Energy Efficient DVFS Scheduling for Mixed-Criticality Systems

Pengcheng Huang, Pratyush Kumar, Georgia Giannopoulou, Lothar Thiele
Computer Engineering and Networks Laboratory, ETH Zurich, 8092 Zurich, Switzerland
firstname.lastname@tik.ee.ethz.ch

ABSTRACT

Consolidating functionalities with different safety requirements into a common platform gives rise to mixed-criticality systems. The state-of-the-art research has focused on providing heterogeneous timing guarantees for tasks of varying criticality levels. This is achieved by dropping less critical tasks when critical tasks overrun. However, with drastically increased computing requirements and the often battery-operated nature of mixed-criticality systems, energy minimization for such systems is also becoming crucial. In fact, this has already been possible since many modern processors are equipped with the capacity of dynamic voltage and frequency scaling (DVFS), where processor frequency can be reduced at runtime to save energy.

We present in this paper the first results known to date on applying DVFS to mixed-criticality systems. We show that DVFS can be used to help critical tasks to meet deadlines by speeding up the processor when they overrun. This will further allow the system to reserve less time budgets for task overrun. Thus, more slack can be explored to reduce the processor frequency to save energy for scenarios when tasks do not overrun. Since overrun is rare, such a strategy can greatly reduce the expected energy consumption for mixed-criticality systems. For solving the energy minimization problem, we formulate a convex program by integrating DVFS with a well-known mixed-criticality scheduling technique – EDF-VD. Furthermore, we present analytical results on this problem and propose an optimal algorithm to solve it. With both theoretical and experimental results, we demonstrate energy savings and various tradeoffs.

Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Real-time and embedded systems; C.4 [**Performance of systems**]: Performance attributes

General Terms

Algorithms, Design, Performance

ESWEEK '14, October 12 - 17 2014, New Delhi, India

http://dx.doi.org/10.1145/2656045.2656057

Keywords

Mixed-Criticality, Energy, Voltage and Frequency Scaling, Real-time, Scheduling

1. INTRODUCTION

Recently, there is an increasing trend to consolidate functionalities of varying importances (criticality levels) into a common computing platform. Such a trend has been observed in several major industries (e.g. automotive and avionics [1]), and has lead to the emergence of mixed-criticality systems – systems for which varying degrees of assurance must be provided to functionalities of varying importances.

A typical example of mixed-criticality systems is the unmanned aerial vehicles (UAVs) [2], where functionalities can be categorized as safety- or mission-critical. The safety-critical functionalities include those like flight control and trajectory computation, losing which a UAV can no longer be safely operated. While the mission-critical functionalities may include object tracking for surveillance purposes, losing which a UAV is still considered safe. Correspondingly, in a system level model for mixed-criticality systems, the design parameters for safety-critical functionalities are usually much more pessimistic than that for mission-critical functionalities (e.g. task worst case execution times (WCETs)).

State-of-the-art. Pessimistic design parameters usually imply resource inefficiency at runtime. In order to further facilitate resource efficient mixed-criticality systems, the state-of-the-art model [3, 4, 5, 6, 7, 8, 9] for mixed-criticality systems specifies system guarantee in a dynamic manner: For each task in the system, its WCET is modeled on *all* existing criticality levels, with the one on a higher criticality level being more pessimistic. And whenever any task overruns its χ criticality level WCET, only tasks with criticality levels higher than χ are guaranteed thereafter. A task set is said to be mixed-criticality schedulable if such dynamic guarantee can be provided by some scheduling algorithm. Indeed, it has been demonstrated in various results (e.g. [3, 4, 8]) that schedulability of mixed-criticality systems can be considerably improved this way.

Motivation. The state-of-the-art research on mixed-criticality systems has primarily focused on providing heterogeneous timing guarantees for tasks of different criticality levels. However, to the best of our knowledge, other non-functional properties like energy consumption are still not explored. Indeed, energy is also a critical issue for safety-critical systems. We outline here three main reasons. First, the volume of computing we embed into safety-critical systems is becoming considerably large. For example, current high-end vehicle computers can consist of more than 100 electronic control units (ECUs) [10]. This requires a

tremendous power supply. Thus, to sustain the computing, energy minimization in the cyber space becomes vital. Second, many safety-critical devices like the medical heart pacemakers are battery-operated. Therefore, low energy consumption could directly lead to significant economic gains by increasing device lifetime and reducing maintenance costs. Last, with drastic increase in power density of modern electronic circuits, chip temperature has become a prominent issue, which could affect system reliability and timing correctness [11]. Hence, energy saving is crucial also due to the thermal issues.

We address in this paper the new challenge to minimize energy for mixed-criticality systems. In particular, we investigate offline dynamic voltage and frequency scaling (DVFS) to reduce energy consumptions while preserving mixed-criticality schedulability of the system. However, applying DVFS in a mixed-criticality setting is non-trivial due to the following features of this problem:

Unclear energy objective. For mixed-criticality systems, there is a bounded uncertain workload we have to guarantee for critical tasks (for the sake of system safety). Though assumed to be improbable, the uncertain workload complicates energy minimization as we do not know for which workload we are going to minimize the energy consumption. Hence, a proper energy objective needs to be justified in the first place.

Conflict between energy minimization and safety guarantee. The main idea of applying DVFS to minimize energy consumption is to stretch task execution times as much as possible by lowering the processor frequency, such that tasks finish "just in time". In other words, DVFS tries to explore all slack in the system and pushes the execution of tasks to the time limit. On the contrary, in order to guarantee system safety, we have to reserve time budget for critical tasks, such that they can still meet their deadlines even if they overrun. This needs to be *prepared* a priori and prevents us from exploring all available slack in the system. Such a conflict needs to be taken into account when solving the mixed-criticality energy minimization problem.

Hardness. Without any detailed settings (e.g. used scheduling algorithm, available frequencies), a first investigation reveals that the mixed-criticality energy minimization problem is NP-hard in the strong sense. This can be shown by examining a subproblem of the energy minimization problem, where there is only one frequency level in the system. In this case, the energy minimization problem is identical to the mixed-criticality scheduling problem [12], which has already been proved to be strongly NP-hard [12]. This confirms the NP-hardness of the mixed-criticality energy minimization problem.

Rationale. Due to the hardness of the energy minimization problem, we will focus in this paper on integrating DVFS with a well-known mixed-criticality scheduling algorithm – EDF-VD [4], which targets dual-criticality tasks with implicit deadlines. In addition, since task overrun is unlikely, we contend in this paper that it is proper to minimize the system energy consumption for the scenario when no task overruns, which we call as the *expected* energy consumption. This is a realistic assumption as the system will most likely only execute under this scenario. Furthermore, as overrun is rare, it is affordable for the system to speedup by DVFS to handle overrun. This will in turn allow the system to reserve less time budgets for task overrun and more slack could be explored by DVFS to minimize the expected energy consumption.

Contribution. We present in this paper results on applying DVFS to mixed-criticality systems for energy minimization. To our best knowledge, these are the first results in this setting. Our detailed contributions are as follows:

- We clarify the characteristics of the mixed-criticality energy minimization problem and show that speeding up to handle overrun is beneficial for minimizing the system's expected energy consumption.
- We integrate DVFS with the EDF-VD scheduling technique and formulate our energy minimization problem as a convex program. Moreover, we show there is a conflict between system timing safety and energy minimization.
- We provide results on reducing the search space for our energy minimization problem and present an optimal algorithm to find the solution to our problem.
- We demonstrate with both theoretical and experimental results energy savings and various tradeoffs for the mixed-criticality energy minimization problem.

Organization. Our paper is organized as follows: Our system models and some preliminaries are given in Section 2. We present a motivational example and our problem definition in Section 3. Section 4 presents a convex program formulation of our energy minimization problem. We present in Section 5 an optimal solution algorithm for our formulated problem. We discuss in Section 6 some tradeoffs and extensions for our proposed techniques. Section 7 presents experimental results and Section 9 concludes the paper.

2. SYSTEM MODELS AND PRELIMINARY

2.1 Mixed-Criticality Sporadic Task Model

We adopt in this paper the state-of-the-art mixed-criticality task model [5, 3, 6, 7, 8, 9, 13]. Given is a dual-criticality sporadic task set $\tau = \{\tau_1, \ldots, \tau_n\}$ scheduled on a uniprocessor. Each task has either high (HI) or low (LO) criticality. In addition, each task τ_i is characterized by a minimal inter-arrival time T_i and a relative deadline D_i. We assume all tasks have implicit deadlines, i.e. $\forall \tau_i \in \tau, T_i = D_i$. A task τ_i may issue an infinite number of jobs. For notational convenience, we denote all χ criticality level tasks by $\tau_\chi = \{\tau_i \mid \chi_i = \chi\}$.

