Energy-Aware Partitioned Fixed-Priority Scheduling for Chip Multi-Processors

Arvind Kandhalu, Junsung Kim, Karthik Lakshmanan, Ragunathan (Raj) Rajkumar
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, USA-15213
{arvindkr, junsungk, klakshma, raj}@ece.cmu.edu

Abstract

Energy management is becoming an increasingly important problem in application domains ranging from embedded devices to data centers. In many such systems, multi-core processors are projected as a promising technology to achieve improved performance with a lower power envelope. Managing the application power consumption under timing constraints poses significant challenges in these emerging platforms. In this paper, we study the energy-efficient scheduling of periodic real-time tasks with implicit deadlines on chip multi-core processors (CMPs). We specifically consider processors with a single voltage and clock frequency domain, such as the state-of-the-art embedded multi-core NVIDIA Tegra 2 processor and enterprise-class processors such as Intel's Itanium 2, i5, i7 and IBM's Power 6 and Power 7 series. The major contributions of this work are (i) we prove that Worst-Fit-Decreasing (WFD) task partitioning when Rate-Monotonic Scheduling (RMS) is used has an approximation ratio of 1.71 for the problem of minimizing the schedulable operating frequency with partitioned fixed-priority scheduling, (ii) we illustrate the major shortcoming of WFD with RMS resulting from not considering task periods during allocation, and (iii) we propose a Single-clock domain multi-processor Frequency Assignment Algorithm (SFAA) that determines a globally energy-efficient frequency while including task period relationships. Our evaluation results show that SFAA provides significant energy gains when compared to WFD. In fact SFAA is shown to save up to 55% *more power compared to WFD for an octa-core processor.*

1. Introduction

Chip Multi-Processors (CMPs) that offer multiple processing cores on a single chip have quickly become prevalent. Major chip makers now have CMPs with 2, 4 or 8 cores [9, 16, 15, 22, 31]. Further, extensive research is underway to build chips with potentially hundreds of cores or many-core systems [3, 30]. Energy management has been a very active research area in the recent past and one of the main motivating factors leading to CMP architectures was the unsustainable ever-increasing frequency and power density trends of traditional single-core architectures. As a result, CMPs come equipped with Dynamic Voltage and Frequency Scaling (DVFS) with multiple operating point steps. The total processor energy consumption consists of two components: *dynamic*, and *static* power. The first relates to the power that is dissipated due to switching activity, while the second is due to leakage current. A common way of reducing dynamic power is to use Dynamic Voltage Frequency Scaling (DVFS) [40]. DVFS changes the processor supply voltage and the clock frequency simultaneously, thereby reducing the energy consumption. A method that has been suggested for reducing the static power consumption is to shutoff the processor cores when idle [18, 24, 19, 33]. For such a mechanism to be employed, dynamic shutdown of processor cores must be supported by the hardware. The energy-efficient scheduling problem in hard real-time systems with DVFS and/or dynamic shutdown-capable processors is to minimize energy consumption while ensuring that all the real-time tasks meet their deadlines.

Several papers in the recent past have proposed DVFS-based solutions for real-time embedded systems running on conventional multi-processor platforms where each processor is located on a separate chip [1, 12, 7, 39]. These studies identify two main dimensions of the problem as task-to-CPU allocation and run-time voltage scaling on individual CPUs [1, 39]. Yet, the emerging CMP platforms have a number of unique traits which make the problem different from conventional multi-processor platforms. While it is natural to have different voltage levels per CPU (per-CPU DVFS capability) in a multi-processor system, the tight coupling of cores on a single chip (CMP) implies that the per-core DVFS feature would come with severe additional circuit complexity, stability, and power delivery problems [3, 13, 30]. In fact, in the state of the art commercial CMPs the processing cores share a common voltage level. A recent study [13], based on detailed VLSI circuit simulations, suggests that the

1533-2306/11 $26.00 © 2011 IEEE
DOI 10.1109/RTCSA.2011.75

potential energy gains of per-core DVFS are likely to remain too modest for justifying the complicated designs. For the next-generation many-core systems, it is likely that only a small number of clusters/blocks each with several cores and independent voltage regulators will be feasible [3]. Independent and effective management of such clusters or in other words *voltage islands* would be the ultimate objective in these next-generation systems [3, 13, 32].

The focus of this paper is on energy-efficient partitioned fixed-priority scheduling of periodic real-time implicit-deadline tasks on multi-core processors with a common voltage and clock domain. Although partitioned multi-core processor real-time scheduling is known to be NP-hard in the strong sense [10, 20], simple and effective partitioning heuristics are known to have reasonable average-case performance in practice [25, 27]. In fact, the partitioned approach is arguably more common due to its simplicity and ease of implementation. Unfortunately, the problem of task-partitioning to minimize energy consumption is NP-hard [1, 12] and in particular the voltage island model poses a number of challenges that have only very recently started to attract attention [35, 38]. Under the global voltage/frequency constraint, the core with the maximum load could be the main deciding factor in the overall CMP energy consumption [35, 38]. This suggests the importance of load balancing. It has also been proven that the total energy consumption of the multi-core processor is minimized under EDF scheduling in the case that the workload is perfectly balanced across the processors [12]. The Worst-Fit-Decreasing (WFD) heuristic can achieve load balancing, and thereby save energy. In this paper, we first prove a 1.71 approximation ratio for WFD with Rate Monotonic Scheduling (RMS) for the problem of minimizing schedulable operating frequency in paritioned fixed-priority scheduling. Then we show that load balancing does not always lead to lowering operating frequency (and hence the energy consumption) for CMP systems using RMS. Subsequently, we propose a partitioned Single-clock multiprocessor Frequency Assignment Algorithm (SFAA) that is targeted at reducing the global operating frequency. Through experimental evaluation we show that SFAA performs significantly better than WFD in terms of power consumption. We use data obtained from measurements of an actual processor (NVIDIA's Tegra 2 processor) to determine the power model used in our evaluation.

