AN ENERGY-EFFICIENT SEMI-PARTITIONED APPROACH FOR HARD REAL-TIME SYSTEMS WITH VOLTAGE AND FREQUENCY ISLANDS

by

Jesse W. Patterson

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Engineering

Approved:

Dr. Thidapat Chantem Major Professor	Dr. Koushik Chakraborty Committee Member
Dr. Ryan Gerdes Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2016

Abstract

An Energy-Efficient Semi-Partitioned Approach for Hard Real-Time Systems with Voltage and Frequency Islands

by

Jesse W. Patterson, Master of Science

Utah State University, 2016

Major Professor: Dr. Thidapat Chantem
Department: Electrical and Computer Engineering

The shift from uniprocessor to multi-core architectures has made it difficult to design predictable hard real-time systems (HRTS) since guaranteeing deadlines while achieving high processor utilization remains a major challenge. In addition, due to increasing demands, energy efficiency has become an important design metric in HRTS. To obtain energy savings, most multi-core systems use dynamic voltage and frequency scaling (DVFS) to reduce dynamic power consumption when the system is underloaded. However, in many multi-core systems, DVFS is implemented using voltage and frequency islands (VFI), implying that individual cores cannot independently select their voltage and frequency (v/f) pairs, thus resulting in less energy savings when existing energy-aware task assignment and scheduling techniques are used. In this thesis, we present an analysis of the increase in energy consumption in the presence of VFI. Further, we propose a semi-partitioned approach called EDF-hv to reduce the energy consumption of HRTS on multi-core systems with VFI. Simulation results revealed that when workload imbalance among the cores is sufficiently high, EDF-hv can reduce system energy consumption by 15.9% on average.

(75 pages)

Public Abstract

An Energy-Efficient Semi-Partitioned Approach for Hard Real-Time Systems with Voltage and Frequency Islands

by

Jesse W. Patterson, Master of Science

Utah State University, 2016

Major Professor: Dr. Thidapat Chantem
Department: Electrical and Computer Engineering

Embedded devices are computer systems with a dedicated function, such as kitchen appliances, electronic toys, phones, et cetera. Embedded devices are commonplace, and many are expected to respond to users in real-time. Such devices are often categorized as real-time embedded systems as they differ from other embedded systems in that work performed by a real-time embedded system must be scheduled such that timeliness can be provided to the user (e.g. when someone is talking on a phone, the phone needs to be able to schedule the sending and receiving of data in a timely manner such that the conversation is intelligible). Further, for real-time embedded systems that perform life-critical functions (e.g. medical equipment, avionics instruments, military devices, et cetera), scheduling must be able to guarantee that all of the work can be performed in the required amount of time. These systems are often referred to as Hard Real-Time Systems (HRTS).

Due to the increase in popularity of multi-core processors, many real-time embedded systems have transitioned from a single core processor architecture to a multi-core architecture. Unfortunately, similar to how scheduling tasking for a group of people is more complicated than scheduling tasking for just one person, the transition to a multi-core architecture has proven to be a difficult problem for scheduling.

Further, energy efficiency is a major design metric in many HRTS, and to obtain energy savings, most embedded systems use dynamic voltage and frequency scaling (DVFS). With DVFS, the processing cores on the multi-core processor can increase and decrease its voltage and frequency to change its energy consumption. Although reducing the voltage and frequency also reduces the amount of work that can be accomplished, typically the relationship between work and energy consumption is quadratic such that reducing the frequency by two reduces energy consumption by four. As such when a processing core only has a little bit of work to be completed, the voltage and frequency can be reduced to allow the core to operate slower and consume significantly less energy. Conversely, when the workload is high, the core can increase its voltage and frequency to be able to meet its schedule. Unfortunately, as clock frequencies have increased and transistors have gotten smaller, being able to implement DVFS in a multi-core processor has gotten more difficult. To mitigate this difficulty, multi-core processors often use a voltage and frequency island (VFI) to implement DVFS, where multiple processing cores reside on a single VFI. Although a VFI does simplify the implementation of DVFS on a multi-core processor, it also causes that all of the cores on the VFI must operate at the same voltage and frequency; which further complicates scheduling.

To be able to guarantee scheduling for HRTS in an energy efficient manner in the presence of VFI, we propose EDF-hv. Simulation results revealed that, when applied correctly, EDF-hv can reduce system energy consumption by 15.9% on average and up to 61.6%.

I dedicate this work to Science.

Acknowledgments

We gratefully acknowledge funding and other support from the Naval Air Warfare Center Weapons Division for this work.

Contents

Page

Abstract . . . . . **iii**

Public Abstract . . . . . **iv**

Acknowledgments . . . . . **vii**

List of Tables . . . . . **x**

List of Figures . . . . . **xi**

Acronyms . . . . . **xiii**

1 Introduction **1**

2 Related Works **4**
2.1 Power Management 4
2.2 Semi-partitioning 5
2.3 Earliest Deadline First Optimal Semi-Partitioned Scheduling (EDF-os) . . . 6

3 System Model **8**
3.1 System Model 8
3.2 Energy Considerations 10

4 Quantifying Imbalance and VFI Cost **13**
4.1 Imbalance 13
4.2 VFI Cost 14
4.2.1 Worst-Case VFI Cost 15
4.2.2 Best-Case VFI Cost 16

5 EDF-hv **17**
5.1 Partitioning for a Load-Balance 17
5.2 Scheduling 20
5.3 Guaranteeing Deadlines 21

6 Simulation Results **26**
6.1 VFI Cost 26
6.1.1 Simulation Setup 26
6.1.2 Results 28
6.2 EDF-hv 32
6.2.1 Simulation Setup 32
6.2.2 Performance of EDF-hv 33
6.2.3 TI C66x DSP Case-Study 36

7 Conclusion and Future Work **40**

References **41**

Appendices **44**

A Handling Oversized Tasks 45

A.1 Bounds with Oversized Tasks 46

A.2 Methods to Handle Oversized Tasks 51

A.3 Best Method for Handling Oversized Tasks 61

List of Tables

Table Page

5.1 Example periodic task set (left) with EDF-os partition onto a system with two cores (right). 21

5.2 Utilization of cores for the task set in Table 5.1 when migrating tasks are and are not present. 21

6.1 Reduction in imbalance with the correlated reduction in VFI cost for the given system utilization based on the average of two, four, and eight cores per VFI. The average over the system utilizations shows a nearly one-to-one correlation between imbalance and VFI cost. 31

A.1 Comparison of methods to handle oversized task using the metrics described in Section A.2. 62

List of Figures

Figure Page

3.1 Example of the limitations on static power management imposed by VFI. Consider a processor with sixteen cores on four homogeneous VFI (i.e. each VFI has four cores). Suppose at some time, the task set requires only six cores, shown in orange, to guarantee deadlines. With four cores per VFI, to ensure six cores are available, two VFI must be powered on. Unfortunately, powering on two VFI means the system has eight operating cores consuming energy. 11

4.1 The maximum VFI cost (Equation 4.7) as a function of the number of cores (M) by the minimum frequency (f_1) that the VFI can reduce to (as a percentage of the maximum frequency, f_{K}) assuming a pure CMOS circuit (i.e., $w_k \propto f_k^3$). 16

5.1 Partitioning the task set, η (top left), where each task is represented by its utilization, onto the set of cores, P (top right), using the WFD (or EDF-os; bottom left) and EDF-hv (bottom right). EDF-hv restricts the available capacity of each core to Φ. 18

6.1 Average VFI cost (left) and imbalance (right) for two (top), four (middle), and eight (bottom) cores, as a function of the average number of tasks per core (i.e., N/M) and system utilization. 29

6.2 Combined average VFI cost (left) and imbalance (right) (i.e., combined average of two, four, and eight cores per VFI), as a function of the average number of tasks per core (i.e., N/M) and system utilization. 30

6.3 Combined average VFI cost (left) and imbalance (right) for the average number of tasks per core (N/M) by the system utilization. 30

6.4 Combined average VFI cost (left) and imbalance (right) for the system utilization by the average number of tasks per core (N/M). 30

6.5 Reduction in energy consumption (i.e. performance) of EDF-hv compared to WFD versus the imbalance of WFD for two- and four-core systems with one VFI and task sets with 1.5, 2.0, 2.5 and 3.0 average number of tasks per core (i.e., N/M). 34

6.6 Reduction in energy consumption (i.e. performance) of EDF-hv compared to WFD versus the imbalance of WFD for four- and eight-core systems with one VFI and task sets with 1.5, 2.0, 2.5 and 3.0 average number of tasks per core (i.e., N/M). 35

6.7 Reduction in energy consumption (i.e. performance) of EDF-hv compared to WFD versus the imbalance of WFD for a four-core system with one VFI using the frequency profile of the TI C66x DSP and task sets with 1.5, 2.0, 2.5 and 3.0 average number of tasks per core (i.e., N/M). 37

A.1 An example of cascading oversized tasks. The original task set, η (top; where each task is represented by its utilization), has only one oversized task, τ_1. When τ_1 is pre-partitioned, τ_2 becomes an oversized task in $\eta^{'}$, even though τ_2 was not an oversized task in η. Further, when τ_2 is pre-partitioned, τ_3 becomes oversized in $\eta^{''}$. At this point, the system has $(\mathrm{M} - 1) = (4 - 1) = 3$ oversized task (i.e. the maximum number of oversized tasks in a 4 core system) in the system. Thus, $\eta^{'''}$ does not have oversized tasks, so the remaining tasks are assigned to the remaining core (bottom). 53

A.2 An example of horizontal multi-partitioning (HMP). The original task set, η (top; where each task's utilization is shown), has one oversized task, τ_1. As such, η is divided into two subsets, η^{A} and η^{B}, as well as P is also divided into two subsets, P^{A} and P^{B} (middle). Although the resulting subsets no longer have any oversized task, the system is clearly not load-balanced (bottom). . 60

Acronyms

CMOS	Complementary Metal-Oxide Semiconductor
DSP	Digital Signal Processor
DVFS	Dynamic Voltage and Frequency Scaling
EDF	Earliest Deadline First
EDF-fm	Earliest Deadline First Fixed or Migrating
EDF-hv	Earliest Deadline First for Hard Real-Time Systems with Voltage and Frequency Islands
EDF-os	Earliest Deadline First Optimal Semi-Partitioned Scheduling
HRTS	Hard Real-Time Systems
HMP	Horizontal Multi-Partitioning
LADVS	Look-Ahead Dynamic Voltage Scaling
P-Fair	Proportionate Fairness
TI	Texas Instrument
v/f	Voltage and Frequency
VFI	Voltage and Frequency Islands
VMP	Vertical Multi-Partitioning
WCET	Worst-Case Execution Time
WFD	Worst-Fit Decreasing

Chapter 1

Introduction

Scheduling tasks is an intrinsic design consideration for real-time systems that obliges attaining the desired confidence level that tasks will complete on time, or at least complete within an acceptable tardiness. For the class of hard real-time systems (HRTS), scheduling must guarantee that all deadlines will complete on time. A wealth of knowledge exists for how to schedule HRTS in a single-core environment [1–5], and the Earliest-Deadline-First (EDF) scheduling algorithm has been shown to be optimal [1]. However, guaranteeing deadlines in an environment with multiple cores [6–12] is still an area of on-going research. In fact, for multi-core HRTS, often partitioned scheduling [6] is used, as it leverages the existing knowledge for scheduling in a single-core environment since each task is assigned to only one core and task migration is prohibited, i.e., each core operates as a single core.

In addition to meeting deadlines, scheduling is also a driving force behind dynamic voltage and frequency scaling (DVFS). DVFS enables a processing core to operate at multiple power states, where each power state is defined by a voltage and frequency (v/f) pair. When a core has low workload/utilization, the core can reduce its v/f pair, thereby operating at a lower voltage and improving energy efficiency. Conversely, when a core has high workload/utilization, it can operate at a high v/f pair so that deadlines can be guaranteed. Typically, DVFS is implemented in a way that it works in tandem with the scheduling to determine the utilization level and corresponding v/f pair.

Unfortunately, one of the shortcomings of partitioned scheduling is that since each task is assigned in its entirety to one core, the workload of the task set is typically distributed across the available cores in a non-uniform or imbalanced manner (i.e., the cores in the multi-core system are assigned different utilizations). We will show later in this thesis that such imbalance in core utilization can be significant. While an imbalance may be benign as deadlines can still be guaranteed, it can be costly from the perspective of energy efficiency. Specifically, some cores will need to operate at a higher v/f pair. Although it may seem that assigning some cores a higher utilization means that the other cores have a lower utilization, and thus the total energy is conserved, because the relation between voltage and energy is quadratic, the additional energy consumption to have some of the cores operate at a higher v/f pair exceeds the energy saved by the other cores operating at a lower v/f pair. From the perspective of energy efficiency, it is hence desirable that all operating cores are assigned the same utilization, a property referred to herein as load-balanced.

To make matters worse, implementing DVFS requires that each core has an independent clock and power supply, which, due to shrinking technology nodes, increased clock speeds, and an increase in the number of processing cores, has become increasingly difficult to realize. To overcome this difficulty, Choudhary and Marculescu proposed the concept of a voltage and frequency island (VFI) [13]. A VFI simplifies the implementation of DVFS by driving multiple cores with a single clock and power supply (i.e., each island has one clock and one power supply distributed to multiple cores). The use of VFI is commonplace in modern multi-core processors. However, one of the drawbacks of VFI is that the cores can no longer set their v/f pair independently of the VFI (i.e., each core must operate at the v/f pair of its VFI) [14], which can reduce the effectiveness of DVFS to save energy. To guarantee deadlines when multiple cores are on a single VFI, the v/f pair of the VFI must be set to the highest v/f pair needed by any of the cores on that VFI. This means that if at least one core on a VFI has a higher utilization than the others, all the cores on that VFI must operate at the increased v/f pair so that the core with a higher utilization can guarantee to meet all of its deadlines. Subsequently, the energy consumption is increased since some cores must now operate above their needed voltage and frequency. Such an increase in energy consumption is referred to herein as the VFI cost.

To mitigate the effects of imbalance in multi-core HRTS that implement DVFS with multiple cores on a single VFI, we propose a scheduling heuristic called Earliest Deadline First for HRTS with VFI (EDF-hv) that attempts to load-balance hard real-time tasks on a multi-core processor. The main contributions of this work are as follows.

- Formal definitions of the concept of imbalance and VFI cost are given and best- and worst-case bounds on the VFI cost are derived.
- A scheduling heuristic, namely EDF-hv, which is suitable for HRTS, is presented to improve the energy efficiency of the system in the presence of VFI.
- Simulation results pertaining to VFI cost, approximating the average behavior, are shown with its correlation to system imbalance.
- Simulation results that allow the performance of EDF-hv to be assessed are presented and a classification of the types of systems that will benefit from EDF-hv is described.
- Finally, in Appendix A, oversized tasks, which are an artifact of EDF-hv, are formally defined as well as bounds on oversized tasks are derived and methods to handle oversized tasks are presented.

The rest of this thesis is organized as follows. In Chapter 2, the related work is presented. Chapter 3 provides the system model. The definitions of imbalance and VFI cost with theoretical maximum and minimum limits are described in Chapter 4. To mitigate the effects of VFI cost, we introduce EDF-hv in Chapter 5. Chapter 6 presents simulation results pertaining to VFI cost as well as the increase in energy efficiency as obtained by EDF-hv. Chapter 7 concludes this thesis and presents future research directions. Finally, Appendix A explores an artifact of EDF-hv, referred to herein as “oversized tasks”.

Chapter 2

Related Works

The related work will be presented in three major categories. First, we will present a brief overview of power management and explain its static and dynamic components. Next, the concepts of global, partitioned, and semi-partitioned scheduling will be discussed. Finally, we will review the portion of the semi-partitioning heuristic that is especially relevant to this work.

2.1 Power Management

We have introduced the concept of DVFS previously; however, DVFS is only a portion of managing power for energy efficiency [14–29]. As explained in [15, 16], the power consumption of real-time systems can be broadly divided into two categories: static power and dynamic power. Static power arises from the leakage current, which is the result of transistors constantly conducting a small amount of current even when they are static/inactive (i.e. when the transistors are not changing between low and high states). Dynamic power is primarily the result of charging and discharging the load capacitance as the transistors change between high and low states as required for computation. Typically, techniques to reduce energy consumption focus on reducing either static or dynamic power and, since static power now dominates energy consumption [14], it is common for designers to first apply appropriate techniques to efficiently manage static power, and then to apply appropriate techniques to efficiently manage dynamic power. This thesis focuses on reducing dynamic power and is intended to work in conjunction, not competition, with techniques to reduce static power [14–21].

The primary method used to minimize dynamic power is DVFS [15]. Many DVFS algorithms exist that are suitable for real-time systems [22–25]. The work in [22] presented three increasingly aggressive DVFS algorithms that leverage knowledge of the task set and already completed tasks in order to set the v/f pair in a way that guarantees deadlines. The most aggressive of these techniques is called look-ahead dynamic voltage scaling (LADVS), which attempts to delay execution as long as possible with the expectation that some tasks will complete early, and thus the core voltage and frequency may not need to be increased to meet the deadlines. The work in [23] and [24] proposed using probabilistic knowledge of the task set to predict the execution time of each task. In [23], the authors considered a task model where each task has a set of possible execution times and each of these possible execution times in the set has a probability of occurrence. Thus, based on this profile, the execution behavior can be predicted. Additionally, while [23] and [24] both placed an emphasis on maintaining a high level of timeliness, [24] specifically attempted to maintain a defined level of reliability. Unfortunately, since both works rely on probabilistic models, neither is suitable for HRTS. The work in [26] also attempted to use a probabilistic method suitable for HRTS, but under the added complexity that the task set is uncertain. In this case, the probabilistic method pertains to HRTS only because, under the assumption of an uncertain workload, no deadlines can fully be guaranteed. It is significant to note that while some existing DVFS-based algorithms are suitable for HRTS, all consider DVFS in a single core environment or on multi-core systems where each core is its own VFI [22–29].