The WCETs of all tasks are modeled on both criticality levels. Namely, each task τ_i has a HI criticality WCET $C_i(\text{HI})$ and a LO criticality WCET $C_i(\text{LO})$. For all tasks, the WCETs on HI criticality are strictly non-decreasing when compared to their WCETs on LO criticality. The rationale behind this is that the WCET on HI criticality is typically more conservative than that on LO criticality (to ensure timing safety). We assume that LO criticality tasks are not allowed to exceed their LO criticality WCETs.

Based on the above model, the system then provides *dynamic* guarantees to all tasks based on their revealed execution times at runtime, which can be specified by a simple mode switch protocol:

- The system starts with the LO mode, where all tasks do not exceed/overrun their LO criticality WCETs and are guaranteed to meet their deadlines.
- If any HI criticality task exceeds its LO criticality WCET, then the system transits immediately to the HI mode, where all LO criticality tasks are dropped and HI criticality tasks are guaranteed to meet their deadlines if they do not exceed their HI criticality WCETs.

For a given scheduling algorithm, the system is said mixed-criticality *schedulable* if such dynamic guarantees are provided to all tasks. Notice, however, the system could further switch back from the HI mode to the LO mode, providing that all tasks can meet their deadlines after transiting back, see e.g. [13].

2.2 Power Model and DVFS

In this paper, we adopt the state-of-the-art power model [14, 15, 16]:

$$P(f) = P_s + P_d(f) = P_s + \beta \cdot f^{\alpha}, \tag{1}$$

where P_s stands for the static power consumption due to leakage current, and P_d represents the active power consumption due to switching activities, which depends on the processor frequency. P_d can be represented as $\beta \cdot f^{\alpha}$, where β is a circuit dependent positive constant and $\alpha \geqslant 2$ (a common assumption is 3 [17, 18]). As a result, the power consumption is a convex increasing function of the processor frequency. Moreover, by dynamic voltage and frequency scaling (DVFS), one can reduce the processor frequency to reduce P_d.

We focus in this paper on minimizing the active energy consumption (due to P_d) by means of DVFS. And in our later formulations we will omit P_s for simplicity of presentation. We assume that the processor frequency can be freely set in the range $[f_{\min}, f_{\max}]$. We additionally assume that the task WCETs as presented in our task model are measured on a base frequency f_b (the processor originally operates on this frequency), $f_{\min} \leqslant f_b \leqslant f_{\max}$. Without loss of generality, we assume that $f_{\max}$ is normalized to 1.

Based on our models, the χ criticality worst-case number of clock cycles of task τ_i can be computed as $C_i(\chi)/\frac{1}{f_b} = C_i(\chi)f_b$, which is a constant under DVFS. For ease of presentation, we will call the invariant

$$C_i(\text{LO})f_b \tag{2}$$

as the *normal* workload of task τ_i, and the invariant

$$(C_i(\text{HI}) - C_i(\text{LO}))f_b \tag{3}$$

as its *extra* workload. If task τ_i runs constantly on frequency f, then its actual χ criticality WCET is $C_i(\chi)f_b/f$.

2.3 Recap of the EDF-VD scheduling technique

We continue to introduce some basic concepts of mixed-criticality scheduling and the EDF-VD scheduling technique [4], on top of which we will design our DVFS strategy.

2.3.1 Preparation for safety

In order to ensure that HI criticality tasks can still meet their deadlines even if they overrun, time budgets need to be reserved for those tasks. Alternatively speaking, the safety of the system needs to be prepared (in terms of extra workload for HI criticality tasks), even when the system is still in the LO mode.

2.3.2 EDF-VD

Earliest Deadline First with Virtual Deadline (EDF-VD) is a mode-switched EDF scheduling technique developed for mixed-criticality task sets [4, 8]. The reservation of time budgets for HI criticality tasks is done in the LO mode. This is achieved by shortening the deadlines of HI criticality tasks. Intuitively, shortening the deadlines of HI criticality tasks will push them to finish earlier in the LO mode, leaving more time until their actual deadlines to accommodate extra workloads. Indeed, this form of safety preparation (i.e. shortening deadlines of HI criticality tasks in the LO mode) has been proved to be effective in improving system schedulability [4]. For further details about EDF-VD, we also refer the readers to [4].

Table 1: Example task set with parameters in units of ms

	χ_i	T_i	C_i(HI)	C_i(LO)
τ_1	HI	8	5	2
τ_2	LO	12	1	1
τ_3	LO	16	2	2

Let us denote with x ($0 < x \leqslant 1$) the factor by which deadlines of HI criticality tasks are uniformly multiplied in the LO mode. And let us use $U_{\chi 1}^{\chi 2}$ to denote $\sum_{\tau_i \in \tau_{\chi_1}} \frac{C_i(\chi_2)}{T_i}$. The original EDF-VD scheduling technique sets x as $\frac{U_{\text{HI}}^{\text{LO}}}{1 - U_{\text{LO}}^{\text{LO}}}$, which is the *minimal* x guaranteeing the system schedulability in the LO mode [4] ($\frac{U_{\text{HI}}^{\text{LO}}}{x} + U_{\text{LO}}^{\text{LO}} \leqslant 1$). Without any DVFS, a tight schedulability test can then be formalized as follows [4].

THEOREM 2.1. *[4] A dual-criticality task set is schedulable under EDF-VD if:*

$$U_{HI}^{HI} + xU_{LO}^{LO} \leqslant 1. \tag{4}$$

2.4 Other notations

For notational simplicity, we use $[\![a]\!]_b$ to represent $\max(a, b)$ and $[\![a]\!]^c$ to represent $\min(a, c)$. Furthermore, $[\![a]\!]_b^c$ represents $[\![[\![a]\!]_b]\!]^c$. For a given domain $[b, c]$, this operation keeps the value of a if $b \leqslant a \leqslant c$, otherwise it equals b if $a < b$ or c if $a > c$.

3. MOTIVATIONAL EXAMPLE AND PROBLEM DEFINITION

Recall our discussion in Section 1, we aim at minimizing the expected system energy consumption by offline DVFS for tasks scheduled under EDF-VD. According to our models, the system operates under the LO mode when all tasks adhere to their LO criticality WCETs. Hence, we will alternatively call the expected energy consumption as the LO mode energy consumption. Due to the improbability of transiting to the HI mode, it is beneficial to speedup HI criticality tasks when they overrun, which will decrease their actual HI criticality WCETs. Thus, less time budgets need to be reserved for HI criticality tasks, and more slack could be used to stretch task executions in the LO mode to reduce energy consumption. We provide here a concrete example to explain this.

EXAMPLE 3.1. *Consider a sporadic implicit deadline task set with task parameters given in Table 1. All task WCETs are measured on the highest frequency level $f_{\max}$. The task set is scheduled by EDF-VD. We compare the expected energy consumptions of 3 different strategies:*

Strategy A *No DVFS – all tasks run on $f_{\max}$.*

Strategy B *One frequency level per task – each task runs on one single frequency in both modes, with τ_1, τ_2 and τ_3 running on frequencies 0.81, 0.49 and 0.49, respectively.*

Strategy C *Speedup for overrun – the HI criticality task τ_1 runs on frequency 0.65 for its normal workload, and on*

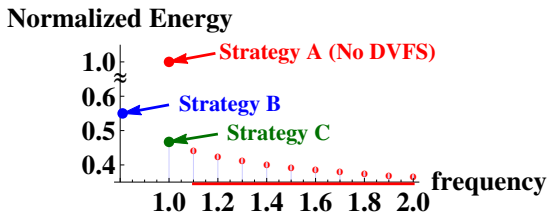


Figure 1: Normalized LO mode energy against the frequency of τ_1's extra workload, with the system energy consumption calculated in 48 ms, α in (1) set to 2.5, $f_{\min}$ set to 0.2, and the energy for Strategy A normalized to 1

frequency 1 ($f_{\max}$) for its extra workload. Both τ_2 and τ_3 run on frequency 0.54.