The rest of this paper is organized as follows. We summarize prior research related to this work in Section II. In Section III, we describe the processor and power model that we use in this paper. In Section IV, we prove the approximation ratio for WFD under RMS and go on to show that WFD may not always provide the globally energy-efficient operating frequency. We then propose our SFAA algorithm Section V. In Section VI we present our evaluation results and finally in Section VII we provide our concluding remarks.

2. Related Work

Energy-aware scheduling for real-time systems on DVFS platforms has been extensively studied [6]. We now briefly describe prior research related to our work.

A theoretical exploration of DVFS systems for real-time jobs was first given by Yao, Demers and Shenker [40] by considering a set of aperiodic jobs on an ideal processor, in which each job is characterized by its release time, deadline, and execution CPU cycles. All jobs have the same power consumption function. An off-line scheduling algorithm was proposed to minimize the energy consumption of task executions. For periodic real-time tasks on an ideal processor, Aydin et. al. [12] showed that an optimal schedule would execute all the tasks at a common speed to fully utilize the processor when all the tasks have the same convex (and increasing) power consumption function. Saewong and Rajkumar [34] considered rate-monotonic task scheduling and determined the minimum constant speed for all tasks to satisfy the schedulability condition of Rate-Monotonic Scheduling (RMS). This scheme is called *Sys-Clock*. They also discussed the energy effects of quantization due to available processor operating frequencies.

Research studies on energy management for multi-processor real-time systems are typically based on independent DVFS capabilities of individual processors. The multi-processor energy-aware task allocation for RMS-based systems is known to be NP-Hard [1] and heuristic algorithms were proposed in [1, 12] to exploit the well-known bin-packing algorithms for task partitioning. In [1], the authors conclude that Time-Demand Analysis-based admission control combined with the Sys-Clock speed assignment scheme has the best overall performance in both off-line and online settings, at the cost of pseduo-polynomial time complexity. Moreover, in off-line settings, the Worst-Fit heuristic has the best overall performance among partitioning heuristics. In this paper, we establish a 1.71 approximation ratio for Worst-Fit Decreasing with Rate-Monotonic Scheduling under Sys-Clock. This result establishes a theoretical bound on the performance of the Worst-Fit Decreasing heuristic for the energy-efficient partitioned fixed-priority scheduling problem.

Energy management of real-time tasks on CMP platforms under the global voltage/frequency constraint has been of interest to the research community recently. In [38], the authors showed that problem is NP-hard considering frame-based systems and provided a 2.371-approximation scheme. In [2], the authors proposed a power-aware scheduler for multicore systems executing

soft real-time workload. In [35], the authors considered the problem of energy-efficient scheduling for periodic hard real-time tasks on CMP systems. The authors proposed a scheme to re-partition tasks at run-time by resorting to task migrations, so as to create more balanced schedules that adapt to dynamic workload variability. Further, they also proposed a dynamic core scaling algorithm adjusting at run-time the number of active cores under the assumption that transitions between off and active states can be done instantaneously and with no additional overheads. However, in practice such transitions are rarely attractive or possible for periodic real-time applications. Finally, the overhead of frequent task migrations may be a concern in practice. In [8], the authors provide a framework for minimizing energy on CMP systems under EDF scheduling. The framework consists of a static and a dynamic phase. In the static phase a subset of cores are selected such that the workload can be completed while meeting their deadlines and a task-to-core allocation is performed as well. The dynamic phase involves managing the selected cores at run-time by exploiting early-completion of task executions. While keeping certain cores of the processor switched off might be useful for reducing static power consumption, not all multi-core processors provide such an option [37]. Exploiting early-completion of tasks could be useful in processors with cores having independent-DVFS capability. However, in case of common voltage/clock domain processors with statically allocated tasks such exploitation may not be effective as it requires that tasks assigned to different cores complete early together. In [21], a polynomial-time complexity algorithm based on binary search to minimize the energy consumption of a multi-core system is proposed. There has also been work that use DVFS techniques to reduce the operating temperature of the processor [23, 5, 4].

3. Task and Power Model

3.1. Task Model

We consider a set of independent periodic hard real-time tasks $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, where n is the number of tasks. Each task τ_i is given by $\{C_i, T_i, D_i\}$, where C_i is the worst-case execution time at a maximum processor frequency of f_{max}, T_i is the period, and D_i is the relative deadline from arrival time. We assume in this paper that $T_i = D_i$ resulting in implicit deadlines. The clock frequency f we refer throughout the paper is a relative frequency normalized to f_{max}. Hence, f will be less than or equal to 1, since $f \leq f_{max}$. The worst-case execution time of task τ_i when running on a processor at frequency f is given by $c_i(f) = C_i/f$. We consider fixed-priority preemptive scheduling; the preemption overheads can be incorporated in C_i if necessary. The total utilization of the task set at f_{max} is given by $U_{tot} = \sum_{i=1}^{n} C_i/T_i$. Note that a necessary but not a sufficient condition for feasibility on a system of m identical multi-core processors is to have a task set whose total utilization does not exceed the total computing capacity. Hence, we assume that the condition $U_{tot} \leq m$ holds throughout the paper. A task is schedulable iff all its instances complete no later than their deadlines. A task set is schedulable iff all its tasks are schedulable. The utilization bound denoted by U_{bound} is the utilization such that all task sets with a utilization that is lower than or equal to U_{bound} are schedulable. We adopt a partitioned approach to multi-core processor scheduling. Tasks are statically assigned to processors. On each processor, the Rate-Monotonic Scheduling (RMS) policy is adopted i.e. tasks are assigned fixed priorities that are inversely proportional to their periods.

