Technical Report

Device Power Management for Real-Time Embedded Systems

Muhammad Ali Awan

Stefan M. Petters

HURRAY-TR-120603

Version:

Date: 06-05-2012

Device Power Management for Real-Time Embedded Systems

Muhammad Ali Awan, Stefan M. Petters

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.hurray.isep.ipp.pt

Abstract

A large part of power dissipation in a system is generated by I/O devices. Increasingly these devices provide power saving mechanisms to inter alia enhance battery life. While I/O device scheduling has been studied in the past for realtime systems, the use of energy resources by these scheduling algorithms may be improved. These approaches are crafted considering a huge overhead of device transition. The technology enhancement has allowed the hardware vendors to reduce the device transition overhead and energy consumption. We propose an intra-task device scheduling algorithm for real time systems that allows to shut-down devices while ensuring the system schedulability. Our results show an energy gain of up to 90% in the best case when compared to the state-of-the-art.

Device Power Management for Real-Time Embedded Systems

Muhammad Ali Awan Stefan M. Petters
CISTER Research Unit, ISEP-IPP Porto, Portugal
maan,smp@isep.ipp.pt

Embedded devices are designed to perform a set of functions. These systems interact with their environment and use input/output (I/O) devices. Typical examples of such systems are cars, satellites or mobile phones. Real-time (RT) embedded systems have additional timing constraints, which are required to be met on top of functional aspects for the overall system to be considered correct. Beyond RT constraints many embedded systems have limited or intermittent power supply. Therefore, energy efficiency is an important aspect that needs to be considered in the design process of such RT systems.

The demand of extra functionality on a single embedded system also results in an increased number of I/O devices. As I/O devices consume considerable amount of energy, they often come with power saving states to minimise their energy consumption. A device can only operate in the active mode, and its transition into and out of sleep state incurs both time and energy overheads. Moreover, the request instant and access interval of the device can usually not be determined beforehand. In order to guarantee the temporal correctness of such RT systems, the device transition delay to bring the device up from sleep needs to be taken into account.

Device power management was extensively studied in a non-real-time setting. Benini et al. [1] divided these techniques into three main categories, 1) time-out based, 2) predictive and 3) stochastic. Time-out based algorithms shutdown the devices when they are idle for a specified threshold. The system wakes up the device on the next request of the task. Predictive techniques adapt themselves with the varying workload of the system. Stochastic methods model the requests behaviour with different probabilistic distributions. The device shut-down times are estimated by solving the stochastic models such as Markov chains. Most of the techniques mentioned above turn-on the device when requested by the application/task.

A RT system does not have such leverage, as a delay in device transition from its sleep state my cause the task to miss its deadline. Therefore, in traditional RT device-scheduling algorithms all devices requested by a task are turned on before the start of its execution and kept active throughout the entire execution time of the corresponding task. This category of device scheduling is known as inter-task device scheduling. However, most devices are used for very short intervals of time thus resulting in wasted energy. Opposed to this, in intra-task device scheduling a device is only turned on when it is requested by the task. However, no such techniques are developed for RT systems with strict timeliness constraints. Major issue is the overhead of the device shut-down transitions and secondly, the device usage instance is not known a priori.

The unused capacity in a system is called system slack. System slack can be categorised as static and dynamic slack. Static slack exists due to a spare capacity in the system, as it is not loaded with what can be guaranteed by the schedulability test. However, dynamic slack is generated online and has two parts. 1) RT tasks usually do not execute for the their worst-case execution time and generate execution slack. 2) Similarly, their arrival is usually delayed beyond the minimum-inter-arrival time and produce sporadic slack. The detailed discussion about these types of slacks is given in [2].

This research effort proposes a static slack container algorithm (SSC) to exploit intra-task device scheduling for RT embedded system. SSC uses the system slack to compensate for the delays of device transitions and guarantee the schedulability of the system. The brief overview of the algorithm is given below. For the detailed algorithms reader is directed to the original publication [3].

