Energy Efficient Real-time Scheduling Algorithm for Microcontroller Units

Douglas Lautner, Xiayu Hua, Miao Song, Scott DeBates, Shangping Ren
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616, USA
e-mail:{dlautner, xhua}@hawk.iit.edu, {msong2, scottdebates}@motorola.com, ren@iit.edu

***Abstract*—Mobile smart devices are advancing with stronger demands of high energy efficiency and longer battery life. Utilizing energy-efficient microcontroller units (MCU) in a mobile device for always-on functionalities is proved to be an effective solution. MCUs have the ability to switch between different running modes dynamically enabling them to have outstanding low power performance while performing real-time sensing tasks. Besides hardware optimization, balancing energy efficiency and quality of service on a MCU lies within a well designed scheduling algorithm. In this paper, we formally define, model and derive a proper scheduling algorithm that guarantees that the task set is schedulable and minimizes power consumption. Our findings open up additional research of optimizing real-time scheduling algorithms with less energy consumption on an MCU while guaranteeing the quality of service, i.e., the schedulability of the given real-time task set.**

Keywords–Mobile device; embedded system; energy efficiency; real-time scheduling

I. Introduction

Battery life is one of the most important features for mobile smart devices. To improve the energy efficiency, mobile smart devices introduced low-power processors, such as microcontroller unit (MCU) based sensor fusion core [1], to perform non-critical control and calculation tasks with lower power consumption due to its special design. Placing a MCU into "shutdown" mode, i.e., turn off the MCU, can stop the power consumption. However, shutting down and rebooting the MCU not only introduces extra energy consumption, but also causes execution delay that may affect the tasks' real-time requirements. On the other hand, keeping the MCU in the standby and idle mode continuously consumes energy, however it guarantees tasks can be executed without delay. schedule the shutdown and reboot on a MCU with the goal of 1) guaranteeing the schedulability of hard real-time tasks and 2) minimizing the energy consumption is a big challenge.

In terms of saving energy by changing the processor into different modes, Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) are two popular research areas [2]–[4]. Most existing DVFS-related work focuses on minimizing the energy consumption while optimizing the quality of service [5]–[7]. Also, assumptions such that resources are always available during performance change [8], or the performance can be switched immediately to any level between the low and maximal performance [9] make existing real-time scheduling analyses with DVFS-enabled resource compliment to our case. While DVFS requires the resource to be always available, DPM-based solutions allow the system to shutdown the resource [3]. DPM solutions can also be implemented with DVFS (if the hardware allows) [3] to further reduce the energy consumption. However, most existing work on DPM targets on minimizing the task set's makespan or maximizing the system's throughput while optimizing the energy consumption. To the best of our knowledge, scheduling hard real-time tasks on a MCU and maximizing the MCU's energy efficiency is still a challenge yet to be solved.

Inspired by the existing work on DVFS and DPM scheduling issues, we investigate, discuss and analyze the challenges of deploying hard real-time tasks on MCU in an energy efficient way, given MCU's ability to switching running mode between standby and shutdown. In particular, we provide the energy-saving condition under which shutting down the MCU saves more energy than leaving it in standby mode and idle. Without creating an impact to the performance, we require the shutdowns should not break the schedulability of the given task set. We have further studied three possible approaches to specifically schedule shutdown time and duration for a MCU to guarantee the task set's schedulability while minimizing the overall energy consumption.

The rest of paper is organized as follows. In Section II, we propose the task model, the resource assumptions and the research goal. Section III proposes and analyzes our three possible solutions in detail. In Section IV, we conclude our current work and propose future extensions.

II. Modeling and Problem Formulation

In this section, we define the task and resource models, respectively, and provide the formal problem formulation.

A. Task Model

We use the task model defined in [10] which makes the following assumptions: 1. Tasks are periodically available with a fixed period; 2. Each task requires a fixed amount of time to finish; 3. Each task's relative deadline (with respect to the time that it becomes available) is the task's period. Specifically,

 ISBN: 978-1-61208-608-8

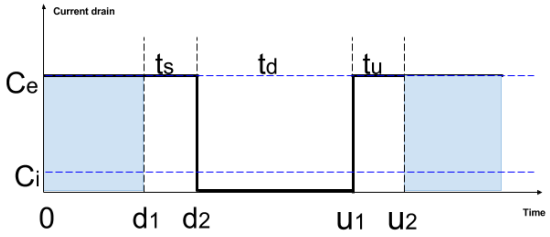

Figure 1. Shutdown time requirement

a task is defined as $\tau(e, p)$ where e is the longest possible execution time, p is the period.