For all three strategies, the system is schedulable according to the EDF-VD test (Theorem 2.1, notice that scaled task WCETs need to be used when running the test due to DVFS). We show the energy dissipations in Figure 1. In fact, task frequencies of Strategy B are derived by Mathematica numerical optimization [19] as the optimal solution when each task runs on a single frequency level. However, by allowing τ_1 to speedup when it overruns (Strategy C), we can get better expected energy saving (8% more compared to Strategy B). If *the maximum available frequency $f_{\max}$ is increased, then by speeding up τ_1's extra workload more, the expected energy consumption can be further reduced (results in this case in Figure 1 are obtained by numerical optimization with our problem formulation in Section 4). This example confirms that speeding up the extra workload for HI criticality tasks can help to achieve better expected energy savings.*

Based on our discussions, the formal definition of our energy minimization problem in this paper is as follows.

DEFINITION 3.1. *(Mixed-Criticality Energy Minimization (MCEM)) Given a dual-criticality sporadic task set τ scheduled under EDF-VD, decide offline:*

- *$\forall \tau_i \in \tau_{LO}$, one frequency level f_i^{LO} on which it should run,*
- *$\forall \tau_i \in \tau_{HI}$, one frequency level f_i^{LO} for its normal workload $C_i(LO)f_b$, and one frequency level f_i^{HI} for its extra overload $(C_i(HI) - C_i(LO))f_b$,*
- *a deadline shortening factor x for all HI criticality tasks,*

such that the LO mode system energy consumption is minimized while mixed-criticality schedulability of the system is satisfied.

Notice that each LO criticality task only needs a frequency in the LO mode as it is dropped in the HI mode. There are still other possibilities to have degraded services for LO criticality tasks in the HI mode (e.g. [13]), however this is beyond the scope of this paper. In addition, for each HI criticality task τ_i, f_i^{HI} is only required if the task actually exceeds its LO criticality WCET at runtime ($\frac{C_i(\text{LO})f_b}{f_i^{\text{LO}}}$ with DVFS). For completeness, we include in our problem definition the deadline shortening factor x as a decision variable. However, this is not strictly necessary since x can be automatically inferred by EDF-VD as the minimal value guaranteeing the system schedulability (Section 2.3). We model the deadline shortening factor explicitly for two main reasons:

- As we will show in Section 4, the choice of x is critical to our energy minimization problem.
- Modeling x as an explicit decision variable will allow us to formulate our problem as a convex program.

4. CONVEX PROBLEM FORMULATION

We have introduced in Section 3 the MCEM problem (Definition 3.1). In this section, we continue to present a convex program formulation of the energy minimization problem. For this purpose, we first discuss the choice of the deadline shortening factor x for the MCEM problem.

4.1 The choice of the deadline shortening factor x

Conflict between schedulability and energy minimization. On the one hand, to improve the system schedulability, we need to choose a deadline shortening factor x as small as possible. This will allow HI criticality tasks to finish their normal workloads as early as possible, leaving enough time to accommodate their extra workloads. Indeed, the original EDF-VD scheduling technique uses a minimal x to shorten deadlines of HI criticality tasks, as presented in Section 2.3. On the other hand, to reduce the LO mode energy consumption, a larger x could be better. With increasing x, deadlines of HI criticality tasks in the LO mode are extended, and tasks have more slack to stretch their executions by DVFS.

Quantification of the feasible range of x. We continue to quantify a feasible range of x that will guarantee mixed-criticality schedulability. Intuitively, if x is too large, then HI criticality tasks would finish their normal workloads too late (in the worst-case, just at their deadlines), and the system might not be schedulable in the HI mode. However, if x is too small, then the system might not even be schedule in the LO mode due to small task deadlines. Those two scenarios give the bounds of x from top and bottom, respectively. In addition, the operating frequencies will also affect the feasible range of x since they will change the actual WCETs of all tasks. Let us denote by $\tilde{C}_i(\chi)$ task τ_i's χ criticality WCET after DVFS. According to our problem definition (Definition 3.1), it is straightforward to show that after DVFS:

$$\tilde{C}_i(\text{LO}) = \frac{C_i(\text{LO})f_b}{f_i^{\text{LO}}}, \ \forall \tau_i \in \tau,$$
$$\tilde{C}_i(\text{HI}) = \frac{C_i(\text{LO})f_b}{f_i^{\text{LO}}} + \frac{(C_i(\text{HI}) - C_i(\text{LO}))f_b}{f_i^{\text{HI}}}, \ \forall \tau_i \in \tau_{\text{HI}}. \tag{5}$$

Let us further define $\tilde{U}_{\chi_1}^{\chi_2}$ as:

$$\tilde{U}_{\chi_1}^{\chi_2} = \sum_{\tau_i \in \tau_{\chi_1}} \frac{\tilde{C}_i(\chi_2)}{T_i}. \tag{6}$$

Formally, the following result is presented to determine the feasible range of x.

LEMMA 4.1. *With DVFS for the MCEM problem, a dual-criticality sporadic task set τ is schedulable under the EDF-VD schedulability test iff there exists a feasible range of x:*

$$0 < x_{LB} \leqslant x \leqslant x_{UB} \leqslant 1, \tag{7}$$

where

$$x_{LB} = \frac{\tilde{U}_{HI}^{LO}}{1 - \tilde{U}_{LO}^{LO}}, \tag{8}$$

and

$$x_{UB} = [\![\frac{1 - \tilde{U}_{HI}^{HI}}{\tilde{U}_{LO}^{LO}}]\!]^1. \tag{9}$$

The widest feasible range is obtained when the processor frequency is set to a constant $f_{\max}$. Furthermore, if

- *$x < x_{LB}$, then LO mode is not schedulable;*
- *$x > x_{UB}$, then HI mode is not schedulable.*

PROOF. All proofs are presented in [20]. □

According to Lemma 4.1, we can compute the upper and lower bounds of x assuming task frequencies are given, and we obtain the widest feasible range of x when the processor frequency is set to $f_{\max}$. (This is intuitive – with increasing frequency of the processor, the schedulability of the system will be improved, and the feasible range of x will widen.) For notational convenience, we will denote the lower and upper bounds in this case as $\hat{x}_{\text{LB}}$ and $\hat{x}_{\text{UB}}$. We call $[\hat{x}_{\text{LB}}, \hat{x}_{\text{UB}}]$ as the maximum feasible range.

Notice that for our energy minimization problem, if we fix x and apply DVFS, we have to ensure that the fixed x will still be in the new feasible range after DVFS. We now illustrate this with an example.

EXAMPLE 4.1. *For the same task set and setting as shown in Example 3.1, we can now compute the maximum feasible range $[\hat{x}_{LB}, \hat{x}_{UB}] = [\frac{6}{19}, 1]$. However, if we fix x to 0.5 and the processor frequency to a constant, e.g. $\frac{3}{4} f_{\max}$, then the new feasible range of x becomes $[\frac{6}{13}, 0.6]$. Since our selection of x (0.5) is still in this range, the system is schedulable under EDF-VD for the new setting.*

4.2 Convex problem formulation

We continue to present a convex program formulation of our energy minimization problem (Definition 3.1)

Our objective is the LO mode system energy consumption, which we calculate as the normalized energy consumption in a hyperperiod $\Pi_{\tau_j \in \tau} T_j$ (the actual energy consumption divided by the hyperperiod length). Notice that in such a hyperperiod, the maximum number of jobs of task τ_i is $\frac{\Pi_{\tau_j \in \tau} T_j}{T_i}$, and we can calculate our energy objective as:

$$\begin{aligned} &\frac{1}{\Pi_{\tau_j \in \tau} T_j} \cdot \sum_{\tau_i \in \tau} \frac{\Pi_{\tau_j \in \tau} T_j}{T_i} C_i(\text{LO}) f_b \cdot \frac{1}{f_i^{\text{LO}}} \cdot \beta \cdot (f_i^{\text{LO}})^{\alpha}, \\ &= \sum_{\tau_i \in \tau} \frac{C_i(\text{LO}) f_b}{T_i} \cdot \beta \cdot (f_i^{\text{LO}})^{\alpha - 1}. \end{aligned} \tag{10}$$

The energy minimization is constrained by:

- System schedulability in the LO mode [4]:

$$\frac{\tilde{U}_{\text{HI}}^{\text{LO}}}{x} + \tilde{U}_{\text{LO}}^{\text{LO}} \leqslant 1. \tag{11}$$