3.2. Processor and Power Model

We consider a homogeneous multi-core processor platform M with m processors $\{p_1, ..., p_m\}$. The processor operating frequency can be anywhere between f_{min} and f_{max}. We also assume that all the processors are ''fed" the same clock signal which reflects many current processors such as NVIDIA's Tegra 2 processor, and will likely remain relevant in modern many-core processors where each cluster of cores will still be present in the same voltage/frequency domain since it is often prohibitively expensive to assign each core its own voltage island. We call such processors as Single-Clock domain Multi Processors (SCMP). In this paper, we restrict our attention to the static frequency assignment problem, where the entire processor is statically assigned an operating frequency so as to ensure the schedulability of all tasks in the system. Runtime frequency scaling is not considered as it is requires coordinated scaling across all cores within a single domain and this could incur high overhead.

As discussed in [34], the dynamic power consumption function can be modelled as a function proportional to f^{α}, where α is a constant. The leakage power of each of the processor is a non-negative constant, denoted by β_2 here. Then, the power consumption function is $(\beta_1 f^{\alpha} + \beta_2)$. In order to obtain the values of the constants in the power consumption function, we measured the power consumption for the NVIDIA Tegra 2 processor [37] used in the Motorola Xoom platform [17]. The Tegra 2 processor has two cores and seven effective operating points. The operating points are shown in Table 1. Before beginning a measurement we first charge the device to its maximum voltage level. At each operating-point we run CPU-bound busy-loops for 15 minutes. We obtain the battery charge level before and after the experiment. In this manner the amount of battery consumption at a particular operating point can be obtained. Ten such

Operating Point	Frequency (MHz)	Voltage (mV)
1	312	750
2	456	825
3	608	900
4	760	975
5	816	1000
6	912	1050
7	1000	1100

Table 1: Tegra 2 processor operating points

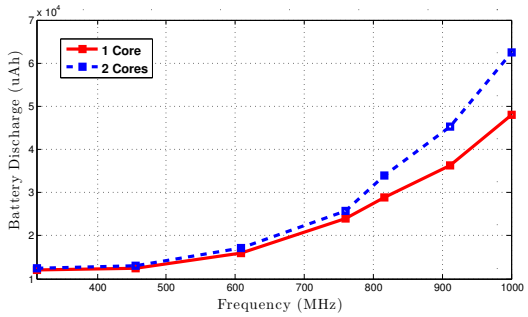


Figure 1: Battery consumption at each of the operating points of the Tegra 2 processor

measurements are taken for each operating point and averaged. It is to be noted that we set the Xoom platform to *Airplane mode*, close all running applications and services, and switch off the display when conducting the experiment. The measurement results are shown in Figure 1.

We have estimated the constants from the actual data to be $\alpha = 3.94565$, $\beta_1 = 3.89462 \times 10^{-26}$ and $\beta_2 = 0.8453 \times 10^{-9}$. Figure 2 shows the actual battery consumption data and estimated data for a single core on the Tegra2 processor.

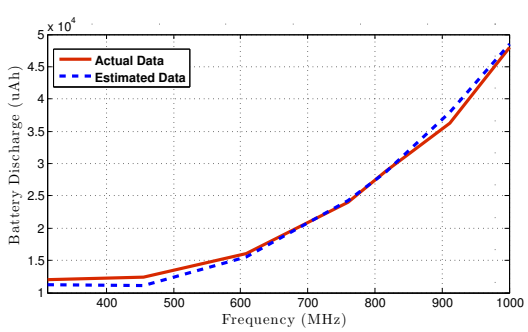


Figure 2: The actual battery discharge and estimated discharge of the Tegra 2 processor.

4. Global Voltage/Frequency constrained CMP Load Balancing under RMS

In this paper, we are concerned with the static frequency assignment problem, where an energy-minimizing operating frequency is assigned to the processor using the *uniform slowdown technique* such that all tasks meet their deadlines. The uniform slowdown technique assigns an unique operating frequency to all the tasks on a processor such that the new effective utilization does not exceed the utilization bound obtained by the admission control algorithm. As such, it is easy to see that this technique can be used in conjunction with any utilization bound-based admission control algorithm. For example, suppose that, two tasks are assigned to processor p_i with total utilization $U_i = 0.414$ in a feasible partition. If the Liu-Layland [26] schedulability bound $n(2^{1/n} - 1)$ is used with $n = 2$, then the processor can be slowed down till the utilization of the processor reaches 0.828. Hence the *slowdown factor (SF)* is $= 2$, i.e. the energy-minimizing operating frequency f_{SF} should be no lower than $0.5 f_{max}$. Sys-Clock [34], a time-demand analysis-based speed assignment technique, can be used to improve energy savings. For SCMP systems with a single clock domain, the operating frequency after slow-down should be chosen as $f_{SF}(M) = max(f_{SF1}, ..., f_{SFm})$, where m is the total number of processors. Dynamic DVFS techniques [28, 29, 41] and those requiring operating frequency change between task context-switches are more suitable for Multi-Clock Multi-Processor (MCMP) systems, i.e. systems where each processor or groups of processors have independent clocks. Also, uniform slow-down technique and Sys-Clock are appropriate for SCMP systems and high DVFS-overhead processors as it does not require any change in the clock-frequency during run-time.

From Figure 1 we see that lowering the operating frequency will result in lowering the energy consumption. The problem of minimizing energy consumption using task partitioning in CMPs can be formally stated as follows:

Given a set Γ of periodic hard real-time tasks and a set M of identical processor-cores with a common voltage/frequency domain, find a task partition and compute the global operating frequency for all the processor-cores such that:

1. the workload can be scheduled by RMS in a feasible manner, and

2. the total energy consumption on all processors $p_1, ..., p_m$ is minimum among all feasible partitions.

The above problem is NP-hard, since it is a more general form of the feasibility problem in partitioned

multi-core processor scheduling which is known to be NP-hard in the strong sense. In fact, even when the task set is known to be *trivially schedulable*, the problem remains NP-hard i.e. although any partitioning would yield a feasible schedule, but computing the one with minimum energy consumption is still intractable [25].

Definition 1. *Given a normalized frequency of $f_{max} = 1$, a processor is said to have a* slowdown factor *of k if it is run at a frequency of $1/k$ and all the tasks on that processor continue to meet their deadlines.*

Hence a scheduling algorithm that achieves a larger slowdown factor on a given task-set results in greater energy savings.

