Exploring Energy Saving for Mixed-Criticality Systems on Multi-cores

Sujay Narayana[†‡], Pengcheng Huang[†], Georgia Giannopoulou[†], Lothar Thiele[†] and R.Venkatesha Prasad[‡]
[†]Computer Engineering and Networks Laboratory, ETH Zurich, 8092 Zurich, Switzerland
[‡]Embedded Software Group, TU Delft, The Netherlands

***Abstract*—In this paper we study a general energy minimization problem for mixed-criticality systems on multi-cores, considering different system operation modes, and static & dynamic energy consumption. While making global scheduling decisions, trade-offs in energy consumption between different modes and also between static and dynamic energy consumption are required. Thus, such a problem is challenging. To this end, we first develop an optimal solution analytically for unicore and a corresponding low-complexity heuristic. Leveraging this, we further propose energy-aware mapping techniques and explore energy savings for multi-cores. To the best of our knowledge, we are the first to investigate mixed-criticality energy minimization in such a general setting. The effectiveness of our approaches in energy reduction is demonstrated through both extensive simulations and a realistic industrial application.**

I. INTRODUCTION

Energy minimization is a prime requirement in the design of embedded systems, not only due to cost reduction reasons, but also due to related thermal issues. With aggressive shrinking of circuit footprint, power density of modern electronic circuits is drastically increasing, leading to rise in temperature. This may result in overheating, causing the failure of embedded systems if it is not handled properly [12, 29].

Recently, we are witnessing the rise of mixed-criticality systems [2]. Such systems are emerging due to strong requirements from major industries like automotive and avionics, with the essential idea to consolidate functionalities of different safety criticality levels onto the same commercial-off-the-shelf (COTS) computing platform. Consequently, the number of computer systems could be greatly reduced, leading to compact systems that may be potentially easier to design, integrate and validate [6].

Although there is rich literature on energy minimization techniques for conventional real-time safety-critical systems, (see e.g. [5, 8] for excellent reviews), applying them to mixed-criticality systems might not be straightforward. On one hand, well-established methods like Dynamic Voltage and Frequency Scaling (DVFS) or Dynamic Power Management (DPM) can be applied to mixed-criticality systems. On the other hand, mixed-criticality systems can react to runtime threats (majorly task overruns) by switching between different operation modes and making different service guarantees on different criticality levels. This could consequently complicate energy reduction decisions since, usually there exists a trade-off between energy consumption in different modes [16].

To date, only a few works have been published to tackle this challenging problem [15, 16, 21, 28]. An optimal solution based on DVFS to minimize energy consumption for unicore mixed-criticality systems is proposed in [16]. However, only dynamic power consumption due to circuit switching activities in one system mode is considered. Static energy consumption due to leakage current and other system modes are neglected. Other related works, although have different assumptions, all share a common concept to sacrifice the performance of low criticality tasks in order to manage/optimize system energy. An approach based on DPM is presented in [21] to trade deadlines of low criticality tasks for energy saving on multi-cores, while deadlines of high criticality tasks are always guaranteed. A DVFS method minimizing energy for unicore mixed-criticality systems is presented in [15], while respecting system reliability requirements. Rather than treating energy as an optimization goal, energy utilization on critical tasks is advocated in [28] when the system is short of energy supply, while allowing deadline misses of low criticality tasks.

In this paper, we present a solution to a general energy minimization problem for multi-core mixed-criticality systems. Focusing on DVFS, we consider both static and dynamic energy consumption and their trade-offs between different system operation modes. We first propose an optimal solution and an effective lightweight heuristic on a unicore. We then extend these results to multi-core systems by designing new energy-aware mapping techniques. Experimental results for our proposed techniques demonstrate energy savings up to 35% on random simulations and 50% for a flight management system.

The rest of this paper is organized as follows: In Sec. II we describe our system model, while in Sec. III we present an example to motivate our problem formulation. We present an optimal analytical solution to our energy minimization problem on a unicore in Sec. IV, while in Sec. V we present a corresponding simple, yet effective, heuristic solution. Leveraging our unicore solutions, we present different energy-aware task mapping techniques in Sec. VI. We evaluate and conclude our proposed techniques in Sec. VII and Sec. VIII.

II. SYSTEM MODEL

We introduce in this section our system model regarding mixed-criticality and DVFS while briefly discussing mixed-criticality scheduling.

A. Mixed-Criticality Task Model

We consider a mixed-criticality task set τ consisting of independent sporadic tasks $\{\tau_1, \tau_2, ..., \tau_{|\tau|}\}$ to be scheduled on a platform with identical preemptive processors $\pi = \{\pi_1, \pi_2, ..., \pi_{|\pi|}\}$. Each task τ_i is specified by a minimum inter-arrival time T_i, a deadline D_i, and an associated criticality level of χ_i. We assume tasks have implicit deadlines ($\forall \tau_i$, $D_i = T_i$) and dual-criticality levels ($\forall \tau_i$, χ_i can be either high (HI) or low (LO)). In addition, to ensure timing safety, worst-case execution time (WCET) estimations of HI criticality tasks are more conservative than those of LO criticality tasks.

τ	a mixed criticality task set
χ	criticality level, $\chi \in \{\text{LO}, \text{HI}\}$
$C_i(\chi)$	WCET of task τ_i on χ criticality level
f_b	frequency at which WCETs are estimated
$[f_{\min}, f_{\max}]$	available frequency range with DVFS
f_i^{χ}	frequency of task τ_i in χ mode
$f_{\chi_1}^{\chi_2}$	frequency of all χ_1 criticality tasks in χ_2 mode
$f_{\chi_1\,\text{opt}}^{\chi_2}$	optimal $f_{\chi_1}^{\chi_2}$ for unicore energy minimization
$U_{\chi_1}^{\chi_2}$	χ_2 mode utilization of all χ_1 criticality tasks
E_{χ}	system energy consumption in χ mode
w_{χ}	weight factor for χ mode
x	deadline shortening factor in EDF-VD
x_{LB}, x_{UB}	lower and upper bounds of x
$\hat{x}_{\text{LB}}$, $\hat{x}_{\text{UB}}$	lower and upper bounds of x at $f_i^{\chi} = f_{\max}$
x_{LBopt}, x_{UBopt}	optimal lower and upper bounds of x

TABLE I: Adopted notations

To further improve resource efficiency, the state of the art mixed-criticality model [2] assumes to measure task WCETs on all criticality levels. Any HI criticality task τ_i has a LO criticality WCET $C_i(\text{LO})$ and a more pessimistic HI criticality WCET $C_i(\text{HI})$. Any LO criticality task τ_i has only a LO criticality WCET $C_i(\text{LO})$ and is not allowed to overrun $C_i(\text{LO})$. At runtime, the system starts with LO operation mode and guarantees deadlines for all HI and LO criticality tasks assuming their LO criticality WCETs. If any HI criticality task overruns its LO criticality WCET, then the system switches to HI operation mode and all LO criticality tasks are dropped in order to guarantee HI criticality tasks. However, the system can switch back to LO mode at any time when there are no pending tasks [26]. We do not discuss the last scenario as it is beyond the scope of this paper. For notational convenience, we define $U_{\chi_1}^{\chi_2}$ for $\chi_1, \chi_2 \in \{\text{LO}, \text{HI}\}$ as follows:

$$U_{\chi_1}^{\chi_2} = \sum_{\tau_i \in \tau \wedge \chi_i = \chi_1} \frac{C_i(\chi_2)}{T_i}.$$

$U_{\chi_1}^{\chi_2}$ denotes the total utilization of all χ_1 criticality tasks with their χ_2 criticality WCETs. For instance, $U_{\text{HI}}^{\text{LO}}$ denotes the utilization of HI criticality tasks with their LO criticality WCETs. We further define τ_{χ} as the set of all χ criticality tasks, where $\chi \in \{\text{LO}, \text{HI}\}$, and use $[\![a]\!]^c$ to represent $\min(a, c)$. All important notations are summarized in Table I.

B. Power Model and DVFS

We adopt a popular power model from [9, 24]. Assuming a homogeneous multicore platform, the power consumption of any processor is formulated as,

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

where $P(f)$ is the total power consumed and P_s stands for the static power consumption due to leakage current. f denotes the operating frequency of the processor and βf^{α} represents the dynamic power consumption caused by switching activities, where α and β are circuit dependent parameters. A common assumption is that $\alpha \geq 2$ [23, 25]. Hence, decreasing processor frequency leads to a convex reduction of dynamic power, as explored by the well-known DVFS technique [8]. However, with reduced frequency, leakage energy will increase as it takes longer for jobs to complete. Thus there is a critical frequency f_{crit} below which it is not beneficial to reduce frequency energy-wise. For any job with workload of nc clock cycles, [25] shows that f_{crit} can be obtained as follows:

$$\frac{d(\frac{nc}{f}P_s + \frac{nc}{f}\beta f^{\alpha})}{df} = 0 \Leftrightarrow f_{\text{crit}} = \sqrt[\alpha]{\frac{P_s}{\beta(\alpha-1)}}. \tag{2}$$

As a result, we assume in this paper that each core in a hardware platform is independently DVFS-capable and can execute with any frequency in $[f_{\min}, f_{\max}]$, where $f_{\min} \geq f_{\text{crit}}$. The WCETs of tasks are estimated at frequency f_b, where $f_{\min} \leq f_b \leq f_{\max}$. Notice that applying DVFS changes the actual WCETs of tasks, such that χ criticality WCET of task τ_i becomes $\dfrac{C_i(\chi)f_b}{f}$ while running at frequency f. Note that we focus on processing energy consumed by CPUs in this paper and neglect communication energy consumed by memory systems, since the former is the major energy source for modern computer systems and can amount to 75%-80% of total system power consumption [10, 27]. Under realistic models of task execution times and system energy when considering communication [30], our proposed techniques could be directly extended to minimize additionally communication energy by treating the communication phases of tasks as separate artificial tasks. Finally, we assume that the processor energy consumption when inactive and the overheads of switching between active and inactive states are negligible. This assumption holds for platforms with deep-sleep modes and negligible power mode transition overheads (e.g. current NXP Kinetis series feature $\sim 4\mu s$ wakeup time from the state-retaining deep-sleep mode [1]). We leave the complete solution with consideration of various overheads as our future work. However, we believe that the important findings of this paper (e.g. energy trade-offs between different systems modes and between static and dynamic energy consumptions, as well as the benefits of isolated task mapping, see Sec. IV and Sec. VI) would still hold.

