On Maximizing Reliability of Real-Time Embedded Applications Under Hard Energy Constraint

Baoxian Zhao, Hakan Aydin, *Member, IEEE*, and Dakai Zhu, *Member, IEEE*

***Abstract*—The dynamic voltage and frequency scaling (DVFS) technique is the basis of numerous state-of-the-art energy management schemes proposed for real-time embedded systems. However, recent research has illustrated the alarmingly negative impact of DVFS on task and system reliability. In this paper, we consider the problem of assigning processing frequencies to a set of real-time tasks in order to maximize the overall reliability, under given time and energy constraints. First, under the frame-based task model, we formulate the problem as a nonlinear optimization problem and show how to obtain the static optimal solution. Then, we propose online (dynamic) algorithms that detect early completions and adjust the task frequencies at runtime, to improve overall reliability. Furthermore, we extend these solutions to the periodic task model, with both static and dynamic solutions. All our solutions ensure that all timing constraints are met while the cumulative energy consumption of tasks does not exceed the given energy budget. Our simulation results indicate that our algorithms perform comparably to a clairvoyant optimal scheduler that knows the exact workload in advance.**

***Index Terms*—Dynamic voltage and frequency scaling, real-time embedded systems, reliability.**

I. Introduction

With THE proliferation of the embedded computing devices, energy management has become critically important. The dynamic voltage and frequency scaling (DVFS) technique [35] has been recognized as the basis of numerous energy management solutions. DVFS exploits the fact that the dynamic power consumption is a strictly convex function of the CPU speed, and attempts to save energy by reducing the supply voltage and frequency at runtime. In real-time embedded systems research where providing *temporal predictability* is of paramount importance, DVFS has been well studied as an effective energy management technique. Existing studies in this area can be divided into two main lines. In *energy-aware* real-time systems research, the problem is to meet the application's timing constraints with minimum energy consumption for various task/processor and scheduling models [5], [27], [30]. On the other hand, for *energy-constrained* real-time embedded systems, energy is set as a *hard* constraint [2], [21], [36]. This research line is motivated by applications where the system has to remain functional during a well-defined mission/operation time, with fixed and nonreplenishable energy budget. Example applications include military, space and disaster recovery applications, as well as emerging portable medical monitoring and life support devices. In such settings, it is imperative to avoid scenarios where the real-time embedded system runs out of energy in the middle of a mission with potentially detrimental consequences [3], [21], [29]. The *Mars Rover* system has been often mentioned as an energy-constrained system that must complete its mission-critical tasks within a well-defined operation time [10], [21], [36]. In energy-constrained real-time systems area, researchers have exploited the problem of maximizing the total utility [36], reward [2], [10], [29], value [3], and throughput [38], without exceeding the available energy budget.

A number of studies analyzed the interplay between energy management and fault tolerance/reliability [18], [24], [28], [34], [39]. Despite the effectiveness of DVFS to reduce energy consumption, recent research [13], [43] has shown that DVFS has a significant and negative effect on the *system reliability*, primarily because of the significantly increased *transient fault rates* at low supply voltage and frequency levels. Hence, there is a growing awareness about the need to apply DVFS only after careful evaluation, especially for mission-critical real-time embedded applications where both high level of reliability and low-energy consumption are important. While some existing studies revisited the *energy-aware* solution frameworks to mitigate the negative impact of DVFS on reliability [32], [40]–[42], to the best of our knowledge, the problem has not been yet addressed in *energy-constrained* settings (with hard energy constraint).

A. Related Work

Simultaneous consideration of energy consumption, timeliness and fault tolerance/reliability dimensions has attracted attention in recent years. In [31], the authors studied the energy minimization problem for mapping frame-based dependent real-time tasks on DVFS-enabled multiprocessor systems and proposed an efficient two-step iterative approach based on genetic algorithms. Notwithstanding its innovative solution, the framework does not consider the reliability dimension. Izosimov *et al.* [18] obtain and solve an optimization problem for mapping a set of tasks with reliability constraints, timing constraints and precedence relations to processors and for determining appropriate fault tolerance policies (re-execution and replication). This study does not explicitly model the effects of DVFS on transient fault rates.

Manuscript received October 01, 2009; revised February 09, 2010; accepted May 15, 2010. Date of publication June 14, 2010; date of current version August 06, 2010. This work was supported in part by the U.S. National Science Foundation under Grant CNS-0720647, Grant CNS-0720651, and Grant CNS-546244 (CAREER Award). Paper no. TII-09-10-0302.

B. Zhao and H. Aydin are with the Department of Computer Science, George Mason University, Fairfax, VA 22030 USA (e-mail: bzhao@gmu.edu; aydin@cs.gmu.edu).

D. Zhu is with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: dzhu@cs.utsa.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2010.2051970

A significant number of existing research work focused on tolerating a fixed number of faults within the context of energy-aware real-time operation [24], [34], [39]. The checkpointing technique was adopted in [24] to tolerate faults at runtime while exploiting the system slack through DVFS to reduce energy consumption. By adopting the checkpointing technique for soft periodic real-time tasks, Zhang *et al.* developed an online fault tolerance algorithm in [39]. Further, they suggested a fixed priority scheme through the Rate-Monotonic algorithm (RMA) to tolerate faults in hard real-time systems. Based on the characterization of feasibility with RMA, in [34], the authors proposed an efficient online scheme to minimize energy consumption by applying DVFS policies and considering the runtime behaviors of tasks and fault occurrences, while still satisfying the timing constraints.

The negative impact of DVFS on transient fault rates and investigation of the resulting reliability/energy consumption tradeoffs gave rise to another line of research. The modeling of system reliability as a function of frequency/voltage assignment and a preliminary analysis of related tradeoff dimensions has been conducted by Zhu *et al.* in [43]. The work in [40] proposed a reliability-aware power management (RA-PM) scheme that dynamically schedules a recovery job at task dispatch time whenever DVFS is applied, preserving the overall (original) reliability. The scheme is further extended to multiple tasks with a common deadline [42] and to periodic real-time tasks [41]. By employing a feedback controller to track the overall miss ratio of tasks in soft real-time systems, Sridharan *et al.* [32] proposed a reliability-aware energy management algorithm to minimize the system energy consumption, while still preserving the overall system reliability. Pop *et al.* propose a novel framework [28] where user-defined reliability goals are first transformed to the objective of tolerating a fixed number of faults through re-execution for individual tasks, by using the reliability model from [43]. The authors developed a constraint-logic-programming (CLP) based solution to minimize energy consumption with these goals, for dependent tasks represented by directed acyclic graphs (DAGs). Finally, Dabiri *et al.* [12] considered the problem of assigning frequency/voltage to tasks for energy minimization subject to reliability and timing constraints.

While existing research explored various aspects of the interplay between DVFS, reliability and energy consumption from *energy-aware* operation point of view, to the best of our knowledge, *maximizing reliability in energy-constrained settings (with hard energy constraint)* has not been studied before. In this research effort, we assume energy-constrained operation settings [2], [10], [21], [29], [36] where the system's energy consumption must not exceed a given bound. By modeling the transient fault rates as a function of task voltage/frequency levels, we consider and optimally solve the problem of determining task level frequency assignments to maximize overall reliability (i.e., the probability of completing the application successfully), without exceeding the given energy budget/allowance or missing the deadlines.

After presenting our system models and assumptions in Section II, we first address and solve the *static* version of the problem under the frame-based task model, where all tasks execute their worst-case workload within the same given deadline (Section III-Af). Then, we extend our framework to dynamic settings and propose three online reclaiming algorithms that detect early completions and adjust the task frequencies at runtime, with the objective of improving the application's reliability by making the best use of excess energy at runtime (Section III-B). Next, we extend these solutions to the periodic task model where tasks may have different periods (Section IV). We show that it is always possible to find an optimal solution with maximum reliability where all the jobs of a given periodic task run at the same frequency throughout the hyperperiod. This important result, in turn, allows us to reuse the static optimal solution we developed for the frame-based systems. Next, we provide dynamic reclaiming algorithms for periodic tasks to further improve the reliability at runtime by using excess energy that arises from early completions to speed up task executions. The experimental evaluation (presented in Sections III-C and IV-C) indicates that our dynamic algorithms (for both frame-based and periodic task models) perform comparably to a clairvoyant optimal scheduler that knows the exact workload in advance. Section V concludes this paper.

II. System Models

A. Power Model

With DVFS, the clock frequency (speed) is reduced alongside with the supply voltage [26] due to the almost linear relationship between the supply voltage and operating frequency [8]. In this paper, we assume that the CPU supply voltage level corresponds to the minimum level necessary to support the target processing frequency. In the context of DVFS, we use the term *frequency change* to stand for the simultaneous adjustment of frequency and voltage.

We adopt the system-level power model proposed in [43] and subsequently used in [40]–[42]. Hence, the system power consumption P is given by

$$P = P_s + \hbar(P_{\text{ind}} + P_d) = P_s + \hbar(P_{\text{ind}} + C_{\text{ef}} f^m) \qquad (1)$$

where P_s is the static power, P_{ind} is the frequency-independent active power, and P_d is the frequency-dependent active power. The static power, which may be removed only by powering off the whole system, includes the power to maintain basic circuits, keep the clock running and the memory in sleep modes [14]. P_{ind} is a constant independent of processing frequencies (i.e., the power consumed by off-chip devices such as main memory and external devices) and can be efficiently removed by putting systems into sleep states [8]. P_d is the power mainly consumed by CPU, in addition to any power that depends on the frequencies [8]. $\hbar$ represents system states and indicates whether active powers are currently consumed in the system. Specifically, when the system is active, $\hbar = 1$; otherwise, the system is in sleep modes or turned off and $\hbar = 0$. The effective switching capacitance C_{ef} and the dynamic power exponent m (which is, in general, no smaller than 2) are system-dependent constants and f is the processing frequency. All frequency values are normalized with respect to the maximum frequency $f_{\max}$ (i.e., $f_{\max} = 1.0$).

Since there exists an excessive time and energy overhead associated with turning on/off the entire system, we assume that the system is always active. As P_s is not manageable, we will ignore the static power and concentrate on the frequency-indepen-

dent active power P_{ind} and frequency-dependent active power P_d in our analysis.

