Shared Recovery for Energy Efficiency and Reliability Enhancements in Real-Time Applications with Precedence Constraints

BAOXIAN ZHAO and HAKAN AYDIN, George Mason University
DAKAI ZHU, University of Texas at San Antonio

While Dynamic Voltage Scaling (DVS) remains as a popular energy management technique for modern computing systems, recent research has identified significant and negative impacts of voltage scaling on system reliability. To preserve system reliability under DVS settings, a number of *reliability-aware power management* (RA-PM) schemes have been recently studied. However, the existing RA-PM schemes normally schedule a separate recovery for each task whose execution is scaled down and are rather conservative. To overcome such conservativeness, we study in this article novel RA-PM schemes based on the *shared recovery* (SHR) technique. Specifically, we consider a set of frame-based real-time tasks with individual deadlines and a common period where the precedence constraints are represented by a *directed acyclic graph* (DAG). We first show that the *earliest deadline first* (EDF) algorithm can always yield a schedule where all timing and precedence constraints are met by considering the effective deadlines of tasks derived from *as late as possible* (ALAP) policy, provided that the task set is feasible. Then, we propose a shared recovery based frequency assignment technique (namely SHR-DAG) and prove its optimality to minimize energy consumption while preserving the system reliability. To exploit additional slack that arises from early completion of tasks, we also study a dynamic extension for SHR-DAG to improve energy efficiency and system reliability at runtime. The results from our extensive simulations show that, compared to the existing RA-PM schemes, SHR-DAG can achieve up to 35% energy savings, which is very close to the maximum achievable energy savings. More interestingly, our extensive evaluation also indicates that the new schemes offer non-trivial improvements on system reliability over the existing RA-PM schemes as well.

Categories and Subject Descriptors: H.4.1 [**Operating Systems**]: Process Management—*Scheduling*; D.4.7 [**Operating Systems**]: Organization and Design—*Real-time systems and embedded systems*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: DVS, reliability-aware power management, real-time systems

ACM Reference Format:
Zhao, B., Aydin, H., and Zhu, D. 2013. Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints. ACM Trans. Des. Autom. Electron. Syst. 18, 2, Article 23 (March 2013), 21 pages.
DOI: http://dx.doi.org/10.1145/2442087.2442094

1. INTRODUCTION

Energy management has become an important research area in the past decade due to dramatic increase in power consumption of modern computing systems and remains

This work was supported by US National Science Foundation awards CNS-1016855, CNS-1016974, and CAREER Awards CNS-0546244 and CNS-0953005.
Author's addresses: B. Zhao and H. Aydin, Computer Science Department, George Mason University, Fairfax, VA; email: bzhao@gmu.edu, aydin@cs.gmu.edu; D. Zhu, Computer Science Department, University of Texas at San Antonio, San Antonio, TX; email: dzhu@cs.utsa.edu.

as one of the grand challenges. The popular energy management technique *dynamic voltage scaling* (DVS) achieves energy savings by simultaneously scaling down processors' supply voltage and processing frequency at runtime [Weiser et al. 1994; Yao et al. 1995]. Based on DVS, many research studies have been recently conducted aiming at minimizing energy consumption while meeting applications' performance requirements, in particular for realtime embedded systems [Aydin et al. 2004; Pillai and Shin 2001; Jejurikar and Gupta 2004; Zhuo and Chakrabarti 2005; Quan and Hu 2007; Wang et al. 2010].

On the other hand, reliability and fault tolerance have been traditional research topics for computing systems and many fault tolerance solutions (based on different redundancy techniques) have been proposed [Pradhan 1996]. While there are a few studies focusing on providing fault tolerance capabilities to soft real-time systems (e.g., [Caccamo and Buttazzo 1998; Mejia-Alvarez et al. 2000; Aydin et al. 1999; Izosimov et al. 2010]), most research has focused on safety-critical hard real-time systems. In particular, to tolerate transient faults (which have been shown to be much more common than permanent faults [Castillo et al. 1982; Iyer et al. 1986]), one can explore temporal redundancy in a system and adopt *backward recovery* technique to reexecute faulty applications.