C. Mixed-Criticality Scheduling

For mixed-criticality multi-core scheduling, either global scheduling [22] or partitioned scheduling [18] can be applied. In this paper, we focus on the latter since it is more common in industrial embedded systems. Mixed-criticality scheduling is strongly *NP*-hard even for simple task models on a uniprocessor [18]; thus, we particularly study the integration of decisions regarding energy saving into a well-known approach, i.e., the partitioned EDF-VD scheduling [4, 13]. Here, HI criticality tasks are first mapped to all processors followed by mapping of LO criticality tasks. Different bin packing techniques can be adopted for task mapping, while system utilization bounds are enforced on all cores to obtain a feasible schedule [4, 13].

After task mapping, scheduling on each processor follows EDF-VD [3]. In EDF-VD, deadlines of all HI criticality tasks are down-scaled by a multiplication factor x $(0 < x \leq 1)$ in LO mode to prioritize their executions. This will leave enough time until their actual deadlines to accommodate extra workload (overrun). Intuitively, a smaller x increases system utilization in LO mode but decreases system utilization in HI mode, as more jobs are completed in LO mode. As a result, it affects scheduling in both LO and HI modes. For a task set τ *running on a unicore*, a feasible range of x exists [3]. Formally, we present this as follows.

Theorem 1. To guarantee system schedulability on a unicore under EDF-VD, x must be set in the following range,

$$0 < \frac{U_{\text{HI}}^{\text{LO}}}{1 - U_{\text{LO}}^{\text{LO}}} \le x \le \left[\!\left[\frac{1 - U_{\text{HI}}^{\text{HI}}}{U_{\text{LO}}^{\text{LO}}} \right]\!\right]^1 .$$

Proof: This follows from (3) and Theorem 2 in [3]. ■

Essentially, this theorem states that there is a lower bound on x, below which LO mode will not be schedulable. Similarly, an upper bound exists, beyond which HI mode will not be schedulable. However, notice that with DVFS, system utilization factors are scaled, and the test in Theorem 1 needs to be performed with those of scaled utilization factors.

III. Problem Formulation

We motivate our work using a general setting for mixed-criticality energy minimization on multi-cores, where both static and dynamic power consumption are considered in all system modes. Based on this, we provide a concrete problem formulation here.

A. Motivation

Minimizing system energy only in one mode may not be proper as the system can operate in different modes: [16] advocates to consider only LO mode energy as they claim the system is most likely to stay in this mode. However, with time the probability of switching from LO to HI mode increases, eventually reaching 1 (see e.g. [17]). To jointly consider energy consumption in different system modes, we define the weight factor w_{LO} (w_{HI}) for LO (HI) mode, which indicates the relative importance of minimizing system energy in each mode. We assume normalized weight factors (i.e. $w_{\text{LO}} + w_{\text{HI}} = 1$). Such weight factors would help to make global energy saving decisions to trade-off energy consumptions in different system modes. In practice, one could interpret them as the percentages of time (or alternatively, the probabilities) that the system operates in the corresponding modes. The motivation to consider static energy is rather straightforward – static power could overwhelm dynamic power as the CMOS technology level improves [19]. Thus, it is critical to minimize static energy along with dynamic energy [11].

We use the following motivational example to quantify the impact of weight factors and static energy on energy minimization. We first evaluate the case when the system runs on a unicore and then proceed to multi-cores with an additional consideration of task mapping strategies.

Example 1. Consider the task set as shown in Table II, which is schedulable on a unicore under EDF-VD (base frequency $f_b = 1.2$GHz). The processor is DVFS-capable, where $[f_{\min}, f_{\max}] = [0.7, 1.2]$GHz. We assume $\alpha = 3$, $\beta = 1\text{W/GHz}^\alpha$ and $P_s = 0.8$W [25]. We calculate the total weighted energy consumption in LO and HI modes with varying weight factors w_{LO} and w_{HI}.

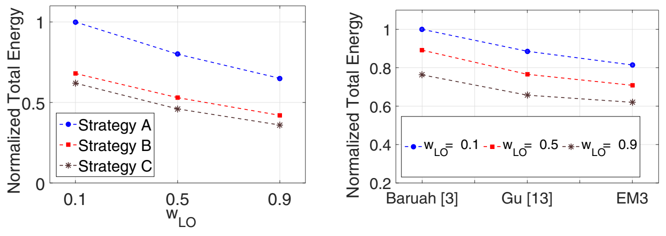
1) Unicore: We consider 3 strategies to show the impact of weight factors and static energy on energy minimization:

SA No DVFS, where tasks execute on base frequency f_b.

SB DVFS where only dynamic energy is optimized (P_s set to zero during optimization according to our algorithm in Sec. V). However, when calculating the total system energy, we include static energy consumption.

τ	χ_i	T_i	C_i(LO)	C_i(HI)
τ_1	HI	40	4	12
τ_2	HI	75	6	18
τ_3	HI	40	3	9
τ_4	LO	100	6	6
τ_5	LO	80	5	5

TABLE II: Example mixed-criticality taskset



(a) Weight & static energy (b) Task mapping

Fig. 1: Motivational example: Energy consumption under impacts of weight factors, static energy and task mapping

SC DVFS where both static and dynamic energy are optimized according to our algorithm in Sec. V.

We summarize our results in Figure 1a. Indeed, we can observe that energy can be greatly reduced by employing DVFS. If we compare Strategy A with C, 41% of total energy consumption is saved by employing DVFS when $w_{\text{LO}} = 0.1$. In addition, we observe that weight factors greatly affect the total energy consumption for all strategies. For example, in Strategy C, the minimal system energy consumption can vary by 38% when w_{LO} changes from 0.1 to 0.9. This further implies that the choice of weights is critical. That is, either neglecting HI mode energy ($w_{\text{LO}} \to 1$) or neglecting LO mode energy ($w_{\text{LO}} \to 0$) would lead to overly optimistic or pessimistic results. However, detailed domain knowledge of the applications may be required to select meaningful weight factors. By comparing strategies B and C, we also observe that optimizing static energy can further reduce the total energy consumption of the system. For all the weight factors considered, Strategy C achieves $\sim$10% more energy savings than Strategy B.

2) Multi-core: For mixed-criticality systems on multi-cores, we have to further consider the impact of task mapping on energy minimization. To this end, we compare three different approaches – two state of the art mapping techniques (Baruah's method [4] and Gu's method [13]) and a new method EM3 proposed in this paper (see Sec. VI).

We apply the aforementioned techniques to the task set of Table II on a dual-core platform, with the same power parameters on each core. We apply our unicore energy minimization technique (Sec. V) individually on all cores. The total energy for each mapping method is calculated by deriving the weighted energy consumption in different system modes on each core and then summing them up across all cores. Figure 1b shows the normalized total energy consumption obtained using different mapping techniques. As we can observe, indeed the task mapping can greatly affect the achievable energy saving. As an example, with $w_{\text{LO}} = 0.5$, EM3 achieves up to 25% energy reduction compared to the previous methods. The reason for this is that our method can balance the workload better across different cores in order to improve energy savings. We provide detailed explanations in Sec. VI.

Similar to unicore, we can also see that weight factors play a critical role in energy minimization for multi-core platforms.

In summary, through the above example, we have demonstrated that, weight factors, static energy and mapping techniques all play a critical role in the mixed-criticality energy minimization problem. Motivated by this, we proceed to formally define our problem.

B. Problem Formulation

As discussed earlier, due to intrinsic intractability of the mixed-criticality scheduling problem, we divide-and-conquer the energy minimization problem by solving two subproblems: (i) We first perform energy-aware task mapping; (ii) we then develop and apply unicore DVFS techniques to all cores. However, the first problem is dependent on the second, as we want to find mapping techniques that could explore best the energy saving potentials of unicore solutions. In the following, we formulate these two subproblems.

1) Unicore problem: For our problem here, we assume that τ is our considered task set on one core. To apply DVFS, the essential problem is to assign each task an operating frequency in each system mode, such that energy is minimized while mixed-criticality real-time guarantees are satisfied.

Let us denote the frequency of task τ_i in mode χ as f_i^{χ}, where $\chi \in \{\text{LO}\,,\, \text{HI}\}$ (uniformly represented by function $\mathbb{F} : \tau \times \{\text{LO}\,,\, \text{HI}\} \to \mathbb{R}^+$). To consider energy minimization for both system modes, we first need to define a proper energy objective. To this end, we express the importance of minimizing LO mode energy with a weight factor w_{LO} (similarly w_{HI} for HI mode), where, $w_{\text{LO}}, w_{\text{HI}} \in [0, 1]$, $w_{\text{LO}} = 1 - w_{\text{HI}}$. We do not pose any restriction on how to obtain w_{LO} and w_{HI}. Our formulation is rather general: if $w_{\text{LO}} = 1 \wedge w_{\text{HI}} = 0$, we minimize only LO mode energy; if $w_{\text{LO}} = 0.5 \wedge w_{\text{HI}} = 0.5$, we then minimize the average energy consumption in both modes.

First, for one hyper-period in LO mode ($\Pi_{\tau} T_i$), we calculate the normalized total energy consumption in this interval as the actual energy consumption divided by the interval length,

$$E_{\text{LO}} = w_{\text{LO}} \sum_{\tau_i \in \tau} \frac{C_i(\text{LO})}{T_i} \frac{f_b}{f_i^{\text{LO}}} (P_s + \beta (f_i^{\text{LO}})^{\alpha}). \tag{3}$$

Similarly, we normalize the system energy consumption in a hyper-period in HI mode ($\Pi_{\chi_i = \text{HI}} T_i$) as

$$E_{\text{HI}} = w_{\text{HI}} \sum_{\tau_i \in \tau_{\text{HI}}} \frac{C_i(\text{HI})}{T_i} \frac{f_b}{f_i^{\text{HI}}} (P_s + \beta (f_i^{\text{HI}})^{\alpha}). \tag{4}$$

Finally, our goal is to minimize the system energy across both operation modes, where,

$$E = E_{\text{LO}} + E_{\text{HI}}, \tag{5}$$

and the optimal task frequencies in different system modes need to be chosen (f_i^{χ}, $\forall \tau_i \forall \chi$). Furthermore, energy should be minimized while satisfying the mixed-criticality real-time requirements. Using Theorem 1, we can get a new schedulability condition when task utilizations are scaled under DVFS. We show the application of DVFS to EDF-VD, extending the result of Theorem 1 using the following corollary.