Although DVFS can reduce energy consumption, the application will take more time to complete at low frequencies. As a result, higher total energy may be consumed because of the prolonged device active times (due to the frequency-independent active power P_{ind}). Therefore, considering the system-level power, lower frequencies may not be always best for energy savings and it is shown in [14] that there exists a minimum energy-efficient voltage/frequency pair. In [43], the energy-efficient frequency for our power model is derived as $f_{\text{ee}} = ((P_{\text{ind}})/((m-1)C_{\text{ef}}))^{1/m}$. Hence, it is not energy-efficient to run any task at a frequency below f_{ee}; doing otherwise will result in more energy consumption. Note that if f_{ee} exceeds the maximum available frequency level $f_{\max}$, then the system should not reduce its speed below $f_{\max}$ [4]. In the following discussion, we use the dynamic power exponent m as 3.

In general, there is a time and energy overhead associated with the runtime voltage and frequency changes in DVFS technique. For instance, the time and energy overhead in frequency/voltage transitions is in the range of 10–120 μs and 0.5–2.6 μ Joules respectively, for the state-of-the-art Mobile AMD Athlon and Intel Xscale architectures [1], [25], [37].

B. Fault Model

During the execution of an application, a fault may occur due to various reasons, such as hardware failure, software errors, electromagnetic interference and cosmic ray radiations. Since $transient$ faults occur much more frequently than $permanent$ faults [9], [16], [17], in this paper, we focus on transient faults, and develop feasible DVFS solutions with a given energy budget to maximize overall reliability.

Traditionally, transient faults have been modeled through Poisson distribution with an average arrival rate λ [39]. In line with the existing literature [19], [33], [43], we assume that the transient faults that occur during the execution of different tasks are *independent*. However, considering the effects of voltage and frequency scaling on transient faults [13], [43], the average rate λ will depend on the system processing frequency and supply voltage. Therefore, the fault rate at frequency f (and its corresponding voltage level) can be generally modeled as

$$\lambda(f) = \lambda_0 \cdot g(f) \tag{2}$$

where λ_0 is the average fault rate corresponding to the maximum frequency $f_{\max}$ (and supply voltage $V_{\max}$). That is, $g(f_{\max}) = 1$.

In general, transient fault rates are exponentially related to the circuit's $critical$ charge (the smallest charge required to cause a soft error in a circuit node) [15]. In our analysis and simulations, we focus on the exponential fault rate model proposed in [43]

$$\lambda(f) = \lambda_0 \cdot g(f) = \lambda_0 \cdot 10^{\frac{d(1-f)}{1-f_{\min}}} \tag{3}$$

where the exponent $d(>0)$ is a constant, indicating the sensitivity of fault rates to voltage and frequency scaling. That is, reducing the supply voltage and frequency for energy savings results in exponentially increased fault rates. The maximum average fault rate is assumed to be $\lambda_{\max} = \lambda_0 \cdot 10^d$ which corresponds to the lowest available frequency $f_{\min}$ (and the supply voltage $V_{\min}$). Note that this fault rate model, proposed in [43] and used in [28], [40]–[42] has been also independently verified in [12]. In this work, we do not consider or model the effect of temperature changes on the transient fault rate.

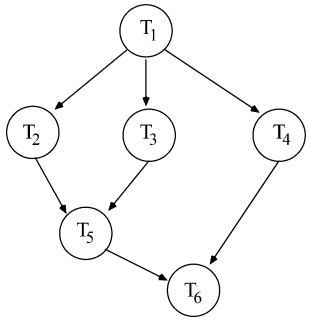


Fig. 1. An example task set with precedence constraints.

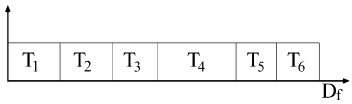


Fig. 2. The execution order within a frame for the given task set.

C. Application Model

In this paper, we first consider and solve the problem of reliability maximization under a given hard energy constraint for frame-based tasks running on a DVFS-enabled uniprocessor system. Specifically, we consider a real-time application represented by a directed acyclic graph (DAG) $G = (V, E)$. The set of vertices (nodes) $V = \{T_1, \ldots, T_n\}$ represents the set of tasks with the common period and deadline D_f. D_f is also called the *frame length*. The set of edges $E = \{E_1, \ldots, E_m\}$ represents a partial order corresponding to the *precedence constraints* [22] among tasks, with the interpretation that whenever the edge $(T_i, T_j) \in E$, the task T_j cannot start to execute until T_i has been completed. We assume that, given a DAG, an execution order of tasks that satisfies the precedence constraints has been determined and that the task indices reflect the execution order in that sequence (i.e., T_i is the ith task to execute in each frame). Such an execution order can be obtained, for example, through *topological sorting* algorithm in time $O(m+n)$. Fig. 1 presents an example task set with six tasks and precedence constraints. In Fig. 2, we show the execution order within a frame, obtained through topological sorting.

The worst-case execution time (WCET) of task T_i is denoted by c_i. We consider a system with DVFS capability where the clock frequency can vary from a minimum available frequency $f_{\min}$ to a maximum frequency $f_{\max}$ (normalized to 1.0). The execution time of task T_i under the frequency f_i is given by $(c_i)/(f_i)$. The *utilization* U of the task set is given as $\sum(c_i)/(D_f \cdot f_{\max}) = \sum(c_i)/(D_f)$; in other words, it corresponds to the *load* under the maximum CPU frequency.

In our model, each task T_i is allowed to have a different frequency-independent power figure P_{ind_i}, since each task may require access to different subsets of external devices. We assume that the system is *energy-constrained* in the sense that it has a

fixed energy budget E_f which is not replenishable during execution and cannot be exceeded in any frame of length D_f.

The *reliability* of a task is defined as the probability of completing the task successfully (i.e., without any transient faults) [40], [42], [43]. Assuming that the transient faults follow a Poisson distribution, the reliability of the task T_i with its WCET c_i is [43]

$$R_i(f_i) = e^{-\lambda(f_i)*\frac{c_i}{f_i}} \tag{4}$$

where f_i is its execution frequency and $\lambda(f_i)$ is defined as in (3). Given an execution order of tasks that satisfies the precedence constraints, the reliability of a real-time system depends on the correct execution of all tasks in an application [19], [33], [42]. In our application model, which consists of n tasks, the system reliability is, therefore, $R = \prod_{i=1}^{n} R_i(f_i)$.

Note that we also extend our framework to periodic tasks with possibly distinct periods in Section IV, at the beginning of which we provide more details about the model we assume in that analysis.

III. Reliability-Aware DVFS for Frame-Based Tasks

Our analysis starts with the case of frame-based tasks where tasks share a common deadline D_f. We first provide the static solution under the assumption that all tasks will take their worst-case workload, and then develop dynamic solutions where task frequencies are adjusted at runtime to improve overall reliability when the actual workload deviates from the worst-case.

A. Static Solution

In this section, we formalize and optimally solve the problem of finding task-level frequency assignments to maximize the system reliability, with a given energy budget E_f during a period D_f.

We first consider the impact of DVFS-induced time and energy overhead [25] on the problem formulation. Let Δ and ψ be the maximum transition time and energy overhead between any two frequency/voltage levels, respectively. In a given frame, during the execution of task set $T_1, \ldots, T_n$, the system's voltage/frequency changes at most n times (at task dispatch times). As a result, the total time overhead due to voltage/frequency scaling within a frame is bounded by $n \cdot \Delta$. Similarly, the total energy dissipation due to transitions is bounded by $n \cdot \psi$. Hence, it is safe to assume that the time and energy available for task execution within a frame is given by $D = D_f - (n \cdot \Delta)$ and $E = E_f - (n \cdot \psi)$, respectively. For simplicity, in the rest of this paper, we refer to D and E as (modified) deadline and energy budgets, respectively.

Recall that the probability of completing the task T_i without a fault (that is, its *reliability*) at the processing frequency f_i is $R_i(f_i) = e^{-\lambda(f_i)*(c_i)/(f_i)}$, where $\lambda(f_i)$ is given by (3). Hence, $R_i(f_i)$ is a strictly concave and increasing function of f_i. The total (i.e., frequency-dependent and frequency-independent) energy consumption of T_i at the frequency f_i can be expressed as [40]

$$E_i(f_i) = P_{\text{ind}_i} \cdot \frac{c_i}{f_i} + C_{\text{ef}} \cdot c_i \cdot f_i^2 \tag{5}$$

Notice that $E_i(f_i)$ is a strictly convex function and is minimized when $f_i = f_{\text{ee}_i}$ (Section II-A).

Let $\varphi_i(f_i) = \lambda(f_i) \cdot (c_i)/(f_i)$. Our problem can be stated as *to find* $f_i (1 \le i \le n)$ *values so as to maximize*

$$R = \prod_{i=1}^{n} R_i = e^{-\sum_{i=1}^{n} \varphi_i(f_i)} \tag{6}$$

Subject to

$$\sum_{i=1}^{n} \frac{c_i}{f_i} \le D \tag{7}$$

$$\sum_{i=1}^{n} E_i(f_i) \le E \tag{8}$$

$$f_{\min} \le f_i \le f_{\max} (1 \le i \le n) f \tag{9}$$

Above, the inequality (7) corresponds to the deadline constraint, while (8) encodes the hard energy constraint. The constraint set (9) gives the range of feasible frequency assignments.

Considering the well-known features of the exponential functions, we can re-express our objective as to *minimize* $\sum_{i=1}^{n} \varphi_i(f_i)$ subject to the constraints (7), (8) and (9). In the rest of this paper, this optimization problem will be called Energy-Constrained Reliability Management (ECRM) problem.

Let E_{limit} be the minimum energy that must be allocated to the given task system to allow their completion before or at the deadline D. Given the task parameters, E_{limit} can be computed by the polynomial-time algorithm developed in [4]. As a by-product, the same algorithm yields also the optimal task-level frequency assignments $(fl_1, fl_2, \ldots, fl_n)$ when the total energy consumption is *exactly* E_{limit}. Obviously, if $E < E_{\text{limit}}$, then there is no solution to our problem, since the system would lack the minimum energy needed for timely completion. Also, let $E_{\max} = \sum_{i=1}^{n} E_i(f_{\max})$ be the energy consumption of the task set when all tasks run at $f_{\max}$. As another boundary condition, when $E \ge E_{\max}$, executing all tasks at the maximum frequency is the optimal solution (since $R_i(f_i)$ is monotonically increasing with f_i and the system has sufficient energy to run at $f_{\max}$ continuously). Therefore, in the remainder of this paper, we will focus exclusively on settings where $E_{\text{limit}} \le E < E_{\max}$.