B. MCU Resource Assumptions

For a MCU resource, it may have several running modes. In this work, we assume that a MCU resource has two modes: *standby* mode that can execute jobs immediately, but consumes power even when idle; *shutdown* mode that cannot execute jobs and does not consume power.

We further assume that a switch between the two modes has time overhead and consumes extra energy.

As the resource model varies in different possible solutions, we define the resource models in each solution individually.

C. Problem Formulation

For a given task set and a MCU resource, schedule the switch of resource modes between standby mode and shutdown mode to: 1) Guarantee that each job of a task in the task set meets its deadline, and 2) Minimize the power consumption.

III. Possible Solutions

In this section, we first calculate the minimal shutdown duration requirement such that a shutdown behavior: 1) Is practical for a MCU to perform from a hardware perspective and 2) Does save power compared to leaving the MCU idle. Then, we provide three possible solutions, as well as our preliminary research results, of how to schedule the shutdown for a MCU with running real-time task sets. For the first solution, we have finished the theoretical analysis for tasks under Earliest Deadline First scheduler. For the Rate Monotonic scheduler case, as well as the remaining two possible solutions, they are work in progress and we will research them in our future work.

A. The minimal shutdown duration requirement

As illustrated in Figure 1, to perform a shutting down, the MCU stops providing computing services to the tasks first (at time d_1). Before cutting the power (at time point d_2), the MCU needs to backup its runtime status, i.e, values in registers or data in the RAM, into storage in state t_s. During time interval t_d, the MCU is in shutdown mode. At time point u_1, the MCU starts the rebooting procedure during t_u and is completed at u_2. In the following sections, we call the MCU as a *computing resource* and formally define the resource model of MCU in the following definition.

Definition 1. *A MCU resource R is defined as a quad notated as $R(t_s, t_u, C_i, C_e)$ where t_s is the time duration of switching from running mode to shutdown mode and t_u is the time duration of switching from shutdown mode to running mode. C_i and C_e are the current drains when the MCU is idle or is calculating, respectively.*

As the MCU's shutdown and reboot procedure needs time and extra energy, we derive the minimal shutdown time in the following lemma.

Lemma 1. *For a given MCU resource $R(t_s, t_u, C_i, C_e)$, the minimal shutdown time T_s that saves energy is determined as:*

$$T_s > \frac{(t_s + t_u) \cdot C_e}{C_i} \tag{1}$$

Proof. As Figure 1 illustrates, the time interval that the MCU is unavailable to execute tasks starts at time point d_1 and ends at u_2, i.e., $T_s = u_2 - d_1$. If the resource is in idle time, it consumes $(u_2 - d_1) \cdot C_i$ power during this time period. On the other hand, if the MCU is shut down, during t_s it consumes $t_s \cdot C_e$ power and during t_u it consumes $t_u \cdot C_e$ power. Compared to leaving the MCU idle, shutting down the MCU saves more energy when the following condition is satisfied.

$$(u_2 - d_1) \cdot C_i > (t_s + t_u) \cdot C_e$$

Hence

$$T_s > \frac{(t_s + t_u) \cdot C_e}{C_i} \tag{2}$$

Obviously, T_s should also be longer than $t_s + t_u$ to have enough time to shutdown and then reboot the MCU. Since $C_e > C_i$, (2) also implies $T_s > t_s + t_u$. □

B. Handle the System Shutdown: When the Mode Switch is periodic

The first approach is to shutdown the MCU periodically. As illustrated in Figure 3, a periodic shutdown is performed with the period and duration of 5 and 2 time units, respectively, i.e., $S(2, 5)$. As a comparison, the case of scheduling on regular resource is illustrated in Figure 2. The advantage of this approach is obvious. To implement this approach, the scheduler only needs a timer, therefore this approach consumes minimal CPU resource compared to the approach we will study in the next section. The major challenge is to determine the shutdown duration time and the shutdown period that guarantees the schedulability of the given task set while minimizes the overall power consumption.