Proposition 1. *For the static frequency assignment problem with a single clock frequency, an ideal energy-aware task partitioning algorithm is one that partitions a given task-set Γ with utilization U (at $f_{max} = 1$) onto m processors such that all the processors are fully utilized at a slowdown factor of (m/U), and all the tasks continue to meet their deadlines.*

Proof: Suppose there is an algorithm A that achieves a slowdown factor $\frac{1}{f_A} > (m/U)$ for taskset τ. In this case, over any interval of time t, the m processors executing at frequency $f_A < (U/m)$ have less than (Ut) cycles. However, the taskset Γ requires (Ut) cycles, and therefore tasks in Γ should miss their deadlines. Hence, by contradiction, no such algorithm A can exist. □

Remark 1. *Based on Proposition 1, a task partitioning algorithm that achieves a minimum per-processor utilization bound of UB on all processors can achieve a slowdown factor of $UB(m/U)$ for a taskset with utilization U (at $f_{max} = 1$) on m processors, since it can utilize all the processors up to UB without missing deadlines at frequency $(U/m)(1/UB)$.*

The convex relationship between the CPU speed and power consumption suggests load-balancing techniques for energy-aware partitioning across DVFS-enabled multi-core processors. It has been shown that if a perfectly balanced partition exists and if Earliest Deadline First (EDF), is used, then it is provably optimal [12]. Among task allocation heuristics, the Worst-Fit Decreasing (WFD) algorithm is known to typically yield well-balanced partitions where the maximum load on any core is small. In fact, the combination of WFD with Sys-Clock has been found to perform well in terms of energy savings [12]. Assuming that the tasks are already sorted in non-increasing order of their utilization, WFD allocates tasks one by one to the core with the least load at a time. For this specific problem, WFD is equivalent to the well-known *List Scheduling Algorithm (LST)* where independent tasks each with a given size in the range [0,1] are partitioned to m CPUs each with unit capacity, with the objective of reducing the maximum size allocated to any core. In our case, the capacity of a bin corresponds to the utilization bound of a task-set assigned to a processor and the size of tasks corresponds to their individual utilization. The result in [11] implies that the maximum load among all cores generated by LST (*and equivalently WFD in case of EDF or RMS with only harmonic tasks*) is no more than $\frac{4}{3} - \frac{1}{3m}$ times that of the optimal. We see that as $m \to \infty$, the ratio becomes $\frac{4}{3}$, which implies that WFD has an approximation ratio of $\frac{4}{3}$ when used with EDF.

The approximation ratio of an algorithm is the ratio between the result obtained by the algorithm and that obtained by the optimal algorithm. An algorithm with approximation ratio k is called a k-approximation algorithm. In our case we see that if the optimal algorithm partitions the task set such that resulting operating frequency is f then WFD will result in an operating frequency of $\frac{4}{3}f$ when used with EDF.

Proposition 2. *The Worst-Fit Decreasing (WFD) heuristic for partitioning independent tasks onto a multi-processor under Rate Monotonic Scheduling (RMS) has an approximation ratio of at most $\frac{4}{3\,ln2}$.*

Proof: This upper bound can be deduced as follows. The least upper utilization bound for a set of n schedulable tasks is given by $U_{bound}(n) = n(2^{1/n} - 1)$. As $n \to \infty$ we know that U_{bound} is $ln2$. Consider that the scenario that the processor frequency determined using optimal allocation is f on m processors. Under the worst-case scenario, the WFD heuristic may result in a packing with per-processor utilization at $\frac{4}{3} - \frac{1}{3m}$ times the optimal with each processor core being subject to the utilization bound of $n(2^{1/n} - 1)$, while the optimal could perfectly balance the load on each processor. We see that the maximum clock frequency in this scenario will be:

$$\{\frac{1}{n(2^{(1/n)}-1)}\} \times \{\frac{4}{3} - \frac{1}{3m}\}.$$

We see that this function is maximized as $n \to \infty$ and $m \to \infty$, where the above expression becomes $\frac{1}{ln2} \times \frac{4}{3} = 1.92$. □

We will now tighten this bound to 1.71 in Theorem 1.

Theorem 1. *The approximation ratio of the Worst-Fit Decreasing (WFD) Partitioning heuristic for Rate-Monotonic Scheduling (RMS) of independent tasks onto a multi-processor platform is at most 1.71.*

Proof: Assume a set of m processors and n independent tasks that need to be partitioned. Without loss of generality, assume that the operating frequency resulting from WFD with RMS is 1.0 (the maximum operating frequency). We need to establish a lower bound f on the operating frequency required by an optimal algorithm to guarantee deadlines, in order to provide an upper bound on the approximation ratio $\frac{1}{f}$ resulting from the

combination of WFD with RMS.

Let us say that the processor frequency of 1.0 is first reached by WFD with RMS when a task τ_i of utilization $u_i = u$ is assigned to processor p_j. Hence, we can restrict our attention to a subset of tasks with utilization greater than or equal to u, since adding additional tasks with utilization lesser than u cannot decrease the frequency required by the optimal algorithm (or increase the approximation ratio).

We now consider three exhaustive cases for p_j:

Case 1: Processor p_j is empty

In this scenario, task τ_i would be assigned to an empty processor under WFD and for the processor frequency to reach 1.0 task τ_i itself should have an utilization of 1.0. The optimal algorithm will also have to choose a frequency of 1.0 on the processor that τ_i is assigned. Therefore, the approximation ratio will be 1.0, which is less than 1.71.

Case 2: Processor p_j has only one task

Consider that there is exactly one task with utilization η already in the processor p_j to which task τ_i is allocated under WFD with RMS.