Lemma 1: In the optimal solution to ECRM, $\forall i \ f_i \ge f_{\text{ee}_i}$ where $f_{\text{ee}_i} = ((P_{\text{ind}_i})/(2C_{\text{ef}}))^{(1/3)}$ is the energy-efficient frequency for T_i.

Proof: This follows from the observation that executing T_i at a speed lower than f_{ee_i} would result in *increased* energy consumption for T_i (Section II-A). As a result, if a task were to execute at a frequency lower than f_{ee_i} in the optimal solution, then increasing its frequency to f_{ee_i} would actually *decrease* its energy consumption [while still satisfying (8)] and *increase* the overall reliability considering the positive impact of higher frequencies on task reliability—giving a contradiction. ■

Thanks to Lemma 1, the constraint (9) can be rewritten as $f_{\text{low}_i} \le f_i \le f_{\max} (1 \le i \le n)$, where f_{low_i} is $\max(f_{\min}, f_{\text{ee}_i})$.

Lemma 2: If $E_{\text{limit}} \le E < E_{\max}$, in the optimal solution to ECRM, the total energy consumption $\sum_{i=1}^{n} E_i(f_i)$ must be equal to E.

Proof: Assume that the statement is false. Since $E < E_{\max}$ by assumption, there must be a speed $f_i < f_{\max}$. In this case, it should be possible to increase f_i by $\varepsilon > 0$ such that $(f_i + \varepsilon) \leq f_{\max}$ and $\sum_{j \neq i} E_j(f_j) + E_i(f_i + \varepsilon) \leq E$. It is clear that the deadline and energy constraints are still satisfied after this modification. Further, the overall system reliability has improved due to the execution of T_i at a speed higher than f_i. Thus, the proposed solution cannot be optimal and we reach a contradiction. ■

Lemma 2 allows us to conclude that if $E_{\text{limit}} \leq E < E_{\max}$, then we can rewrite the constraint (8) as an equality in the form of $\sum_{i=1}^{n} E_i(f_i) = E$. Consequently, we obtain a new nonlinear (convex) optimization problem ECRM' defined as:
find f_i $(1 \leq i \leq n)$ *values so as to minimize*

$$\sum_{i=1}^{n} \varphi_i(f_i) \tag{10}$$

Subject to

$$\sum_{i=1}^{n} \frac{c_i}{f_i} \leq D \tag{11}$$

$$\sum_{i=1}^{n} E_i(f_i) = E \tag{12}$$

$$f_{\text{low}_i} \leq f_i \leq f_{\max} (1 \leq i \leq n) \tag{13}$$

The problem ECRM' can be solved, for instance, by *Quasi-Newton* techniques developed for constrained nonlinear optimization [7]. The technique exploits the well-known Kuhn–Tucker optimality conditions for nonlinear programs in an iterative fashion by transforming the original problem to a quadratic programming problem and solving it optimally [7]. While optimal, a theoretical complication with this approach is that it is practically impossible to express the maximum number of iterations as a function of the number of unknowns which, in this case, corresponds to the number of tasks n. However, in our experience, the algorithm is rather fast: For instance on a 1 GHz CPU with 1 GB memory, our implementation was able to return the optimal solution in less than 1.8 s even for task sets with 1000 tasks. The reported experimental results are based on using this optimal algorithm.

However, we also developed a heuristic algorithm that provably runs in polynomial-time. This algorithm, named ECRM-LU, satisfies the deadline, energy and frequency range constraints. Further, it yields solutions that are extremely close to the optimal solution. ECRM-LU proceeds as follows. We temporarily ignore the deadline constraint (11) and solve the problem ECRM' only by considering the energy constraint (12) and frequency range constraints (13). Notice that, by excluding the deadline constraint, the problem is transformed to a separable convex optimization problem with n unknowns, $2n$ inequality constraints and a single equality constraint. This problem, in turn, can be solved in time $O(n^3)$ by iteratively manipulating the Kuhn–Tucker optimality conditions in a way similar to the technique illustrated in algorithm given in [4]. Now, if this solution satisfies also the deadline constraint (11), obviously, it is also the solution to ECRM'. Otherwise, we rewrite the constraint set (13) as

$$fl_i \leq f_i \leq f_{\max} (1 \leq i \leq n) \tag{14}$$

where fl_i is the frequency assignment to task T_i in the solution where the task set completes *at exactly* D and with energy consumption E_{limit}. Again, the $\{fl_i\}$ values can be computed in time $O(n^3)$ [4]. By enforcing the constraint set (14), we make sure that the final speed assignments satisfy also the deadline constraint. Once again, this version of the problem where the deadline constraint is handled implicitly by enforcing the lower bounds on frequency assignments can be solved in time $O(n^3)$. Hence, the overall time complexity of ECRM-LU is also $O(n^3)$. Our extensive simulation studies show that ECRM-LU performs very well compared to the optimal solution through the almost entire spectrum: the reliability figures yielded by ECRM-LU are close to the optimal one by a margin of 0.03%, when $(E)/(E_{\text{limit}}) \geq 1.02$. In a tiny portion of the interval where $1.0 \leq (E)/(E_{\text{limit}}) < 1.02$, we observed a difference of at most 1%.

B. Dynamic Reliability-Aware Scheduling

The static solution presented in the previous section is optimal under the assumption that all tasks will present their worst-case workload (i.e., their WCETs). While provisioning for worst-case scenarios is imperative in real-time systems, in practice, many real-time tasks complete early without consuming their WCETs. In fact, numerous DVFS studies published in recent past were based on detecting and reclaiming unused CPU time (i.e., dynamic *slack*) to enhance energy savings by reducing the processing frequency at runtime [5], [6], [27]. A similar opportunity exists here: the *excess energy* that arises from early completions of tasks, can be used to *increase* the speeds of tasks at runtime, to improve the system reliability. Clearly, utmost care must be exercised to make sure that the system remain within its energy allowance (budget) before making such adjustments.

In this section, we develop online (dynamic) reliability-aware schemes for reclaiming the excess energy at runtime. In the following algorithms, we assume that n tasks in the real-time embedded application are executed in the order $T_1, T_2, \ldots, T_n$. The three dynamic algorithms that we developed are the following.

- BR: dynamic *basic reclaiming* algorithm. In this solution, an initial speed assignment is made by solving the static problem presented in the preceding section, assuming worst-case workload for each task. At task completion points, the excess energy that may be available (due to early completions) is effectively recycled within the system: a new speed (frequency) assignment is made for the *remaining* tasks by considering the remaining (updated) energy budget and time to deadline. This assignment is obtained by reinvoking our optimal solution to the problem ECRM.
- GRE: dynamic *greedy* algorithm. Although BR satisfies the energy and deadline constraints, it is pessimistic in the sense that it assumes WCETs for all tasks when redistributing the excess energy. An alternative approach may be to allocate the excess energy *entirely* to the next task[1] T_{next} at each task completion point, relying on the fact that T_{next} is also likely to complete early and release excess energy for the use of the remaining tasks. In the mean time,

[1]Note that if the next task cannot be assigned the entire excess energy due to the maximum frequency limitations, then the following task(s) will be able to reclaim energy at the next task completion points.

the reliability of T_{next} will be significantly improved due to execution at high speed. GRE preserves feasibility in terms of timing and energy constraints: compared to the initial static solution obtained by ECRM, the task speeds never decrease (guaranteeing the timely completion) and only the excess energy that is obtained at runtime is reassigned.

- AGR: dynamic *aggressive* algorithm. This scheme represents the most speculative solution, in the sense that it counts on *probable* early completions before execution, and makes speed assignments accordingly. The main idea is to *aggressively give sufficient energy to the current task while still leaving minimum required energy for the tasks to follow, based on their WCETs*. It is speculative, because under a worst-case scenario (where most of the tasks present high workload), many tasks in the chain would be forced to execute at low speeds to guarantee the completion within the energy budget, significantly lowering the overall reliability. However, in settings where the actual workload is likely to deviate from the worst-case with high probability, this strategy will (and, as we show in the performance evaluation section, does) pay off. Specifically, AGR is implemented through the following steps: First, E_{limit} [which is defined as the minimum energy needed to complete all the tasks by their deadline (Section III-A)] is computed. Next, a speed assignment $(fl_1, fl_2, \ldots, fl_n)$ based on WCETs of all tasks but with energy allocation equal to E_{limit} is computed. $T_2, \ldots, T_n$ are *tentatively* assigned the frequencies $(fl_2, \ldots, fl_n)$ and their energy consumption with these assignments and WCETs (E_{reserve}) are evaluated. Then, the entire remaining energy (i.e., $E - E_{\text{reserve}}$) is allocated to the first task T_1, allowing it to execute as fast as possible within the given constraints. At task completion points, the above steps are repeated by considering the early completions and *actual* remaining energy for remaining tasks. The fact that this solution preserves the energy and deadline constraints follows from the properties of the speed assignments that corresponds to E_{limit}, which is the minimum energy needed for a feasible solution.

C. Simulation Results and Discussion

To evaluate the performance of our dynamic algorithms under varying workload conditions, we designed a discrete-event simulator in C. In our simulator, we implemented *Basic Reclaiming (BR)* scheme, *Greedy Reclaiming (GRE)* scheme, *Aggressive (AGR)* scheme, in addition to the *Static* scheme which computes the processing frequencies using the optimal solution to problem ECRM assuming the worst-case workload for each task. *Static* does not use any online component in the sense that no dynamic speed adjustment is performed, regardless of the actual workload. Finally, we also implemented the *Clairvoyant* scheme (denoted as *Bound*), that knows the *actual* workload of each task in advance and computes the optimal speed assignments to maximize the reliability by solving the problem ECRM accordingly. *Bound* is not a practical scheme (since it assumes the knowledge of the actual workload in advance); however, it characterizes the upper bound on the performance of *any* static or dynamic algorithm.