Corollary 1. Consider a task set τ scheduled by EDF-VD on a unicore. Assume that, with DVFS, task τ_i has frequency f_i^{χ} in system mode χ, then the feasible range of x is,

$$0 < \frac{\widetilde{U}_{\text{HI}}^{\text{LO}}}{1 - \widetilde{U}_{\text{LO}}^{\text{LO}}} \leq x \leq [\![\frac{1 - \widetilde{U}_{\text{HI}}^{\text{HI}}}{\widetilde{U}_{\text{LO}}^{\text{LO}}}]\!]^1 \tag{6}$$

where

$$\widetilde{U}_{\chi_1}^{\chi_2} = \sum_{\tau_i \in \tau_{\chi_1}} \frac{\widetilde{C}_i(\chi_2)}{T_i}\,; \quad \widetilde{C}_i(\text{LO}) = \frac{C_i(\text{LO}) f_b}{f_i^{\text{LO}}}\,, \forall \tau_i \in \tau;$$

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

Notice that, according to Corollary 1, with increasing f_i^{χ}, $\forall \tau_i$, the lower bound of x after DVFS does not increase, while the upper bound does not decrease as all utilization factors decrease. Thus, we know for any possible DVFS strategy, x must be within the absolute respective bounds when $f_i^{\chi} = f_{\max}$, $\forall \tau_i \forall \chi$. We denote the lower and upper bounds in this case as $\hat{x}_{\text{LB}}$, $\hat{x}_{\text{UB}} \in [0,\, 1]$, respectively.

To apply DVFS to save energy, we need to ensure that (5) is minimized while (6) is satisfied. With reformatting and constraint transformation, we can formulate our unicore energy minimization as a convex program. We capture this with the following theorem.

Theorem 2. The unicore energy minimization problem can be formulated as a convex program given by:

$$\begin{aligned} \textbf{minimize} \quad & E = E_{\text{LO}} + E_{\text{HI}} && (7) \\ \textbf{s.t.} \quad & \frac{\widetilde{U}_{\text{HI}}^{\text{LO}}}{x} + \widetilde{U}_{\text{LO}}^{\text{LO}} \leq 1 && (8) \\ & x \widetilde{U}_{\text{LO}}^{\text{LO}} + \widetilde{U}_{\text{HI}}^{\text{HI}} \leq 1 && (9) \\ & x \in [\hat{x}_{\text{LB}}\,,\, \hat{x}_{\text{UB}}] && (10) \\ & \forall \tau_i\,, \forall \chi\,, f_i^{\chi} \in [f_{\min}\,,\, f_{\max}] && (11) \end{aligned}$$

The convexity of the program can be easily verified [7]. Our energy objective is a convex function of task frequencies and the left-hand sides of our constraints are also convex functions of task frequencies and the deadline scaling factor. Although the convex formulation suggests practical algorithms [7] to solve our problem, we will theoretically investigate our problem and develop more insights in later sections.

2) Multi-core problem: Assuming partitioned scheduling in this paper, we still need to find an energy efficient task mapping onto a multi-core platform, such that by applying unicore DVFS locally on each core, the total energy consumed by all tasks on all cores is minimized. Let us denote task mapping with function $\mathbb{M} : \tau \to \pi$. In addition, since we apply EDF-VD on each core locally, we define the deadline scaling factors on all cores through function $\mathbb{X} : \pi \to \mathbb{R}^+$. Our multi-core problem can then be formalized as follows.

Definition 1 (Mixed-Criticality Energy Minimization on Multi-core)**.** Given a task set τ on a DVFS-capable platform π, find $\mathbb{M}$, $\mathbb{X}$ and $\mathbb{F}$ such that the total energy consumption on all cores for different modes is minimized.

IV. A Unicore Optimal Solution

We proceed to acquire a deeper insight into our unicore energy minimization problem. Since we have a convex formulation (Theorem 2), we first apply the Karush-Kuhn-Tucker (KKT) optimality conditions [20] to our problem. Due to the intrinsic computation complexity involved, we then show how to reduce the actual search space.

A. KKT conditions

Let us first introduce the KKT conditions for solving general optimization problems. Consider a problem of the form

$$\begin{array}{rll} \textbf{minimize} & f(z) & \\ \textbf{s.t.} & h_i(z) = 0 & \forall i = 1, ..., n \\ & g_j(z) \le 0 & \forall j = 1, ..., m \end{array}$$

where $h_i(z)$ and $g_j(z)$ are the equality and inequality constraints for a continuous function $f(z)$, respectively.

Theorem 3 (KKT conditions)**.** According to [20], if the objective and constraint functions are continuously differentiable, then a global minimal solution z^* exists, when there exist Lagrange multipliers λ_i $(1 \le i \le n)$ and KKT multipliers μ_j $(1 \le j \le m)$, such that

$$\nabla_x f(z^*) + \sum_{i=1}^{n} \lambda_i \nabla_x h_i(z^*) + \sum_{j=1}^{m} \mu_j \nabla_x g_j(z^*) = 0$$
$$\textbf{s.t.}\ \ h_i(z^*) = 0 \quad \forall i = 1, ...n; \quad g_j(z^*) \le 0 \quad \forall j = 1, ...m;$$
$$\mu_j g_j(z^*) = 0 \quad \forall j = 1, ...m; \quad \mu_j \ge 0 \quad \forall j = 1, ...m.$$

B. Complexity involved with KKT for our problem

As we have a convex program with continuous decision variables for our unicore problem (i.e. task frequencies are continuously available and the deadline scaling factor is also a continuous variable, see Theorem 2), we can directly apply Theorem 3 to find the optimal solution. However, this could be impractical as it leads to an exponential computational complexity.

Theorem 4. Solving our unicore mixed-criticality energy minimization problem directly by KKT conditions requires solving $2^{2|\tau_{\text{LO}}|+4|\tau_{\text{HI}}|+4}$ systems of non-linear equations.

Proof: Notice that for our unicore problem as formulated in Theorem 2, only inequality constraints exist. Thus, we need to introduce only the slack variables μ_j in each of the complementary slackness equations, $\mu_i g_i(z) = 0$, where at least one of μ_i and $g_i(z)$ must be 0. With m such conditions, there would be 2^m possible cases ($\mu_i = 0$ or $g_i(z) = 0$). For constraints (8) - (10), we need to introduce four slack variables (in (10) we have to consider both the upper and lower bounds). According to (11), for each HI criticality task, we have to introduce two slack variables for its frequency in each system mode, leading to a total of $4|\tau_{\text{HI}}|$ slack variables. Similarly, we have $2|\tau_{\text{LO}}|$ slack variables for LO criticality tasks (they are not executed in HI mode). This would lead to $2^{2|\tau_{\text{LO}}|+4|\tau_{\text{HI}}|+4}$ binary cases to check for the optimal solution, where in each case, we have to solve a system of non-linear equations. ■

To reduce the intrinsic high complexity encountered, we proceed towards an in-depth analysis of the energy minimization problem, aiming to reduce the actual search space.

C. Reduction of search space

1) Optimality condition in f_i^{χ}: Considering only dynamic energy consumption in LO mode, [16] proved that the system energy consumption is minimum when all tasks of the same criticality level share the same frequency in each mode. We now generalize this result to consider additional static energy consumption and system energy in both LO and HI modes. Let us introduce 3 frequency variables $f_{\text{LO}}^{\text{LO}}$, $f_{\text{HI}}^{\text{LO}}$ and $f_{\text{HI}}^{\text{HI}}$, where $f_{\chi_1}^{\chi_2}$ represents the frequency of all χ_1 criticality tasks in χ_2 system mode. Formally, we have the following result.

Theorem 5. For the unicore mixed-criticality energy minimization problem as specified in Theorem 2, in an optimal solution all tasks of the same criticality level share the same frequency in each mode, i.e.,

$$\forall \tau_i \in \tau_{\text{LO}}, f_i^{\text{LO}} = f_{\text{LO}}^{\text{LO}}; \quad \forall \tau_i \in \tau_{\text{HI}}, f_i^{\text{LO}} = f_{\text{HI}}^{\text{LO}} \wedge f_i^{\text{HI}} = f_{\text{HI}}^{\text{HI}}.$$

Proof: Detailed proof is provided in Appendix A. ■

Theorem 5 can be derived by applying KKT conditions separately to several subproblems of our unicore problem. We refer the interested readers to Appendix A for detailed explanations. Using Theorem 5, the search space in task frequencies can be greatly reduced and the energy objective (3) and (4) can be simplified as follows,

$$\begin{aligned} E_{\text{LO}} =& w_{\text{LO}} f_b U_{\text{LO}}^{\text{LO}} (P_s / f_{\text{LO}}^{\text{LO}} + \beta (f_{\text{LO}}^{\text{LO}})^{\alpha-1}) \\ &+ w_{\text{LO}} f_b U_{\text{HI}}^{\text{LO}} (P_s / f_{\text{HI}}^{\text{LO}} + \beta (f_{\text{HI}}^{\text{LO}})^{\alpha-1}), \\ E_{\text{HI}} =& w_{\text{HI}} f_b U_{\text{HI}}^{\text{HI}} (P_s / f_{\text{HI}}^{\text{HI}} + \beta (f_{\text{HI}}^{\text{HI}})^{\alpha-1}). \end{aligned} \tag{12}$$

2) Optimality condition in x: We continue to derive the necessary conditions in x (the deadline shortening factor) for any unicore solution to be optimal. We find this by relating the choice of x with energy consumption in both system modes.

According to Theorem 5, let us denote the frequencies in an optimal solution as $f_{\text{LO}\,\text{opt}}^{\text{LO}}$, $f_{\text{HI}\,\text{opt}}^{\text{LO}}$ and $f_{\text{HI}\,\text{opt}}^{\text{HI}}$. Let us further define K, L and M as follows,

$$\begin{aligned} K = \sum_{\chi_i=\text{HI}} \frac{C_i(\text{LO}) f_b}{T_i}, \qquad L = \sum_{\chi_i=\text{LO}} \frac{C_i(\text{LO}) f_b}{T_i}, \\ M = 1 - \sum_{\chi_i=\text{HI}} \frac{(C_i(\text{HI}) - C_i(\text{LO})) f_b}{T_i f_{\text{HI}\,\text{opt}}^{\text{HI}}}. \end{aligned} \tag{13}$$

Using Corollary 1, we can derive the lower and upper bounds for x in an optimal solution as,

$$x_{\text{LB}\,\text{opt}} = \frac{\frac{U_{\text{HI}}^{\text{LO}}}{f_{\text{HI}\,\text{opt}}^{\text{LO}}} f_b}{1 - \frac{U_{\text{LO}}^{\text{LO}}}{f_{\text{LO}\,\text{opt}}^{\text{LO}}} f_b} = \frac{K / f_{\text{HI}\,\text{opt}}^{\text{LO}}}{1 - L / f_{\text{LO}\,\text{opt}}^{\text{LO}}}, \tag{14}$$

$$\begin{aligned} x_{\text{UB}\,\text{opt}} &= \left[\!\!\left[\frac{1 - \frac{U_{\text{HI}}^{\text{LO}}}{f_{\text{HI}\,\text{opt}}^{\text{LO}}} f_b - \frac{U_{\text{HI}}^{\text{HI}} - U_{\text{HI}}^{\text{LO}}}{f_{\text{HI}\,\text{opt}}^{\text{HI}}} f_b}{\frac{U_{\text{LO}}^{\text{LO}}}{f_{\text{LO}\,\text{opt}}^{\text{LO}}} f_b} \right]\!\!\right]^1 \\ &= \left[\!\!\left[\frac{M - K / f_{\text{HI}\,\text{opt}}^{\text{LO}}}{L / f_{\text{LO}\,\text{opt}}^{\text{LO}}} \right]\!\!\right]^1. \end{aligned} \tag{15}$$

where $[\![a]\!]^c = \min(a, c)$. Now, we consider choosing the optimal deadline scaling factor x_{opt}. To ensure schedulability, it must follow that, $x_{\text{LB}\,\text{opt}} \le x_{\text{opt}} \le x_{\text{UB}\,\text{opt}}$ (see Corollary 1).

