utilization first (SUF) approach and the heuristic search based energy minimization (HSEM) approach presented in our earlier work (Li et al., 2014) from the following perspectives:

- sensitivity to LO-mode and HI-mode utilization;
- sensitivity to the number of tasks in the task set;
- sensitivity to the fault arrival rate.

Though Huang et al.'s approach (Huang et al., 2014) does not guarantee the reliability constraint and hence cannot be used to solve our formulated problem, we still add the comparison between our HSFA approach and Huang et al.'s approach as an additional set of experiments.

7.2. Experiment results and discussions

For comparison purposes, energy consumption is normalized to the value when all tasks run under the highest frequency $f_{\max} = 1$. For each experiment, we run the compared algorithms for 100 randomly generated task sets, the average energy consumption and the standard derivation are used in the comparisons.

7.2.1. Sensitivity to LO-mode and HI-mode utilization

In the first set of experiments, each task set has six tasks. Among which, three are of HI-criticality and the other three are of LO-criticality. We set $U_H(\Gamma_H) = 0.3$ and vary $U_L(\Gamma_L)$ from 0.3 to 0.8 with step of 0.1. The average normalized energy consumption is illustrated in Fig. 4. It is not hard to find that all compared approaches consume more energy under higher task set utilization. This is because that task set of higher utilization has less slack time, hence, less chance to scale down task working frequencies for energy saving purposes. In addition, with the SUF and HSEM approaches, each HI-criticality task is assumed to have a carry-over instance and hence the resource demand calculation is too pessimistic. With the HSFA algorithm, the over demand is excluded and the energy saving performance is improved. These explain why the HSFA approach can save up to 15% and 10% more energy than SUF and HSEM approaches, respectively.

As shown in Fig. 5, the standard derivation of all the compared approaches become smaller when $U_L(\Gamma_L)$ is increased. Under lower $U_L(\Gamma_L)$, the SUF algorithm has much higher variation than the HSEM and the HSFA algorithm. However, when the $U_L(\Gamma_L)$ increases, the variation of the three approaches almost converge at the same point.

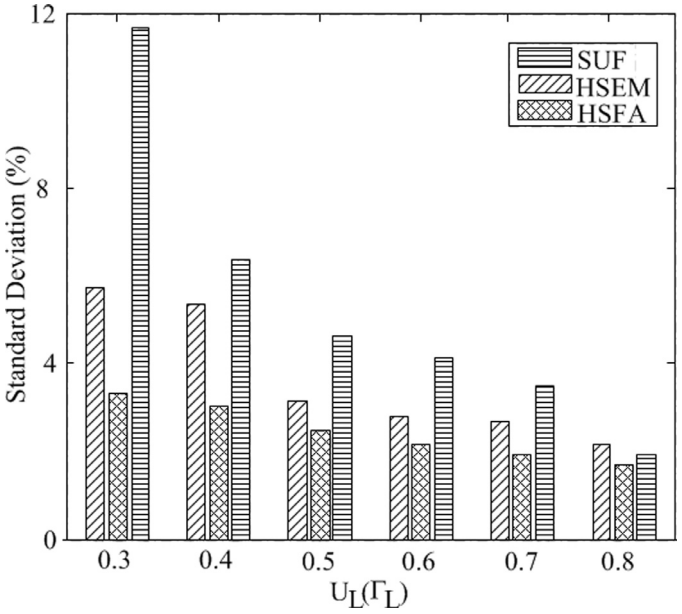

Fig. 5. Energy consumption variation under varied $U_L(\Gamma_L)$.

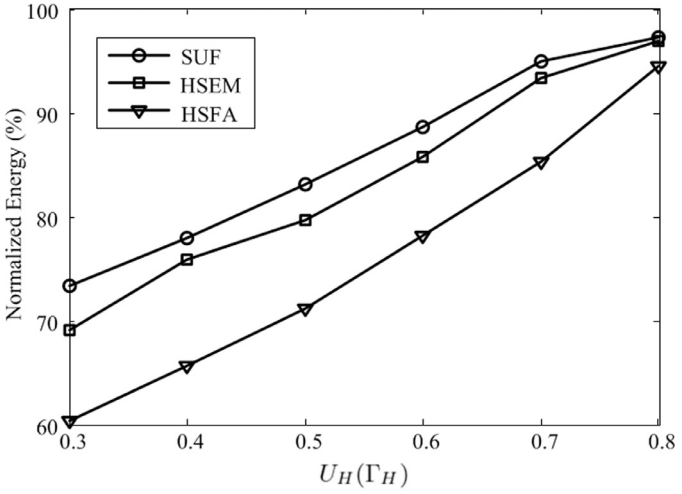

Fig. 6. Energy consumption under varied $U_H(\Gamma_H)$.