Although both fault tolerance and energy management have been extensively (but independently) studied, the co-management of system reliability and energy efficiency has caught researchers' attention only very recently [Melhem et al. 2004; Zhang and Chakrabarty 2003; Acharya and Mahapatra 2008; Wei et al. 2006; Liu et al. 2010]. Moreover, recent research also reported significant reliability degradation for systems that exploit DVS features, where the probability of failure for an application executed at low supply voltage and processing frequency levels can increase by several orders of magnitude [Ernst et al. 2004; Zhu et al. 2004]. To tackle such negative impacts of DVS on reliability, a number of *reliability-aware power management* (RA-PM) schemes have been studied for various real-time systems [Zhu 2006; Zhu and Aydin 2006, 2009]. The central idea of RA-PM is to reserve a portion of system slack for scheduling a recovery task before exploiting the remaining slack for DVS to save energy. In RA-PM, should an error caused by transient faults be detected at the end of task's scaled execution (through sanity checks or other error detection techniques [Pradhan 1996]), the recovery is invoked in the form of reexecution at the maximum clock frequency before the task's deadline to preserve the system reliability [Zhu 2006].

A common feature of the existing RA-PM schemes is that a separate recovery task is scheduled for each task that is executed at scaled supply voltage and processing frequency in order to preserve the reliability of all tasks. However, such recovery tasks may not be needed at the same time, especially for frame-based real-time tasks under non-preemptive scheduling. Thus, the existing individual-recovery based RA-PM schemes are rather conservative, in the sense that they reserve more slack for recovery and less slack for DVS with a negative impact on energy savings. To address such conservativeness, for a set of independent frame-based real-time tasks with a common deadline, a *shared-recovery* (SHR) based scheme was suggested in Zhao et al. [2009]. In the SHR scheme, all tasks whose executions are scaled through DVS can share a single recovery block.

In a number of applications (such as those found in feedback-based control applications), real-time tasks depend on each other for input/output data and thus have precedence constraints. In this work, by extending the basic idea of shared-recovery technique [Zhao et al. 2009], we develop SHR-based RA-PM schemes for a set of frame-based *dependent* real-time tasks with individual deadlines and a common period, where the precedence constraints are represented by a *directed acyclic graph* (DAG).

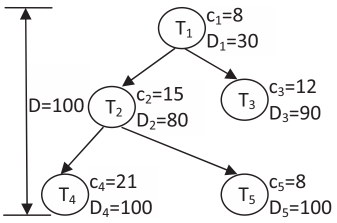


Fig. 1. An example application with five dependent tasks.

The main contributions of this research effort can be summarized as follows.

—First, a shared recovery (SHR) technique is proposed to address the conservativeness of the existing RA-PM schemes;
—Second, based on the derived effective deadlines of tasks represented by DAGs, the Earliest Deadline First (EDF) policy [Baruah et al. 1990] is shown to be a feasible scheduling algorithm;
—Third, an optimal shared-recovery based RA-PM scheme for DAG-represented tasks is studied, in order to minimize energy consumption while preserving system reliability;
—Fourth, the online shared-recovery based RA-PM scheme that exploits dynamic slack at runtime is developed to further improve energy efficiency and system reliability;
—Last, the proposed schemes are evaluated through extensive simulations.

The remainder of this article is organized as follows. Section 2 presents system models and states our assumptions, followed by a motivational example and problem description. The static and online shared-recovery based schemes for real-time tasks represented by DAGs are addressed and evaluated in Sections 3 and 4, respectively. Section 5 reviews the related work and Section 6 concludes the article.

2. SYSTEM MODELS

2.1. Application Model

In this work, we consider a real-time application consisting of n frame-based dependent tasks running on a single-processor system. The tasks are represented by a directed acyclic graph (DAG): $G = (V, E)$. The set of vertices (or nodes) in G represent the set of tasks $V = \{T_1, \ldots, T_n\}$. The set of edges $E = \{E_1, \ldots, E_m\}$ represent the precedence constraints among tasks, where an edge $(T_i, T_j) \in E$ indicates that task T_j cannot start to execute until T_i has finished its execution [Buttazzo 2004].

Each task T_i is characterized by its *worst-case execution time* (WCET) c_i and relative deadline D_i. As we consider systems with variable processing frequencies, it is assumed that the WCET of task T_i corresponds to execution time under the maximum processing frequency f_{max} of the system. Moreover, we assume normalized processing frequencies where f_{max} is assumed to be 1. When task T_i is executed at a lower frequency $f_i (< f_{max})$, its execution time is assumed to scale linearly and hence becomes $\frac{c_i}{f_i}$. The common period for all tasks (i.e., the frame of the application) is denoted by D. That is, the application that consists of n dependent tasks is reexecuted in every period of D. We assume $D_i \le D$ $(i = 1, \ldots, n)$.

As an example, Figure 1 shows the DAG for an application with five dependent tasks, where each task is annotated with its WCET c_i and deadline D_i (in milliseconds). The common period (i.e., the application's frame length) is $D = 100$ms.