From (14) and (15), we observe that $f^{\mathrm{LO}}_{\mathrm{LO\,opt}}$ and/or $f^{\mathrm{LO}}_{\mathrm{HI\,opt}}$ can be decreased by increasing $x_{\mathrm{LB\,opt}}$ or decreasing $x_{\mathrm{UB\,opt}}$, thus minimizing LO mode energy[1]. Similarly, $f^{\mathrm{HI}}_{\mathrm{HI\,opt}}$ can be decreased by decreasing $x_{\mathrm{UB\,opt}}$ to save energy in HI mode. As a result, as long as task frequencies are not minimal ($f_{\min}$) and $x_{\mathrm{LB\,opt}} \neq x_{\mathrm{UB\,opt}}$, we can reduce either LO mode frequencies or HI mode frequencies to bring the lower and upper bounds on x closer. This process can only stop if, (i) all task frequencies are already lowered to $f_{\min}$, or (ii) $x_{\mathrm{LB\,opt}} = x_{\mathrm{UB\,opt}}$. We summarize our observations on this necessary optimality condition using the following theorem.

Theorem 6. An optimal solution to our unicore problem as formulated in Theorem 2 exists in one of the two following cases:

1 Extreme case, i.e. $f^{\mathrm{LO}}_{\mathrm{LO}} = f^{\mathrm{LO}}_{\mathrm{HI}} = f^{\mathrm{HI}}_{\mathrm{HI}} = f_{\min}$.

2 Equilibrium case, i.e. $x_{\mathrm{LB\,opt}} = x_{\mathrm{opt}} = x_{\mathrm{UB\,opt}}$.

Proof: This directly follows from discussions above. ■

Notice that Theorem 6 generalizes a similar condition in [16], where only dynamic energy in LO mode is considered.

D. Optimal solution with KKT

Using Theorem 5 and Theorem 6, we have a much reduced search space. We could apply KKT conditions again to our unicore problem. Notice that, according to Theorem 6, the optimal solution could only exist in the extreme case or the equilibrium case. For the extreme case, we only need to test whether the system is feasible while running at $f_{\min}$. Therefore, we only need to apply KKT to the equilibrium case. We can add the equilibrium constraints to our problem formulation as shown in Theorem 2, while minimizing a much simplified energy objective (12). Our simplified problem can be formulated as follows:

$$\begin{aligned} \textbf{minimize} \quad & (12) \\ \textbf{s.t.} \quad & (14); (15); x_{\mathrm{LB\,opt}} = x_{\mathrm{opt}} = x_{\mathrm{UB\,opt}}; \\ & (8); (9); f^{\mathrm{LO}}_{\mathrm{LO}}, f^{\mathrm{LO}}_{\mathrm{HI}}, f^{\mathrm{HI}}_{\mathrm{HI}} \in [f_{\min}, f_{\max}]. \end{aligned} \tag{16}$$

Notice that, at equilibrium, we can remove the $[\![\cdot]\!]^1$ operator to make $x_{\mathrm{UB\,opt}}$ differentiable. For a feasible system, when $x_{\mathrm{LB\,opt}} = x_{\mathrm{UB\,opt}} < 1$, this is evident; when $x_{\mathrm{LB\,opt}} = x_{\mathrm{UB\,opt}} = 1$, using (14) and (15) we have,

$$\frac{1-\left(\frac{U^{\mathrm{LO}}_{\mathrm{HI}}}{f^{\mathrm{LO}}_{\mathrm{HI\ opt}}}\right)f_b-\left(\frac{U^{\mathrm{HI}}_{\mathrm{HI}}-U^{\mathrm{LO}}_{\mathrm{HI}}}{f^{\mathrm{HI}}_{\mathrm{HI\ opt}}}\right)f_b}{\left(\frac{U^{\mathrm{LO}}_{\mathrm{LO}}}{f^{\mathrm{LO}}_{\mathrm{LO\ opt}}}\right)f_b} \le \frac{1-\left(\frac{U^{\mathrm{LO}}_{\mathrm{HI}}}{f^{\mathrm{LO}}_{\mathrm{HI\ opt}}}\right)f_b}{\left(\frac{U^{\mathrm{LO}}_{\mathrm{LO}}}{f^{\mathrm{LO}}_{\mathrm{LO\ opt}}}f_b\right)} = 1. \tag{17}$$

As a result we can apply KKT to our simplified problem (16). We need to introduce two Lagrange multipliers λ_i for the equilibrium constraint, two slack variables (μ_j) for each of the frequency variables $f^{\mathrm{LO}}_{\mathrm{LO}}$, $f^{\mathrm{LO}}_{\mathrm{HI}}$ and $f^{\mathrm{HI}}_{\mathrm{HI}}$. Then, Theorem 3 can be directly applied. A critical observation is that, with reduced search space, we only need to solve 2^6 systems of non-linear equations as compared to $2^{2|\tau_{\mathrm{LO}}|+4|\tau_{\mathrm{HI}}|+4}$ for our original problem (Theorem 4).

[1]Recall that $f_{\min} \geq f_{\mathrm{crit}}$ and reducing frequency is always beneficial in saving both static and dynamic energy.

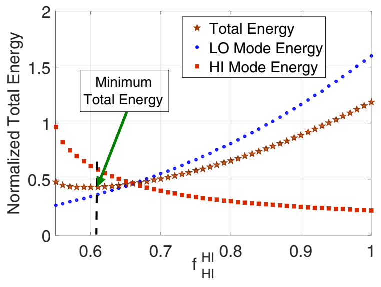


Fig. 2: LO mode and HI mode energy as a function of $f^{\mathrm{HI}}_{\mathrm{HI}}$

V. A Simple & Effective Unicore Heuristic

As shown in Sec. IV, even by limiting the problem search space (without loss of optimality), applying the well-known KKT approach to our problem still incurs high complexity (2^6 systems of non-linear equations). We proceed now to develop a computationally simple yet effective heuristic solution to our unicore problem.

A. Intuition

As already discussed, there exists a trade-off between energy consumption in different system modes. With increasing $f^{\mathrm{HI}}_{\mathrm{HI}}$, system energy consumption in HI mode will be increased (see (12)). However, this could generate more system slack in LO mode, allowing reduction of LO mode task frequencies to reduce LO mode energy. In other words, increasing $f^{\mathrm{HI}}_{\mathrm{HI}}$ could potentially increase the optimal upper bound of the scaling factor $x_{\mathrm{UB\,opt}}$ as presented in (15). Due to the equilibrium optimality condition (Theorem 6), this could enable us to increase the optimal lower bound of x as well by decreasing task frequencies in LO mode, see (14). Similarly, increasing LO mode task frequencies could potentially decrease HI mode system energy.

Moreover, the main intuition is that the second order differentials of both LO and HI mode minimal energy consumption (12) are non-negative with respect to $f^{\mathrm{HI}}_{\mathrm{HI}}$. According to (1) and (12), with increasing $f^{\mathrm{HI}}_{\mathrm{HI}}$, dynamic energy consumption in HI mode becomes dominant compared to static energy consumption, leading to a higher rate in energy increase. Furthermore, as $f^{\mathrm{HI}}_{\mathrm{HI}}$ increases, LO mode energy reduces asymptotically to an absolute lower bound when all task frequencies in LO mode are $f_{\min}$. The above observations are clear if we consider the same example from Table II and plot $E_{\mathrm{LO\,opt}}$ and $E_{\mathrm{HI\,opt}}$ as a function of $f^{\mathrm{HI}}_{\mathrm{HI}}$, as shown in Figure 2. We will show later in this section how to derive $E_{\mathrm{LO\,opt}}$ and $E_{\mathrm{HI\,opt}}$. Due to the trade-off between LO and HI mode energy consumption, there exists a $f^{\mathrm{HI}}_{\mathrm{HI}}$ in the feasible range leading to the minimum overall energy. We aim to find such a $f^{\mathrm{HI}}_{\mathrm{HI}}$ using a simple heuristic.

B. Heuristic algorithm

In the extreme case (i.e., all task frequencies are $f_{\min}$), we need to confirm the system schedulability using Corollary 1.

