# Orchestrating Mixed-Criticality Melody

## Reconciling Energy with Safety for Mixed-Criticality Embedded Real-Time Systems

Sujay Narayana

Master of Science Thesis


TU Delft Delft University of Technology

Embedded Software

# Orchestrating Mixed-Criticality Melody

## Reconciling Energy with Safety for Mixed-Criticality Embedded Real-Time Systems

---

Master of Science Thesis

For the degree of Master of Science in
Embedded Systems
at Delft University of Technology

By
Sujay Narayana

August 28, 2015


Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft
The Netherlands



The work in this thesis was supported by Embedded Software Department, TU Delft and was carried out at Computer Engineering and Networks group of Department of Information Technology and Electrical Engineering, Swiss Federal Institute of Technology (ETH), Zurich. Their cooperation is hereby gratefully acknowledged.


**Delft University of Technology**
**Dept. of Software and Computer Technology**

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for the acceptance of a thesis entitled

# Orchestrating Mixed-Criticality Melody :

## Reconciling Energy with Safety for Mixed-Criticality Embedded Real-Time Systems

by

Sujay Narayana

in partial fulfillment of the requirements for the degree of

Master of Science Embedded Systems

Dated: August 28, 2015

Chairman: _____
Prof. dr. K.G. Langendoen

Supervisor (TU Delft): _____
Dr. R. Venkatesha Prasad

Supervisor (ETH Zurich): _____
Prof. Dr. Lothar Thiele

Committee member: _____
Dr. ir. Fernando Kuipers

# Abstract

Embedded systems are getting into various domains of our daily life as well as in many of the highly sophisticated large systems, such as air planes, military tanks, rockets, satellites. These large systems consist of many modules that are executing umpteen number of tasks semi-independently. However, not all tasks have the same levels of priority and/or criticality. One way is to design individual systems with dedicated processors to avoid the dependency as proposed by the industry. However, mixed-criticality notion helps to enhance system performance, and reduce system cost, size, and weight. The idea is to integrate functionalities of different safety criticality levels into a common computing platform. Further, the energy consumption of these systems should also be taken into account. While there are many algorithms under the broad umbrella of scheduling - preemptive, non-preemptive, etc., – solutions that jointly minimize both static and dynamic energy consumption in mixed-criticality systems on multi-cores under partitioned scheduling are, hitherto, not addressed in depth.

To reconcile the conflicting requirements of safety and energy: (i) we formulate a general energy minimization problem; (ii) we provide an analytical optimal solution on unicore systems and a corresponding low-complexity heuristic and (iii) we provide energy-aware mapping techniques based on our unicore solutions on multi-cores. Effectiveness in energy reduction is demonstrated for our solutions through extensive simulations with synthetic task sets.

# Table of Contents

# List of Figures

# List of Notations

| | |
|---|---|
| $\tau$ | Mixed criticality task set mapped on unicore or multi-core. |
| $\chi$ | Criticality level, $\chi \in \{LO\,, \text{HI}\}$. |
| $C_i(\chi)$ | WCET of task $\tau_i$ in $\chi$ criticality mode. |
| $T_i$ | Minimum inter-arrival time of task $\tau_i$. |
| $f_b$ | Frequency at which WCETs $C_i(\chi)$ are calculated. |
| $[f_{min}\,,\ f_{max}]$ | Processor frequency range within which DVFS can be employed. |
| $f_i^{\chi}$ | Frequency of task $\tau_i$ in $\chi$ mode. |
| $f_{\chi 1}^{\chi 2}$ | Frequency at which all $\chi_1$ criticality tasks can be executed in $\chi_2$ mode. |
| $f_{\chi 1\,opt}^{\chi 2}$ | Optimum frequency at which all $\chi_1$ criticality tasks can be executed in $\chi_2$ mode. |
| $f_{\chi 1\,min}^{\chi 2}$ | Minimum required frequency at which all $\chi_1$ criticality tasks must be executed in $\chi_2$ mode so that the taskset is schedulable. |
| $U_{\chi 1}^{\chi 2}$ | $\chi_2$ mode utilization of all $\chi_1$ criticality tasks. |
| $E_{\chi}$ | Total normalized energy of the system in $\chi$ mode. |
| $w_{\chi}$ | $\chi$ mode weight factor. |
| $x$ | Deadline shortening factor in the EDF-VD algorithm. |
| $[\hat{x}_{\text{LB}}\,,\ \hat{x}_{\text{UB}}]$ | Maximum feasible range of $x$. |
| $[x_{\text{LB}}\,, x_{\text{UB}}]$ | Lower and upper bound of $x$. |
| $[x_{\text{LB}opt}\,, x_{\text{UB}opt}]$ | Optimal lower and upper bound of $x$. |
| $x_{opt}$ | Optimal $x$ at which the energy consumption is minimum. |
| $[\![a]\!]^c$ | $\min(a, c)$. |
| $[\![a]\!]_c$ | $\max(a, c)$. |
| $P_s$ | Normalized static power. |
| $\alpha$ | Activity factor which is constant for a processor. |
| $\beta$ | Dynamic power dissipation capacitance of a processor. |
| $m$ | Number of cores in the processor. |

# Acknowledgements

The work presented in this thesis was carried out at Swiss Federal Institute of Technology (ETH, Zurich) with support from the Embedded Software group of TU Delft.

Firstly, I wholeheartedly thank Prof. Dr. Lothar Thiele for providing me an opportunity to work under his guidance at Computer Engineering group, ETH Zurich. I am grateful to him for his support in getting insights into this new topic. I am indebted to him for giving me time and for the discussions I had with him.

I express my gratitude to Dr. R. Venkatesha Prasad who has been supporting me as more than a supervisor all the way through. He has also been my guide, support and motivation for many of my courses and project works. Saying just "Thanks" would not be sufficient as I owe him much more.

My special thanks to Ir. Pengcheng Huang and Ms. Georgia Giannopoulou for their valuable inputs at each step of my thesis work at ETH Zurich.

I would like to extend my thanks to Ir. Vijay S. Rao for helping me in various ways during my stay at Delft.

Furthermore, I would also like to acknowledge my parents and my friends who have always kept me motivating during my entire study period.

Finally, I would appreciate the support from IDEA League committee for providing me the scholarship with the agreement to work at ETH Zurich.

Delft, University of Technology — Sujay Narayana
August 28, 2015

# Chapter 1

# Introduction

In recent days, most of the computer systems currently built in areas such as avionics, automotive industry, medical and robotic applications are mixed-critical [1] – they contain complex functionalities for different safety-criticality levels. Mixed-criticality is the concept of integrating applications with different criticality levels (safety[1], mission, real-time, non real-time) on a single computational platform. For instance, in real-time systems such as Unmanned Aerial Vehicles (UAV), there are safety-critical functionalities (stabilization) and mission-critical applications (surveillance). When these tasks fail there are different consequences based on their criticality levels. Thus it is important to consider the requirements of these tasks based on their criticality levels. We explain many aspects of mixed criticality systems in the sequel below.

The applications are consolidated into common computing platforms to further improve scalability, increase reliability, enhance system performance, and reduce system cost, size, and weight. The emergence of such mixed-criticality systems, while increasing market competitiveness of their developers, pose tremendous challenges in terms of analyzing and certifying system properties like timing and safety [2]. This is largely due to the fact that sharing resources among different criticality levels could lead to mutual interferences jeopardizing their properties which could have been certified otherwise under no resource sharing [3]. For example, if a low criticality task occupies a shared resource for excessive amount of time, being it the processor, bus, memory or IO, the execution of a high criticality task could be greatly impacted, leading to missed deadlines and jeopardizing system safety. To ensure timing safety, there arises a need for proper scheduling techniques that can guarantee the execution of high criticality tasks within their deadlines. In recent years, many scheduling techniques for mixed-criticality systems have been proposed [1].

[1]Here the notion of *safety* is that the tasks **must** be completed within the given *hard deadline*. Now on, *Criticality* here means safety criticality unless explicitly mentioned otherwise.

One of the main reasons for mixing tasks together on a single platform is to minimize system energy. Battery operated devices such as pacemakers, UAVs and satellites may be mixed-critical and energy becomes a main concern in these devices. However, energy reduction should not trade-off timing safety of high critical tasks. Handling mixed-criticality system is already difficult and we are adding one more dimension to this – *energy*. The main goal of this thesis is to minimize energy on such systems without violating timing safety and schedulability of the system. We propose an optimal solution and also a low complexity heuristic to minimize energy on mixed-criticality systems. With one of our multi-core mapping techniques, we also show that almost the same amount of energy can be saved even without mixing tasks on a processor core. The overview of this chapter is as follows: In Section 1.1, we discuss the background of mixed-critical systems and what is certification in the mixed-criticality systems. In Section 1.2, we explain with an example, why scheduling is an important problem in mixed-criticality system. Section 1.3 describes the motivation for this research and our contributions are listed in Section 1.4. Finally, the outline of this report is presented in Section 1.5.

## 1.1 Mixed-Criticality revisited

In the current decade, it has become a common trend in real-time embedded systems to integrate multiple applications on a common platform. Devices such as Unmanned Aerial Vehicles (UAV), which operate in proximity to large civilian populations, must guarantee safety along with their mission objectives [4, 5]. In such systems, the criticality level of the system in terms of functionality can be divided into two groups:

- **Safety-critical:** This group involves the applications that are performed to ensure safe operation of the system. For instance, tasks such as flight control and stability, actuation and power system control in avionics and UAVs fall under safety-critical operations. To operate such devices over civilian locations, they must be certified for flight critical functionalities by statutory Certification Authorities (CA). Safety systems can be further classified as fail-safe and fail-operational. A system is fail-safe if it can reach a safe state in case of failures such that no harm is caused to other devices or personnel (e.g., a train that can be stopped during emergency). A system is fail-operational if no safe state can be reached in case of a system failure (e.g., a rocket losing its stability). Such systems must always be reliable, robust and be able to take decisions immediately to ensure safety. In this thesis, we refer to fail-operational systems.
- **Mission-critical:** Tasks that perform mission related operations such as navigation, video surveillance and weapons management in UAVs, external communication and entertainment systems in flights, driving comfort features such as reverse/backing aid, power steering and night vision fall under mission critical functionalities. These applications are also validated against completion of tasks within predefined deadlines but are not necessarily certified as rigorously as safety-critical applications.

For the integration of mixed-criticality system on a single computational platform, the safety-critical tasks should be certified properly to promise timing safety: these tasks are expected to complete their execution within their specified deadlines. Currently the certification of mixed-criticality system is done in two steps [1]:

- **Non-safety-critical certification:** System designers (developers) analyze both safety and non-safety-critical jobs and estimate their Worst Case Execution Times (WCETs), i.e., the time taken by a task to complete its execution in worst case scenarios. With the measured WCET, the system designers determine if both the low and high critical jobs are correctly executed within their deadlines. However, the certification process is less rigorous than CA's certification process on safety-critical tasks, especially for WCET estimation. The WCETs proposed by system developers are termed as LO WCETs (which is followed in the literature).
- **Safety-critical certification:** Safety-critical applications are always certified by Certification Authorities (CA). Since the tasks involved are of high criticality levels, CAs apply complex methods to analyze them, and they make some assumption on their WCETs. To ensure safety, CAs tends to be very conservative and the WCET estimations are more pessimistic than those the system designer would specify. With such WCETs, the high criticality tasks are always expected to complete their execution within their deadlines. The WCETs proposed by CAs are termed as HI[2] WCETs.

The presence of two WCETs for a safety-critical task makes the system scheduling difficult as the task may switch between LO WCET or HI WCET at any time. Also, the system is unaware of random overruns, when the task may execute in its LO WCET or HI WCET. Considering only HI WCET during scheduling may underutilize the system and only LO WCET may fail to guarantee timing safety.

To evaluate the safety of mixed-criticality systems, several certification standards exist for major industries like automotive and avionics. Few of them are:

- ARINC 653 (Avionics Application Standard Software Interface) is a software standard for safety-critical Real-Time Operating Systems in avionics department. The standard mainly relies on partitioning system where applications with different criticality levels are temporally isolated [6, 7].
- DO-178B is one of the standards in Airborne Systems and Equipment Certification used by the U.S. Federal Aviation Administration (FAA) for the certification of commercial airplanes. The standard allows up to five criticality design assurance levels A (very high critical level) to E (least critical level), namely [8]:

  *Level A - Catastrophic:* Very high critical level because failures may cause a flight crash and the consequences are fatal, affecting lives.

[2]The notations LO and HI are used here adhering to the convention in the literature on mixed-criticality systems.

*Level B - Hazardous:* High critical level, failures have a negative impact on safety. The flight has to be landed immediately else, causes fatal injuries among the passengers.

*Level C - Major:* Significantly critical, but failure has a lesser impact than Hazardous level. Failure may cause discomfort to passengers.

*Level D - Minor:* Less critical applications whose failure is noticeable but may cause little inconvenience for passengers.

*Level E - No Effect:* Failure has no effect on the safety of aircraft or passengers.

- Automotive Open System Architecture (AUTOSAR) is a software standard for certifying safety-critical applications in automotive domain [9]. The standard is used to certify different subsystems and ECU software in the automotive industry.

The certification process is very tedious as even the least critical components have to be certified at the highest criticality level sometimes because of resource sharing and interdependency among tasks. When tasks with different criticality levels execute on the same platform, there may exist the interference of low criticality tasks on high criticality tasks. Hence, certification of safety according to industrial standards is a great challenge to guarantee interference freeness among tasks. As an easy way, industry prefers isolated system where tasks with different criticality levels are not mixed on a processor. Though energy, cost and resources increase due to multiple processors, timing safety is ensured.

The ultimate goal of certification in mixed-criticality systems is to assure the execution of tasks with different criticality levels in the system without jeopardizing safety/security guarantees. In many applications such as rockets and missiles, some systems may undergo multiple sets of certification tests to ensure additional safety. This gives rise to task scheduling problem as there will be more than one WCET for high criticality tasks, one provided by system designers and another by CAs. In next section, we describe with an example, why scheduling mixed-criticality tasks is not as simple as that of normal real-time systems.

## 1.2 Mixed-criticality scheduling

In literature, we find many techniques and algorithms to schedule tasks in real-time systems [10, 11]. In 1973, Liu and Layland presented an optimal scheduling technique called Rate Monotonic (RM) to schedule fixed priority periodic real-time tasks [12]. Earliest Deadline First (EDF) is one of the efficient and optimal algorithm for dynamic scheduling of preemptive periodic and aperiodic tasks [13]. In EDF, priority for execution is assigned to a job with the earliest deadline, during run-time. EDF has 100% processor utilization factor as it can schedule any task set with system task utilization (sum of the value of each task's WCET divided by its period) less than or equal to 1. However, EDF is designed for single WCET tasks and is not aware of random overruns that can occur in mixed-criticality systems.

| $\tau$ | Criticality | $T(ms)$ | $C$(LO) (ms) | $C$(HI) (ms) |
|---|---|---|---|---|
| $\tau_1$ | LO | 4 | 2 | 2 |
| $\tau_2$ | HI | 6 | 1 | 5 |

**Table 1.1:** A mixed-criticality taskset

We show with an example, why most efficient techniques such as EDF may fail in scheduling mixed-criticality systems.

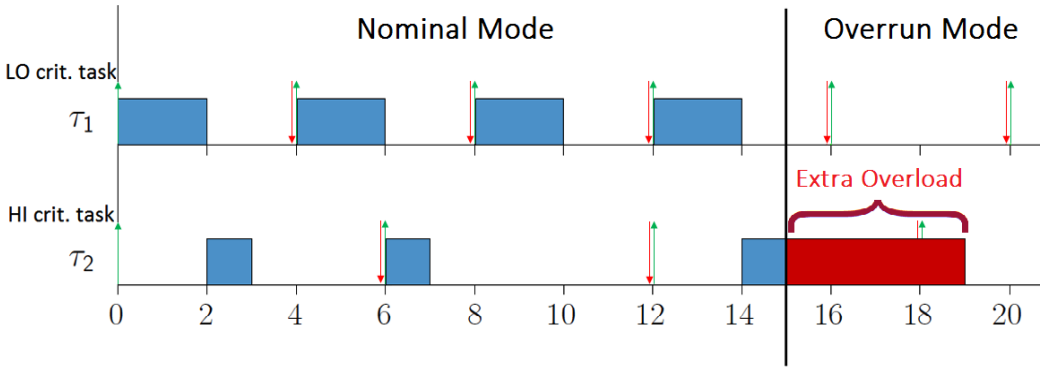


**Figure 1.1:** Mixed-criticality task set scheduled under EDF

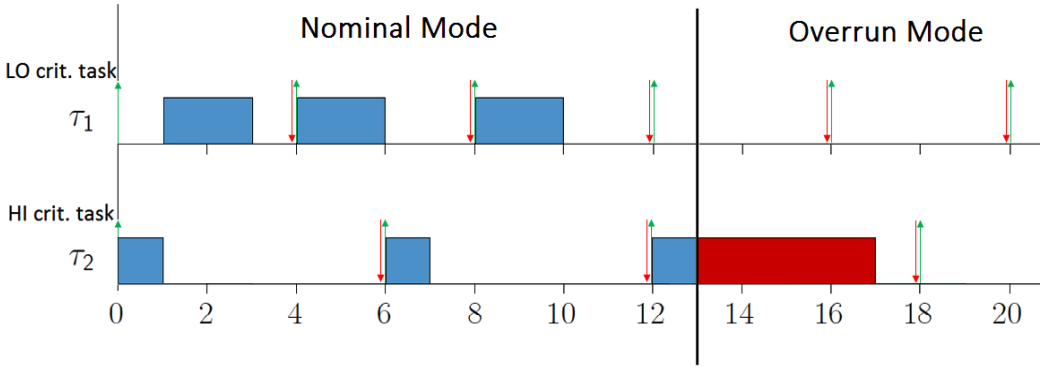


**Figure 1.2:** Mixed-criticality task set scheduled successfully

**Example 1.** Let us consider a mixed-criticality task set $\tau$ to be implemented on a preemptive uniprocessor. The system is comprised of dual-criticality tasks: one low (LO) criticality task $\tau_1$ and one high (HI) criticality task $\tau_2$. $\tau_2$ is subject to certification whereas $\tau_1$ not. The considered tasks with their low and high criticality WCETS and arrival time periods are listed in Table 1.1. $C_i$(LO) is the WCET estimated by the developer and $C_i$(HI) by CAs. We assume that all tasks initially release jobs at time zero and have the deadlines equal to their respective periods. Let us employ EDF to schedule the tasks as shown in Figure 1.1.

Let us assume that the system starts in nominal mode (LO criticality mode) where both low and high criticality tasks are executed with their nominal WCETs. At time zero, since $\tau_1$ has the earliest deadline of $4ms$, it gets to execute first accord-

ing to EDF test. It completes execution at $2ms$ and then $\tau_2$ starts executing. The process continues until $14ms$ where both the tasks complete their execution within their respective deadlines. At time $14ms$, $\tau_2$ starts executing its third job. Since $\tau_2$ is of high criticality level, let us assume that it cannot complete its execution within its low criticality WCET of $1ms$. Therefore, at $15ms$, the system shows high criticality behaviour (HI criticality mode) where $\tau_2$ executes for extra $C_2(\text{HI}) - C_2(\text{LO})$ workload i.e, $4ms$. This requires $\tau_2$ to execute until $19ms$ but it has already missed its deadline at $18ms$. Hence, the system scheduled using EDF fails certification.

On the other hand, if we prioritize high criticality task even in nominal scenario when tasks do not overrun, then $\tau_2$ would meet their deadlines even under the worst-case scenarios estimated by the Certification Authority. This is shown in Figure 1.2. However, the job released by low criticality task $\tau_1$ at $12ms$ fails to execute within its deadline but without affecting the safety. Thus, we achieve the goal of mixed-criticality system by ensuring safety even if we put low criticality task in jeopardy. Therefore, to promise safety, most of the scheduling algorithms drop tasks with a criticality level less or equal to present executing task that overruns its nominal WCET.

Example 1 clearly shows that the requirements in mixed-criticality cannot be completely addressed using EDF. This is because, EDF assumes that all tasks are equally critical, so do the other traditional scheduling techniques such as RM. This gives rise to important and interesting scheduling problems in the mixed-criticality system.

In next sections, we discuss briefly about processor energy, motivations for this thesis work and our contributions.

## 1.3 Motivation and Problem statement

Much of the existing and growing literature on mixed-criticality systems targets functional requirements such as scheduling and resource management [1]. However, to our best knowledge, non-functional properties like energy consumption are not explored much. In battery operated mixed-criticality devices such as drones, robots and medical gadgets, energy minimization is more prominent to minimize power costs. The energy issue is an active area of research with the goal of extending the battery life of these portable consumer devices. The power consumption in modern processors can be divided into two categories: Dynamic power and Static power.

1. **Dynamic power:** Dynamic power consumption is caused due to the switching activity of transistors, and charging and discharging of capacitors in processors when tasks are executed. This power is proportional to the supply voltage and frequency at which the tasks are executed [14, 15]. Dynamic Voltage and Frequency Scaling (DVFS) is a common way of reducing dynamic power consumption in real-time systems and is followed from past two decades [14, 16, 15]. The basic idea of the DVFS strategy is to trade off performance for power consumption by lowering the operating voltage and

frequency of the processor, thereby extending the task execution time. However, the task deadline should be still met.

