Contents lists available at SciVerse ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

A heuristic energy-aware approach for hard real-time systems on multi-core platforms

Da He *, Wolfgang Mueller

University of Paderborn/C-Lab, Fuerstenallee 11, 33102 Paderborn, Germany

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Dynamic power management
Dynamic voltage and frequency scaling
Hard real-time systems
Cluster-based multi-core processor

ABSTRACT

Due to the rapidly growing requirements of low power consumption and long battery life, the energy efficiency is becoming one of the most important concerns in the electronic system design. At the system level, the Dynamic Power Management (DPM) and Dynamic Voltage (and Frequency) Scaling (DVS) are two widely applied run-time techniques to adjust the trade-off between the system performance and power dissipation. In addition, the chip multi-core processor platforms have become the de-facto solution to cope with the continuous increase of the system complexity. In this article, we study the problem of combined application of DPM and DVS in the context of hard real-time systems on cluster-based multi-core processor platforms. We propose a heuristic algorithm based on the simulated annealing approach and introduce its online execution making the system adaptive to the run-time changes. Our approach considers multiple low power states with non-negligible state switching overhead. The experimental results show that our algorithm can significantly reduce the power consumption in comparison with existing algorithms.

1. Introduction

In recent years, the power consumption of modern electronic systems (especially battery-driven systems) is becoming one of the most important design concerns. Low energy consumption, long battery life and low heat dissipation are major development requirements and objectives to reduce the system operation costs. From the system level point of view, the Dynamic Power Management (DPM) and Dynamic Voltage (and Frequency) Scaling (DVS) are two well established techniques to obtain the best trade-off between the system performance and power consumption during run-time. In general, the main idea behind DPM tries to selectively shut down the unused system components and wake them up when required. Since the switching on/off process usually needs to retain the register contents and stabilize the power supply, a careless shutdown is not always justified. For instance, the entry latency of deep-sleep mode on Intel® PXA270 takes 600 μs [1] and is obviously non-negligible if the task execution time is in the same order of magnitude. Thus, to capture this issue Benini et al. [2] introduced the concept of break even time, which describes the time needed at least to stay at a low power state to compensate the switching overhead. In contrast to the DPM, the DVS technique is applied while the components are in the active state and tries to slow down them to achieve power saving. In CMOS-based technology, if only dynamic power consumption is considered, the energy consumption of a component over a time interval is a convex and increasing function of speed (frequency). Due to the continuous advancement in deep sub-micron process technology towards nanoscale circuits, the leakage power becomes dominant. Thus, the energy consumption becomes merely a convex function. The critical speed is defined to cover this aspect [3], i.e. no tasks should ever run below this speed. Note that we are not interested in scaling down the frequency while keeping the supply voltage, because it is not beneficial from the energy saving point of view.[1] However, it may provide advantages in terms of thermal control to extend battery life. Since this work focuses on the energy aspect, scaling the frequency alone is out of scope of this article.

In general, both DPM and DVS have to be used with great caution in the context of hard real-time systems due to timing constraints. An unjustified shutdown or slowdown may cause a delayed task execution and therefore a deadline miss. Furthermore, some studies [4–6] have reported that the DPM- and DVS-strategies are working contradictory, i.e. the DPM strategy tries to finish the tasks as fast as possible, so that more idle time is available for staying at the sleep states, while the DVS strategy attempts to complete the tasks as

[1] Without changing the supply voltage, the dynamic power consumption decreases only linearly with regard to the speed, however, this also results in the (linear) runtime increase. Hence, there is no gain in terms of the energy saving.

* Corresponding author. Tel.: +49 5251606128.
E-mail addresses: da.he@c-lab.de (D. He), wolfgang@acm.org (W. Mueller).

0141-9331/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.micpro.2013.04.007

slow as possible to reduce the active energy consumption.[2] In fact, the problem of the optimal application of DPM and DVS for hard real-time tasks is $\mathcal{NP}$-hard [7].

In the early days, silicon vendors were continuously pursuing high speed single-core processor to deal with growing performance requirements. However, nowadays more and more focus has been put on the multi-core processor platform due to lower power dissipation. A convincing confirmation was the cancellation of Tejas and the move towards multi-core platforms by Intel in May 2004 [8]. In the context of multi-core platforms, the DPM and DVS can be applied in different levels, either on the entire processor chip or on the individual core. They are referred to as the full-chip platforms and per-core platforms, respectively. In the early years, full-chip platforms were commonly adopted because of the cost-efficiency of a shared power supply net. However, they lack the flexibility for power management, because all the cores have to operate at the same speed (e.g. Intel Core™ 2 Quad [9]) simultaneously. With the introduction of *Frequency/Voltage Island* and on-chip voltage regulator, per-core platforms (e.g. AMD Phenom™ Quad-Core [10]) gain more and more interests. However, it suffers the problem of high implementation costs, which could become impractical if the number of cores dramatically increases. In this article our focus is on the cluster-based multi-core platform, which combines the advantages of the per-core and full-chip platforms and offers the best compromise. The cores are divided into clusters and the cores in the same cluster must operate at the same speed. Clearly, the cluster-based multi-core platform is a general form of the per-core and full-chip platforms. Hence, the approach presented in this article is applicable for them as well. Speaking of multi-core real-time systems, there are two types of real-time scheduling algorithms [8]: the partitioned scheduling and the global scheduling. In this article we concentrate on the former, because it provides the major advantage that the well-established single-core processor real-time scheduling, such as *Earliest Deadline First* (EDF) and *Rate Monotonic* (RM), can be adopted.