2.2 Semi-partitioning

The work in [30] categorizes multiprocessor real-time scheduling algorithms as global, partitioned, or a hybrid of the two. These scheduling algorithms are classified based on the amount of task migration that is allowed among the cores. As mentioned previously, partitioned scheduling prohibits migration and, conversely, global scheduling permits unrestricted migration. Further, [30] calculated how much of the system's total capacity the task set can require while still guaranteeing to produce a valid schedule. Partitioned scheduling minimizes scheduling overhead, as there is no task migration, and has been shown to be able to guarantee deadlines using EDF, however, for partitioning heuristics, if the task set will use more than approximately 50% of the capacity of a system, the task set is not guaranteed to be schedulable. On the other hand, global scheduling can schedule up to 100% of the systems capacity; however the scheduling overhead is increased and deadlines are not guaranteed by EDF.

In an attempt to get the best of partitioned and global scheduling, a hybrid approach to scheduling that allows restricted task migration has been developed [31–37] and is typically referred to as semi-partitioning. In semi-partitioning, limited task migration is permitted for specific tasks while other tasks are fixed only to execute on an assigned core. In this thesis, we follow the semi-partitioning heuristics proposed by Anderson et. al. in [32, 37]. The work in [32] presents EDF-fm, a semi-partitioning algorithm that guarantees feasibility up to 100% of the system capacity with the restriction that no single task can have a utilization greater than 50% of a single core. In [37], EDF-fm was extended to EDF-os, which removes the per task utilization requirements, i.e., tasks can have a utilization up to 100% of a single core. Neither of these semi-partitioning heuristics can guarantee deadlines; however, both have a bound on the how late the tasks can be, referred to as bounded tardiness. Further, in both EDF-fm and EDF-os, for migrating tasks, to reduce migration overhead, the task can only migrate from one core to another after a job/iteration of the task has fully completed, a property referred to herein as boundary-limited.

2.3 Earliest Deadline First Optimal Semi-Partitioned Scheduling (EDF-os)

EDF-os [37] is designed for periodic tasks on a homogeneous multi-core system. As a semi-partitioning algorithm, task are either fixed (i.e., the task is assigned to execute on only one core) or migrating (i.e., the task is assigned to execute a defined share of jobs on a subset of the cores). In EDF-os, tasks are partitioned in two steps. In the first step, as many tasks as possible are assigned to one of the cores as fixed tasks using the worst-fit decreasing (WFD) heuristic [6]. When a task is encountered that cannot be assigned to a core using WFD, the second part of the partition begins. In the second step, using a next-fit-like heuristic (note that tasks are still in decreasing order), the remaining tasks are assigned a share on cores that still have available capacity until the complete utilization of the task is allotted. Under EDF-os, a migrating task can execute on more than two cores;

however, each core is limited to a maximum of two migrating tasks. After the tasks have been partitioned, the migrating tasks will execute the defined share of jobs on each of its cores and migration is determined with a proportionate fairness (p-fair) algorithm [38, 39]. Fixed tasks are prioritized using EDF while migrating tasks have fixed priority over fixed tasks. Additionally, on cores that have two migrating tasks, one migrating task has a fixed priority over the other. Due to the method in which tasks are assigned to the cores, Anderson et al. were able to derive tardiness bounds for the tasks [37].

Chapter 3

System Model

As this thesis focuses on reducing dynamic power consumption for improved energy efficiency, in this chapter, we will first present the system model and then discuss its implications for energy efficiency.

3.1 System Model

We consider a periodic task set, $\eta = \{\tau_1, \tau_2, ..., \tau_{\mathrm{N}}\}$, of N independent tasks to be scheduled on a homogeneous multi-core system with M identical processing cores, $P = \{p_1, p_1, ..., p_{\mathrm{M}}\}$, on a set of I VFI, $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_{\mathrm{I}}\}$, where N, M, and I are positive integers. The system must have at least one VFI and, since the system is homogeneous, each VFI, γ_l, must have the same number of cores on it. For ease of discussion, P_l is the subset of cores on γ_l and M_l is the number of cores in P_l. Further, since this work considers multi-core systems with multiple cores per VFI, $\mathrm{M} \geq \mathrm{M}_l > 1$.

Each VFI can adjust its voltage independently of the other VFI to operate at a set of K discrete frequencies, $\lambda = \{f_1, f_2, ..., f_{\mathrm{K}}\}$ such that $f_1 < f_2 < ... < f_{\mathrm{K}}$, where f_1 is the minimum/slowest frequency and f_{K} is the maximum/fastest frequency. For the sake of clarity, scheduling is described in terms of quanta of execution, where each quantum is the number of clock cycles sufficient to yield a reasonable unit of execution for scheduling (While the proper selection of a quantum size is an important aspect of scheduling, it is out of the scope of this work.). As such, each frequency, f_k, represents the number of quanta per second.

When a VFI is operating at frequency f_k, each core on the VFI has a corresponding dynamic power consumption of w_k. As such, there is a set of K power consumption levels $W = \{w_1, w_2, ..., w_{\mathrm{K}}\}$, referred to herein as power states, that correspond with the set of frequencies in λ and $0 < w_1 < w_2 < ... < w_\mathrm{K}$.

Each task, τ_i, is defined by a worst-case execution time (WCET), C_i, a period, T_i, a relative deadline, D_i, a utilization, u_i, and a profile of execution, x_i. The WCET is the absolute maximum number of quanta required for the task to complete and includes overhead due to context switches, etc. The period is the time, in seconds, for how often the task must execute. The deadlines of each task for this work are implicit, i.e., $T_i = D_i$; the task must complete before the start of the next period. The utilization, u_i, is the share of a core's utilization that the task must be allotted in order to meet its deadline and can be calculated by:

$$u_i = \frac{C_i}{f_\mathrm{K} \cdot T_i}. \tag{3.1}$$

The utilization of each task is restricted to $0 < u_i \leq 1$, i.e., a task must require a positive non-zero utilization, but no single task can require more that the total capacity of a single core. The profile of execution follows the probabilistic model proposed in [23] such that x_i, is a set of X_i potential execution times, c_α, with their probability of occurrence, ρ_α. As such, $x_i = \{(\rho_1, c_1), (\rho_2, c_2), ..., (\rho_{\mathrm{X}_i}, c_{\mathrm{X}_i})\}$, $\mathrm{X}_i > 0$, and $c_1 < c_2 < ... < c_{\mathrm{X}_i} = C_i$. Also,

$$\sum_{\alpha=1}^{\mathrm{X}_i} \rho_\alpha = 1. \tag{3.2}$$

Additionally, tasks in the task set are sorted in non-decreasing order of utilization such that:

$$u_1 \geq u_2 \geq ... \geq u_\mathrm{N}. \tag{3.3}$$

Since the task set, η, will be scheduled on the set of cores, P, the system utilization, U, is the sum of the utilization of the tasks and can be calculated as:

$$\mathrm{U} = \sum_{i=1}^{\mathrm{N}} u_i. \tag{3.4}$$

In order to guarantee deadlines with EDF, the cores cannot be overloaded. Hence, $\mathrm{U} \leq \mathrm{M}$.

During partitioning, the tasks are partitioned to the cores such that the task set, η, is divided into task subsets where each subset η_j is assigned to core p_j. However, following the EDF-os [37] semi-partitioning heuristic, each task is assigned a share, $s_{i,j}$, on each core such that:

$$\sum_{j=1}^{\mathrm{M}} s_{i,j} = u_i. \tag{3.5}$$

Note that shares can be zero and some tasks may have only one non-zero share. Such tasks are referred to as fixed tasks. Conversely, tasks that have a non-zero share on more than one core are considered migrating tasks. As mentioned, migrations are boundary limited and determined by p-fair [38,39]. Thus, all jobs of a fixed task will execute on a single core, i.e., the fixed task do not migrate, while the jobs of a migrating task will migrate between cores with a non-zero share such that the number of jobs executed on each core is equal to the share assigned to that core. Therefore, the utilization of each core U_j, is:

$$U_j = \sum_{i=1}^{\mathrm{N}} s_{i,j}. \tag{3.6}$$

For ease of discussion, η_j is the subset of tasks with a non-zero share on p_j, and N_j is the number of tasks in η_j.

3.2 Energy Considerations

As mentioned in Chapter 2, power management typically considers both static and dynamic power separately, and, since energy consumption is dominated by static power, typically techniques to manage static power are applied first and then dynamic power is managed. As such, in the system model, P and Γ do not necessarily represent the entire set of available cores and VFI of the hardware. Since HRTS are typically greatly over designed, generally it is not efficient to manage the energy solely from the perspective of dynamic power. In the interest of energy efficiency, typically as many cores as possible would be shut-off using power-gating or at least set in low-power modes [15,16]. However, since VFI limit the control to the operating cores, often the hardware cannot power-gate

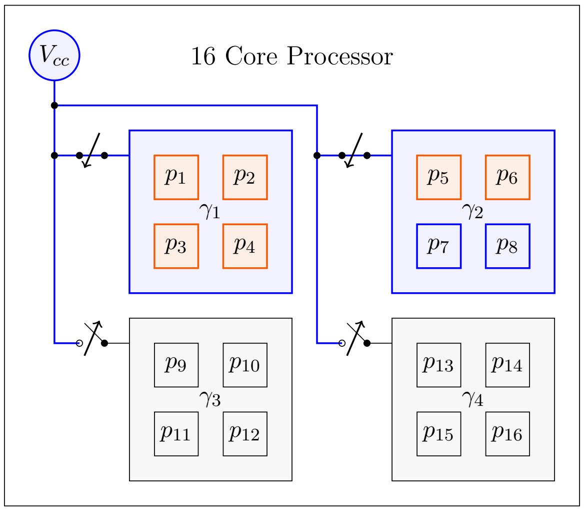


Fig. 3.1: Example of the limitations on static power management imposed by VFI. Consider a processor with sixteen cores on four homogeneous VFI (i.e. each VFI has four cores). Suppose at some time, the task set requires only six cores, shown in orange, to guarantee deadlines. With four cores per VFI, to ensure six cores are available, two VFI must be powered on. Unfortunately, powering on two VFI means the system has eight operating cores consuming energy.

each core separately, but rather to each VFI. Consider the example shown in Figure 3.1. The system has four VFI, each with four cores. For a given time, suppose the task set has a utilization of $U = 5.8$. In this case, since the utilization is greater than five, a minimum of six cores is required to execute the task set in a way that deadlines can be guaranteed with EDF. However, since there are four cores per VFI, the system cannot turn on exactly six cores. Thus, eight cores must be turned on, meaning $\mathrm{M} = 8$ and $\mathrm{I} = 2$. In this case, since eight cores must be turned on in order to guarantee deadlines, it is advantageous from the perspective of energy efficiency to have each core operate with a utilization of $U_j = 0.725$ (i.e. load-balanced), rather than have five cores operate at $U_1 = U_2 = U_3 = U_4 = U_5 = 1$, the sixth core operate at $U_6 = 0.8$, and the remaining two cores be idle.

A second energy efficiency consideration is the implications of the minimum available frequency and power state, i.e. f_1 and w_1, respectively. It has been shown in [14] that there is a point at which reducing the v/f pair of the cores on a VFI actually increases total energy consumption. This happens because reducing the v/f pair means that the cores must operate for a longer period of time to accomplish the same amount of work. Thus, more energy is consumed by the static power from keeping the cores operating for a longer period than is conserved by the dynamic power in reducing the v/f pair. Unfortunately, this point assumes that the cores on a VFI could be shut off or otherwise set in low-power modes as soon as work is completed, which may not always be possible (the reasoning is explained further in the next paragraph). As such, to maintain energy efficiency, f_1 and w_1 may not necessarily correspond to the lowest v/f pair available in the system, but rather may be set to a higher v/f pair based on the behavior of the task set.

Finally, during run time, tasks may complete in less than their WCET, which can cause there to be periods where there are no tasks to be executed, and thus, the core is idle (or, as mentioned in the preceding paragraph, the core may be idle as a result of the v/f pair being set to a v/f pair above the hardware's minimum capabilities in order to reduce static power consumption). Further, when a core is idle, since tasks are periodic, each core's scheduler knows how long until the next task will be available to execute. Although using static power management during idle periods is ideal, there are two practical restrictions. First, since DVFS can typically manage the dynamic power at a rate significantly faster than power-gating or low-power modes can (e.g. changing the v/f pair takes significantly less time than power-gating a VFI), the idle periods must be sufficiently long for static power methods to be implemented. Second, since the VFI limits the control of static power methods to each VFI, using static power techniques during idle periods is only possible if all cores on the VFI are experiencing idle periods.

Therefore, in addition to limiting the effectiveness of DVFS, the presence of VFI can also limit the effectiveness of static power management techniques. In this way, the number of operating cores may be less than optimal for static power management as well as static power techniques may be limited, and thus, improved dynamic power management is advantageous. For this reason, dynamic power is still an important consideration for energy efficiency in HRTS.

Chapter 4
Quantifying Imbalance and VFI Cost

We now consider the imbalance and the VFI cost. As has been mentioned previously, imbalance occurs in pure partitioning heuristics when some cores in a multi-core system are assigned a greater workload than others. VFI cost is the increase in energy consumption that occurs when one or more cores on a VFI operate at a v/f pair above that which is necessary to guarantee deadlines because another core on the VFI needs the increased v/f pair to guarantee deadlines. In this chapter, we will formally define the imbalance and the and VFI cost as well as present a best- and worst-case analysis of the VFI cost. In later chapters, we will show the correlation between imbalance and VFI cost.

4.1 Imbalance

To achieve an ideal load-balance on the system, each core should be assigned the same utilization. The utilization, Φ, that each core needs to be assigned in order to attain a load-balance can be calculated by:

$$\Phi = \frac{\mathrm{U}}{\mathrm{M}}. \tag{4.1}$$

Unfortunately, since the utilization of each task can be any value which is greater than zero but less than or equal to one, there is no guarantee that the tasks can be partitioned in such a way to achieve a load-balance. Further, while the WFD heuristic does approach a load-balanced partition, and thus is often used for energy-efficiency, it is not guaranteed to find a load-balanced solution, even if one exists. Formally, the imbalance, ϕ, is the amount by which the utilization of the cores differs from the ideal load-balance:

$$\phi = \frac{\sum_{j=1}^{\mathrm{M}} \frac{|\Phi - U_j|}{\Phi}}{\mathrm{M}} = \frac{\sum_{j=1}^{\mathrm{M}} |\Phi - U_j|}{\mathrm{U}}. \tag{4.2}$$

4.2 VFI Cost

The VFI cost is the amount of energy consumed as a result of one or more cores operating at a v/f pair that is higher than is necessary (Recall the VFI must operate at the highest v/f pair required by any of the cores on it.). More formally, for a period of T quanta, $[0, \mathrm{T})$, on a given VFI, γ_l, for each quantum, t, each core in P_l will calculate an optimal operating frequency using some DVFS-based scheduling algorithm such as LADVS [22]. Next, γ_l will set its frequency to the maximum of the frequencies calculated by the cores in P_l. Let the optimal frequency, f_o, calculated by each core at quantum t be denoted by $f_o(j,t)$, with the corresponding power state given by $w_o(j,t)$. In addition, let the frequency set by γ_l be $f_{VFI}(l,t)$, with the corresponding power state given by $w_{VFI}(l,t)$. The VFI cost, Ψ, for a single quantum, denoted as $\Psi(t)$, can be calculated as:

$$\Psi(t) = \frac{\sum_{j=1}^{\mathrm{M}} [w_{VFI}(l,t) - w_o(j,t)] \forall p_j \in P_l}{\mathrm{M}_l \cdot w_{VFI}(l,t)}. \tag{4.3}$$

Since the VFI cost of the system is simply the average of all VFI in the system over all of the quanta, the VFI cost can be calculated by:

$$\Psi = (\mathrm{I} \cdot \mathrm{T})^{-1} \sum_{l=1}^{\mathrm{I}} \sum_{t=0}^{\mathrm{T}-1} \frac{\sum_{j=1}^{\mathrm{M}} [w_{VFI}(l,t) - w_o(j,t)] \forall p_j \in P_l}{\mathrm{M}_l \cdot w_{VFI}(l,t)}. \tag{4.4}$$

Now, since the value of $w_{VFI}(l,t)$ and $w_o(j,t)$ for any value of t are in the power state set W and $w_{VFI}(l,t) \geq w_o(j,t)$, then:

$$w_{VFI}(l,t) > w_{VFI}(l,t) - w_o(j,t).$$

Thus,

$$\mathrm{M}_l \cdot w_{VFI}(l,t) > \sum_{j=1}^{\mathrm{M}} [w_{VFI}(l,t) - w_o(j,t)] \forall p_j \in P_l. \tag{4.5}$$

As a result, the VFI cost is strictly less than one. Additionally, from Equation 4.4, it is possible to determine worst- and best-case VFI cost, as described next.