2. **Static power:** Static power is leakage power in the circuit which exists even in the absence of any CMOS switching activity. Static power can be minimized by shutting off the processor cores when there are no tasks executing [17, 18]. Even though static power consumption is comparatively less than dynamic power consumption in previous process technology nodes, this is not true anymore. As the CMOS technology level in processor design reduces, the static power accounts more of the total power (up to 44 % in 50 $nm$ technology) [19, 20].

In this paper, we address the energy reduction of mixed-criticality systems without violating the timing safety. With continuous shrinking of circuit footprint, the issue of operating temperature and the battery operated nature of many electronic systems, energy saving is becoming a prominent issue [21]. Unfortunately, so far, there are only a few results such as [22] that addresses energy in mixed-criticality systems. This is the strong motivation for this research followed by the following rationale:

1. **Energy minimization:** The total energy of the system is the sum of both dynamic and static energy and are to be minimized. Since static energy also has almost equal weight as dynamic energy, there is a need for minimizing static energy along with dynamic energy. There are tremendous works in literature where energy-aware scheduling is performed on real-time systems using online or offline DVFS. However, energy minimization in mixed-criticality systems is not explored much.

2. **Energy minimization vs Scheduling:** Proper scheduling is necessary and energy minimization is important in mixed-criticality systems. To guarantee safety, the high criticality tasks should execute as fast as possible so that they can still meet their deadlines even if they overrun their low criticality WCETs. On the contrary, DVFS minimizes energy by stretching task execution time by lowering the execution frequency. Therefore, the conflicts between energy minimization and guaranteeing safety have to be carefully handled so that both energy minimization and safety are achieved.

3. **Trade-offs between nominal and critical mode:** To guarantee safety, speeding up the overload of the system such that high criticality tasks can still meet their deadlines is intuitive. This even helps in reducing nominal energy consumption. This problem has already been addressed by Huang et.al. in [23]. However, energy minimization in the overrun mode is also important as the system can continue in overrun mode for a long duration. It becomes unclear in this case to speedup the execution to guarantee safety: the task can complete more workload in nominal mode by increasing execution frequency such that more slack is available in overrun mode to guarantee safety and to reduce overload energy; or save nominal energy by speeding up extra workload if the system is active for less time in overrun mode. Hence, there

exists a trade-off between energy consumptions in both modes subjecting to safety constraint, making the overall energy minimization problem complex.

4. **Energy aware scheduling on multi-cores:** Multi-core is an emerging technology and are widely used in mobile real-time systems such as smart phones, tablets, and laptops. Multi-cores deliver a higher throughput at lower power consumption than unicores. With safety as objective, many researchers have proposed scheduling algorithms and task mapping techniques for multi-core platform [1, 24, 25] in mixed-criticality systems. However, there is hardly any work done in terms of energy. The energy problem on multi-cores is even more complex than on unicores. There are many factors to be considered that add extra effort in case of multi-cores: choosing the optimal number of cores for execution as more cores lead to increase in static energy and fewer number of cores forces to increase execution frequency, in turn, dynamic energy; selecting energy-aware bin packing techniques [26, 27, 28]; scheduling paradigms (global or partitioned) in terms of energy [29, 30], etc.

5. **Isolation and energy minimization:** Though mixed-criticality notion reduces cost and size of the system, traditional isolation methods are preferred in the industry to guarantee safety [31]. According to this standard, tasks with different criticality levels are not mixed on the same processor core. This prevents the interference of low criticality tasks on high criticality tasks. However, it has not been explored yet, how good to "mix" or "isolate" tasks on different cores when it comes to energy minimization.

Hence, the overall energy minimization problem on unicore and multi-core in mixed-criticality systems is important and challenging. Even the energy issue on the dual-criticality and uniprocessor system is unsolved yet. This strongly motivated us to put our efforts in this work where we minimize the energy consumption of mixed-criticality systems without violating the timing safety of highly critical tasks.

## 1.4 Contribution

In this work, we address energy minimization problem in mixed-criticality systems and our contributions are many folds:

- We formulate a general energy minimization problem on both unicore and multi-cores and propose an optimal solution while considering system safety and mutual interferences among different criticality levels.
- As opposed to the state of the art where energy saving is considered only in nominal mode, we consider energy minimization in both nominal and overrun scenarios.
- We also reduce static energy consumption considerably in unicore and multi-cores using DVFS and also by switching the processor into sleep mode when it is idle. In addition, we adjust the number of active cores to reduce leakage power consumption in multi-cores.

- We present a computationally attractive heuristic with comparable performance to the optimal solution on unicore and multi-cores.
- We propose an energy efficient multi-core mapping technique extending our unicore solution to judiciously explore the energy savings on multi-cores in mixed-criticality systems.
- We additionally provide an energy-aware isolated mapping technique for multi-cores where the tasks with different criticality levels are not mixed as preferred by the industry.
- We evaluate our approach by considering a realistic Flight Management System (FMS) application on unicore platform.
- We conduct extensive simulations to evaluate energy-aware mapping techniques on multi-cores. We also demonstrate with our results that both mapping methods – "mixing of tasks" and "maintaining isolation" almost save same amount of energy.

## 1.5 Thesis Outline

The rest of the article is structured as follows: In Chapter. 2, we discuss the relevant literature with respect to this thesis. In Chapter. 3 we describe the system model and in Chapter. 4 we present a convex problem formulation of the energy minimization problem. In Chapter. 5 we propose an optimal analytical solution and a simple heuristic to fulfill the energy minimization objective. We present different energy efficient multi-core mapping techniques in Chapter. 6 and evaluate our model in Chapter. 7. Finally we conclude in Chapter. 8.

# Chapter 2

# Related Work

The real-time research community has contributed many techniques for handling interferences in mixed-criticality systems [32, 33, 2]. It is interesting to see that all the techniques proposed so far share a common concept - to have multiple system models on different criticality levels and to enforce asymmetric interference among those levels [1]. In particular, the worst-case-execution-time (WCET) of all tasks is measured at all criticality levels, with the one at a higher criticality level being more pessimistic. At runtime, whenever any task exceeds its execution time on a certain criticality level, only tasks with higher criticality levels are guaranteed thereafter. In other words, high criticality tasks are allowed to interfere with low criticality level tasks (e.g., terminating them). However, the other way is forbidden. Many conventional scheduling techniques (e.g., fixed-priority, EDF, TDMA) have been extended to the mixed-criticality context. In this chapter, we list few relevant works done in the field of mixed-criticality systems. First, we discuss the important works done on scheduling, followed by mapping mixed-criticality tasks on multi-core and energy minimization.

## 2.1 Scheduling

In the literature, we find many scheduling algorithms for executing tasks in a mixed-criticality system.

- The concept of mixed-criticality was introduced in 2007 by Vestal of Honeywell Aerospace [34]. He extended Audsley's fixed priority real-time scheduling theory to verify mixed-criticality systems, focusing on unicore. Vestal proved that traditional algorithms such as Rate Monotonic and Deadline Monotonic were not optimal for mixed-criticality systems. The proposed technique mainly used Response Time Analysis to ensure timing safety.
- Vestal's work was improved by Baruah et al., [35] in 2008 using response-time analysis for fixed-priority scheduling of a sporadic task model. They also

conducted a thorough study of system feasibility and prove that EDF fails when criticality levels are introduced in real-time systems even if there exists a feasible solution.

- S. Baruah et al., proposed an effective scheduling algorithm called Own Criticality Based Priority (OCBP) [36] that extends Audsley's priority-based real-time scheduling approach. OCBP algorithm selects a task and first assigns it with a current level priority. If the task is schedulable, then it is moved to next higher priority. If the task is not schedulable, then another feasible task is assigned with the current priority level. When no task is schedulable at current priority level or when all the tasks are assigned with priorities, the algorithm terminates. The authors show that OCBP is better in terms of speedup factor among all the fixed-job-priority algorithms for non-recurrent task set. They also demonstrate that OCBP is optimal in terms of speedup factor and significantly better in performance than conventional approaches.

- A dynamic scheduling algorithm called Criticality Based Earliest Deadline First (CBEDF) was proposed by T. Park [37] in 2011. CBEDF comprises of two sub-algorithms: one to locate empty slack in offline mode and another to schedule tasks during runtime. The algorithm maintains two task queues: one for safety-critical and another for non-safety-critical tasks. First, the critical tasks are scheduled offline using their WCETs. Next, the non-safety tasks are inserted into the available slacks. Finally, the scheduling is performed during runtime. The authors also proved that CBEDF dominates Baruah's OCBP.

- Pellizzoni et. al. [38] proposed a reservation-based approach, providing isolation guarantees in mixed-criticality systems. The approach is similar to Platform Based Design [39] which distinguishes functional and physical isolation to deal with data exchange and system resource sharing. Petters et. al. [40] also addressed issues in mixed-criticality systems and proposed temporal isolation based scheduling technique. However, the approach was not energy aware and also relies on over provisioning system resources.

- In 2012, Baruah et. al. proposed a simple and efficient algorithm called EDF-VD (Earliest Deadline First - Virtual Deadlines) for scheduling mixed-criticality tasks. The idea is to set earlier deadlines (virtual deadlines) for tasks on different criticality levels so that a higher-criticality task finishes its low criticality workload early and save time for its extra workload. The authors also show that the algorithm has a speedup factor of $\frac{4}{3}$ and can schedule $\frac{4}{3}$ times faster than any optimal clairvoyant algorithm can.

- Ekberg and Yi [41] used demand bound approach to analyze the constrained deadline tasks scheduled by the EDF-VD. The tasks on different criticality levels are set with the virtual deadlines depending on the demand bound of the tasks. Similar to EDF-VD, all low criticality tasks are abandoned when the system switches to overload mode.

- Easwaran improved Ekberg and Yi's method for dual-criticality systems by suggesting a new demand-based schedulability test. This test was an improved

version of Ekberg's model. Easwaran also introduced deadline tightening strategy for high criticality tasks in his improved test.

## 2.2 Multi-core Mapping

We find many real-time task scheduling algorithms for the multi-core platform in literature [42, 43, 44] out of which Partitioned [45, 30] and Global scheduling [29] algorithms are widely used. In global scheduling, a task may be executed on different processors (or processor cores) and in partitioned scheduling, the assignment of a task is fixed to a particular processor. Because of their simplicity and efficiency, partitioned scheduling algorithms are preferred than global scheduling. This even holds true for mixed-criticality systems as partitioned scheduling can provide isolation and reduce interference between tasks with different criticality levels.

- In 2009, Anderson et al., discussed mapping and scheduling of mixed-criticality tasks in the context of multi-processor platform [46]. The authors addressed five different criticality Level A (very high critical) to E (very low critical). The tasks were mapped on available processor cores using different techniques: Tasks with Level A are scheduled statically (cyclic executive); Level B tasks using partitioned preemptive EDF; Level C and D with global preemptive EDF; and global best-effort for Level E tasks.

- Li and Baruah combined EDF-VD with global scheduling algorithm fp-EDF [47] to schedule mixed-criticality tasks on multi-processor platform. Evaluations indicated that combination was inefficient for globally partitioned systems.

- Baruah.et al., also extended EDF-VD on the multi-core platform using partitioned mapping [24]. The mapping is done in two steps: First, all the high criticality tasks are mapped on different cores using First-Fit bin packing [26] technique. In the second stage, all low criticality tasks are mapped on available cores using the First-Fit method. Finally, EDF-VD is used to schedule tasks on each core. This approach showed better performance in terms of schedulability as compared to fp-EDF, and reduced the number of processor cores used for execution.

- A novel and a different approach for scheduling multi-criticality tasks on multi-cores was provided by Kritikakou et al., [48]. The aim was to decrease the interference of low criticality tasks due to shared memory and bus. The execution times of high criticality tasks are monitored until a point when further interference cannot be tolerated. In situation of high interference, the low criticality tasks are abandoned to guarantee the certification of high criticality tasks.

- Giannopoulou et al., [3] use partition and time-triggered approach to schedule mixed-criticality applications on resource sharing multi-core systems. In this approach, only tasks with same criticality level can access the multi-processor bus at any point of time. This helps to reduce interference and ensure safety

since lower criticality tasks are not allowed to execute simultaneously along with high criticality tasks.

- C. Gu .et al., [49] propose a new variant of multi-processor scheduling algorithm that is based on Ekberg and Yi's single core approach [41]. Task mapping is achieved in two steps: first all high criticality tasks are allocated on cores using Worst-Fit, followed by low criticality tasks using the First-Fit. Finally, Ekberg's algorithm is used to schedule tasks on each core. Since Worst-Fit tries to distribute the slack on available cores evenly, the high criticality tasks are circulated on available cores. However, the LO criticality tasks are loaded on the first available core, thus decreasing the interference on HI criticality tasks.

- Mixed-criticality scheduling upon varying-speed multi-processors was addressed by Z. Guo et al., in [50]. They construct a linear program (LP) based on scheduling conditions. The program implements processor-sharing approach and according to its solution, the mixed-criticality tasks are scheduled on multi-processors.

- G. Liu et al., present a novel method called MinLoad algorithm [51] for scheduling mixed-criticality parallel jobs. The idea is to decompose parallel tasks into sequential tasks and map them into different processors. The jobs on all the processors are executed in parallel, thus achieving intra-task parallelism.

- Recently, a fault-tolerant scheduling algorithm for mixed-critical applications on multi-processor platforms was presented by M. Bagheri and G. Jervan [52]. The authors suggest a framework to handle both computation and inter-task communication in NoC-based multi-processors. They also propose a mixed-criticality scheduling method that can ensure safety even in the presence of transient faults.

## 2.3 Energy minimization

Energy-efficient scheduling has been an active research topic in the past decade. There are many excellent techniques proposed such as DVFS for single and multi-processor systems. Most of them are for general real-time systems where all tasks are considered to be equally critical. However, there are few explorations done in terms of mixed-criticality system and energy minimization.

- An approach is presented in [53] to trade the deadline missing of low criticality tasks with the energy savings on multi-cores while deadlines of high criticality tasks are always guaranteed. The authors propose a power-aware scheduling algorithm called LPDPM-MC for mixed-criticality systems, based on LPDPM algorithm.

- M. Volp et al., [54] discuss and demonstrate how energy handling can lead to failures in ensuring mixed-criticality safety. They consider a real-time scenario

and show that handling of energy can lead to safety violations and can be avoided only when energy becomes equal resource as time. They also discuss the energy threats and re-evaluate scheduling techniques such as OCBP for energy budget.

- A method to minimize energy for unicore mixed-criticality systems is presented in [22], while respecting system reliability and timing requirements. Rather than treating energy as an optimization goal, the situation when energy supply is going down is handled. Focus on the energy utilization in high criticality tasks is advocated while allowing deadline misses of low criticality tasks.

- Huang et al., proposed an energy efficient algorithm [23] to minimize the energy consumption for the nominal scenario where tasks do not overrun. An optimal solution for the unicore system is proposed in this regard at the cost of increased energy consumptions for other system scenarios.

As described above, most of the proposals discuss about scheduling mixed-criticality tasks on unicore and multi-cores. While energy minimization is achieved by [53, 22, 23], either they concentrate energy reduction in only one of the criticality modes or trade-off energy for timing guarantee. Of these works, Huang's work [23] is the closest to our work where a solution for saving dynamic energy in nominal mode is proposed.

# Chapter 3

# System Model

In this chapter, we first define the mixed-criticality system and power model used in our work. In Section 3.1, we describe our task model that is based on traditional mixed-criticality assumptions in literature. We then discuss the power model adopted in our work in Section 3.2. In Section 3.3, a mixed-criticality scheduling technique (EDF-VD) employed to fulfill the requirements of certifiability is discussed with an example. Finally, we employ energy minimization strategy (DVFS) to EDF-VD, which is described in Section 3.4.

## 3.1 Mixed-Criticality Task Model

We consider a mixed-criticality task set $\tau$ containing $n$ independent sporadic tasks $\tau_1, \tau_2, ....\tau_n$ to be scheduled on $m$ identical preemptive processors. The relative deadline and minimum inter-arrival time of task $\tau_i$ are denoted as $D_i$ and $T_i$, respectively. Tasks have implicit deadlines, i.e. $\forall\tau_i$, $D_i = T_i$. A task can release an infinite number of jobs separated by its minimum inter-arrival time. We consider two-criticality levels in our model: each task $\tau_i$ is associated with a safety criticality level $\chi_i$, being either High (HI) or Low (LO). To ensure safety, HI criticality tasks are more conservative than LO criticality tasks. As a result, WCETs of HI criticality tasks are more pessimistic than that of LO criticality tasks. To further improve resource efficiency, the state-of-the-art assumption [1] is to measure task WCETs on all criticality levels

1. For any HI criticality task $\tau_i$, it has a LO criticality WCET $C_i$(LO) and a more pessimistic HI criticality WCET $C_i$(HI).

2. A LO criticality task $\tau_i$ is not allowed to overrun its LO criticality WCET $C_i$(LO).

Based on the above model, resource efficiency is achieved through dynamic resource management:

- The methodology is to start the system execution with LO mode where all tasks execute according to their LO criticality WCETs.
- If any HI criticality task overruns its LO criticality WCET, then the system switches to HI mode and all LO criticality tasks are dropped to guarantee HI criticality WCETs for HI criticality tasks.

However, the system can switch back to LO criticality mode at any time when there are no pending tasks available for execution. We do not discuss this scenario since it is beyond the scope of this work. Notice that, on a multi-core platform, the mixed-criticality mode switch can be performed locally by assuming partitioned scheduling [30] or globally by assuming global scheduling [29].

For notational convenience, we define $U_{\chi_1}^{\chi_2}$ for $\chi_1$ , $\chi_2 \in \{\text{LO}, \text{HI}\}$ as follows:

$$U_{\chi_1}^{\chi_2} = \sum_{\tau_i \in \tau \wedge \chi_i = \chi_1} \frac{C_i(\chi_2)}{T_i}. \tag{3.1}$$

$U_{\chi_1}^{\chi_2}$ is defined as the total utilization of all $\chi_1$ criticality tasks with their $\chi_2$ criticality WCETs, on all cores. For instance, $U_{\text{HI}}^{\text{LO}}$ denotes the utilization of HI criticality tasks with their LO criticality WCETs (LO mode behavior of HI criticality tasks). We further define $\tau_\chi$ as the $\chi$ criticality tasks present in task set $\tau$ where $\chi \in \{\text{LO} , \text{HI}\}$.

**Other notations:** For ease of presentation, we use $[\![a]\!]^c$ to represent $\min(a, c)$ and $[\![a]\!]_c$ to represent $\max(a, c)$.

## 3.2 Power Model and DVFS

In this paper, we adopt a popular power model presented in [55, 56]. Assuming a homogeneous multiprocessor platform, the total power consumption of any processor is formulated as

$$P(f) = P_s + P_d = P_s + \beta \cdot f^\alpha,$$

where $P(f)$ is the total power consumed and $P_s$ stands for the static power consumption due to leakage current. $f$ is the frequency at which the processor executes tasks and $\beta \cdot f^\alpha$ represents the dynamic power consumption ($P_d$) caused by switching activities, where $\alpha$ and $\beta$ are circuit dependent positive constants. An assumption is made in [57, 58] that $\alpha \geq 2$. Hence dynamic power of the system is a convex increasing function. In order to minimize dynamic energy, the working frequency can be decreased by means of DVFS. However, with reduced frequency the leakage energy is increasing as it takes longer to finish a job. Thus, there is a critical frequency $f_{crit}$ below which energy-wise it is not beneficial to reduce[1]. For any job

[1]More details on $f_{crit}$ can be found in [59]

with workload of $nc$ clock cycles, [59, 56] shows that the critical frequency can be obtained as follows,

$$\begin{aligned} &\frac{d(\frac{nc}{f} \cdot P_s + \frac{nc}{f} \cdot \beta \cdot f^{\alpha})}{df} = 0 \\ \Leftrightarrow &f_{crit} = \sqrt[\alpha]{\frac{P_s}{\beta \cdot (\alpha - 1)}}. \end{aligned} \tag{3.2}$$

As a result, we assume in this paper that the hardware platform is DVFS enabled and can execute with any frequency between $f_{\min}$ and $f_{\max}$, where $f_{\min} \geq f_{crit}$. To simplify presentation, we assume $f_{\max}$ is normalized to 1 and $f_b$ is the frequency on which task WCETs are measured without employing DVFS such that $f_{\min} \leq f_b \leq f_{\max}$. Notice that applying DVFS strategy changes actual WCETs of tasks: a task's $\chi$ criticality WCET becomes $\frac{C_i(\chi) f_b}{f}$ while running at frequency $f$.

In our system model, we consider only dynamic and static power for minimization. However, there is power consumption due to shared components such as processor caches, shared memory and buses in multi-core platform which we do not count as it is not in the scope of our work. Hence, the total power consumption of a multi-core is obtained by summing up the total power consumption of individual cores. To save static power, we switch the processor core to idle/sleep mode when no tasks are executed. For simplicity, we assume that sleep mode power is zero[2] and do not consider the active to idle (and vice-versa) mode switching time overheads.[3]

## 3.3 Mixed-Criticality Scheduling

For mixed-criticality multi-core scheduling, either global scheduling [29] or partitioned scheduling [30] can be applied. In this paper we focus on the latter. Since mixed-criticality scheduling is strongly NP-hard even for simple task models on a uniprocessor [30], we particularly study the integration of energy decisions into a well-known approach – the partitioned EDF-VD scheduling [24].