2.2. Power and Energy Model

Considering the almost linear relationship between the supply voltage and operating frequency, DVS scales down the supply voltage alongside the processing frequency to save energy. In what follows, when there is no ambiguity, we use the term *frequency change* to stand for changing the supply voltage and processing frequency simultaneously. Assuming a DVS-capable system, we adopt the simple system-level power model proposed in Zhu et al. [2004] and subsequently used in prior RA-PM research [Zhu 2006; Zhao et al. 2008; Zhu and Aydin 2009], where the system power consumption at frequency f is given by:

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

Here P_s is the static power, which includes the power to maintain basic circuits, and keep the clock running and the memory in sleep modes. It can be removed only by powering off the whole system. The active power consist of two parts: the frequency-independent active power $P_{ind}$and the frequency-dependent active power P_d. P_{ind} includes the power consumed by off-chip devices (such as main memory and external devices) that can be efficiently removed by putting systems into sleep states. In addition to the dynamic power of CPU, P_d includes the power consumed by any devices that depends on processing frequencies. $\hbar$ represents system states and indicates whether active powers are currently consumed in the system. When the system is active, $\hbar = 1$; otherwise, $\hbar = 0$. 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.

Since there exists an excessive overhead associated with turning on/off a system [Elnozahy et al. 2002], we assume that the system is always on while the periodic real-time application is running. That is, P_s is always consumed and not manageable. Thus, we concentrate on managing the active power (P_{ind} and P_d) in this work. Note that, although DVS can reduce energy consumption due to reduced frequency-dependent active power P_d at lower processing frequencies, it will take longer for an application to finish its execution and thus may consume more energy due to frequency-independent active power P_{ind}. Thus, it may not be always energy efficient to execute applications at lower frequencies and it has been shown that there exists a *minimum energy-efficient* frequency [Jejurikar and Gupta 2004; Zhuo and Chakrabarti 2005], which can be found as [Zhu and Aydin 2006]:

$$f_{ee} = \sqrt[m]{\frac{P_{ind}}{(m-1)C_{ef}}}. \tag{2}$$

Assuming the operating frequency of the system under consideration can vary from a minimum available frequency f_{min} to the maximum frequency f_{max}, the lowest frequency to execute a task ia $f_{low} = \max\{f_{min}, f_{ee}\}$. For cases where the frequency-independent active power P_{ind} is excessive and f_{ee} exceeds f_{max}, operating the system at f_{max} would be the most energy efficient option and no DVS is needed [Aydin et al. 2006]. For systems considered in this article, we assume that $f_{ee} \leq f_{max}$.

2.3. Fault and Recovery Models

During the execution of an application, a fault may occur due to various reasons, such as hardware failure, software errors and the effect of electromagnetic interference and cosmic ray radiations. Since transient faults occur much more frequently than permanent faults [Castillo et al. 1982; Iyer et al. 1986], in this article, we focus on transient faults, especially the ones caused by cosmic ray radiations and electromagnetic interference. More specifically, we focus on the transient faults at the hardware level and assume that software does not have any design faults. Hence, errors detected at the task level are considered as manifestation of transient hardware faults.

Traditionally, transient faults have been modeled by Poisson distributions [Zhang and Chakrabarty 2003]. However, considering the effects of voltage scaling on transient faults [Ernst et al. 2004; Zhu et al. 2004], the average occurrence rate λ of soft errors caused by transient faults will depend on the system supply voltage (and thus processing frequency). In particular, it is known that the likelihood of creating a critical charge (which may cause soft errors at the hardware level) increases drastically at low voltage levels. In our analysis and simulations, we focus on the exponential rate model which was developed by using historical data and first principles in Zhu et al. [2004]. Other alternative models with similar exponential fault rate characterizations have been suggested [Chandra and Aitken 2008; Dabiri et al. 2008], but we adopt the the model in [Zhu et al. 2004] which has been widely widely used in prior RA-PM research [Ejlali et al. 2005; Pop et al. 2007; Zhu 2006; Zhu and Aydin 2006, 2009]:

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

where λ_0 is the average error rate corresponding to the maximum frequency $f_{max} = 1$ (and supply voltage V_{max}). The exponent d (> 0) is a constant that indicates the sensitivity of error rates to voltage scaling. That is, reducing the supply voltage and frequency for energy savings can result in exponentially increased error rates. At the minimum frequency f_{min} (and the supply voltage V_{min}), the maximum error rate is $\lambda_{max} = \lambda_0 \cdot 10^d$.