4.2.1 Worst-Case VFI Cost

The maximum VFI cost will occur when the difference between the set VFI power state $w_{VFI}(l,t)$ and the cores power state $w_o(j,t)$ is largest for all of the cores on the VFI for all quantum. The maximum possible value of $w_{VFI}(l,t)$ is w_{K}, and the minimum possible value of $w_o(j,t)$ is w_1, however, $w_{VFI}(l,t)$ can only be w_{K} if the value of $w_o(j,t) = w_{\mathrm{K}}$ for at least one core on the VFI. Thus, the maximum VFI cost at a single quantum t, denoted as $\Psi_{max}(t)$, will occur when $w_o(j,t) = w_{\mathrm{K}}$ for only one core and $w_o(j,t) = w_1$ for all other cores. Therefore, the maximum VFI cost for a single quantum is:

$$\Psi_{max}(t) = \frac{\sum_{j=1}^{\mathrm{M}_l-1}[w_{\mathrm{K}} - w_1]}{\mathrm{M}_l \cdot w_{\mathrm{K}}}$$

$$\Psi_{max}(t) = \frac{(\mathrm{M}_l - 1)(w_{\mathrm{K}} - w_1)}{\mathrm{M}_l \cdot w_{\mathrm{K}}}$$

$$\Psi_{max}(t) = \frac{\mathrm{M}_l \cdot w_{\mathrm{K}} - w_{\mathrm{K}} - \mathrm{M}_l \cdot w_1 + w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}$$

$$\Psi_{max}(t) = 1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}. \tag{4.6}$$

Since the maximum VFI cost, Ψ_{max}, will be achieved when all quanta are at the maximum for all VFIs in the systems, substituting Equation 4.6 into Equation 4.4 produces:

$$\Psi_{max} = (\mathrm{I} \cdot \mathrm{T})^{-1} \sum_{l=1}^{\mathrm{I}} \sum_{t=0}^{\mathrm{T}-1} \left(1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}\right)$$

$$\Psi_{max} = (\mathrm{I} \cdot \mathrm{T})^{-1} \sum_{l=1}^{\mathrm{I}} \left(1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}\right) \sum_{t=0}^{\mathrm{T}-1} 1$$

$$\Psi_{max} = \mathrm{T}(\mathrm{I} \cdot \mathrm{T})^{-1} \left(1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}\right) \sum_{l=1}^{\mathrm{I}} 1$$

$$\Psi_{max} = \mathrm{I} \cdot \mathrm{I}^{-1} \left(1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}\right)$$

$$\Psi_{max} = 1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l w_{\mathrm{K}}}. \tag{4.7}$$

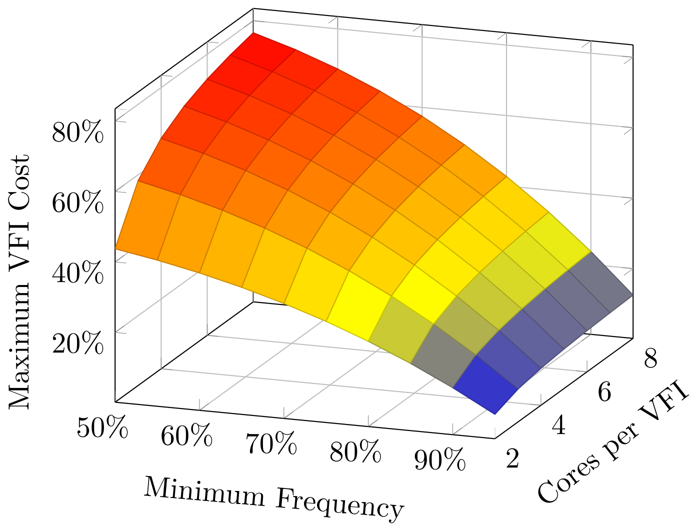


Fig. 4.1: The maximum VFI cost (Equation 4.7) as a function of the number of cores (M) by the minimum frequency (f_1) that the VFI can reduce to (as a percentage of the maximum frequency, f_{K}) assuming a pure CMOS circuit (i.e., $w_k \propto f_k^3$).

Thus, the maximum VFI cost of the system is the same as the maximum VFI cost of a single quantum. The maximum VFI cost is shown in Figure 4.1 for 2 to 8 cores per VFI, assuming a pure CMOS circuit, i.e., $w_k \propto f_k^3$, for systems that can reduce the frequency from 95% of the maximum to 50% of the maximum (currently, most cores can reduce to about 70% of the maximum frequency).

4.2.2 Best-Case VFI Cost

Conversely, to achieve the minimum VFI cost, Ψ_{min}, the difference between $w_{VFI}(l,t)$ and $w_o(j,t)$ must be minimized for all of the cores on each VFI for all quanta. This will occur when $w_{VFI}(l,t) = w_o(j,t)$. Using Equation 4.4 when $w_{VFI}(l,t) = w_o(j,t)$:

$$\Psi_{min}(t) = (\mathrm{I} \cdot \mathrm{T})^{-1} \sum_{l=1}^{\mathrm{I}} \sum_{t=0}^{\mathrm{T}-1} \frac{\sum_{j=1}^{\mathrm{M}}(0)}{\mathrm{M}_l \cdot w_{VFI}(l,t)} = 0. \tag{4.8}$$

Therefore, it is possible for there to be no VFI cost regardless of whether the system is operating at its maximum, minimum, or any other power state.

Chapter 5

EDF-hv

EDF-hv modifies the partitioning process of EDF-os to load-balance a system instead of guaranteeing feasibility up to 100% of the system's capacity. With this modification, EDF-hv then extends the scheduling procedure of EDF-os to guarantee deadlines instead of providing tardiness bounds. In this chapter, we first present the partitioning process of EDF-hv. We then describe the modifications to the task scheduling process. We end this section with some important properties of EDF-hv and provide a proof that EDF-hv can guarantee hard real-time deadlines.

5.1 Partitioning for a Load-Balance

EDF-hv follows the partitioning process of EDF-os with two main modifications in order to achieve a load-balance. First, we restrict the available capacity of each core to be no more than Φ. Second, since no task can have a utilization greater than that of a core's capacity in order to guarantee feasibility, it is also necessary to restrict the utilization of each task such that $u_i \leq \Phi$ (considerations for task sets with task that have a $u_i > \Phi$, referred to as oversized tasks, are presented in Appendix A). An example comparing the partition of EDF-hv to WDF and EDF-os is shown in Figure 5.1. In this example, there are six tasks to be partitioned onto four cores. With WFD, one task is assigned to each core, until all of the cores have one task. Next, the remaining two tasks are assigned to the cores with the lowest utilization. Since all of the tasks can fit onto the cores, the second step of the partitioning process is not initiated, and, thus, the partition is the same for EDF-os as it is for WFD (recall that EDF-os is designed to increase feasibility, not to improve the load-balance). In contrast, with EDF-hv, the capacity of each core is restricted to Φ such that, after the initial four tasks have been assigned, the remaining two tasks will no longer

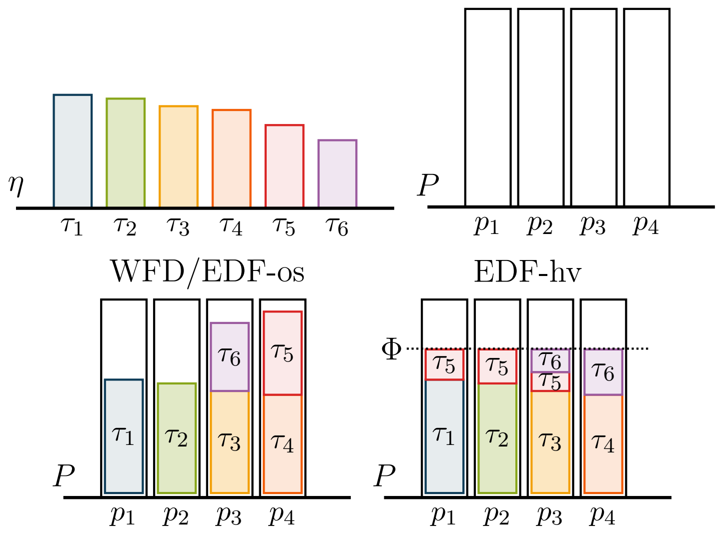


Fig. 5.1: Partitioning the task set, η (top left), where each task is represented by its utilization, onto the set of cores, P (top right), using the WFD (or EDF-os; bottom left) and EDF-hv (bottom right). EDF-hv restricts the available capacity of each core to Φ.

fit into the remaining available capacity of each core and thus, the remaining two tasks are assigned as migrating tasks with shares on multiple cores.

Since EDF-hv follows the partitioning process of EDF-os, EDF-hv can now guarantee that the resultant partition to be load-balanced since EDF-os can guarantee feasibility up to 100% of the system's capacity; as stated by the following theorem.

Theorem 1. *Given a task set η where $u_i \leq \Phi$, $i = 1, ..., \mathrm{N}$, and a set of M cores, EDF-hv partitions the tasks among the cores in such a way as to achieve a load-balance.*

Proof. As previously mentioned, Anderson et al. showed that the partitioning process of EDF-os can guarantee feasibility with a system utilization bound of 100% while allowing individual task utilization of up to 100% of a core's capacity [37]. As such, for a system where $\mathrm{U} = \mathrm{M}$, EDF-os guarantees that the system can be partitioned. Interestingly, in this case, EDF-os also load-balances the system since each core must be assigned 100% of its utilization, otherwise, $\sum_{j=1}^{\mathrm{M}} U_j < \mathrm{U} = \mathrm{M}$ (Note that, in this case, $\Phi = 1$, which is the utilization bound on the individual tasks.). The partitioning process of EDF-hv is identical to that of EDF-os, with the exception of the two modifications. These restrictions effectively induce the case where the task set utilization is equal to the available system capacity, which EDF-os has already been shown to guarantee feasibility. Therefore, EDF-hv is guaranteed to be load-balanced. □

Caveat 1. *Restricting the utilization of each task to $u_i \leq \Phi$ yields a circular definition that can cause an invalid state. Consider Equation 4.1 and Equation 3.4:*

$$\Phi = \frac{\mathrm{U}}{\mathrm{M}} = \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}}.$$

Thus, if $u_i \leq \Phi$, then:

$$u_i \leq \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}}.$$

Considering the case of τ_1:

$$u_1 \leq \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}}.$$

$$\mathrm{M} \cdot u_1 \leq u_1 + u_2 + ... + u_{\mathrm{N}} \tag{5.1}$$

Recall that tasks are sorted in a non-decreasing order of utilization, i.e., the first task has the largest utilization, and thus, Equation 5.1 yields that M times the largest utilization must be less than or equal to the sum of N tasks. As such, Equation 5.1 is only true if $\mathrm{N} > \mathrm{M}$, except in the case that $\mathrm{N} = \mathrm{M}$ and $u_1 = u_2 = ... = u_{\mathrm{N}}$. Since assuming all tasks have the same utilization is not very practical, EDF-hv therefore requires that $\mathrm{N} > \mathrm{M}$.

Lemma 1. *Each core, p_j, has at least one fixed task.*

Proof. In EDF-hv $\mathrm{N} > \mathrm{M}$, $\Phi = \frac{\mathrm{U}}{\mathrm{M}}$, and tasks are first partitioned using WFD. Under the WFD heuristic, a task will be assigned to an empty core before a core will be assigned a second task. Further, since all tasks have a utilization, $u_i \leq \Phi$, it is guaranteed that at least one task can fit on each core. Finally, since there are more tasks than cores (i.e., $\mathrm{N} > \mathrm{M}$), each core must be assigned at least one fixed task. □

Lemma 2. *Each core, p_j, has a maximum of two migrating tasks.*

Proof. Following the proof for Property 3 of EDF-os in [37], it can be shown by induction that during the partitioning process of EDF-hv, when assigning a migrating task to a core, there can be at most one migrating task already assigned. □

5.2 Scheduling

The reason that tardiness bounds can be derived in EDF-os, but deadlines cannot be guaranteed is that cores with migrating tasks can be overloaded when migrating tasks are executing. As an example, consider the periodic task set for a multi-core system with two cores, as shown in Table 5.1. For simplicity, task periods are given in quanta to avoid having to consider the frequency of the cores. The resultant EDF-os partition is shown on the right side of Table 5.1. In this system, τ_1 and τ_2 are fixed tasks of p_1 and p_2, respectively, while τ_3 is a migrating task on both p_1 and p_2. Based on the shares of τ_3 on p_1 and p_2, of every eleven jobs of τ_3, five will execute on processor p_1, while six will execute on p_2. Although the assigned utilization of p_1 and p_2 is 100% each, p_1 and p_2 are overloaded when they have to execute both their fixed and migrating tasks. Since all three tasks have the same period, it is easy to see that during periods where p_1 has only its fixed task, τ_1, to execute, it only needs 15 of the 20 quanta in that period, i.e., the utilization during this period is only $\frac{15}{20} = 75\%$. However, during periods where p_1 has its fixed task, τ_1, and its migrating task, τ_3, to execute, it must execute $15 + 11 = 26$ cycles of 20, i.e., the utilization during this period is $\frac{15+11}{20} = \frac{26}{20} = 130\%$. Therefore, p_1 is overloaded. Similarly, p_2 is overloaded, as shown in the Table 5.2. As such, in EDF-os, since the cores can be overloaded when they have both their fixed and migrating tasks, deadlines cannot be guaranteed. However, since the cores are not fully utilized when they do not have their migrating tasks, the cores can "catch-up", thus, the tardiness of deadlines can be bounded.

Therefore, for EDF-hv to guarantee deadlines, the cores must not be overloaded. Anderson et. al. in [37] propose that EDF-os can be suitable for HRTS by increasing the capacity of the system, e.g. increasing the number of cores, using cores with higher clock speed, etc., until the tardiness bounds are zero. However, this approach is not ideal as the tardiness bounds in EDF-os are not dependent on the task utilizations due to the priori-

Table 5.1: Example periodic task set (left) with EDF-os partition onto a system with two cores (right).

Task Set				Partition		
Task	**WCET**	**Period (Quanta)**	**Utilization**	**Task**	p_1	p_1
τ_1	15	20	75%	τ_1	75%	0%
τ_2	14	20	70%	τ_2	0%	70%
τ_3	11	20	55%	τ_3	25%	30%

Table 5.2: Utilization of cores for the task set in Table 5.1 when migrating tasks are and are not present.

Core	**Task Set**	**Tasks**	**Utilization**
p_1	Fixed Tasks	τ_1	$\frac{15}{20} = 75\%$
	Migrating Tasks	τ_1, τ_3	$\frac{15+11}{20} = 130\%$
p_2	Fixed Tasks	τ_2	$\frac{15}{20} = 70\%$
	Migrating Tasks	τ_2, τ_3	$\frac{14+11}{20} = 125\%$

tization of the migrating tasks. For example, let us reconsider the task set in Table 5.1. For p_1 to not be overloaded, during a period where both τ_1 and τ_3 are on p_1, since τ_3 is a migrating task, it must complete τ_3 first, and then τ_1 can execute. As such, the capacity of p_1 must increase to $\frac{11+15}{20} = 130\%$ (which is the value calculated in Table 5.2). However, if the WCET and period of τ_3 are increase to 22 and 40, respectively, even though the utilization of the system remains constant, for periods when both τ_1 and τ_3 are on p_1, the capacity must increase to $\frac{22+15}{20} = 185\%$. As such, prioritizing migrating tasks over fixed task yields that guaranteeing deadlines is no longer solely a function of the utilization.

As a result, in EDF-hv, all tasks are scheduled with EDF, i.e., migrating tasks are not prioritized over fixed tasks. Clearly, by changing the prioritization of the migrating and fixed tasks, the tardiness bounds in [37] for EDF-os are no longer valid. However, as will be proved next, tardiness bounds are not necessary for EDF-hv since deadlines can be guaranteed.

5.3 Guaranteeing Deadlines

Even with all tasks being scheduled with EDF, task deadlines cannot be guaranteed when a core is overloaded. In the previous section, it was shown that EDF-hv can load-balance the task set on the cores. However, this is insufficient to guarantee deadlines, because the utilization considered during the partitioning process, shown in Equation 3.6, is the average utilization of each core. As has been discussed, the actual core utilization fluctuates depending on whether the migrating task(s) are present. Thus, to guaranteed deadlines, the core must not be overloaded even when experiencing the maximum utilization.

To determine the maximum utilization that a core can experience, recall that a fixed task has a non-zero share equal to the task utilization on only one core, and migrating tasks have more than one non-zero share less than the task's utilization on the cores. However, during execution, since migrations are boundary limited, each core will have to be able to execute the complete utilization of each task assigned to it. As such, for cores with only fixed tasks, the maximum utilization is simply the sum of the shares assigned to that core as given by Equation 3.6. For cores with one migrating task, the maximum utilization is the sum of the shares of the fixed tasks, plus the utilization of the migrating task. Finally, for cores with two migrating task, the maximum utilization is, in the best-case, the sum of the shares of the fixed tasks plus the utilization of the first migrating task (since the first migrating task must have a utilization greater than or equal to the second migrating task by Equation 3.3), or, in the worst-case, the sum of the shares of the fixed tasks plus the utilization of both migrating tasks. The best-case for cores with two migrating tasks occurs when the migrating tasks have migration patterns that ensure only one migrating task at a time will be present on the core; otherwise, the worst-case will occur. Unfortunately, unless the two migrating tasks have harmonic periods and both tasks have harmonic shares on all of the cores with non-zero shares, simulation or another intensive calculation is required to determine if the best-case or worst-case will occur. Thus, to simplify calculation, since the maximum utilization can be bounded by the worst-case, the worst-case can be used. Therefore, the maximum utilization that a core can experience is:

$$U_{j_{max}} = \sum_{i=1}^{\mathrm{N}_j} u_i \forall \tau_i \in \eta_j. \tag{5.2}$$

For cores with only fixed tasks or cores with only one migrating task, Equation 5.2 is the maximum utilization that a core can experience, however, for cores with two migrating tasks, Equation 5.2 is an upper bound on the maximum utilization. Further, the maximum utilization may only occur rarely (e.g. if a core with a single migrating task is assigned only a hundredth share of the migrating tasks utilization, the maximum utilization will only occur one out of a hundred iterations of the task), yet it is the upper limit on the utilization that the core can experience. As such, even if the maximum utilization only occurs rarely, to guarantee deadlines under all conditions, the maximum deadline must be less than or equal to one.