For partitioned EDF-VD, HI criticality tasks are first mapped to all processors followed by mapping of LO criticality tasks. During this process, First-Fit bin packing is used for mapping while system utilization factors on each core are set to admit feasible local schedules [24]. After mapping tasks onto processors, scheduling on each processor follows EDF-VD [32]. In EDF-VD, the deadlines of all HI criticality tasks are shortened by a multiplication factor $x$ ($0 \leq x \leq 1$) in LO mode, thus prioritizing their executions and leaving enough time until their actual deadlines to accommodate extra executions (overrun). Intuitively, a smaller $x$ causes increased system utilization in LO mode but decreasing system utilization in HI mode as more jobs are finished in LO mode. Consequently, it will affect system schedulability in both LO and HI modes. Let us relax our notation and assume task set $\tau$ is run on a unicore. The following results from [32] limits the feasible range of $x$.

[2]The sleep mode power is not zero in practice as the processor should be ON, and be able transit to active mode again. However, sleep mode power is constant independent of execution frequency.

[3]The switching time overheads can be included in WCETs of the tasks.

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

$$\begin{aligned} 0 < \frac{U_{\mathrm{HI}}^{\mathrm{LO}}}{1 - U_{\mathrm{LO}}^{\mathrm{LO}}} \leq x \\ x \leq \left[\!\left[ \frac{1 - U_{\mathrm{HI}}^{\mathrm{HI}}}{U_{\mathrm{LO}}^{\mathrm{LO}}} \right]\!\right]^{1}. \end{aligned} \tag{3.3}$$

*Proof.* This follows from (3) and Theorem 2 in [32]. □

Once the deadlines of HI criticality tasks are shortened, then scheduling is performed as follows:

1. The system starts with LO mode and all LO criticality tasks are scheduled under EDF, with their actual deadlines equal to their periods ($T_i$).
2. All HI criticality tasks are executed with their modified deadlines ($\hat{T}_i = x \cdot T_i$).
3. If any HI criticality task fails to execute within its nominal WCET, then the system immediately enters into HI criticality mode and all HI criticality tasks are executed with their actual (unmodified) deadlines. All LO criticality tasks are suspended henceforth.

For better understanding of the EDF-VD test, let us consider the same task set mentioned in Example 1 and schedule it using EDF-VD on a unicore.

**Example 2.** As a first step, let us calculate the task utilization in both LO and HI criticality modes. Then, we find the feasible range of $x$ to shorten the deadlines of HI criticality tasks, and finally schedule the task set.

*Step 1:* Using (3.1), we can calculate task utilization in different modes as:

$$\begin{aligned} U_{\mathrm{LO}}^{\mathrm{LO}} &= \sum_{\tau_i \in \tau \wedge \chi_i = \mathrm{LO}} \frac{C_i(\mathrm{LO})}{T_i} = \frac{2}{4} = \frac{1}{2} \\ U_{\mathrm{HI}}^{\mathrm{LO}} &= \sum_{\tau_i \in \tau \wedge \chi_i = \mathrm{HI}} \frac{C_i(\mathrm{LO})}{T_i} = \frac{1}{6} \\ U_{\mathrm{HI}}^{\mathrm{HI}} &= \sum_{\tau_i \in \tau \wedge \chi_i = \mathrm{HI}} \frac{C_i(\mathrm{HI})}{T_i} = \frac{5}{6} \end{aligned}$$

*Step 2:* Using (3.3), we can calculate the bound for $x$:

$$\begin{aligned} 0 \;<\; \frac{\frac{1}{6}}{1 - \frac{1}{2}} \;\leq\; x \;\leq\; \frac{1 - \frac{5}{6}}{\frac{1}{2}} \;\leq\; 1 \\ \iff x = \frac{1}{3} \end{aligned}$$

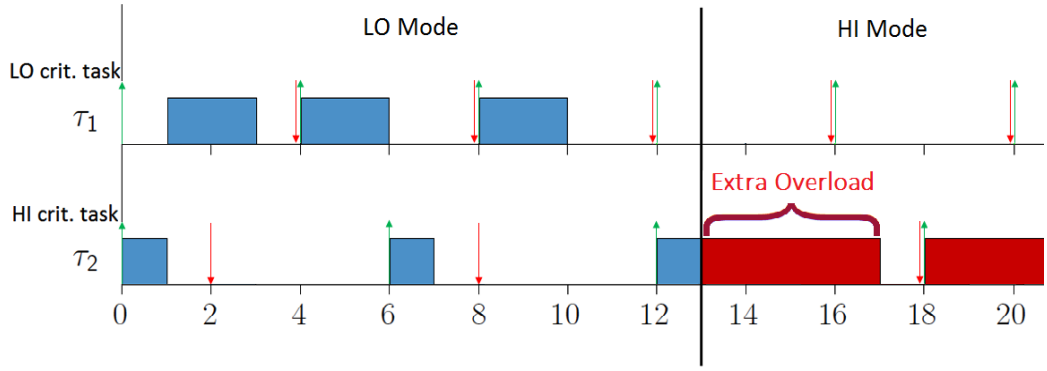


**Figure 3.1:** Mixed-criticality task set scheduled under EDF-VD

**Note:** We obtained $x = \frac{1}{3} =$ constant. However, when $x$ is bounded, fixing $x$ with the least possible value ensures more safety as HI criticality tasks get more priority because of shortened deadlines.

*Step 3:* The deadlines of all HI criticality tasks are shortened by $x$:

$$\forall \tau_i \in \chi_i = \text{HI}, \quad \hat{T}_i = x \cdot T_i$$
$$\iff \hat{T}_2 = \left(\frac{1}{3}\right) \cdot 6 = 2$$

*Step 4:* The task is scheduled under EDF as shown in Figure 3.1. Assuming that the $3^{rd}$ job of $\tau_2$ overruns its nominal WCET, we observe that $\tau_2$ meets its deadline, thus guaranteeing safety.

## 3.4 Scheduling and Energy Minimization

Now, let us apply energy minimization technique DVFS on EDF-VD so that execution frequency of tasks can be changed to ensure safety as well as conserve energy. Essentially, Theorem 1 states that there is a lower bound on $x$, below which the LO mode will not be schedulable. Similarly, an upper bound exists, beyond which the HI mode will not be schedulable. However, notice that with DVFS, the system utilization factors are modified, and the test as specified in Theorem 1 needs to be performed with the scaled utilization factors. Hence with Theorem 1, we can get a new schedulability condition for the EDF-VD test, by scaling task utilizations with execution frequency.

**Corollary 1.** For a task set $\tau$ scheduled by EDF-VD on a unicore. Assume that with DVFS, task $\tau_i$ has frequency $f_i^{\chi}$ in $\chi$ system mode, then the feasible range of $x$ is,

$$0 < \frac{\widetilde{U}_{\text{HI}}^{\text{LO}}}{1 - \widetilde{U}_{\text{LO}}^{\text{LO}}} \leq x \leq \left[\!\left[ \frac{1 - \widetilde{U}_{\text{HI}}^{\text{HI}}}{\widetilde{U}_{\text{LO}}^{\text{LO}}} \right]\!\right]^1 \tag{3.4}$$

where

$$\widetilde{U}_{\chi_1}^{\chi_2} = \sum_{\tau_i \in \tau_{\chi_1}} \frac{\widetilde{C}_i(\chi_2)}{T_i},$$

$$\widetilde{C}_i(\mathrm{LO}) = \frac{C_i(\mathrm{LO}) f_b}{f_i^{\mathrm{LO}}}, \forall \tau_i \in \tau$$

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

Notice that, according to Corollary 1, with increasing $f_i^{\chi}$, $\forall \tau_i$, 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 where $f_i^{\chi} = f_{\max}$, $\forall \tau_i$. We denote the lower and upper bounds in this case as $\hat{x}_{\mathrm{LB}}, \hat{x}_{\mathrm{UB}} \in [0, 1]$, respectively.

# Chapter 4

# Motivational Example and Problem Definition

In this chapter, we provide a motivational example and describe the problem definition. In Section 4.1, with a concrete example we explain the need for energy minimization and analyze the factors that affect the energy consumption in unicore and multi-core. We then present a convex problem formulation of the energy minimization problem in Section 4.2.

## 4.1 Motivational Example

| $\tau$ | $\chi_i$ | $T_i$ | $C_i$(LO) | $C_i$(HI) |
|---|---|---|---|---|
| $\tau_1$ | HI | 40 | 4 | 12 |
| $\tau_2$ | HI | 75 | 6 | 18 |
| $\tau_3$ | HI | 40 | 3 | 9 |
| $\tau_4$ | LO | 100 | 6 | 6 |
| $\tau_5$ | LO | 80 | 5 | 5 |

**Table 4.1:** A mixed-criticality taskset

Minimizing only LO mode or only HI mode energy is not appropriate as the system might switch between modes at any time. To indicate the importance of energy minimization in a particular mode, we define weight factors $w_{\text{LO}}$ and $w_{\text{HI}}$ for LO and HI modes respectively, where $w_{\text{LO}}, w_{\text{HI}} \in [0, 1]$, $w_{\text{LO}} = 1 - w_{\text{HI}}$. The weight factors are the percentages that the system operates in LO and HI modes. As discussed in Chapter 1, we also include static energy minimization in our problem. Let us consider an example to show the impact of weight factors and static energy in minimizing energy on mixed-criticality systems. First, we consider a unicore platform to analyze the problem and explain the need for energy minimization in

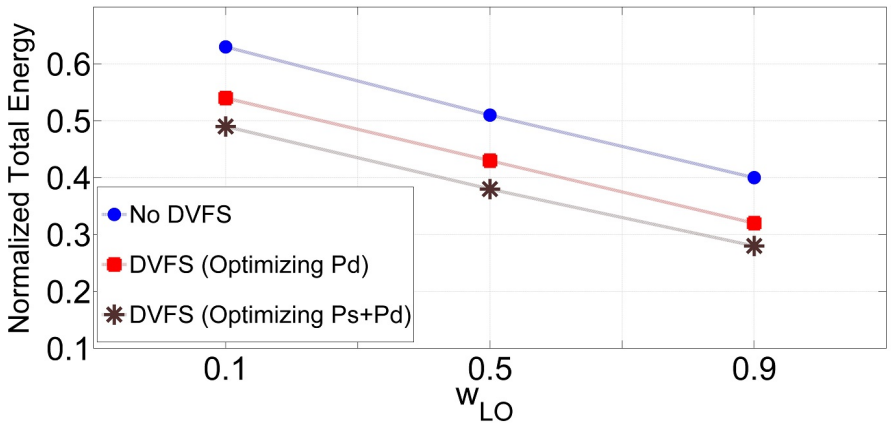


**Figure 4.1:** Energy dissipation with different mode weights

both LO and HI modes. Then, we map the tasks on multi-core to evaluate the effects of First-Fit and Worst-Fit mapping techniques in minimizing energy on multi-cores.

**Example 3.** The considered tasks are shown in Table 4.1. The task set is schedulable on a uniprocessor according to the EDF-VD test with base frequency $f_b = 0.9$. The processor is capable of DVFS in the range $[f_{min}, f_{max}] = [0.5, 1]$ and we consider $\alpha = 2$ and $\beta = 0.8$ [58, 59]. Assuming the normalized static power $P_s = 0.2$ [58], we calculate the normalized total energy dissipation of the system with different mode weights $w_{\text{LO}}$ and $w_{\text{HI}}$.

### 4.1.1 Unicore

We consider 3 strategies to demonstrate the impact of weight factors and static energy on energy minimization on a unicore:

- **Strategy A:** Without DVFS, where all tasks are executed with base frequency $f_b$.
- **Strategy B:** With DVFS, where only dynamic energy is optimized (set $P_s = 0$ during optimization). The optimal frequencies are calculated using the algorithm that we present in Section. 5.8 and finally total energy is calculated as the weighted sum of static and dynamic energy in both the modes.
- **Strategy C:** With DVFS, considering both static and dynamic energy for optimization. The optimal frequencies are calculated using the algorithm which we present in Section. 5.8 and finally total energy is calculated as the weighted sum of static and dynamic energy in both the modes.

In strategies (B) and (C), we consider energy saving in both LO and HI mode modeled with the weighted sum of LO and HI mode energy. The expected energy consumptions are listed in Figure 4.1.

Indeed, it is evident from the results we obtained that better energy saving can be achieved by employing DVFS (Strategies B and C as compared to Strategy A). To

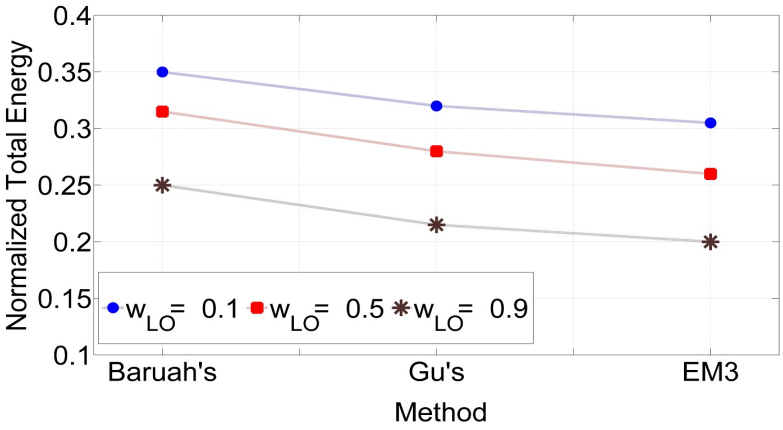


**Figure 4.2:** Comparison of normalized energy consumption with different mapping techniques on multi-core

demonstrate the importance of weight factors, let us consider the task utilization, expected energy outcome and the weight factors into account at first. Note that in the considered task set, $U_{\mathrm{HI}}^{\mathrm{HI}} > \left(U_{\mathrm{LO}}^{\mathrm{LO}} + U_{\mathrm{HI}}^{\mathrm{LO}}\right)$. Hence the system is more active in HI mode as compared to LO mode, thus increasing the dynamic power in HI mode. With this notion, if $w_{\mathrm{LO}} = w_{\mathrm{HI}}$, one can easily predict that more preference has to be given for minimizing HI mode energy. However, if $w_{\mathrm{LO}} \gg w_{\mathrm{HI}}$, giving higher priority to minimize HI mode energy is not optimal as the system operates in LO mode most of the times. This is clear from the plot that energy consumption is less when $w_{\mathrm{LO}} \geq w_{\mathrm{HI}}$. Thus, weight factors provide the freedom to assign priority to energy saving in a particular mode.

However in Strategy C, static energy is also considered in the energy minimization problem. We observe in the plot that Strategy C performs better in energy minimization as compared to Strategy B. In Strategy B, the energy objective is unaware of the static energy, thus aiming to find a solution that depends only on dynamic energy consumption. On the contrary, in Strategy C, both the static and dynamic energy together are minimized by applying DVFS. Thus, the overall energy minimization becomes a complex problem that not only depends on the utilization of tasks but also the mode weights and static energy consumption.

### 4.1.2 Multi-core

To analyze how mapping policy affects the energy consumption on multiple cores, we first consider two existing mapping techniques presented by S. Baruah et al. [24] (Baruah's method hereafter) and C. Gu et al.[49] (Gu's method hereafter). Then, we present a new energy efficient mapping technique namely EM3. In Baruah's method, both LO and HI criticality tasks are mapped on cores using the First Fit (FF) mapping technique which favors schedulability. In Gu's method, all HI criticality tasks are mapped using the Worst Fit (WF) and all LO criticality tasks are mapped using the FF technique. In EM3 method, both HI and LO criticality tasks are mapped using WF. These three methods are explained in detail in Chapter 6.

Let us consider the same task set listed in Table 4.1 on multi-core. Tasks are mapped using Baruah's method, Gu's method, and EM3 method separately. Finally, DVFS is applied on all cores (described in Chapter 6) in all the above methods. The normalized total energy of the system for different mode mapping methods and weights is shown in Figure 4.2. We observe in the plot that for all combinations of $w_{\mathrm{LO}}$ and $w_{\mathrm{HI}}$, Baruah's mapping method has maximum energy consumption, followed by Gu's method. The consumption is the least in EM3 among all the methods. Though Baruah's mapping method aims at the usage of minimal cores and schedulability, it does not favors energy minimization. The reason is that FF does not balances the task utilization between the available cores but load the first available core. Thus, increasing utilization in a core restrains the frequency reduction, therefore increasing the energy exponentially. Since Gu's method employs WF for mapping HI criticality tasks, the tasks are evenly distributed on all available cores, thus facilitating frequency reduction. EM3 is efficient in saving energy as WF technique is used for mapping both LO and HI criticality tasks. Thus, mapping technique plays an important role in saving energy on a multi-core.

With a motivational example, we have demonstrated why weight factors, static energy and mapping techniques should be considered in energy minimization problem. With these insights, we present the problem formulation for energy minimization objective.

## 4.2 Problem Formulation

We divide the multi-core mixed-criticality energy minimization problem into two objectives. First, we have to find an energy-aware task-to-processor mapping. After this, energy is minimized on individual cores assuming the EDF-VD scheduling. The first step, in fact, depends on the second one, since we should find mappings that best explore the energy saving potential of the unicore DVFS techniques. For this reason, we first solve DVFS scheduling under EDF-VD for unicore and then we extend the work for multi-cores. However, notice that the energy minimization problem on unicore has already been studied in [23] where only one system scenario (LO mode) is considered and only dynamic energy consumption is reduced. In this paper, we remove the limitations in [23] by considering energy minimization in both LO and HI modes and by including static energy while optimizing total energy. We also extend the work to multi-cores.

### 4.2.1 Unicore problem

Let us consider first, the energy minimization problem on a unicore. We use $\tau$ to denote the task set on one processor. To apply DVFS, the essential problem is to assign each task with a frequency to run in each criticality mode, such that energy is minimized while mixed-criticality real-time guarantees are satisfied. Let us denote the frequency for task $\tau_i \in \tau$ in mode $\chi$ as $f_i^{\chi}$, where $\chi \in \{\mathrm{LO}, \mathrm{HI}\}$. To consider energy minimization for both system scenarios, we have to define a proper energy objective taking into account energy consumptions in both LO and HI modes. To

this end, we express the importance of minimizing LO mode energy with a weight factor $w_{\mathrm{LO}}$, similarly $w_{\mathrm{HI}}$ for HI mode, where, $w_{\mathrm{LO}}, w_{\mathrm{HI}} \in [0,1]$, $w_{\mathrm{LO}} = 1 - w_{\mathrm{HI}}$. Notice that we do not pose any restriction on how to obtain $w_{\mathrm{LO}}$ and $w_{\mathrm{HI}}$. In practice, one could set them as the percentage that the system operates in LO and HI modes. Indeed, they can be set just as simple relative weight factors that express the relative importance to minimize energy in both modes. The formulation is general in the sense that if $w_{\mathrm{LO}} = 1 \wedge w_{\mathrm{HI}} = 0$, we minimize the LO mode energy, and if $w_{\mathrm{LO}} = 0.5 \wedge w_{\mathrm{HI}} = 0.5$, we then minimize the average energy consumption in both modes. Now for one hyper-period in LO mode, we represent the normalized total energy consumption in a hyperperiod $\Pi_{\tau_i \in \tau} T_i$ (the actual energy consumption divided by the hyper-period length) as:

$$E_{\mathrm{LO}} = w_{\mathrm{LO}} \cdot \sum_{\tau_i \in \tau} \frac{C_i(\mathrm{LO})}{T_i} \cdot \frac{f_b}{f_i^{\mathrm{LO}}} \cdot (P_s + \beta \cdot (f_i^{\mathrm{LO}})^{\alpha}). \tag{4.1}$$

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

$$E_{\mathrm{HI}} = w_{\mathrm{HI}} \cdot \sum_{\tau_i \in \tau_{\mathrm{HI}}} \frac{C_i(\mathrm{HI})}{T_i} \cdot \frac{f_b}{f_i^{\mathrm{HI}}} \cdot (P_s + \beta \cdot (f_i^{\mathrm{HI}})^{\alpha}). \tag{4.2}$$

Thus, our goal is to minimize the system energy across both criticality modes:

$$E = E_{\mathrm{LO}} + E_{\mathrm{HI}} \tag{4.3}$$

Energy should be minimized while satisfying the mixed-criticality real-time requirements. Now, to apply DVFS to save energy, we need to ensure that (4.3) is minimized while (3.4) is satisfied. With reformatting and constraint transformation, we can formulate our unicore energy minimization as a convex program.