In our simulations, we considered 1000 task sets each containing eight tasks, with the frame/period length of $D = 1000$. The total *utilization (U)* of each task set under maximum frequency is varied from 0.2 to 1.0 (full load). The worst-case execution time *(WCET)* of each task (under $f_{\max}$) is randomly generated from $0.01 \cdot U \cdot D$ to $0.9 \cdot U \cdot D$. To model the variations in the actual workload, we use the ratio $(ACET)/(WCET)$, which denotes the ratio of the *average-case execution time (ACET)* to the worst-case execution time. The lower this ratio, the more the actual workload deviates from the worst-case. For each task set and utilization value, $(ACET)/(WCET)$ is changed from 0.2 to 1.0. The actual workload ac_i of each task is generated randomly, using normal distribution. Each of the 1000 task sets is executed 1000 times for a given $(ACET)/(WCET)$ and U value. The results that are shown correspond to the average of all runs.

We model a DVFS-enabled CPU where the normalized processing frequencies can change from $f_{\min} = 0.1$ to $f_{\max} = 1.0$. The power figures are normalized with respect to the frequency-dependent power component P_d, which is taken as 1.6 W at the maximum frequency (modeled after the power consumption of Intel XScale as in [1], [37]). We model the frequency-independent power P_{ind} following the methodology and actual external device specifications from [11]. As in [11], we assume that each task uses up to 2 devices during execution, which gives a normalized frequency-independent power range of [0, 2] per task. The specific P_{ind} value for each task is determined randomly according to uniform distribution in this range.

To analyze the impact of system's energy budget E on the performance, we varied E from E_{limit} (minimum energy needed to meet the deadline, see Section III-A) to $E_{\max}$ (energy consumption at $f_{\max}$). The ratio $(E)/(E_{\text{limit}})$ shown in the plots is a measure of the available energy in the system; for example, when $(E)/(E_{\text{limit}}) = 1.2$ the system has 20% more energy than the minimum needed to meet the deadline. We assume that the transient faults' occurrence is determined by Poisson distribution and given by (3), where $d = 3$ and $\lambda_0 = 10^{-9}$. $\lambda_0 = 10^{-9}$ corresponds to 100 FITs (failures in time, in terms of errors per billion hours of use) per megabit at $f_{\max}$, which is a reasonable SER rate as suggested in [15] and [44]. For each task, once the actual workload ac_i and its runtime frequency f_i is determined by the underlying algorithm at runtime, its reliability is computed as $R_i(f_i) = e^{-\lambda(f_i)*(\text{ac}_i)/(f_i)}$. The overall reliability of the task set is then derived as $R = \prod_{i=1}^{n} R_i(f_i)$.

Fig. 3 shows the relationship between the probability of failure *(PoF)* and the energy budget when $(ACET)/(WCET) = 0.5$ and $U = 0.4$. As expected, *PoF* (defined as $1 - R$) generally decreases with increasing energy budget $((E)/(E_{\text{limit}})$ ratio), since more energy enables the system to use higher processing speeds with improved reliability. The clairvoyant *Bound* scheme achieves a constant probability of failure, because even when $E/E_{\text{limit}} = 1$, all tasks can be executed at $f_{\max}$ thanks to *a priori* knowledge of actual execution times, which are, on the average, half of the worst-case in these experiments—since processing speeds beyond $f_{\max}$ are not available, giving more energy to *Bound* does not further help. We observe that, among the other schemes, the static scheme (which does not perform any dynamic energy reclamation) performs worst and *AGR* is the best, with very close performance to *Bound* when $((E)/(E_{\text{limit}}) > 1.1)$. This result indicates

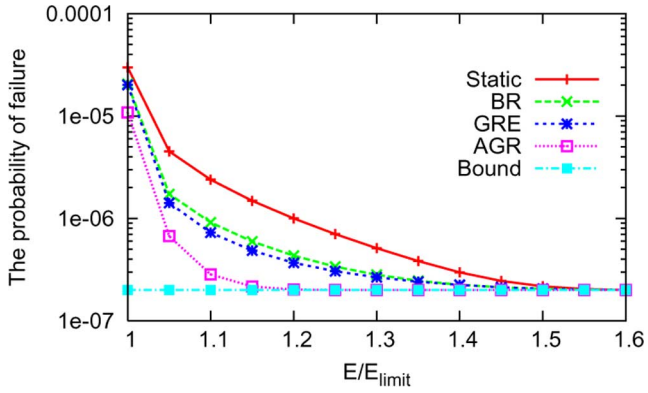


Fig. 3. The probability of failure versus E/E_{limit} with $(ACET)/(WCET) = 0.5$ and $U = 0.4$.

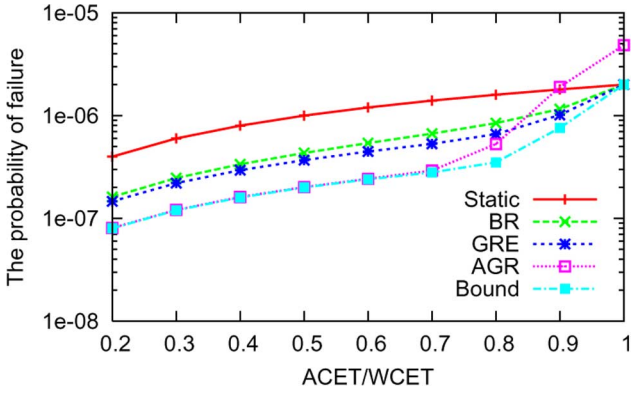


Fig. 5. The probability of failure versus the $(ACET)/(WCET)$ ratio with $U = 0.4$ and $E/E_{\text{limit}} = 1.2$.

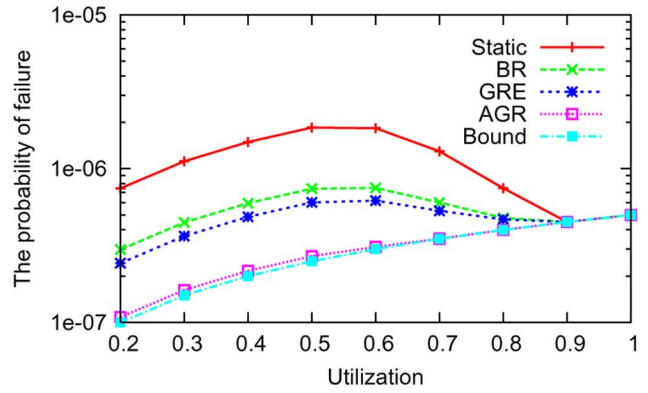


Fig. 4. The probability of failure versus utilization (U) with $(ACET)/(WCET) = 0.5$ and $E/E_{\text{limit}} = 1.15$.

that aggressively giving maximum energy to the tasks that will execute early typically pays off in these settings, since these tasks are also likely to generate excess energy due to early completions, which can be allocated to the later tasks. *BR* is a little worse than *GRE*; but both perform consistently better than *Static*, verifying the benefits of dynamic reclaiming. Observe that once the available energy is 40% or more compared to E_{limit}, all dynamic schemes perform almost the same.

Fig. 4 illustrates how the probability of failure changes as a function of task utilization U, for $(ACET)/(WCET) = 0.5$ and $E/E_{\text{limit}} = 1.15$. In general, the impact of increasing utilization is manifested in two different ways on the overall reliability. First, as the utilization increases, the workload (in terms of number of cycles) increases, translating into the increased probability of incurring transient faults, under comparable transient fault rates. Second, with increased utilization, the system may be forced to adopt higher frequencies to meet the deadline, depending on the available energy. These two factors tend to affect the overall reliability in reverse directions as implied by eq. (4). The relative ordering of the schemes remains the same as in Fig. 3. We observe two interesting trends: For *Static*, *BR* and *AGR* schemes, as we increase the utilization towards the range of 0.5–0.6, the probability of failure first increases. In the utilization range [0.2–0.5], for these three schemes, the increase in frequency is modest and the *PoF* is primarily determined by the increase in the workload. However, higher utilization values force the system to adopt significantly higher frequencies in order to meet the deadline and the positive impact of this on reducing the fault rates becomes the primary factor after a certain point. In fact, after $U = 0.8$, all tasks are executed at speeds close to $f_{\max}$ and then *PoF* drops sharply to a minimal value. On the other hand, the probability of failure for *AGR* and *Bound* are quite low and it tends to increase monotonically (but relatively modestly) throughout the spectrum. This is because, due to low ratio of $(ACET)/(WCET) = 0.5$, *AGR* and *Bound* can execute almost all the tasks with $f_{\max}$ while remaining within the energy budget, even starting at $U = 0.2$. Therefore, when executing all tasks with fixed speed $f_{\max}$, the *PoF* for *AGR* and *Bound* is mainly determined by the execution times, which increase with increasing utilization. Also, we notice that when the utilization increases from 0.9 to 1.0, the *PoF* for all schemes increases since all the tasks can be executed with $f_{\max}$ starting from $U = 0.9$.

Fig. 5 shows the impact of the variability in the actual workload (i.e., the $(ACET)/(WCET)$ ratio) on the probability of failure, with $E/E_{\text{limit}} = 1.2$ and $U = 0.4$. In general, we find that the probability of failure increases with the increased ratio of $(ACET)/(WCET)$. This is to be expected, because with the increased ratio of $(ACET)/(WCET)$, at runtime, tasks execute longer and they are subject to transient faults with higher probabilities. However, observe that the dynamic schemes are able to significantly reduce the probability of failure compared to *Static* thanks to online reclaiming features, especially at low $(ACET)/(WCET)$ ratios. When $(ACET)/(WCET) = 0.9$, the probabilities of failure of *Static*, *BR* and *GRE* converge to that of *Bound*, since there are almost no early completions or excess energy at runtime. However, it is interesting to note that the probability of failure of *AGR* is slightly higher from $U = 0.9$. This is because, when all tasks almost take their *WCET* with the high utilization $U \geq 0.9$, the expected early completions typically do not occur, while the aggressive nature of *AGR* still forces the later tasks to execute at relatively low speeds, causing a loss in reliability.