Theorem 2. *If $U_{j_{max}} \leq 1$, all task deadlines are guaranteed under EDF-hv.*

Proof. EDF is known to be able to guarantee deadlines on a core with a periodic task set with implicit deadlines as long as the utilization of that processor is less than or equal to one [1]. EDF-hv also assumes a periodic task set but allowing tasks to migrate at job boundaries complicates the behavior of the task set. While cores with only fixed tasks have a periodic task set with implicit deadlines, meaning deadlines can be guaranteed by the properties of EDF, for a core with both fixed and migrating tasks, some share of the time the migrating tasks will not be executing jobs on the core. However, when the migrating tasks are executing on the cores, the jobs are released at period boundaries and have implicit deadlines. In fact, the task will still have all the properties of a periodic task with implicit deadlines, with the exception that for some periods, the execution time will be zero. Thus the total utilization of a core with migrating tasks will always be less than or equal to the maximum utilization. Therefore, since the maximum utilization of all fixed and migrating tasks is less than or equal to one, deadlines can be guaranteed by EDF. □

Therefore, in EDF-hv, guaranteeing deadlines is dependent on the maximum utilization that any of the cores in the system can experience. The maximum utilization that any of the cores in the system can experience can be bounded by the total system utilization.

Theorem 3. *EDF-hv is guaranteed to satisfy Theorem 2 if* $\mathrm{U} \leq \frac{1}{3}$.

Proof. Since EDF-hv will load-balance the system, the average utilization of each core will be Φ, however, the maximum utilization that a core can experience is given by Equation 5.2. From Lemma 1 and Lemma 2, each core is guaranteed to have at least one fixed task and at most two migrating tasks. Further, from Equation 3.3, since fixed tasks are assigned before migrating tasks, the fixed task on any core must have a utilization greater than or equal to the utilization of the migrating tasks. From Theorem 1, the maximum utilization that a task can have in EDF-hv is Φ. If a core is assigned a fixed task with a utilization of Φ, then all of the core's capacity is assigned, and thus, the core will not have any migrating tasks. However, if a core is assigned a fixed task of slightly less than Φ, then the core can have up to two migrating tasks. As such, the maximum utilization that a core can experience can be bounded by:

$$U_{j_{max}} < 3\Phi.$$

Thus, since by Theorem 2, to guarantee deadlines:

$$U_{j_{max}} \leq 1 \rightarrow 3\Phi \leq 1 \rightarrow \Phi \leq \frac{1}{3}.$$

By Equation 4.1:

$$\frac{\mathrm{U}}{\mathrm{M}} \leq \frac{1}{3}$$

$$\mathrm{U} \leq \frac{\mathrm{M}}{3}.$$

Therefore, if the total task set utilization is less than or equal to one-third of the system capacity, the system can guarantee deadlines. □

Unfortunately, only utilizing one-third of the system's capacity is not very practical for energy efficiency as modern processors are unable to reduce their v/f pair below approximately 70%. Fortunately, Theorem 3 is the worst-case and thus is not applicable to most systems (recall that Theorem 2 is sufficient to guarantee deadlines). However, it still follows that a drawback of guaranteeing deadlines is that the actual feasibility of EDF-hv is reduced as not every task set can satisfy Theorem 2. In fact, satisfying Theorem 2 actually reduces the feasibility of EDF-hv to less than that of WFD, as described in the following theorem.

Theorem 4. *EDF-hv has lower feasibility than WFD.*

Proof. Suppose a task set cannot be partitioned using WFD on to a multi-core processor. This means that one or more tasks is large enough that, after at least the first M tasks have been scheduled, they cannot fit onto any core without exceeding the core's capacity. In this case, under EDF-hv, the previously assigned tasks will become fixed tasks and then the task(s) that did not fit on any single core in WFD will be assigned as migrating task(s). Although this approach will satisfy the average utilization because the maximum utilization is the sum of the fixed and migrating tasks, even if a core is only assigned one migrating task, the maximum utilization will violate the capacity of the core since we know that the migrating task did not fit inside the capacity of any of the cores (which is why it could not be partitioned with WFD). Therefore, any system that cannot be partitioned with WFD cannot be scheduled with EDF-hv and still guarantee deadlines. Further, it is trivial to show by example that some systems can be partitioned by WFD, but not by EDF-hv. Therefore, EDF-hv has lower feasibility than WFD. □

While Theorem 4 indicates that EDF-hv has lower feasibility than WFD, it is more energy efficient, as will be shown in Chapter 6.

Chapter 6

Simulation Results

While we have described the best- and worst-case VFI cost and proposed EDF-hv to mitigate that cost, the best- and worst-case scenarios rarely occur in reality. Hence, we conducted simulations to assess the average VFI cost, as well as the performance of EDF-hv. We will show the simulation results of the average-case behavior of VFI cost first, and then we will show the performance of EDF-hv, including a case-study using the Texas Instrument (TI) C66x Multi-core Digital Signal Processor (DSP).

6.1 VFI Cost

We performed simulations consisting of randomly generated task sets for several different utilizations to assess how the average VFI cost correlates with imbalance.

6.1.1 Simulation Setup

To minimize biasing in the randomly generated task sets, a slightly modified version of the UUniFast algorithm in [40] was used; shown in Algorithm 1. Since the UUniFast algorithm assumes a single core system, it needed to be modified so that the target utilization can exceed 100% while restricting the individual task utilization to 100%. Due to this modification, the UUniFast algorithm occasionally failed to produce a valid task set. When this situation arose, the task set was discarded, and a new task set was randomly generated. In addition, not every valid task set can be partitioned unless the system utilization is less than approximately 50% [1]. Since the VFI cost only applies to partitioned scheduling for HRTS, if a task set was encountered that could not be partitioned, the task set was also discarded and another task set was generated randomly.

The maximum allowable period for a task was set to 100 quanta with a minimum

Algorithm 1 Modified UUniFast

```
$valid = false$;
while $(!valid)$ do
    $sumU = targetU * M$;
    for $(k = 1; k < N; k++)$ do
        $next = sumU - sumU * Random()^{1/(k-N)}$;
        if $(next > 1)$ then
            $next = 1$;
        end if
        $taskSet[k].utilization = next$;
        $sumU = sumU - next$;
    end for
    if $(sumU \leq 1)$ then
        $valid = true$;
        $taskSet[N].utilization = next$;
    end if
end while
```

allowable period of 5 quanta. Additionally, each task was required to have a minimum WCET of one quantum. As such, the minimum allowable task size is 1%, i.e., one quantum with a period of 100 quanta. For implementation purposes, the system utilization was allowed to deviate from the desired utilization by up to 1%. An execution profile was then generated for each task with a random number of probability and execution time entries. The probabilities for each profile were generated using the original UUniFast algorithm. The execution times were similarly generated using a modified version of UUniFast where, for each iteration, the actual execution time was taken as some percentage of the WCET.

In addition to generating tasks, we also generated the available VFI frequencies and corresponding power states. Since the VFI cost is a ratio between the power states, the values of the frequencies and power states themselves are not as important as the ratio between the frequencies and power states. For this set of simulations, cores in a VFI could reduce their frequency to up to 70% of the maximum frequency at 5% granularity. The corresponding power was calculated as $w_k \propto f_k^3$.

For given a number of tasks, number of cores per VFI, and system utilization, 20 task sets were randomly generated. Since each task has a random execution profile, the task sets were simulated three times to obtain meaningful data. Thus, the VFI cost for the given number of tasks, number of cores per VFI, and system utilization is the average of the VFI cost of all 60 simulations, i.e., 20 task sets each simulated 3 times.

Simulations were run for systems configured with two, four, and eight cores per VFI. Each set of simulations only considered one VFI per system, since, as shown in Equation 4.4, the VFI cost of a system is simply the average of the VFI cost of each VFI. Thus, since the simulation is already the average of multiple simulations, the results of M cores on a single VFI are representative of kM cores on k VFIs. Further, to be able to compare the results of the VFI with two, four, and eight cores per VFI, values of N were selected such that the average number of tasks per core, i.e, $\frac{\mathrm{N}}{\mathrm{M}}$, varied from 1.5 to 5 (at intervals of 0.5). Finally, task sets with total utilization between 40% and 90% (at intervals of 10%) were considered. The minimum utilization of 40% was selected since each core can reduce its frequency to 70%, implying that, for a total utilization at or below 30%, the VFI cost is negligible. Conversely, the maximum utilization was set at 90% since partitioning task set with a 100% total utilization is only feasible in rare cases. Once partitioned, tasks are scheduled with EDF and the optimal frequency for each core is calculated using the LADVS algorithm [22].

6.1.2 Results

The VFI cost and imbalance from the simulations are shown for VFI configurations of two, four, and eight cores per VFI in Figure 6.1 and the average of the three VFI configurations are shown in Figure 6.2. Since these graphs are produced from a set of randomly generated data, they do not represent exact values, but can be used to establish general trends. As shown in the graphs, as the utilization increases, the VFI cost also increases. In addition, as the average number of tasks per core increases, both the imbalance and VFI cost are reduced.

Another trend shown in Figure 6.2 is that the VFI cost increases as the number of cores per VFI increases. This behavior follows the mathematical model in Equation 4.6. The mathematical limit for two, four, and eight cores per VFI are 33.3%, 50.0%, and 58.3%, respectively. The maximum simulated VFI cost for each configuration are 23.9%, 37.3%, and

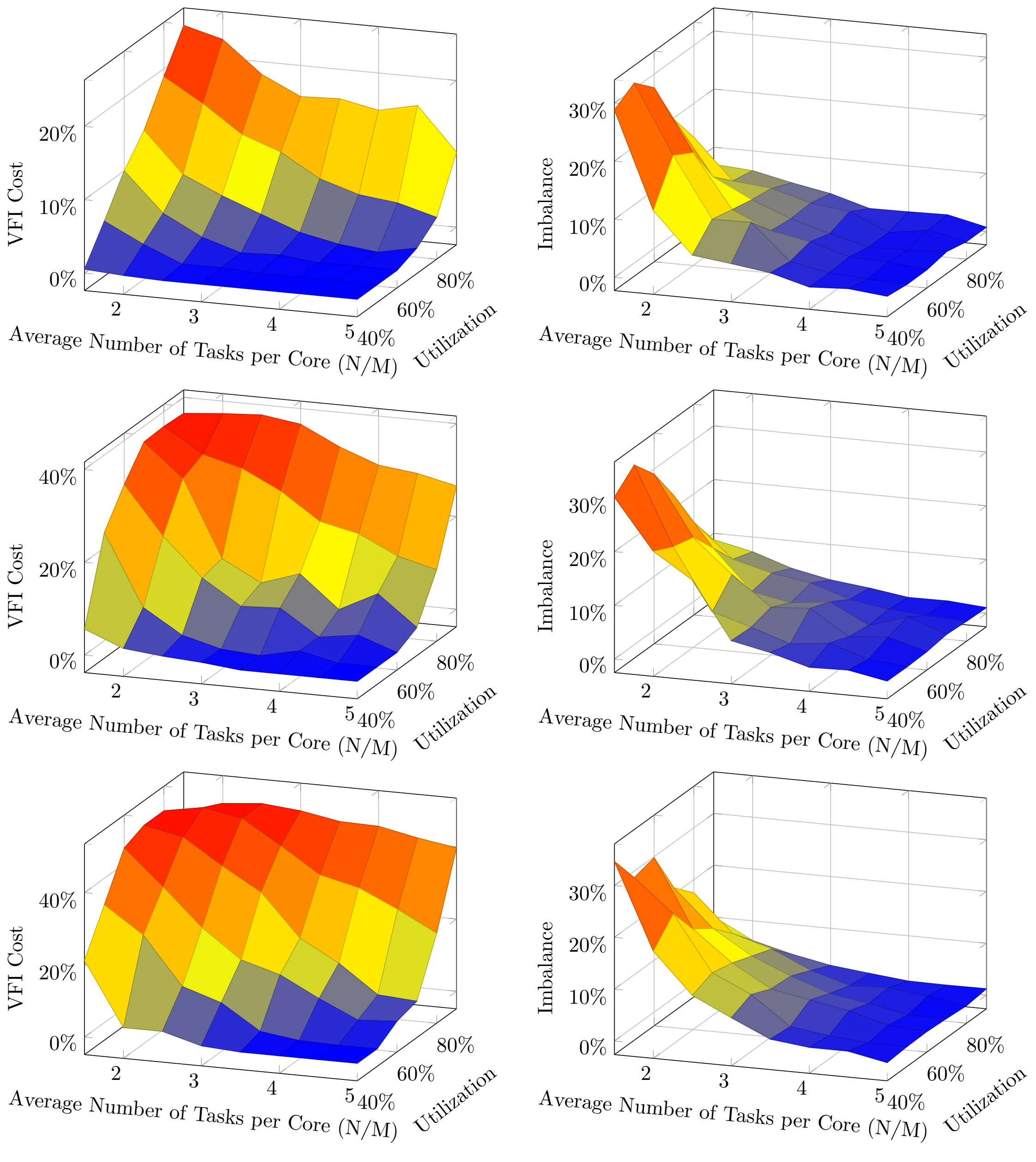


Fig. 6.1: Average VFI cost (left) and imbalance (right) for two (top), four (middle), and eight (bottom) cores, as a function of the average number of tasks per core (i.e., N/M) and system utilization.

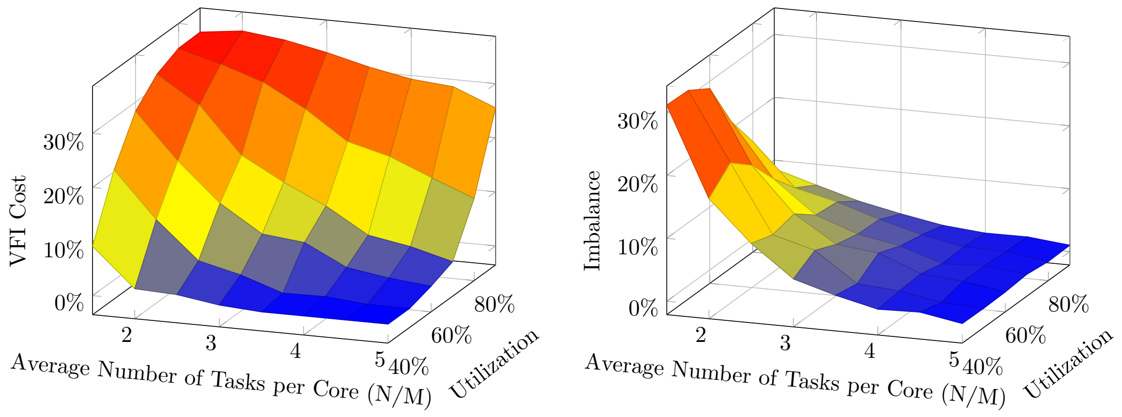


Fig. 6.2: Combined average VFI cost (left) and imbalance (right) (i.e., combined average of two, four, and eight cores per VFI), as a function of the average number of tasks per core (i.e., N/M) and system utilization.

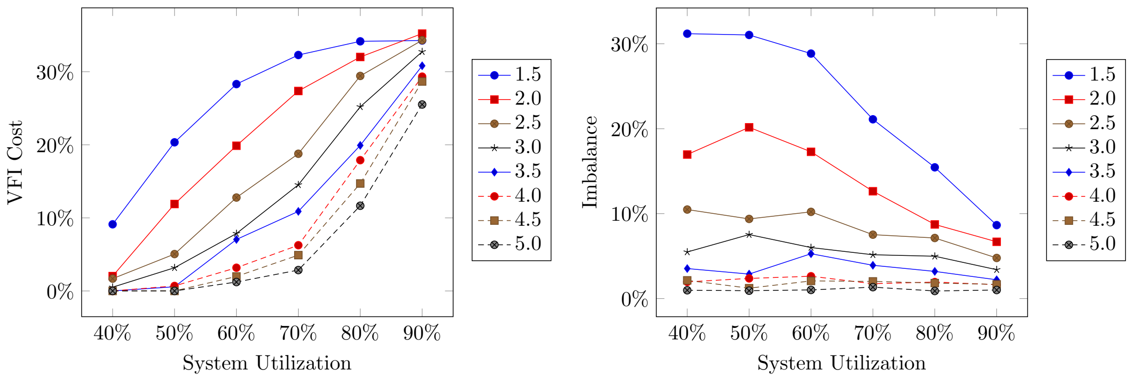


Fig. 6.3: Combined average VFI cost (left) and imbalance (right) for the average number of tasks per core (N/M) by the system utilization.

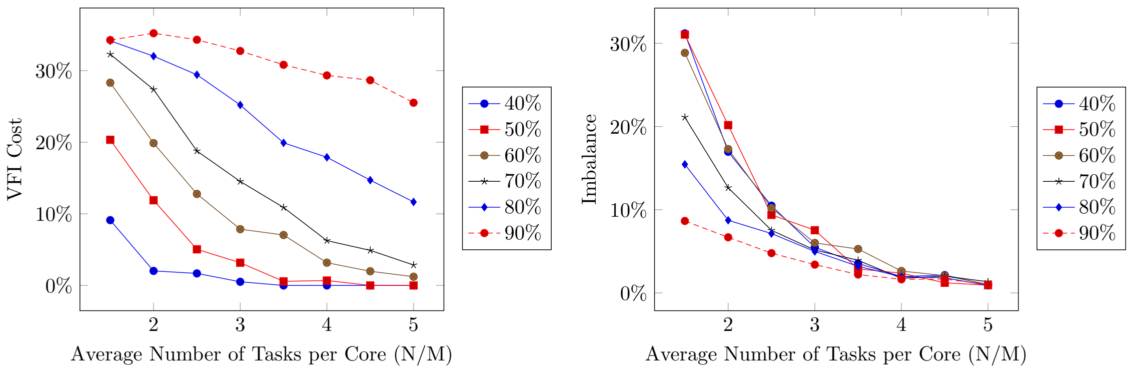


Fig. 6.4: Combined average VFI cost (left) and imbalance (right) for the system utilization by the average number of tasks per core (N/M).

Table 6.1: Reduction in imbalance with the correlated reduction in VFI cost for the given system utilization based on the average of two, four, and eight cores per VFI. The average over the system utilizations shows a nearly one-to-one correlation between imbalance and VFI cost.