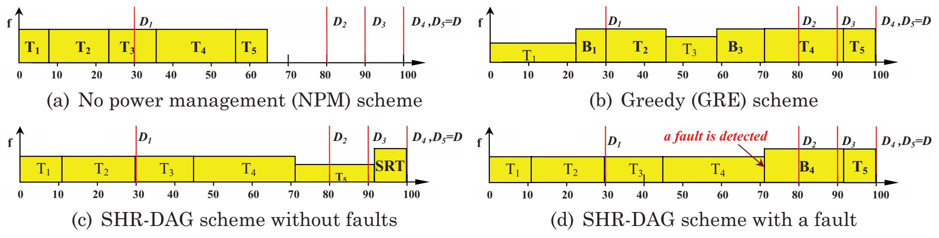
Therefore, based on the Poisson distribution, the *reliability* of a task T_i executed at frequency f_i, defined as the probability of completing its execution without encountering errors triggered by transient faults, can be found as $R_i(f_i) = e^{-\lambda(f_i)\cdot\frac{c_i}{f_i}}$ [Zhu et al. 2004], where $\lambda(f_i)$ is given as in Equation (3). The *original* reliability of task T_i is defined as the reliability level obtained when executed at f_{max} (without voltage scaling) and is given as $R_i^0 = R_i(f_{max}) = e^{-\lambda_0\cdot c_i}$ [Zhu et al. 2004]. As the correctness of an application normally depends on the successful execution of all its tasks, the original system reliability is defined as $R_{sys}^0 = \prod_{i=1}^{n} R_i^0$ [Zhu 2006].

When a task is executed at a low frequency level, its reliability can degrade significantly without special provisions. In settings where soft errors caused by transient faults can be detected (for example, through sanity checks [Pradhan 1996]) at the end of a task's execution, we can exploit temporal redundancy (i.e., system slack) and *backward recovery* technique to enhance system reliability. As in the existing RA-PM schemes [Ejlali et al. 2005; Pop et al. 2007; Zhu and Aydin 2006, 2009], we assume that the backward recovery takes the form of reexecution of the faulty task at f_{max} to preserve its original reliability [Zhu 2006].

Problem Description. Taking the negative effects of DVS on system reliability into consideration, for frame-based dependent real-time tasks with individual deadlines and a common period that are represented by a DAG, the problem to be addressed in this article is to: find the execution order of tasks and their frequency assignment to minimize system energy consumption while preserving the system original reliability without violating tasks' deadline and precedence constraints. Before presenting the details of our new shared-recovery based RA-PM schemes, in what follows, we first illustrate the conservativeness of the existing individual-recovery based RA-PM schemes through a concrete example.

2.4. A Motivational Example

For the example application with five tasks shown in Figure 1, a feasible schedule when all tasks run at f_{max}, that is, with no power management (NPM), is given in



(a) No power management (NPM) scheme (b) Greedy (GRE) scheme

(c) SHR-DAG scheme without faults (d) SHR-DAG scheme with a fault

Fig. 2. Schedules for the motivational example under different schemes.

Figure 2(a). Tasks are executed in increasing order of their indices and the schedule satisfies the tasks' individual deadlines and precedence constraints. Assuming that $\lambda_0 = 10^{-6}$, $d = 5$ and $f_{min} = 0.1$ [Zhu 2006], the *original* system reliability can be evaluated as $R^0_{sys} = 99.9999936\%$.

From Figure 2(a), we can see that the system has access to $100 - 64 = 36$ (ms) of slack, which can be utilized to scale down the execution of tasks to save energy. Next, consider the application of the *Greedy* (GRE) RA-PM scheme [Zhu and Aydin 2006] where two tasks T_1 and T_3 are chosen for management (Figure 2(b)). Two separate recovery tasks (denoted by $\{B_i\}$) are statically scheduled, which leaves 16ms slack for scaling down the execution of T_1 and T_3. The remaining tasks execute at f_{max} and do not need recovery tasks since they preserve their original reliability by definition. It can be found out that the energy consumption under the GRE schedule is $0.84 \cdot E$, where E is the energy consumed by all tasks under NPM. Moreover, since both scaled tasks T_1 and T_3 have their own recovery tasks, the reliability levels of all tasks will be equal to or higher than their original reliability level under NPM [Zhu 2006]. The system reliability, which is the product of the individual task reliabilities, can be found as $R^{gre}_{sys} = 99.9999956\%$, which is slightly better than R^0_{sys} and thus the GRE scheme preserves the system's original reliability as expected [Zhu and Aydin 2006].