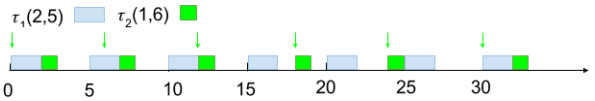

Figure 2. Regular resource

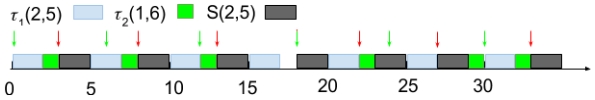

Figure 3. Periodic resource

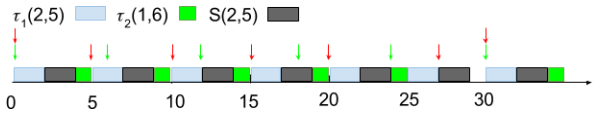

Figure 4. Resource with shutdown task insertion

Under this approach, the resource's properties match the definition of a periodic resource [11]–[13], with respect to the task set, as it switches periodically between available and unavailable. More specifically, this resource is categorized to a *fixed-pattern periodic resource* in [12] and [13]. Compared to the existing scheduling research work on periodic resources, the challenges of our current problem is to determine both the shutdown frequency and the shutdown duration which are assumed to be given in the existing work.

In our work, we calculate these two parameters to guarantee the task set's schedulability (if schedulable on a resource without shutting down) while minimizing the energy consumption. Specifically, we leverage the schedulability analyses in [12] and [13], change the resource period and downtime into variables, determine their solution domains with respect to the task set's schedulability and then find the optimal (or near optimal) solution in the solution domain that minimizes the power consumption.

A possible approach to find the optimal solution is to derive the schedulability condition from periodic resource bounds [11]. From the necessary schedulability bound, we can derive the dependency between the shutdown period and its duration under the schedulability requirement. Then, based on this dependency, we further use heuristic ways (to deliver a near optimal solution) to search the optimal pair of period and duration values.

To perform the theoretical analysis, the pessimistic way is to first calculate the bound, and use the bound to further calculate the θ parameters. Following Lemmas and theorems provide the shutdown time bound for the EDF scheduler. For convenience, we reuse the MCU notation R and redefine the resource model in the following definition for schedulability analysis pruposes.

Definition 2. *A MCU resource with periodic sleep is defined as a tuple and is denoted as $R(\theta, \pi)$ where θ is the resource available time, with respect to the task set, and π is the system sleep period.*

Note that both t_s and t_u time duration is not counted in θ as tasks are not able to be executed during these two durations.

Lemma 2. *For a given resource $R(\theta, \pi)$ and a task set $T = \{\tau_1, ..., \tau_n\}$ where τ_i represents a independent task $\tau_i(e_i, p_i)$. The minimal task period in T is denoted as p_{min}. With Earliest Deadline First scheduler, T is guaranteed to be schedulable if the following relation is valid: $\pi < \frac{p_{min} \cdot (\theta - U_T) + \theta^2}{\theta}$, where U_T is the total utilization rate of T, i.e., $U_T = \sum_{i-1}^{n} \frac{e_i}{p_i}$.*

Proof. We first introduce the linear demand bound function $ldbf(t) = U_T \cdot t$ for EDF from [14], [15]. This function describes that for any time interval with given length t, the overall resource demand, such like the overall CPU cycles needed, will not acceed $ldbf(t)$.

We then calculate in the worst case, at least how much resource can be offered by the MCU with periodic shutdown in a given time length t, which is calculated as the supply bound function $sbf(t)$. Obviously, for any time interval with same length, the worst case is that this time interval starts with a shutting down, as illustrated in Figure 5.

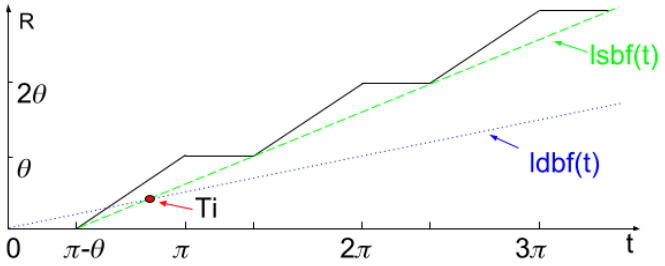

Figure 5. Demand and supply bound functions

Specifically, from time 0 to time θ, the job receives no resource. At time θ to the end of this resource period, the MCU is up and running and executes the job. As the MCU periodically turns on and off, this procedure repeats in each MCU period. We formalize this periodic procedure as:

$$sbf(t) = \begin{cases} 0, \textit{if } t < \pi - \theta, \\ \lfloor \frac{t}{\pi} \rfloor \cdot \theta + max(0, (t \ mod \ \pi - (\pi - \theta))), \textit{otherwise} \end{cases} \quad (3)$$

In (3), if the time interval is shorter than $\pi - \theta$, then in the worst case, the MCU is shut down during the entire time interval and hence there is zero resource provided by the MCU in this time interval. In the case that $t \geq \pi - \theta$, we first calculate how many complete MCU peirods are in this interval, i.e., $\lfloor \frac{t}{\pi} \rfloor$. Since for one complete MCU period, MCU provides θ executable time, this parts contributes $\lfloor \frac{t}{\pi} \rfloor \cdot \theta$ executable time. For the remaining part, i.e., $t \ mod \ \pi$, it is shorter than a period. If it's also shorter than the shutdown time, in the worst case this part contributes no executable time, otherwise it provides $t \ mod \ (\pi - \theta)$ executable time.