Initially the algorithm computes the device budget D_b offline that comes from the static slack in the system. The device budget D_b of the system is the maximum available spare time that can be used to compensate for the devices transition delays without causing any task to miss its deadline under worst-case assumptions. It is computed from the demand bound function [3], [4].

Our algorithm uses a sporadic task model, in which a task τ_i is categorised by $\langle C_i, T_i, Z_i \rangle$, where C_i, T_i and Z_i are the worst-case execution time, minimum-inter-arrival time and the device used by τ_i respectively. We assume one device per task and it is used at most once during any instance of τ_i execution. Each independent task releases its instance after every T_i. The transition delay of Z_i is denoted as tr_z. For simplicity sake, τ_i deadline is considered equal to T_i.

For the ease of presentation here, we assume all the devices are compatible with the intra-task device scheduling and hold this inequality $T_i - C_i \geq tr_z$. Our SSC algorithm turns off the device Z_i when it has been used by τ_i and the next utilisation time of Z_i is greater or equal to its transition delay tr_z. A timer is set accordingly to wake up the device. An interrupt service routine (ISR) signals the system when the timer corresponding to Z_i expires. Whenever the timer associated to any Z_i expires, a system consults the following principles. The effective slack mentioned in the principles corresponds to the execution slack with a deadline less than the absolute deadline of τ_i using Z_i.

1) Principle 1: $D_b \geq tr_z$
   - Device is kept off
   - Timer associated to this device is removed

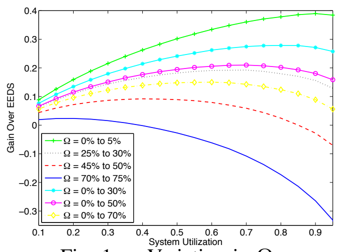

Fig. 1. Variation in Ω

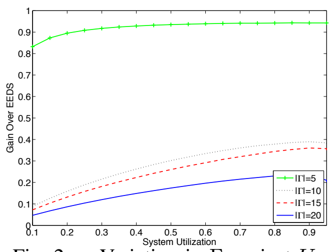

Fig. 2. Variation in Γ against U

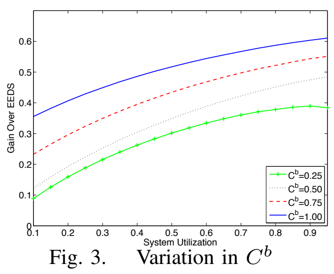

Fig. 3. Variation in C^b

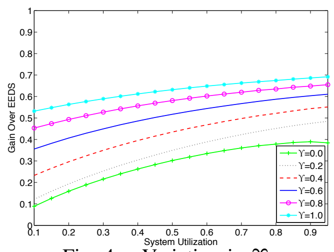

Fig. 4. Variation in Υ

2) Principle 2: $D_b < tr_z \&\&$ Effective slack $\neq 0$
   - Keep the device off
   - Extend the timer equal to the size of effective slack
3) Principle 3: $D_b < tr_z \&\&$ Effective slack $= 0$
   - Turn on the device
   - Remove the timer associated to this device

It has been shown in the original publication [3] that τ_i requesting Z_i during a sleep state will maintain its schedulability with the given principles. The device budget in the system is replenished when the system is in idle state. We have also proposed a device budget reclamation algorithm. D_b by definition is the highest priority budget in the system. When Z_i is allocated a part of D_b to compensate for its transition delay, the analysis assumes no other task is executing and/or waiting for its device transition during this interval. In the case when another job is executing during the transition phase of Z_i, the device budget may be reclaimed depending on the priority of the workload executed in this interval. Similarly, in case of multiple tasks are waiting for their device active state, a system should only consider a budget consumption of single device in the overlapping period as their wake-up transition happens in parallel. The complexity of our device scheduling algorithm and budget reclamation algorithm is $O(\ell)$, where ℓ is the number of tasks in the system.

