*Proceedings of the 48th IEEE Conference on Decision and Control (CDC 2009), Shanghai, China, December 16-18, 2009.*

# Periodic Power Management Schemes for Real-Time Event Streams

Kai Huang, Luca Santinelli*, Jian-Jia Chen, Lothar Thiele, Giorgio C. Buttazzo*
Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
Email: {firstname.lastname}@tik.ee.ethz.ch
* Real-Time Systems Laboratory, Scuola Superiore Sant'Anna of Pisa, Italy
Email: {luca.santinelli, giorgio}@sssup.it

***Abstract*— Power dissipation has constrained the performance boosting of modern computer systems in the past decade. Dynamic power management (DPM) has been implemented in many systems to change the system (or device) state dynamically to reduce the power consumption. This paper explores how to efficiently and effectively reduce the energy consumption to handle event streams with hard real-time or quality of service (QoS) guarantees. We adopt Real-Time Calculus to describe the event arrival by arrival curves in the interval domain. To reduce the implementation overhead, we propose a periodic scheme to determine when to turn on/off the system (or device). This paper first presents two approaches to derive periodic scheme to cope with systems with only one event stream, in which one approach derives an optimal solution for periodic power management with higher complexity and the other derives approximated solutions with lower complexity. Then, extensions are proposed to deal with multiple event streams. Simulation results reveal the effectiveness of our approaches.**

**Keywords:** Power Management, Energy Minimization, Real-Time Event Streams, Real-Time Calculus, QoS.

## I. Introduction

Power dissipation has been an important design issue in a wide range of computer systems in the past decade. Power management with energy efficiency considerations is not only useful for mobile devices for the improvement on operating duration but also helpful for server systems for the reduction of power bills. Dynamic power consumption due to switching activities and static power consumption due to the leakage current are two major sources of power consumption of a CMOS circuit [12]. For micrometer-scale semiconductor technology, the dynamic power dominates the power consumption of a processor. However, for technology in the deep sub-micron (DSM) domain, the leakage power consumption is comparable to or even more than the dynamic power dissipation.

The dynamic voltage scaling (DVS) technique was introduced to reduce the dynamic energy consumption by trading the performance for energy savings. For DVS processors, a higher supply voltage, generally, leads to not only a higher execution speed/frequency but also higher power consumption. As a result, DVS scheduling algorithms, e.g., [2], [26], [27], tend to execute events as slowly as possible, without any violation of timing constraints. On the other hand, dynamic power management (DPM) can be applied to control the change of system mode to consume less leakage power, e.g., to a sleep mode. For DVS systems with non-negligible leakage power consumption, to minimize the energy consumption for execution, there is a *critical speed*, in which executing at any speed lower than the critical speed consumes more energy than at the critical speed [5], [12]. However, returning from the sleep mode has timing and energy overheads, due to the wakeup/shutdown of the processor and data fetch in the register/cache. For example, the Transmeta processor in 70nm technology has $483\mu J$ energy overhead and less than 2 msec timing overhead [12].

For non-DVS systems with the sleep mode, Baptiste [3] proposes an algorithm based on dynamic programming to control when to turn on/off the system for aperiodic real-time events with the same execution time. For multiple low-power modes, Augustine et al. [1] determine the mode that the processor should enter for aperiodic real-time events and propose a competitive algorithm for on-line use. Swaminathan et al. [18], [19] explore dynamic power management of real-time events in controlling shutting down and waking up system devices for energy efficiency. To aggregate the idle time for energy reduction, Shrivastava et al. [17] propose a framework for code transformations.

Leakage-aware scheduling has also been recently explored on DVS platforms, such as [4], [5], [10]–[13]. In particular, researches in [5], [12], [13] propose energy-efficient scheduling on a processor by procrastination scheduling to control when to turn off the processor. Jejurikar and Gupta [11] then further consider real-time events that might complete earlier than their worst-case estimation. Fixed-priority scheduling is also considered by Jejurikar and Gupta [10] and Chen and Kuo [4]. For uniprocessor scheduling of aperiodic real-time events, Irani et al. [9] propose a 3-approximation algorithm for the minimization of energy consumption. Niu and Quan [16] apply similar procrastination strategies for periodic real-time events with leakage considerations. The basic idea behind the above results is to execute at some speed (mostly at the critical speed) and control the procrastination of the real-time events as long as possible so that the idle interval is long enough to reduce the energy consumption.

Most of the above approaches require either precise information of event arrivals, such as systems with periodic real-time events [4], [5], [10]–[13] or aperiodic real-time events with known arrival time [1], [3], [9]. However, in practical, the precise information of event arrival time might not be known in advance since the arrival time depends on many

The work was partially supported by European Integrated Project SHAPES (grant no. 26825) under IST FET – Advanced Computing Architecture (ACA) and the European Community's Seventh Framework Programme FP7/2007-2013 project Predator (grant no. 216008).

factors. When the precise information of event arrivals is unknown, to our best knowledge, the only known approach is to apply the on-line algorithms proposed by Irani et al. [9] and Augustine et al. [1] to control when to turn on the system. However, since the on-line algorithms in [1], [9] greedily stay in the sleep mode as long as possible without referring to incoming events in the near future, the resulting schedule might make an event miss its deadline.

To model such irregular events, Real-Time Calculus, extended from Network Calculus [7], was proposed by Thiele et al. [20] to characterize events with arrival curves by evaluating how often system functions will be called, how much data is provided as input to the system, and how much data is generated by the system back to its environment. Specifically, the arrival curve of an event stream describes the upper and lower bounds of the number of events arriving to the system for a specified interval. Therefore, schedulability analysis can be done based on the arrival curves of event streams. For scheduling event streams based on Real-Time Calculus in DVS systems under buffer constraints, Maxiaguine et al. [14] develop adaptive algorithms to control the execution speed dynamically at periodic intervals of predefined length without exploiting the possibility for finding the optimal interval length with respect to the power consumption.

This paper explores how to apply DPM to reduce the energy consumption while satisfying the real-time or quality of service (QoS) constraints. We consider systems (or devices) with active, standby, and sleep modes with different power consumptions. Similar to the approaches in [4], [10]–[12], for systems with DVS capability, we assume that the execution of events is at the critical speed and explore the energy reduction by applying DPM for reducing the energy consumption for idling. Distinct from the on-line adaptive algorithms in [14], to reduce the run-time overhead for determining when to perform mode changes, we propose off-line algorithms to derived optimal or approximated solutions to periodic power management for controlling when to change the system mode periodically. The light run-time overhead of the periodic power management schemes is very suitable for devices that only have limited power on computation. For scheduling one event stream under the real-time or QoS constraints, we develop an approach to derive optimal solutions and another to derive approximated solutions with lower complexity. Then, by applying the Modular Performance Analysis [24], we extend the developed approaches to cope with multiple event streams. To demonstrate the performance of the proposed approaches, several case studies are explored, in which the results reveal the effectiveness of our approaches.