Since (3) is a step-function which is difficult to conduct further proof, we further derivate the linear supply bound function $lsbf(t) = \frac{\theta}{\pi} \cdot t - \theta + \frac{\theta^2}{\pi}$.

The green dot line in Figure 5 illustrates the intuition of $lsbf(t)$.

To guarantee the schedulability of a task set, the following condition should be satisfied [15], [16], i.e., when the length of a time interval is equal or longer than p_{min}, then $lsbf(t) \geq ldbf(t)$. With this condition, we derive the final conclusion. First, with (3), $ldbf(t)$ and $lsbf(t) \geq ldbf(t)$ we have

$$\frac{\theta}{\pi} \cdot t - \theta + \frac{\theta^2}{\pi} > U_T \cdot t \quad \rightarrow \quad t \geq \frac{\pi\theta - \theta^2}{\theta - U_T}$$

As illustrated in Figure 5, we denote the time point $\frac{\pi\theta - \theta^2}{\theta - U}$ as T_i. The intuition of T_i is that if a time interval is longer than T_i, then the minimal resource supply is guaranteed to be larger than the maximal possible resource demand. To guarantee the schedulability of the task set, we only need to guarantee that

 ISBN: 978-1-61208-608-8

for a time interval with length p_{min}, the resource supply is no less than the demand, therefore we have:

$$t \geq \frac{\pi\theta - \theta^2}{\theta - U} \rightarrow p_{min} \geq \frac{\pi\theta - \theta^2}{\theta - U} \rightarrow \pi < \frac{p_{min} \cdot (\theta - U_T) + \theta^2}{\theta} \quad (4)$$

□

With Lemma 1, we now have the relation between π and θ.

Theorem 1. *For a given resource $R(\theta, \pi)$ and a task set $T = \{\tau_1, ..., \tau_n\}$ where τ_i represents a independent task $\tau_i(e_i, p_i)$. The minimal task period in T is denoted as p_{min}. With Earliest Deadline First scheduler, T is guaranteed to be schedulable if the following relation is valid:*

$$\begin{cases} \pi < \frac{p_{min} \cdot (\theta - U_T) + \theta^2}{\theta} \\ \theta > \frac{(t_s + t_u) \cdot C_m}{C_i} \end{cases} \quad (5)$$

where U_T is the total utilization rate of T, i.e., $u = \sum_{i-1}^{n} \frac{e_i}{p_i}$.

With Theorem 1, we create the solution domain for π and θ. If there is no solution in this domain, then the task set is non-schedulable even without a MCU shutdown. If this domain is non-empty, the next step is to find the optimal solution ,i.e. (π, θ) value pair that can minimize the power consumption in a T's hyperperiod. The approach could be either theoretical or heuristic searching approaches, we will further study in our future work.

C. Handle the System Shutdown: Treat the Shutdown as a Special Task

Aside from turning down the MCU periodically, another possible solution is to treat the turning down as a special task. As illustrated in Figure 4, a periodic shutdown $S(2, 5)$, i.e., shutdown two time units for every five time units, is treated as a regular task. Specifically, we first determine the shutdown duration e and shutdown period p and insert the shutdown into the original task set as a special task. We then use the original scheduler to schedule both the new task set with the special task. This solution is applicable for the classical schedulers, such as RM or EDF, without changing the original system too much. Also, comparing to the periodic shutdown, the shutdown is more flexible, i.e., the shutdown can be arranged to strart from anytime within its perioid, therefore it should have both higher system utilization rate and energy efficiency compared to the periodic shutdown approach.

This solution has two challenges. First, the shutdown task is non-preemptive. Second, we still need to determine the e' and p' of the shutdown task to 1) guarantee the task set's schedulability and 2) minimize p' while maximizing e'.