In the equilibrium case, we propose a heuristic approach to find the optimal solution based on the intuitions discussed in this section. Let us denote with $E'_{\mathrm{LO\,opt}}$ ($E'_{\mathrm{HI\,opt}}$) the first order differential of the minimal LO (HI) mode energy consumption with respect to $f^{\mathrm{HI}}_{\mathrm{HI}}$. E'_{opt} would be non-decreasing since the second order differentials of $E_{\mathrm{LO\,opt}}$ and $E_{\mathrm{HI\,opt}}$ are both non-negative. Let the feasible range of $f^{\mathrm{HI}}_{\mathrm{HI}}$ be $[f^{\mathrm{HI}}_{\mathrm{HI\,min}}, f^{\mathrm{HI}}_{\mathrm{HI\,max}}]$, where $f^{\mathrm{HI}}_{\mathrm{HI\,min}}$ is the minimum HI mode frequency required to

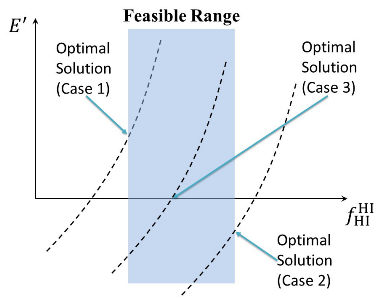

Feasible Range
E'
Optimal Solution (Case 1)
Optimal Solution (Case 3)
$f_{\mathrm{HI}}^{\mathrm{HI}}$
Optimal Solution (Case 2)

Fig. 3: Different cases to check in Algorithm 1

Algorithm 1: A simple and effective unicore heuristic

```
input : $\tau$, $f_b$, $f_{\min}$, $f_{\max}$, $w_{\mathrm{LO}}$ and $w_{\mathrm{HI}}$
output: $f_{\mathrm{LO\,opt}}^{\mathrm{LO}}$, $f_{\mathrm{HI\ opt}}^{\mathrm{LO}}$, $f_{\mathrm{HI\,opt}}^{\mathrm{HI}}$, $x_{\mathrm{opt}}$
1  if System feasible when $f_{\mathrm{LO}}^{\mathrm{LO}} = f_{\mathrm{HI}}^{\mathrm{LO}} = f_{\mathrm{HI}}^{\mathrm{HI}} = f_{\max}$ according to Corollary 1 then
2      if System feasible when $f_{\mathrm{LO}}^{\mathrm{LO}} = f_{\mathrm{HI}}^{\mathrm{LO}} = f_{\mathrm{HI}}^{\mathrm{HI}} = f_{\min}$ according to Corollary 1 then
3          $f_{\chi_1}^{\chi_2} \leftarrow f_{\min}\ \forall\chi_1\forall\chi_2$;
4      else
5          Determine the feasible range of $f_{\mathrm{HI}}^{\mathrm{HI}}$ according to the Corollary 1 ($f_{\mathrm{HI}}^{\mathrm{LO}}$ and $f_{\mathrm{LO}}^{\mathrm{LO}}$ set to $f_{\max}$);
6          If $E'$ is non-negative at the smallest feasible $f_{\mathrm{HI}}^{\mathrm{HI}}$, set this as $f_{\mathrm{HI\,opt}}^{\mathrm{HI}}$ (case 1);
7          If $E'$ is non-positive at the biggest feasible $f_{\mathrm{HI}}^{\mathrm{HI}}$, set this as $f_{\mathrm{HI\,opt}}^{\mathrm{HI}}$ (case 2);
8          If the above does not hold, we do a binary search to find the $f_{\mathrm{HI}}^{\mathrm{HI}}$ such that $E' = 0$; set this as $f_{\mathrm{HI\,opt}}^{\mathrm{HI}}$ (case 3);
9      return Success;
10 else
11     return Failure;
```

ensure HI mode schedulability (when LO mode frequencies of tasks are fixed at $f_{\max}$) and $f_{\mathrm{HI\,max}}^{\mathrm{HI}}$ is simply $f_{\max}$. If there exists an optimal $f_{\mathrm{HI}}^{\mathrm{HI}}$ within this range, then it must be in one of these three cases as shown in Figure 3.

C1 $E'_{\mathrm{opt}} \geq 0$ in the feasible range of $f_{\mathrm{HI}}^{\mathrm{HI}}$: E_{opt} is always increasing, and the optimal solution exists at $f_{\mathrm{HI\,min}}^{\mathrm{HI}}$.

C2 $E' \leq 0$ in the feasible range: E_{opt} is always decreasing and the optimal solution exists when $f_{\mathrm{HI\,opt}}^{\mathrm{HI}} = f_{\max}$.

C3 The above do not hold: There exists a stationary point $E' = 0$ such that the minimal energy can be achieved. We perform binary search to locate this stationary point.

The above heuristic solution is shown in Steps 7-9 in Algorithm 1, which only involves logarithmic computation complexity due to binary search. As for the calculation of the 1st order differentials, we refer the interested readers to Appendix B. Further, $f_{\mathrm{LO\,opt}}^{\mathrm{LO}}$, $f_{\mathrm{HI\ opt}}^{\mathrm{LO}}$ and x_{opt} are also calculated there based on $f_{\mathrm{HI\,opt}}^{\mathrm{HI}}$.

VI. Energy Minimization on Multi-cores

In Sec. IV and Sec. V, we presented our unicore solutions to the mixed-criticality energy minimization problem. Focusing on partitioned scheduling, we proceed now to extend our unicore techniques to multi-cores by further proposing energy-aware task mapping techniques. For all the mapping techniques, we assume that unused cores are turned off in order to conserve energy. In the following, we first briefly discuss two existing mixed-criticality task mapping techniques. We then propose new energy-aware mapping techniques and platform allocation strategies.

A. Overview of Baruah's and Gu's methods

Let us first discuss two state of the art mixed-criticality mapping techniques, which are designed to enhance system schedulability rather than energy efficiency.

Baruah's method [4]: In this method, tasks are mapped onto multiple cores using First-Fit (FF) bin-packing, i.e., any task is assigned to an immediate core where it fits first. This is done first for HI criticality tasks and then for LO criticality tasks, while system utilizations are upper bounded in each phase to admit a feasible schedule ($U_{\mathrm{HI}}^{\mathrm{LO}} + U_{\mathrm{LO}}^{\mathrm{LO}} \leq 3/4 \wedge U_{\mathrm{HI}}^{\mathrm{HI}} \leq 3/4$). When mapping is successful, local EDF-VD scheduling can be applied independently on each core [3]. However, due to the fact that Baruah's method only aims at enhancing system schedulability (i.e., using FF to maximize the system utilization on each core), it leaves little room or slack to apply DVFS to save energy.

Gu's method [13]: To further enhance mixed-criticality schedulability on multi-cores, this method first assigns all HI criticality tasks onto multi-cores by Worst-Fit (WF) bin-packing, with the intuition that HI criticality workload should be balanced across all cores to further admit a fair mix of HI and LO criticality workloads on each core. Mapping of LO criticality tasks is done in the same way as that for Baruah's method. However, Gu's method still allows high utilization of cores (due to greedy mapping of LO criticality tasks), where little room/slack can be explored to save energy by DVFS.

B. Energy Minimized Mixed-criticality Mapping (EM3)

Both Baruah's and Gu's methods can incur heavily loaded cores, where little can be done by DVFS to save energy. Thus, we first develop a technique aiming at balancing the mixed-criticality workloads on all cores. Achieving this is rather straightforward, we apply WF to map both HI and LO criticality tasks. Supposing we have allocated k cores to be used:

1 HI criticality tasks are mapped onto k cores using WF in the order of decreasing utilization, with cumulative HI mode utilization on each core upper bounded by $\frac{3}{4}$.

2 LO criticality tasks are mapped onto k cores using WF in order of decreasing utilization, with cumulative LO mode utilization on each core upper bounded by $\frac{3}{4}$.

3 DVFS strategy is applied to the task set on each core using the heuristic solution presented in Sec. V, and all tasks are scheduled using EDF-VD.

4 To find the optimal number of processors being used, k_{opt}, we repeat the above steps and perform a linear search across all feasible allocations to find k_{opt} with the minimum total energy consumption.

Note: (i) If a core contains only LO criticality tasks after the task mapping, we ensure that the system utilization does not exceed 1 rather than applying the sophisticated mixed-criticality schedulability test [4], since we have a single criticality scheduling problem. (ii) If possible, we select the minimal set of cores among all allocations leading to

Algorithm 2: Isolated Mixed-criticality Mapping Method

```
input : $\tau$, $\pi$, $f_b$, $f_{\min}$, $f_{\max}$, $w_{\text{LO}}$ and $w_{\text{HI}}$
output: $\mathbb{M}$, $\mathbb{F}$, $\mathbb{X}$
1 Set $\bar{l}_0 = \lceil U_{\text{LO}}^{\text{LO}} \frac{f_b}{f_{\max}} \rceil$ and $\bar{h}_0 = \lceil U_{\text{HI}}^{\text{HI}} \frac{f_b}{f_{\max}} \rceil$;
2 if $\bar{l}_0 + \bar{h}_0 \leq |\pi|$ then
3     for $\bar{l}_0 + \bar{h}_0 \leq i \leq |\pi|$ do
4         linear search for $\bar{l}$ and $\bar{h}$, such that $\bar{l}_i \geq \bar{l}_0 \wedge$
          $\bar{h}_i \geq \bar{h}_0 \wedge \bar{l}_i + \bar{h}_i = i \wedge$ energy is minimized;
5     Among all $i$, select $\bar{l}_i$ and $\bar{h}_i$ such that total energy
      is minimized and output $\mathbb{M}$, $\mathbb{F}$, $\mathbb{X}$ in this case;
6     return Success;
7 else
8     return Failure;
```

the minimal energy consumption. This could minimize the overhead incurred by utilizing additional cores (e.g., non-zero inactive/sleep state energy and power state transition overheads), which we do not explicitly address in this paper.

C. Isolated Mixed criticality Mapping (IM3 Method)

The above described EM3 method aims to balance mixed-criticality workloads on multi-cores to save energy. We will now present a drastically different approach to solve the multi-core energy minimization problem by isolating tasks of different criticality levels on different cores. Although this is a common industrial practice to provide independent guarantees to different criticality levels, the hope is that criticality isolation could also enable energy saving.

Conventionally, when only schedulability is considered, the mainstream research advocates integrating workloads of different criticality levels on each processing core, such that smart resource management can be deployed to reconfigure the system under runtime threats by exploring the asymmetric guarantees on different criticality levels. As a result, resource efficiency can be enhanced compared to isolation methods [2]. However, when energy minimization is the primary goal, mixing workloads from different criticality levels might not be advantageous. On one hand, mixing workloads on all cores can help to achieve global workload balancing, leading to better energy savings. On the other hand, applying mixed-criticality scheduling reduces the maximum attainable system utilization on each core (e.g. $\frac{3}{4}$ for partitioned EDF-VD [4]), thus hampering energy savings.

In the new IM3 method, all LO criticality tasks are mapped onto $\bar{l}$ cores and all HI criticality tasks are mapped onto another $\bar{h}$ cores, such that $2 \leq (\bar{l} + \bar{h}) \leq |\pi|$, where $|\pi|$ is the total number of cores available. Both LO and HI criticality tasks are mapped onto their dedicated cores, using WF in the order of decreasing utilization. The choice of $\bar{l}$ and $\bar{h}$ depends on the total utilization of LO and HI criticality tasks, respectively. We apply a simple heuristic to find the best $\bar{l}$ and $\bar{h}$ such that the total normalized energy consumption is minimum, as presented in Algorithm 2.