Observe that, for frame-based real-time tasks that are executed nonpreemptively one after another within each frame, the multiple recovery tasks (e.g., B_1 and B_3 in the given example) will not be invoked simultaneously. Thus, scheduling a separate recovery task for each task whose execution is scaled down is a conservative approach that over-provisions for reliability and limits the amount of slack for DVS and energy savings. In the *shared recovery* (SHR) scheme proposed in Zhao et al. [2009], a single recovery block can be shared among tasks in order to reexecute any faulty scaled task at runtime [Zhao et al. 2009]. Under the SHR-based RA-PM schemes the system will typically have access to more slack for DVS and yield more energy savings.

Consider extending the SHR scheme and applying to our task set with precedence constraints and individual deadlines. This technique is denoted by SHR-DAG. For instance, we can have one recovery task (i.e., *SRT*) of size 8ms at the very beginning and the remaining slack can be used to scaled down all tasks. The recovery task is large enough to recover task T_1 (at f_{max}) in case it encounters soft errors during its scaled execution. When the scaled execution of T_1 completes successfully, we can adjust the recovery task size to be 15ms, which will be large enough to reexecute T_2 by borrowing slack from the following tasks and so on. For the case where the scaled execution of all tasks complete successfully, the resulting schedule is shown in Figure 2(c). It can be found out the energy consumption of all tasks under the SHR-DAG scheme is $0.56 \cdot E$, which is an improvement of 33% over that of the existing GRE scheme.

Observe that, as long as there are no errors caused by transient faults, each scaled task, at its dispatch time, will have access to a recovery block that is large enough for reexecution under the new SHR-DAG scheme. Once soft errors are detected and the recovery block is used to reexecute the faulty task, the remaining tasks within the current frame may have to be executed in *contingency mode* at f_{max}. Otherwise, the system may not have enough slack to recover from further soft errors and the original reliability of remaining tasks cannot be preserved. With the contingency execution of remaining tasks until (and only until) the end of current frame, we will later prove that the system's original reliability can be preserved under the new SHR-DAG scheme by maintaining the original reliability of each and every task.

Figure 2(d) shows a scenario where the shared recovery block is used to reexecute the faulty task T_4 and the following task T_5 is executed at the maximum frequency f_{max}. We can further compute the achieved system reliability under SHR-DAG as $R_{sys}^{shr} = 99.99999999995\%$, which turns out to be even better than that of GRE. In fact, by optimizing the most common case (i.e., by covering all single-error scenarios), SHR-DAG is generally more effective than GRE (which, in our example, cannot recover from any single fault, but can recover from two separate errors in both T_1 and T_3, respectively). Our simulation results will further confirm this observation.

3. STATIC SHARED-RECOVERY RA-PM SCHEME FOR DEPENDENT TASKS

List scheduling is the most commonly used scheduling approach for dependent tasks represented by DAGs [Pop et al. 2007]. However, finding a feasible and optimal schedule of dependent tasks to maximize the opportunities for energy and reliability management is not trivial, especially for tasks with individual deadlines.

In this section, based on the effective deadlines of tasks (as defined in Section 3.1), we first show that the *Earliest Deadline First* (EDF) scheduling policy can always find a valid schedule if the tasks are feasible. Then, we propose the shared-recovery based RA-PM scheme for dependent tasks (denoted as SHR-DAG), which is proved to guarantee the system's original reliability. Next, for the valid EDF schedule (i.e., execution order) of tasks, the optimal frequency assignment for SHR-DAG to minimize the fault-free energy consumption of the tasks is discussed, followed by the evaluations of SHR-DAG through simulations.

3.1. Feasible Schedule and EDF Scheduling

In the existence of precedence constraints, a task may have to complete well before its deadline to ensure that all its successor tasks can finish in time. Therefore, based on the *as late as possible* (ALAP) policy [Kung et al. 1985], we can define the *effective deadline* of a task T_i as follows:

$$D_i^e = \begin{cases} D_i, & Succ(T_i) = \emptyset \\ \min\left\{D_i, D_j^e - c_j\right\}, & T_j \in Succ(T_i), \end{cases} \tag{4}$$

where $Succ(T_i)$ is the set of successor tasks of T_i. That is, if there exists an edge $(T_i, T_j) \in E$, we have $T_j \in Succ(T_i)$.

THEOREM 3.1. *If a set of frame-based tasks with individual deadlines represented by a DAG are feasible then the EDF scheduling policy, when invoked with the tasks' effective deadlines, can meet all the timing and precedence constraints of tasks.*

PROOF. We prove this theorem in two parts: First, from Equation (4), we can see that the effective deadline of each task is the latest time by which a task has to complete its execution in a valid schedule. Therefore, if the tasks under consideration are feasible and there exists a valid schedule, all tasks will complete their executions before their effective deadlines. Therefore, replacing tasks' original deadlines with their effective deadlines does not hurt the feasibility of the task set.