- System schedulability in the HI mode (Theorem 2.1):

$$\tilde{U}_{\text{HI}}^{\text{HI}} + x \tilde{U}_{\text{LO}}^{\text{LO}} \leqslant 1. \tag{12}$$

- Maximum feasible range of x (Lemma 4.1):

$$x \in [\hat{x}_{\text{LB}}, \hat{x}_{\text{UB}}]. \tag{13}$$

- Available frequencies:

$$\begin{aligned} \forall \tau_i \in \tau_{\text{HI}}, \forall \chi \in \{\text{HI}, \text{LO}\}, f_i^{\chi} \in [f_{\min}, f_{\max}], \\ \forall \tau_i \in \tau_{\text{LO}}, f_i^{\text{LO}} \in [f_{\min}, f_{\max}]. \end{aligned} \tag{14}$$

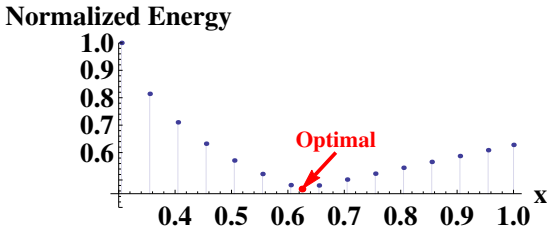


Figure 2: Impact of x on minimal expected energy

Now, we can summarize a complete formulation for the mixed-criticality energy minimization problem:

$$\begin{aligned} &\textbf{minimize} \quad (10), \\ &\textbf{s.t.} \quad (5), (6), (11), (12), (13), (14). \end{aligned} \tag{15}$$

A remark on the above problem formulation is that in the feasible range of x and task frequencies, the problem is a convex programming problem (our energy objective is a convex function of task frequencies and the schedulability constraints can be formulated in standard convex programming form). For space reasons, we leave a detailed convex presentation of our problem to [20]. Due to this convex formulation, existing algorithms (e.g. [21]) can be used to solve our formulated problem. We now show with an example the impact of the deadline shortening factor x on the minimal LO mode energy by solving the formulated problems.

EXAMPLE 4.2. *Consider the same task set and setting as shown in Example 3.1. We use the Mathematica built-in optimization tool [19] to solve the formulated energy minimization problem. In fact, Strategy C as shown in Example 3.1 is the optimal solution, where x equals 0.625. We plot here in Figure 2 the minimal expected energy as a function of the deadline shortening factor x in the maximum feasible range (for each data point, we fix the value of x in our problem formulation and run numerical optimization). As we can see, indeed, there is a gain in energy saving with increasing x. However, if x is bigger than 0.625, then the LO mode energy consumption will increase. This can be intuitively explained: The HI mode schedulability of the system will be jeopardized when x is too big ("less" safety preparation). And to ensure the HI mode schedulability, the system needs to speedup in the LO mode. In other words, energy minimization conflicts with the HI mode schedulability when $x > 0.625$.*

5. AN OPTIMAL SOLUTION ALGORITHM

We have formulated in Section 4 our energy minimization problem (Definition 3.1) as a convex program. Although this makes solving the problem easier by using existing methods (e.g. [22]), theoretical investigations of the MCEM problem are still needed to reveal its problem structure and give insights into this problem. In the following, we will reduce the actual problem space of the MCEM problem, identify its optimality conditions and propose an optimal algorithm to solve it.

5.1 Problem space reduction

We first show that the decision space for the MCEM problem can be reduced, due to convexity of the power function (1) and insights into this problem. In particular, we present the following results for problem space reduction.

Normal workload frequencies. Our first result restricts the frequencies any optimal solution could use for task normal workloads (as defined in (2)).

THEOREM 5.1. *In order to minimize the expected energy consumption for the MCEM problem (Definition 3.1), for task normal workloads, all HI criticality tasks should run with the same frequency f_{HI}^{LO}, and all LO criticality tasks should run with the same frequency f_{LO}^{LO}. In other words, $\forall \tau_i \in \tau_{HI}, f_i^{LO} = f_{HI}^{LO}$, and $\forall \tau_i \in \tau_{LO}, f_i^{LO} = f_{LO}^{LO}$.*

The fundamental reason for Theorem 5.1 is the convexity of the power function. We refer interested readers to detailed proof in [20]. Furthermore, in the LO mode, HI criticality tasks should potentially run on a different frequency than that of LO criticality tasks. This is because stretching the executions of HI criticality tasks will have a different impact on system schedulability due to shortened deadlines.

Extra workload frequencies. We continue to show that, in any optimal solution to the MCEM problem, the frequencies for the extra workloads of HI criticality tasks (as defined in (3)) can also be limited. In fact, $\forall \tau_i \in \tau_{\text{HI}}$, setting the frequency f_i^{HI} (with which τ_i's extra workload runs) the same will not change the optimal energy consumption. The intuitive explanation is as follows: Suppose that in an optimal solution to our problem, $f_i^{\text{HI}} \neq f_j^{\text{HI}}$, where $\{\tau_i, \tau_j\} \subset \tau_{\text{HI}}$. Notice first that modifying f_i^{HI} alone will *only* change $\tilde{U}_{\text{HI}}^{\text{HI}}$, see (5) and (6). Hence, as long as we keep $\tilde{U}_{\text{HI}}^{\text{HI}}$ the same, by e.g. letting $f_i^{\text{HI}} = f_j^{\text{HI}}$, the solution will remain feasible (according to Lemma 4.1, the feasible range of x will not change). As a result, the LO mode energy consumption (10) of the modified solution remains the same (still optimal). Formally, this is summarized as follows.

THEOREM 5.2. *If an optimal solution exists for the MCEM problem (Definition 3.1), where $\exists\{\tau_i, \tau_j\} \subset \tau_{HI}, f_i^{HI} \neq f_j^{HI}$, then by letting $f_i^{HI} = f_j^{HI}$, the optimal energy consumption remains the same.*

Thus, without giving up optimality of the solution, we can fix $f_i^{\text{HI}} (\forall \tau_i \in \tau_{\text{HI}})$ to a same value (denoted as $f_{\text{HI}}^{\text{HI}}$). Summarizing Theorem 5.1 and Theorem 5.2, we have:

$$\begin{aligned} &\forall \tau_i \in \tau_{\text{LO}}, f_i^{\text{LO}} = f_{\text{LO}}^{\text{LO}}, \\ &\forall \tau_i \in \tau_{\text{HI}}, f_i^{\text{LO}} = f_{\text{HI}}^{\text{LO}}, f_i^{\text{HI}} = f_{\text{HI}}^{\text{HI}}. \end{aligned} \tag{16}$$

5.2 Optimality condition

We proceed to study the exact optimality conditions for the MCEM problem. For a given task set τ, let us first define several invariants as follows:

$$\begin{aligned} K &= \sum_{\tau_i \in \tau_{\text{HI}}} \frac{C_i(\text{LO}) f_b}{T_i}, \\ L &= \sum_{\tau_i \in \tau_{\text{LO}}} \frac{C_i(\text{LO}) f_b}{T_i}, \\ M &= 1 - \sum_{\tau_i \in \tau_{\text{HI}}} \frac{(C_i(\text{HI}) - C_i(\text{LO})) f_b}{T_i f_{\max}}. \end{aligned} \tag{17}$$

Based on Theorem 5.1 and Theorem 5.2, let us denote an optimal solution to our problem as x_{opt}, $f_{\text{HI}\,opt}^{\text{LO}}$, $f_{\text{LO}\,opt}^{\text{LO}}$ and $f_{\text{HI}\,opt}^{\text{HI}}$. According to (5), (6) and Lemma 4.1, we can derive the feasible range of x in this case as:

$$\begin{aligned} x_{\text{LB}opt} &= \frac{\frac{K}{f_{\text{HI}\,opt}^{\text{LO}}}}{1 - \frac{L}{f_{\text{LO}\,opt}^{\text{LO}}}}, \\ x_{\text{UB}opt} &= \left[\!\!\left[\frac{1 - \frac{K}{f_{\text{HI}\,opt}^{\text{LO}}} - \frac{(1-M) f_{\max}}{f_{\text{HI}\,opt}^{\text{HI}}}}{\frac{L}{f_{\text{LO}\,opt}^{\text{LO}}}} \right]\!\!\right]^1. \end{aligned} \tag{18}$$

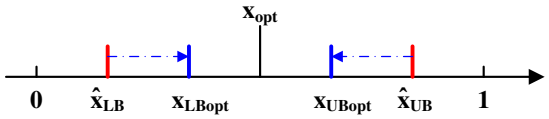


Figure 3: Bounds of x in an optimal solution

Notice that all frequencies ($f_{\text{HI}\,opt}^{\text{LO}}$, $f_{\text{LO}\,opt}^{\text{LO}}$ and $f_{\text{HI}\,opt}^{\text{HI}}$) are less than or equal to $f_{\max}$, and the optimal solution is by definition feasible, i.e. $0 < x_{\text{LB}opt} \leqslant x_{opt} \leqslant x_{\text{UB}opt} \leqslant 1$. Furthermore, according to Lemma 4.1, we have $\hat{x}_{\text{LB}} \leqslant x_{\text{LB}opt} \wedge x_{\text{UB}opt} \leqslant \hat{x}_{\text{UB}}$. ($\tilde{U}_{\chi_1}^{\chi_2}$ is strictly non-decreasing in an optimal solution compared to when the processor frequency is a constant $f_{\max}$. According to Lemma 4.1, we can observe that $x_{\text{LB}opt}$ can not decrease when compared to $\hat{x}_{\text{LB}}$. Similarly, $x_{\text{UB}opt}$ will not increase.) Let us depict the bounds of x and the optimal selection of it in Figure 3.