To explain Algorithm 2 briefly, we first set the minimum number of required cores for LO criticality tasks to be schedulable as $\lceil U_{\text{LO}}^{\text{LO}} \frac{f_b}{f_{\max}} \rceil$ (for HI criticality tasks as $\lceil U_{\text{HI}}^{\text{HI}} \frac{f_b}{f_{\max}} \rceil$). Next, we do linear search through the feasible space of total allocated cores, the allocations of LO criticality cores and HI criticality cores, such that the system energy on all allocated cores is a minimum. We select in the end the total core allocation and corresponding $\bar{l}$ and $\bar{h}$ such that energy is a minimum across all possible allocations. Note that, the total energy of the system is computed using (12) for all core allocations and task mappings. Since LO and HI criticality tasks are isolated, the utilization of LO criticality tasks can be optimally up-scaled to 1 on $\bar{l}$ cores under DVFS and EDF scheduling [16] (when isolated, LO criticality tasks only have one system mode as they only have one level of WCETs). For HI criticality tasks, since two levels of WCETs exist, we apply EDF-VD (by excluding LO criticality tasks due to isolation) and our unicore solution (Sec. V) on each of the $\bar{h}$ cores. Due to isolation, it is not needed to consider weight factors for LO criticality tasks as they are not abandoned. However, it is required for HI criticality tasks as two operation modes exist, and those tasks can have different frequencies in different modes to minimize the overall energy consumption.

VII. Evaluation

In this section, we evaluate our proposed energy minimization techniques for mixed-criticality systems with extensive simulations on both randomly generated synthetic task sets and a real-life Flight Management System (FMS). Assuming synthetic task sets, we first present the results with unicore and then with multi-cores. We then perform energy evaluations on a realistic FMS. Last, we evaluate the proposed task mapping techniques in terms of multi-core schedulability for synthetic task sets.

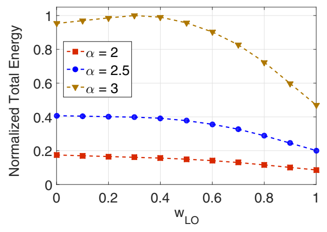
A. Experimental setup for extensive simulations

To validate our proposed techniques in a general setting, we implemented a well-known mixed-criticality task generator [4, 14, 16] to generate 1000 random feasible task sets at each system utilization point. The task generator was controlled by the following parameters: (i) U_t, the maximum of task utilizations in LO and HI system modes; (ii) $[U_{ll},\, U_{lu}]$, utilization of any LO criticality task is uniformly drawn from this range; (iii) $[U_{hl}, U_{hu}]$, LO mode utilization of any HI criticality task is uniformly drawn from this range; (iv) λ, the ratio of $C_i(\text{HI})$ to $C_i(\text{LO})$ for any HI criticality task; and (v) P_{HI}, the probability that a task is of HI criticality level. In all experiments, we assume that each core on the platform is DVFS-capable with $\{f_{\min},\, f_b,\, f_{\max}\} = \{0.55,\, 0.85,\, 1\}$GHz. We fix $P_s = 0.5$W, $\alpha = 2$ and $\beta = 1.76\text{W/GHz}^{\alpha}$ [23, 25], unless stated otherwise. With this setup, we apply our proposed techniques to the synthetic task sets to minimize energy; we calculate normalized energy consumption averaged over 1000 task sets at each system utilization point. Note that for our experiments on multi-cores, we assume that unused cores are turned off to further save energy.

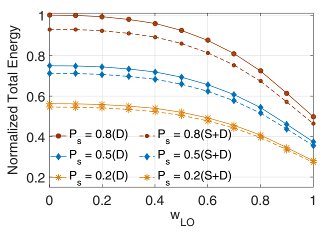
B. Evaluation on unicore

We consider $U_t = 0.75$, $[U_{ll}, U_{lu}] = [0.001, 0.01]$, $[U_{hl}, U_{hu}] = [0.05, 0.1]$, and we fix $w_{\text{LO}} = w_{\text{HI}} = 0.5$, $P_{\text{HI}} = 0.5$ and $\lambda = 1.4$, unless stated otherwise. With our experiments, we study the impact of various factors on unicore energy minimization.

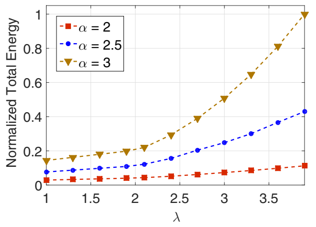
1) Impact of Weights: First, we show the impact of weight factors on energy minimization for different system modes. We calculate the normalized total energy with different weight factors w_{LO} (and $w_{\text{HI}} = 1 - w_{\text{LO}}$) in the range $[0, 1]$ in steps of 0.1, for $\alpha = 2, 2.5, 3$. Our results are shown in Figure 4a. As



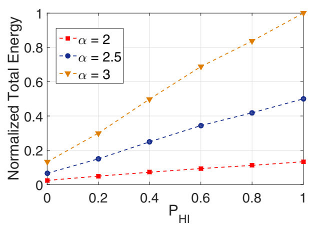
(a) Impact of weight factors



(b) Impact of static energy

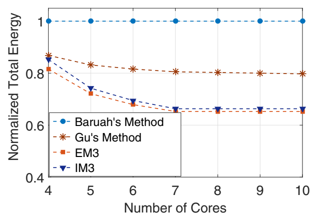


(c) Impact of λ

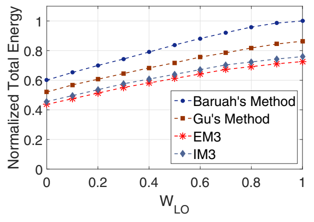


(d) Impact of P_{HI}

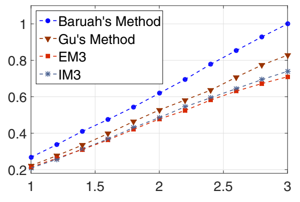
Fig. 4: Comparison of normalized total energy on unicore under impacts of α, weight factors, static energy, λ and P_{HI}



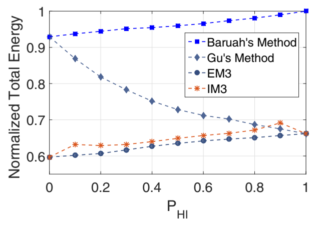
(a) Impact of number of cores



(b) Impact of weight factors



(c) Impact of total utilization



(d) Impact of P_{HI}

Fig. 5: Comparison of normalized total energy on multi-core under impacts of #core, weight factors, system utilization and P_{HI}

we can observe, the choice of weight factors can greatly affect potential energy savings. In particular, the minimal expected energy decreases by $\sim$50% with w_{LO} increasing from 0 to 1. The reason is that the system utilization in HI mode is significantly higher than that in LO mode ($\sim$40% more due to task generation parameters). This further implies that the choice of the right weight factors is crucial to the mixed-criticality energy minimization problems – considering either only HI mode ($w_{\text{LO}} = 0$) or LO mode ($w_{\text{LO}} = 1$) would lead to overly optimistic or pessimistic results. In practice, knowledge of the particular system considered would help choosing such weight factors. Furthermore, as expected, with higher dynamic power (α), the minimal system energy consumption is increased.

2) Impact of P_s: We continue to show the impact of static energy on energy minimization. We calculate the total energy with (S+D) or without (D) considering P_s during optimization. However, in calculating total energy both static and dynamic energy are included in both cases. Our results are presented in Figure 4b for w_{LO} in the range $[0, 1]$, and $P_s = 0.2, 0.5, 0.8$W. It is evident from the figure that static energy plays an important role in our energy minimization problem; with increasing P_s, the achievable minimal energy consumption is increased significantly. For example, when $w_{\text{LO}} = 0.5$, the minimal system energy consumption is increased by 31% (D) and 28% (S+D) when P_s increases from 0.5W to 0.8W. Furthermore, in all cases, including static energy during optimization saves more energy. In particular, with increasing P_s, it is more important to consider minimizing static energy. When $P_s = 0.2$W, minimizing in addition static energy leads to 4% more reduction of total energy. This is further increased to 8% when $P_s = 0.8$W.

3) Impact of λ: To show the impact of extra workload, we consider $\lambda = \frac{C(\text{HI})}{C(\text{LO})} \geq 1$ and 1000 different task sets for each value of λ. When $\lambda = 1$, there is no extra workload as HI criticality tasks would never exceed their LO criticality WCETs. With $\lambda > 1$, there is always a timing safety concern for HI criticality tasks as HI mode WCET increases. Figure 4c shows the normalized minimal energy with λ in the range $[1, 3.9]$. As we see, with increasing λ, the minimal expected energy consumption increases, simply because of increase in HI mode system load. This trend with increasing energy stops at $\lambda = 3.9$, after which the system becomes infeasible even when we set all task frequencies to $f_{\max}$.

4) Impact of P_{HI}: Finally, we demonstrate the impact of P_{HI} on energy minimization. We vary P_{HI} in the range $[0, 1]$ in steps of 0.1 and the results are shown in in Figure 4d for $\alpha = 2, 2.5, 3$. As we can observe, the expected minimal energy increases with increasing P_{HI}. This is because, if P_{HI} increases, more HI criticality tasks are generated, leading to higher extra workload in HI criticality system mode. However, the LO mode system load is not affected since this is fixed by U_t in our task generator [2, 4, 16]. As a result, the total system workload increases across LO and HI modes, leading to increases in system energy consumption.

C. Evaluation on Multi-cores

Here, we consider 4 cores, $U_t = 3$, $[U_{ll}, U_{lu}] = [U_{hl}, U_{hu}] = [0.005, 0.01]$, $P_s = 0.5$, $P_{\text{HI}} = 0.5$, $\lambda = 1.4$ and $w_{\text{LO}} = w_{\text{HI}} = 0.5$ in all cases, unless stated otherwise. It should be noted that weight factors are not necessary for LO criticality tasks in IM3. However, for a fair comparison with other methods, we include them and use the same equation (12) to calculate the overall energy on each core. We obtain our results by simulating different task mapping techniques on the random task sets. Later we apply our unicore method (see Sec. V) to minimize energy on all cores and to calculate the total normalized energy.

1) Impact of number of cores: We obtain our results by performing experiments on 4 to 10 cores and show our results in Figure 5a. It is evident from the figure that, for all varying number of cores, EM3/IM3 saves energy considerably compared to other mapping techniques, e.g., 34%/31% more saving than Baruah's and 22%/20% more than Gu's method for 7 cores. The energy consumption for Baruah's method is constant – in all the cases, it employs FF method to map tasks, leading to 4 utilized cores and high utilizations on those cores. This decreases the chance to down-scale frequency on active cores to save energy. In Gu's method, only LO criticality tasks are mapped using FF, whereas the HI criticality tasks are mapped using WF to balance their workload on all cores. This leads to better balanced workload across the cores, as well as better energy savings than Baruah's method. Our proposed EM3 method performs load balancing for tasks on all criticality levels, leading to highest energy savings in those three cases.