The rest of this paper is organized as follows: Section II provides system models. Section III presents our approaches for one event stream, while Section IV copes with multiple event streams. Simulations results are presented in Section V. Section VI concludes this paper.

## II. System Models and Problem Definition

*a) Hardware Model:* We consider a system (or a device) that has three power consumption modes, including

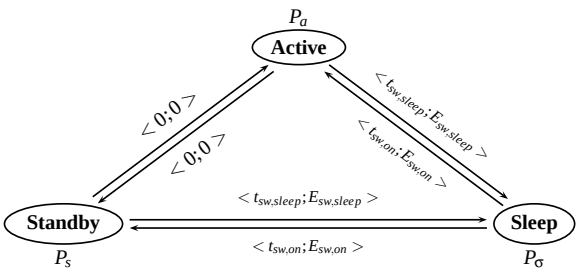


Fig. 1. Example for state transit, where the tuple one each transit is the timing overhead and energy overhead.

*active*, *standby*, and *sleep* modes. The power consumption in the sleep mode is $P_\sigma$. To serve an event, the system must be in the active mode with power consumption $P_a$, in which $P_a > P_\sigma$. Once there is no event to serve, the system can enter the sleep mode. However, switching from the sleep mode to the active mode takes time, denoted by $t_{sw,on}$, and requires additional energy overhead, denoted by $E_{sw,on}$. To prevent the system from frequent mode switches, the system can also stay in the standby mode. The power consumption $P_s$ in the standby mode, by definition, is less than $P_a$ and is more than $P_\sigma$. In this paper, we assume that switching between the standby mode and the active mode has negligible overhead, compared to the other switches, which is the same as the assumption in [25], [28]. Moreover, switching from the active (also standby) mode to the sleep mode takes time, denoted by $t_{sw,sleep}$, and requires additional energy overhead, denoted by $E_{sw,sleep}$. Fig. 1 illustrates the state diagram of these three modes. For simplicity, once the system issues a mode switch from one mode to another mode, we assume that the power consumption of the system before the system enters the new mode is $P_\sigma$.

*b) Event Model:* This paper focuses on events that arrive at the system inputs irregularly. To model such events, we adopt the arrival curves based on Real-Time Calculus [20]. Specifically, a trace of an event stream can conveniently be described by means of a cumulative function $R(t)$, defined as the number of events seen on the event stream in the time interval $[0,t)$. While any $R$ always describes one concrete trace of an event stream, a tuple $\bar{\alpha}(\Delta) = [\bar{\alpha}^u(\Delta), \bar{\alpha}^l(\Delta)]$ of upper and lower arrival curves provides an abstract event stream model, providing bounds on admissible traces of an event stream. The upper arrival curve $\bar{\alpha}^u(\Delta)$ provides an upper bound on the number of events that are seen in any time interval of length $\Delta$, while the lower arrival curve $\bar{\alpha}^l(\Delta)$ analogously provides a lower bound. Please refer to [20] for detailed discussion.

The concept of arrival curves unifies many other common timing models of event streams. For example, a periodic event stream can be modeled by a set of step functions where $\bar{\alpha}^u(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor + 1$ and $\bar{\alpha}^l(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor$. For a sporadic event stream with minimal inter arrival distance $p$ and maximal inter arrival distance $p'$, the upper and lower arrival curve is $\bar{\alpha}^u(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor + 1$, $\bar{\alpha}^l(\Delta) = \left\lfloor \frac{\Delta}{p'} \right\rfloor$, respectively. Moreover, for an event stream with period $p$, jitter $j$, and minimal inter arrival distance $d$, the upper arrival curve is $\bar{\alpha}^u(\Delta) =$

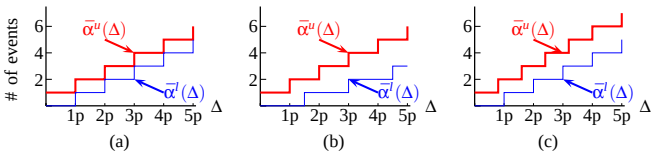


Fig. 2. Examples for arrival curves, where (a) periodic events with period $p$, (b) events with minimal inter-arrival distance $p$ and maximal inter-arrival distance $p' = 1.5p$, and (c) events with period $p$, jitter $j = p$, and minimal inter-arrival distance $d = 0.8p$.

$\min\{\lceil \frac{\Delta+j}{p} \rceil, \lceil \frac{\Delta}{d} \rceil\}$. Fig. 2 illustrates arrival curves for the above cases. For details, please refer to [20].

Analogously to the cumulative function $R(t)$, the concrete availability of the system can be described by a cumulative function $C(t)$, that is defined as the number of available resources, e.g., processor cycles or bus capacity, in the time interval $[0,t)$. Analogous to arrival curves that provide an abstract event stream model, a tuple $\beta(\Delta) = [\beta^u(\Delta), \beta^l(\Delta)]$ of upper and lower service curves then provides an abstract resource model. The upper and lower service curve provides an upper and lower bound on the available resources in any time interval of length $\Delta$.

For an event stream $S_i$, the upper arrival curve $\bar{\alpha}_i^u(\Delta)$ and the lower arrival curve $\bar{\alpha}_i^l(\Delta)$ can be obtained by applying Real-Time Calculus. Note that an arrival curve $\bar{\alpha}_i(\Delta)$ specifies the (upper-bounded or lower-bounded) number of events of event stream $S_i$ for every time interval $\Delta$ while a service curve $\beta(\Delta)$ specifies the (upper-bounded or lower-bounded) available amount of time for execution for every time interval $\Delta$. Therefore, the arrival curve $\bar{\alpha}_i^u(\Delta)$ (respectively, $\bar{\alpha}_i^l(\Delta)$) has to be transformed to the arrival curve $\alpha_i^u(\Delta)$ (respectively, $\alpha_i^l(\Delta)$) to indicate the amount of computation time required for the arrived events in intervals. If the execution time $w_i$ associated to an event in stream $S_i$ is bounded by $c_i^l \le w_i \le c_i^u$ then the transformation can be done by $\alpha_i^u = c_i^u \bar{\alpha}_i^u$, $\alpha_i^l = c_i^l \bar{\alpha}_i^l$ and back by $\bar{\alpha}_i^u = \lceil \alpha^u / c_i^l \rceil$, $\bar{\alpha}_i^l = \lfloor \alpha^l / c_i^u \rfloor$. In the case of variable workloads, workload curves [15] can be applied. Moreover, to satisfy the real-time or QoS constraint of event stream $S_i$, the response time of an event in event stream $S_i$ must be no more than its specified relative deadline $D_i$, where the response time of an event is its finishing time minus the arrival time of the event.