These patterns can be also used to *establish guidelines* for system designers who need to figure out the *minimum amount of energy supply* that must be provided to the system, to achieve a certain target probability of failure. Fig. 6 establishes these thresholds for $(ACET)/(WCET) = 0.5$ and $U = 0.4$. As can be seen from the figure, to achieve a target *PoF*, the amount of energy that must be supplied to the system is largest for *Static*, that cannot reclaim excess energy at runtime. This

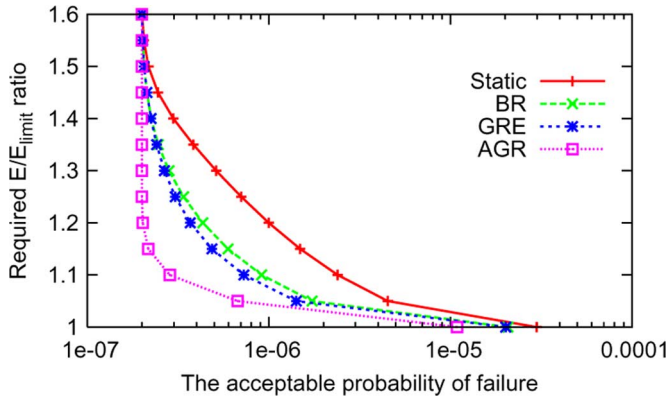


Fig. 6. The acceptable probability of failure versus the required E/E_{limit} ratio.

TABLE I
DYNAMIC RECLAIMING OVERHEADS

Scheme	4 Tasks	8 Tasks	16 Tasks
AGR	0.67ms	0.68ms	0.68ms
GRE	0.65ms	0.65ms	0.66ms
BR	1.6ms	2.9ms	5.5ms
*BR**	0.86ms	1.23ms	1.79ms

amount sharply decreases with the use of dynamic reclaiming algorithms, and is minimum for the best-performing algorithm *AGR*. For example, 20% additional energy (beyond E_{limit}) must be provided to the system to achieve a probability of failure of 10^{-6}, with *Static*. However, if dynamic schemes (e.g., *BR* or *GRE*) are available, then 7% additional energy is sufficient. The difference between the schemes becomes very important at *low* values for the target probability of failure (e.g., smaller than 10^{-6}); indicating that in safety-critical applications, the available energy budget may be a prime factor to achieve a target reliability figure.

Runtime Characteristics: We also conducted experiments to measure the runtime overhead of dynamic reclaiming routines that require computation of excess energy and reassignment of CPU frequencies. The results, presented in Table I for task sets of different sizes, show the extra CPU time used by *AGR, GRE* and *BR* at each invocation point. These results are obtained on a 1 GHz CPU with 1 GB of memory. *GRE* has the lowest overhead, by virtue of modifying the frequency of only one task at each invocation. *AGR* has also a rather low overhead, which is different from *BR* which needs to resolve the ECRM problem at each invocation, by considering the remaining energy. We also implemented an alternative version of *BR*, denoted by BR^*, that uses the polynomial-time algorithm ECRM-LU as the main optimization routine. The results suggests that BR^* is very effective in reducing the runtime overhead of the basic reclaiming scheme. Overall, we can conclude that the dynamic reclaiming algorithms can be implemented with low overhead in practice.

IV. EXTENSION TO PERIODIC TASKS

In this section, we extend our solutions to a more general model where tasks may have different periods. Specifically, we consider a set of periodic real-time tasks $\Gamma = (T_1, \ldots, T_n)$, where the task T_i has the period p_i. The task T_i is characterized by its worst-case execution time (under maximum frequency) c_i and frequency-independent power figure P_{ind_i}. The first job of each task is assumed to arrive at time 0. The jth job (instance) of T_i, referred as T_{ij}, arrives at time $(j-1)\cdot p_i$ and has a deadline $j\cdot p_i$. The power and fault models are the same as those assumed in Section III.

With the periodic task model and DVFS, the effective system utilization depends on the execution frequencies of the tasks. Hence, the effective utilization of task T_i at frequency f_i is $U_i(f_i) = u_i/f_i$, where $u_i = (c_i)/(p_i)$ is the nominal utilization of the task T_i under maximum frequency. The (nominal) system utilization under maximum frequency ($\sum U_i(1.0) = \sum_{i=1}^{n} u_i$) is denoted by U in this section.

We assume preemptive earliest-deadline-first (EDF) scheduling, which is known to be an optimal real-time scheduling policy: with EDF, the necessary and sufficient condition for the feasibility of the task set is $U \leq 1.0$ [20]. Hence, throughout this section, we assume that the system utilization U when all tasks are executed at $f_{\max}$ never exceeds 100% in order to guarantee the feasibility.

A. Static Solution

For periodic real-time tasks, considering that it is sufficient and necessary to obtain an optimal solution during a hyperperiod (the least common multiple (LCM) of all the periods), we can restate our problem as: *Given the energy budget within the hyperperiod, find the optimal frequency assignments to individual jobs so that the overall reliability is maximized while the feasibility is preserved.*

As in Section III, we assume that the available energy budget E during LCM has been adjusted by taking into account the total number of jobs $N = \sum_{i=1}^{n} \text{LCM}/p_i$ and transition energy overhead ψ. In the same vein, the transition time overhead can be factored in the worst-case execution time c_i of each job. Note that for periodic task sets, the overall reliability of the application is the product of *all* the jobs of all the tasks that are executed during the hyperperiod. Moreover, in general, each task instance T_{ij} (the jth job of task T_i) may have its own frequency f_{ij} which could be potentially different than the frequency of other jobs of the same task T_i in the optimal solution. Considering that the total number of jobs during the hyperperiod could be exponential in the number of tasks, at first, the problem looks rather challenging since even the number of unknowns (f_{ij} values) is prohibitively large. Fortunately, the following theorem indicates that one can commit to *task-level* frequency assignments where a given task runs at a constant speed f_i, without compromising optimality.

Theorem 1: For periodic task sets, it is always possible to find an optimal solution where all the jobs of a given task T_i run at the same frequency throughout the hyperperiod.

Proof: Consider an optimal solution with overall reliability R, where the task instance (job) T_{ij} runs at frequency f_{ij}. Call this schedule S. We will show that one can transform S to another schedule S' where: i) the feasibility is preserved; ii) the energy budget constraint is satisfied; iii) the overall reliability is no less than R; and iv) all jobs of a given task T_i run at constant speed f_i. During the hyperperiod, there are $k_i = (\text{LCM})/(p_i)$

jobs of task T_i. Now, consider the schedule S', where T_i runs at constant speed $f_i = (\sum_{j=1}^{k_i} f_{ij})/(k_i)$.

First, we show that S' is feasible with these frequency assignments. To start with, observe that $f_i \leq \max\{f_{ij}\}$. If S is feasible, $\max\{f_{ij}\} \leq 1.0$, and we get $f_i \leq 1.0$ implying that no task runs at a frequency higher than $f_{\max} = 1.0$ in S'. Similarly, one can easily see that $f_i \geq \min\{f_{ij}\} \geq f_{\min}$. Moreover, since S is feasible, the total execution time during hyperperiod should not exceed the length of LCM, that is, $\sum_{i=1}^{n}\sum_{j=1}^{k_i}(c_i)/(f_{ij}) \leq \text{LCM}$. In other words, $\sum_{i=1}^{n}\sum_{j=1}^{k_i}(c_i)/(f_{ij})(1)/(\text{LCM}) \leq 1$. Applying first $u_i = (c_i)/(p_i)$, and then $(p_i)/(\text{LCM}) = (1)/(k_i)$, we conclude that $\sum_{i=1}^{n}(1)/(k_i)\sum_{j=1}^{k_i}(u_i)/(f_{ij}) \leq 1$ holds.

On the other hand, the effective system utilization in S' is $U' = \sum_{i=1}^{n}(u_i)/(f_i)$. Due to convex nature of the function $U(f) = (u/f), (u_i)/(f_i) \leq (1)/(k_i)\sum_{j=1}^{k_i}(u_i)/(f_{ij})$ holds. Consequently, $U' = \sum_{i=1}^{n}(u_i)/(f_i) \leq \sum_{i=1}^{n}(1)/(k_i)\sum_{j=1}^{k_i}(u_i)/(f_{ij}) \leq 1$, giving $U' \leq 1$, which is necessary *and* sufficient for feasibility under EDF. As a result S' is still feasible.

Next, we show that, in S' the energy constraint is not violated. Let the energy consumption of schedules S and S' during LCM be X and X', respectively. Obviously, $X \leq E$ by assumption. Since $f_i = (\sum_{j=1}^{k_i} f_{ij})/(k_i)$ and $E_i(f)$ is a strictly convex function, one can infer that the energy consumption of a given task T_i in S', namely, $k_i \cdot E_i(f_i)$, does not exceed the energy consumption of the same task in S, namely $\sum_{j=1}^{k_i} E_i(f_{ij})$. Hence, we get $\sum_{i=1}^{n} k_i \cdot E_i(f_i) \leq \sum_{i=1}^{n}\sum_{j=1}^{k_i} E_i(f_{ij})$, giving $X' \leq X \leq E$ and showing that S' still satisfies the energy constraint.