**Theorem 2.** The unicore energy minimization problem can be formulated as a convex program as follows,

$$\textbf{min} \qquad E = E_{\mathrm{LO}} + E_{\mathrm{HI}} \tag{4.4}$$

$$\textbf{s.t.} \qquad \frac{\widetilde{U}_{\mathrm{HI}}^{\mathrm{LO}}}{x} + \widetilde{U}_{\mathrm{LO}}^{\mathrm{LO}} \leq 1 \tag{4.5}$$

$$x\widetilde{U}_{\mathrm{LO}}^{\mathrm{LO}} + \widetilde{U}_{\mathrm{HI}}^{\mathrm{HI}} \leq 1 \tag{4.6}$$

$$x \in [\hat{x}_{\mathrm{LB}}\,,\, \hat{x}_{\mathrm{UB}}] \tag{4.7}$$

$$\forall \tau_i\,,\, \forall \chi\,,\, f_i^{\chi} \in [f_{\min}\,,\, f_{\max}] \tag{4.8}$$

According to our problem formulation and (4.1), in order to minimize energy in LO mode we can reduce $f_i^{\mathrm{LO}}$. However, this increases system utilization in LO mode and decreases free slack available until any HI criticality task's actual deadline when mode switch happens. Thus, to maintain the schedulability in HI mode, $f_i^{\mathrm{HI}}$ has to be increased thereby jeopardizing HI mode energy consumption. Hence, there is always a trade-off between energy saving in LO and HI modes, where the intension is to minimize the weighted sum of energy in both modes. Moreover, though the convex formulation suggests practical algorithms [60] can be applied to solve the optimization problem, we will theoretically investigate the energy dependency across different modes in the later chapters.

### 4.2.2 Energy-Aware Mapping

Assuming partitioned scheduling in this paper, the next problem is to find energy efficient mapping of tasks onto multi-core processors, such that by applying unicore DVFS locally on each processor, the total energy consumed by all tasks on all cores is minimized. Multi-cores are more energy efficient than equivalent unicore as multiple cores can execute the same tasks simultaneously but with lower execution frequencies. The leakage power can be adjusted by suitably selecting the number of active cores. Leakage power is a linear function of execution frequency and dynamic power is a cubic function [61]. Thus, the ratio of static power to the dynamic power increases as frequency decreases. Therefore, if the number of active cores is increased, then leakage power may start dominating. This energy trade-off makes the problem different than in unicore where only one core is available. Hence, it is necessary to choose an optimal number of cores and implement a good mapping procedure where both dynamic and static power is reduced and also safety is guaranteed.

# Chapter 5

# Unicore Optimal Solution

We investigate in this chapter, an optimal solution for the mixed-criticality energy minimization problem on a unicore. Since we have a convex formulation (4.4)−(4.8), we first apply the Karush-Kuhn-Tucker (KKT) optimality conditions [62] to find optimal frequencies $f_i^\chi$. To perform this, we first explain in Section 5.1, what are KKT conditions. Then, we apply KKT conditions for our problem which is described in Section 5.2. One can solve our energy minimization problem using KKT conditions but the complete solution by KKT incurs great computation complexity. Hence, we do not use KKT conditions to find the optimal solution but only to demonstrate the complexity of the problem. However, we use KKT conditions to derive a reduced search space for our problem which is explained in Section 5.3 and Section 5.4 in detail. This allows us to develop an optimal solution and an efficient heuristic that are explained in Section 5.7 and Section 5.8 respectively.

## 5.1 KKT Conditions

Karush-Kuhn-Tucker (KKT) conditions are the first order conditions for a solution to be optimal in non-linear programming. KKT conditions were named after Harold W. Kuhn, and Albert W. Tucker who published their work in 1951. The KKT approach allows both equality and inequality constraints in a problem unlike the method of Lagrange multipliers that is subject to only equality constraints [63].

**KKT conditions:** Let us consider the general optimization problem of the form

$$\begin{aligned} &\textbf{minimize } f(x) \\ \textbf{s.t. } &h_i(x) = 0 \ \ \forall i = 1, ..., n \\ &g_j(x) \leq 0 \ \ \forall j = 1, ..., m \end{aligned}$$

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

**Theorem 3.** According to the KKT conditions [62], when the objective function is convex and the constraints are affine, then a global minimal solution $x^*$ exists when

$$\begin{aligned} \nabla_x f(x^*) + \sum_{i=1}^{n} \lambda_i \nabla_x h_i(x^*) + \sum_{j=1}^{m} \mu_j \nabla_x g_j(x^*) = 0 \\ \textbf{s.t. } h_i(x*) = 0 \quad \forall i = 1, ...n \\ g_j(x*) \leq 0 \quad \forall j = 1, ...m \\ \mu_j g_j(x*) = 0 \quad \forall j = 1, ...m \\ \mu_j \geq 0 \quad \forall j = 1, ...m \end{aligned}$$

## 5.2 Optimal Solution Using KKT Conditions

Since the cost function (4.4) in our energy minimization objective is convex, we can apply Theorem 3 to find the optimal solution. Hence our objective with variables $f_i^{\chi}$ , $\forall \tau_i, \forall \chi$ and $x$ and constraints (4.5) - (4.8) can be written in terms of KKT conditions as

$$\begin{aligned} &\nabla_{f,x} \Bigg( (E_{\mathrm{LO}} + E_{\mathrm{HI}}) + \mu_1 \left( \frac{\widetilde{U}_{\mathrm{HI}}^{\mathrm{LO}}}{x} + \widetilde{U}_{\mathrm{LO}}^{\mathrm{LO}} - 1 \right) \\ &+ \mu_2 \left( x \widetilde{U}_{\mathrm{LO}}^{\mathrm{LO}} + \widetilde{U}_{\mathrm{HI}}^{\mathrm{HI}} - 1 \right) - \mu_3 \left( x - \hat{x}_{\mathrm{LB}} \right) + \mu_4 \left( x - \hat{x}_{\mathrm{UB}} \right) \\ &- \sum_{\tau_i \in \tau_{\mathrm{LO}}} \left( \check{\mu}_i^{\mathrm{LO}} \left( f_i^{\mathrm{LO}} - f_{min} \right) - \hat{\mu}_i^{\mathrm{LO}} \left( f_i^{\mathrm{LO}} - f_{max} \right) \right) \\ &- \sum_{\tau_i \in \tau_{\mathrm{HI}}} \left( \check{\mu}_i^{HL} \left( f_i^{\mathrm{LO}} - f_{min} \right) - \hat{\mu}_i^{HL} \left( f_i^{\mathrm{LO}} - f_{max} \right) \right) \\ &- \sum_{\tau_i \in \tau_{\mathrm{HI}}} \left( \check{\mu}_i^{\mathrm{HI}} \left( f_i^{\mathrm{HI}} - f_{min} \right) - \hat{\mu}_i^{\mathrm{HI}} \left( f_i^{\mathrm{HI}} - f_{max} \right) \right) \Bigg) = 0 \end{aligned} \tag{5.1}$$

$$\begin{aligned} \textbf{s.t.} \quad & \mu_j \geq 0 \quad \forall j = 1, ..., 4 \,; \\ & \mu_1 \left( \frac{\widetilde{U}_{\mathrm{HI}}^{\mathrm{LO}}}{x} + \widetilde{U}_{\mathrm{LO}}^{\mathrm{LO}} - 1 \right) = 0 \,;\, \mu_2 \left( x \widetilde{U}_{\mathrm{LO}}^{\mathrm{LO}} + \widetilde{U}_{\mathrm{HI}}^{\mathrm{HI}} - 1 \right) = 0 \,; \\ & \mu_3 \left( x - \hat{x}_{\mathrm{LB}} \right) = 0 \,;\, \mu_4 \left( x - \hat{x}_{\mathrm{UB}} \right) = 0 \,; && \text{(C1)} \\ \forall \tau_i \,, & \sum_{\tau_i \in \tau_{\mathrm{LO}}} \left( \check{\mu}_i^{\mathrm{LO}} \left( f_i^{\mathrm{LO}} - f_{min} \right) + \hat{\mu}_i^{\mathrm{LO}} \left( f_i^{\mathrm{LO}} - f_{max} \right) \right) = 0 \,; && \text{(C2)} \\ & \sum_{\tau_i \in \tau_{\mathrm{HI}}} \left( \check{\mu}_i^{\mathrm{HI}} \left( f_i^{\mathrm{HI}} - f_{min} \right) + \hat{\mu}_i^{\mathrm{HI}} \left( f_i^{\mathrm{HI}} - f_{max} \right) \right) = 0 \,; && \text{(C3)} \\ & \sum_{\tau_i \in \tau_{\mathrm{HI}}} \left( \check{\mu}_i^{HL} \left( f_i^{\mathrm{LO}} - f_{min} \right) + \hat{\mu}_i^{HL} \left( f_i^{\mathrm{LO}} - f_{max} \right) \right) = 0 && \text{(C4)} \end{aligned}$$

where $E_{\mathrm{LO}}$ and $E_{\mathrm{HI}}$ are represented by (4.1) and (4.2) respectively.

To obtain the solution, in each of the complementary slackness equations $\mu_i g_i(x) = 0$, at least one of the two factors $\mu_i$ and/or $g_i(x)$ must be 0. With $m$ such conditions, there would potentially be $2^m$ possible cases (such as $\mu_i = 0$ or $g_i(x) = 0$) to consider. Thus, if there are $|\tau_{\text{LO}}|$ LO-criticality tasks and $|\tau_{\text{HI}}|$ HI-criticality tasks in the system, then we have 4 constraints to be considered from (C1), $2|\tau_{\text{LO}}|$ constraints from (C2), $2|\tau_{\text{HI}}|$ from (C3) and $2|\tau_{\text{HI}}|$ constraints from (C4) for solving (5.1). This would lead to $2^{2|\tau_{\text{LO}}|+4|\tau_{\text{HI}}|+4}$ possible cases in order to find optimal solution. This leads to exponential complexity which is impractical to solve. Hence, we proceed towards the in-depth analysis of the energy minimization problem and optimality conditions so that search space can be reduced. To simplify the problem, first we use KKT conditions to find optimality condition in $f_i^{\chi}$. We then investigate if the simplified objective can be efficiently solved using KKT conditions.

## 5.3 Optimality Condition in $f_i^{\chi}$

Considering only LO mode energy and only dynamic energy consumption, Huang, et al., proved that the expected energy consumption is minimum when all tasks of the same criticality level share the same frequency in each mode [23]. Now, we generalize this result to consider in addition static energy consumption and energy in both HI and LO modes. We proceed to prove this using the KKT optimality conditions.

**Theorem 4.** For the unicore mixed-criticality energy minimization problem as specified in Theorem 2, in an optimal solution, all tasks of same criticality level share the same frequency in each mode, i.e.,

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

*Proof.* At 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. Similarly, the same can also be proven for LO mode energy as the energy objective for LO and HI mode is similar.

Considering only two HI criticality tasks $\tau_i$ and $\tau_j$ in the system, we prove using KKT conditions that they share same execution frequency. By induction, it is evident that the result should also hold for '$n$' tasks in the system.

We denote HI mode utilization of task $\tau_i$ and $\tau_j$ as $u_i$ and $u_j$ respectively such that $u_i = \frac{C_i(\text{HI})}{T_i}$ and $u_j = \frac{C_j(\text{HI})}{T_j}$.

Hence, from (4.3), we can write HI mode energy dissipation as

$$\begin{aligned} E_{\text{HI}} = w_{\text{HI}} \left( u_i \frac{f_b}{f_i^{\text{HI}}} \left( P_s + \beta (f_i^{\text{HI}})^{\alpha} \right) \right) \\ + w_{\text{HI}} \left( u_j \frac{f_b}{f_j^{\text{HI}}} \left( P_s + \beta (f_j^{\text{HI}})^{\alpha} \right) \right) \end{aligned} \tag{5.2}$$

For the lowest energy solution, (5.2) should be minimized. Let $u_{i+j}$ be the allowed HI criticality utilization in the system. $f_i^{\text{HI}}$ and $f_j^{\text{HI}}$ are the variables which are to be decided while applying DVFS subject to $u_i \frac{f_b}{f_i^{\text{HI}}} + u_j \frac{f_b}{f_j^{\text{HI}}} \leq u_{i+j}$ so that schedulability of the system is not violated. We find the optimal frequencies $f_i^{\text{HI}}$ and $f_j^{\text{HI}}$ using Karush-Kuhn-Tucker (KKT)[62] conditions. The HI-mode energy minimization objective for KKT conditions can be written as:

$$\textbf{minimize } (5.2)$$

$$\textbf{s.t} \qquad u_i \frac{f_b}{f_i^{\text{HI}}} + u_j \frac{f_b}{f_j^{\text{HI}}} \leq u_{i+j} \tag{5.3}$$

Adjoining the inequality constraint (5.3) with energy objective by considering KKT multiplier ($\mu$), (5.2) is minimized when

$$\nabla_{(f_i^{\text{HI}}, f_j^{\text{HI}})} E_{\text{HI}} + \mu \nabla_{(f_i^{\text{HI}}, f_j^{\text{HI}})} g\left(f_i^{\text{HI}}, f_j^{\text{HI}}\right) = 0 \tag{5.4}$$

subject to

$$g\left(f_i^{\text{HI}}, f_j^{\text{HI}}\right) = u_i \frac{f_b}{f_i^{\text{HI}}} + u_j \frac{f_b}{f_j^{\text{HI}}} \leq u_{i+j}; \tag{5.5}$$

$$\mu g\left(f_i^{\text{HI}}, f_j^{\text{HI}}\right) = 0 \quad \text{i.e,} \quad \mu \left(u_i \frac{f_b}{f_i^{\text{HI}}} + u_j \frac{f_b}{f_j^{\text{HI}}} - u_{i+j}\right) = 0; \quad \text{and } \mu \geq 0 \tag{5.6}$$

Solving (5.4) further,

$$\begin{pmatrix} \frac{\partial \bar{E}_{\text{HI}}}{\partial f_i} \\ \frac{\partial \bar{E}_{\text{HI}}}{\partial f_j} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{5.7}$$

$$\Leftrightarrow w_{\text{HI}} f_b(-u_i P_s (f_i^{\text{HI}})^{-2} + u_i \beta(\alpha-1)(f_i^{\text{HI}})^{\alpha-2}) + \mu f_b(u_i (f_i^{\text{HI}})^{-2}) = 0$$

$$\Leftrightarrow \beta(\alpha-1)(x)^{\alpha} = P_s - \frac{\mu}{w_{\text{HI}}}$$

$$\Leftrightarrow f_i^{\text{HI}} = \left(\frac{P_s - \frac{\mu}{w_{\text{HI}}}}{\beta(\alpha-1)}\right)^{1/\alpha} \tag{5.8}$$

and

$$\Leftrightarrow w_{\text{HI}} f_b(-u_j P_s (f_j^{\text{HI}})^{-2} + u_j \beta(\alpha-1)(f_j^{\text{HI}})^{\alpha-2}) + \mu f_b(u_j (f_j^{\text{HI}})^{-2}) = 0$$

$$\Leftrightarrow \beta(\alpha-1)(y)^{\alpha} = P_s - \frac{\mu}{w_{\text{HI}}}$$

$$\Leftrightarrow f_j^{\text{HI}} = \left(\frac{P_s - \frac{\mu}{w_{\text{HI}}}}{\beta(\alpha-1)}\right)^{1/\alpha} \tag{5.9}$$

From (5.8) and( 5.9), we obtain $f_i^{\text{HI}} = f_j^{\text{HI}}$ for which (5.2) is minimized. Hence, both the tasks share same execution frequency. By induction, this holds true even when there are more than two HI criticality tasks in the system. Hence all HI criticality

tasks in a tas kset share same execution frequency in HI mode behavior. Similarly, it can be proved that all LO criticality tasks in LO mode behaviours share same execution frequency and all HI criticality tasks in LO mode behaviours share same execution frequency. □

From Theorem 4, we can simplify the energy objective (4.1) and (4.2) as follows:

$$\begin{aligned} E_{\mathrm{LO}} =& w_{\mathrm{LO}} \cdot f_b \cdot U_{\mathrm{LO}}^{\mathrm{LO}} \cdot (P_s / f_{\mathrm{LO}}^{\mathrm{LO}} + \beta \cdot (f_{\mathrm{LO}}^{\mathrm{LO}})^{\alpha-1}) \\ &+ w_{\mathrm{LO}} \cdot f_b \cdot U_{\mathrm{HI}}^{\mathrm{LO}} \cdot (P_s / f_{\mathrm{HI}}^{\mathrm{LO}} + \beta \cdot (f_{\mathrm{HI}}^{\mathrm{LO}})^{\alpha-1}) \\ E_{\mathrm{HI}} =& w_{\mathrm{HI}} \cdot f_b \cdot U_{\mathrm{HI}}^{\mathrm{HI}} \cdot (P_s / f_{\mathrm{HI}}^{\mathrm{HI}} + \beta \cdot (f_{\mathrm{HI}}^{\mathrm{HI}})^{\alpha-1}) \end{aligned} \tag{5.10}$$

Theorem 4 serves in reducing the schedulability conditions represented in (3.4) for the EDF-VD with DVFS strategy. We now derive new schedulability conditions on the way to our energy minimization problem.

**Theorem 5.** Using Corollary 1 and Theorem 4, the feasible range of $x$ in the EDF-VD test after employing DVFS strategy can be formulated as:

$$x \geq \frac{\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{LO}}}}{1 - \frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}}} \tag{5.11}$$

$$x \frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}} + \frac{(f_{\mathrm{HI}}^{\mathrm{HI}} - f_{\mathrm{HI}}^{\mathrm{LO}})}{f_{\mathrm{HI}}^{\mathrm{LO}}} \frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{HI}}} + \frac{U_{\mathrm{HI}}^{\mathrm{HI}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{HI}}} \leq 1 \tag{5.12}$$

*Proof.* We follow the same procedure as in Baruah's EDF-VD algorithm [32] towards deriving the schedulability conditions in $x$ but with scaled utilization factors. We derive new schedulability conditions on the way to our energy minimization problem. Detailed proof is provided in (A.1). □

## 5.4 Optimality Condition in $x$

We continue to derive the necessary conditions in $x$ for any unicore DVFS strategy to be optimal. According to Theorem 4, let us denote the frequencies in an optimal solution as $f_{\mathrm{LO}\,opt}^{\mathrm{LO}}$, $f_{\mathrm{HI}\ opt}^{\mathrm{LO}}$ and $f_{\mathrm{HI}\,opt}^{\mathrm{HI}}$. Let us further define $K$, $L$ and $M$ as follows:

$$\begin{aligned} K &= \sum_{\chi_i = \mathrm{HI}} \frac{C_i(\mathrm{LO}) f_b}{T_i}, \\ L &= \sum_{\chi_i = \mathrm{LO}} \frac{C_i(\mathrm{LO}) f_b}{T_i}, \\ M &= 1 - \sum_{\chi_i = \mathrm{HI}} \frac{(C_i(\mathrm{HI}) - C_i(\mathrm{LO})) f_b}{T_i f_{\mathrm{HI}\,opt}^{\mathrm{HI}}}, \end{aligned} \tag{5.13}$$

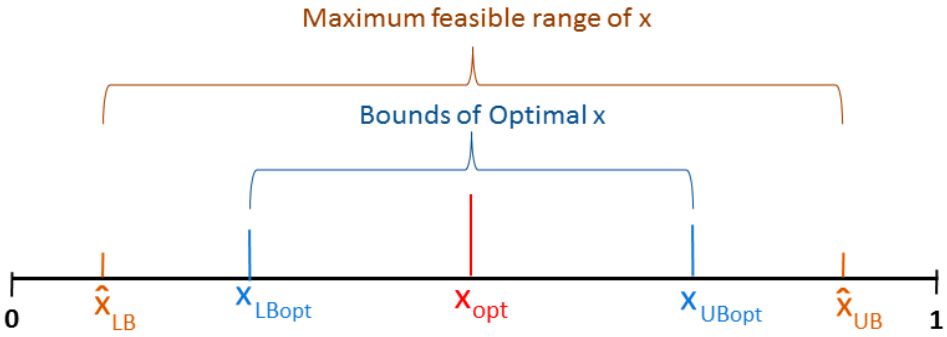


**Figure 5.1:** Bounds of $x$

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

$$x_{\mathrm{LB}opt} = \frac{\frac{U_{\mathrm{HI}}^{\mathrm{LO}}}{f_{\mathrm{HI}\ opt}^{\mathrm{LO}}} f_b}{1 - \frac{U_{\mathrm{LO}}^{\mathrm{LO}}}{f_{\mathrm{LO}\ opt}^{\mathrm{LO}}} f_b} = \frac{K / f_{\mathrm{HI}\ opt}^{\mathrm{LO}}}{1 - L / f_{\mathrm{LO}\ opt}^{\mathrm{LO}}} \tag{5.14}$$

$$\begin{aligned} x_{\mathrm{UB}opt} &= \left\lceil \frac{1 - \frac{U_{\mathrm{HI}}^{\mathrm{LO}}}{f_{\mathrm{HI}\ opt}^{\mathrm{LO}}} f_b - \frac{U_{\mathrm{HI}}^{\mathrm{HI}} - U_{\mathrm{HI}}^{\mathrm{LO}}}{f_{\mathrm{HI}\,opt}^{\mathrm{HI}}} f_b}{\frac{U_{\mathrm{LO}}^{\mathrm{LO}}}{f_{\mathrm{LO}\ opt}^{\mathrm{LO}}} f_b} \right\rceil^1 \\ &= \left\lceil \frac{M - K / f_{\mathrm{HI}\ opt}^{\mathrm{LO}}}{L / f_{\mathrm{LO}\ opt}^{\mathrm{LO}}} \right\rceil^1 \end{aligned} \tag{5.15}$$

Now, we consider choosing the optimal deadline scaling factor $x_{opt}$ and establish the following necessary condition for any optimal solution to exist. A pictorial representation of bounds of $x$ is made in Figure 5.1.