Utilization	**Imbalance**	**VFI Cost**
40%	30.2%	9.1%
50%	30.1%	20.3%
60%	27.8%	27.1%
70%	19.8%	29.4%
80%	14.6%	22.5%
90%	7.6%	8.8%
Average	21.7%	19.5%

48.6%, respectively. While the simulated maximum VFI cost for all three VFI configurations occurred at a high utilization with a low average number of tasks per core, the simulated results attained an average of 76.6% of the mathematical limit with the configurations with more cores per VFI being closer to the mathematical limit in all three cases. Further, the maximum simulated VFI cost for all three configurations occurred when the imbalance was at its highest for the given utilization.

A two-dimensional plot of the VFI cost and imbalance by the average number of tasks per core is shown in Figure 6.3. As the average number of tasks increase, the imbalance decreases. Further, for the lower average number of tasks per core, as the utilization increases, the imbalance decreases. This is expected since as the utilization increases, the probability that the task set is not able to be partitioned also increases. Since any randomly generated task set that could not be partitioned was not considered in this work, the amount of imbalance decreases as necessary to be feasible for partitioning.

Let us now consider Figure 6.4, which is a two-dimensional plot of the VFI cost and imbalance by system utilization. Although the VFI cost is clearly dominated by system utilization, there is a strong correlation between the VFI cost and imbalance. The total reduction in imbalance and VFI cost for each utilization is shown in Table 6.1 and the average correlation is near one to one. As such, although the VFI cost is primarily influenced by the utilization, our data suggests that it is also correlated to imbalance and thus can be partially mitigated by load-balancing the system.

6.2 EDF-hv

Now that we have established the average behavior of VFI cost and demonstrated its correlation to imbalance, we will demonstrate the improvement in energy efficiency (i.e. performance) of EDF-hv as compared to WFD. Although it may seem that EDF-hv should be compared to EDF-os, since the former is heavily based on the latter, a fair comparison cannot be made, as EDF-os is intended for soft real-time systems. Therefore, we compare the energy efficiency of EDF-hv against that of WFD as both are suitable for HRTS.

We will assess the performance of EDF-hv in two sets of simulations. For the first set, the performance of EDF-hv was compared to WFD on a theoretical system with two, four, and eight cores per VFI. The second set of simulations is a case-study that explored the performance of EDF-hv on a system modeled after the TI C66x DSP with four cores per VFI. The difference between the theoretical system and the TI C66x DSP system was the number of available v/f pairs. In the theoretical system, the VFI could reduce their frequency to 60% of the maximum frequency with a granularity of 1% (i.e. 40 v/f pairs) in order to emulate the theoretical limits of DVFS. The TI C66x DSP system could reduce frequency to 57.1% of the maximum frequency with a granularity of 10.7% (i.e. 4 v/f pairs) to emulate the actually available frequencies of the TI C66x DSP. The motivation behind the first set of simulations was to establish the performance capabilities of EDF-hv with minimal effects of the frequency profile, while the second set is intended to demonstrate some of the limitations imposed by the hardware. Many modern multi-core processors have significantly more than four v/f pairs, so the TI C66x DSP was specifically chosen due to its limited v/f pairs to demonstrate the effects of the number of v/f pairs on the performance.

We will describe our setup of the two sets of simulations and explain the results for each set next.

6.2.1 Simulation Setup

The simulation setup to demonstrate the performance of EDF-hv is similar to that for VFI cost with four exceptions. First, to establish a baseline for EDF-hv, task execution profiles were not considered, i.e., each task was assumed to require its WCET. Second, instead of generating 20 task for a given system utilization and average number of tasks per core, 100 task sets were generated. Third, the range and granularity of the system utilization for the two sets of simulations differed from that used for the VFI cost. For the theoretical system, system utilizations from 50% to 85% were considered at 5% intervals. The minimum system utilization of 50% was used since simulations on VFI cost showed that the imbalance between 40% and 50% yielded fairly similar results in most cases. The maximum total utilization was reduced to 85%, as only task sets that could meet Theorem 2 were considered, which was not typical of task sets with utilization above 85%. For the TI C66x DSP system, system utilizations from 45% to 85% were considered at 10% intervals to demonstrate the effects of limited v/f pairs on performance as compared to the theoretical system. Finally, as may be obvious, the theoretical and TI C66x DSP systems both used a different frequency profile for the VFI as described in the preceding section.

6.2.2 Performance of EDF-hv

The results comparing EDF-hv to WFD are shown in Figure 6.5 and Figure 6.6. Each graph in Figure 6.5 and Figure 6.6 show the reduction in energy consumption (i.e. the performance) of EDF-hv as compared to WFD for the given average number of tasks per core for the given number of cores. The following five general trends are seen in both figures. First, the amount of imbalance in a system decreases as the number of cores increase. Second, the amount of imbalance in the system decreases as the average number of tasks per core increase. Third, the correlation of imbalance and energy savings decreases as the number of cores increase. Fourth, the correlation of imbalance and energy savings increases as the average number of cores increases. Fifth, the correlation of imbalance and energy savings increases as the utilization of the system increases.

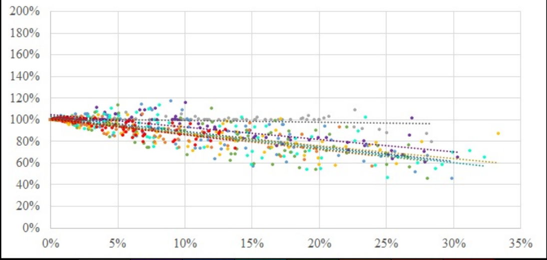
On average EDF-hv consumed 4.8% less energy than WFD. Each plot shows the performance, i.e., the energy consumption decrease of EFD-hv as compared to WFD, of each system versus its imbalance and represents a specific configuration of the number of cores per VFI and an average number of tasks per core. Since it is difficult to determine quantitative values from the plots in Figure 6.5 and Figure 6.6, a table of the average performance

Number of Cores

Average Number of Tasks per Core (N/M)

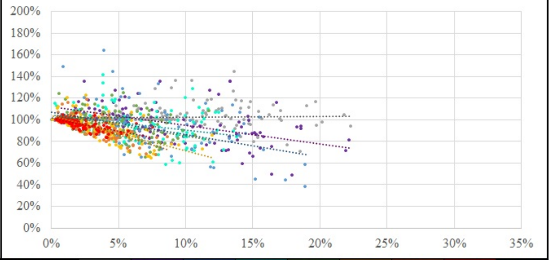
1.5

2



Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	99.7%	94.6%	90.9%	89.1%	88.0%	90.4%	93.2%	95.0%	92.6%
Imbalance	8.8%	8.8%	8.8%	9.4%	9.4%	8.0%	5.3%	4.0%	7.8%

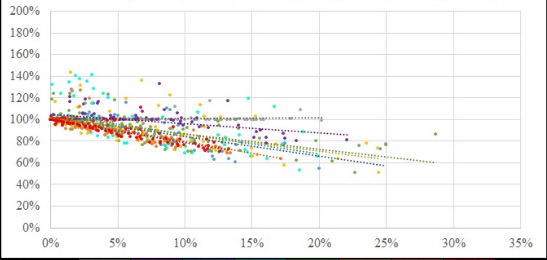
4



Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	101.6%	96.4%	91.9%	95.2%	95.5%	91.2%	93.0%	95.5%	95.0%
Imbalance	7.4%	9.0%	7.2%	6.6%	4.7%	3.9%	3.1%	2.3%	5.5%

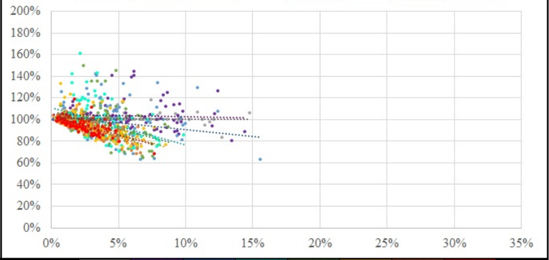
2.0

2



Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.5%	100.2%	92.5%	95.4%	91.9%	92.6%	92.3%	91.4%	94.6%
Imbalance	6.7%	5.3%	5.4%	5.2%	6.2%	6.0%	4.3%	4.2%	5.4%

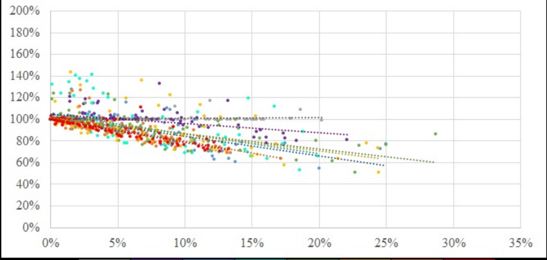
4



Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.7%	103.6%	97.5%	99.2%	94.5%	93.9%	92.8%	94.3%	97.1%
Imbalance	4.9%	5.1%	4.5%	3.4%	4.0%	3.3%	2.6%	2.4%	3.8%

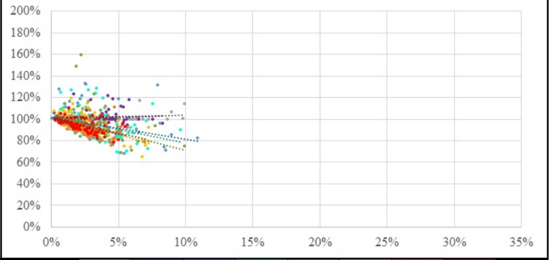
2.5

2



Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.1%	100.0%	94.5%	93.3%	91.2%	92.5%	93.2%	92.9%	94.7%
Imbalance	3.9%	4.6%	3.8%	3.7%	4.2%	3.6%	3.0%	2.9%	3.7%

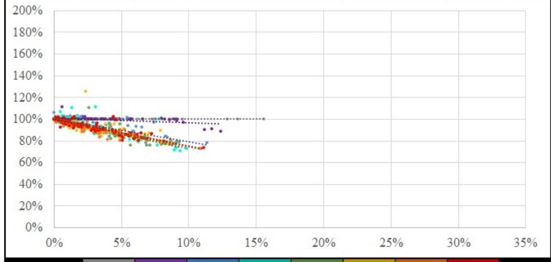
4



Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.4%	101.8%	95.6%	96.1%	95.1%	95.6%	93.1%	94.6%	96.6%
Imbalance	3.3%	3.1%	2.9%	2.9%	2.6%	2.4%	2.5%	2.1%	2.7%

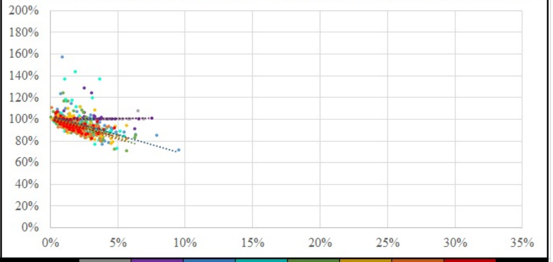
3.0

2



Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.0%	99.8%	94.5%	93.4%	92.5%	92.9%	92.2%	93.2%	94.8%
Imbalance	3.0%	3.0%	2.7%	2.6%	2.8%	2.7%	2.7%	2.6%	2.8%

4



Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.1%	100.7%	95.8%	95.1%	94.7%	94.6%	93.0%	94.8%	96.1%
Imbalance	2.2%	2.1%	2.2%	2.2%	1.9%	1.8%	1.9%	1.6%	2.0%

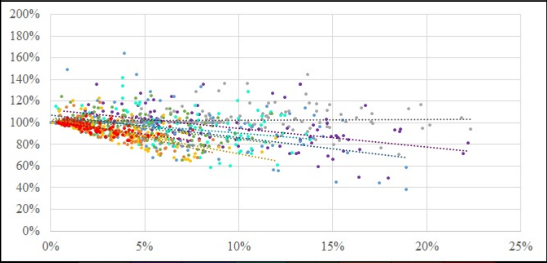
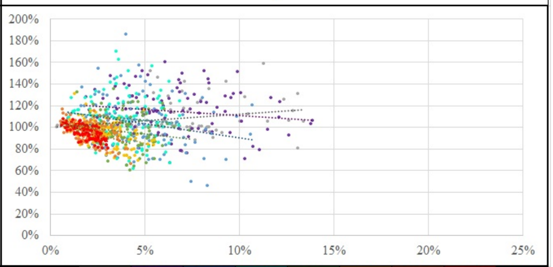
Fig. 6.5: Reduction in energy consumption (i.e. performance) of EDF-hv compared to WFD versus the imbalance of WFD for two- and four-core systems with one VFI and task sets with 1.5, 2.0, 2.5 and 3.0 average number of tasks per core (i.e., N/M).

Number of Cores

4 | 8

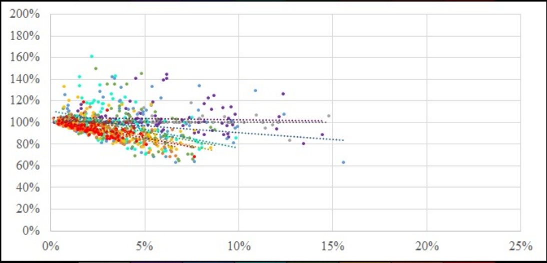
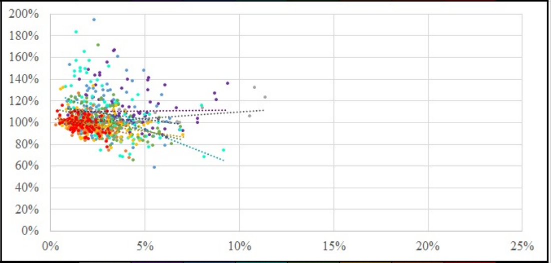
Average Number of Tasks per Core (N/M)

1.5

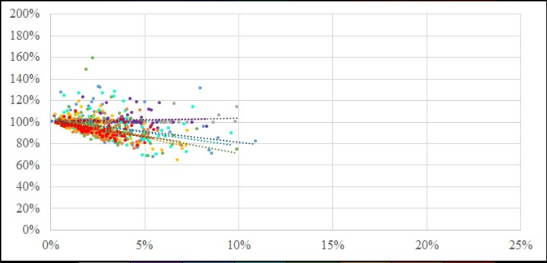
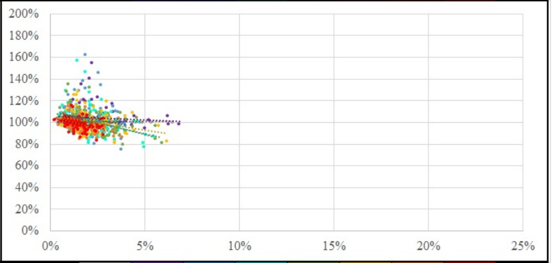
Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total	Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	101.6%	96.4%	91.9%	95.2%	95.5%	91.2%	93.0%	95.5%	95.0%	Performance	107.2%	114.8%	104.0%	108.3%	96.6%	95.2%	95.5%	95.1%	#####
Imbalance	7.4%	9.0%	7.2%	6.6%	4.7%	3.9%	3.1%	2.3%	5.5%	Imbalance	5.9%	6.5%	4.8%	4.0%	3.8%	2.9%	2.2%	1.9%	4.0%

2.0

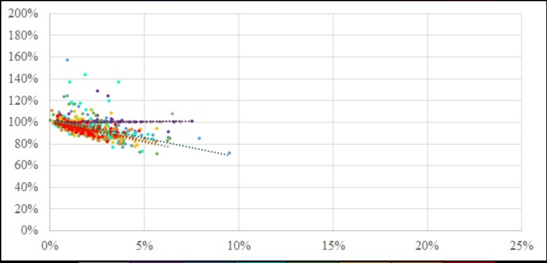
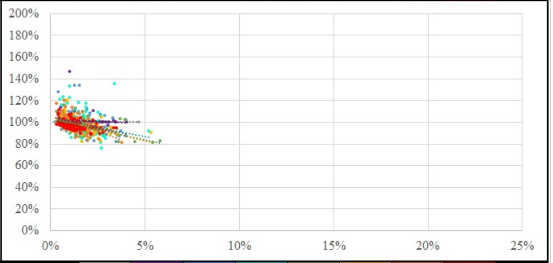
Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total	Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.7%	103.6%	97.5%	99.2%	94.5%	93.9%	92.8%	94.3%	97.1%	Performance	102.1%	110.7%	108.1%	107.1%	101.4%	98.0%	97.6%	98.6%	#####
Imbalance	4.9%	5.1%	4.5%	3.4%	4.0%	3.3%	2.6%	2.4%	3.8%	Imbalance	3.5%	3.6%	3.0%	3.1%	2.9%	2.6%	2.0%	1.7%	2.8%

2.5

Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total	Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.4%	101.8%	95.6%	96.1%	95.1%	95.6%	93.1%	94.6%	96.6%	Performance	100.7%	104.3%	103.5%	101.5%	101.1%	100.7%	98.1%	98.8%	#####
Imbalance	3.3%	3.1%	2.9%	2.9%	2.6%	2.4%	2.5%	2.1%	2.7%	Imbalance	2.2%	2.3%	2.0%	2.3%	2.1%	2.1%	1.7%	1.5%	2.0%

3.0

Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total	Utilization	50%	55%	60%	65%	70%	75%	80%	85%	Total
Performance	100.1%	100.7%	95.8%	95.1%	94.7%	94.6%	93.0%	94.8%	96.1%	Performance	100.0%	101.4%	99.2%	98.4%	96.7%	97.7%	98.0%	97.8%	98.7%
Imbalance	2.2%	2.1%	2.2%	2.2%	1.9%	1.8%	1.9%	1.6%	2.0%	Imbalance	1.5%	1.6%	1.6%	1.6%	1.6%	1.5%	1.4%	1.2%	1.5%

Fig. 6.6: Reduction in energy consumption (i.e. performance) of EDF-hv compared to WFD versus the imbalance of WFD for four- and eight-core systems with one VFI and task sets with 1.5, 2.0, 2.5 and 3.0 average number of tasks per core (i.e., N/M).

and imbalance at each utilization is shown below each plot.