Finally, along the same lines, we show that reliability R' of S' is no worse than the reliability R of the schedule S. Using the given parameters, we compute $R = \Pi_{i=1}^{n} R_i = \Pi_{i=1}^{n}\Pi_{j=1}^{k_i} R_{ij} = e^{-\sum_{i=1}^{n}\sum_{j=1}^{k_i}\varphi_i(f_{ij})}$, while the new reliability is $R' = \Pi_{i=1}^{n} R'_i = e^{-\sum_{i=1}^{n} k_i\varphi_i(f_i)}$. Once again, since φ_i is a convex function, $k_i\varphi_i(f_i) \leq \sum_{j=1}^{k_i}\varphi_i(f_{ij})$ holds due to the fact that $f_i = (\sum_{j=1}^{k_i} f_{ij})/(k_i)$. As a result, in the new schedule, none of task-level reliabilities, which is the product of the reliabilities of individual jobs for a given task, decreases. This implies that the total reliability over all the tasks, R', cannot be worse than the original reliability R, completing the proof. ∎

Thanks to Theorem 1, the total energy consumption of T_i within LCM, at the optimal frequency level f_i can be expressed as

$$\begin{aligned} E_i(f_i) &= (P_{\text{ind}_i} + C_{\text{ef}} f_i^3)\frac{u_i}{f_i}\text{LCM} \\ &= (P_{\text{ind}_i} + C_{\text{ef}} f_i^3)\frac{c_i}{f_i} n_i \end{aligned} \tag{15}$$

where $n_i = (\text{LCM})/(p_i)$ is the number of jobs of T_i within a LCM. Therefore, our problem for the periodic tasks can be formally expressed *to find task-level frequency* f_i $(1 \leq i \leq n)$ *so as to maximize*

$$R = \prod_{i=1}^{n} R_i = e^{-\text{LCM}\sum_{i=1}^{n}\frac{\varphi_i(f_i)}{p_i}} = e^{-\sum_{i=1}^{n} n_i\varphi_i(f_i)} \tag{16}$$

Subject to

$$\sum_{i=1}^{n}\frac{u_i}{f_i} \leq 1 \tag{17}$$

$$\sum_{i=1}^{n} E_i(f_i) \leq E \tag{18}$$

$$f_{\min} \leq f_i \leq f_{\max}(1 \leq i \leq n) \tag{19}$$

where the inequality (17) encodes the feasibility constraint for EDF. Once again, by noting the monotonic features of the exponential functions, our objective can be re-expressed as to *minimize:* $\sum_{i=1}^{n} n_i\varphi_i(f_i)$ with the constraints (17), (18) and (19). This optimization problem is called Periodic Energy-Constrained Reliability Management (P-ECRM) problem.

To solve P-ECRM, similar to the solution for ECRM, we first need to find E_{limit} and $E_{\max}$. Here, E_{limit} is redefined as the minimum energy that must be allocated to the given periodic task set to preserve the feasibility with EDF during the hyperperiod, and $E_{\max}$ represents the energy consumption of the task set when all jobs run at $f_{\max}$. With given task parameters, E_{limit} can be computed by the polynomial-time algorithm suggested in [4] and $E_{\max}$ is equal to $\sum_{i=1}^{n} E_i(f_{\max})$. Obviously, if $E < E_{\text{limit}}$, then there is no solution to P-ECRM because E_{limit} is the minimum energy needed to guarantee the timing constraints. Likewise, when $E \geq E_{\max}$, we can simply assign the maximum frequency to all the tasks without exceeding the energy budget E, to achieve the best system reliability. Therefore, to solve P-ECRM, one needs to focus mainly on the case where $E_{\text{limit}} \leq E < E_{\max}$.

Let $\varphi'_i(f_i) = n_i \cdot \varphi_i(f_i)$, which is still a convex function. A close inspection of P-ECRM reveals that it has the same *mathematical form* as the ECRM problem introduced in Section III, since task execution times and the deadline in ECRM are replaced by task utilizations and 1.0 (EDF's utilization bound). As a result, the same solution methodology suggested in Section III-A is applicable.

B. Dynamic Solutions

Similar to the case of frame-based tasks, the static solution is optimal only when all tasks execute their WCETs in every instance. Since early completions are rather common in actual executions, we have again opportunities for improving overall reliability by speeding up execution at runtime by recycling the excess energy. In this section, we suggest three online (dynamic) reliability-aware schemes for reclaiming the excess energy that becomes available for periodic tasks. In all these algorithms, recomputation and reallocation of excess energy are performed only at job completion points.

In fact, it is tempting to use the framework suggested in Section III-A, by recomputing the new frequency assignments by considering *all* jobs yet-to-complete. However, unlike the case of frame-based tasks, under preemptive EDF scheduling we may have a number of *preempted* jobs in the ready queue that already started their execution in addition to jobs that will be released in the future, at reclamation points. These preempted jobs may have varying amounts of workload already completed under a prior frequency assignment. In our schemes, to preserve runtime efficiency and to be able to use P-ECRM

algorithm with slight modifications for task-level frequency assignment, such preempted jobs are not considered for excess energy allocation. Rather, excess energy is distributed only to the jobs that will be released in the future. In addition, Theorem 1 implies that a task-level frequency assignment gives an optimal redistribution of excess energy to these jobs. Consequently, the proposed algorithms take into account the energy needed to allow the preempted jobs to complete their remaining workload at their current frequency assignments. The algorithms that we propose are the following.

- ***P-BR***: dynamic *basic reclaiming* algorithm for periodic tasks. Initially all frequency assignments are made by solving the P-ECRM problem under worst-case workload assumption. When a job completes without consuming its WCET, this excess energy is reallocated to jobs that have not yet started their executions. The new task-level frequency assignments for these jobs can be again obtained by invoking our optimal solution to the problem P-ECRM with updated energy budget. When reinvoking the algorithm P-ECRM at runtime, one needs to only update n_i in expressions (15) and (16) for the number of future jobs of task T_i. Compared to the initial speed assignments, the new frequencies never decrease due to excess energy available on the runtime. As a result, the feasibility of the resulting schedule is guaranteed in *P-BR* scheme.
- ***P-GRE***: dynamic *greedy* algorithm for periodic tasks. *P-GRE* is an alternative approach that attempts to allocate the entire excess energy to the next task to execute (T_{next}), when a task instance completes. In other words, all future jobs of T_{next} will be given a new (higher) frequency and their reliability will be significantly improved. If T_{next} cannot use the entire excess energy due to the maximum frequency limitations, then the remaining part is allocated to another task at the next job completion point. Again, the feasibility of *P-GRE* follows from the fact that none of the tasks runs at a frequency level lower than its original frequency assigned by the static solution.
- ***P-AGR***: dynamic *aggressive* algorithm for periodic tasks. This scheme extends the speculative *AGR* scheme proposed for the frame-based tasks, by speeding up all the future jobs of the next task to be dispatched as much as possible, by reserving only a minimum amount of energy for instances of other tasks. First, we compute E_{limit} and store the frequency assignments $(fl_1, fl_2, \ldots, fl_n)$ obtained through E_{limit}. Then, at reclamation point, if a job task T_i is about to start its execution for the first time, we recompute the frequency for the jobs of the remaining tasks $\Gamma - T_i$ by evaluating the speed assignments with E_{limit} and their energy consumption with these assignments and WCETs (E_{reserve}) is evaluated. Then, the entire remaining energy (i.e., $E - E_{\text{reserve}}$) is allocated exclusively to the current and future job instances of the task T_i, allowing it to execute as fast as possible within the given constraints. This solution also preserves the energy and feasibility constraints, which can be seen from the properties of the frequency assignments that correspond to E_{limit} (the minimum energy requirement for a feasible solution).

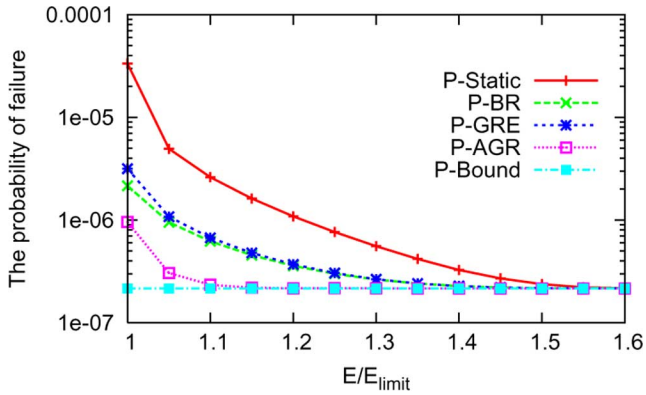


Fig. 7. The probability of failure versus E/E_{limit} with $(ACET)/(WCET) = 0.5$ and $U = 0.4$.

C. Simulation Results and Discussion

In this section, we evaluate the performance of our dynamic reclaiming schemes proposed for periodic tasks. In addition to the *Static Scheme (P-Static)*, we implemented in our simulator *P-BR, P-GRE*, and *P-AGR*. We also implemented the *Clairvoyant Scheme (P-Bound)* that computes an absolute bound on the optimal solution, by using the P-ECRM algorithm and the actual workload information *in advance*.

The experimental methodology we follow in this section is parallel to that described in Section III-C. We generated 1000 task sets each with eight tasks. The periods of the tasks are generated randomly in the range [72, 1080] ms, which are comparable to task period values encountered in real applications [23]. We made sure that the least common multiple period LCM of the periods is 1080 for every task set. This allows us to compare different task sets over the same operation period $\text{LCM} = 1080$ for various utilization and energy budget values.

Fig. 7 shows how the probability of failure changes with increasing energy budget when $(ACET)/(WCET) = 0.5$ and $U = 0.4$. First, we observe the same decreasing trends for probability of failure with increasing energy budget and same relative order of reliability among these schemes as those in Fig. 3 obtained for frame-based tasks. Compared to Fig. 3, there are two main differences: First, *P-BR* provides a slightly better reliability performance compared to *P-GRE*. In the case of periodic tasks, there are typically a large number of jobs that can benefit from dynamic reclamation during the hyperperiod, and a more balanced energy allocation that considers all the tasks tends to be better than the greedy approach. Second, the performances of three dynamic schemes show further improvements with respect to the static scheme *P-Static*, even when $(E)/(E_{\text{limit}}) = 1$. With our dynamic schemes, the more jobs complete early, the more jobs in the future can be executed at the higher frequencies (occasionally, even at $f_{\max}$) due to the dynamic reclaiming. In fact, even when $E = E_{\text{limit}}$, many jobs complete early when $(ACET)/(WCET) < 1.0$. Hence, the advantages of dynamic schemes over the static scheme *P-Static* become more pronounced with increased number of jobs that translate to increased number of reclamation/speedup points.

Fig. 8 illustrates the relationship between the probability of failure and task set utilization U for $(ACET)/(WCET) = 0.5$ and $E/E_{\text{limit}} = 1.15$. Again, the main trends observed for

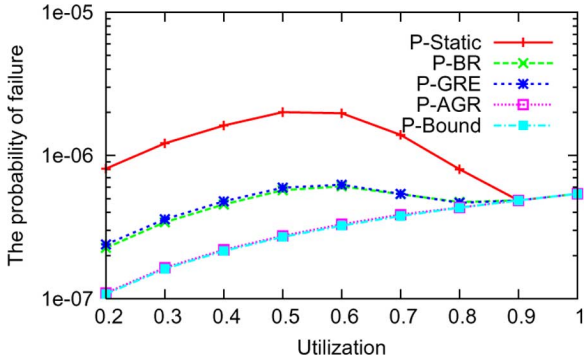


Fig. 8. The probability of failure versus utilization(U) with $(ACET)/(WCET) = 0.5$ and $E/E_{\text{limit}} = 1.15$.