Four different cases can hence arise:

Case 1: $x_{\text{LB}opt} < x_{opt} < x_{\text{UB}opt}$
In this case, according to (18), we can further reduce $f_{\text{HI}\,opt}^{\text{LO}}$ and/or $f_{\text{LO}\,opt}^{\text{LO}}$ (so that $x_{\text{LB}opt}$ will be increased and $x_{\text{UB}opt}$ will be potentially decreased). This will help to reduce the expected energy consumption in the LO mode. Since we assumed this is the optimal solution, it can only be the case that $f_{\text{HI}\,opt}^{\text{LO}}$ and $f_{\text{LO}\,opt}^{\text{LO}}$ cannot be reduced, i.e. $f_{\text{HI}\,opt}^{\text{LO}} = f_{\text{LO}\,opt}^{\text{LO}} = f_{\min}$. However, manipulating $f_{\text{HI}\,opt}^{\text{HI}}$ to change $x_{\text{UB}opt}$ will not affect the expected energy consumption, which is independent of $f_{\text{HI}\,opt}^{\text{HI}}$, see (10) and (16). Hence, $f_{\text{HI}\,opt}^{\text{HI}}$ can be freely set as long as system schedulability is guaranteed.

Case 2: $x_{\text{LB}opt} < x_{opt} \wedge x_{opt} = x_{\text{UB}opt}$
In this case, if we choose x_{opt} to be smaller such that it still fulfills $x_{\text{LB}opt} < x_{opt}$, the newly obtained solution is still feasible. Since we are not modifying task frequencies, the new solution is still optimal w.r.t. energy minimization. Thus, this case is identical to Case 1.

Case 3: $x_{\text{LB}opt} = x_{opt} \wedge x_{opt} < x_{\text{UB}opt}$
Similar to the argument in Case 2, we can increase x_{opt} without affecting the optimality of the solution. Also this case is identical to Case 1.

Case 4: $x_{\text{LB}opt} = x_{opt} = x_{\text{UB}opt}$
In this case, any reduction of $f_{\text{HI}\,opt}^{\text{LO}}$ or $f_{\text{LO}\,opt}^{\text{LO}}$ will violate the feasibility of the solution (according to (18), $x_{\text{LB}opt}$ will be increased, and $x_{\text{UB}opt}$ will not increase such that $x_{\text{LB}opt} > x_{\text{UB}opt}$). Since we have so far considered all possible cases of an optimal solution, and Case 1, Case 2 and Case 3 are identical, it follows that if an optimal solution does not exist in a lowest energy state (task normal workload frequencies are $f_{\min}$), it must then exist in Case 4.

Formally, we summarize our observations as follows.

THEOREM 5.3. *An optimal solution to the MCEM problem can only exist in two cases:*

1. *At a lowest energy state: frequencies $f_{HI\,opt}^{LO}$ and $f_{LO\,opt}^{LO}$ are set to $f_{\min}$.*
2. *At an "equilibrium" state: $x_{LBopt} = x_{opt} = x_{UBopt}$, where x_{LBopt} and x_{UBopt} are derived according to (18).*

We proceed to show that if an optimal solution exists for our energy minimization problem for some $f_{\text{HI}\,opt}^{\text{HI}}$, $f_{\text{HI}\,opt}^{\text{LO}}$

and $f^{\mathrm{LO}}_{\mathrm{LO}\,opt}$, then an optimal solution with the same minimal LO mode energy also exists when $f^{\mathrm{HI}}_{\mathrm{HI}\,opt}$ is fixed to $f_{\max}$. This is intuitive since with increasing $f^{\mathrm{HI}}_{\mathrm{HI}\,opt}$, the existing optimal solution will still be feasible (the actual HI criticality WCETs of HI criticality tasks are decreased and the system is less loaded). In fact, we have already shown in Example 3.1 that with increasing speedup for the extra workload of HI criticality tasks, the LO mode energy consumption can be reduced.

Formally, we have the following result.

THEOREM 5.4. *For the MCEM problem, with increasing f^{HI}_{HI}, the system's minimal expected energy does not increase.*

As a result of Theorem 5.4, we can fix $f^{\mathrm{HI}}_{\mathrm{HI}}$ to $f_{\max}$ without sacrificing optimality of the solution found. Based on this, the calculation of $x_{\mathrm{UB}\,opt}$ as shown in (18) can be simplified:

$$x_{\mathrm{UB}\,opt} = \left[\!\!\left[\frac{M - \frac{K}{f^{\mathrm{LO}}_{\mathrm{HI}\,opt}}}{\frac{L}{f^{\mathrm{LO}}_{\mathrm{LO}\,opt}}} \right]\!\!\right]^{1}. \tag{19}$$

5.3 Our algorithm

Based on our theoretical results in this section, we now present an algorithm (Algorithm 1) to find the optimal solution. We first check in our algorithm whether a feasible solution exists or not (Line 1-2). This is done by calculating the maximum feasible range of the x. If no such feasible range exists, our algorithm terminates and signals failure. Otherwise, we continue to check the two optimality conditions according to Theorem 5.3 (to search for the optimal solution). According to Theorem 5.4, we can first fix $f^{\mathrm{HI}}_{\mathrm{HI}\,opt}$ to $f_{\max}$ (Line 3). We then check whether a lowest energy solution exists. For such a solution, $f^{\mathrm{LO}}_{\mathrm{HI}\,opt} = f^{\mathrm{LO}}_{\mathrm{LO}\,opt} = f_{\min}$. Hence, we can calculate the bounds of x in this case (Line 4). If the system is feasible for this setting, our algorithm succeeds. Otherwise, at least one "equilibrium" solution must exist (detailed proof in [20]). We continue to find the minimal energy "equilibrium" solution. We can add the "equilibrium" constraint (i.e. $x_{\mathrm{LB}} = x_{\mathrm{UB}} = x$) to our convex program. This yields a single variable constrained minimization problem in a continuous range, which can be solved by standard gradient based methods. Line 9-18 gives our results on solving such a simplified optimization problem (for details, please refer to our proof of Theorem 5.5 in [20]).

Our results in this section can be summarized as follows.

THEOREM 5.5. *Algorithm 1 yields an optimal solution to the MCEM problem if there exists one.*

Remarks. According to Theorem 5.4, computing $f^{\mathrm{HI}}_{\mathrm{HI}\,opt}$ takes $O(1)$ time (Line 3 in Algorithm 1). The time complexity of calculating K, L and M each is pseudo polynomial in the number of total tasks in the system (by (17)). In addition, calculating each of $\hat{x}_{\mathrm{LB}}$, $\hat{x}_{\mathrm{UB}}$, $\check{x}_{\mathrm{LB}}$, and $\check{x}_{\mathrm{UB}}$ takes also time pseudo polynomial in the number of total tasks (by (5), (6), (8) and (9)). Therefore, according to Algorithm 1, computing x_{opt}, $f^{\mathrm{LO}}_{\mathrm{HI}\,opt}$ and $f^{\mathrm{LO}}_{\mathrm{LO}\,opt}$ take pseudo polynomial time in the number of tasks in the system.

We conclude this section with an example to show the application of our proposed algorithm.