*c) Scheduling Policies:* For scheduling event streams, we consider the fixed-priority (FP) scheduling and the earliest-deadline-first (EDF) scheduling. For FP scheduling, event streams are prioritized a priori. Once an event of an event stream arrives to the system, the priority of the event is set to the pre-defined priority of the event stream. For EDF scheduling, the highest priority is given to the event with the earliest deadline. For both FP and EDF scheduling, the system executes the incomplete event with the highest priority. If there are more than one event with the highest priority, we break ties by applying the first-come-first-serve (FCFS) strategy. Therefore, events in the same event stream will be executed in the FCFS manner.

*d) Periodic Power Management and Problem Definition:* This paper explores how to efficiently and effectively minimize the energy consumption to serve a set of event streams $\mathcal{S}$ under the real-time or QoS requirements. Of course, one could determine when to transit between modes to reduce the energy consumption dynamically, but the computational overhead is quite significant since the decision must be done by evaluating the arrival curves dynamically.

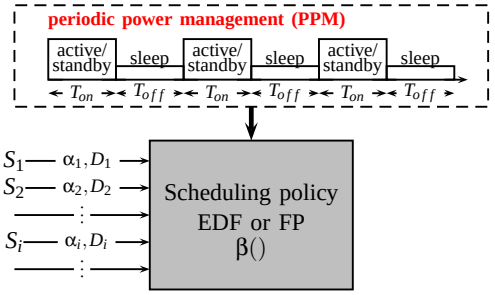


Fig. 3. The abstract model of the periodic power management problem.

In this paper, we propose *periodic power management schemes* (PPM), abstractly illustrated in Fig. 3, in which the power management is done by analyzing the arrival curves of event streams $\mathcal{S}$ statically. Specifically, the periodic power management schemes first decide the period $T = T_{on} + T_{off}$ for power management, then switch the system to the standby mode for $T_{on}$ time units, following by $T_{off}$ time units in the sleep mode. Therefore, given a time interval $L$, where $L \gg T$ and $\frac{L}{T}$ is an integer, suppose that $\gamma_i(L)$ is the number of events of event stream $S_i$ served in interval $L$. If all the served events finish in time interval $L$, the energy consumption $E(L, T_{on}, T_{off})$ by applying the PPM is

$$\begin{aligned} E(L,T_{on},T_{off}) &= \frac{L}{T_{on}+T_{off}}(E_{sw,on}+E_{sw,sleep}) \\ &\quad + \frac{L\cdot T_{on}}{T_{on}+T_{off}}P_s + \frac{L\cdot T_{off}}{T_{on}+T_{off}}P_\sigma \\ &\quad + \sum_{S_i\in\mathcal{S}} c_i\cdot\gamma_i(L)(P_a-P_s) \\ &= \frac{L\cdot E_{sw}}{T_{on}+T_{off}} + \frac{L\cdot T_{on}(P_s-P_\sigma)}{T_{on}+T_{off}} \\ &\quad + L\cdot P_\sigma + \sum_{S_i\in\mathcal{S}} c_i\cdot\gamma_i(L)(P_a-P_s) \end{aligned}$$

where $E_{sw}$ is $E_{sw,on} + E_{sw,sleep}$ for brevity.

As a result, for an $L$ that is sufficiently large, without changing the scheduling policy, the minimization of energy consumption $E(L, T_{on}, T_{off})$ is to find $T_{on}$ and $T_{off}$ such that the *average idle power consumption* $P(T_{on}, T_{off})$

$$P(T_{on},T_{off}) = \frac{E_{sw} + T_{on}(P_s - P_\sigma)}{T_{on}+T_{off}} \quad (1)$$

is minimized. We now define the PPM problem studied in this paper as follows:

> *Given a set of event streams $\mathcal{S}$ under the real-time or QoS requirements, the objective of the studied problem is to find a periodic power management characterized by $T_{on}$ and $T_{off}$ that minimizes the average standby power consumption, in which the response time of any event of event stream $S_i$ in $\mathcal{S}$ must be no more than $D_i$.* ■

## III. One Event Stream

As we are interested in one event stream in this section, suppose that $S_1$ is the given event stream. For event streams described by Real-Time Calculus, one can apply Real-Time Interface [21] to verify whether a system can provide guarantee output service $\beta^G(\Delta)$. Correspondingly, to guarantee that all events in an event stream can be processed while respecting all timing constraints, the event stream demands a service bound $\beta^A(\Delta)$. To satisfy the required deadline $D_1$, $\beta^A(\Delta)$ can be computed as $\beta^A(\Delta) = \alpha_1^u(\Delta - D_1)$. To check the schedulability of event stream $S_1$ in the system, the following predicate has to be true:

$$\beta^G(\Delta) \geq \beta^A(\Delta), \qquad \forall \Delta \geq 0 \tag{2}$$

For PPM with specified $T_{on}$ and $T_{off}$, the guarantee service of the system can be refined as:

$$\beta^G(\Delta) = \max\Big(\Big\lfloor \frac{\Delta}{T_{on}+T_{off}} \Big\rfloor \cdot T_{on}, \Delta - \Big\lceil \frac{\Delta}{T_{on}+T_{off}} \Big\rceil \cdot T_{off}\Big) \tag{3}$$

For the rest of this section, we are going to present our schemes to find the pair $(T_{on}, T_{off}) \in \mathbb{R}^+ \times \mathbb{R}^+$ to minimize the average idle power consumption such that the service constraint $\beta^G(\Delta)$ in (2) is satisfied, and, hence, all events in stream $S_1$ have response time shorter than the timing constraint $D_1$.