**Theorem 6.** An optimal solution to our unicore problem as formulated in Theorem 2, exists in two cases:

1. When $f_{\mathrm{LO}}^{\mathrm{LO}} = f_{\mathrm{HI}}^{\mathrm{LO}} = f_{\mathrm{HI}}^{\mathrm{HI}} = f_{\min}$ i.e., at the extreme case.
2. When $x_{\mathrm{LB}opt} = x_{opt} = x_{\mathrm{UB}opt}$, i.e., at equilibrium.

*Proof.* To ensure system schedulability, it must follow that $x_{\mathrm{LB}opt} \leq x_{opt} \leq x_{\mathrm{UB}opt}$ (see Corollary 1). From (5.14) and (5.15), we notice that $f_{\mathrm{LO}\ opt}^{\mathrm{LO}}$ and/or $f_{\mathrm{HI}\ opt}^{\mathrm{LO}}$ can be decreased by increasing $x_{\mathrm{LB}opt}$ and decreasing $x_{\mathrm{UB}opt}$, thus minimizing LO mode energy[1]. Similarly, $f_{\mathrm{HI}\,opt}^{\mathrm{HI}}$ can be decreased by decreasing $x_{\mathrm{UB}opt}$ to save energy consumption in HI mode. As a result, as long as task frequencies are not minimal 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 the frequencies are already lowered to $f_{\min}$ or, (ii) $x_{\mathrm{LB}opt} = x_{\mathrm{UB}opt}$.

□

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

Notice that Theorem 6 generalizes a similar condition in [23], where only dynamic energy in LO system mode is considered. Using Theorem 6 and the optimality conditions in $f_i^\chi$ and $x$, we now simplify our objective (5.10) to ease the adoption of KKT conditions. This is presented in next section. With results in Section 5.3 and Section 5.4, we now reduce the complexity of the objective, our approach is to find the inter-relations among frequencies $f^{\text{LO}}_{\text{LO}\,opt}$, $f^{\text{LO}}_{\text{HI}\ opt}$ and $f^{\text{HI}}_{\text{HI}\,opt}$. We then reduce our problem to a minimization problem in a two-dimensional space.

## 5.5 Simplified Energy Objective

From Theorem 6, we have at equilibrium, $x_{\text{LB}opt} = x_{opt} = x_{\text{UB}opt} \leq 1$. In addition, we can prove that in such a case, we can remove the $[\![\cdot]\!]$ operation when computing the upper bound on $x$: If $x_{\text{LB}opt} = x_{\text{UB}opt} < 1$, this is apparent. Otherwise, $x_{\text{LB}opt} = x_{\text{UB}opt} = 1$, we can derive that,

$$\begin{aligned} x_{\text{UB}opt} &= \left[\!\!\left[ \frac{1 - \frac{U^{\text{LO}}_{\text{HI}}}{f^{\text{LO}}_{\text{HI}\ opt}} f_b - \frac{U^{\text{HI}}_{\text{HI}} - U^{\text{LO}}_{\text{HI}}}{f^{\text{HI}}_{\text{HI}\,opt}} f_b}{\frac{U^{\text{LO}}_{\text{LO}}}{f^{\text{LO}}_{\text{LO}\,opt}} f_b} \right]\!\!\right]^1 \\ &\leq \left[\!\!\left[ \frac{1 - \frac{U^{\text{LO}}_{\text{HI}}}{f^{\text{LO}}_{\text{HI}\ opt}} f_b}{\frac{U^{\text{LO}}_{\text{LO}}}{f^{\text{LO}}_{\text{LO}\,opt}} f_b} \right]\!\!\right]^1 = 1. \end{aligned} \tag{5.16}$$

Thus, from (5.14) and (5.15), we have

$$\frac{K/f^{\text{LO}}_{\text{HI}\ opt}}{1 - L/f^{\text{LO}}_{\text{HI}\ opt}} = x_{opt} = \frac{M - K/f^{\text{LO}}_{\text{HI}\ opt}}{L/f^{\text{LO}}_{\text{LO}\,opt}}, \tag{5.17}$$

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

$$\Leftrightarrow x_{opt} = M.$$

Hence, at equilibrium, $x_{\text{LB}opt} = x_{opt} = x_{\text{UB}opt} = M$, where $M$ is a function of $f^{\text{HI}}_{\text{HI}\,opt}$ as defined in (5.13). Furthermore, due to the equilibrium condition, we can establish a relation between $f^{\text{LO}}_{\text{HI}\ opt}$ and $f^{\text{LO}}_{\text{LO}\,opt}$.

$$\begin{aligned} M &= \frac{M - K/f^{\text{LO}}_{\text{HI}\ opt}}{L/f^{\text{LO}}_{\text{LO}\,opt}} \\ \Leftrightarrow f^{\text{LO}}_{\text{HI}\ opt} &= \frac{K}{M.(1 - L/f^{\text{LO}}_{\text{LO}\,opt})}. \end{aligned} \tag{5.18}$$

Thus, through (5.18), we represent $f^{\text{LO}}_{\text{HI}\ opt}$ as a function of $f^{\text{LO}}_{\text{LO}\,opt}$ and $f^{\text{HI}}_{\text{HI}\,opt}$ (and $M$ is a function of $f^{\text{HI}}_{\text{HI}\,opt}$). As a result, we finally reduce our problem to a continuous

optimization problem in a two-dimensional space. Substituting (5.18) in (5.10), we get the final simplified energy objective as

$$
\begin{aligned}
E_{\mathrm{LO}} &= w_{\mathrm{LO}} f_b U_{\mathrm{LO}}^{\mathrm{LO}} \left( \frac{P_s}{f_{\mathrm{LO}\,opt}^{\mathrm{LO}}} + \beta (f_{\mathrm{LO}\,opt}^{\mathrm{LO}})^{\alpha-1} \right) \\
&+ w_{\mathrm{LO}} f_b U_{\mathrm{HI}}^{\mathrm{LO}} \left( \frac{P_s M (1 - \frac{L}{f_{\mathrm{LO}\,opt}^{\mathrm{LO}}})}{K} + \beta \left( \frac{K}{M \left( 1 - \frac{L}{f_{\mathrm{LO}\,opt}^{\mathrm{LO}}} \right)} \right)^{\alpha-1} \right) \\
E_{\mathrm{HI}} &= w_{\mathrm{HI}} f_b U_{\mathrm{HI}}^{\mathrm{HI}} \left( \frac{P_s}{f_{\mathrm{HI}\,opt}^{\mathrm{HI}}} + \beta (f_{\mathrm{HI}\,opt}^{\mathrm{HI}})^{\alpha-1} \right) \\
E &= E_{\mathrm{LO}} + E_{\mathrm{HI}}
\end{aligned}
\tag{5.19}
$$

This objective is still convex but simplified to two-dimensional space with the reduced number of constraints. Hence we apply KKT conditions to find the optimal solution.

## 5.6 KKT Conditions for Simplified Objective

From Theorem 6 , it is straight forward to check if the optimal solution exists in extreme case. When the solution exists at equlibrium, we can apply KKT conditions to solve the problem. Since the objective is now two-dimensional with $f_{\mathrm{LO}\,opt}^{\mathrm{LO}}$ and $f_{\mathrm{HI}\,opt}^{\mathrm{HI}}$ as variables, we first proceed to find $f_{\mathrm{LO}\,opt}^{\mathrm{LO}}, f_{\mathrm{HI}\,opt}^{\mathrm{HI}}$ and $x$ using KKT conditions and then $f_{\mathrm{HI}\,opt}^{\mathrm{LO}}$ can calculated using (5.18).

The new objective with reduced search space for the KKT conditions can be written as

$$
\begin{aligned}
&\textbf{minimize} \quad (5.19) \\
&\textbf{s.t.} \quad x_{opt} = M\,;\, x_{opt} \geq x_{\mathrm{LB}opt}\,;\, x_{opt} \leq x_{\mathrm{UB}opt}\,; \\
&f_{\mathrm{LO}}^{\mathrm{LO}} \geq f_{min}\,;\, f_{\mathrm{LO}}^{\mathrm{LO}} \leq f_{max}\,; f_{\mathrm{HI}}^{\mathrm{HI}} \geq f_{min}\,;\, f_{\mathrm{HI}}^{\mathrm{HI}} \leq f_{max}.
\end{aligned}
$$

Considering Lagrange's constant $\lambda_1$ and converting the inequality constraints to Lagrangian form using KKT multipliers ($\mu_i$), we can formulate the conditions as

$$
\begin{aligned}
&\nabla_{f,x} \big[ (E_{\mathrm{LO}} + E_{\mathrm{HI}}) + \lambda_1 (x_{opt} - M) - \mu_1 (x_{opt} - x_{\mathrm{LB}opt}) \\
&+ \mu_2 (x_{opt} - x_{\mathrm{UB}opt}) - \mu_3 (f_{\mathrm{LO}}^{\mathrm{LO}} - f_{min}) + \mu_4 (f_{\mathrm{LO}}^{\mathrm{LO}} - f_{max}) \\
&- \mu_5 (f_{\mathrm{HI}}^{\mathrm{HI}} - f_{min}) + \mu_6 (f_{\mathrm{HI}}^{\mathrm{HI}} - f_{max}) \big] = 0
\end{aligned}
$$

$$
\begin{aligned}
\textbf{s.t.} \quad & x_{opt} = M\,;\, \mu_1 (x_{opt} - x_{\mathrm{LB}opt}) = 0\,; \\
& \mu_2 (x_{opt} - x_{\mathrm{UB}opt}) = 0\,;\, \mu_3 (f_{\mathrm{LO}}^{\mathrm{LO}} - f_{min}) = 0\,; \\
& \mu_6 (f_{\mathrm{LO}}^{\mathrm{LO}} - f_{max}) = 0\,;\, \mu_5 (f_{\mathrm{HI}}^{\mathrm{HI}} - f_{min}) = 0 \\
& \mu_6 (f_{\mathrm{HI}}^{\mathrm{HI}} - f_{max}) = 0 \text{ and } \mu_i \geq 0 \quad \forall i = 1, ..., 6
\end{aligned}
$$

where $E_{\mathrm{LO}}$ and $E_{\mathrm{HI}}$ are represented by (5.10), and $x_{\mathrm{LB}opt}$ and $x_{\mathrm{UB}opt}$ are represented by (5.14) and (5.15) respectively.

**Algorithm 1:** Finding optimal $f_{\text{LO}}^{\text{LO}}$, $f_{\text{HI}}^{\text{LO}}$ and $f_{\text{HI}}^{\text{HI}}$

**input** : $\tau$, $f_b$, $f_{\min}$, $f_{\max}$, $w_{\text{LO}}$ and $w_{\text{HI}}$
**output**: $f_{\text{LO}\,opt}^{\text{LO}}$, $f_{\text{HI}\ opt}^{\text{LO}}$, $f_{\text{HI}\,opt}^{\text{HI}}$, $x_{opt}$
**if** System feasible when $f_{\text{LO}}^{\text{LO}} = f_{\text{HI}}^{\text{LO}} = f_{\text{HI}}^{\text{HI}} = f_{\max}$ according to Corollary 1 **then**
  **if** System feasible when $f_{\text{LO}}^{\text{LO}} = f_{\text{HI}}^{\text{LO}} = f_{\text{HI}}^{\text{HI}} = f_{\min}$ according to Corollary 1 **then**
    1.$f_{\chi_1}^{\chi_2} \leftarrow f_{\min}\ \forall\chi_1\forall\chi_2$;
  **else**
    2. According to (5.13) and (5.18), reduce the energy objective given by Theorem 2 into a function of $f_{\text{HI}}^{\text{LO}}$ and $f_{\text{HI}}^{\text{HI}}$;
    3. Solve the two-dimensional continuous optimization problem with constraints that $f_{\min} \leq f_{\chi_1}^{\chi_2} \leq f_{\max}\ \forall\chi_1\forall\chi_2$, obtain $f_{\text{LO}\,opt}^{\text{LO}}$ and $f_{\text{HI}\,opt}^{\text{HI}}$;
    4. Set $f_{\text{HI}\ opt}^{\text{LO}}$ according to (5.18) and $x_{opt}$ according to (5.17);
    **return** Success;
  **end**
**else**
  **return** Failure;
**end**

Thus, we obtain KKT conditions with reduced complexity as compared to (5.1). However, even this simplified conditions are computationally intense because of six inequality constants ($\mu_1 - \mu_6$). This leads to 64 cases which are to be considered to find optimal frequencies. Hence, we do not use convex solvers such as KKT conditions to solve our energy minimization problem. Rather, we develop a simple algorithm with the insights from simplified energy objective to find the solution to our unicore problem.

## 5.7 Optimal Solution

Note that, if the objective is only to save LO mode energy as in [23], then fixing $f_{\text{HI}}^{\text{HI}}$ to $f_{\max}$ would be optimal. Intuitively we can see that this would relax the system operation in LO mode. In other words, the upper bound on $x$ is maximized, giving more room for reducing LO mode task frequencies to increase the lower bound on $x$. However, when reducing HI mode energy $f_{\text{HI}}^{\text{HI}}$ is also the objective, the problem becomes more complex as energy consumptions in both modes are dependent and need to be minimized together. Based on this observation and the established optimality conditions, we now present our proposed algorithm. We propose our optimal solution as shown in Algorithm 1. Before finding optimal frequencies, we first need to check whether the system is feasible (by assuming maximum frequency $f_{\max}$). This is done according to Corollary 1. If feasible, then the solution exists in two optimal cases as suggested by Theorem 6.

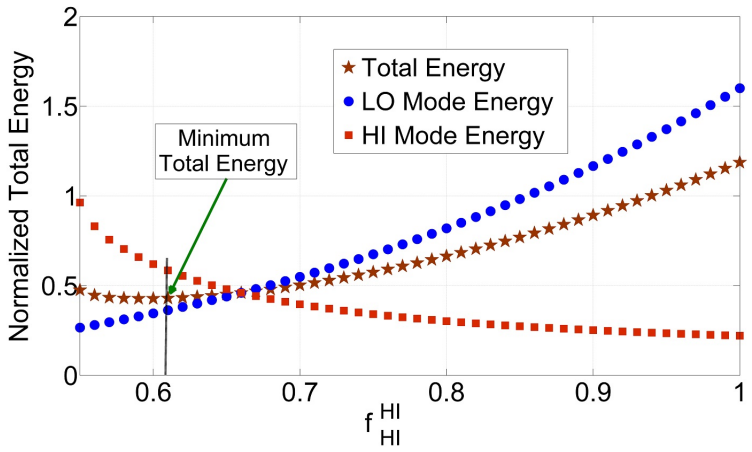


**Figure 5.2:** LO mode and HI mode energy as a function of $f_{\mathrm{HI}}^{\mathrm{HI}}$

### 5.7.1 Extreme case ($f_{\mathrm{LO}}^{\mathrm{LO}} = f_{\mathrm{HI}}^{\mathrm{LO}} = f_{\mathrm{HI}}^{\mathrm{HI}} = f_{\min}$)

In the extreme case, i.e., when all the mode frequencies are set to $f_{\min}$, we just need to check the system schedulability in such an extreme case. This is straight forward to verify with Corollary 1. Thus, we will concentrate on the other possibility, i.e., the equilibrium case.

### 5.7.2 Equilibrium case ($x_{\mathrm{LB}opt} = x_{opt} = x_{\mathrm{UB}opt}$)

In this case, our first approach is to solve a two-dimensional optimization problem (5.19) to find the optimal frequency assignment. However, in general, it is difficult to obtain a closed form solution that is not bound to any specific task set. This is mainly due to mathematical complexity of the final reduced objective function. Therefore, one has to resort to numerical solutions to solve the two-dimensional continuous optimization problem. Hence, we proceed to present a simple heuristic to solve the problem when the solution exists in the equilibrium condition.

## 5.8 Heuristic Solution

According to (5.10), we observe that, with increasing $f_{\mathrm{HI}}^{\mathrm{HI}}$, the HI mode energy consumption increases at a higher rate. This is due to the fact that the first and second order differential of $E_{\mathrm{HI}}$ with respect to $f_{\mathrm{HI}}^{\mathrm{HI}}$ is positive. Furthermore, we observe that the LO mode energy $E_{\mathrm{LO}}$ decreases as $f_{\mathrm{HI}}^{\mathrm{HI}}$ increases. The reason is that under this case, as HI mode frequency is increased, the LO mode becomes more relaxed where frequencies can be reduced for energy saving. Hence the first order differential of $E_{\mathrm{LO}}$ with respect to $f_{\mathrm{HI}}^{\mathrm{HI}}$ is negative. Furthermore, as $f_{\mathrm{HI}}^{\mathrm{HI}}$ gets larger, the LO mode frequencies would eventually converge to $f_{\min}$, thus minimizing $E_{\mathrm{LO}}$. Hence, the rate at which $E_{\mathrm{LO}}$ drops will decrease or the second order differential of $E_{\mathrm{LO}}$ is non-negative. The above observations are clear if we consider the same example from Table 4.1 and plot $E_{\mathrm{LO}}$ and $E_{\mathrm{HI}}$ as a function of $f_{\mathrm{HI}}^{\mathrm{HI}}$. This is demonstrated in

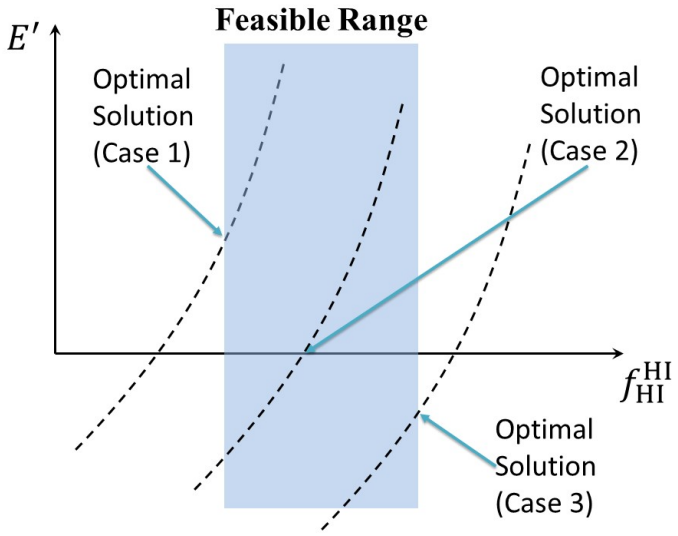


**Figure 5.3:** Different cases to check in Algorithm 2

**Algorithm 2:** Simple heuristic to find the solution in equilibrium case

**input** : $\tau$, $f_b$, $f_{\min}$, $f_{\max}$, $w_{\mathrm{LO}}$ and $w_{\mathrm{HI}}$
**output**: $f_{\mathrm{LO}\,opt}^{\mathrm{LO}}$, $f_{\mathrm{HI}\ opt}^{\mathrm{LO}}$, $f_{\mathrm{HI}\,opt}^{\mathrm{HI}}$, $x_{opt}$
1. Determine the feasible range of $f_{\mathrm{HI}}^{\mathrm{HI}}$ according to the Corollary 1;
2. If $E'$ is non-negative at the smallest feasible $f_{\mathrm{HI}}^{\mathrm{HI}}$, set this as $f_{\mathrm{HI}\ opt}^{\mathrm{LO}}$;
3. If $E'$ is non-positive at the biggest feasible $f_{\mathrm{HI}}^{\mathrm{HI}}$, set this as $f_{\mathrm{HI}\ opt}^{\mathrm{LO}}$;
4. If the above does not hold, we do a binary search to find the $f_{\mathrm{HI}}^{\mathrm{HI}}$ where $E' = 0$, set it as $f_{\mathrm{HI}\ opt}^{\mathrm{LO}}$;

Figure 5.2. Furthermore, since $E_{\mathrm{LO}}$ is decreasing while $E_{\mathrm{HI}}$ is increasing, there is a trade-off between LO and HI mode energy consumptions. Therefore, there exists an $f_{\mathrm{HI}}^{\mathrm{HI}}$ in between $f_{\min}$ and $f_{\max}$, which leads to the minimum overall energy. Our goal is to find such $f_{\mathrm{HI}}^{\mathrm{HI}}$ which we achieve using a simple heuristic solution.

We now present our heuristic to solve the two-dimensional problem (5.19) when the solution exists at equilibrium. The steps are listed in Algorithm 2. Let us denote $E'_{\mathrm{LO}}$ and $E'_{\mathrm{HI}}$ as the first order differential of LO and HI mode energy with respect to $f_{\mathrm{HI}}^{\mathrm{HI}}$, respectively. We have $E' = E'_{\mathrm{LO}} + E'_{\mathrm{HI}}$. Let the feasible range of $f_{\mathrm{HI}}^{\mathrm{HI}}$ be $[f_{\mathrm{HI}\,min}^{\mathrm{HI}}, f_{max}]$ where $f_{\mathrm{HI}\,min}^{\mathrm{HI}}$ is the minimum HI mode frequency required to ensure HI mode schedulability of $\tau$. It is calculated by fixing $f_{\mathrm{LO}\,opt}^{\mathrm{LO}} = f_{\mathrm{HI}\ opt}^{\mathrm{LO}} = f_{max}$ in (5.18) and can be formulated as:

$$f_{\mathrm{HI}\,min}^{\mathrm{HI}} = \frac{U_{\mathrm{HI}}^{\mathrm{HI}} - U_{\mathrm{HI}}^{\mathrm{LO}}}{1 - \frac{K}{f_{max} - L}} \tag{5.20}$$

If there exists an optimal $f_{\mathrm{HI}}^{\mathrm{HI}}$ within the range, then it can be in one of these three cases as shown in Figure 5.3:

**Case 1:** If $E'$ is always increasing, then the solution exists when $f_{\mathrm{HI}\,opt}^{\mathrm{HI}} = f_{\mathrm{HI}\,min}^{\mathrm{HI}}$, being the lowest energy point.