$(u + \eta) \geq U_{bound}(2) = 82.8\%$

The two task utilization bound of $2(\sqrt{2}-1)$ should be exceeded so that an operating frequency of at least 1.0 will be required to meet the deadlines of all the tasks (if $(u + \eta) < U_{bound}$ then a lower operating frequency than 1.0 will be sufficient).

Due to the worst-fit allocation, all the remaining processors (other than p_j) have an utilization greater than or equal to η. The set of tasks Γ can be considered to be union of two sets $\Gamma_1 = \{\tau_k | u_k \geq \eta\}$ and $\Gamma_2 = \{\tau_l | \eta > u_l \geq u\}$. We see that task $\tau_i \in \Gamma_2$.

As τ_i is being added to the processor p_j of utilization η with a single task, each of the tasks in Γ_1 should be by themselves on their processors. Let the number of tasks in Γ_1 be α. This implies that α processors would be taken by tasks in Γ_1 and the remaining $(m - \alpha)$ processors would be occupied by tasks in Γ_2, where α denotes the total number of tasks in Γ_1. Also the remaining tasks in Γ_2 will be assigned to the remaining $(m-\alpha)$ processors such that at least two tasks are allocated together. This is due to the fact that if one of these tasks were on a separate processor its utilization would have been lesser than η and would have been chosen as the candidate processor for allocating τ_i under WFD.

We have established that α processors would have a task each from Γ_1 and the remaining $(m-\alpha)$ processors would have at least two tasks each from Γ_2. In that case the optimal algorithm would have to either pack τ_i with either one of the processors with a task each of utilization at least η or with at least two tasks of utilization greater than or equal to u. We see that if the optimal algorithm picks the processor having one task with utilization η then it does not do better than WFD by a factor of $\frac{1}{0.828}$, which is less than 1.71. Hence, we are only concerned with the set of processors having at least two tasks already packed onto them.

Case 3: Processor p_j has at least two tasks

Consider the scenario that there are at least two tasks in processor p_j where τ_i is considered for allocation. Let the utilization of processor p_j be η before allocating task τ_i. This implies that each of the other $m - 1$ processor cores have at least a utilization of η.

After assigning task τ_i to p_j, the operating frequency of p_j goes to 1.0 for that processor and thereby for all the processors is 1.0 as they are all within the same clock-frequency domain.

This implies,

$$U_j \geq U_{bound}^{p_j} \implies (\eta + u) \geq U_{bound}^{p_j} \tag{1}$$

We can effectively ignore all processors (say α processors) with a single task, since these tasks have a utilization greater than η, and adding any other task with such tasks in the optimal allocation will result in a utilization of $(\eta + u)$ with an operating frequency of at least $U_{bound}^{p_j} \geq ln2 \approx 0.69$ and approximation ratio lesser than 1.44.

We know from [36] that the utilization bound for three tasks, where one of the task has utilization u is given by:

$$2\{\{\frac{2}{1+u}\}^{1/2} - 1\} + u \tag{2}$$

From equations 1 and 2, we have,

$$\eta + u \geq (2\{\{\frac{2}{1+u}\}^{1/2} - 1\} + u)$$
$$\implies \eta \geq (2\{\{\frac{2}{1+u}\}^{1/2} - 1)\}$$

When there are at least two tasks per processor core before the allocation of τ_i, we have at least $2(m-\alpha)+1$ tasks to be allocated on $(m-\alpha)$ processors. By Pigeon-Hole Principle, even the optimal allocation will still have to pack three tasks on to at least one processor (say p_k). Then the operating frequency f of p_k will be at least $3u$ since each task has an utilization of at least u. This implies that the optimal processor will have to choose a frequency that is $max(\eta, 3u)$. This operating frequency is minimized when $\eta = 3u$. Hence we have,

$$3u = (2\{\{\frac{2}{1+u}\}^{1/2} - 1)\} \implies u \approx 0.195584 \tag{3}$$

Thus we have $f \geq 3u \implies f \geq 0.586752$. Hence the ratio of the operating frequency when using WFD to that of using optimal is $1/f \leq 1.71$ Also if more than three tasks are allocated to processor p_k, then the ratio between WFD and the optimal will only be smaller. For example if there are four tasks then we have,

$$4u = (3\{\{\frac{2}{1+u}\}^{1/3} - 1)\} \implies u \approx 0.15 \tag{4}$$

and the ratio will be 1.64. The ratio will become one when there are infinite number of tasks. The worst case happens when have three tasks.

□

5. Global Operating Frequency Minimization

For RMS, load-balancing does not always result in lowering energy consumption. This is illustrated by the following example.

5.1. Motivating Example

Consider the case where four tasks $\tau_1 = (4, 10, 10)$, $\tau_2 = (6, 14, 14)$, $\tau_3 = (4, 10, 10)$ and $\tau_4 = (6, 14, 14)$ need to be allocated on two processors p_1 and p_2 (where the execution times are assumed to be obtained at the maximum operating frequency of $f_{max} = 1$). Let us assume an ideal processor with infinite frequency granularity, where the processor can operate at any operating frequency from 0 to 1. We also assume that rate-monotonic scheduling is used on each processor.

Consider the static frequency assignment problem in this setup, where a single clock frequency f_{SF} needs to be assigned for all the processors, and no run-time frequency changes are allowed. Consider two different task allocations:

Case 1: Tasks τ_1 and τ_2 are assigned to p_1, and tasks τ_3 and τ_4 are assigned to p_2.
The critical scheduling instant, when jobs of all the tasks are released simultaneously on all the processors, is shown in Figure 3a. Observe that between the release of tasks τ_1 and τ_2 on p_1 and the completion of the second instance of τ_1, there are no idle cycles. Similarly on p_2, until the completion of the second instance of τ_3 there are no idle cycles. This implies that slowing down the processor will result in the tasks missing their deadlines. Hence although the utilizations of the processors are less than 1.0 and equal ($\approx$ 0.83), the operating frequency should be set as 1.0 i.e. $f_{SF} = f_{max} = 1.0$, since decreasing the frequency would increase the execution times of each task causing the tasks to miss their deadlines.