For the first challenge, one possible approach is to align the shutdown task's period p' to a certain task and extends the chosen task's execution time. For example, when using a Rate Monotonic scheduler, if we align the shutdown task to the task with shortest period (i.e., highest priority) $\tau_1(e_1, p_1)$, then we have a new τ_1' where $e_1' = e_1 + e'$. After this transition, the number of tasks remains unchanged and the shutdown task is non-preemptive as it is always accompanied by the execution of τ_1 which is non-preemptive to all other tasks. Then, the problem changes to determine the longest e_i' that can still guarantee the task set's schedulability.

This approach may guarantee the schedulability and handle the non-preemptive property of the shutdown task, however the second challenge is still unsolved.

D. Handle the System Shutdown: New Scheduling Algorithm

The third possible approach is to design a new algorithm which can dynamically switch the MCU into sleep mode. This approach has the highest effect with respect to the energy efficiency as it can utilize all possible chances to put the MCU into sleep mode. However, it also has the most difficult challenges.

For this approach, the immediate problem is for the scheduler to know whether an incoming idle time is long enough to switch the MCU into sleep mode and then switch back before the release of the next job. As current schedulers only handle resource conflicts, i.e, when more jobs are ready to run but only less resources are available, a scheduler determines which jobs to run. Under this context, a scheduler is not able to be aware of the incoming job running situations. Therefore, new improvements are required for the scheduling algorithms to further determine the MCU's running modes.

The implementation of this approach is most likely to be heuristic as it is essentially an integrated linear programming solution and it is expensive resource wise. The basic approach of our solution could be to find the optimal schedule first using an integrated linear solution, and then gradually release the conditions to reach a near optimal solution, which balances both the scheduling overhead and the energy efficiency.

IV. Conclusion and Future Work

In this work, we formally formulated the scheduling problems of deploying periodic hard real-time tasks on a MCU that has multiple running modes. To achieve the goal of 1) Guaranteeing the task set's schedulability and 2) Maximizing the energy efficiency, we further provided analyses of three possible solutions, i.e., 1) Periodic shutdown, 2) Treat shutdown as a special task, and 3) Design a new scheduling algorithm. In our future work, we will continually study each of these three possible solutions and find the optimal ones that save the maximal energy while guaranteeing the schedulability.

References

[1] D. Lautner, X. Hua, S. Debates, M. Song, J. Shah, and S. Ren, "Baas (bluetooth-as-a-sensor): conception, design and implementation on mobile platforms," in *Proceedings of the Symposium on Applied Computing*. ACM, 2017, pp. 550–556.

[2] R. David, P. Bogdan, and R. Marculescu, "Dynamic power management for multicores: Case study using the intel scc," in *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*. IEEE, 2012, pp. 147–152.

[3] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal dvfs and dpm combination," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 3s, p. 111, 2014.

[4] S. Mittal, "Power management techniques for data centers: A survey," *arXiv preprint arXiv:1404.6681*, 2014.

 ISBN: 978-1-61208-608-8

[5] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 1. ACM, 2001, pp. 50–61.

[6] 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, no. 8, pp. 1467–1478, 2008.

[7] J.-J. Chen, N. Stoimenov, and L. Thiele, "Feasibility analysis of on-line dvs algorithms for scheduling arbitrary event streams," in *Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE*. IEEE, 2009, pp. 261–270.

[8] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 24, no. 1, pp. 18–28, 2005.

[9] R. Jejurikar and R. Gupta, "Energy aware task scheduling with task synchronization for embedded real time systems," in *Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems*. ACM, 2002, pp. 164–169.

[10] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[11] I. Shin and I. Lee, "Compositional real-time scheduling framework with periodic model," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, p. 30, 2008.

[12] A. K. Mok, X. Feng, and D. Chen, "Resource partition for real-time systems," in *Real-Time Technology and Applications Symposium, 2001. Proceedings. Seventh IEEE*. IEEE, 2001, pp. 75–84.

[13] S. Shirero, M. Takashi, and H. Kei, "On the schedulability conditions on partial time slots," in *Real-Time Computing Systems and Applications, 1999. RTCSA'99. Sixth International Conference on*. IEEE, 1999, pp. 166–173.

[14] A. Easwaran, M. Anand, and I. Lee, "Compositional analysis framework using edp resource models," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*. IEEE, 2007, pp. 129–138.

[15] X. Hua, C. Guo, H. Wu, D. Lautner, and S. Ren, "Schedulability analysis for real-time task set on resource with performance degradation and periodic rejuvenation," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2015 IEEE 21st International Conference on*. IEEE, 2015, pp. 197–206.

[16] X. Feng, "Design of real-time virtual resource architecture for large-scale embedded systems," Ph.D. dissertation, 2004.

 ISBN: 978-1-61208-608-8