**Case 2:** If $E'$ is always decreasing, then the solution exists when $f_{\mathrm{HI}\,opt}^{\mathrm{HI}} = f_{max}$, being the lowest energy point.

**Case 3:** If the above cases does not hold, then there exists a stationary point in $E'$ for which $E' = 0$. Such minimal energy location of $f_{\mathrm{HI}}^{\mathrm{HI}}$ in the range can be found using binary search.

Thus, we have a simple heuristic to find the mode frequencies when the solution exists at equilibrium.

# Chapter 6

# Energy Minimization on Multi-cores

Now, we extend the unicore energy efficient algorithm presented in the previous section, to multi-cores. We consider the same energy objective represented in (5.19) for multi-cores. This chapter is organized as follows: Firstly, we examine two existing bin packing techniques for mapping tasks on multi-cores and analyze if they are energy-aware. This is detailed in Section 6.2 and Section 6.3. In Section 6.4, we propose an energy minimized multi-core mapping technique called Energy Minimized Mixed-Criticality Mapping Technique (EM3) where tasks with different criticality are mixed on a core. Finally, an energy-aware mapping method called Isolated Mixed-Criticality Mapping Method (IM3) is presented in Section 6.5 where the tasks with different criticality levels are isolated.

Minimizing the dynamic energy consumption by adapting parameters such as frequency in multi-core is not as straight-forward as that of the unicore case. When a task set has to be scheduled on a multi-core, few questions arise when energy reduction is considered : (i) how to map the tasks onto available cores? (ii) at what frequencies the tasks have to be executed on each core? (iii) what is the optimal number of active cores to be used such that the total energy is minimized? To understand the problem on multi-core and to answer these questions, we consider the same example presented in Section. 4.1 and analyze two existing mapping techniques - Baruah's method and Gu's method.

## 6.1 Overview of Existing Methods

In this section, we describe Baruah's and Gu's method for multi-core mapping.

## 6.2 Baruah's method

The mapping procedure followed in Baruah's method is briefly introduced here. In this method, tasks are mapped onto multiple cores using the First-Fit task allocating technique where a task is assigned to immediately available core. The process is executed in two phases:

- First, all HI criticality tasks sorted with decreasing utilization are assigned to processors using the FF such that the cumulative HI criticality utilization in each processor is less than 3/4 (It follows from [32] that, task system that is clairvoyant schedulable on a speed 3/4 processor is scheduled by the EDF-VD. Hence, HI criticality utilization $\leq 3/4$).
- In the second phase, all LO criticality tasks sorted with decreasing utilization are assigned to processors using the FF such that the cumulative LO criticality utilization in each processor is less than 3/4.

When the mapping is successful, the tasks on each core are scheduled independently with EDF-VD scheduling algorithm [32]. The energy comparison between Baruah's method, Gu's method and EM3 is shown in Figure 4.2. Baruah's method does not consider energy minimization while mapping but aims at schedulability. Hence, each nonempty core holds maximum possible utilization, thus, loading the core and increasing the execution frequency. Therefore, the energy consumption is at its worst when FF technique is used for mapping.

## 6.3 Gu's method

In this method, first, all HI criticality tasks are mapped onto multiple cores using the Worst-Fit allocating technique. When it is the turn for assigning LO criticality tasks in the second phase, the tasks are mapped using the FF technique. Though Gu's method also favors schedulability, it performs better than Baruah's method in terms of energy. The reason is that the HI criticality tasks are distributed over all the available cores, thus providing more slack to decrease the execution frequency on each core.

**Note:** We slightly modify both Baruah's and Gu's method for a fair comparison with the proposed methods, EM3 and IM3: If a core contains only LO criticality tasks while assigning tasks in the second phase, we ensure that the cumulative sum of utilization of tasks does not exceed 1 instead of 3/4. In such cases, the tasks on that core are scheduled using EDF as there is only one criticality level. Furthermore, we also switch the processor to sleep mode when it is idle.

Hence, the results motivate us to use WF technique for mapping both LO and HI criticality tasks on multi-core, with energy minimization as our goal. With this notion, we present a new energy efficient multi-core mapping procedure for mixed-criticality tasks where the WF technique is used for bin packing.

## 6.4 Energy Minimized Mixed-Criticality Mapping (EM3)

EM3 makes use of the WF technique to allocate both LO and HI criticality tasks on multiple cores. Utilizing all the available cores for mapping may not be optimal as each core contributes its static energy for total energy consumption. Therefore, we apply a simple heuristic to identify the number of cores to be used for mapping the tasks. In this approach, first we find the optimal number of active cores for mapping. Then, we map the tasks on cores using the WF technique. Finally, DVFS is employed on all cores using the unicore solution presented in (5.8) and tasks are scheduled using EDF-VD.

The procedure followed in EM3 is as follows: If $m$ is the total number of cores available, then for all possible values of integer $n$, such that $2 \leq n \leq m$, tasks are mapped onto $n$ cores using the WF technique. Since our solution for energy minimization employs the EDF-VD for scheduling, we ensure that the cumulative $\chi$ criticality utilization of tasks in each processor does not exceed 3/4. However, if a core contains tasks with only single criticality level, we ensure that the cumulative sum of utilization does not exceed 1. Finally, the tasks on each core are scheduled using EDF-VD. The heuristic is carried out in 3 phases:

- In the first phase, all HI criticality tasks are allocated onto $n$ cores using the WF method, with their decreasing utilization. We ensure that the cumulative HI criticality utilization of tasks on each core does not exceed 3/4. If there are any HI criticality tasks left for mapping, then $n$ is incremented by 1 such that $n \leq m$ and the first phase is repeated. If $n > m$, then mapping is terminated, declaring failure.

- In the second phase, all LO criticality tasks are allocated on $n$ cores using the WF method. If there exists a HI criticality task on the core, then we ensure that the cumulative LO criticality utilization of tasks in each core does not exceed 3/4; else 1. If no tasks are left for allocation, then mapping is completed and we proceed to next step; else $n$ is incremented by 1 such that $n \leq m$ and the first phase is executed again. If $n > m$, then mapping is terminated, declaring failure.

- Finally, DVFS strategy is applied for tasks on each core using the heuristics presented in Section. 5.8 and all tasks are scheduled using EDF-VD. However, if there is any core with only $\chi$ criticality tasks present, then traditional EDF is applied.

The total energy of the system is calculated using (5.10) for all possibilities of $n$. The number of cores $n$ for which the total energy is minimum is the optimal number of cores to be used for mapping.

Though EM3 shows significant amount of energy saving, tasks with different criticality levels are mixed and HI criticality tasks always face interference from LO criticality tasks. Traditional isolated partitioning is still used in industry to avoid task interference and guarantee safety. In isolated partitioning, the tasks with different criticality levels are not mixed on a core. This also eases the scheduling as

each core contains only single criticality tasks and conventional algorithms such as EDF can be used. Supporting isolated mapping, we present another mapping procedure where tasks with different criticality levels are not mixed on a core and show that the energy consumption is almost same as that in EM3. We call this method as "Isolated Mixed-Criticality Mapping Method".

## 6.5 Isolated Mixed-Criticality Mapping Method (IM3)

In this method, we provide complete isolation between tasks with different criticality levels. Hence, all LO criticality tasks are mapped onto $l$ cores, and all HI criticality tasks are mapped onto $h$ cores such that $2 \leq (l + h) \leq m$. Since LO and HI criticality tasks are isolated, the LO criticality tasks can be scheduled using the EDF. However, EDF cannot be employed for HI criticality tasks as the system shows both LO and HI mode behaviors. Therefore, we apply EDF-VD test on HI criticality tasks.

Before presenting our algorithm (Section. 6.5.3), we first define the energy model and schedulability conditions for both LO and HI criticality tasks (Section. 6.5.1 and Section. 6.5.2).

### 6.5.1 Conditions for LO criticality tasks

The energy objective formulated in (5.10) holds for LO criticality tasks but can be simplified because $U_{\mathrm{HI}}^{\mathrm{LO}} = U_{\mathrm{HI}}^{\mathrm{HI}} = 0$ in $l$ cores. It should be noted that the weight factors can be neglected in the case of LO criticality tasks as there is no mode switching. However, for a fair comparison with EM3, we consider weight factor in our energy objective and it will not affect the optimal frequencies. Thus, the simplified power model for tasks on each core $i$ , $\forall i \in \{1, ....l\}$ can be formulated from (5.10) as:

$$E_i = w_{\mathrm{LO}} \cdot f_b \cdot U_{\mathrm{LO}}^{\mathrm{LO}} \cdot (P_s / f_{\mathrm{LO}\,i}^{\mathrm{LO}} + \beta \cdot (f_{\mathrm{LO}\,i}^{\mathrm{LO}})^{\alpha-1}) \tag{6.1}$$

The objective is to minimize $E_i$, on each core $i$.

The necessary condition for EDF is: $U_{LO}^{LO} \leq 1$. To save energy, we can utilize the available slack on each core $i$ by decreasing the frequency to the possible minimum. Hence, the schedulability condition after employing DVFS changes to $\frac{U_{LO}^{LO}}{f_{LO}^{LO}} \leq 1$ or, in other words, $f_{LO}^{LO} = U_{LO}^{LO}$. Therefore, we set the execution frequency of tasks on $l$ cores equal to their LO criticality utilization, i.e., $\forall i \in \{1, ....l\}$ , $(f_{LO\,opt}^{LO})_i = U_{LO\,i}^{LO}$.

### 6.5.2 Conditions for HI criticality tasks

Since the system has nominal and overload mode in the case of HI criticality tasks, there exists two frequencies: one for LO mode behavior and another for HI mode behavior. With this notion, the energy objective on each core $j$, $\forall j \in \{1, ....h\}$ can be formulated as:

$$\begin{aligned} E_j = w_{\text{LO}} \cdot f_b \cdot U^{\text{LO}}_{\text{HI}\,j} \cdot (P_s / f^{\text{LO}}_{\text{HI}\,j} + \beta \cdot (f^{\text{LO}}_{\text{HI}\,j})^{\alpha-1}) \\ + w_{\text{HI}} \cdot f_b \cdot U^{\text{HI}}_{\text{HI}\,j} \cdot (P_s / f^{\text{HI}}_{\text{HI}\,j} + \beta \cdot (f^{\text{HI}}_{\text{HI}\,j})^{\alpha-1}) \end{aligned} \tag{6.2}$$

Therefore, the objective is to save both LO mode energy and HI mode energy of HI criticality tasks depending on task utilization and weight factors, i.e., minimize $E_j$, $\forall j \in \{1, ....h\}$.

For all HI criticality tasks to be schedulable on each core $j$ the necessary condition in the EDF-VD can be written as,

$$\frac{U^{\text{LO}}_{\text{HI}\,j}}{f^{\text{LO}}_{\text{HI}\,j}} + \frac{(U^{\text{HI}}_{\text{HI}\,j} - U^{\text{LO}}_{\text{HI}\,j})}{f^{\text{HI}}_{\text{HI}\,j}} \leq 1 \tag{6.3}$$

Now, to minimize energy, we can decrease $f^{\text{LO}}_{\text{HI}\,j}$ and/or $f^{\text{HI}}_{\text{HI}\,j}$ until (6.3) equalizes to 1, or until $f_{min}$ is reached. Hence, in an optimal solution,

$$(f^{\text{LO}}_{\text{HI}\,opt})_j = \left[\!\!\left[ \frac{U^{\text{LO}}_{\text{HI}\,j}}{1 - \frac{(U^{\text{HI}}_{\text{HI}\,j} - U^{\text{LO}}_{\text{HI}\,j})}{(f^{\text{HI}}_{\text{HI}\,opt})_j}} \right]\!\!\right]_{f_{min}} \tag{6.4}$$

Thus, we have inter-relations between LO and HI mode frequencies in an optimal solution.

### 6.5.3 Algorithm

The pseudocode of our approach is presented in Algorithm 3 and the algorithm is as follows: Alike EM3, we find the optimal active cores to be used for scheduling in IM3. We apply a simple heuristic to identify the optimal $l$ and $h$ such that the total energy consumption is at the minimum. Once $l$ and $h$ are known, the $\chi$ criticality tasks are allocated using the WF technique in decreasing utilization order. Finally, EDF is employed to schedule LO criticality tasks and EDF-VD for HI criticality tasks. We ensure that, the cumulative sum of LO criticality utilization on $l$ cores and the cumulative sum of HI criticality utilization on $h$ cores does not exceed 1. The steps involved are as follows:

1. First, we find the minimum cores ($l'$ for LO criticality tasks and $h'$ for HI criticality tasks) required for successful scheduling. Therefore, we set $l' = \left\lceil U^{LO}_{LO} \right\rceil$ and $h' = \left\lceil U^{HI}_{HI} \right\rceil$. If $(l' + h') > m$, then terminate the mapping process and declare failure; else, set $l = l'$ and $h = h'$, and proceed to step 2.

2. Allocate all LO criticality tasks onto $l$ cores and all HI criticality tasks onto $h$ cores using the WF technique, ensuring that the cumulative sum of LO criticality utilization on $l$ cores and the cumulative sum of HI criticality utilization on $h$ cores does not exceed 1.

**Algorithm 3:** Task mapping using IM3 for finding optimal frequencies for each core

**input** : $\tau$, $f_b$, $f_{\min}$, $f_{\max}$, $m$, $w_{\text{LO}}$ and $w_{\text{HI}}$
**output**: $(f^{\text{LO}}_{\text{LO}\,opt})_i$, $(f^{\text{LO}}_{\text{HI}\,opt})_i$, $(f^{\text{HI}}_{\text{HI}\,opt})_i$, $(f^{\text{LO}}_{\text{LO}\,opt})_j$, $(f^{\text{LO}}_{\text{HI}\,opt})_j$, $(f^{\text{HI}}_{\text{HI}\,opt})_j$, $l\,, h$

```
Set l' = ⌈U_LO^LO⌉ and h' = ⌈U_HI^HI⌉;
if (l' + h') ≤ m then
    Set l = l' and h = h';
    ▷Set △ = {} (Empty list of set of m cores);
    while l ≤ m − h' do
        while h ≤ m − l' do
            if l + h > m then
                break;
            else
                Allocate all LO and HI criticality tasks onto l and h cores
                respectively using the WF;
                if No tasks are left for mapping then
                    Add the set of l , h cores to △ with list index (l , h);
                end
            end
            h = h + 1;
        end
        l = l + 1;
    end
    for all list items (set of cores) in △ do
        Calculate optimal frequencies for tasks in each core using heuristics
        presented in (6.5) and find total energy of the system;
    end
    The list item in △ with lowest total energy is the optimal mapping and so the
    optimal frequencies for each core.
    Set the corresponding (f^LO_LO opt)_i, (f^LO_HI opt)_i, (f^HI_HI opt)_i , l, ∀i ∈ [1 , l];
    Set the corresponding (f^LO_LO opt)_j, (f^LO_HI opt)_j, (f^HI_HI opt)_j , h, ∀i ∈ [1 , h];
    return Success;
else
    return Failure;
end
```

3. DVFS strategy is applied for LO criticality tasks on $l$ cores as described in Sec 6.5.1 and they are scheduled using EDF algorithm.

4. Similarly, DVFS is applied for HI criticality tasks residing on $h$ cores. We find the optimal frequencies $(f^{\text{LO}}_{\text{HI}\,opt})_j$ and $(f^{\text{HI}}_{\text{HI}\,opt})_j$ using a simple heuristic that is similar to the unicore solution, presented in Section. 5.8. First, we find the minimum required HI mode frequency $(f^{\text{HI}}_{\text{HI}\,min})_j$ for the task set to be schedulable on core $j$. This is done by fixing $f^{\text{LO}}_{\text{HI}\,j} = f_{max}$ in (6.3) as,

$$(f^{\mathrm{HI}}_{\mathrm{HI}\,min})_j = \left[\!\!\left[ \frac{(U^{\mathrm{HI}}_{\mathrm{HI}\,j} - U^{\mathrm{LO}}_{\mathrm{HI}\,j})}{1 - (\frac{U^{\mathrm{LO}}_{\mathrm{HI}\,j}}{f_{max}})} \right]\!\!\right]_{f_{min}} \tag{6.5}$$

Then, for all values of $f^{\mathrm{HI}}_{\mathrm{HI}\,j} \in [(f^{\mathrm{HI}}_{\mathrm{HI}\,min})_j\,,\, f_{max}]$, with a small increment $\Delta$, the total energy is calculated. The $f^{\mathrm{HI}}_{\mathrm{HI}\,j}$ for which the total energy is minimum is the optimal HI mode frequency and the corresponding $(f^{\mathrm{LO}}_{\mathrm{HI}\,opt})_j$ is found using (6.4).

5. Repeat steps 2 to 4 for all combinations of $l \in [l',\, m - h']$ and $h \in [h'\,,\, m - l']$ and compute the total energy of the system. The values of $l$ and $h$ for which the total energy calculated in step 5 is minimum, is the optimal number of active cores and so, the corresponding frequencies.

The advantage of IM3 is that the tasks with different criticality levels are completely isolated and are scheduled on a multi-core processor. Though EM3 gives better results, the energy difference between EM3 and IM3 is comparatively less. EM3 is better for resource constraint systems where tasks with different criticality levels communicate with each other. IM3 can be a suitable method where complete isolation is required which are most likely preferred by the industry.

# Chapter 7

# Evaluation

We first evaluate our unicore solution with a task set from real-time Flight Management System (FMS), and later multi-core mapping techniques with extensive simulations using synthetic task sets. We demonstrate energy savings and the effect of various parameters on energy minimization. We also compare our approach against existing Bin packing techniques – Baruah's and Gu's method. Though these techniques do not focus on energy and are not the same as what we have proposed they are the closest in the current literature.

| $\tau$ | Criticality | $T_i(ms)$ | $C_i$(LO) (ms) | $C_i$(HI) (ms) |
|---|---|---|---|---|
| $\tau_1$ | HI | 5000 | 15 | 21 |
| $\tau_2$ | HI | 200 | 18 | 25 |
| $\tau_3$ | HI | 1000 | 16 | 22 |
| $\tau_4$ | HI | 1600 | 20 | 28 |
| $\tau_5$ | HI | 100 | 18 | 26 |
| $\tau_6$ | HI | 1000 | 17 | 24 |
| $\tau_7$ | HI | 1000 | 15 | 21 |
| $\tau_8$ | LO | 1000 | 100 | 100 |
| $\tau_9$ | LO | 1000 | 80 | 80 |
| $\tau_{10}$ | LO | 1000 | 140 | 140 |
| $\tau_{11}$ | LO | 1000 | 100 | 100 |

**Table 7.1:** FMS task set

## 7.1 Experimental Setup

For unicore, we conduct experiments using an FMS task set, listed in Table 7.1. The task set consists of seven HI criticality tasks and four LO criticality tasks. In all the experiments on unicore, we assume that the processor is DVFS capable

with $[f_{\min}\,,\ f_b\,,\ f_{\max}] = [0.5\,,\ 0.8\,,\ 1]$ and has capacitance factor $\beta = 0.8$. In all the simulations, we consider $P_s = 0.2$ and $\alpha = 2$, unless it is otherwise mentioned.

To evaluate the mapping techniques on multi-core, we implement a task generator using Mathematica to generate 100 random feasible task sets. Each task set consists of $80 - 100$ mixed criticality tasks that sum up to a total LO or HI criticality utilization of 3.0. The number of LO and HI criticality tasks in each task set was varying as it was generated randomly . We assume that each core is DVFS capable with $[f_{\min}\,,\ f_b\,,\ f_{\max}] = [0.4\,,\ 0.85\,,\ 1]$. We fix $P_s = 0.3$, $\alpha = 2$, $\beta = 0.8$, $w_{\text{LO}} = w_{\text{HI}} = 0.5$ and consider 6 cores in all the cases unless it is otherwise mentioned. The normalized total energy consumption that we present for multi-core is the sum of normalized energy on each core.

## 7.2 Evaluation on Unicore

First, we apply our proposed solution to the FMS task set on a unicore. We explain in this section, the effects of different parameters on the expected minimal energy.

### 7.2.1 Impact of Weight Factors

First, we show the importance of weight factors in energy minimization. We calculate the normalized total energy dissipation in the system with different mode weights $w_{\text{LO}}$ and $w_{\text{HI}}$ for $\alpha = 2\,,3\,,4$.