Second, we show that the schedule obtained by the EDF scheduling based on tasks' effective deadlines is valid if there exists a valid schedule for the original problem. Note that, if tasks are executed under EDF and their effective deadlines, the precedence constraints among tasks are automatically satisfied (as a task always has a smaller effective deadline than its successor tasks). Thus, based on their effective deadlines, we can treat the tasks as independent tasks under EDF scheduling. For independent tasks, it is well-known that the schedule under EDF scheduling is feasible if there exists a feasible schedule for the tasks [Buttazzo 2004], which concludes the proof. □

3.2. Shared-Recovery for Reliability Preservation of Dependent Tasks (SHR-DAG)

As opposed to scheduling multiple separate recovery tasks statically as in the conventional RA-PM schemes [Zhu and Aydin 2006, 2009], the central idea of the *shared-recovery* technique is to let scaled tasks share a single recovery block and thus use more slack for DVS to improve energy savings [Zhao et al. 2009]. Moreover, once the execution order of tasks is known in a valid schedule, the size of the required recovery block at any given time depends only on the currently running task. Hence, in contrast to the original SHR scheme [Zhao et al. 2009], where the size of the recovery block is determined by the size of the largest task, our new SHR-DAG scheme varies the recovery block size dynamically for improved energy savings through better slack usage.

Without loss of generality, suppose that the execution order of tasks in a valid schedule is $T_1, T_2, \ldots, T_n$. If every task T_a before task T_i $(a < i)$ is executed at the scaled frequency f_a without incurring soft errors caused by transient faults, we will need to reserve a recovery block of size c_i before dispatching task T_i at the scaled frequency f_i, which has to satisfy the following time constraint:

$$\sum_{a=1}^{i} \frac{c_a}{f_a} + c_i \leq D_i^e, \tag{5}$$

where D_i^e is the effective deadline of task T_i. Moreover, in case T_i encounters a soft error, its recovery as well as all remaining tasks have to finish in time at the maximum frequency $f_{max} = 1$. That is, we need to have:

$$\sum_{a=1}^{i} \frac{c_a}{f_a} + \sum_{x=i}^{k} c_x \leq D_x^e, \ \forall\, k:\ (i+1) \leq k \leq n. \tag{6}$$

Hence, for feasibility, the scaled frequency f_i for task T_i has to satisfy $\sum_{a=1}^{i} \frac{c_a}{f_a} \leq b_i$, where $b_i = min\{D_k^e - \sum_{x=i}^{k} c_x | i \leq k \leq n\}$. Therefore, the optimization problem, denoted as *SHR-DAG*, can be formally expressed as: find the frequency assignments $\{f_1, \ldots, f_n\}$ to minimize

$$E_{total} = \sum_{i=1}^{n} E_i(f_i) = \sum_{i=1}^{n} \left(P_{ind} + C_{ef} f_i^m\right) \frac{c_i}{f_i}. \tag{7}$$

Subject to

$$\sum_{a=1}^{i} \frac{c_a}{f_a} \leq b_i \ (\forall\, i:\ 1 \leq i \leq n). \tag{8}$$

$$f_{low} \leq f_i \leq f_{max} \ (\forall\, i:\ 1 \leq i \leq n) \tag{9}$$

Recall that $f_{low} = \max\{f_{min}, f_{ee}\}$. Once we find out the valid frequency assignment, the online operation is the same as the SHR scheme [Zhao et al. 2009]: as long as there are no errors, tasks can be executed at their scaled frequencies. Once a soft error is detected at runtime, the recovery of the faulty task is immediately dispatched in the form of reexecution and the system switches to a *contingency mode* where all remaining tasks within the current frame are executed at f_{max}.

In Zhu [2006], it is shown that the overall reliability of a task T_i that is executed at a scaled frequency $f_i < f_{max}$, by also taking into consideration its recovery task, is given as:

$$R_i = R_i(f_i) + (1 - R_i(f_i)) \cdot R_i^0 > R_i^0, \tag{10}$$

where the first term on the right side is the probability of the scaled execution of the primary task completing successfully, and the second term captures the probability of the recovery executing at f_{max} successfully when the primary task fails. That is, with the help of the recovery task, the task T_i's original reliability R_i^0 is preserved.