Furthermore, we observe that IM3 method achieves almost comparable energy savings compared to EM3 method, which matches our intuition in designing IM3 (see Sec. VI). This leads us to the conclusion that criticality isolation is a good option when minimizing system energy on multicores, with the additional benefit of providing independences between different criticality levels. Lastly, with increasing number of cores, minimal system energy for all methods tends to decrease as more cores can be used to balance workload.

2) Impact of weight factors: We evaluate our proposed methods for different weight factors and show our results in Figure 5b. As we can observe, for $w_{\mathrm{LO}} \in [0\,,\,1]$ in steps of 0.1, EM3 always shows the best performance in energy saving and IM3 follows closely. We also notice that, as w_{LO} increases, the total energy consumption also increases. This is because on an average the LO mode utilization is higher than HI mode utilization for the generated task sets here. Hence, when LO mode weight factor increases, the total energy depends more on LO mode energy and increases.

3) Impact of task utilization: Clearly, with increasing system utilization, the system operates longer and it also increases the achievable minimal energy. We evaluate this by increasing the system utilization for generated task sets from 1 to 3 in steps of 0.2. The obtained results are shown in Figure 5c. For low system utilization – independent of task mapping techniques – all tasks can execute closely at $f_{\min}$ in both system modes and we observe minor or no differences among the different methods. This trend changes while increasing the system utilization. In fact at higher system utilizations both EM3 and IM3 perform better as they explore more the load balancing among cores to save energy. For instance, when system utilization is 3, EM3/IM3 saves $35\%/32\%$ more energy than Baruah's method and $23\%/21\%$ more than Gu's method.

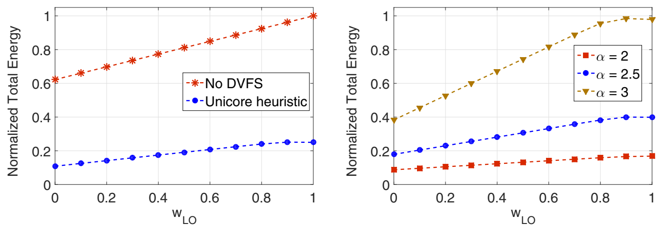
4) Impact of P_{HI}: To show the impact of P_{HI}, we vary P_{HI} in the range $[0, 1]$ in steps of 0.1. Figure 5d shows the normalized total energy for different mapping techniques with varying P_{HI}. When the number of HI criticality tasks increases, the extra workload adds up to the expected minimal energy. This can be observed in Baruah's method, EM3 and IM3. However, Gu's method is a special case; it only performs load balancing for HI criticality tasks and is mostly influenced by P_{HI}. With increasing P_{HI}, it performs better load balancing, which improves energy saving and diminishes the impact of increased HI mode workload.

D. Evaluation with a flight management system

For unicore, we conducted experiments on a real-world avionics application (a flight management system (FMS)), which is used for aircraft localization, nearest airport selection and trajectory computation. The task set consists of 11 tasks as listed in Table III. In all experiments, we assume the processor is DVFS-capable with $f_{\min}/f_b/f_{\max} = 0.5/0.8/1$GHz, $\beta = 1.76$W/GHz$^{\alpha}$ and $P_s = 0.8$W. We assume $\alpha = 2$, unless otherwise mentioned.

τ	Criticality	$T_i(ms)$	C_i(LO) (ms)	C_i(HI) (ms)
τ_1	HI	5000	15	21
τ_2	HI	200	18	25
τ_3	HI	1000	16	22
τ_4	HI	1600	20	28
τ_5	HI	100	18	26
τ_6	HI	1000	17	24
τ_7	HI	1000	15	21
τ_8	LO	1000	100	100
τ_9	LO	1000	80	80
τ_{10}	LO	1000	140	140
τ_{11}	LO	1000	100	100

TABLE III: FMS task set



(a) Energy minimization (b) Impact of weight factors

Fig. 6: Experiments on a flight management system

We apply our proposed unicore heuristic on FMS and summarize our results in Figure 6. Indeed, we can observe that energy minimization is achieved by applying our unicore heuristic: $30\% - 50\%$ of total energy is reduced by employing DVFS for weight factors in the range $0 - 1$ (see Figure 6a). Additionally, we observe the impact of weight factors on the minimal expected energy for $\alpha = 2, 2.5, 3$ in Figure 6b: The minimal expected energy increases with increasing w_{LO}, since LO mode utilization is higher than HI mode utilization in the considered task set; therefore, when more importance is given to minimize LO mode energy, the expected overall energy increases. Furthermore, as expected, with increasing α (dynamic power), the minimal system energy increases.

E. Schedulability evaluation with different mapping methods

We evaluate our proposed mapping methods for schedulability compared to the state-of-the-art techniques (Baruah's [4] and Gu's [13] methods). We generate 1000 random task sets at each utilization point similarly to our previous experiments, and map them onto a multi-core platform consisting of 4 cores. We use task generation parameters as follows: $\lambda = 1.25$, $[U_{ll}, U_{lu}] = [0.002, 0.02]$, $[U_{hl}, U_{hu}] = [0.01, 0.1]$ and $P_{\mathrm{HI}} = 0.2$. The number of task sets that are schedulable for different mapping techniques is evaluated with varying U_t in the range $2.7 - 3.1$, in steps of 0.1.

We summarize our results in Table IV. As we can see, when utilization is low ($U_t = 2.7$), no matter what mapping technique we use, all task sets are schedulable on 4 cores. As the system utilization increases, IM3 shows the worst performance in terms of schedulability as it cannot mix workload from different criticality levels to improve schedulability. In

U_t	Baruah	Gu	EM3	IM3
2.7	1000	1000	1000	1000
2.8	1000	1000	1000	823
2.9	1000	1000	1000	648
3.0	926	918	811	312
3.1	0	0	0	0

TABLE IV: Number of task sets that are schedulable with different mapping techniques

addition, we observe that Baruah's method is actually better than Gu's method in terms of schedulability. The reason is that in Gu's method, further fine tunings on mixing workloads on all cores and on EDF-VD scheduling are performed, which are not considered here. Last, we can find that the EM3 mapping technique has a schedulability performance close to Baruah's and Gu's methods.

VIII. Conclusion

Mixed-criticality systems are emerging as a significant trend for future automotive, avionic and medical systems. In this paper, we have explored energy minimization for such systems on modern DVFS-capable multi-core processors. Compared to the state of the art, we took a general setting where both static and dynamic energy consumption in all system operation modes are considered. To tackle the difficulty in trading off energy consumption in different modes to jointly minimize the overall energy, we first proposed an optimal unicore solution and then a corresponding low computation complexity heuristic. Based on this, we further developed energy-aware mapping techniques to explore energy savings on multi-cores. Experiments were conducted for both a flight management system and synthesized task sets. The results validated our proposed techniques, demonstrating energy savings as high as 35% (random simulations on multi-cores).

Our proposed techniques assume EDF scheduling and focus on processing energy of the CPUs. It would be interesting to further consider other scheduling techniques, energy sources (e.g. communication energy) and various system overheads (e.g. timing and energy overheads when switching between different power states). However, we believe that the important findings in this paper would still hold in such an extension, e.g. trade-offs between energy consumptions in different system modes and the energy benefits of isolated scheduling.

Acknowledgments The authors would like to thank the anonymous reviewers for their valuable feedbacks, and Rehan Ahmed for the suggestions on English writing.

References

[1] Energy-efficient solutions: Enabling a new generation of applications. https://cache.nxp.com/files/shared/doc/NEWENERGEFFICWP.pdf. Accessed: 2016-01-13.

[2] R. D. A. Burns. Mixed criticality systems - a review. www-users.cs.york.ac.uk/burns/review.pdf, August 2015.

[3] 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 *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, pages 145–154, July 2012.

[4] Baruah, S. and Chattopadhyay, B. and Li, H. and Shin, I. Mixed-criticality scheduling on multiprocessors. *Real-Time Systems*, 50(1):142–177, 2014.

[5] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on VLSI*, 8(3):299–316, 2000.

[6] P. Bieber, F. Boniol, M. Boyer, E. Noulard, and C. Pagetti. New challenges for future avionic architectures. *AerospaceLab*, (4):p–1, 2012.

[7] Boyd, S. and Vandenberghe, L. Convex optimization, cambridge university press, isbn: 9780521833783. 2004.

[8] J.-J. Chen and C.-F. Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms. In *Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 28–38. IEEE, 2007.

[9] J.-J. Chen and C.-F. Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms. In *IEEE RTCSA*, pages 28–38, Aug 2007.

[10] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu. Memory power management via dynamic voltage/frequency scaling. In *Proceedings of the 8th ACM international conference on Autonomic computing*, pages 31–40. ACM, 2011.

[11] D. Duarte, N. Vijaykrishnan, M. Irwin, H.-S. Kim, and G. McFarland. Impact of scaling on the effectiveness of dynamic power reduction schemes. In *Proceedings, Computer Design: VLSI in Computers and Processors, IEEE*, pages 382–387, 2002.

[12] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *Proceedings, Symposium on ICSA*, pages 365–376. IEEE, 2011.

[13] C. Gu, N. Guan, Q. Deng, and W. Yi. Partitioned mixed-criticality scheduling on multiprocessor platforms. In *DATE*, pages 1–6. IEEE, 2014.

[14] N. Guan, P. Ekberg, M. Stigge, and W. Yi. Effective and efficient scheduling of certifiable mixed-criticality sporadic task systems. In *IEEE RTSS*, pages 13–23, 2011.

[15] C. Guo. Empirical study of energy minimization issues for mixed-criticality systems with reliability constraints. 2014.

[16] P. Huang, P. Kumar, G. Giannopoulou, and L. Thiele. Energy efficient dvfs scheduling for mixed-criticality systems. In *Embedded Software (EMSOFT)*, pages 1–10, Oct 2014.

[17] P. Huang, H. Yang, and L. Thiele. On the scheduling of fault-tolerant mixed-criticality systems. In *Proceedings of DAC*, DAC '14, pages 131:1–131:6, 2014.

[18] O. Kelly, H. Aydin, and B. Zhao. On partitioned scheduling of fixed-priority mixed-criticality task sets. In *IEEE TrustCom*, pages 1051–1059, 2011.

[19] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *computer*, 36(12):68–75, 2003.

[20] H. Kuhn and A. Tucker. Nonlinear programming. In *Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, 1951.

[21] V. Legout, M. Jan, and L. Pautet. Mixed-criticality multiprocessor real-time systems: Energy consumption vs deadline misses. In *First Workshop on Real-Time Mixed Criticality Systems (ReTiMiCS)*, pages 1–6, 2013.