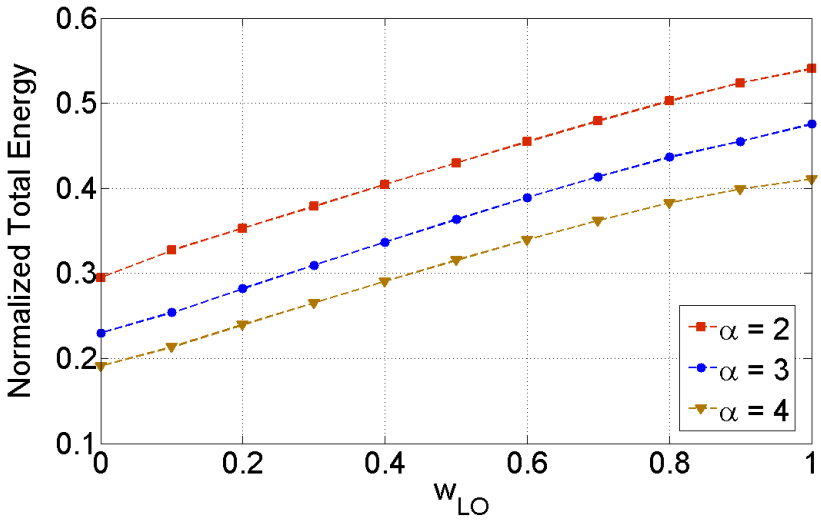


**Figure 7.1:** Impact of weight factors on energy minimization

The results are shown in Figure 7.1. As we see in the plot, the minimal expected energy increases with $w_{\text{LO}}$. This is because, the LO mode utilization is higher than the HI mode utilization $\left(U_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} > U_{\text{HI}}^{\text{HI}}\right)$ in the considered task set. Therefore, when the system is active in LO mode for excessive amount of time both the static and dynamic energy increases. However, if we consider the case where $U_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} <$

$U_{\mathrm{HI}}^{\mathrm{HI}}$, the energy consumption decreases with increase in $w_{\mathrm{LO}}$. Thus, weight factors make a big difference in reducing energy and enables us to set weightage to particular mode for energy saving. Hence, they are the main decision makers in our energy minimization problem. Furthermore, we observe that the energy consumption of the system decreases when $\alpha$ is increased. However, $\alpha$ is a constant fixed for any processor.

### 7.2.2 Static energy saving

We continue to show the static energy savings that we can achieve with our proposed algorithms. We calculate the static energy saving for different LO mode weights and $P_s$ in the range $0.1 - 0.3$, in increments of 0.05.

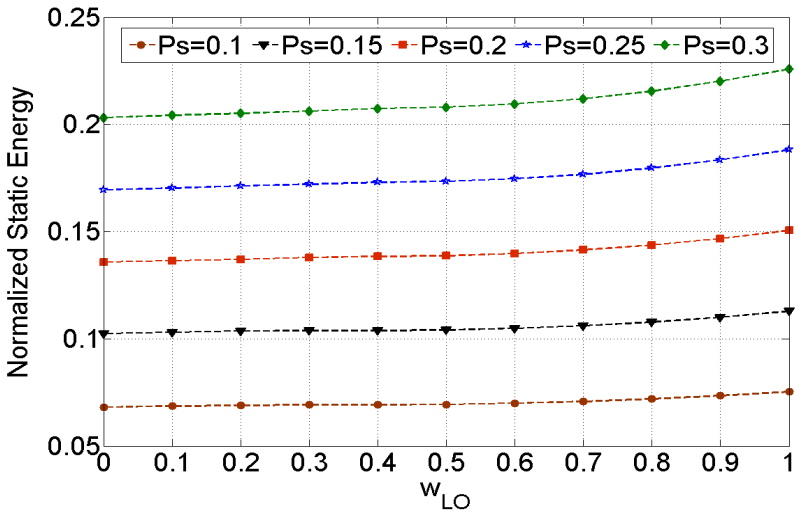


**Figure 7.2:** Static energy saving

The static energy of the system after applying our energy minimization algorithm is shown in Figure 7.2. It is evident from the plots that for all possibilities of $w_{\mathrm{LO}}$, static energy is saved. For instance, 32% of static energy is saved for the considered task set when $w_{\mathrm{LO}} = 0$ and $P_s = 0.3$. However, the amount of energy saved is not constant. Since HI mode utilization is less than that of LO mode in the system, there is more slack available for decreasing frequency in HI mode. Hence, when the system is more active in HI mode, both the static and dynamic energy consumptions are saved in a balanced way. Furthermore, as $w_{\mathrm{LO}}$ increases, the system is active in LO mode for excessive amount of time. Thus, it becomes more important now to save energy in LO mode. However, the LO mode utilization is comparatively higher than that of HI mode. Therefore, more weightage is given to save dynamic energy consumption as it is the cubic function of frequency and static energy is a linear function[1]. Thus, the static energy reduction decreases as $w_{\mathrm{LO}}$ increases in the considered task set.

[1] it should be noted that decreasing frequency increases static energy

### 7.2.3 Impact of $\frac{C(HI)}{C(LO)}$ Ratio

To analyze the impact of the extra workload, we consider $\lambda = \frac{C(HI)}{C(LO)} \geq 1$. We also set $w_{LO} = 0.2$ and $w_{HI} = 0.8$. When $\lambda = 1$, then there is no extra overhead as the HI criticality tasks would never exceed their LO criticality utilizations. With $\lambda > 1$, there is always a safety concern for HI criticality tasks as HI mode WCET increases.

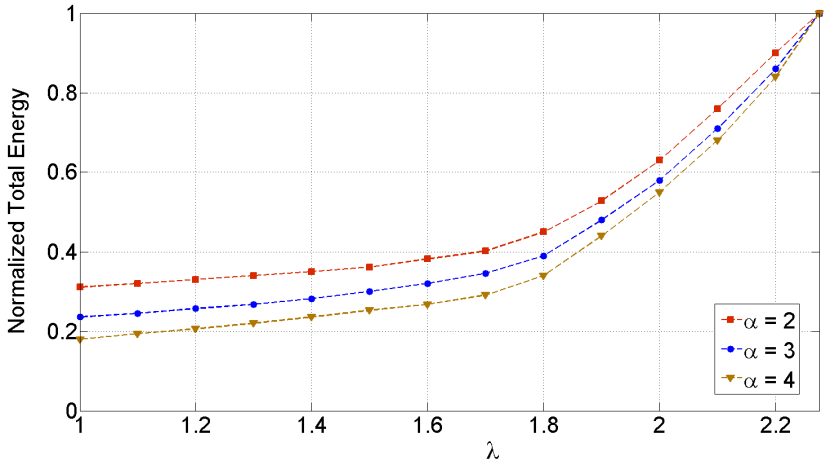


**Figure 7.3:** Impact of the ratio between $C(HI)$ and $C(LO)$ on energy minimization

Figure 7.3 shows the total energy at different values of $\lambda$, in the range $1 - 2.275$. As we see, with increased extra workload, the minimal expected energy increases. This is because, the HI mode utilization increases when $\lambda$ is increased. It should also be noted that $w_{HI} > w_{LO}$, which means that the system is active in HI mode for a longer duration. The trend with increasing energy stops at $\lambda = 2.275$, after which the system becomes infeasible even when we set the frequencies to $f_{max}$. Furthermore, we also observe that, as $\alpha$ increases, more energy is saved.

## 7.3 Evaluation on Multi-core

Experiments were conducted to study the effects of various factors on the energy consumption on multi-cores. The results obtained by simulating the random task sets on multi-core platform with existing bin packing methods and our proposed methods are shown in Figure 7.4. We observe in the plots that EM3 method saves energy considerably compared to the other mapping techniques (36% more saving than Baruah's and 24% more than Gu's method). The energy saving in IM3 is almost the same as that of EM3 (6% less than EM3). In EM3, the tasks are distributed on all active cores, whereas in IM3, LO and HI criticality tasks are distributed over $l$ and $h$ cores respectively. Hence, the distribution of LO and HI criticality tasks may not be balanced, thus loading $l$ or $h$ cores depending on the task utilization in particular mode. However, in case of EM3, which allows less

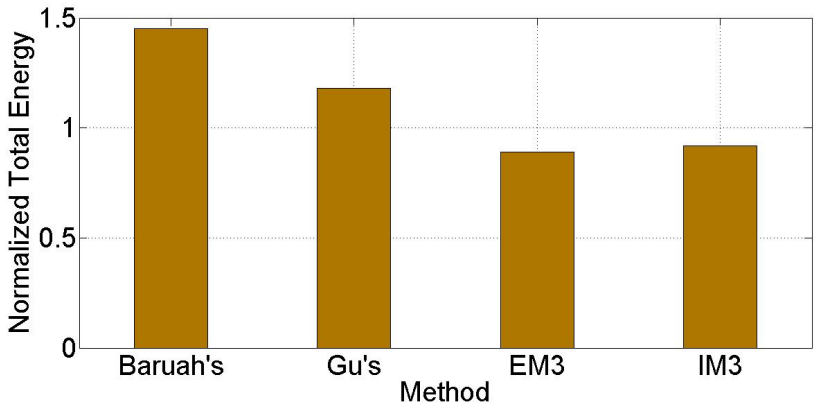


**Figure 7.4:** Comparison of normalized energy consumption with different mapping techniques

slack for decreasing execution frequencies, the maximum $\chi$ criticality utilization on each core is 3/4. In IM3, since the maximum LO criticality utilization allowed on $l$ cores, and the sum of LO and HI criticality utilization allowed on $h$ cores is 1, the execution frequencies can be adjusted more freely.

### 7.3.1 Impact of Number of Cores

The number of active cores selected for mapping plays an important role in minimizing the energy. We continue to present the impact of number of cores on energy minimization.

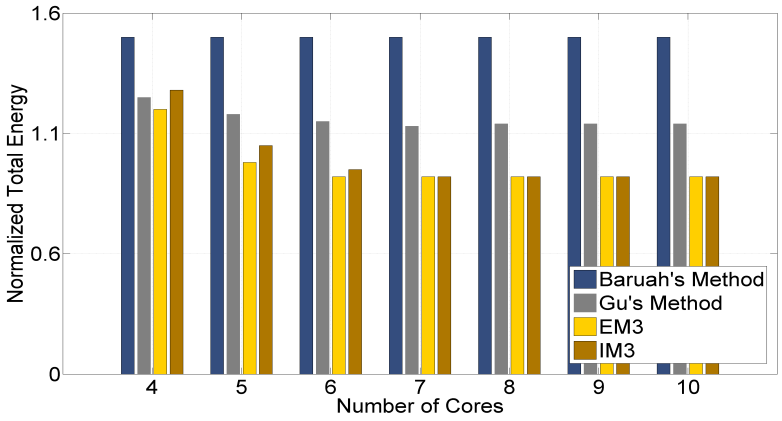


**Figure 7.5:** Comparison of normalized energy consumption with different number of cores

The outcome of our experiment with different number of cores (4 to 10) is shown in Figure 7.5. It is evident from the plots that the energy consumption in Baruah's method is constant in all the cases because it employs FF method to allocate both criticality level tasks. Since all the tasks can be scheduled using 4 available cores,

FF packs all the tasks in 4 cores, thus increasing the mode frequencies. Considering Gu's method, only LO criticality tasks are mapped using FF, whereas the HI criticality tasks are distributed on all the available cores as much as possible. This contributes in saving HI mode energy and therefore, the energy consumption is less compared to Baruah's method. On the contrary, EM3 method uses WF to map all the tasks, thus saving energy in both the modes depending on the weight factors. IM3 shows its worst performance when the number of cores available (cores = 4) is almost the same as that of total utilization of tasks (3). However, IM3 saves more energy if the number of cores is increased up to an extent (cores = 7) as the tasks can be evenly distributed. Further increase in the number of cores does not help because the optimal mode frequencies in all the cores are already at $f_{\min}$. When the number of cores is increased, the total static energy consumption of the system also increases.

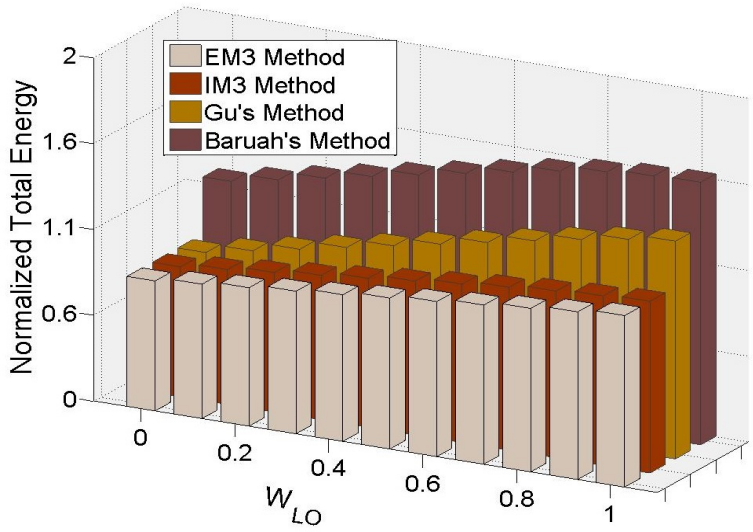


**Figure 7.6:** Impact of weight factors on energy minimization

### 7.3.2 Impact of Weight Factors

Weight factors are the main decision makers that provide the preference for energy saving in a certain mode. We execute the random task sets for different mode weights and the obtained results are shown in Figure 7.6. Our energy minimization objective (5.10) depends on the weighted sum of respective mode frequencies and their weights. Hence, for all possibilities of $w_{\mathrm{LO}} \in [0\,,\,3]$ in steps of 0.2, EM3 always shows the best performance in saving energy and IM3 follows it with a small difference. FF always consumes more energy being the worst in the considered mapping techniques. We notice in the plots that as $w_{\mathrm{LO}}$ increases, the total energy consumption also increases. This is because the LO mode utilization is higher in the generated task set. Hence, when LO mode gets more weight, the energy consumption depends on LO mode utilization of both LO and HI criticality tasks unlike HI mode energy where only $U_{\mathrm{HI}}^{\mathrm{HI}}$ and $w_{\mathrm{HI}}$ are considered. On the contrary, when $U_{\mathrm{HI}}^{\mathrm{HI}} \gg U_{\mathrm{HI}}^{\mathrm{LO}} + U_{\mathrm{LO}}^{\mathrm{LO}}$, then the total energy decreases as LO mode weight

increases.

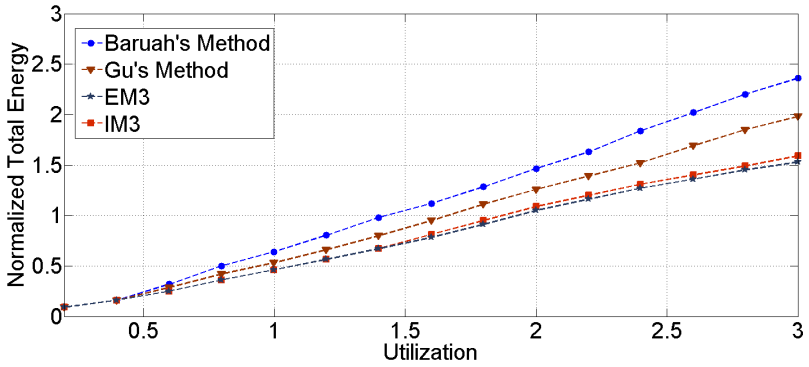


**Figure 7.7:** Impact of task utilization on energy minimization

### 7.3.3 Impact of Task Utilization

It is trivial that, increasing utilization increases the execution times of tasks, thus keeping the system to be active for a longer duration. In the experiments, the total utilization of tasks for each task set was increased from 0.2 to 3 in steps of 0.2. In all the cases, we fix $w_{\mathrm{LO}} = 0.8$ and $w_{\mathrm{HI}} = 0.2$ and four number of cores. When the utilization of the task set is very low, then no matter what bin packing method we use (FF or WF), all the tasks execute at $f_{min}$ in both the modes. This follows up to an extent (0.4 in our case) depending on the number of LO and HI criticality tasks in the system and also weights. Further increase in the utilization forces FF technique to fill all tasks in the same core as much as possible, whereas WF distributes them on all available cores, thus saving energy. We observed that, the energy saved by employing EM3 and IM3 was same when the total utilization was until 1.4. Beyond 1.4, the energy consumption was comparatively more in IM3 than EM3.

## 7.4 Validation

In all the simulations that we performed, we used our unicore heuristic to find the mode frequencies. To validate our optimal solution and heuristic, we used "NMinimize" function of Mathematica with our energy objective and scheduling constraints as input to the function. We compared the solution provided by NMinimize and our approaches for the FMS task set with the same parameters considered in Section 7.2.1. We observed that the optimal frequencies calculated by NMinimize and the one computed by our optimal solution are the same. However, in some cases we obtained a maximum difference of 0.00085%[2] between the optimal frequencies and the one calculated by our heuristic. This difference is negligible as even the latest processors do not support DVFS with such frequency variations.

[2]Total energy was calculated for all values of feasible $f_{\mathrm{HI}}^{\mathrm{HI}}$ with increments of 0.001

# Chapter 8

# Conclusion and Future Work

## 8.1 Conclusion

Embedded systems are making inroads into our daily lives. These systems need to execute various tasks that are having different levels of priorities and criticality within a single such system or in the larger integrated system of systems. Thus scheduling tasks on processors where they have different levels of criticality is an important issue. Further, ensuring timing safety and at the same time minimizing energy consumption in battery operated mixed-criticality systems have been a concern. This thesis considered mixed criticality systems that try to ensure safety as well as minimize energy. The real-time research community and industry have proposed many techniques to tackle the safety issues. However, there is not much work in the literature, concentrating on energy efficiency of these mixed criticality systems. This necessitates dedicated investigations into this class of mixed criticality problems owing to recent developments of technology.

In this thesis, we explore the energy minimization possibilities in these systems and provide an optimal solution for the problem. We also propose a low-complexity heuristic for reducing energy consumption in these systems and show that as high as 36% of the total energy can be saved. Unlike existing solutions, we minimize both static and dynamic energy consumption and also save energy in both LO and HI mode behaviors of the system. However, the main issues arise when saving energy. Several important questions in this regard were addressed in this report pertaining to saving energy.

In this work, we studied the factors that affect the energy consumption of mixed criticality tasks on unicore and multi-cores. We proposed weight factors to trade-off energy saving between different criticality levels. We apply the widely used energy conservation technique DVFS on the EDF-VD scheduling technique to save energy and also to guarantee timing safety. We demonstrate that, the problem is convex and solvers such as KKT conditions can be used to find the optimal solution. However, KKT conditions lead to high computational complexity and therefore, we

provide an/the optimal energy saving algorithm and also a low-complexity heuristic since the optimal computation is NP-hard.

We also extended the work to multi-cores by proposing two energy aware mapping techniques: one with mixing different criticality tasks together which are efficient for resource constraint devices; and another without mixing tasks with different criticality levels, thus providing complete isolation as preferred by the industry. We show that, using our isolated mapping technique, significant energy saving (6% less than non-isolated method in our simulations for a particular task set) can be achieved without jeopardizing system safety. We validate our proposed techniques by considering an FMS task set and also with extensive simulations. We also show that the solution provided by our heuristic is very close to the optimal solution and is negligible (maximum difference of 0.00085% between the optimal frequencies and the frequencies calculated by our heuristic). We believe that the proposed energy saving technique is implementable (with few other considerations such as overheads introduced by DVFS) as the modern processors and micro-controllers are facilitated with DVFS capability.

## 8.2 Future Work

Possibly this is the beginning of new class of scheduling solutions focused jointly on energy and time criticality. The present work, though has shed light on many important issues and has provided insights, much more investigations are still needed. The work here ccan be improved in many ways. We enlist some of them here:

- We considered EDF-VD scheduling technique in our energy minimizing problem. However, the optimal solution and heuristic can be generalized independent of scheduling techniques chosen.
- We consider only two criticality levels - LO and HI in our problem. The solution that we provided can be extended to multi-level mixed-criticality systems where tasks with more than two criticality levels are considered.
- Further improvement in minimizing energy in HI mode can be achieved once the jobs executing in both LO and HI mode completes their execution (in this case, the HI mode utilization can be stretched to 1 using DVFS as the LO criticality tasks are abandoned).
- The extra overheads added because of DVFS need to be handled. The overheads can be listed as follows: (i) Frequency scaling timing overhead (DVFS adds extra timing overhead in the processor to change the execution frequencies which can be added to WCET of tasks, but not straight forward), (ii) dynamic power reduction overheads (dynamic power does not change abruptly but gradually changes when DVFS is applied) and (iii) sleep mode switching overhead. Extra time is taken to switch the processor from active mode to sleep mode and vice versa. Moreover, the power change is not abrupt.

The above issues if considered holistically, in our opinion, will lead to important strides in embedded systems of the future.

# Chapter 9

# Publications

1. Sujay Narayana, Pengcheng Huang, R Venkatesha Prasad, Lothar Thiele and Georgia Giannopoulou. Energy and Timliness – Energy Minimization for Mixed-Criticality Systems on Multi-cores. Submitted to 22nd IEEE Real-Time Embedded Technology & Applications Symposium (RTAS 2016).

2. Sujay Narayana, Pengcheng Huang, R Venkatesha Prasad, Lothar Thiele and Georgia Giannopoulou. Energy Minimization for Isolated Mixed-Criticality Systems on Multi-cores. To be Submitted at ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'16).