Reviewing the formulation of the average idle power consumption $P(T_{on}, T_{off}) = \frac{E_{sw} + T_{on}(P_s - P_\sigma)}{T_{on}+T_{off}}$, there are two cases. (1) If $\frac{E_{sw}}{P_s - P_\delta} \geq T_{off}$, we know that $P(T_{on}, T_{off})$ is minimized when $T_{on}$ is set to $\infty$. (2) If $\frac{E_{sw}}{P_s - P_\delta} < T_{off}$, the minimal $T_{on}$ under the service constraint $\beta^G(\Delta)$ minimizes the average idle power consumption $P(T_{on}, T_{off})$. In this sense, $\frac{E_{sw}}{P_s - P_\delta}$, can be seen as the break-even time of the system. Our approaches proposed in this paper are based on (1) the finding of the minimal $T_{on}$ under the service constraint $\beta^G(\Delta)$, provided that $T_{off}$ is given, and (2) the exploration of the best $T_{off}$. One could also derive solutions in another direction by searching the best $T_{off}$ for a specified $T_{on}$ along with the exploration on $T_{on}$, but the procedure would be more complicated.

### A. Finding the Minimal $T_{on}$ Optimally and Approximately

By the service guarantee curve $\beta^G$ in (3), the service demand curve $\beta^A = \alpha_1^u(\Delta - D_1)$, and the schedulability definition in (2), the minimal $T_{on}$ to fulfil the schedulability requirement in terms of a given $T_{off}$ can be defined as:

$$T_{on}^{\min} = \min\{T_{on} : \beta^G(\Delta) \geq \beta^A(\Delta), \forall \Delta \geq 0\}. \tag{4}$$

To our best knowledge, there is no explicit form to compute $T_{on}^{min}$. Furthermore, due to the complex shape of the arrival curves, exhaustive testing of (2) is the only way to determine the minimum from all possible $T_{on}$.

Instead of calculating the exact $T_{on}^{\min}$, we propose an alternative approach, namely bounded-delay approximation, to find an approximated minimal $\tilde{T}_{on}$. The bounded-delay approach, on one hand, can reduce the computational complexity of finding the minimal $\tilde{T}_{on}$, and, on the other hand, can provide means to solve the PPM problem efficiently.

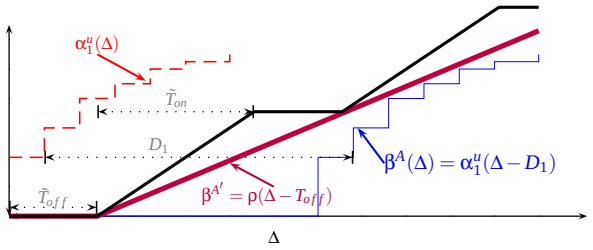


Fig. 4. An example for the bounded delay approximation, in which only part of the upper arrival curve $\alpha_1^u(\Delta)$ is presented for simplicity.

The basic idea of the proposed approach is to compute a minimal *bounded-delay function* $\beta^{A'}(\Delta)$, then derive the minimal $T_{on}$ based on $\beta^{A'}$. A bounded-delay function $\mathbf{bdf}(\Delta, \rho, T_{off})$, defined by the slope $\rho$ and the bounded-delay $T_{off}$ for interval length $\Delta$, is $\max\{0, \rho \cdot (\Delta - T_{off})\}$.

For a given bounded-delay function with slope $\rho$ and bounded-delay $T_{off}$, we can construct a PPM with

$$\tilde{T}_{on} = \frac{\rho \cdot T_{off}}{1 - \rho} \tag{5}$$

such that the resulting service curve of the PPM is no less than the minimal $\mathbf{bdf}(\Delta, \rho, T_{off})$ for any $\Delta \geq 0$. Fig. 4 illustrates an example to derive $\tilde{T}_{on}$. From above definitions, we can have the following lemma.

*Lemma 1:* For specified $T_{off} > 0$ and $0 < \rho \leq 1$:
(1) If $\mathbf{bdf}(\Delta, \rho, T_{off}) \geq \alpha_1^u(\Delta - D_1)$, then, for any $\rho' > \rho$, $\mathbf{bdf}(\Delta, \rho', T_{off}) \geq \alpha_1^u(\Delta - D_1)$.
(2) If $\mathbf{bdf}(\Delta, \rho, T_{off}) < \alpha_1^u(\Delta - D_1)$, then, for any $\rho' < \rho$, $\mathbf{bdf}(\Delta, \rho', T_{off}) < \alpha_1^u(\Delta - D_1)$.

*Proof:* This is simply based on the construction of the bounded delay function. ■

By (5) and Lemma 1, finding the minimum $\rho$, namely $\rho_{\min, T_{off}}$, under the constraint of the service demand $\beta^A$, is equivalent to the derivation of the minimal $\tilde{T}_{on}$ in the bounded-delay approximation, where

$$\rho_{\min, T_{off}} = \inf\{\rho : \mathbf{bdf}(\Delta, \rho, T_{off}) \geq \alpha_1^u(\Delta - D_1), \forall \Delta \geq 0\}.$$

Now we can formally define $\tilde{T}_{on}$ as following.

*Definition 1:* The minimal $T_{on}$ acquired from the bounded-delay approximation is a function of $T_{off}$:

$$\tilde{T}_{on} = \frac{T_{off} \cdot \rho_{\min, T_{off}}}{1 - \rho_{\min, T_{off}}} \stackrel{\text{def}}{=} f(T_{off}) \tag{6}$$

To compute $\rho_{\min, T_{off}}$, based on Lemma 1, we can simply apply binary search of $\rho$ in the range of $[0, 1]$. Suppose that there are $n$ possible values of $\rho$, the number of exploration required to derive $\rho_{\min, T_{off}}$ is $O(\log n)$. Compared to the search of optimal $T_{on}^{\min}$ with respect to a given $T_{off}$ in (4) with $O(n)$ explorations of possible combinations, the binary search greatly improves the running time, since verifying whether $\beta^G(\Delta) \geq \beta^A(\Delta)$ for all $\Delta \geq 0$ is time-consuming.

Moreover, the derived $\tilde{T}_{on}$ has certain nice property in the following lemma, which will be used to improve the time complexity for searching the optimal PPM.

*Lemma 2:* Given a $\beta^A$, the function $f(T_{off})$ defined in (6) is strictly increasing and $\frac{T_{off}}{f(T_{off})} > \frac{(1+\varepsilon)T_{off}}{f((1+\varepsilon)T_{off})}$ for any $\varepsilon > 0$.