Case 2: Tasks τ_1 and τ_3 are assigned to p_1, while tasks τ_2 and τ_4 are assigned to p_2.
The critical scheduling instant for this case is shown in Figure 3b. Observe the idle cycles between the completion of the tasks and their next arrival. These idle cycles can be used for slowing down the processor. Hence, although the utilizations are not balanced, processor p_1 can be slowed down to $0.8f$ and processor p_2 can be slowed down to $0.86f$ respectively. In a system with a single clock domain, both the processors can operate at a clock frequency $f_{SF} = 0.86$ and all the tasks will continue to meet their deadlines.

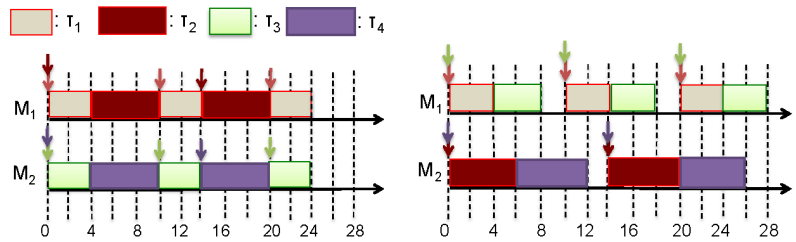
The above example illustrates that balancing the utilization may not always give the energy-minimizing task partition and that the periods of tasks have an important role to play as well. While WFD only uses the task utilization for partitioning, we see that a partitioning algorithm that considers task periods as well is required.

5.2. Static Frequency Assignment Algorithm (SFAA) for DVFS enabled SCMP platforms

Algorithm 1 $\beta = SFAA(\Gamma, m)$

Input: Γ **Output:** β, P
Sort tasks in Γ in decreasing order of size
Initialize Φ to zero
for i = 1 to n **do**
  for Each task τ_k in Γ **do**
    for j = 1 to m **do**
      $p_j \cup \tau_k$
      $\vartheta_{k,j}$ = $SysClock(p_j)$
      $\theta_{k,j}$ = $\vartheta_{k,j} - \Phi(p_j)$
      $p_j - \tau_k$
    end for
    $\Theta_k = max(\theta_{k,1}, ..., \theta_{k,m})$
  end for
  $\psi = q|\vartheta_{k,q} = min(\vartheta_{k,1}, ..., \vartheta_{k,m})$
  Assign τ_l to $P(p_\psi)|l = max_{\tau_l \in \Gamma}(\Theta_1, ..., \Theta_n)$
  Update $\Phi(p_\psi)$ with $SysClock(p_\psi)$
  Γ = $\Gamma - \tau_l$
end for
β = $max(\Phi)$
return β and P

We propose a frequency assignment algorithm for identical multi-core processors with a common clock input called SCMP Frequency Assignment Algorithm (SFAA) given in Algorithm 1. SFAA uses the Sys-Clock [34] algorithm to determine the lowest possible operating frequency for a given task-set. We briefly describe Sys-Clock here. In a system using a fixed-priority preemptive scheduling policy, the workload needed to complete a task's request composes of the task's own execution and the preemption by higher priority tasks. When there are multiple task periods, the preemption is not uniformly distributed over the task's critical zone. The workload hence varies in preempting processor cycles and depends on when the task completes. Since the workload changes at the end of each idle period, Sys-Clock determines the clock frequencies which allow a task to complete execution exactly at the end of each idle period between the earliest and latest possible completion time. The



(a) Case 1: The maximum of the normalized *sys-clock* frequencies of the two processors is equal to one i.e. f_{SF} = 1.0

(b) Case 2: The maximum of the normalized *sys-clock* frequencies of the two processors is less than one i.e. f_{SF} = 0.86

Figure 3: The effect of task period ratios of the allocated tasks.

energy-minimizing clock frequency of a task is chosen to be the minimum clock frequency among these frequencies. Then the Sys-Clock frequency is the energy-minimizing clock frequency that allows every task in the given task-set to complete before its deadline. This Sys-Clock frequency is optimal for fixed-priority preemptive scheduling policies that use a single operating frequency.

SFAA first adds each of the tasks in the task set Γ to each of the processors and determines a weight for the tasks given by Θ. If the addition of a task to a processor makes the task set allocated to that processor unschedulable then that processor is skipped. The weight Θ is the difference in Sys-clock frequencies before and after the assignment of the task and assigns the maximum of the differences as Θ. This Θ calculation is done for each of the processors and the task having the maximum weight is chosen to be the next task to be allocated from Γ. Compared to WFD where only utilization of the tasks are considered for allocation, SFAA considers also the effect of periods of the tasks on the operating frequency. The weight Θ effectively captures the increase in operating frequency due to utilization as well as the period of the task using Sys-clock. After the weight calculation, the processor that the task with maximum weight should be allocated is selected. This is done by selecting the processor with the minimum Sys-clock frequency so that the increase in global operating frequency is minimized. After allocating the task, it is removed from Γ. This process is repeated for the total number of tasks (n) in Γ. SFAA returns β which is the global operating frequency and P the task partition.