[22] H. Li and S. Baruah. Global mixed-criticality scheduling on multiprocessors. In *ECRTS, p:166-175*, 2012.

[23] A. Nelson, O. Moreira, A. Molnos, S. Stuijk, B. Nguyen, and K. Goossens. Power minimisation for real-time dataflow applications. In *2011 14th Euromicro Conference on Digital System Design (DSD)*, pages 117–124, Aug 2011.

[24] S. Pagani and J.-J. Chen. Energy efficiency analysis for the single frequency approximation (sfa) scheme. In *IEEE RTCSA*, pages 82–91, 2013.

[25] S. Pagani and J.-J. Chen. Energy efficient task partitioning based on the single frequency approximation scheme. In *IEEE RTSS*, pages 308–318, 2013.

[26] F. Santy, L. George, P. Thierry, and J. Goossens. Relaxing mixed-criticality scheduling strictness for task sets scheduled with fp. In *ECRTS*, pages 155–165. IEEE, 2012.

[27] T. Vogelsang. Understanding the energy consumption of dynamic random access memories. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microar-*

chitecture, pages 363–374. IEEE Computer Society, 2010.

[28] M. Volp, M. Hahnel, and A. Lackorzynski. Has energy surpassed timeliness? scheduling energy-constrained mixed-criticality systems. In *IEEE RTAS*, pages 275–284, 2014.

[29] K. W. Wong. System and method for providing a thermal shutdown circuit with temperature warning flags, Nov. 18 2008. US Patent 7,454,640.

[30] H. Yun, P.-L. Wu, A. Arya, C. Kim, T. Abdelzaher, and L. Sha. System-wide energy optimization for multiple dvs components and real-time tasks. *Real-Time Systems*, 47(5):489–515, 2011.

APPENDIX A
PROOF OF THEOREM 5

Proof: First, let us consider only HI mode energy and show that all HI criticality tasks should share same execution frequency in HI mode in an optimal solution. Likewise, we can prove similar statements for LO mode task frequencies.

Considering only two HI criticality tasks τ_i and τ_j in the system, we prove using KKT conditions that they share same execution frequency; by induction, this will hold for any pair of HI criticality tasks and our statement follows.

We denote HI mode utilizations of τ_i and τ_j on base frequency f_b as u_i and u_j ($u_i = \frac{C_i(\mathrm{HI})}{T_i}$, $u_j = \frac{C_j(\mathrm{HI})}{T_j}$), respectively. Using (5), we can compute HI mode energy as

$$E_{\mathrm{HI}} = w_{\mathrm{HI}}\left(u_i\frac{f_b}{f_i^{\mathrm{HI}}}\left(P_s + \beta(f_i^{\mathrm{HI}})^{\alpha}\right)\right) + w_{\mathrm{HI}}\left(u_j\frac{f_b}{f_j^{\mathrm{HI}}}\left(P_s + \beta(f_j^{\mathrm{HI}})^{\alpha}\right)\right) \tag{18}$$

Let u_{i+j} be the allowed total HI criticality system utilization after any DVFS strategy. In (18), f_i^{HI} and f_j^{HI} are the variables to be decided, such that while applying DVFS to minimize energy E_{HI}, $u_i\frac{f_b}{f_i^{\mathrm{HI}}} + u_j\frac{f_b}{f_j^{\mathrm{HI}}} \le u_{i+j}$ so that system schedulability is preserved. We find the optimal f_i^{HI} and f_j^{HI} using KKT conditions. Let us first summarize our problem as

$$\textbf{minimize } (18) \quad \textbf{s.t. } u_i\frac{f_b}{f_i^{\mathrm{HI}}} + u_j\frac{f_b}{f_j^{\mathrm{HI}}} \le u_{i+j} \tag{19}$$

Based on Theorem 3, let us adjoin the inequality constraint with the energy objective and introduce a KKT multiplier (μ), (18) is at a minimum when

$$\nabla_{(f_i^{\mathrm{HI}},f_j^{\mathrm{HI}})}E_{\mathrm{HI}} + \mu\nabla_{(f_i^{\mathrm{HI}},f_j^{\mathrm{HI}})}g\left(f_i^{\mathrm{HI}}, f_j^{\mathrm{HI}}\right) = 0 \tag{20}$$

$$\textbf{s.t. } g\left(f_i^{\mathrm{HI}}, f_j^{\mathrm{HI}}\right) = u_i\frac{f_b}{f_i^{\mathrm{HI}}} + u_j\frac{f_b}{f_j^{\mathrm{HI}}} - u_{i+j} \le 0; \quad \mu\left(u_i\frac{f_b}{f_i^{\mathrm{HI}}} + u_j\frac{f_b}{f_j^{\mathrm{HI}}} - u_{i+j}\right) = 0 \wedge \mu \ge 0 \tag{21}$$

Under KKT, (20) and (21) must both hold, by solving (20) alone further, we have

$$\begin{pmatrix} \frac{\partial(\cdot)}{\partial f_i^{\mathrm{HI}}} \\ \frac{\partial(\cdot)}{\partial f_j^{\mathrm{HI}}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{22}$$

$$\Leftrightarrow w_{\mathrm{HI}}f_bu_i(-P_s(f_i^{\mathrm{HI}})^{-2} + \beta(\alpha-1)(f_i^{\mathrm{HI}})^{\alpha-2}) = \mu f_bu_i(f_i^{\mathrm{HI}})^{-2}$$

$$\Leftrightarrow \beta(\alpha-1)(f_i^{\mathrm{HI}})^{\alpha} = P_s + \frac{\mu}{w_{\mathrm{HI}}}$$

$$\Leftrightarrow f_i^{\mathrm{HI}} = \left(\frac{P_s + \frac{\mu}{w_{\mathrm{HI}}}}{\beta(\alpha-1)}\right)^{1/\alpha} \tag{23}$$

and similarly,

$$\Leftrightarrow w_{\mathrm{HI}}f_bu_j(-P_s(f_j^{\mathrm{HI}})^{-2} + \beta(\alpha-1)(f_j^{\mathrm{HI}})^{\alpha-2}) = \mu f_bu_j(f_j^{\mathrm{HI}})^{-2}$$

$$\Leftrightarrow \beta(\alpha-1)(f_j^{\mathrm{HI}})^{\alpha} = P_s + \frac{\mu}{w_{\mathrm{HI}}}$$

$$\Leftrightarrow f_j^{\mathrm{HI}} = \left(\frac{P_s + \frac{\mu}{w_{\mathrm{HI}}}}{\beta(\alpha-1)}\right)^{1/\alpha} \tag{24}$$

From (23) and (24), we obtain $f_i^{\mathrm{HI}} = f_j^{\mathrm{HI}}$ for which (18) is minimized. Hence, both tasks share the same execution frequency. By induction, any pair of HI criticality tasks share the same frequency and our statement holds. Similarly, it can be proved that all LO criticality tasks in LO system mode share same execution frequency and all HI criticality tasks in LO system mode share same execution frequency. ■

APPENDIX B
1ST ORDER DIFFERENTIAL CALCULATION FOR ALGORITHM 1

After presenting the general idea for our heuristic solution, we proceed to detail how we calculate E'_{LOopt} and E'_{HIopt}. E_{HI} is a single-variate monotonically increasing function of $f_{\mathrm{HI}}^{\mathrm{HI}}$ (see (12)). Therefore, with fixed $f_{\mathrm{HI}}^{\mathrm{HI}}$, E_{HIopt} equals E_{HI} and the first order differential can be directly calculated. For E'_{LOopt}, the calculation is less trivial as we still need to find the right LO mode task frequencies leading to the minimal LO mode energy. We achieve this by first deriving E_{LOopt} for any given $f_{\mathrm{HI}}^{\mathrm{HI}}$ based on a similar condition to Theorem 6; we then calculate E'_{LOopt} empirically as the principal linear part.

Considering only E_{LO}, with fixed $f_{\mathrm{HI}}^{\mathrm{HI}}$, we can prove the LO mode energy is a minimum either when all LO mode task frequencies are minimum ($f_{\min}$) or at the equilibrium case, similarly to Theorem 6. For the former case, E_{LOopt} can be directly calculated; for the equilibrium case, we have

$$\frac{K/f_{\mathrm{HI\,opt}}^{\mathrm{LO}}}{1 - L/f_{\mathrm{HI\,opt}}^{\mathrm{LO}}} = x_{\mathrm{opt}} = \frac{M - K/f_{\mathrm{HI\,opt}}^{\mathrm{LO}}}{L/f_{\mathrm{LO\,opt}}^{\mathrm{LO}}}, \tag{25}$$

$$\Leftrightarrow \frac{K/f_{\mathrm{HI\,opt}}^{\mathrm{LO}} + M - K/f_{\mathrm{HI\,opt}}^{\mathrm{LO}}}{1 - L/f_{\mathrm{LO\,opt}}^{\mathrm{LO}} + L/f_{\mathrm{LO\,opt}}^{\mathrm{LO}}} = x_{\mathrm{opt}},$$

$$\Leftrightarrow x_{\mathrm{opt}} = M,$$

where M is a function of $f_{\mathrm{HI}}^{\mathrm{HI}}$ defined in (13) ($f_{\mathrm{HI\,opt}}^{\mathrm{HI}}$ replaced with $f_{\mathrm{HI}}^{\mathrm{HI}}$ as we are fixing the latter). We can derive one step further and establish a relation between $f_{\mathrm{HI\,opt}}^{\mathrm{LO}}$ and $f_{\mathrm{LO\,opt}}^{\mathrm{LO}}$:

$$M = \frac{M - K/f_{\mathrm{HI\,opt}}^{\mathrm{LO}}}{L/f_{\mathrm{LO\,opt}}^{\mathrm{LO}}} \Leftrightarrow f_{\mathrm{HI\,opt}}^{\mathrm{LO}} = \frac{K/M}{1 - L/f_{\mathrm{LO\,opt}}^{\mathrm{LO}}}. \tag{26}$$

Thus, we can represent $f_{\mathrm{HI\,opt}}^{\mathrm{LO}}$ as a function of $f_{\mathrm{LO\,opt}}^{\mathrm{LO}}$; consequently, to find E_{LOopt}, we finally have a single-variate optimization problem in a constrained search space where both $f_{\mathrm{HI\,opt}}^{\mathrm{LO}}$ and $f_{\mathrm{LO\,opt}}^{\mathrm{LO}}$ must be drawn from the feasible frequency space. Such a problem can be easily solved by looking at the first order conditions for optimization.