This article contains two main contributions. Firstly, we propose a simulated annealing based heuristic algorithm to minimize the energy consumption of hard real-time systems using DPM and DVS on cluster-based multi-core platforms, which was rarely addressed in the existing work. In addition, our algorithm considers multiple sleep states with non-negligible state switching overhead for both DPM and DVS techniques, which is often ignored in existing studies as well. Furthermore, in the context of DPM/DVS based energy-aware real-time scheduling there are online and offline approaches. Obviously, the online approaches are more advanced in terms of the flexibility, since they are adaptive to the system changes. However, in the literature, most of the online approaches consider only dynamic slack and lack the ability to explore the static slack, because the sophisticated static slack exploration algorithms are usually very time-consuming. In this work, the second main contribution is to propose a technique allowing our algorithm to be executed in an adaptive fashion, which enables a completely online solution to explore the static and dynamic slack with logarithmic time overhead at each scheduling point.

The remainder of this article is organized as follows. The Section 2 gives an overview of related work. In the Section 3 we formally define the system model and the problem. The Section 4 describes the details of the main algorithm. Afterwards, the Section 5 shows how the algorithm could be performed in an adaptive fashion with logarithmic complexity and the Section 6 extends the technique by taking the non-negligible DVS state switching overhead into consideration. Finally, before we conclude the article in the Section 8, the experiment results are presented in the Section 7.

2. Related work

There have been extensive research work in the field of applying DPM and DVS in real-time systems. In the context of single-core processor platforms the existing studies can be roughly divided into three categories, the DVS-only, DPM-only and DVS/DPM-combined approaches. By considering the DVS-only approaches, the primary focus is on the power reduction of the processors. One of the earliest optimal offline DVS algorithms with polynomial complexity is proposed by Yao et al. [11]. Zhang et al. [12] introduced a full polynomial time approximation algorithm. A collaborative approach based on the operating system and compiler is proposed in [13], where the decision of frequency adjustment is made by means of the inserted code instruments in the application. The problem of energy-aware scheduling for non-preemptive real-time tasks is addressed in [14]. In order to cover the resource sharing aspect in real-time systems, the authors of [15] proposed a static dual speed algorithm based on the stack resource policy. A dynamic speed adjustment scheme and a dynamic slack reclaiming algorithm are presented as well to further reduce the energy consumption at run-time. Zhong and Xu [16] proposed a polynomial time approximation scheme using dynamic programming technique to statically assign frequencies to tasks while minimizing the total system wide energy consumption. The authors of [17] proposed a power aware real-time scheduling with three components: (i) an offline algorithm to compute the optimal speed assignment, (ii) an online dynamic slack reclaiming framework and (iii) an online aggressive speed adjustment scheme that takes the advantage of the dynamic slack of future tasks. Moreover, the work [18] has shown an online DVS approach by exploiting the dynamic slack with a complexity of $O(1)$. A performance comparison of different DVS approaches for real-time systems under a uniform simulation framework is given in [19]. A comprehensive survey of the DVS-only approaches for both single-core and multi-core processor platforms is given in [20].

The DPM-only work is mostly proposed for device usage scheduling. The survey [2] gives the basics of the DPM technique. In [21] an online DPM algorithm in conjunction with EDF is proposed, where the tasks are procrastinated as much as possible to create large device idle intervals. The work by Swaminathan and Chakrabarty [22] proposed an offline optimal device scheduling algorithm for hard real-time systems based on pruning techniques. A heuristic search algorithm is proposed as well to find near optimal solution. The authors of [23] proposed a rate-harmonized task schedule, where an artificial task period is introduced and all tasks are only eligible to execute at the new period boundaries. This has the effect that some ready tasks may be delayed in order to prolong the current idle time for maximizing the DPM usage. Lampka et al. [24] introduced a dynamic counter approach to decide the number of upcoming events and therefore bound the future workload. Based on this information the shutdown policy can be safely applied.

Moreover, the relationship of DPM and DVS attracted more attentions in the context of system-wide energy efficient real-time scheduling. Generally, DPM and DVS are applied for the devices and processors, respectively. Devadas and Aydin [4] studied the exact interplay of DPM and DVS. However, their focus is on the frame-based task model, where all tasks share common period/deadline. In the work of Jejurikar and Gupta [25] the concept of critical speed was introduced. The task should never run under the critical speed, otherwise the power consumption increases. However, they ignored DPM state switching overhead. Other related work are presented in [5,26]. However, none of them is fully online and they work only with a fixed real-time schedule EDF.

[2] The active energy consumption of a processor is the energy consumed during the task execution.