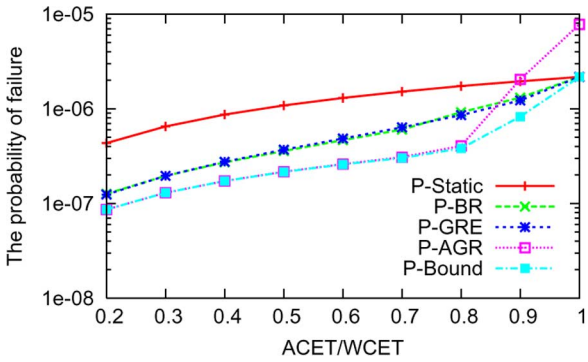


Fig. 9. The probability of failure versus the $(ACET)/(WCET)$ ratio with $U = 0.4$ and $E/E_{\text{limit}} = 1.2$.

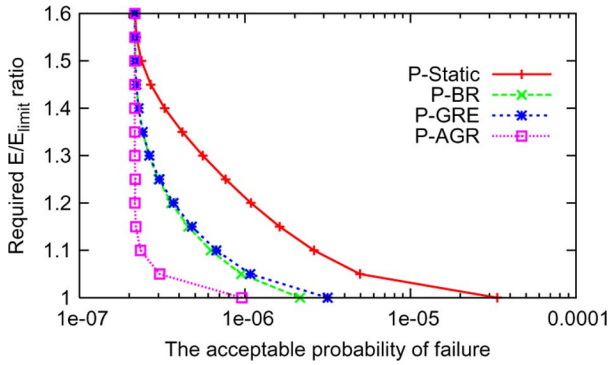


Fig. 10. The acceptable probability of failure versus the required E/E_{limit} ratio.

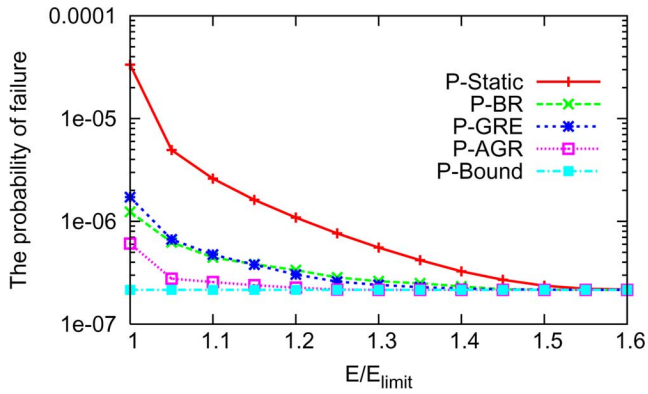


Fig. 11. The probability of failure versus E/E_{limit} with $(ACET)/(WCET) = 0.5$ and $U = 0.4$ (LCM $= 2160$).

frame-based tasks can be found in the case of periodic tasks. Specifically, increasing the utilization has potential impact on two factors that affect the reliability differently: the increase in task workloads tends to increase *PoF*; but for cases where the system has to use significantly higher frequencies, this becomes the dominant factor and *PoF* drops. The latter case can be observed in the utilization range [0.5–0.9] for *P-Static, P-BR* and *P-GRE*. A notable difference compared to the experiments with frame-based tasks is that the performance of *P-AGR* is almost identical to that of *P-Bound*. The reason is that with large number of jobs during the hyperperiod, we typically have sufficient energy to aggressively assign the current job a frequency close to $f_{\max}$, which approaches that assigned by *P-Bound* through the advance knowledge of actual workload. Due to the jobs' very frequent early completions (recall that $(ACET)/(WCET) = 0.5$ in these experiments), the speculative strategy of *P-AGR* pays off.

Fig. 9 shows the impact of the variability in the actual workload (i.e., the $(ACET)/(WCET)$ ratio) on the probability of failure, with $E/E_{\text{limit}} = 1.2$ and $U = 0.4$. In general, similar to the results presented in Fig. 5 for frame-based tasks, under fixed utilization, the probability of failure is primarily determined by the effective workload, which tends to increase with larger $(ACET)/(WCET)$ ratios. This results in monotonically increasing *PoF* values. Compared to the results for frame-based tasks, the difference between static schemes and our three dynamic schemes becomes bigger, which verifies that the dynamic schemes can achieve better reliability for periodic tasks since there are more available energy to be dynamically reclaimed due to early completions of numerous jobs during the hyperperiod. An interesting observation is that the almost perfect performance of *P-AGR* degrades rapidly when $(ACET)/(WCET) > 0.8$, implying that it should be used only when there is great confidence about high variability in the actual workload. With settings $(ACET)/(WCET) = 0.5$ and $U = 0.4$, Fig. 10 shows the required minimum energy supply to achieve a target (acceptable) probability of failure. As we can see, if the system's acceptable probability of failure is 10^{-6}, we need to allocate 20% additional energy (beyond E_{limit}) when using *Static* scheme. However, when dynamic schemes (e.g., *P-BR* or *P-GRE*) are applied, then 5% additional energy is sufficient. Furthermore, when using *P-AGR*, as long as the likelihood of early completions is high (such as the case here with $(ACET)/(WCET) = 0.5$), there is practically no need to allocate energy beyond E_{limit} if the target probability of failure is less than 10^{-6}. We also repeated the experiments for a larger LCM, namely for LCM $= 2160$. The results are quite similar to those reported above. Due to space limitations, we only report the impact of energy budget and utilization on the probability failure, in Figs. 11 and 12, respectively.

To summarize, these results indicate that our dynamic schemes can achieve even better performance compared to the static schemes in the case of periodic tasks, because there are more dynamic energy reclamation opportunities with the larger number of jobs that execute within the hyperperiod.

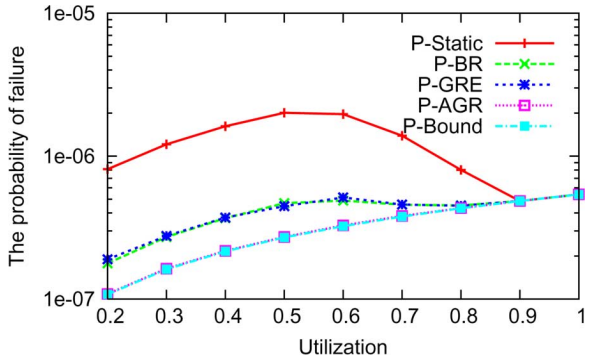


Fig. 12. The probability of failure versus utilization(U) with $(ACET)/(WCET) = 0.5$ and $E/E_{\text{limit}} = 1.15$ (LCM $= 2160$).

TABLE II
DYNAMIC RECLAIMING OVERHEADS

Scheme	4 Tasks	8 Tasks	16 Tasks
P-AGR	0.68ms	0.69ms	0.70ms
P-GRE	0.66ms	0.66ms	0.66ms
P-BR	1.9ms	3.4ms	6.5ms
*P-BR**	0.98ms	1.42ms	2.13ms

Runtime Characteristics: Our measurements to explore the runtime overhead of dynamic reclaiming algorithms for periodic task sets, presented in Table II, yielded figures similar to those obtained for frame-based tasks (Table I). Again, the numbers correspond to average CPU time used by the reclamation routines at each invocation (job completion) point. *P-BR* a has a larger overhead compared to *BR*, primarily due to the need of solving P-ECRM online for large number of jobs. However, we note that its overhead can be reduced by employing the ECRM-LU algorithm (adapted to periodic settings) as the optimization routine. That variation is denoted as *P-BR** in Table II.

D. Practical Considerations

Our solution framework assumed frequency levels that can assume any value in the range $[f_{\min}, f_{\max}]$. In *discrete-frequency* settings where the frequency levels can assume k distinct levels $f_1, \ldots, f_k$, the problem gets decidedly more complex. In fact, in [2], it is shown that merely deciding the feasibility of a set of frame-based tasks with common release time and deadline in discrete frequency settings under hard energy constraint is NP-Complete. This immediately implies that the more general problem of maximizing the reliability of frame-based tasks under both hard deadline and energy constraints is intractable as well. For periodic tasks having access to nonsharable resources protected by semaphores and mutex locks, the use of resource access protocols such as *Priority Inheritance Protocol, Priority Ceiling Protocol* and *Stack Resource Policy* [22] will be necessary. Such settings, in general, imply nontrivial feasibility conditions and warrant further investigation.

V. CONCLUSION

Recent research has identified significant and negative impact of the popular energy management technique DVFS on the reliability of real-time embedded systems. In this paper, we considered both frame-based and periodic task models and showed how to compute frequency assignments (which translate to task-level energy allocations) to maximize the overall reliability, while satisfying a hard energy constraint. By assuming a frequency value that can vary from a lower bound to an upper bound, we first presented our static optimal scheme. Then, we extended our framework with several online (dynamic) schemes that exploit early task completions and speed up task executions at runtime, to improve overall reliability. All our static and dynamic solutions guarantee both timing and energy budget constraints in all execution scenarios. Moreover, the experimental results indicate that our algorithms perform comparably to a clairvoyant optimal scheduler that can maximize overall reliability thanks to its advance knowledge about the exact workload. To the best of our knowledge, this problem has not been addressed in the research literature in the past.