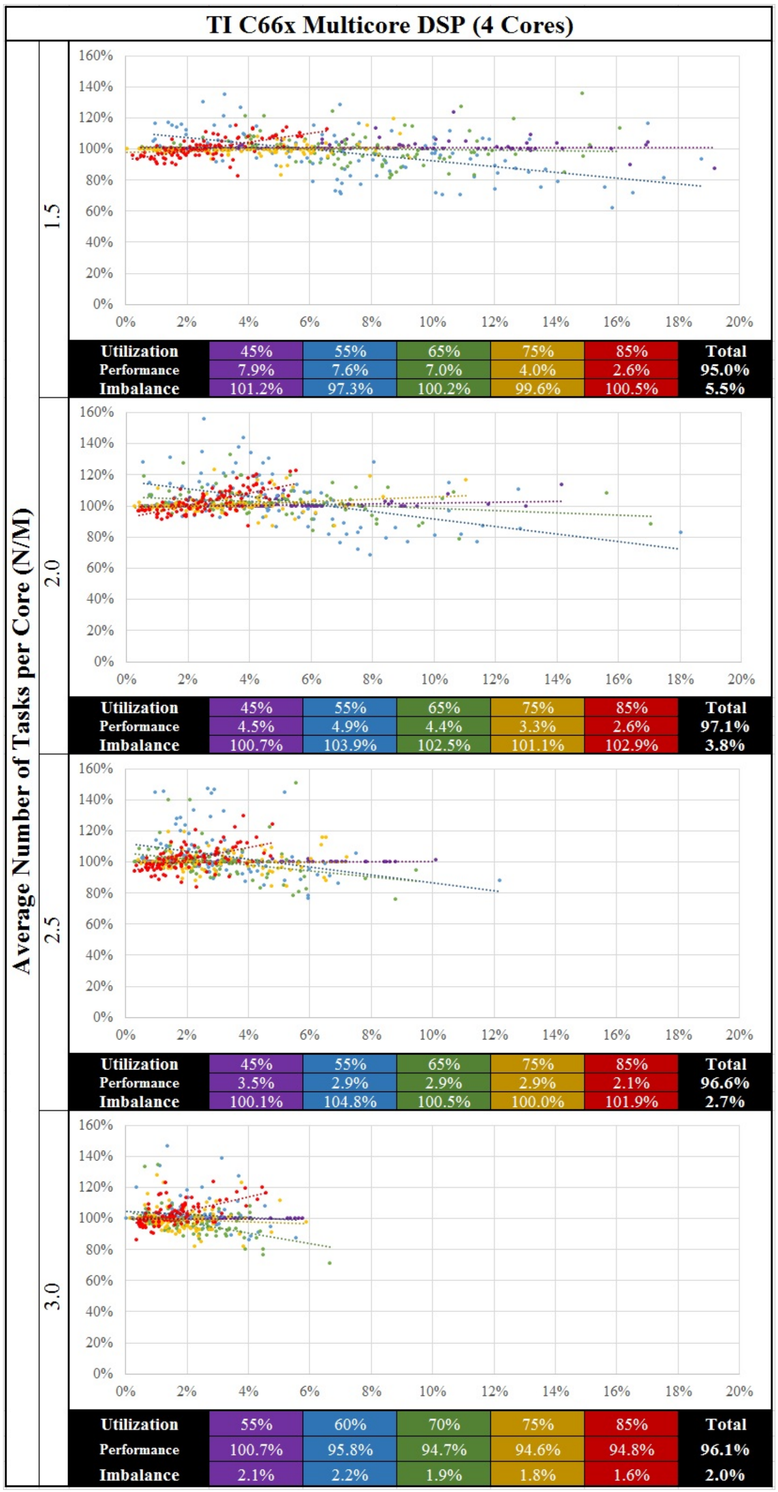
For utilizations less than 60%, EDF-hv performed worse on average than WFD, which was expected since the utilizations of these systems is lower than the VFI could reduce their frequency. However, for utilizations 60% and above, EDF-hv consumed 6.3% less energy than WFD. Further, as shown in Figure 6.5 and Figure 6.6, there is a negative correlation between the amount of imbalance in a task set and the amount of energy conserved by EDF-hv. This is also expected since the primary way that EDF-hv conserves energy is by load-balancing the task set. Additionally, as the average number of tasks per core increased, so did the negative correlation between imbalance and reduction in energy.

Figure 6.5 and Figure 6.6 further show that a reduction in energy consumption around 20% was common with a significant number of task sets achieving even better than that (the maximum observed reduction was 61.6%). For systems with a utilization of 60% or greater and an imbalance of 5% or greater, the average energy consumption was reduced by 15.9%. As such, on average, task sets with greater imbalance typically yielded greater energy reduction.

Unfortunately, some task sets consumed significantly more energy with EDF-hv than with WFD. Typically, such task sets had low imbalance as well as utilization near or below 60%. However, there were cases where there was a reasonable to high imbalance with a utilization well above 60% and where EDF-hv performed worse than WFD. This is likely because, since EDF-hv typically has a maximum fluctuation in utilization greater than the maximum utilization experienced by WFD, as alluded to in Theorem 4, the system selects higher frequencies (and consequently power states) than WFD and thus more energy is consumed.

6.2.3 TI C66x DSP Case-Study

The performance of EDF-hv on a four-core system with one VFI using the frequency profile of the TI C66x DSP is shown in Figure 6.7. Due to the limited number of v/f pairs, the performance is limited as compared to the theoretical system discussed in the previous section. On the TI C66x DSP, EDF-hv performed slightly worse than WFD when applied



Utilization	45%	55%	65%	75%	85%	Total
Performance	7.9%	7.6%	7.0%	4.0%	2.6%	95.0%
Imbalance	101.2%	97.3%	100.2%	99.6%	100.5%	5.5%

Utilization	45%	55%	65%	75%	85%	Total
Performance	4.5%	4.9%	4.4%	3.3%	2.6%	97.1%
Imbalance	100.7%	103.9%	102.5%	101.1%	102.9%	3.8%

Utilization	45%	55%	65%	75%	85%	Total
Performance	3.5%	2.9%	2.9%	2.9%	2.1%	96.6%
Imbalance	100.1%	104.8%	100.5%	100.0%	101.9%	2.7%

Utilization	55%	60%	70%	75%	85%	Total
Performance	100.7%	95.8%	94.7%	94.6%	94.8%	96.1%
Imbalance	2.1%	2.2%	1.9%	1.8%	1.6%	2.0%

Fig. 6.7: Reduction in energy consumption (i.e. performance) of EDF-hv compared to WFD versus the imbalance of WFD for a four-core system with one VFI using the frequency profile of the TI C66x DSP and task sets with 1.5, 2.0, 2.5 and 3.0 average number of tasks per core (i.e., N/M).

generally; consuming just under 1% more energy. When applied only to systems with at least 5% of imbalance, EDF-hv only performed 2.1% better than WFD, which is a dramatic decrease in performance as compared to the 15.9% observed in the theoretical system.

Looking at the trends of the data to explain this dramatic decrease in performance on the TI C66x DSP system, task sets with a system utilization of 45%, 55%, and 65% performed comparable to the theoretical system, while task sets with 75% system utilization performed about the same with EDF-hv and WFD regardless of imbalance and task sets with 85% system utilization tended to perform worse with EDF-hv as imbalance increased. The trend that the performance of EDF-hv got worse for higher utilizations of EDF-hv differed from that of the theoretical system, which saw the correlation between the performance of EDF-hv and imbalance increase as system utilization increased. This is likely due to the fact that EDF-hv will experience fluctuations in utilization higher than WFD as a result of the migrating tasks, combined with the fact that systems with high utilization and tasks that require their full WCET to execute will rarely use the v/f pairs significantly below the system utilization. Thus, on the TI C66x DSP, task sets with 85% typically only used the highest two v/f pairs, and, as imbalance increased, more time was spent at the highest v/f pair. Therefore, as imbalance increased, the total energy consumption increased as a result. This suggests that in systems with limited v/f pairs and high system utilization, EDF-hv is less effective than WFD as the fluctuations in utilization due to migrating tasks increase the energy consumption above that which is saved by load-balancing. When considering only task sets with at least 5% imbalance and a system utilization of 55% and 65%, the performance of EDF-hv was 8.4% better than WFD. Thus, reducing the number of v/f pairs by a factor of 10 showed a reduced the performance nearly 50%. Thus, it can be observed that EDF-hv is best suited for hardware with a high number of available v/f pairs.

One final observation regarding the TI C66x DSP system is that although it may seem constrictive that EDF-hv only improved performance for task sets with a system utilization of 55% or 65%, in real-world systems, EDF-hv is likely applicable to systems with much higher utilizations. As mentioned, for these simulations, task execution profiles were not considered and thus, task were assumed to need there WCET. However, typically, tasks do not require their WCET, and thus, although the WCET must be accommodated to guarantee deadlines, the real-world average utilization is likely to be significantly less. As such, even though the WCET may show a system utilization of 85%, the actual utilization may be much lower, which means that systems with high utilization, which are most desirable to decrease static power cost, will typically perform better with EDF-hv.

Chapter 7
Conclusion and Future Work

The best- and worst-case VFI cost have been derived, and both the mathematical model and simulation results suggest that the VFI cost can be a significant contributor to the overall energy consumption. The proposed algorithm, EDF-hv, can help to mitigate VFI cost in HRTS with multiple cores per VFI. Our results also show that the energy savings from EDF-hv can be significant, especially for systems with high imbalance and high utilizations.

Several aspects of this work can be further explored to improve energy efficiency. First, a task scheduling algorithm as well as a DVFS algorithm that is explicitly designed to accommodate migrating tasks can be developed. Second, further exploration into the relationship between VFI cost and imbalance may provide insight into ways to improve EDF-hv. A third area of research is to remove the boundary limited property of EDF-hv. This would increase the scheduling overhead, which would increase energy consumption, however, removing the boundary limited property could yield energy saving to justify the increased overhead. Fourth, while this thesis has explored the energy cost associated with VFI from a dynamic power perspective, there are aspects of static power relating to VFI that could further be explored. Finally, in addition to saving energy, load-balancing with EDF-hv may have some advantages for improving the wear state of cores.

References

[1] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *JAMC*, vol. 20, pp. 46–61, 1973.

[2] S. Cheng, J. A. Stankovic, and K. Ramamritham, “Dynamic scheduling of groups of tasks with precedence constraints in distributed hard real-time systems,” in *Proceedings of IEEE Real-Time Systems Symposium*, Dec. 1986, pp. 166–174.

[3] J. Lehoczky, L. Sha, and Y. Ding, “The rate monotonic scheduling algorithm: exact characterization and average case behavior,” in *Proceedings of IEEE Real-Time Systems Symposium*, Dec. 1989, pp. 166–171.

[4] G. C. Buttazzo, “Rate monotonic vs. EDF: Judgment day,” *Real-Time Systems*, vol. 29, pp. 5–26, 2005.

[5] H. M. Goldberg, “Jackson’s conjecture on earliest due date scheduling,” *Mathematics of Operations Research*, vol. 5, pp. 460–466, 1980.

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

[7] T. P. Baker and S. K. Baruah, “An analysis of global EDF schedulability forarbitrary-deadline sporadic task systems,” *Real-Time Systems*, vol. 43, pp. 3–24, 2009.

[8] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, “On the competitiveness of on-line real-time task scheduling,” *Real-Time Systems*, vol. 4, pp. 125–144, 1992.

[9] S. Baruah, “Scheduling periodic tasks on uniform multiprocessors,” *Information Processing Letters*, vol. 80, pp. 97–104, 2001.

[10] H. Cho, B. Ravindran, and E. D. Jensen, “An optimal real-time scheduling algorithm for multiprocessors,” in *Proceedings of IEEE Real-Time Systems Symposium*, Dec. 2006, pp. 101–110.

[11] U. C. Devi, “Soft real-time scheduling on multiprocessors,” Ph.D. dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, 2006.

[12] N. Fisher, J. Goossens, and S. Baruah, “Optimal online multiprocessor scheduling of sporadic real-time tasks is impossible,” *Real-Time Systems*, vol. 45, pp. 26–71, 2010.

[13] P. Choudhary and D. Marculescu, “Power management of voltage/frequency island-based systems using hardware-based methods,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, pp. 427–438, 2009.

[14] E. L. Sueur and G. Heiser, “Dynamic voltage and frequency scaling: The laws of diminishing returns,” in *Proceedings of the 2010 international conference on Power aware computing and systems*, Oct. 2010, pp. 1–8.

[15] S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems," *International Journal of Computer Aided Engineering and Technology*, vol. 6, pp. 440–459, 2014.

[16] P.-E. Gaillardon, E. Beigne, S. Lesecq, and G. D. Micheli, "A survey on low-power techniques with emerging technologies: from devices to systems," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, pp. 12.1–12.26, 2015.

[17] A. Csendes, "Survey of dynamic voltage scaling methods for energy efficient embedded systems," in *Proceedings of the 8th International Conference on Applied Informatics*, Jan. 2010, pp. 413–420.

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

[19] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *Proceedings of the 41st Annual Design Automation Conference*, 2004, pp. 275–280.

[20] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo, "Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems," in *Real-Time and Embedded Technology and Applications Symposium*, Apr. 2006, pp. 408–417.

[21] J.-J. Chen and T.-W. Kuo, "Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems," in *IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2007, pp. 289–294.

[22] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, Dec. 2001, pp. 89–102.

[23] T. Zitterell and C. Scholl, "A probabilistic and energy-efficient scheduling approach for online application in real-time systems," in *Proceedings of the 47th Design Automation Conference*, Jun. 2010, pp. 42–47.

[24] D. Zhu, H. Aydin, and J. J. Chen, "Optimistic reliability aware energy management for real-time tasks with probabilistic execution times," in *Real-Time Systems Symposium*, Nov. 2008, pp. 313–322.

[25] C. Xian, Y. H. Lu, and Z. Li, "Dynamic voltage scaling for multitasking real-time systems with uncertain execution time," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 1467–1478, 2008.

[26] T. A. AlEnawy and H. Aydin, "Energy-constrained scheduling for weakly-hard real-time systems," in *Proceedings of IEEE Real-Time Systems Symposium*, Dec. 2005, pp. 376–385.

[27] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," in *Design Automation Conference, 2001. Proceedings*, Jun. 2001, pp. 828–833.

[28] F. Yao, A. Demers, and S. Shenker, “A scheduling model for reduced cpu energy,” in *Proceedings of Foundations of Computer Science*, Oct. 1995, pp. 374–382.

[29] X. Zhong and C.-Z. Xu, “Energy-aware modeling and scheduling of real-time tasks for dynamic voltage scaling,” in *Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International*, Dec. 2005, pp. 358–372.

[30] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. H. Anderson, and S. Baruah, “A categorization of real-time multiprocessor scheduling problems and algorithms,” *Handbook on scheduling algorithms, methods, and models*, pp. 30.1–30.19, 2004.

[31] B. Andersson and E. Tovar, “Multiprocessor scheduling with few preemptions,” in *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA’06)*, Aug. 2006, pp. 322–334.

[32] J. H. Anderson, V. Bud, and U. C. Devi, “An EDF-based restricted-migration scheduling algorithm for multiprocessor soft real-time systems,” *Real-Time Systems*, vol. 38, pp. 85–131, 2008.

[33] S. Kato, N. Yamasaki, and Y. Ishikawa, “Semi-partitioned scheduling of sporadic task systems on multiprocessors,” in *21st Euromicro Conference on Real-Time Systems*, Jul. 2009, pp. 249–258.

[34] F. Dorin, P. M. Yomsi, J. Goossens, and P. Richard, “Semi-partitioned hard real-time scheduling with restricted migrations upon identical multiprocessor platforms,” *Computing Research Repository (CoRR)*, vol. 1006.2637, 2010.

[35] A. Bastoni, B. B. Brandenburg, and J. H. Anderson, “Is semi-partitioned scheduling practical?” in *23rd Euromicro Conference on Real-Time Systems*, Jul. 2011, pp. 125–135.

[36] M. Naghibzadeh, P. Neamatollahi, R. Ramezani, A. Rezaeian, and T. Dehghani, “Efficient semi-partitioning and rate-monotonic scheduling hard real-time tasks on multi-core systems,” in *8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, Jun. 2013, pp. 85–88.

[37] J. H. Anderson, J. P. Erickson, U. C. Devi, and B. N. Casses, “Optimal semi-partitioned scheduling in soft real-time systems,” in *International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug. 2014, pp. 1–10.

[38] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. H. Anderson, and S. Baruah, “Fair multiprocessor scheduling,” *Handbook on scheduling algorithms, methods, and models*, pp. 31.1–31.21, 2004.

[39] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, “Proportionate progress: A notion of fairness in resource allocation,” *Algorithmica*, vol. 15, pp. 600–625, 1996.

[40] E. Bini and G. C. Buttazzo, “Biasing effects in schedulability measures,” in *Proceedings of Euromicro Conference on Real-Time Systems*, Jun. 2004, pp. 196–203.

Appendices

Appendix A

Handling Oversized Tasks

Thus far, EDF-hv has only considered systems where the utilization of each task is less than or equal to the utilization that each core needs to be assigned to be load-balanced, i.e. $u_i \leq \Phi$, however, traditionally, tasks can have a utilization up to 100% of a core's capacity. It follows then that one complication of load-balancing is the possibility that a task set to be scheduled may contain one or more tasks with a utilization greater than Φ. As such, in this appendix, we will explore how EDF-hv can cope with task sets where at least one task has a utilization such that $u_i > \Phi$.

When a task set contains at least one task such that $u_i > \Phi$, EDF-hv now has to attempt to schedule a task that has a utilization greater than the available capacity of any of the cores. As such, herein, these tasks are referred to as "oversized tasks", which is formally defined by the following definition.