# Appendix A

# Proofs

## A.1 Schedulability conditions for EDF - VD with DVFS strategy

We consider the same task model and follow the same procedure as in [32] and present the modified schedulability conditions as follows:

**Theorem 7.** All the tasks are schedulable in their LO criticality behaviors of $\tau$ if

$$x \geq \frac{\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{LO}}}}{1 - \frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}}} \tag{A.1}$$

*Proof.* If EDF has to schedule all the tasks, then the total lo criticality utilization has to be less than 1. The deadlines of all HI criticality tasks are shortened by a multiplication factor $x$. From the utilization bound result of EDF,

$$U_{\mathrm{LO}}^{\mathrm{LO}} + \frac{U_{\mathrm{HI}}^{\mathrm{LO}}}{x} \leq 1$$

By applying DVFS strategy on tasks in LO mode, the execution times of tasks are increased as described in Corollary 1. Hence we can conclude that

$$\frac{U_{\mathrm{LO}}^{\mathrm{LO}}}{f_{\mathrm{LO}}^{\mathrm{LO}}} + \frac{U_{\mathrm{HI}}^{\mathrm{LO}}}{x f_{\mathrm{HI}}^{\mathrm{LO}}} \leq 1$$

$$\Leftrightarrow x \geq \frac{\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{LO}}}}{1 - \frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}}}$$

□

**Theorem 8.** All the tasks are schedulable in their LO criticality behaviors of $\tau$ if

$$x\frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}} + \frac{(f_{\mathrm{HI}}^{\mathrm{HI}} - f_{\mathrm{HI}}^{\mathrm{LO}})}{f_{\mathrm{HI}}^{\mathrm{LO}}}\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{HI}}} + \frac{U_{\mathrm{HI}}^{\mathrm{HI}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{HI}}} \leq 1$$

*Proof.* Even though $\tau$ satisfies condition A.1 in LO criticality behaviours, all the tasks may not meet their deadlines in HI criticality behaviors of $\tau$. Let us consider a minimal instance of jobs $I$ that is released by the task set $\tau$ where only one task misses it's deadline. Since all the tasks satisfy condition A.1, the deadline miss has to be from a HI criticality task. Let $t^*$ be the time instant at which a HI criticality task enters into HI criticality behavior for the first time in the instance $I$. Let the earliest release of all the jobs happens at instance 0 and $t_f$ be the time instant at which the deadline is missed by a job. Hence all the tasks in $I$ have deadlines less than $t_f$; and executes completely except the one that misses it's deadline. Let $a_1$ be the release time of a job in $I$ with the earliest release time of all the jobs that execute in $[t^*, t_f]$ and let $d_1$ be it's deadline. For each $i$, $1 \leq i \leq n$, let $\eta_i$ be the amount of execution over the interval $[0, t_f]$. For each $i$, $1 \leq i \leq n$, let $u_i(\chi)$ be $\frac{C_i(\chi)}{T_i}$, the utilization of $\chi$ criticality task.

**Fact 1.** All jobs executing in $[t^*, t_f]$ have deadline $\leq t_f$.

*Proof.* This follows from [32] □

**Fact 2.** Any LO criticality task $\tau_i$ has

$$\eta_i \leq \frac{u_i(\mathrm{LO})}{f_i^{\mathrm{LO}}}(a_1 + x(t_f - a_1)) \tag{A.2}$$

*Proof.* The proof provided in [32] still holds in this fact as we only change the computation time of tasks by changing the execution frequency. □

**Fact 3.** Any HI criticality task $\tau_i$ has

$$\eta_i \leq \frac{a_1}{x}\frac{u_i(\mathrm{LO})}{f_i^{\mathrm{LO}}} + (t_f - a_1)\left(\frac{(f_i^{\mathrm{HI}} - f_i^{\mathrm{LO}})}{f_i^{\mathrm{HI}}}\frac{u_i(\mathrm{LO})}{f_i^{\mathrm{LO}}} + \frac{u_i(\mathrm{HI})}{f_i^{\mathrm{HI}}}\right) \tag{A.3}$$

*Proof.* For any HI criticality task, we consider two cases where $\tau_i$ does not release any job at or after release time $< a_1$ and when it does.

*Case 1 : $\tau_i$ does not release any job at or after $a_1$*

We know that $a_1$ is the release time of job that executes in $[t^*, t_f]$ with the earliest release time in $I$. Hence any HI criticality job released before $a_1$ always executes in it's LO criticality behavior. In other words, the job should have completed it's execution before $t^*$. Also, any job with release time $a_i$ in $I$ has the modified absolute deadline $\leq a_i + x(t_f - a_i)$. This is because, since $x \leq 1$, and is same for all the tasks in $I$, for instance, at $x = 1$ (maximum value), the modified deadline $\leq t_f$. This is true for all $i$ in $1 \leq i \leq n$. Hence it is also true that all jobs have their absolute modified deadlines $\leq a_1 + x(t_f - a_1)$. If not, consider some job with

a modified deadline $\geq a_1 + x(t_f - a_1)$ and let $t^{'}$ be the latest instant at which this job executes. From the assumption made for $I$, it is evident that all the jobs having release times greater than $t^{'}$ also misses the deadline.

Since the modified deadline of jobs released before $a_1$ has modified deadline $\leq a_1 + x(t_f - a_1)$, their actual deadlines are $\leq \frac{a_1}{x} + x(t_f - a_1)$

Therefore, their cumulative execution requirement

$$\eta_i \leq \frac{a_1}{x}\frac{u_i(\text{LO})}{f_i^{\text{LO}}} + (t_f - a_1)\frac{u_i(\text{LO})}{f_i^{\text{LO}}} \tag{A.4}$$

*Case 2 : $\tau_i$ releases a job at or after $a_1$*

Let $J_i$ be the first job with release time $a_i$ that is released at or after $a_1$.

In this case, we can consider two scenarios:

*Scenario 1 : $J_i$ completes it's partial execution before $t^*$ and partial after $t^*$.*

In this scenario, the cumulative execution demand of all jobs of $t_i$ is at most

$$\eta_i \leq a_i\frac{u_i(\text{LO})}{f_i^{\text{LO}}} + (t_f - a_i)\left(\frac{u_i(\text{LO})}{f_i^{\text{LO}}} + \frac{u_i(\text{HI}) - u_i(\text{LO})}{f_i^{\text{HI}}}\right) \tag{A.5}$$

*Scenario 2 : $J_i$ starts and completes it's execution after $t^*$*

In this scenario, the cumulative execution demand of all jobs of $t_i$ is atmost

$$\eta_i \leq a_i\frac{u_i(\text{LO})}{f_i^{\text{LO}}} + (t_f - a_i)\frac{u_i(\text{HI})}{f_i^{\text{HI}}} \tag{A.6}$$

□

Since the execution demand presented in A.5 is higher than that in A.4 and A.6, we can discard conditions A.4 and A.6.

Hence the execution demand for any HI criticality task has the execution demand

$$\eta_i \leq \frac{a_1}{x}\frac{u_i(\text{LO})}{f_i^{\text{LO}}} + (t_f - a_1)\left(\frac{(f_i^{\text{HI}} - f_i^{\text{LO}})}{f_i^{\text{HI}}}\frac{u_i(\text{LO})}{f_i^{\text{LO}}} + \frac{u_i(\text{HI})}{f_i^{\text{HI}}}\right) \tag{A.7}$$

Summing up the cumulative demand of all tasks over $[0, t_f]$:

$$\sum_{\chi_i=\text{LO}} \eta_i + \sum_{\chi_i=\text{HI}} \eta_i$$

$$\leq \sum_{\chi_i=\text{LO}} \frac{u_i(\text{LO})}{f_i^{\text{LO}}}(a_1 + x(t_f - a_1)) + \sum_{\chi_i=\text{HI}} \frac{a_1}{x}\frac{u_i(\text{LO})}{f_i^{\text{LO}}}$$

$$+(t_f - a_1)\left(\frac{(f_i^{\text{HI}} - f_{\text{HI}}^{\text{LO}})}{f_i^{\text{HI}}}\frac{u_i(\text{LO})}{f_i^{\text{LO}}} + \frac{u_i(\text{HI})}{f_i^{\text{HI}}}\right)$$

$$\leq \frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}}\left(a_1 + x(t_f - a_1)\right) + \frac{a_1}{x}\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{LO}}}$$
$$+(t_f - a_1)\left(\frac{(f_{\mathrm{HI}}^{\mathrm{HI}} - f_{\mathrm{HI}}^{\mathrm{LO}})}{f_{\mathrm{HI}}^{\mathrm{HI}}}\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{LO}}}(\tau + \frac{U_{\mathrm{HI}}^{\mathrm{HI}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{HI}}}\right)$$

(Using condition A.1)

$$\leq a_1 + (t_f - a_1)\left(x\frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}} + \frac{(f_{\mathrm{HI}}^{\mathrm{HI}} - f_{\mathrm{HI}}^{\mathrm{LO}})}{f_{\mathrm{HI}}^{\mathrm{HI}}}\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{LO}}} + \frac{U_{\mathrm{HI}}^{\mathrm{HI}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{HI}}}\right)$$

It follows from in feasibility of this instance that:

$$a_1 + (t_f - a_1)\left(x\frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}} + \frac{(f_{\mathrm{HI}}^{\mathrm{HI}} - f_{\mathrm{HI}}^{\mathrm{LO}})}{f_{\mathrm{HI}}^{\mathrm{HI}}}\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{LO}}} + \frac{U_{\mathrm{HI}}^{\mathrm{HI}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{HI}}}\right) > t_f$$

Taking contrapositive,

$$x\frac{U_{\mathrm{LO}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{LO}}^{\mathrm{LO}}} + \frac{(f_{\mathrm{HI}}^{\mathrm{HI}} - f_{\mathrm{HI}}^{\mathrm{LO}})}{f_{\mathrm{HI}}^{\mathrm{HI}}}\frac{U_{\mathrm{HI}}^{\mathrm{LO}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{LO}}} + \frac{U_{\mathrm{HI}}^{\mathrm{HI}}(\tau)}{f_{\mathrm{HI}}^{\mathrm{HI}}} \leq 1$$

□

# Bibliography


[1] R. D. A. Burns, "Mixed criticality systems - a review." www-users.cs.york.ac.uk/burns/review.pdf, June 2014. Accessed: 14-05-2015.

[2] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie, "Scheduling real-time mixed-criticality jobs," *Computers, IEEE Transactions on*, vol. 61, pp. 1140–1152, Aug 2012.

[3] G. Giannopoulou, N. Stoimenov, P. Huang, and L. Thiele, "Scheduling of mixed-criticality applications on resource-sharing multicore systems," in *Embedded Software (EMSOFT), 2013 Proceedings of the International Conference on*, pp. 1–15, Sept 2013.

[4] S. Schreiner, K. Gruttner, S. Rosinger, and A. Rettberg, "Autonomous flight control meets custom payload processing: A mixed-critical avionics architecture approach for civilian uavs," in *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2014 IEEE 17th International Symposium on*, pp. 348–357, June 2014.

[5] R. Urzi, "A research agenda for mixed-criticality systems." http://www.cse.wustl.edu/~cdgill/CPSWEEK09_MCAR/RBO-09-130%20Joint%20MCAR%20White%20Paper%20PA%20approved.pdf.

[6] P. Prisaznuk, "Arinc 653 role in integrated modular avionics (ima)," in *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*, pp. 1.E.5–1–1.E.5–10, Oct 2008.

[7] ARINC, "Arinc 653-1 avionics application software standard interface." http://www.arinc.com/,Tech.Rep., June 2003. Accessed: 14-05-2015.

[8] H. Stallbaum and M. Rzepka, "Toward do-178b-compliant test models," in *Model-Driven Engineering, Verification, and Validation (MoDeVVa), 2010 Workshop on*, pp. 25–30, Oct 2010.

[9] AUTOSAR, "Autosar specification v4.2." OfficialwebsiteoftheAUTOSAR: http://www.autosar.org/specifications/release-42/, August 2015. Accessed: 01-08-2015.

[10] G. Buttazzo, M. Bertogna, and G. Yao, "Limited preemptive scheduling for real-time systems. a survey," *Industrial Informatics, IEEE Transactions on*, vol. 9, pp. 3–15, Feb 2013.

[11] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Comput. Surv.*, vol. 43, pp. 35:1–35:44, Oct. 2011.

[12] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, pp. 46–61, Jan. 1973.

[13] M. Dertouzos, "Control robotics :the procedural control of physical processors," in *IFIP Congress*, 1974.

[14] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *Solid-State Circuits, IEEE Journal of*, vol. 27, pp. 473–484, Apr 1992.

[15] J.-J. Chen and C.-F. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms," in *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, RTCSA '07, (Washington, DC, USA), pp. 28–38, IEEE Computer Society, 2007.

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

[17] R. Jejurikar and R. Gupta, "Procrastination scheduling in fixed priority real-time systems," *SIGPLAN Not.*, vol. 39, pp. 57–66, June 2004.

[18] Y.-H. Lee, K. Reddy, and C. Krishna, "Scheduling techniques for reducing leakage power in hard real-time systems," in *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on*, pp. 105–112, July 2003.

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

[20] H. Hanson, S. Keckler, and D. Burger, "Static energy reduction techniques in microprocessor caches," in *The University of Texas at Austin*, 2003.

[21] C. Stangaciu, M. Micea, and V. Cretu, "Energy efficiency in real-time systems: A brief overview," in *Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium on*, pp. 275–280, May 2013.

[22] C. Guo, "Empirical study of energy minimization issues for mixed-criticality systems with reliability constraints," in *The First Workshop on Low-Power Dependable Computing (LPDC)*, 2014.

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

[24] Baruah, S. and Chattopadhyay, B. and Li, H. and Shin, I., "Mixed-criticality scheduling on multiprocessors," *Real-Time Systems*, vol. 50, no. 1, pp. 142–177, 2014.

[25] M. Bagheri and G. Jervan, "Fault-tolerant scheduling of mixed-critical applications on multi-processor platforms," in *Embedded and Ubiquitous Computing (EUC), 2014 12th IEEE International Conference on*, pp. 25–32, Aug 2014.

[26] J. M. López, J. L. Díaz, and D. F. García, "Utilization bounds for edf scheduling on real-time multiprocessor systems," *Real-Time Syst.*, vol. 28, pp. 39–68, Oct. 2004.

[27] A. Burchard, J. Liebeherr, Y. Oh, and S. Son, "New strategies for assigning real-time tasks to multiprocessor systems," *Computers, IEEE Transactions on*, vol. 44, pp. 1429–1442, Dec 1995.

[28] J. Liebeherr, A. Burchard, Y. Oh, and S. H. Son, "New strategies for assigning real-time tasks to multiprocessor systems," *IEEE Trans. Comput.*, vol. 44, pp. 1429–1442, Dec. 1995.

[29] H. Li and S. Baruah, "Outstanding paper award: Global mixed-criticality scheduling on multiprocessors," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, pp. 166–175, July 2012.

[30] O. Kelly, H. Aydin, and B. Zhao, "On partitioned scheduling of fixed-priority mixed-criticality task sets," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pp. 1051–1059, Nov 2011.

[31] IEC, "Iec 61508 standard." OfficialwebsiteoftheIEC:http://www.iec.ch/about/brochures/pdf/technology/functional_safety.pdf, August 2015. Accessed: 01-08-2015.

[32] 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*, pp. 145–154, July 2012.

[33] N. Guan, P. Ekberg, M. Stigge, and W. Yi, "Effective and efficient scheduling of certifiable mixed-criticality sporadic task systems," in *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pp. 13–23, Nov 2011.

[34] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pp. 239–243, Dec 2007.

[35] S. Baruah and S. Vestal, "Schedulability analysis of sporadic tasks with multiple criticality specifications," in *Real-Time Systems, 2008. ECRTS '08. Euromicro Conference on*, pp. 147–155, July 2008.

[36] S. Baruah, H. Li, and L. Stougie, "Towards the design of certifiable mixed-criticality systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, pp. 13–22, April 2010.

[37] T. Park and S. Kim, "Dynamic scheduling algorithm and its schedulability analysis for certifiable dual-criticality systems," in *Proceedings of the Ninth ACM International Conference on Embedded Software*, EMSOFT '11, (New York, NY, USA), pp. 253–262, ACM, 2011.

[38] R. Pellizzoni, P. Meredith, M.-Y. Nam, M. Sun, M. Caccamo, and L. Sha, "Handling mixed-criticality in soc-based real-time embedded systems," in *Proceedings of the Seventh ACM International Conference on Embedded Software*, EMSOFT '09, (New York, NY, USA), pp. 235–244, ACM, 2009.

[39] A. Sangiovanni-Vincentelli, "Quo vadis, sld? reasoning about the trends and challenges of system level design," *Proceedings of the IEEE*, vol. 95, pp. 467–506, March 2007.

[40] S. Petters, M. Lawitzky, R. Heffernan, and K. Elphinstone, "Towards real multi-criticality scheduling," in *Embedded and Real-Time Computing Systems and Applications, 2009. RTCSA '09. 15th IEEE International Conference on*, pp. 155–164, Aug 2009.

[41] P. Ekberg and W. Yi, "Outstanding paper award: Bounding and shaping the demand of mixed-criticality sporadic tasks," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, pp. 135–144, July 2012.

[42] A. Srinivasan, P. Holman, J. Anderson, and S. Baruah, "The case for fair multiprocessor scheduling," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pp. 10 pp.–, April 2003.

[43] F. Kong, W. Yi, and Q. Deng, "Energy-efficient scheduling of real-time tasks on cluster-based multicores," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pp. 1–6, March 2011.

[44] S. Funk, J. Goossens, and S. Baruah, "On-line scheduling on uniform multiprocessors," in *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*, pp. 183–192, Dec 2001.

[45] S. Baruah and N. Fisher, "The partitioned scheduling of sporadic real-time tasks on multiprocessor platforms," in *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*, pp. 346–353, June 2005.

[46] J. Anderson, S. Baruah, and B. Brandenburg, "Multicore operating-system support for mixed criticality," in *Workshop on Mixed Criticality: Roadmap to Evolving UAV Certification, San Francisco*, 2009.

[47] S. Baruah, "Optimal utilization bounds for the fixed-priority scheduling of periodic task systems on identical multiprocessors," *Computers, IEEE Transactions on*, vol. 53, pp. 781–784, June 2004.

[48] A. Kritikakou, O. Baldellon, C. Pagetti, C. Rochange, M. Roy, and F. Vargas, "Monitoring on-line timing information to support mixed-critical workloads," *RTSS, WiP*, 2013.

[49] C. Gu, N. Guan, Q. Deng, and W. Yi, "Partitioned mixed-criticality scheduling on multiprocessor platforms," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pp. 1–6, IEEE, 2014.

[50] Z. Guo and S. Baruah, "Mixed-criticality scheduling upon varying-speed multiprocessors," in *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*, pp. 237–244, Aug 2014.

[51] G. Liu, Y. Lu, S. Wang, and Z. Gu, "Partitioned multiprocessor scheduling of mixed-criticality parallel jobs," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*, pp. 1–10, Aug 2014.

[52] M. Bagheri and G. Jervan, "Fault-tolerant scheduling of mixed-critical applications on multi-processor platforms," in *Embedded and Ubiquitous Computing (EUC), 2014 12th IEEE International Conference on*, pp. 25–32, Aug 2014.

[53] 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)*, pp. 1–6, 2013.

[54] M. Volp, M. Hahnel, and A. Lackorzynski, "Has energy surpassed timeliness? scheduling energy-constrained mixed-criticality systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2014 IEEE 20th*, pp. 275–284, IEEE, 2014.

[55] 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, 2007. RTCSA 2007. 13th IEEE International Conference on*, pp. 28–38, Aug 2007.

[56] S. Pagani and J.-J. Chen, "Energy efficiency analysis for the single frequency approximation (sfa) scheme," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2013 IEEE 19th International Conference on*, pp. 82–91, Aug 2013.

[57] 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), 2011 14th Euromicro Conference on*, pp. 117–124, Aug 2011.

[58] S. Pagani and J.-J. Chen, "Energy efficient task partitioning based on the single frequency approximation scheme," in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, pp. 308–318, Dec 2013.

[59] A. Kandhalu, J. Kim, K. Lakshmanan, and R. Rajkumar, "Energy-aware partitioned fixed-priority scheduling for chip multi-processors," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2011 IEEE 17th International Conference on*, vol. 1, pp. 93–102, Aug 2011.

[60] Boyd, S. and Vandenberghe, L., "Convex optimization, cambridge university press, isbn: 9780521833783," in *Cambridge University Press*, 2004.

[61] E. Seo, J. Jeong, S. Park, and J. Lee, "Energy efficient scheduling of real-time tasks on multicore processors," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, pp. 1540–1552, Nov 2008.

[62] H. Kuhn and A. Tucker, "Nonlinear programming," in *Second Berkeley Symposium on Mathematical Statistics and Probability*, pp. 481–492, University of California Press, Berkeley and Los Angeles, 1951.

[63] H. Nakayama, H. Sayama, and Y. Sawaragi, "A generalized lagrangian function and multiplier method," *Journal of Optimization Theory and Applications*, vol. 17, pp. 211–227, 1975.