REFERENCES

[1] *Intel Xscale Microarchitecture*, [Online]. Available: http://developer.intel.com/design/intelxscale/
[2] T. AlEnawy and H. Aydin, "On energy-constrained real-time scheduling," in *Proc. 16th Euromicro Conf. Real-Time Syst. (ECRTS'04)*, Catania, Sicily, Italy, Jun. 2004.
[3] T. AlEnawy and H. Aydin, "Energy-constrained scheduling for weakly-hard real-time systems," in *Proc. 26th IEEE Real-Time Syst. Symp. (RTSS'05)*, Miami, FL, Dec. 2005.
[4] H. Aydin, V. Devadas, and D. Zhu, "System-level energy management for periodic real-time tasks," in *Proc. 27th IEEE Real-Time Syst. Symp. (RTSS'06)*, Rio de Janeiro, Brazil, Dec. 2006.
[5] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez, "Dynamic and aggressive power-aware scheduling techniques for real-time systems," in *Proc. 22th IEEE Real-Time Syst. Symp. (RTSS'01)*, London, U.K., Dec. 2001.
[6] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Trans. Computers*, vol. 53, no. 10, pp. 584–600, May 2004.
[7] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. New York: Wiley, 2005.
[8] T. Burd and R. Brodersen, "Energy efficient CMOS microprocessor design," in *Proc. 28th Hawaii Int. Conf. Syst. Sciences. (HICSS'95)*, Kihei, Maui, HI, Jan. 1995.
[9] X. Castillo, S. McConnel, and D. Siewiorek, "Derivation and calibration of a transient error reliability model," *IEEE Trans. Computers*, vol. 31, no. 7, pp. 658–671, Jul. 1982.
[10] J.-J. Chen and T.-W. Kuo, "Voltage-scaling scheduling for periodic real-time tasks in reward maximization," in *Proc. 26th IEEE Real-Time Syst. Symp. (RTSS'05)*, Miami, FL, Dec. 2005.
[11] H. Cheng and S. Goddard, "SYS-EDF: A system-wide energy efficient scheduling algorithm for hard real-time systems," *Int. J. Embedded Syst. Low Power Real-Time Embedded Comput.*, vol. 4, no. 4, pp. 141–151, 2009.
[12] F. Dabiri, N. Amini, M. Rofouei, and M. Sarrafzadeh, "Reliability-aware optimization for DVS-enabled real-time embedded systems," in *Proc. Int. Symp. Quality of Electronic Design (ISQED'08)*, San Jose, CA, Mar. 2008.
[13] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner, "Razor: Circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, no. 6, pp. 10–20, Nov. –Dec. 2004.
[14] X. Fan, C. Ellis, and A. Lebeck, "The synergy between power-aware memory systems and processor voltage," in *Proc. Workshop on Power-Aware Comput. Syst. (PACS'03)*, San Diego, CA, Dec. 2003.
[15] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," *IEEE Trans. Nuclear Sci.*, vol. 47, no. 6, pp. 2586–2594, Dec. 2000.
[16] R. K. Iyer and D. J. Rossetti, "A measurement-based model for workload dependence of CPU errors," *IEEE Trans. Computers*, vol. 35, no. 6, pp. 511–519, June 1986.
[17] R. K. Iyer, D. J. Rossetti, and M. Hsueh, "Measurement and modeling of computer reliability as affected by system activity," *ACM Trans. Comput. Syst.*, vol. 4, no. 3, pp. 214–237, Aug. 1986.

[18] V. Izosimov, P. Pop, P. Eles, and Z. Peng, "Design optimization of time-and cost-constrained fault-tolerant distributed embedded systems," in *Proc. Design, Autom. Test in Europe (DATE'05)*, Munich, Germany, Mar. 2005.
[19] S. Kartik and C. S. R. Murthy, "Task allocation algorithms for maximizing reliability of distributed computing systems," *IEEE Trans. Computers*, vol. 46, no. 6, pp. 719–724, June 1997.
[20] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in hard real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.
[21] J. Liu, P. Chou, N. Bagherzadeh, and F. Kurdahi, "Power-aware scheduling under timing constraints for mission-critical embedded systems," in *Proc. ACM/IEEE Design Autom. Conf. (DAC'01)*, Las Vegas, NV, Jul. 2001.
[22] J. W. S. Liu, *Real-Time Systems*. Upper Saddle River, NJ: Prentice-Hall, 2000.
[23] C. D. Locke, D. R. Vogel, and T. J. Mesler, "Building a predictable avionics platform in ADA: A case study," in *Proc. IEEE Real-Time Syst. Symp. (RTSS'91)*, San Antonio, TX, Dec. 1991.
[24] R. Melhem, D. Mossé, and E. Elnozahy, "The interplay of power management and fault recovery in real-time systems," *IEEE Trans. Computers*, vol. 53, no. 2, pp. 217–231, Feb. 2004.
[25] B. Mochocki, X. S. Hu, and G. Quan, "Transition-overhead-aware voltage scheduling for fixed-priority real-time systems," *ACM Trans. Design Autom. Electron. Syst. (TODAES)*, vol. 12, no. 2, pp. 1–26, Apr. 2007.
[26] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic power-aware scheduling for real-time applications," in *Proc. 2000 Int. Symp. Low Power Electron. Design (ISLPLED'00)*, Rapallo, Italy, Jul. 2000.
[27] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low power embedded operating systems," in *Proc. ACM Symp. Oper. Syst. Principles (SOSP'01)*, Chateau Lake Louise, Banff, Canada, Oct. 2001.
[28] P. Pop, K. H. Poulsen, V. Izosimov, and P. Eles, "Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems," in *Proc. Int. Conf. Hardware/Software Codesign and System Synthesis* (CODES + ISSS'07), Salzburg, Austria, Sep. 2007.
[29] C. Rusu, R. Melhem, and D. Mosse, "Maximizing rewards for real-time applications with energy constraints," *ACM Trans. Embedded Comput. Syst.*, vol. 2, no. 4, pp. 537–559, Nov. 2003.
[30] S. Saewong and R. Rajkumar, "Practical voltage scaling for fixed priority RT-systems," in *Proc. 9th IEEE Real-Time and Embedded Technol. Appl. Symp. (RTAS'03)*, Toronto, Canada, May 2003.
[31] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "Energy-efficient mapping and scheduling for DVS enabled distributed embedded systems," in *Proc. Design, Autom. Test in Europe (DATE'02)*, Paris, France, Mar. 2002.
[32] R. Sridharan, N. Gupta, and R. Mahapatra, "Feedback-controlled reliability-aware power management for real-time embedded systems," in *Proc. ACM/IEEE Design Autom. Conf. (DAC'08)*, Anaheim, CA, Jun. 2008.
[33] S. Srinivasan and N. Jha, "Safety and reliability driven task allocation in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 3, pp. 238–251, Mar. 1999.
[34] T. Wei, P. Mishra, K. Wu, and H. Liang, "Online task-scheduling for fault-tolerant low-energy real-time systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD'06)*, San Jose, CA, Nov. 2006.
[35] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proc. 1st USENIX Conf. Oper. Syst. Design and Implementation (OSDI'94)*, Monterey, CA, Nov. 1994.
[36] H. Wu, B. Ravindran, and E. Jensen, "Utility accrual real-time scheduling under the unimodal arbitrary arrival model with energy bounds," *IEEE Trans. Computers*, vol. 56, no. 10, pp. 1358–1371, Oct. 2007.
[37] R. Xu, D. Mosse, and R. Melhem, "Minimizing expected energy consumption in real-time systems through dynamic voltage scaling," *ACM Trans. Embedded Comput. Syst.*, vol. 25, no. 4, pp. 1–40, Dec. 2007.
[38] F. Zhang and S. Chanson, "Throughput and value maximization in wireless packet scheduling under energy and time constraints," in *Proc. 24th IEEE Real-Time Syst. Symp. (RTSS'03)*, Cancun, Mexico, Dec. 2003.
[39] Y. Zhang and K. Chakrabarty, "Energy-aware adaptive checkpointing in embedded real-time systems," in *Proc. Design, Autom. Test in Europe (DATE'03)*, Munich, Germany, Mar. 2003.
[40] D. Zhu, "Reliability-aware dynamic energy management in dependable embedded real-time systems," in *Proc. IEEE Real-Time and Embedded Technol. Appl. Symp. (RTAS'06)*, San Jose, CA, Apr. 2006.
[41] D. Zhu and H. Aydin, "Reliability-aware energy management for periodic real-time tasks," *IEEE Trans. Computers*, vol. 58, no. 10, pp. 1382–1397, Oct. 2009.
[42] D. Zhu and H. Aydin, "Energy management for real-time embedded systems with reliability requirements," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD'06)*, San Jose, CA, Nov. 2006.
[43] D. Zhu, R. Melhem, and D. Mossé, "The effects of energy management on reliability in real-time embedded systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD'04)*, San Jose, CA, Nov. 2004.
[44] J. Ziegler, Trends in Electronic Reliability Effects of Terrestrial Cosmic Rays. [Online]. Available: http://www.srim.org/SER/SERTrends.htm 2004

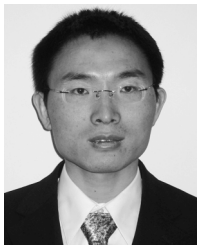
Baoxian Zhao received the B.Sc. and M.Sc. degrees in computer engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2002 and 2005, respectively. He is currently working towards the Ph.D. degree at the Department of Computer Science, George Mason University, Fairfax, VA.

His research interests include fault tolerance and low-power computing in real-time embedded systems.

Hakan Aydin (M'03) received the B.S. and M.S. degrees in control and computer engineering from Istanbul Technical University, Istanbul, Turkey, in 1991 and 1994, respectively, and the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, in 2001.

He is currently an Associate Professor in the Computer Science Department, George Mason University. His research interests include real-time systems, low-power computing, and fault tolerance.

Prof. Aydin was a recipient of the U.S. National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award in 2006. He has served on the program committees of several conferences and workshops, including the IEEE Real-Time Systems Symposium and the IEEE Real-Time Technology and Applications Symposium.



Dakai Zhu (M'04) received the B.E. degree in computer science and engineering from Xi'an Jiaotong University, Xian, China, in 1996, the M.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 1999, and the M.S. and Ph.D. degrees in computer science from the University of Pittsburgh, Pittsburgh, PA, in 2001 and 2004, respectively.

He joined the University of Texas at San Antonio as an Assistant Professor in 2005. His research interests include real-time systems, power aware computing and fault-tolerant systems. He has served on program committees (PCs) for several major IEEE- and ACM-sponsored real-time conferences (e.g., RTAS and RTSS).

Prof. Zhu is a member of the IEEE Computer Society.