For the second set of experiments, we fix $U_L(\Gamma_L) = 0.3$ and vary $U_H(\Gamma_H)$ from 0.3 to 0.8. The experimental results are depicted in Figs. 6 and 7. It is not hard to find that HI-mode utilization also impacts the system's energy saving under the LO-mode. The reason is as follows. The system initially runs under the LO-mode and may change to the HI-mode at any time point. In order to guarantee HI-criticality task deadlines under the HI-mode, enough slack time must be reserved under the LO-mode to accommodate the possible task overruns. Higher HI-mode utilization means more slack time reservation, and hence, as shown in Fig. 6, less energy saving can be achieved.

7.2.2. Sensitivity to the number of tasks in the task set

In this set of experiments, we set $U_H(\Gamma_H) = 0.3$, $U_L(\Gamma_L) = 0.6$ and increase the number of tasks from 6 to 12 (same number of LO-criticality and HI-criticality tasks) to evaluate how the number of tasks impacts the proposed approach. The experimental results depicted in Figs. 8 and 9 reveal that the energy saving performance is slightly improved with the increase of the number of tasks in the task set.

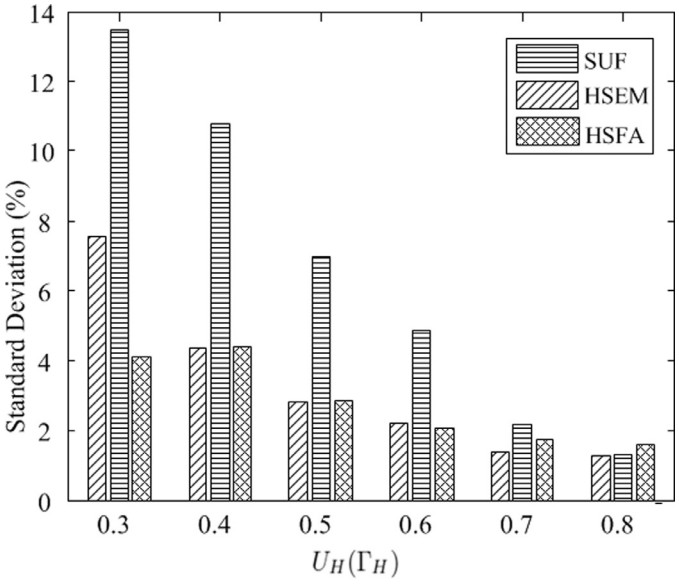

Fig. 7. Energy consumption variation under varied $U_H(\Gamma_H)$.

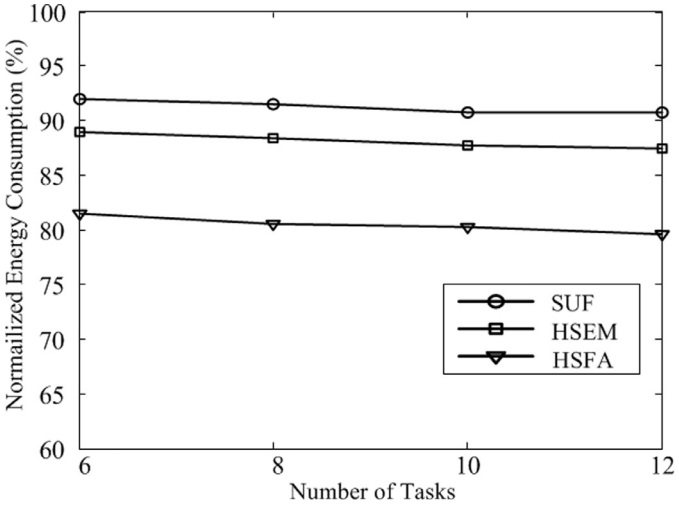

Fig. 8. Energy consumption under a different number of tasks.