Definition 1. *An oversized task is a task with a utilization such that: $u_i > \Phi$.*

Another way to consider oversized tasks is by combining Definition 1 with Equation 4.1:

$$u_i > \Phi = \frac{\mathrm{U}}{\mathrm{M}}$$

$$\frac{u_i}{\mathrm{U}} > \frac{1}{\mathrm{M}} = \mathrm{M}^{-1}. \tag{A.1}$$

Therefore, an oversized task is any task with a utilization that comprises more than M^{-1} of the total system utilization.

Oversized tasks impede the ability of EDF-hv to achieve a load-balance and thereby threaten to increase energy consumption. To cope with the presence of oversized tasks, this appendix derives bounds for oversized tasks and explores three methods, namely pre-partitioning, oversized migration, and multi-partitioning, to handle oversized tasks to attain, or at lease approach, a load-balance.

A.1 Bounds with Oversized Tasks

To understand the impact of oversized tasks in EDF-hv, we first explore the properties of oversized tasks and derive bounds on the system utilization, the deviation in utilization of oversized tasks for systems with multiple oversized tasks, and the maximum number of oversized tasks that can be present in a system.

For ease of discussion, let σ be the number of oversized tasks in the system.

Lemma 3. *If a system has any oversized tasks (i.e. $\sigma > 0$), then the first σ tasks in η are oversized and any task after the first σ are not oversized.*

Proof. Recall that the tasks in η are sorted in non-ascending order based on the utilizations of the tasks. Further, by Definition 1, an oversized task has a utilization strictly greater than Φ, while, conversely, a non-oversized tasks have a utilization less than or equal to Φ. Therefore, if a system has any oversized tasks, they are the tasks with the greatest utilization and, thus, must be sorted before any non-oversized tasks in η. □

Lemma 4. *The number of oversized tasks is strictly less than the number of tasks:*

$$\sigma < \mathrm{N}.$$

Proof. By counter example, consider when $\sigma \geq \mathrm{N}$. First, if $\sigma > \mathrm{N}$, then the number of oversized tasks in the task set is greater than the number of tasks in the task set, which is clearly not possible. Second, the case where $\sigma = \mathrm{N}$, then $u_{\mathrm{N}} = u_{\sigma}$, which, by Definition 1 and from Equation 3.4 and Equation 4.1:

$$u_{\mathrm{N}} > \Phi = \frac{\mathrm{U}}{\mathrm{M}} = \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}}$$

$$\mathrm{M} \cdot u_{\mathrm{N}} > \sum_{i=1}^{\mathrm{N}} u_i$$

Recall that, by Equation 3.3, tasks are sorted in non-ascending order by utilization such that u_{N} must be the smallest task by utilization in the system, if one exists. Thus:

$$\sum_{i=1}^{\mathrm{N}} u_i \geq \mathrm{N} \cdot u_{\mathrm{N}},$$

which yields that:

$$\mathrm{M} \cdot u_{\mathrm{N}} > \sum_{i=1}^{\mathrm{N}} u_i \geq \mathrm{N} \cdot u_{\mathrm{N}}$$

$$\mathrm{M} \cdot u_{\mathrm{N}} > \mathrm{N} \cdot u_{\mathrm{N}}$$

$$\mathrm{M} \cdot u_{\mathrm{N}} - \mathrm{N} \cdot u_{\mathrm{N}} = (\mathrm{M} - \mathrm{N})u_{\mathrm{N}} > 0,$$

which, since the utilization must be a positive value, is only true if $\mathrm{N} < \mathrm{M}$. However, by Caveat 1, $\mathrm{N} > \mathrm{M}$, therefore, since the number of oversized tasks cannot be greater than or equal to the number of tasks, $\sigma < \mathrm{N}$. $\square$

Utilization Bound of Oversized Tasks

We will now show the utilization bound for oversized tasks. Considering the simplest case where a system has only one oversized task (i.e. $\sigma = 1$), then τ_1 is oversized and, thus, by Definition 1 and Equation 3.4:

$$u_1 > \Phi = \frac{\mathrm{U}}{\mathrm{M}} = \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}}.$$

Since τ_1 is the first task in η:

$$u_1 > \frac{u_1 + \sum_{i=2}^{\mathrm{N}} u_i}{\mathrm{M}} = \frac{u_1}{\mathrm{M}} + \frac{\sum_{i=2}^{\mathrm{N}} u_i}{\mathrm{M}}$$

$$u_1 - \frac{u_1}{\mathrm{M}} = \frac{(\mathrm{M}-1)u_1}{\mathrm{M}} > \frac{\sum_{i=2}^{\mathrm{N}} u_i}{\mathrm{M}}$$

$$(\mathrm{M}-1)u_1 > \sum_{i=2}^{\mathrm{N}} u_i$$

$$u_1 > \frac{\sum_{i=2}^{\mathrm{N}} u_i}{\mathrm{M}-1} = \frac{\mathrm{U}-u_1}{\mathrm{M}-1}. \tag{A.2}$$

Thus, from Equation A.2, if the first task in a system is oversized than the utilization of the oversized task must be greater than the utilization of the other tasks divided across all but one of the cores.

To extend Equation A.2 for a system with σ oversized tasks, consider first the relationship between oversized tasks. Since task are sorted in non-ascending order by Equation 3.3:

$$u_1 \geq u_2 \geq ... \geq u_\sigma > \Phi. \tag{A.3}$$

As such, u_σ is the smallest of the oversized tasks, but can be equal to the other oversized tasks. Therefore:

$$u_1 \geq ... \geq u_\sigma > \Phi = \frac{\mathrm{U}}{\mathrm{M}} = \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}}$$

$$u_\sigma > \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}} = \frac{\sum_{i=1}^{\sigma} u_i + \sum_{i=\sigma+1}^{\mathrm{N}} u_i}{\mathrm{M}} = \frac{\sum_{i=1}^{\sigma} u_i}{\mathrm{M}} + \frac{\sum_{i=\sigma+1}^{\mathrm{N}} u_i}{\mathrm{M}}$$

$$u_\sigma - \frac{\sum_{i=1}^{\sigma} u_i}{\mathrm{M}} = \frac{\mathrm{M}\cdot u_\sigma - \sum_{i=1}^{\sigma} u_i}{\mathrm{M}} > \frac{\sum_{i=\sigma+1}^{\mathrm{N}} u_i}{\mathrm{M}}$$

$$\mathrm{M}\cdot u_\sigma - \sum_{i=1}^{\sigma} u_i > \sum_{i=\sigma+1}^{\mathrm{N}} \cdot u_i$$

$$\mathrm{M}\cdot u_\sigma - \sum_{i=\sigma+1}^{\mathrm{N}} u_i > \sum_{i=1}^{\sigma} u_i. \tag{A.4}$$

As such, the utilization of the oversized tasks must be less than M times the smallest oversized task minus the sum of the non-oversized tasks. Although this may seem overly complicated for practical use now, Equation A.4 will be used in the next two subsections to bound the amount of deviation that can exist in the utilization of oversized tasks and the maximum number of oversized tasks in a system.

Maximum Deviation in the Utilization of Oversized Tasks

Now we consider how much greater one oversized task can be than another in terms of utilization when there are multiple oversized tasks in a system. More specifically, how much greater can each oversized task be than u_σ (i.e. the smallest of the oversized tasks) or, more formally:

$$\sum_{i=1}^{\sigma}(u_i - u_\sigma). \tag{A.5}$$

From Equation A.5, the sum of the utilization of the oversized tasks can be written as:

$$\sum_{i=1}^{\sigma} u_i = \sigma \cdot u_\sigma + \sum_{i=1}^{\sigma}(u_i - u_\sigma). \tag{A.6}$$

Now, substituting Equation A.6 into Equation A.4:

$$\mathrm{M} \cdot u_\sigma - \sum_{i=\sigma+1}^{\mathrm{N}} u_i > \sigma \cdot u_\sigma + \sum_{i=1}^{\sigma}(u_i - u_\sigma)$$

$$\mathrm{M} \cdot u_\sigma - \sigma \cdot u_\sigma - \sum_{i=\sigma+1}^{\mathrm{N}} u_i > \sum_{i=1}^{\sigma}(u_i - u_\sigma)$$

$$\sum_{i=1}^{\sigma}(u_i - u_\sigma) < (\mathrm{M} - \sigma)u_\sigma - \sum_{i=\sigma+1}^{\mathrm{N}} u_i. \tag{A.7}$$

Therefore, the deviation of oversized tasks is dependent on the number of cores, the number of oversized tasks, and the utilization of the non-oversized tasks.

Maximum Number of Oversized Tasks

As may be observed from Equation A.7, if the number of oversized tasks is greater than M, then the deviation from the smallest oversized task must be a negative number, which would violate Equation A.3. It follows that the number of oversized tasks cannot exceed the number of cores, as is described in the following Theorem.

Theorem 5. *The number of oversized tasks, σ, is strictly less than the number of cores,* M*:*

$$\sigma < \mathrm{M}$$

Proof. Since the utilization of each task is a positive quantity, the sum of utilizations must be a positive quantity. Thus, by counter example, if $\sigma \geq \mathrm{M}$, then:

$$\mathrm{M} - \sigma \leq 0,$$

which means:

$$(\mathrm{M} - \sigma)u_\sigma \leq 0,$$

since u_σ is a positive value. Further, recall from Lemma 4 that $\mathrm{N} > \sigma$, and thus:

$$\sum_{i=\sigma+1}^{\mathrm{N}} u_i > 0,$$

since there must be at least one non-oversized task in the systems. Therefore, considering Equation A.7:

$$\sum_{i=1}^{\sigma}(u_i - u_\sigma) < (\mathrm{M} - \sigma)u_\sigma - \sum_{i=\sigma+1}^{\mathrm{N}} u_i < 0$$

$$\sum_{i=1}^{\sigma}(u_i - u_\sigma) = \sum_{i=1}^{\sigma} u_i - \sum_{i=1}^{\sigma} u_\sigma < 0$$

$$\sum_{i=1}^{\sigma} u_i < \sum_{i=1}^{\sigma} u_\sigma = \sigma \cdot u_\sigma,$$

which is untrue since, by Equation A.3, u_σ is less than or equal to the utilization of any of the other oversized tasks and, thus, the sum of the utilization of the oversized tasks cannot be strictly less than the number of oversized tasks times the utilization of the smallest oversized task. Therefore, the number of oversized tasks must be less than the number of cores. □

From the system model and Theorem 5, the relationship between the number of tasks, the number of cores, and the number of oversized tasks can be determined for EDF-hv.

Lemma 5. *In EDF-hv, there must be at least two non-oversized tasks in the system.*

Proof. Since N $>$ M (by Caveat 1), M $>$ 1 (since EDF-hv is for multi-core systems), and, M $> \sigma$ (by Theorem 5), then N $>$ M $> \sigma$. Thus, since N, M, and σ are positive integer values, N must be at least $(\sigma + 2)$. Therefore, there must be at least two non-oversized tasks in the system. $\square$

A.2 Methods to Handle Oversized Tasks

Now that we have established bounds on the oversized tasks, we will explore different methods to handle systems with oversized tasks in the context of EDF-hv with the intent to determine the best method. Note that in this section we assume that $\sigma > 0$ for all systems (otherwise there are no oversized tasks).

We investigated three methods for handling oversized tasks; namely, pre-partitioning, oversized migration, and multi-partitioning. In the remainder of this appendix, first we will present the metrics to evaluate each method, then, each method will be defined and investigated using the bounds derived in the previous section, and finally, each method will be evaluated based upon the three metrics.

Metrics for Methods to Handle Oversized Tasks

Since oversized tasks impede the ability of EDF-hv to load-balance the system, it is desirable that a method to handle oversized tasks be able to reduce the imbalance in a system such that it is load-balanced (i.e. $\phi = 0$). As such, the first metric is how much the imbalance in the system is reduced. As mentioned, for this metric, it is desirable that the imbalance in the system be zero, but we will consider methods that at least reduce the imbalance in the system.

For a second metric, recall that the purpose of EDF-hv is to reduce energy consumption, and thus, if a method to handle oversized tasks threatens to increase energy consumption, then there is little purpose to consider it for EDF-hv. Recall further that although the average utilization is load-balanced in EDF-hv, each core experiences some fluctuations in utilization. If these fluctuations are increased, then the maximum utilization that a core may experience can be increased; which threatens to increase energy consumption as the DVFS algorithm may have to select higher v/f pairs in order to guarantee deadlines. As such, if a method to handle oversized tasks significantly increases the fluctuations in utilization, then the energy consumption may also increase. It follows then that the second metric is how much a method may increase fluctuations in utilization.

Finally, it is desirable that a method is able to be applied generally to handle oversized tasks. As such, a third metric is if the method imposes additional requirements on EDF-hv. This metric includes if the requirement on the number of tasks or number of cores is increased or if the method impedes the probability of being able to guarantee deadlines (i.e. Theorem 2).

Pre-Partitioning

Pre-partitioning follows the idea that although load-balancing all cores is optimal, energy savings can likely be attained by load-balancing as many cores as possible. As such, pre-partitioning assigns the oversized tasks to one or more of the cores before scheduling with EDF-hv. In pre-partitioning, if the first task, τ_1, in the task set, η, is oversized, τ_1 is assigned to p_1 before EDF-hv partitions η to the set of cores, P. Now, since τ_1 is assigned p_1, τ_1 is removed from η, such that $\eta^{'} = \{\tau_2, \tau_3, ..., \tau_{\mathrm{N}}\}$, because τ_1 has already been scheduled on a core. Further, since $u_1 > \Phi$, then $U_1 > \Phi$ (since τ_1 has been assigned to it) and, thus, to stay as close to a load-balance as possible, p_1 is removed from P, such that $P^{'} = \{p_2, p_3, ..., p_{\mathrm{M}}\}$, so that no further tasks will be assigned a share on p_1. Now, EDF-hv attempts to schedule the tasks in $\eta^{'}$ to the set of cores in $P^{'}$, however, since Φ was calculated considering the task set η on P, a new target utilization, $\Phi^{'}$, needs to be calculated based on $\eta^{'}$ on $P^{'}$. The utilization of $\eta^{'}$ and the number of cores in $P^{'}$ is given

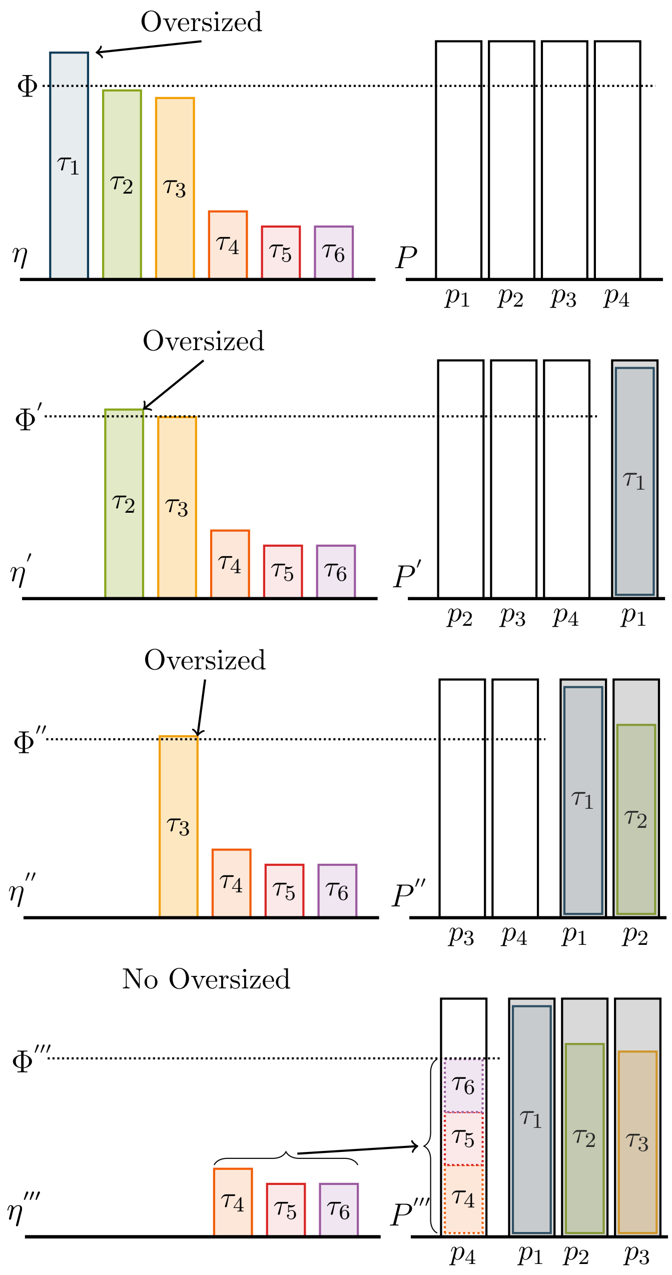


Fig. A.1: An example of cascading oversized tasks. The original task set, η (top; where each task is represented by its utilization), has only one oversized task, τ_1. When τ_1 is pre-partitioned, τ_2 becomes an oversized task in $\eta^{'}$, even though τ_2 was not an oversized task in η. Further, when τ_2 is pre-partitioned, τ_3 becomes oversized in $\eta^{''}$. At this point, the system has $(\mathrm{M}-1) = (4-1) = 3$ oversized task (i.e. the maximum number of oversized tasks in a 4 core system) in the system. Thus, $\eta^{'''}$ does not have oversized tasks, so the remaining tasks are assigned to the remaining core (bottom).

by:

$$\mathrm{U}^{'} = \mathrm{U} - u_1 = \sum_{i=2}^{\mathrm{N}} u_i \tag{A.8}$$

$$\mathrm{N}^{'} = \mathrm{N} - 1 \tag{A.9}$$

$$\mathrm{M}^{'} = \mathrm{M} - 1 \tag{A.10}$$

Thus, by removing τ_1 from η and p_1 from P, the property of EDF-hv that $\mathrm{N} > \mathrm{M}$ is ensured because both N and M are decremented by 1. Further, the relationship between Φ and $\Phi^{'}$ is given by the following theorem.