*Proof:* From the definition of $f$, one has $\rho_{\min,T_{off}} < \rho_{\min,(1+\varepsilon)T_{off}}$ and thereby $f(T_{off}) < f((1+\varepsilon)T_{off})$, which proves the property for strict increase. Because $\rho_{\min,T_{off}} < \rho_{min,(1+\varepsilon)T_{off}}$, we can derive $\frac{1}{1+\frac{T_{off}}{f(T_{off})}} < \frac{1}{1+\frac{(1+\varepsilon)T_{off}}{f((1+\varepsilon)T_{off})}}$, then, $\frac{T_{off}}{f(T_{off})} > \frac{(1+\varepsilon)T_{off}}{f((1+\varepsilon)T_{off})}$. ■

### B. Feasible Region of $T_{off}$

Before presenting how to search the optimal $T_{off}$, we will first discuss about the feasible region of $T_{off}$. Intuitively, if $T_{off}$ is smaller than the break-even time, i.e., $\frac{E_{sw}}{P_s - P_\delta}$, turning the system to the sleep mode consumes more energy than the energy overhead $E_{sw}$ for mode switching. The sleep mode thereby introduces additional energy consumption. Therefore, for searching the optimal $T_{off}$, the region $\left[0, \frac{E_{sw}}{P_s - P_\delta}\right]$ can be safely discarded. Moreover, as $T_{off}$ must also satisfy the timing overhead for mode switches, we also know that $T_{off}$ must be no less than $t_{sw}$, where $t_{sw} = t_{sw,sleep} + t_{sw,on}$.

There is also an upper bound for $T_{off}$. On one hand, $T_{off}$ should be smaller than $D_1 - c_1$. Otherwise, no event can be finished before its deadline. On the other hand, as the system provides no service when it is off, that imposes a maximum service $\beta^G_\top(\Delta) = \max\{0, \Delta - T_{off}\}$. According to Real-Time Interface in (2), we know that predicate

$$\beta^G_\top(\Delta) = \max\{0, \Delta - T_{off}\} \geq \beta^A_1(\Delta) = \alpha_1(\Delta - D_1) \qquad (7)$$

has to be true to satisfy the timing constraint. By inverting (7), we can compute the maximum $T_{off}$ as

$$T^{max}_{off} = \max\left\{T_{off} : \beta^G_\top(\Delta) \geq \beta^A_1(\Delta), \forall \Delta \geq 0\right\}. \qquad (8)$$

In summary, to find an optimal PPM, the feasible region of $T_{off} \in [T^l_{off}, T^r_{off}]$ can be bounded as follows:

$$T^l_{off} = \max\left\{t_{sw}, \frac{E_{sw}}{P_S - P_\delta}\right\} \qquad (9)$$

$$T^r_{off} = \min\left\{D_1 - c_1, T^{max}_{off}\right\} \qquad (10)$$

### C. Optimal and Approximated PPMs

We now present how to apply the results in Sections III-A and III-B for deriving $T_{on}$ and $T_{off}$ to minimize the average idle power consumption $P(T_{on}, T_{off})$. Depending on the bounded-delay approximation and the derivation of minimum $T_{on}$ with respect to a given $T_{off}$, we will present two approaches, namely BDA and OPT, to derive $T_{on}$ and $T_{off}$. To show the difference between these two schemes, for a given $T_{off}$, Fig. 5 presents an example for illustrating the irregular pattern of the minimum average idle power consumption by solving (4) and the convexity of the average idle power consumption by applying bounded-delay approximation.

**Approach OPT** As shown in Fig. 5, the minimal average idle power consumption in the feasible region of $T_{off}$ is irregular. Therefore, to find the optimal solution, we have to explore all the feasible solution space of $T_{off}$. Suppose that there are $m$ values of $T_{off}$ within the region $[T^l_{off}, T^r_{off}]$, the complexity of the overall algorithm thereby is $O(n \cdot m)$. The pseudo-code of the OPT scheme is shown in Algorithm 1.

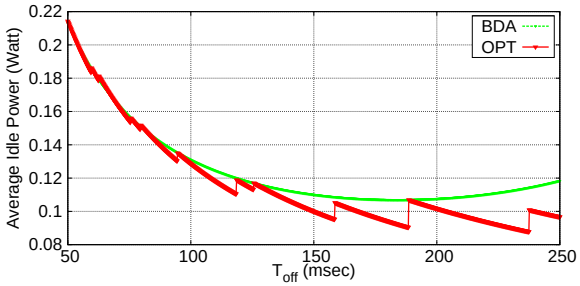


Fig. 5. The relation of the minimal average idle power consumption and $T_{off}$ for the OPT and BDA approaches. The stream and device are $S_1$ and IBM Microdrive in Table I and II of Section V, respectively.

**Algorithm 1** OPT

**Input:** $\alpha_1$, $D_1$, $T^l_{off}$, $T^r_{off}$, $\varepsilon$, $P_{min} = \infty$
**Output:** $T'_{on}$, $T'_{off}$

```
for T_off = T^l_off to T^r_off step ε do
    exhaustively find T^min_on by testing (4)
    if P(T^min_on, T_off) < P_min then
        T'_on ← T^min_on; T'_off ← T_off
        P_min ← P(T^min_on, T_off)
    end if
end for
```

**Approach BDA** Before presenting the BDA algorithm, we first show the convexity of the objective function $P(T_{on}, T_{off})$ acquired from the application of bounded-delay approximated $\tilde{T}_{on}$ defined in Def. 1.

*Theorem 1:* Using the bounded-delay algorithm approach to compute $\tilde{T}_{on} = f(T_{off})$ as depicted in (6), $P(\tilde{T}_{on}, T_{off}) = P(f(T_{off}), T_{off})$ is a convex function.

*Proof:* The objective function $P(\tilde{T}_{on}, T_{off})$ can be split into two parts: $\frac{E_{sw}}{T_{on}+T_{off}}$ and $(P_s + P_\delta) \cdot \frac{T_{on}}{T_{on}+T_{off}}$. For the first part $\frac{E_{sw}}{T_{on}+T_{off}} = \frac{E_{sw}}{f(T_{off})+T_{off}}$, $f(T_{off}) + T_{off}$ is strictly increasing according to Lemma 2. Therefore $\frac{E_{sw}}{T_{on}+T_{off}}$ is a monotonically decreasing convex function. For the second part $(P_s + P_\delta) \cdot \frac{T_{on}}{T_{on}+T_{off}} = \frac{P_s+P_\delta}{1+\frac{T_{off}}{f(T_{off})}}$, according to Lemma 2, we know that $\frac{1}{1+\frac{T_{off}}{f(T_{off})}}$ is monotonically increasing and is a convex function as well. As a linear combination of convex functions is a still a convex function, the original function $P(\tilde{T}_{on}, T_{off})$ is a convex function of $T_{off}$. ■

Based on the result from Thm. 1, exhaustive search for every $T_{off}$ is not necessary. For instance, the complexity is reduced to $O(\log n \cdot \log m)$ by applying a bisection search for the feasible region of $T_{off}$. The pseudo code of the algorithm is described in the Algorithm 2.