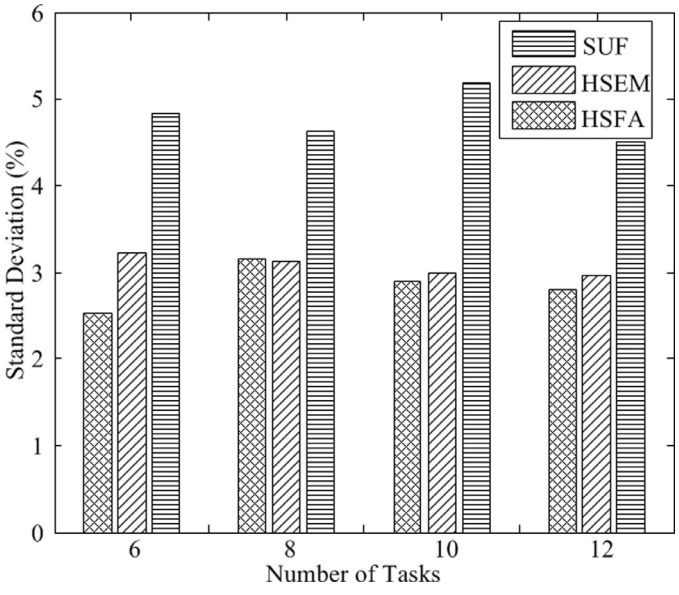

Fig. 9. Energy consumption variation under different number of tasks.

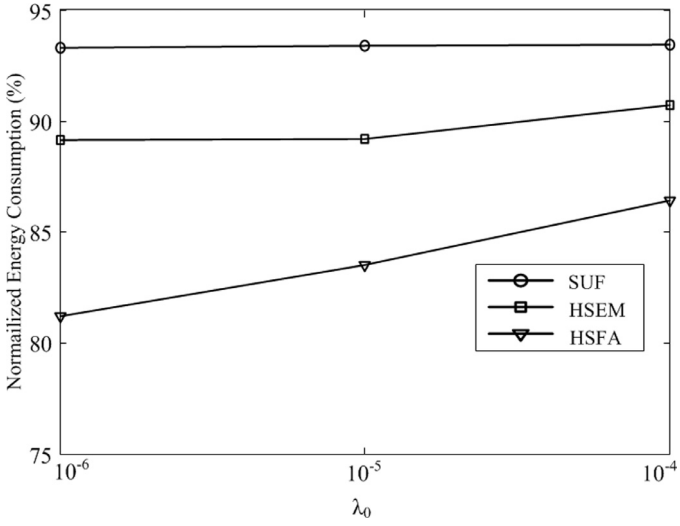

Fig. 10. Energy consumption under different λ_0.

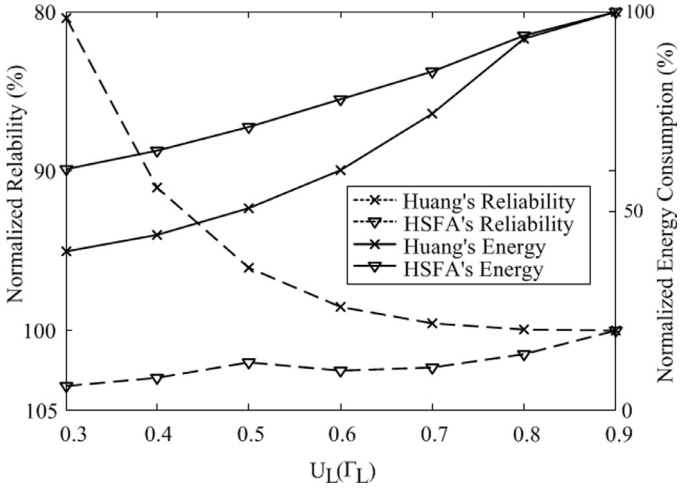

Fig. 11. Comparisons between HSFA and Huang's Approach.

By comparing the Figs. 6 and 4, we can conclude that the number of tasks has less impact on the energy saving performance than task utilization.

7.2.3. Sensitivity to the fault arrival rate

In this subsection, we evaluate the impact of fault arrival rate by varying λ_0 from 10^{-4} to 10^{-6}. $U_H(\Gamma_H)$, $U_L(\Gamma_L)$ and the number of tasks are set to 0.3, 0.6 and 6, respectively. As illustrated in Fig. 10, the increase of fault arrival rate λ_0 will degrade the energy saving performance. Larger λ_0 indicates more slack time to be reserved for fault recovery, and hence, less slack time remains for energy saving purpose.