EXAMPLE 5.1. *Consider again the task set as shown in Example 3.1 under the same setting. We already calculated in Example 4.1 the maximum feasible range of x as $[\frac{6}{19}, 1]$, hence a feasible solution always exists. According to Algorithm 1, we can first set f^{HI}_{HI} to 1 (normalized $f_{\max}$). We continue to check whether a lowest energy solution exist. This is done by calculating $\check{x}_{LB} = -30$ and $\check{x}_{UB} = -\frac{3}{5}$. Therefore, no lowest energy solution exists and we check for the "equilibrium" condition. According to the algorithm, we can derive $x_{opt} = 0.625$, $f^{LO}_{HI\,opt} = 0.65$, and $f^{LO}_{LO\,opt} = 0.54$.*

Algorithm 1: Pseudo code – Find the optimal solution

Input: τ, f_b, $f_{\min}$, $f_{\max}$
Output: $f^{\mathrm{HI}}_{\mathrm{HI}\,opt}, f^{\mathrm{LO}}_{\mathrm{HI}\,opt}, f^{\mathrm{LO}}_{\mathrm{LO}\,opt}, x_{opt}$

1 Calculate $\hat{x}_{\mathrm{LB}}$ and $\hat{x}_{\mathrm{UB}}$ when the processor frequency is a constant $f_{\max}$ according to Lemma 4.1 ((8) and (9));
2 **if** $0 < \hat{x}_{LB} \leqslant \hat{x}_{UP} \leqslant 1$ **then**
3 $\quad f^{\mathrm{HI}}_{\mathrm{HI}\,opt} \leftarrow f_{\max}$;
4 $\quad$ Calculate according to (8) and (9) x_{LB} and x_{UB} when $f^{\mathrm{LO}}_{\mathrm{HI}} = f^{\mathrm{LO}}_{\mathrm{LO}} = f_{\min}$, denoted as $\check{x}_{\mathrm{LB}}$ and $\check{x}_{\mathrm{UB}}$;
5 $\quad$ **if** $0 < \check{x}_{LB} \leqslant \check{x}_{UB} \leqslant 1$ **then**
6 $\quad\quad f^{\mathrm{LO}}_{\chi\ opt} \leftarrow f_{\min}$ $(\chi \in \{\mathrm{HI}, \mathrm{LO}\})$;
7 $\quad\quad$ Set x_{opt} as any value in $[\check{x}_{\mathrm{LB}}, \check{x}_{\mathrm{UB}}]$;
8 $\quad$ **else**
9 $\quad\quad$ Set $\check{f}^{\mathrm{LO}}_{\mathrm{LO}} = \left[\!\!\left[\frac{L}{1 - \frac{K}{M f_{\max}}} \right]\!\!\right]_{f_{\min}}$;
10 $\quad\quad$ **if** $f_{\min} > K/M$ **then**
11 $\quad\quad\quad$ Set $\hat{f}^{\mathrm{LO}}_{\mathrm{LO}} = \left[\!\!\left[\frac{L}{1 - \frac{K}{M f_{\min}}} \right]\!\!\right]^{f_{\max}}$;
12 $\quad\quad$ **else**
13 $\quad\quad\quad$ Set $\hat{f}^{\mathrm{LO}}_{\mathrm{LO}} = f_{\max}$;
14 $\quad\quad$ **end**
15 $\quad\quad x_{opt} \leftarrow M$;
16 $\quad\quad f^{\mathrm{LO}}_{\mathrm{LO}\,opt} \leftarrow [\![K \cdot M^{-\frac{\alpha-1}{\alpha}} + L]\!]^{\hat{f}^{\mathrm{LO}}_{\mathrm{LO}}}_{\check{f}^{\mathrm{LO}}_{\mathrm{LO}}}$;
17 $\quad\quad f^{\mathrm{LO}}_{\mathrm{HI}\ opt} \leftarrow \frac{K}{M \cdot (1 - L/f^{\mathrm{LO}}_{\mathrm{LO}\,opt})}$;
18 $\quad$ **end**
19 **else**
20 $\quad$ **return** Failure;
21 **end**
22 **return** Success;

6. DISCUSSION

We proceed in this section to investigate some tradeoffs for the MCEM problem, as well as extensions of our proposed techniques.

6.1 Extra workload and expected energy consumption

Intuitively, if extra workloads for HI criticality tasks are increased, then more time budgets should be reserved for the HI mode instead of being explored to save the LO mode energy. This is a meaningful tradeoff, as for increased extra workloads to accommodate, the system timing safety could be enhanced. However, this is at the cost of increased LO mode energy. Formally, this is summarized as follows.

LEMMA 6.1. *If the extra workload $(C_i(HI) - C_i(LO))f_b$ is increased for some HI criticality task τ_i, then the minimal expected energy consumption of the system is strictly non-decreasing.*

Notice that, with increasing extra workload, the system could still be feasible when the processor's frequency is the lowest ($f_{\min}$). In this case, the minimal expected energy will

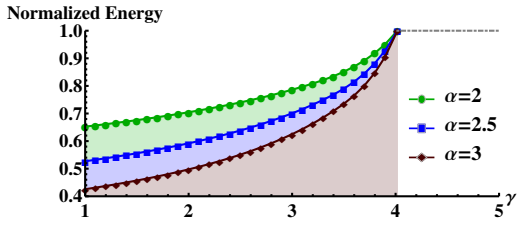


Figure 4: FMS: impact of extra workload (modeled by parameter $\gamma = \frac{C_i(\text{HI})}{C_i(\text{LO})}$) on energy minimization, $f_{\min} = 0.2$ (results computed by Algorithm 1)

remain the same. Otherwise, the minimal expected energy strictly increases with increasing extra workload.

6.2 Extension – Discrete frequency levels

Often, frequency levels may not be continuously available. In this case, we have to consider discrete frequency levels. Due to this limitation, our theoretical results for energy minimization in the continuous case will not hold anymore. However, we can still formulate the expected energy minimization problem as a mixed-integer convex program.

Let us assume that, the list of available frequencies in increasing order is $F = \{f_1, f_2, \cdots, f_{|F|}\}$, where $f_1 = f_{\min}$ and $f_{|F|} = f_{\max}$. Then, for each LO criticality task, we have to decide the number of clock cycles it spends on each frequency level. In addition, for each HI criticality task τ_i, we have to decide for its normal workload, the number of clock cycles it spends on each frequency level (similarly for its extra workload). And the system schedulability needs to be guaranteed for both modes. Based on the above discussions, we can modify our problem formulation in Section 4.2 and formulate a mixed integer convex program (the number of clock cycles can only be integers).

An alternative solution is to apply the techniques in this paper and use discrete frequency levels to "mimic" the frequencies returned by our techniques (details in [20]).

To further account transition overheads in timing and energy when switching between different frequency levels, a solution could be obtained by integrating existing techniques, e.g. [23], with our proposed techniques. We leave this to our future work.

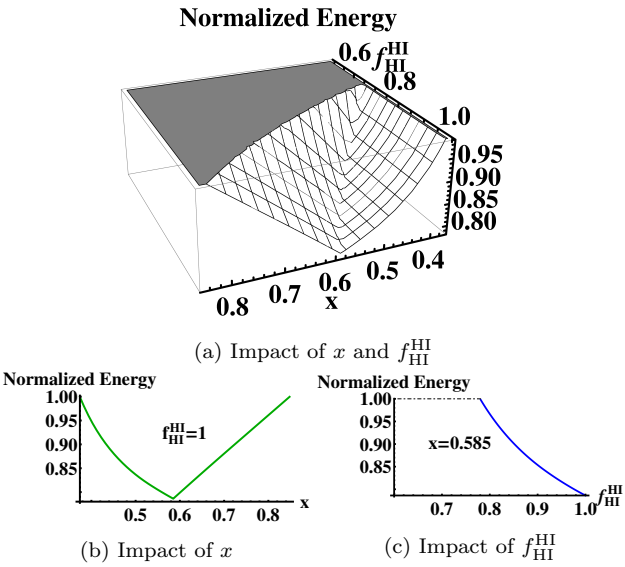
7. EVALUATION

We evaluate now our proposed techniques with an avionics use-case and extensive simulations on synthetic task sets. Energy savings and various tradeoffs are illustrated by our results. For all our experiments, we assume task WCETs in our system model are measured on the maximum frequency (i.e. $f_b = f_{\max}$). We normalize the LO mode energy consumption when all tasks run on $f_{\max}$ to 1.