**Algorithm 2** BDA

**Input:** $\alpha_1$, $D_1$, $T^l_{off}$, $T^r_{off}$, $\varepsilon$
**Output:** $T'_{on}$, $T'_{off}$

**if** $T^r_{off} - T^l_{off} < \varepsilon$ **then**
  **if** $P\big(T^l_{off}, f(T^l_{off})\big) < P\big(T^r_{off}, f(T^r_{off})\big)$ **then**
    return $\{T'_{on} \leftarrow f(T^l_{off}); T'_{off} \leftarrow T^l_{off}\}$
  **else**
    return $\{T'_{on} \leftarrow f(T^r_{off}); T'_{off} \leftarrow T^r_{off}\}$
  **end if**
**end if**
$\rho_l \leftarrow P'(T^l_{off}, f(T^l_{off}))$ ▷ $P'$ is the derivative of $P$ with respective to $T_{off}$
$\rho_m \leftarrow P'\Big(\frac{T^l_{off}+T^r_{off}}{2}, f(\frac{T^l_{off}+T^r_{off}}{2})\Big)$
**if** $\rho_l \cdot \rho_m > 0$ **then**
  $T^l_{off} \leftarrow T^m_{off}$
**else**
  $T^r_{off} \leftarrow T^m_{off}$
**end if**
recursively call BDA with the new $T^l_{off}$ and $T^r_{off}$

## IV. Multiple Event Streams

This section presents how to cope with multiple event streams by extending the bounded delay approximation (BDA) scheme and the optimal periodic power management (OPT) scheme presented in Section III. Due to space limitation, we will focus our discussions on fixed priority (FP) scheduling, and at the end of this section, we will briefly show how to handle earliest deadline first (EDF) scheduling.

Suppose that there are $N$ event streams in $\mathcal{S}$, where $N \geq 2$. For FP scheduling, without loss of generality, we order the event streams $S_1, S_2, \ldots, S_N$ according to their priorities, where the priority of event stream $S_i$ is higher than that of $S_k$ when $k > i$. Suppose that $\beta^l_1(\Delta)$ is the lower service curve of the system. By Real-Time Calculus [20], we know that the remaining lower service curve $\beta'_1(\Delta)$ after serving event stream $S_1$ is $\sup_{0\leq\lambda\leq\Delta}\{\beta^l_1(\lambda) - \alpha^u(\lambda)\}$. In FP scheduling, the remaining service will be used to serve the other event streams, in which $\beta'_1(\Delta)$ is the available service curve of event stream $S_2$. For example, as illustrated in Fig. 6 for three event streams, for FP scheduling, the schedulability analysis can be decomposed as three components by using the remaining service curve left by higher priority streams.

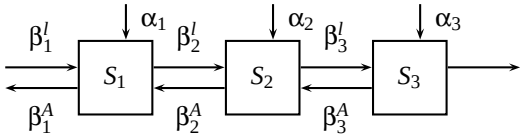

Fig. 6. An example for fixed-priority scheduling

Therefore, to guarantee the satisfaction of timing constraint for event stream $S_N$, as shown in Section III, the service provided to stream $S_N$ must be at least $\beta^A_N(\Delta) = \alpha^u_N(\Delta - D_N)$. This also implies that the remaining service curve after serving streams $S_1, S_2, \ldots, S_{N-1}$ must be at least $\beta^A_N(\Delta)$. To derive the service bound $\beta^A_1(\Delta)$, we have to compute the service bounds $\beta^A_{N-1}(\Delta), \beta^A_{N-2}(\Delta), \ldots, \beta^A_2(\Delta)$, sequentially. Suppose that $\beta^A_k(\Delta)$ has been derived, we can apply the following equation to derive $\beta^\sharp_{k-1}(\Delta)$ so that the remaining service curve is guaranteed to be no less than $\beta^A_k(\Delta)$ if $\beta^l_{k-1}(\Delta)$ is no less than $\beta^\sharp_{k-1}(\Delta)$:

$$\beta^\sharp_{k-1}(\Delta) = \beta^A_k(\Delta-\lambda) + \alpha^u_{k-1}(\Delta-\lambda) \quad (11)$$
$$\text{where } \lambda = \sup\{\tau : \beta^A_k(\Delta-\tau) = \beta^A_k(\Delta)\}.$$

To guarantee the timing constraint of event stream $S_{k-1}$, we also know that $\beta^l_{k-1}(\Delta)$ must be no less than $\alpha^u_{k-1}(\Delta - D_{k-1})$. Therefore, we know that

$$\beta^A_{k-1}(\Delta) = \max\{\beta^\sharp_{k-1}(\Delta), \alpha^u_{k-1}(\Delta - D_{k-1})\}. \quad (12)$$

By applying (12) for $k = N-1, N-2, \ldots, 2$, we can then derive the lower service curve, i.e., $\beta^A_1(\Delta)$, that the system must provide for satisfying the timing constraints. Then, the bounded delay approximation (BDA) scheme and the optimal periodic power management (OPT) scheme can be applied to minimize the average idle power consumption $P(T_{on}, T_{off})$ by setting $\beta^A(\Delta)$ to $\beta^A_1(\Delta)$ derived above.

For EDF scheduling, as shown in [22], the service bound $\beta^A(\Delta)$ is simply $\sum_{S_i\in\mathcal{S}} \alpha^u_i(\Delta - D_i)$. For FCFS, the service bound $\beta^A(\Delta)$ is $\sum_{S_i\in\mathcal{S}} \alpha^u_i(\Delta - D_{\min})$, where $D_{\min}$ is the minimum relative deadline of the event streams in $\mathcal{S}$.

TABLE I
Event stream setting according to [23].

| | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ | $S_8$ | $S_9$ | $S_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| p (msec) | 198 | 102 | 283 | 354 | 239 | 194 | 148 | 114 | 313 | 119 |
| j (msec) | 387 | 70 | 269 | 387 | 222 | 260 | 91 | 13 | 302 | 187 |
| d (msec) | 48 | 45 | 58 | 17 | 65 | 32 | 78 | - | 86 | 89 |
| c (msec) | 12 | 7 | 7 | 11 | 8 | 5 | 13 | 14 | 5 | 6 |

TABLE II
Power profiles for devices according to [6].

| Device Name | $P_a$ (Watt) | $P_s$ (Watt) | $P_\sigma$ (Watt) | $t_{sw}$ (sec) | $E_{sw}$ (mJoule) |
|---|---|---|---|---|---|
| Realtek Ethernet | 0.19 | 0.125 | 0.085 | 0.01 | 0.8 |
| Maxstream | 0.75 | 0.1 | 0.05 | 0.04 | 7.6 |
| IBM Microdrive | 1.3 | 0.5 | 0.1 | 0.012 | 9.6 |
| SST Flash | 0.125 | 0.05 | 0.001 | 0.001 | 0.098 |

## V. Performance Evaluations

This section provides simulation results for the PPM schemes derived from the proposed OPT and BDA approaches. All results are obtained from a simulation host with Intel 1.7GHz processor and 1 GB RAM.