7.2.4. Comparison with Huang et al.'s approach

We also set up experiments to compare Huang et al.'s approach (Huang et al., 2014) with our HSFA approach. We set $U_H(\Gamma_H) = 0.3$ and increase $U_L(\Gamma_L)$ from 0.3 to 0.8. According to the results illustrated in Fig. 11, Huang et al.'s approach has better energy saving performance when $U_L(\Gamma_L)$ is less than 0.9. When the $U_L(\Gamma_L)$ is set at 0.9, both HSFA and Huang et al.'s approaches converge to the same point and almost no energy saving can be achieved. Though Huang et al.'s approach can save more energy under low utilization, but their solution cannot meet the reliability constraint. As shown in Fig. 11, when $U_L(\Gamma_L) = 0.3$, the achieved reliability is only 80% of the requirement.

Hence, Huang et al.'s approach cannot be used to solve our formulated problem.

8. Conclusion

In this paper, we have studied the mixed-criticality task set scheduling problem with the goal of minimizing system energy consumption while at the same time guaranteeing the satisfaction of task set deadline and reliability requirements. We have first established the theoretical foundation that decides under given task virtual deadline and execution frequency assignments whether the tasks' deadline and reliability constraints can be satisfied under EDF-VD scheduling algorithm. Based on the theoretic analysis, we have developed a heuristic search based frequency assignment algorithm that minimizes system's energy consumption and at the same time guarantees both task reliability and schedulability constraints are satisfied. Empirical evaluations show that the proposed approach can save up to 10% more energy compared to our earlier work in the literature.

Acknowledgment

This work was supported, in part, by NSF CNS 1018731 and NSF Career 0746643.

References

Axer, P., Sebastian, M., Ernst, R., 2011. Reliability analysis for MPSOCS with mixed-critical, hard real-time constraints. In: Proceedings of the Seventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. ACM, New York, NY, USA, pp. 149–158.

Aydin, H., Devadas, V., Zhu, D., 2006. System-level energy management for periodic real-time tasks. In: Proceedings of 27th IEEE International Real-Time Systems Symposium, pp. 313–322.

Aydin, H., Zhu, D., 2009. Reliability-aware energy management for periodic real-time tasks. IEEE Trans. Comput., 58 (10), 1382–1397.

Barhorst, J., Belote, T., Binns, P., Hoffman, J., Paunicka, J., Sarathy, P., Scoredos, J., Stanfill, P., Stuart, D., Urzi, R., 2009. A research agenda for mixed-criticality systems. In: Proceedings of the Cyber-Physical Systems Week.

Baruah, S., Bonifaci, V., D'Angelo, G., Li, H., Marchetti-Spaccamela, A., Megow, N., Stougie, L., 2012. Scheduling real-time mixed-criticality jobs. IEEE Trans. Comput., 61 (8), 1140–1152.

Baruah, S., Bonifaci, V., D'Angelo, G., Li, H., Marchetti-Spaccamela, A., Van der Ster, S., Stougie, L., 2012. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In: Proceedings of 24th Euromicro Conference on Real-Time Systems (ECRTS), 2012, pp. 145–154.

Baruah, S., Vestal, S., 2008. Schedulability analysis of sporadic tasks with multiple criticality specifications. In: Proceedings of the Euromicro Conference on Real-Time Systems, 2008. ECRTS '08, pp. 147–155.

Bini, E., Buttazzo, G., 2005. Measuring the performance of schedulability tests. Real-Time Syst. 30 (1-2), 129–154.

Burd, T.D., Brodersen, R.W., 1995. Energy efficient cmos microprocessor design. In: Proceedings of the 28th Hawaii International Conference on System Sciences, pp. 288–297.

Burns, A., Davis, R.I., 2013. Mixed criticality systems: A review. Technical Report MCC-1(b). Department of Computer Science, University of York, East Lansing, Michigan.

Buttazzo, G., Lipari, G., Abeni, L., 1998. Elastic task model for adaptive rate control. In: Proceedings of the 19th IEEE Real-Time Systems Symposium, 1998, pp. 286–295.

Ekberg, P., Yi, W., 2012. Bounding and shaping the demand of mixed-criticality sporadic tasks. In: Proceedings of the 24th Euromicro Conference on Real-Time Systems (ECRTS), 2012, pp. 135–144.