7.1 Flight management system

Our first set of experiments are conducted on a subset of the flight management system (FMS), which consists of 7 HI criticality tasks and 4 LO criticality tasks (detailed parameters can be found in [13]). We apply our proposed techniques to FMS and tune relative parameters to evaluate their impacts on the minimal expected energy.

Impact of extra workload. We first show the impact of extra workload on the minimal LO mode energy. For this set of experiments, we use a factor γ ($\geqslant 1$) to uniformly model the extra workload – $\forall \tau_i \in \tau_{\text{HI}}, \frac{C_i(\text{HI})}{C_i(\text{LO})} = \gamma$. If $\gamma = 1$, then



(a) Impact of x and $f_{\text{HI}}^{\text{HI}}$

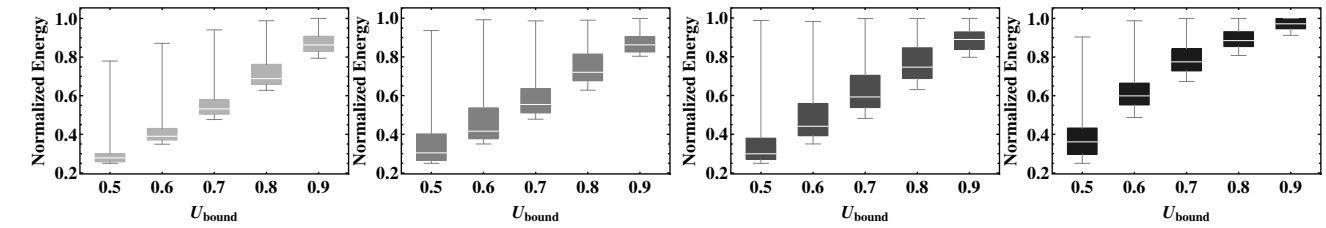
(b) Impact of x

(c) Impact of $f_{\text{HI}}^{\text{HI}}$

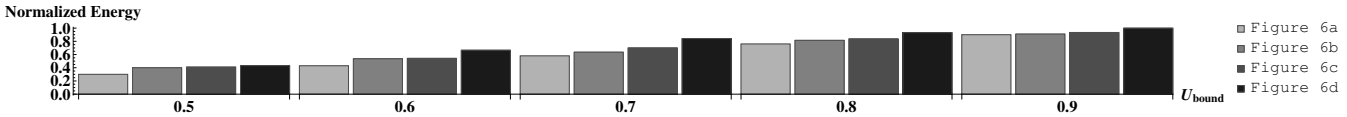
Figure 5: FMS: impact of x and $f_{\text{HI}}^{\text{HI}}$ on energy minimization, $f_{\min} = 0.2$, $\alpha = 2$, $\gamma = 3$. Gray area in Figure 5a indicates that FMS is not schedulable (similar in Figure 5c, dot-dashed area). Results are derived by fixing the corresponding parameters in our problem formulation and applying numerical optimization.

HI criticality tasks could never exceed their LO criticality WCETs. However, if there is a safety concern for HI criticality tasks ($\gamma > 1$), then we have to reserve time budget to accommodate the extra workload of HI criticality tasks. This prevents us from exploring all available slack in the LO mode to save energy. Figure 4 shows how the extra workload limits energy savings for FMS. As we can see, with increased extra workload (bigger γ), the minimal expected energy is increased, which matches our analysis. This trend stops after $\gamma > 4.02$, in which case the system is infeasible even if tasks run on $f_{\max}$ (dash-dotted line in Figure 4). Furthermore, we can also observe that, for increasing α, the energy saving is also increased (according to the power function (1), larger α means more energy saving by DVFS).

Impacts of x and $f_{\text{HI}}^{\text{HI}}$. We continue to present the impacts of x and $f_{\text{HI}}^{\text{HI}}$ on energy minimization. As discussed in Section 4, with increasing deadline shortening factor x (larger deadlines for HI criticality tasks in the LO mode), we can have more slack to apply DVFS in the LO mode for better energy savings. However, large x could jeopardize the system schedulability in the HI mode (see (12)). This implies that there is a limit on energy saving by increasing x due to the constraint of the HI mode schedulability. In addition, with higher speedup for extra workload (i.e. increasing $f_{\text{HI}}^{\text{HI}}$), less time budget needs to be reserved to handle overrun and more energy savings can be achieved (see Theorem 5.4). Figure 5 illustrates the impacts of both factors. Indeed, our results show that, with increasing x, the expected energy consumption can be reduced. However, when x is too large, the expected energy will increase (e.g. when $f_{\text{HI}}^{\text{HI}} = 1$, the expected energy consumption will increase when $x > 0.585$, as shown in Figure 5b). In addition, our results confirm that for increased $f_{\text{HI}}^{\text{HI}}$, the expected energy consumption can be reduced. For example, when $x = 0.585$, the LO mode energy consumption will continuously decrease when $f_{\text{HI}}^{\text{HI}} > 0.78$ (as shown in Figure 5c).



(a) $\mathcal{P}_{\text{HI}} = 0.2, \gamma = 2$ (b) $\mathcal{P}_{\text{HI}} = 0.2, \gamma = 8$ (c) $\mathcal{P}_{\text{HI}} = 0.4, \gamma = 2$ (d) $\mathcal{P}_{\text{HI}} = 0.2, \gamma = 2$



(e) Bar-chart plot of our results in all cases for the 3rd quartile (75%) data points in the box-whisker plots

Figure 6: Box-whisker plot of experimental results for random task sets, with $u_- = 0.01, u_+ = 0.2, T_- = 200$ ms, $T_+ = 2$ s, $\alpha = 3$, $f_{\min} = 0.5$. For Figure 6a, Figure 6b, and Figure 6c, we apply our proposed method (Algorithm 1). For Figure 6d, we apply discrete DVFS by adapting our method (details in [20]), assuming only $f_{\min}$ and $f_{\max}$ available.

7.2 Extensive simulations

In order to validate our proposed techniques on general task sets, we now apply them to randomly generated task sets. We adopt a similar random task generator as used in [4]. The random task generation is controlled by the following parameters:

- $[u_-, u_+]$: the LO mode utilization of any task τ_i, $\frac{C_i(\text{LO})}{T_i}$, is uniformly drawn from this range: $0 < u_- < u_+ \leqslant 1$;
- U_{bound}: the total LO mode utilization, which is defined as $U_{\text{bound}} = \sum_{\tau_i} \frac{C_i(\text{LO})}{T_i}$;
- $[T_-, T_+]$: task minimal inter-arrival times are uniformly drawn from this range;
- γ: the ratio of $C_i(\text{HI})$ to $C_i(\text{LO})$ for any HI criticality task;
- $\mathcal{P}_{\text{HI}}$: the probability that a task is a HI criticality task.

The random task generator starts with an empty task set and incrementally adds new random tasks into this set until certain system utilization U_{bound} is reached. For each data point (a specific system utilization U_{bound}), we generate 200 random task sets and conduct our experiments. Furthermore, we omit infeasible task sets at each data point.

We summarize our results in Figure 6, where we observe:

- With increasing utilization of the system (U_{bound}), the expected energy saving is decreased. This is intuitive as we have less time slack under higher system utilization. Therefore, less energy savings can be achieved by exploring time slack to stretch task executions. This is confirmed by all our results in Figure 6, e.g. the middle quartile (average) LO mode energy consumptions increase with increasing U_{bound}.
- With increasing extra workload, the expected energy saving is decreased. This can be observed by comparing Figure 6a to Figure 6b – when γ is increased from 2 to 8, the expected energy saving is dropped. For example, with $U_{\text{bound}} = 0.5$, the maximum LO mode energy consumption is 0.78 when $\gamma = 2$. This is increased to 0.94 when $\gamma = 8$. Our results here match our analysis in Lemma 6.1.
- With more HI criticality tasks, the expected energy saving is reduced (compare Figure 6a to Figure 6c, where $\mathcal{P}_{\text{HI}}$ is increased from 0.2 to 0.4). For example, with $U_{\text{bound}} = 0.8$, the average LO mode energy consumption is 0.69 when $\mathcal{P}_{\text{HI}} = 0.2$. This is increased to 0.75 when $\mathcal{P}_{\text{HI}} = 0.4$. The intuitive explanation is as follows: with more HI criticality tasks, the extra workloads for those tasks are also increased, hence more time budget needs to be reserved for those extra workload instead of being explored to save energy.
- With the restriction of discrete frequency levels, the expected energy saving is reduced (compare Figure 6a to Figure 6d). Here, we obtain our results with discrete frequency levels by using them to "mimic" the frequency levels calculated by our algorithm (for details, please refer to [20]). According to our results, with continuous frequency levels, the average LO mode energy consumption when $U_{\text{bound}} = 0.7$ is 0.53. This is increased to 0.78 under discrete frequency levels. However, considerable energy reductions can still be achieved by adapting our proposed techniques, e.g. when $U_{\text{bound}} = 0.8$, we can still achieve on average 12% energy reduction according to Figure 6d.