**Simulation Setup** We take the stream set studied in [8], [23] for our case study. Table I specifies the parameters for this 10-stream set. A stream $S_i$ is specified by its period $p_i$, jitter $j_i$, minimal inter-arrival distance $d_i$, and worst case execution time $c_i$, as defined in Section II. The relative deadline $D_i$ of $S_i$ is defined as $D_i = \chi * p_i$ and varies according to the *deadline factor* $\chi$. In our simulations, we adopt the power profiles for four devices in [6], presented in Table II.

We simulate scenarios for both one stream and multiple streams. Due to space limitation, we only report random subsets of the 10-stream set for multiple streams. For instance,

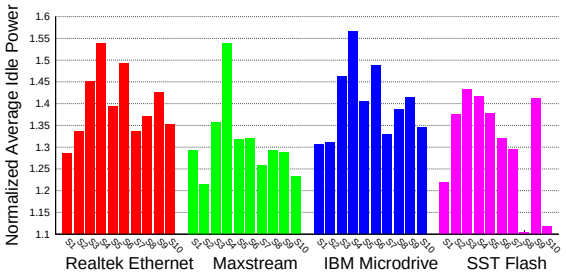


Fig. 7. Normalized average idle power of the PPM schemes dervide by the BDA approach for single stream scenarios with $\chi = 1.6$.

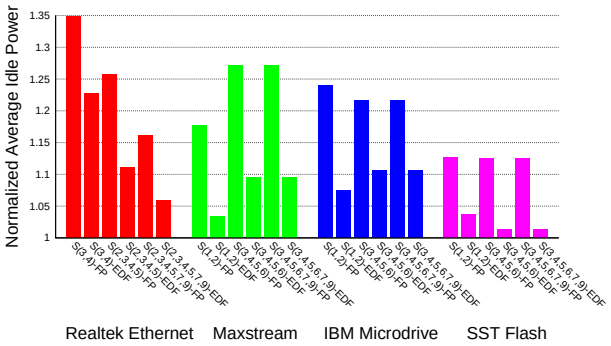


Fig. 8. Normalized average idle power of PPM schemes derived by the BDA approach for multiple-stream scenarios with $\chi = 2$ by applying EDF and FP scheduling.

$S(3,4)$ represents a scenario considering only streams $S_3$ and $S_4$ in Table I. In the case of fixed-priority scheduling for multiple streams, the priority corresponds to the stream index, e.g., $S_3$ has higher priority than $S_4$ for $S(3,4)$.

Since all PPM schemes derived from both OPT and BDA algorithms have the same energy consumption for event processing, we compare the average idle power, defined in (1). We also report the computation time required to derive PPM schemes for both BDA and OPT approaches.

**Simulation Results** First, we show the effectiveness of the BDA approach. Fig. 7 and Fig. 8 show the *normalized average idle power* of the PPM schemes derived by the BDA approach with respect to those by the OPT approach for single and multiple stream scenarios, respectively, subjected to four devices and different scheduling. As shown in these two figures, the PPM scheme derived by BDA approach reasonably approximates the optimal scheme obtained from the OPT approach with respect to different subset of the stream set, different devices, and different scheduling policies. In general, the BDA approach derives better schemes for multiple-stream scenarios than for single-stream scenarios. The reason is that with more streams involved, the result demand curve is smoothen by the individual streams, resulting in a closer match of the bounded delay function. The PPM scheme derived from the bounded delay function approximates the optimum better accordingly.

We also investigate how the average idle power changes as the relative deadline of event streams varies. As our PPM schemes are *time-driven*, we use an event-driven (ED) scheme as a reference, where the device is turned to the

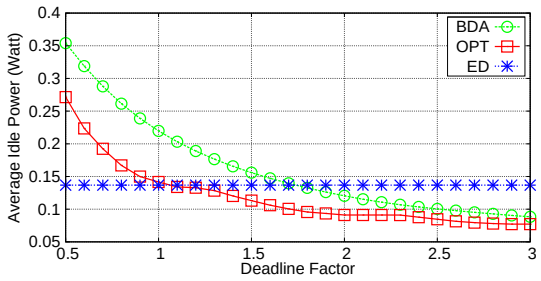


Fig. 9. Average idle power of stream $S_8$ for the IBM Microdrive.

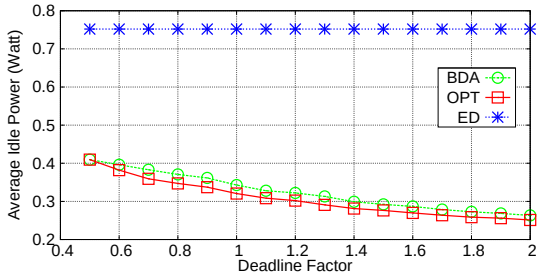


Fig. 10. Average idle power of the 10-stream scenario for the IBM Microdrive with EDF scheduling.

sleep mode when there is no event to be processed, and is awaken for event processing whenever an event arrives. Fig. 9 and Fig. 10 depict results for single and multiple stream scenarios, respectively. As shown in these figures, the BDA approach effectively approximates the optimum. We also observe that the average idle power decreases as the relative deadline increases for both single and multiple stream cases. The reason is that for longer relative deadlines, more arrived events can be accumulated for each activation of the device. Another observation is that the ED scheme is effective for one stream scenario and short relative deadlines. As more streams are involved and the relative deadline increases, our dervied PPM schemes considerably outperform the ED scheme, due to the amount of mode-switch operations reduced.

Second, we demonstrate the efficiency for deriving a minimal PPM scheme for both BDA and OPT approaches by reporting the computation expense. Fig. 11 depicts the computation time for different stream combinations, given a fixed deadline factor. The BDA approach is about two orders of magnitude faster than the OPT approach. Fig. 12 shows the relation of computation time and the deadline factor for the 10-stream scenario. As the figure shown, the computation time for the OPT approach increases as the relative deadline increases, while the time remains in the same scale for the BDA approach. Note that the OPT approach can be much slower for a smaller granularity of $\varepsilon$ in Algorithm. 1. We can conclude the BDA approach efficiently derives approximation solutions.