We have extended the discrete event simulator SPARTS V-2.0 (Simulator for Power Aware and Real-Time System) [5] for the experiments to evaluate the effectiveness of our proposed algorithm. To cover the wide variety of applications we have used task-sets ranging from a larger number of fine grained small tasks (20) to a small umber of coarse grained tasks (5). We have varied actual execution time, sporadic delay and device usage time with C^b, Υ and Ω respectively. The actual execution time of τ_i and its sporadic delay is varied within a range of $[C^b * C_i; C_i]$ and $[T_i; T_i + T_i * \Upsilon]$ respectively. The default values of $C^b = 1, \Upsilon = 0, \Omega =$ (0% to 5%) and task-set size $|T| = 10$. Energy Efficient Device Scheduler (EEDS) [6] is selected from state-of-the-art for the comparison.

The effect of variation in device usage time Ω (given as a percentage of task's actual-execution time) on the gain of SSC over EEDS is illustrated in Figure 1. SSC performance dominates if τ_i uses their corresponding Z_i for the a small percentages of C_i. Gain of SSC with a task-set size variation is demonstrated Figure 2. Smaller task-set sizes are favourable to SSC and the gain of SSC increases with the system utilisation. EEDS cannot extend sleep intervals of the devices at higher utilisation. Moreover, with an increase in the task-set size, D_b has to service extra devices and thus the gain of SSC decreases.

The effect of variation in actual execution time of each task is shown in Figure 3. The low value of C^b corresponds to high execution slack and vice versa. The gain of SSC reduces with a decrease in C^b for an obvious reason that if tasks finish their execution earlier than C_i, EEDS has a chance to turn their corresponding devices off immediately. The variation of sporadic slack is observed in Figure 4. Large value of Υ corresponds to high sporadic slack. Extra sporadic slack allows for larger gains in energy consumption. SSC makes an efficient use of the sporadic slack because device is only woken up on demand and kept in sleep mode if the task arrives later than its T_i. However, EEDS has the requirement to keep the device on during C_i; therefore, devices are woken up assuming a worst-case scenario of task arrival after every T_i.

I. Conclusions

This paper presents the intra-task device scheduling algorithm, which requests the device on demand rather than keeping it unnecessary active throughout the execution of its corresponding job. Our algorithm makes explicit use of static and dynamic slack. Our extensive evaluation demonstrates its efficiency. Furthermore, it has low complexity when compared to the state-of-the-art and reduces the assumptions that restrict the practical implementation of these approaches. In the future, we intended to further relax the assumptions made in this research effort that will enhance the applicability of this algorithm to more versatile systems. Our goal is to allow device sharing among jobs and add flexibility to use multiple devices in a single job.

Acknowledgment

This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within REPOMUC project, ref. FCOMP–01-0124-FEDER-015050, and by FCT and the EU ARTEMIS JU funding, within RECOMP project, ref. ARTEMIS/0202/2009, JU grant nr. 100202.

References

[1] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *Trans. Very Large Scale Integration Syst.*, vol. 8, no. 3, pp. 299 –316, june 2000. 1

[2] M. A. Awan and S. M. Petters, "Enhanced race-to-halt: A leakage-aware energy management approach for dynamic priority systems," in *23rd ECRTS*, 2011, pp. 92–101. 1

[3] ——, "Online intra-task device scheduling for hard real-time systems," in *SIES12*, june 2012, pp. 1–8. 1, 2

[4] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *J. Real–Time Syst.*, vol. 2, pp. 301–324, 1990. 1

[5] B. Nikolic, M. A. Awan, and S. M. Petters, "SPARTS: Simulator for power aware and real-time systems," in *8th IEEE Int. Conf. Emb. Softw. & Syst.* Changsha, China: IEEE, Nov 2011. 2

[6] H. Cheng and S. Goddard, "Online energy-aware i/o device scheduling for hard real-time systems," in *43rd DATE*. Leuven, Belgium: European Design and Automation Association, 2006, pp. 1055–1060. 2