Ekberg, P., Yi, W., 2014. Bounding and shaping the demand of generalized mixed-criticality sporadic task systems. Real-Time Syst. 50 (1), 48–86.

Guo, Y., Zhu, D., Aydin, H., 2013. Generalized standby-sparing techniques for energy-efficient fault tolerance in multiprocessor real-time systems. In: Proceedings of the IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2013, pp. 62–71.

Huang, P., Kumar, P., Giannopoulou, G., Thiele, L., 2014. Energy efficient dvfs scheduling for mixed-criticality systems. In: Proceedings of the 14th International Conference on Embedded Software, pp. 11:1–11:10.

Koren, G., Shasha, D., 1995. Skip-over: algorithms and complexity for overloaded systems that allow skips. In: Proceedings of the 16th IEEE Real-Time Systems Symposium, 1995., pp. 110–117.

Li, Z., Hua, X., Guo, C., Ren, S., 2014. Empirical study of energy minimization issues for mixed-criticality systems with reliability constraint. In: Proceedings of the 1st Workshop on Low-Power Dependable Computing (LPDC), 2014.

Li, Z., Wang, L., Ren, S., Quan, G., 2013. Energy minimization for checkpointing-based approach to guaranteeing real-time systems reliability. In: Proceedings of the IEEE 16th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2013, pp. 1–8.

Niu, L., Xu, J., 2015. Improving schedulability and energy efficiency for window-constrained real-time systems with reliability requirement. J. Syst. Archit. 61 (5C6), 210–226.

Park, T., Kim, S., 2011. Dynamic scheduling algorithm and its schedulability analysis for certifiable dual-criticality systems. In: Proceedings of the Ninth ACM International Conference on Embedded Software. ACM, New York, NY, USA, pp. 253–262.

Pathan, R., 2014. Fault-tolerant and real-time scheduling for mixed-criticality systems. Real-Time Syst. 50 (4), 509–547.

Su, H., Zhu, D., 2013. An elastic mixed-criticality task model and its scheduling algorithm. In: Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE), 2013, pp. 147–152.

Zhang, T., Guan, N., Deng, Q., Yi, W., 2014. On the analysis of EDF-VD scheduled mixed-criticality real-time systems. In: Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES), 2014, pp. 179–188.

Zhao, B., Aydin, H., Zhu, D., 2009. Enhanced reliability-aware power management through shared recovery technique. In: Proceedings of the International Conference on Computer-Aided Design, pp. 63–70.

Zhao, B., Aydin, H., Zhu, D., 2011. Generalized reliability-oriented energy management for real-time embedded applications. In: Proceedings of the Design Automation Conference, pp. 381–386.

Zhao, B., Aydin, H., Zhu, D., 2012. Energy management under general task-level reliability constraints. In: Proceedings of the 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), 0, pp. 285–294.

Zhu, D., 2011. Reliability-aware dynamic energy management in dependable embedded real-time systems. ACM Trans. Embed. Comput. Syst. 10 (2), 1–27.

Zhu, D., Melhem, R., Mosse, D., 2004. The effects of energy management on reliability in real-time embedded systems. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp. 35–40.

Zheng Li received the B.S. degree in Computer Science and M.S. degree in Communication and Information System from the University of Electronic Science and Technology of China, and Ph.D. degree in Computer Science from Illinois Institute of Technology, in USA. He is currently an assistant professor in the School of Computer Science at Western Illinois University. His research interests include real-time system design, many-core computing and recongurable computing.

Chunhui Guo received his BSEE and MSEE from Shandong University, China, in 2010 and 2013, respectively. He is now a Ph.D candidate in the Computer Science Department at Illinois Institute of Technology. His current research interests mainly focus on real-time systems and Cyber-Physical System.

Xiayu Hua received his B.S. degree from the Northwestern Polytechnic University, China, in 2008 and his M.S. degree from the East China Normal University, China, in 2012. He is now a Ph.D candidate in the Computer Science Department at Illinois Institute of Technology. His research interests are in distributed file system, virtualization technology, real-time scheduling and cloud computing.

Shangping Ren received her B.S. and M.S. in Computer Science from Hefei Polytechnique University in China, and Ph.D degree from the University of Illinois at Urbana-Champaign, in USA. She is currently a Professor in the Department of Computer Science at the Illinois Institute of Technology. Her research interests include coordination models, distributed real-time systems, and programming languages. She is a senior member of the IEEE.