For the purpose of clear comparison between different cases, we present a bar-chart plot of our results in Fig 6e (only for the 3rd quartile (75%) of the box-whisker plots). The trend becomes evident: with increasing extra workload, more HI criticality tasks or the limitation of discrete frequency levels, the energy saving is reduced.

8. RELATED WORK

We survey in this section the main results in two research areas related to this work: mixed-criticality scheduling and system level energy minimization.

8.1 Mixed-Criticality Scheduling

To date, a common model exists in the literature to specify mixed-criticality systems [24, 25, 8, 5, 6, 13]. In this model, varying degrees of timing assurance for tasks of different criticality levels are considered: All task WCETs are modeled on all existing criticality levels, with the WCET on a higher criticality being more pessimistic. At runtime, whenever any task violates its WCET on some criticality level, all tasks with criticality levels no higher than this criticality level are dropped to guarantee the more critical tasks. Different scheduling techniques (e.g. fixed priority [26, 3], earliest deadline first (EDF) [6, 7, 8], and time-triggered [9]) are extended to this mixed-criticality setting.

However, the current research on mixed-criticality systems has primarily focused on real-time guarantees [24, 25, 8, 5, 6,

13] and functional safety [27]. To the best of our knowledge, energy minimization for mixed-criticality systems has not been explored yet.

8.2 Energy minimization techniques

Energy-aware embedded system design has been established for many years. The system level energy minimization techniques can be broadly categorized into dynamic voltage and frequency scaling (DVFS) [28, 29, 30] and dynamic power management (DPM) [31, 32]. DVFS minimizes the dynamic energy consumption due to circuit switching activities by reducing the processor voltage and frequency levels. In contrast, DPM minimizes the static energy consumption due to leakage current by turning off the processor or switching the processor to sleep mode. Both DVFS and DPM have been extensively studied in the literature. DVFS can be further categorized into offline [28] and online [29]. For offline techniques, task frequencies are assigned a priori, while for online techniques, the system adapts the DVFS strategy in response to runtime events (e.g. temperature variation [29]). For a comprehensive survey on energy minimization techniques, we refer the readers to [32, 14].

Notice, however, that conventional DVFS techniques are designed out of the context of mixed-criticality. Due to energy concerns of mixed-criticality systems, it is important to extend the research on DVFS to those systems.

9. CONCLUSION

We solve in this paper the energy minimization problem for mixed-criticality systems, where critical tasks must meet their deadlines even if they exceed their expected WCETs. We investigate the characteristics of this problem and show there exists a conflict between safety and energy minimization. We further show that speeding up the system to handle overrun is beneficial for minimizing the expected energy consumption of the system. We integrate continuous DVFS with the EDF-VD scheduling technique and formulate our energy minimization problem as a convex program. Furthermore, we provide an optimal algorithm to find the solution to our problem. Tradeoffs and extensions for our techniques are discussed. The proposed techniques are validated by both an industrial use-case and extensive simulations.

10. REFERENCES

[1] "Mixed criticality systems." `http://cordis.europa.eu/fp7/ict/embedded-systems-engineering/documents/sra-mixed-criticality-systems.pdf`.

[2] K. P. Valavanis and K. P. Valavanis, *Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy*. Springer Publishing Company, Incorporated, 2007.

[3] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in *RTSS*, pp. 34–43, 2011.

[4] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *ECRTS*, pp. 145–154, 2012.

[5] F. Santy, L. George, P. Thierry, and J. Goossens, "Relaxing mixed-criticality scheduling strictness for task sets scheduled with fp," in *ECRTS*, pp. 155–165, 2012.

[6] T. Park and S. Kim, "Dynamic scheduling algorithm and its schedulability analysis for certifiable dual-criticality systems," in *EMSOFT*, pp. 253–262, 2011.

[7] S. K. Baruah, V. Bonifaci, G. DâĂŹAngelo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, "Mixed-criticality scheduling of sporadic task systems," in *Algorithms–ESA 2011*, pp. 555–566, Springer, 2011.

[8] P. Ekberg and W. Yi, "Bounding and shaping the demand of mixed-criticality sporadic tasks," in *ECRTS*, pp. 135–144, 2012.

[9] S. Baruah and G. Fohler, "Certification-cognizant time-triggered scheduling of mixed-criticality systems," in *RTSS*, pp. 3–12, 2011.

[10] "Automotive electronics." `http://en.wikipedia.org/wiki/Automotive_electronics`.

[11] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *Real-Time and Embedded Technology and Applications Symposium*, pp. 131–140, April 2009.

[12] S. Baruah, "Mixed criticality schedulability analysis is highly intractable," 2009.

[13] P. Huang, G. Giannopoulou, N. Stoimenov, and L. Thiele, "Service adaptions for mixed-criticality systems," in *ASP-DAC*, pp. 125–130, 2014.

[14] J.-J. Chen and C.-F. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms," in *International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 28–38, IEEE, 2007.

[15] S. Pagani and J.-J. Chen, "Energy efficiency analysis for the single frequency approximation (sfa) scheme," in *RTCSA*, pp. 82–91, 2013.

[16] D. Zhu, R. Melhem, and D. Mossé, "The effects of energy management on reliability in real-time embedded systems," in *IEEE/ACM International Conference on Computer Aided Design*, pp. 35–40, IEEE, 2004.

[17] A. Nelson, O. Moreira, A. Molnos, S. Stuijk, B. Nguyen, and K. Goossens, "Power minimisation for real-time dataflow applications," in *Digital System Design (DSD)*, pp. 117–124, Aug 2011.

[18] S. Pagani and J.-J. Chen, "Energy efficient task partitioning based on the single frequency approximation scheme," in *RTSS*, pp. 308–318, Dec 2013.

[19] "Constrained optimization." `https://www.wolfram.com/technology/guide/ConstrainedNonlinearOptimization/`.

[20] P. Huang, P. Kumar, G. Giannopoulou, and L. Thilele, "Energy efficient dvfs scheduling for mixed-criticality systems," Tech. Rep. 354, ETH Zurich, Laboratory TIK, July 2014.

[21] Y. E. Nesterov and M. J. Todd, "Self-scaled barriers and interior-point methods for convex programming," *Mathematics of Operations research*, vol. 22, no. 1, pp. 1–42, 1997.

[22] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[23] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi, "Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, (Washington, DC, USA), 2004.

[24] S. Baruah and S. Vestal, "Schedulability analysis of sporadic tasks with multiple criticality specifications," in *ECRTS*, pp. 147–155, 2008.

[25] H. Li and S. Baruah, "Load-based schedulability analysis of certifiable mixed-criticality systems," in *EMSOFT*, pp. 99–108, 2010.

[26] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *RTSS*, pp. 239–243, 2007.

[27] P. Huang, H. Yang, and L. Thiele, "On the scheduling of fault-tolerant mixed-criticality systems," in *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*, pp. 131:1–131:6, 2014.

[28] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced cpu energy," in *36th Annual Symposium on Foundations of Computer Science*, pp. 374–382, Oct 1995.

[29] J. Tschanz, N. S. Kim, S. Dighe, J. Howard, G. Ruhl, S. Vangal, S. Narendra, Y. Hoskote, H. Wilson, C. Lam, M. Shuman, C. Tokunaga, D. Somasekhar, S. Tang, D. Finan, T. Karnik, N. Borkar, N. Kurd, and V. De, "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging," in *International Solid-State Circuits Conference*, pp. 292–604, Feb 2007.

[30] P. Huang, O. Moreira, K. Goossens, and A. Molnos, "Throughput-constrained voltage and frequency scaling for real-time heterogeneous multiprocessors," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pp. 1517–1524, 2013.

[31] L. Benini, A. Bogliolo, A. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 813–833, Jun 1999.

[32] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, pp. 299–316, June 2000.