LEMMA 3.2. *For an application with a set of frame-based dependent tasks, the system's original reliability in every frame will be preserved under the SHR-DAG scheme.*

PROOF. We will prove the statement by showing that SHR-DAG preserves the original reliability of every task, in each period. Consider the first task T_1 executed by SHR-DAG. If $f_1 = f_{max}$, its reliability is exactly $R_1^0 = R_1(f_{max})$, hence no reliability degradation occurs. Otherwise, if $f_1 < f_{max}$, SHR-DAG *must have assigned a recovery block with size of* c_i before its deadline and its reliability can be given by Equation (10), which is guaranteed to be greater than R_1^0.

For the second task T_2, if T_1 fails and the shared recovery has been used, T_2 and all remaining tasks within the current frame are executed at f_{max}, which again maintains their original reliability levels. Otherwise, if T_1 does not fail, the same argument can be repeated for T_2. Continuing in this way, we can show that the reliability of a task set under SHR-DAG is never worse than its original reliability while satisfying the deadline constraints. □

3.3. Optimal Frequency Assignments

In this section, for a given valid schedule with fixed execution order of tasks, we discuss how to obtain the optimal frequency assignments assuming all tasks take their WCETs.

We first define the following problem, denoted as the *SHR-DAG$_{part}$*, that ignores the frequency lower and upper bound constraints of the original *SHR-DAG* problem: find the frequency assignments $\{f_1, \ldots, f_n\}$ to minimize

$$E_{total} = \sum_{j=1}^{n} E_j(f_j) = \sum_{j=1}^{n} \left(P_{ind} + C_{ef} \cdot f_j^m\right) \frac{c_j}{f_j}. \tag{11}$$

Subject to

$$\sum_{a=1}^{j} \frac{c_a}{f_a} \le b_j \ (\forall\, j:\ 1 \le j \le n). \tag{12}$$

In order to solve the *SHR-DAG$_{part}$* problem, we can apply the necessary and sufficient Kuhn-Tucker conditions [Luenberger 1984] for optimality and then obtain the following expressions:

$$\left((m-1)C_{ef} \cdot f_j^{m-2} - \frac{P_{ind}}{f_j^2}\right) c_j - \sum_{a=j}^{n} \mu_a \frac{c_j}{f_j^2} = 0 \ (\forall\, j : 1 \le j \le n) \tag{13}$$

$$\mu_j\left(\sum_{a=1}^{j}\frac{c_a}{f_a}-b_j\right)=0\ (\forall\ j:1\le j\le n) \tag{14}$$

$$\mu_j\ge 0\ (\forall\ j:1\le j\le n), \tag{15}$$

where $\mu_1,\dots,\mu_n$ are Lagrange multipliers. Expression (13) can be further written as:

$$f_j=\left(\frac{P_{ind}+\sum_{a=j}^{n}\mu_a}{(m-1)C_{ef}}\right)^{\frac{1}{m}}\ (\forall\ j:1\le j\le n). \tag{16}$$

Based on expressions (14), (15) and (16), it is clear that if $\forall\ j:1\le j\le n,\ \mu_j=0$, then the optimal solution is $f_j=f_{ee}\ \forall\ j$ and that solution should satisfy the constraint set (12). Otherwise, there must exist at least one strictly positive μ_j value. In this case, in order to solve the *SHR-DAG$_{part}$* problem, we can use the iterative techniques to exploit the Kuhn-Tucker conditions, as in Aydin et al. [1999, 2006] to adjust the solution while guaranteeing the completion time constraints through the *interval intensity* principle introduced in [Yao et al. 1995].

Next, we consider the problem, called *SHR-DAG-L*, which adds the frequency lower bound constraints but still ignores the frequency upper bound constraints ($f_j\le f_{max}\ j=1,\dots,n$). Formally, we define the problem *SHR-DAG-L* as to minimize

$$E_{total}=\sum_{j=1}^{n}E_j(f_j) \tag{17}$$

Subject to

$$\sum_{a=1}^{j}\frac{c_a}{f_a}\le b_j\ (\forall\ j:\ 1\le j\le n) \tag{18}$$

$$f_{low}\le f_j\ (\forall\ j:\ 1\le j\le n). \tag{19}$$

LEMMA 3.3. *A set of frequency assignments $F=\{f_1,\dots,f_n\}$ is the solution to SHR-DAG-L if F is the solution to SHR-DAG$_{part}$ and satisfies the constraint set (19).*