6. Evaluation Results

6.1. Simulation Settings

For evaluating the benefits of using the proposed algorithm, SFAA, we conducted a rigorous simulation by varying the number of cores and the number of tasks in a given scenario. 25 tasks are randomly generated for dual-core and quad-core scenarios, and 80 tasks are

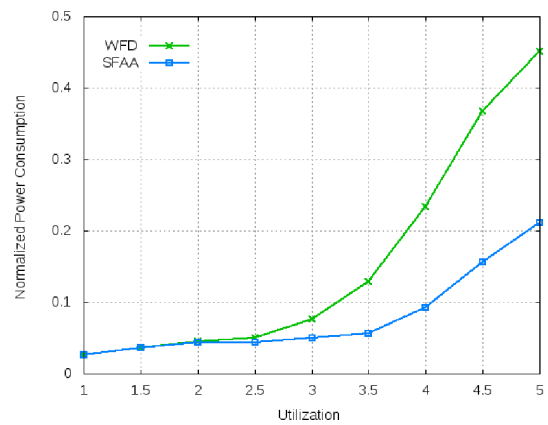


Figure 4: Normalized power consumption performance of WFD and SFAA for 80 Tasks with an octa-core processor.

randomly generated for octa-core case. The period of each task has a uniform probability of having short ($1ms - 10ms$), medium ($10ms - 100ms$), and long ($100ms - 1000ms$) period as mentioned in, [1, 34], and Task periods are uniformly distributed in each range. In multiprocessor real-time system analysis, for a given task set, it is often helpful to determine the largest utilization among all the tasks in the set. This parameter, called the *utilization factor*, will be denoted by χ and defined by $\chi = max(u_i)\ \forall\ \tau_i \in \Gamma$. In our simulations we set χ to 0.5. We also varied U_{total}, the total utilization of given tasks, between $0.1m$ to $0.6m$ every $0.05m$ (in Figures 5 and 6). In Figure 4, the utilization U_{total} was varied from $0.125m$ to $0.625m$ every $0.0625m$. There is a natural constraint on the possible values of χ: $U_{total}/n \leq \chi \leq 1.0$, which corresponds to having no constraint on (or knowledge of) upper bounds on individual task utilization. Task utilizations u_i are generated uniformly in the interval $[0.001, \chi]$ while making sure that $\Sigma_i u_i = U_{total}$. Each data point is obtained by averaging sum of all results from 100 iterations. The results we present are obtained using energy-frequency

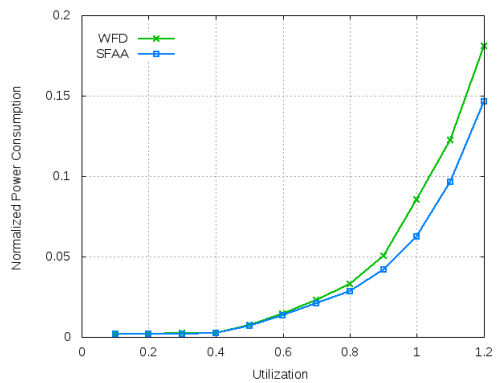


Figure 5: Normalized power consumption performance of WFD and SFAA for 25 Tasks with a dual-core processor.

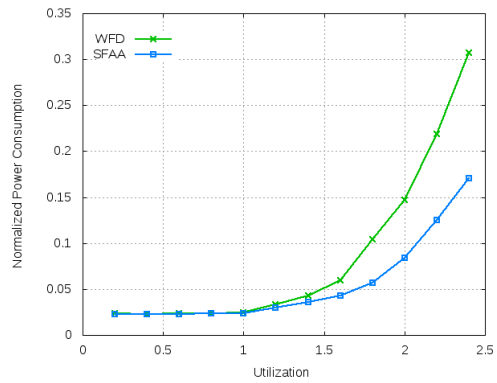


Figure 6: Normalized power consumption performance of WFD and SFAA for 25 Tasks with a quad-core processor.

relationship from an *actual* DVFS architecture, namely the NVIDIA Tegra2 processor.

6.2. Results and Discussion

For the first experiment, we set the number of cores to 8 and number of tasks to 80 as conducted in [1]. Figure 4 shows the results of the first experiment. We compared the performance of WFD [1] and SFAA. We see that as the utilization increases SFAA performs better. SFAA can save up to *55.6%* of power consumption compared to the WFD-based task allocation algorithm, where U_{total} is 5. As U_{total} increases, tasks can have higher utilization. Hence the allocation of tasks have a more pronounced effect on the operating frequency. For tasks with larger utilization, allocating right tasks together is very significant as mentioned in Section 5.1.

Figure 5 and Figure 6 depict the results for the dual-core processor and the quad-core processor, respectively. Randomly generated 25 tasks are used for these experiments. Although we performed tests with varying number of tasks, we show the results of 25-task cases due to the space constraint. However, the general trend in total utilization still holds. Since Tegra 2 processor has a dual-core processor and a quad-core Tegra 3 will be released soon [14], in this experiment we consider processors with 2 and 4 cores to reflect the current embedded multi-core processors. In Figure 5, upto *17%* of power can be saved by using SFAA compared to when using WFD. Figure 6 also shows that SFAA is upto *45.2%* better than WFD. The trend shown in Figure 4 can be observed in Figure 5 and 6, too. The same explanation can be applied here.

7. Concluding Remarks

In this paper, we have considered the problem of energy-aware partitioning for real-time tasks that are scheduled according to Rate-Monotonic Scheduling (RMS) on CMP platforms with a single voltage/frequency domain. The voltage island model with a global voltage/frequency constraint suggests that the core with maximum load becomes the critical factor in determining the operating frequency which determines the overall CMP energy consumption. This implies that load balancing could be helpful and it has been shown that the Worst-Fit Decreasing (WFD) heuristic results in nearly balanced partitions [12]. In this paper, we prove that WFD when RMS is used is a 1.71 approximation algorithm for partitioned fixed-priority preemptive scheduling. We then show that WFD does not always result in the energy-minimizing partitioning as it does not consider task periods during allocation. We propose a Single-clock domain multi-processor Frequency Assignment Algorithm (SFAA) determines a globally energy-efficient frequency by including task period relationships. Our experimental evaluation verified the effectiveness of our solution to reduce the energy consumption on CMP platforms. The experimental evaluation uses a power model developed using energy-consumption measurements from NVIDIA's Tegra 2 processor. We have shown that SFAA can save up to 55% more power compared to WFD for an octa-core processor.

It is to be noted that while the theoretical results and proposed algorithm assume infinite operating frequency granularity, they are still valid with discrete operating points. The lowering of the operating frequency will only result in a lower operating point being used. However, there could be scenarios where a much higher operating point is chosen than desired due to lack of operating-point granularity in the processor. In such cases, there will be idle-duration that cannot be exploited using DVFS. However, this idle-duration could be used for dynamic sleep. We will be exploring this in our future work. In battery powered systems such as mobile phones and tablets, devices apart from the processor such as the display and communication radios consume energy as well. As part of future work, we are looking to explore mechanisms for reducing power consumption of such devices as well.