Theorem 6. *After a system has been pre-partitioned, the resulting target utilization, $\Phi^{'}$, to achieve a load-balance for the tasks in $\eta^{'}$ to the set of cores in $P^{'}$ is strictly less than the original target utilization, i.e.: $\Phi > \Phi^{'}$*

Proof. From Equation A.3, if there are any oversized tasks in η:

$$u_1 > \Phi = \frac{\mathrm{U}}{\mathrm{M}}$$

$$\mathrm{M} \cdot u_1 > \mathrm{U}$$

$$-\mathrm{U} > -\mathrm{M} \cdot u_1.$$

Now, adding $\mathrm{U} \cdot \mathrm{M}$ to both sides of the inequality:

$$\mathrm{U} \cdot \mathrm{M} - \mathrm{U} > \mathrm{U} \cdot \mathrm{M} - \mathrm{M} \cdot u_1$$

$$\mathrm{U}(\mathrm{M} - 1) > \mathrm{M}(\mathrm{U} - u_1).$$

From Equation A.8 and Equation A.10:

$$\mathrm{U} \cdot \mathrm{M}^{'} > \mathrm{M} \cdot \mathrm{U}^{'}$$

$$\frac{\mathrm{U}}{\mathrm{M}} > \frac{\mathrm{U}^{'}}{\mathrm{M}^{'}}.$$

Therefore, from Equation 4.1: $\Phi > \Phi^{'}$. $\square$

Unfortunately, since $\Phi > \Phi^{'}$, if there were multiple oversized tasks in η, then there will still be oversized tasks in $\eta^{'}$ as, from Equation A.3 and Theorem 6:

$$u_1 \geq u_2 \geq ... \geq u_\sigma > \Phi > \Phi^{'}.$$

Thus, in the presence of multiple oversized tasks, pre-partitioning has to be applied iteratively. For ease of discussion, each iteration of pre-partitioning is indicated with an additional prime, e.g. Φ, $\Phi^{'}$, $\Phi^{''}$, $\Phi^{'''}$, et cetera.

Since by Theorem 6, $\Phi > \Phi^{'}$, then $\Phi^{'} > \Phi^{''}$, $\Phi^{''} > \Phi^{'''}$, et cetera, which means that it is possible that tasks that were not oversized in η, are now oversized in $\eta^{'}$; a property referred to herein as cascading oversized tasks. An example of cascading oversized tasks is demonstrated in Figure A.1. Fortunately, even in the presence of cascading oversized tasks, Theorem 5 remains true as, when $\sigma = (\mathrm{M} - 1)$, either natively or by cascading, then $\mathrm{M}^{(\mathrm{M}-1)'} = 1$ and, thus, by Equation 4.1, $\Phi^{(\mathrm{M}-1)'} = \mathrm{U}^{(\mathrm{M}-1)'}$. Therefore, the remaining tasks can be scheduled on the remaining core. Further, in the case that there are $(\mathrm{M} - 1)$ oversized tasks in the system, pre-partitioning will produce a partition that is identical to WFD.

A result of Theorem 6 is that pre-partitioning can only approach but never attain a load-balance as, even in the best case, where there is only one oversized task, the load-balanced portion of the partition will always be at least slightly less than the first oversized task. However, since each oversized task is assigned to its own core, the oversized tasks will not migrate. Although the remaining non-oversized tasks scheduled by EDF-hv may migrate (assuming there are less than (M-1) oversized tasks in the system), the pre-partitioned tasks will not and thus, pre-partitioning should not increase the fluctuations in utilization due to migration above that which would be expected of EDF-hv without oversized tasks. Finally, pre-partitioning does not add any additional requirements to EDF-hv, and thus can be applied as a general solution to handling oversized tasks.

Oversized Migration

Migrating tasks in EDF-hv are assigned non-zero shares on multiple cores; thus, no single core bears the entirety of the migrating task's utilization. As such, in oversized migration, oversized tasks are allowed to be migrating tasks so no single core is assigned all of an oversized task and, thus, each core can have a utilization equal to the target load-balance. As such, oversized migration attains a load-balance.

In oversized migration, the oversized tasks in η are reordered with all the oversized task after the non-oversized task such that:

$$\eta = \{\tau_{\sigma+1}, \tau_{\sigma+2}, ..., \tau_{\mathrm{N}}, \tau_1, \tau_2, ..., \tau_\sigma\} \tag{A.11}$$

This way, the non-oversized tasks will be assigned to the cores with EDF-hv before the oversized tasks are. Unfortunately, one complication of oversized migration is that reordering the tasks violates Equation 3.3, which then invalidates the Theorem 3. However, since Theorem 3 considers the absolute worst-case, and Equation 5.2 provides ways to validate that a specific system will still be able to guarantee deadlines by Theorem 2, violating the Theorem 3 may not be an issue since the system can still be verified whether or not it can guarantee its deadlines.

An additional complication of oversized migration is that EDF-hv requires that each core has at least one fixed task. If each core is not assigned at least one fixed task, then it is guaranteed that energy will be wasted. Thus, as the purpose of EDF-hv is to improve energy efficiency, there must be enough non-oversized tasks in the system for each core to be assigned at least one non-oversized tasks before the oversized tasks are scheduled as migrating tasks. Thus, oversized migration adds the following requirement on the number of tasks in η:

$$\mathrm{N} \geq \sigma + \mathrm{M}. \tag{A.12}$$

Note that if there is only one oversized task in the system, then the restriction in Equation A.12 is identical to the system model.

One drawback of oversized migration is now the largest task in the system are migrating tasks, which means that the fluctuations in utilization due to migrating tasks may be significantly more than would be expected in EDF-hv without oversized migration. Thus, oversized migration may frustrate the DVFS algorithm such that more energy is consumed as a result of migrating tasks than is saved by the load-balance.

In summary, oversized migration can attain a load-balance; unfortunately, oversized migration threatens to increase fluctuations in utilization due to large migrating tasks, which may result in decreased energy efficiency. Further, oversized migration adds requirements to the number of tasks in the system as well as decreases the likelihood that EDF-hv can guarantee deadlines.

Multi-Partitioning

Multi-partitioning is similar to pre-partitioning except that instead of removing one oversized task at a time from η and assigning it to a core, the task set is partitioned into task subsets, η^{A}, η^{B}, η^{C}, et cetera, as well as the cores in P are partitioned into subsets, P^{A}, P^{B}, P^{C}, et cetera, and then each task subset is assigned to a subset of the cores. After the task subsets have been assigned to a core subset, the task subsets are scheduled using EDF-hv to the core subset it was assigned. There are many ways in which a multi-partition can be accomplished, however, relevant to EDF-hv, multi-partitioning can be broken into two major categories: vertical and horizontal. In vertical multi-partitioning (VMP), each task subset is assigned to the full set of cores, P, and then run in parallel. In horizontal multi-partitioning (HMP), each task subset is assigned to a unique core subset (e.g. the cores in P^{A} cannot be in any other core subset). In both VMP and HMP task subsets are scheduled with EDF-hv. Although it is possible to have a hybrid of VMP and HMP, this work focuses on VMP and HMP separately (the reasoning behind this decision will be more apparent in the next subsection). Further, while it is possible to have more than two task subsets, since it is trivial to apply the principals of a two-way partition to a multi-partition with more than two task subsets, this work will focus on multi-partitioning under the assumption of two-way partition (i.e. there are only two task subsets in each multi-partition). The remainder of this subsection will first address VMP and then HMP will be considered.

Vertical Multi-Partitioning (VMP)

In VMP, there needs to be twice the number of tasks as normally required since the tasks will be partition into two subsets, η^{A} and η^{B}, but the cores will not be partitioned. Thus:

$$\mathrm{N} \geq 2(\mathrm{M}+1) \tag{A.13}$$

In theory, if each task subset can attain a load-balance, then running two load-balanced partitions in parallel on the set of cores will be load-balanced. Thus, the ability of VMP to attain a load-balance is dependent on each task subset to eliminate oversized tasks. Unfortunately, in implementation, this is not possible.

Theorem 7. *VMP is unable to achieve a load-balance.*

Proof. From Equation A.13, VMP requires a system with $\mathrm{N} \geq 2(\mathrm{M}+1)$ where $\sigma > 0$ tasks are oversized. From Equation A.3:

$$u_1 \geq u_2 \geq ... \geq u_\sigma > \Phi = \frac{\mathrm{U}}{\mathrm{M}}$$

Since η^{A} and η^{B} are unique subsets of η, each has fewer tasks than η, which yields:

$$\mathrm{U} > \mathrm{U}^{\mathrm{A}} \rightarrow \frac{\mathrm{U}}{\mathrm{M}} > \frac{\mathrm{U}^{\mathrm{A}}}{\mathrm{M}} \rightarrow \Phi > \Phi^{\mathrm{A}}$$

$$\mathrm{U} > \mathrm{U}^{\mathrm{B}} \rightarrow \frac{\mathrm{U}}{\mathrm{M}} > \frac{\mathrm{U}^{\mathrm{B}}}{\mathrm{M}} \rightarrow \Phi > \Phi^{\mathrm{B}}$$

Thus:

$$u_1 \geq u_2 \geq ... \geq u_\sigma > \Phi > \Phi^{\mathrm{A}}$$

$$u_1 \geq u_2 \geq ... \geq u_\sigma > \Phi > \Phi^{\mathrm{B}}$$

As such, if a task was oversized in η, it is still oversized in η^{A} or η^{B} and therefore, VMP does not eliminate oversized tasks and, thus, cannot achieve a load-balance. □

In addition to being unable to achieve a load-balance, since the two subsets are expected to run in parallel on the same core set, the maximum number of allowed migrating task per core is doubled. This means that the potential for fluctuations in utilization due to migrating is increased. As such, since VMP cannot eliminate oversized tasks nor attain a load-balance as well as it is likely to increase fluctuations, VMP has little merit as a method to handle oversized tasks.

Horizontal Multi-Partitioning (HMP)

In HMP, since each task subset, η^{A} and η^{B}, will be assigned to a unique subset of the cores, there must be at least one task for each core as well as an additional task for each subset. Thus:

$$\mathrm{N} \geq \mathrm{M} + 2 \tag{A.14}$$

In addition to increasing the minimum number of tasks, since the set of cores will be partitioned into subsets, and EDF-hv is only applicable to multi-core systems, the number of cores must be at least four.

Similar to VMP, for HMP to attain a load-balance, each task subset needs to be able to eliminate oversized tasks. However, in contrast to VMP, for HMP to attain a load balance, each task subset would need to attain the same target utilization as the other task subsets such that:

$$\mathrm{U}^{\mathrm{A}} = \mathrm{U}^{\mathrm{B}} \tag{A.15}$$

Although it is possible for HMP to eliminate oversized tasks, HMP cannot satisfy Equation A.15 and thus cannot attain a load-balance, as is shown in the following two theorems.

Theorem 8. *HMP can eliminate oversized tasks.*

Proof. As derived in the proof for Theorem 7, $\mathrm{U} > \mathrm{U}^{\mathrm{A}}$ and $\mathrm{U} > \mathrm{U}^{\mathrm{B}}$. Similarly, since the cores subsets, P^{A} and P^{B}, are unique subsets of P, $\mathrm{M} > \mathrm{M}^{\mathrm{A}}$ and $\mathrm{M} > \mathrm{M}^{\mathrm{B}}$. However,

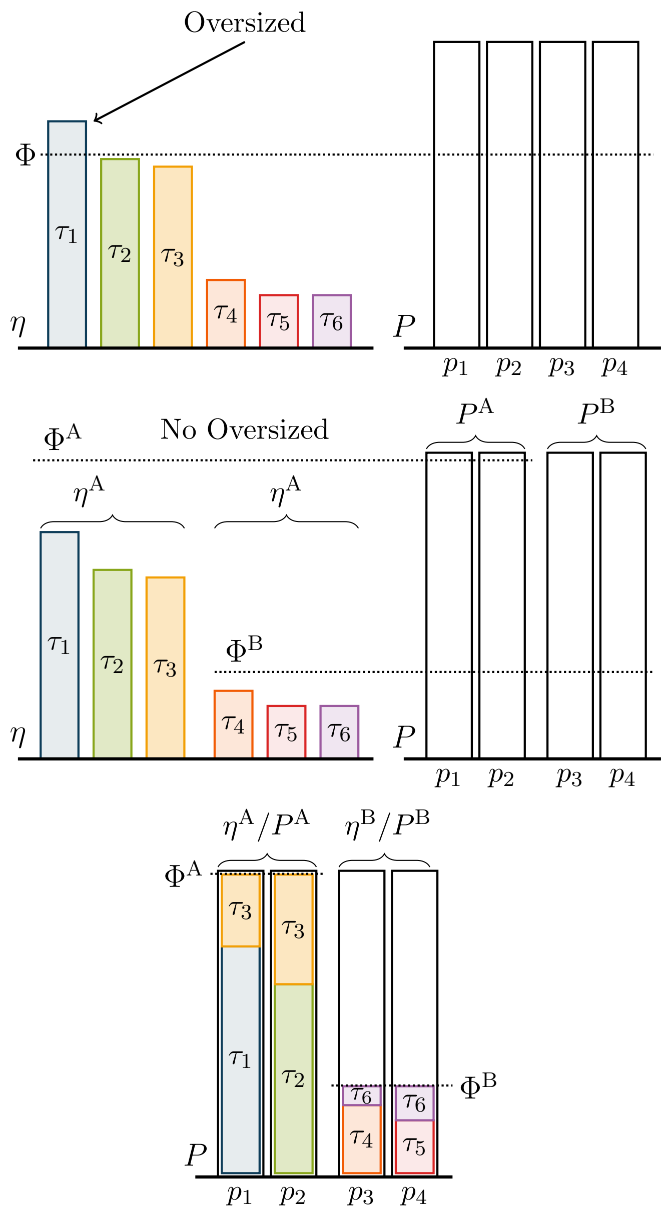


Fig. A.2: An example of horizontal multi-partitioning (HMP). The original task set, η (top; where each task's utilization is shown), has one oversized task, τ_1. As such, η is divided into two subsets, η^{A} and η^{B}, as well as P is also divided into two subsets, P^{A} and P^{B} (middle). Although the resulting subsets no longer have any oversized task, the system is clearly not load-balanced (bottom).

the relationship between $\mathrm{U^A}$ and $\mathrm{U^B}$ as well as the relationship between $\mathrm{M^A}$ and $\mathrm{M^B}$ is ambiguous as one quantity can be greater than, equal to, or less than the other. As such, since Φ^A and Φ^B is the ratio of $\frac{\mathrm{U^A}}{\mathrm{M^A}}$ and $\frac{\mathrm{U^B}}{\mathrm{M^B}}$, respectively, in the event that $\mathrm{U^A} > \mathrm{U^B}$ and $\mathrm{M^A} \leq \mathrm{M^B}$ then: $\Phi^\mathrm{A} > \Phi$ (it is trivial to show that the same can be true for Φ^B). Although there are other combinations to show that Φ^A can be greater than Φ, this example is sufficient to show it is possible. As such, since Φ^A can be greater than Φ, then it is possible for Φ^A to be greater than the oversized tasks that existed in η, and, therefore, HMP can eliminate oversized tasks. An example of a system in which HMP eliminates oversized tasks is shown in Figure A.2. □

Theorem 9. *HMP cannot attain a load-balance.*

Proof. Building on the proof for Theorem 8, it is trivial to show that when $\Phi^\mathrm{A} > \Phi$, $\Phi^\mathrm{B} < \Phi$ (the opposite is also true, but for this proof it is assumed that $\Phi^\mathrm{A} > \Phi$). As such, since Φ^A must be greater than Φ in order to eliminate oversized tasks:

$$\Phi^\mathrm{A} > \Phi > \Phi^\mathrm{B}$$

Therefore, to eliminate oversized tasks, Equation A.15 cannot be true, and thus HMP cannot attain a load-balance. □

In summary, HMP can eliminate oversized tasks but cannot attain a load-balanced solution. Since HPM does not increase the number of migrating tasks, HMP should not add additional fluctuations in utilization due to the migrating tasks. Additionally, HMP does add requirements to the number of tasks as well as the number of cores in the system.

A.3 Best Method for Handling Oversized Tasks

We have developed three methods to handle oversized tasks. Table A.1 shows a comparison of the three methods, with multi-partitioning broken into its vertical and horizontal parts, based on the three metrics described in Section A.2. As shown, pre-partitioning is the only method that could be applied to EDF-hv without any additional requirements

Table A.1: Comparison of methods to handle oversized task using the metrics described in Section A.2.

	Load-Balance	**Increase Fluctuations**	**Increase Requirements**
Pre-Partitioning	No	No	No
Oversized Migration	Yes	Yes	$\mathrm{N} \geq \sigma + \mathrm{M}$ Decrease Feasibility
Vertical Multi-Partitioning	No	Yes	$\mathrm{N} \geq 2(\mathrm{M}+1)$ Decrease Feasibility Violates Theorem 3
Horizontal Multi-Partitioning	No	No	$\mathrm{M} \geq 4$ $\mathrm{N} \geq \mathrm{M}+2$

and therefore is the only general solution to handling oversized tasks. Oversized migration was the only solution that could attain a load-balance, however, it is likely to significantly increase the fluctuations in utilization as a result of migrating tasks since the largest tasks in the system will be migrating tasks. Further, allowing the largest tasks to be migrating tasks also decreases the likelihood of being able to meet Theorem 2. As such, while oversized migration may be able to provide a load-balanced solution, it is not likely to be applicable in many systems. VMP was unable to meet any of the metrics and is thus unsuitable to handle oversized tasks. HMP was unable to attain a load-balance and increased the requirements on the number of tasks and number of cores. However, something that was not captured in Table A.1 is that HMP is able to eliminate oversize tasks. Thus, while HMP fails to attain a full load-balance, oversized tasks can be eliminated such that each partition can be load-balanced. As such, HMP may have use in systems with multiple VFI, such that the cores on each VFI can be load-balanced.