## VI. Conclusion

This paper explores how to apply dynamic power management to reduce the energy consumption under the real-time or quality of service constraints. Based on off-line analysis,

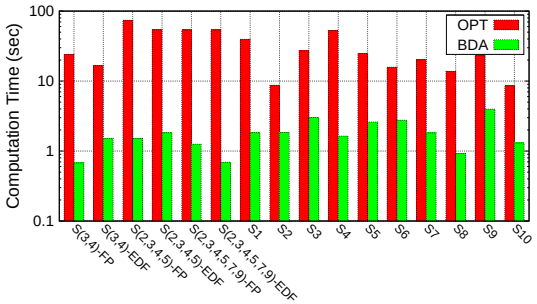


Fig. 11. Computation time of different scenarios for the IBM Microdrive with $\chi = 2$.

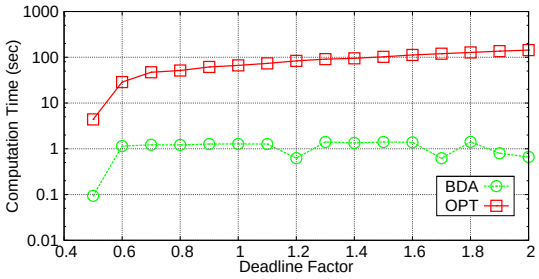


Fig. 12. Computation time of the 10-stream scenario for the IBM MicroDrive with EDF scheduling.

our proposed periodic power management has light run-time overhead and is very suitable for devices that have limited power on computation, such as micro-control units (MCU), that. We consider systems (or devices) with active, standby, and sleep modes with different power consumptions. For scheduling one event stream under the real-time or QoS constraints, we develop an approach to derive optimal solutions and another to derive approximated solutions with lower complexity. Extensions to multiple events streams are also presented by applying the Modular Performance Analysis [24]. To demonstrate the performance of the proposed schemes, several case studies are explored, in which the results reveal the effectiveness of our approaches.

## REFERENCES


[1] J. Augustine, S. Irani, and C. Swamy. Optimal power-down strategies. In *45th Symposium on Foundations of Computer Science (FOCS)*, pages 530–539, Oct. 2004.

[2] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS)*, pages 95–105, 2001.

[3] P. Baptiste. Scheduling unit tasks to minimize the number of idle periods: A polynomial time algorithm for offline dynamic power management. In *Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 364–367, 2006.

[4] J.-J. Chen and T.-W. Kuo. Procrastination for leakage-aware rate-monotonic scheduling on a dynamic voltage scaling processor. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 153–162, 2006.

[5] J.-J. Chen and T.-W. Kuo. Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems. In *International Conference on Computer-Aided Design (ICCAD)*, pages 289–294, 2007.

[6] H. Cheng and S. Goddard. Online energy-aware I/O device scheduling for hard real-time systems. In *Proceedings of the 9th Design, Automation and Test in Europe (DATE)*, pages 1055–1060, 2006.

[7] R. Cruz. A calculus for network delay. I. network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, Jan 1991.

[8] A. Hamann and R. Ernst. Tdma time slot and turn optimization with evolutionary search techniques. In *Proceedings of the 8th Design, Automation and Test in Europe (DATE)*, pages 312–317, 2005.

[9] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 37–46, 2003.

[10] R. Jejurikar and R. K. Gupta. Procrastination scheduling in fixed priority real-time systems. In *Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 57–66, 2004.

[11] R. Jejurikar and R. K. Gupta. Dynamic slack reclamation with procrastination scheduling in real-time embedded systems. In *Proceedings of the 42nd ACM/IEEE Design Automation Conference (DAC)*, pages 111–116, 2005.

[12] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st ACM/IEEE Design Automation Conference (DAC)*, pages 275–280, 2004.

[13] Y.-H. Lee, K. P. Reddy, and C. M. Krishna. Scheduling techniques for reducing leakage power in hard real-time systems. In *15th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 105–112, 2003.

[14] A. Maxiaguine, S. Chakraborty, and L. Thiele. DVS for buffer-constrained architectures with predictable QoS-energy tradeoffs. In *the International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS)*, pages 111–116, 2005.

[15] A. Maxiaguine, S. Künzli, and L. Thiele. Workload characterization model for tasks with variable execution demand. In *Proceedings of the 7th Design, Automation and Test in Europe (DATE)*, 2004.

[16] L. Niu and G. Quan. Reducing both dynamic and leakage energy consumption for hard real-time systems. In *Proceedings of the international conference on Compilers, architecture, and synthesis for embedded systems (CASES)*, pages 140–148, 2004.

[17] A. Shrivastava, E. Earlie, N. Dutt, and A. Nicolau. Aggregating processor free time for energy reduction. In *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (CODES+ISSS)*, pages 154–159, 2005.

[18] V. Swaminathan and C. Chakrabarti. Energy-conscious, deterministic I/O device scheduling in hard real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(7):847–858, 2003.

[19] V. Swaminathan and K. Chakrabarty. Pruning-based, energy-optimal, deterministic I/O device scheduling for hard real-time systems. *ACM Transactions in Embedded Computing Systems*, 4(1):141–167, 2005.

[20] L. Thiele, S. Chakraborty, and M. Naedele. Real-time Calculus for Scheduling Hard Real-time Systems. *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 4:101–104, 2000.

[21] L. Thiele, E. Wandeler, and N. Stoimenov. Real-time interfaces for composing real-time systems. In *International Conference On Embedded Software (EMSOFT)*, pages 34–43, 2006.

[22] E. Wandeler and L. Thiele. Interface-based design of real-time systems with hierarchical scheduling. In *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 243–252, Apr. 2006.

[23] E. Wandeler and L. Thiele. Optimal TDMA time slot and cycle length allocation for hard real-time systems. In *11th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 479–484, 2006.

[24] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis - A case study. *International Journal on Software Tools for Technology Transfer (STTT)*, 8(6):649–667, Oct. 2006.

[25] C.-Y. Yang, J.-J. Chen, C.-M. Hung, and T.-W. Kuo. System-level energy-efficiency for real-time tasks. In *the 10th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC)*, pages 266–273, 2007.

[26] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 374–382, 1995.

[27] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *Proceedings of the 39th ACM/IEEE Design Automation Conference (DAC)*, pages 183–188, 2002.

[28] J. Zhuo and C. Chakrabarti. System-level energy-efficient dynamic task scheduling. In *Proceedings of the 42nd ACM/IEEE Design Automation Conference(DAC)*, pages 628–631, 2005.