References

[1] T. A. AlEnawy and H. Aydin. Energy-aware task allocation for rate monotonic scheduling. volume 0, pages 213–223, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[2] D. Bautista, J. Sahuquillo, H. Hassan, S. Petit, and J. Duato. A simple power-aware scheduling for multicore systems when running real-time applications.

[3] S. Borkar. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 746–749, New York, NY, USA, 2007. ACM.

[4] T. Chantem, R. Dick, and X. Hu. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 288 –293, march 2008.

[5] V. Chaturvedi and G. Quan. Leakage conscious dvs scheduling for peak temperature minimization. In *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*, pages 135 –140, jan. 2011.

[6] J.-J. Chen and C.-F. Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms. *Real-Time Computing Systems and Applications, International Workshop on*, 0:28–38, 2007.

[7] J.-J. Chen, C.-Y. Yang, H.-I. Lu, and T.-W. Kuo. Approximation algorithms for multiprocessor energy-efficient scheduling of periodic real-time tasks with uncertain task execution time. In *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE*, pages 13 –23, april 2008.

[8] V. Devadas and H. Aydin. Coordinated power management of periodic real-time tasks on chip multiprocessors. In *International Conference on Green Computing*, 2010.

[9] J. Dorsey, S. Searles, M. Ciraula, S. Johnson, N. Bujanos, D. Wu, M. Braganza, S. Meyers, E. Fang, and R. Kumar. An integrated quad-core opteron processor. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 102 –103, feb. 2007.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[11] R. L. Graham and R. L. Grahamt. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

[12] A. Hakan and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 113.2, Washington, DC, USA, 2003. IEEE Computer Society.

[13] S. Herbert. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *in International Symposium on Low Power Electronics and Design*, 2007.

[14] http://www.engadget.com/2011/01/24/nvidia-tegra-3-equipped-with-1-5ghz-quad-core-madness-teased b/. Nvidia tegra 3 confirmed.

[15] http://www.intel.com/design/intarch/xeon/ specifications xeon.htm. Intel xeon specifications.

[16] http://www.intel.com/products/processor/corei7/ specifications.htm. Intel i7 processor specifications.

[17] http://www/motorola.com. Motorola xoom android tablet.

[18] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4):41, 2007.

[19] R. Jejurikar and R. Gupta. Procrastination scheduling in fixed priority real-time systems. *SIGPLAN Not.*, 39(7):57–66, 2004.

[20] J.Y.-T.Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic real-time tasks. *Performance Evaluation*, 1982.

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

[22] R. Kumar and G. Hinton. A family of 45nm ia processors. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 58 –59, feb. 2009.

[23] J. Lee and N. S. Kim. Optimizing throughput of power- and thermal-constrained multicore processors using dvfs and per-core power-gating. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 47–50, New York, NY, USA, 2009. ACM.

[24] Y.-H. Lee, K. P. Reddy, and C. M. Krishna. Scheduling techniques for reducing leakage power in hard real-time systems. *Real-Time Systems Euromicro Conference on*, 0:105, 2003.

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

[26] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.

[27] J. M. Lopez, M. Garcia, J. L. Diaz, and D. F. Garcia. Utilization bounds for multiprocessor rate-monotonic scheduling. *Real-Time Syst.*, 24(1):5–28, 2003.

[28] J. R. Lorch and A. J. Smith. Improving dynamic voltage scaling algorithms with pace. In *SIGMETRICS '01: Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–61, New York, NY, USA, 2001. ACM.

[29] J. R. Lorch and A. J. Smith. Operating system modifications for task-based speed and voltage. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 215–229, New York, NY, USA, 2003. ACM.

[30] L. Mosley. Power delivery challenges for multicore processors. In *Proceedings of CARTS*, 2008.

[31] A. Naveh, E. Rotem, A. Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar. Power and thermal management in the intel core duo processor. volume 10, 2006.

[32] X. Qi and D. Zhu. Power management for real-time embedded systems on block-partitioned multicore platforms. In *Proceedings of the 2008 International Conference on Embedded Software and Systems*, pages 110–117, Washington, DC, USA, 2008. IEEE Computer Society.

[33] A. Rowe, K. Lakshmanan, H. Zhu, and R. Rajkumar. Rate-harmonized scheduling for saving energy. In *RTSS '08: Proceedings of the 2008 Real-Time Systems Symposium*, pages 113–122, Washington, DC, USA, 2008. IEEE Computer Society.

[34] S. Saewong and R. Rajkumar. Practical voltage-scaling for fixed-priority rt-systems. volume 0, page 106, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[35] E. Seo, J. Jeong, S. Park, and J. Lee. Energy efficient scheduling of real-time tasks on multicore processors. *Parallel and Distributed Systems, IEEE Transactions on*, 19(11):1540 –1552, nov. 2008.

[36] J. Strosnider, J. Lehoczky, and L. Sha. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *Computers, IEEE Transactions on*, 44(1):73 –91, jan 1995.

[37] N. whitepaper. The benefits of multiple cpu cores in mobile devices.

[38] C.-Y. Yang, J.-J. Chen, and T.-W. Kuo. An approximation algorithm for energy-efficient scheduling on a chip multiprocessor. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 468 – 473 Vol. 1, march 2005.

[39] C.-Y. Yang, J.-J. Chen, T.-W. Kuo, and L. Thiele. An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 694 –699, april 2009.

[40] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, page 374, Washington, DC, USA, 1995. IEEE Computer Society.

[41] Y. Zhu and F. Mueller. Feedback edf scheduling exploiting dynamic voltage scaling. In *RTAS '04: Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, page 84, Washington, DC, USA, 2004. IEEE Computer Society.