PROOF. The necessary and sufficient Kuhn-Tucker conditions for the problem SHR-DAG-L include the constraints (18), (19), and:

$$E_j'(f_j)-\sum_{a=j}^{n}\mu_a\frac{c_j}{f_j^2}-\overline{\mu_j}=0\ (\forall\ j:1\le j\le n) \tag{20}$$

$$\mu_j\left(\sum_{a=1}^{j}\frac{c_a}{f_a}-b_j\right)=0\ (\forall\ j:1\le j\le n) \tag{21}$$

$$\overline{\mu_j}(f_{low}-f_j)=0\ (\forall\ j:1\le j\le n) \tag{22}$$

$$\mu_j\ge 0\ (\forall\ j:1\le j\le n) \tag{23}$$

$$\overline{\mu_j}\ge 0\ (\forall\ j:1\le j\le n), \tag{24}$$

where $\{\mu_j\}$ and $\{\overline{\mu_j}\}$ are the Lagrange multipliers. Then we can see that the condition given in the lemma coincides with the case where $\overline{\mu_j}=0$. Indeed, in that case, necessary and sufficient Kuhn-Tucker conditions for SHR-DAG-L become identical to that to SHR-DAG$_{part}$. Therefore, Lemma 3.3 holds. □

LEMMA 3.4. *If F violates the lower bound constraints given by (19), then, in the solution of SHR-DAG-L, $f_i = f_{low}$ $\forall i \in \Delta$, where $\Delta = \{i | \frac{f_{low}^2}{c_i \sum_{a=i}^n \mu_a} E_i'(f_{low}) \geq \frac{f_{low}^2}{c_j \sum_{a=j}^n \mu_a} E_j'(f_{low}) \forall j\}$.*

PROOF. If F violates the lower bound constraints, then due to (22), we can conclude that $\exists\ \overline{\mu_j} > 0$ such that $f_j = f_{low}$. We will prove that, under the condition specified in the lemma, the Lagrange multipliers $\overline{\mu_i} > 0$ for $\forall i \in \Delta$, which will imply (by (22)) that $f_i = f_{low}$ $\forall i \in \Delta$.

Assume $\exists\ m \in \Delta$ such that $\overline{\mu_m} = 0$. both the previous discussion, we know that $\exists\ \overline{\mu_j} > 0$ and $f_j = f_{low}$. Using (20), we can obtain $\frac{f_m^2}{c_m \sum_{a=m}^n \mu_a} E_m'(f_m) = \frac{f_{low}^2}{c_j \sum_{a=j}^n \mu_a} E_j'(f_{low}) - \mu_j \frac{f_{low}^2}{c_j \sum_{a=j}^n \mu_a}$.

Then we obtain $\frac{f_{low}^2}{c_j \sum_{a=j}^n \mu_a} E_j'(f_{low}) > \frac{f_m^2}{c_m \sum_{a=m}^n \mu_a} E_m'(f_m)$. Moreover, the function $\frac{f_m^2}{c_j \sum_{a=m}^n \mu_a} E_m'(f_m)$ is strictly increasing with increasing f_m. Observe that since $f_m \geq f_{low}$, hence, $\frac{f_{low}^2}{c_j \sum_{a=j}^n \mu_a} E_j'(f_{low}) > \frac{f_m^2}{c_m \sum_{a=m}^n \mu_a} E_m'(f_m) \geq \frac{f_{low}^2}{c_m \sum_{a=m}^n \mu_a} E_m'(f_{low})$, which contradicts the assumption that $m \in \Delta$. □

LEMMA 3.5. *If the solution of SHR-DAG-L violates upper bound constraints given by (9), then, in the solution of SHR-DAG, $f_i = f_{max}$ $\forall i \in \Lambda$, where $\Lambda = \{i | \frac{f_{max}^2}{c_i \sum_{a=i}^n \mu_a} E_i'(f_{max}) \leq \frac{f_{max}^2}{c_j \sum_{a=j}^n \mu_a} E_j'(f_{max}) \forall j\}$*

PROOF. The proof is very similar to that of Lemma 3.4. First, we obtain the necessary and sufficient Kuhn-Tucker conditions for the problem *SHR-DAG*, which include (8), (9) and:

$$E_j'(f_j) - \sum_{a=j}^n \mu_a \frac{c_j}{f_j^2} - \overline{\mu_j} + \nu_j = 0\ (\forall\ j : 1 \leq j \leq n) \tag{25}$$

$$\mu_j \left(\sum_{a=1}^j \frac{c_a}{f_a} - b_j \right) = 0\ (\forall\ j : 1 \leq j \leq n) \tag{26}$$

$$\overline{\mu_j}(f_{low} - f_j) = 0\ (\forall\ j : 1 \leq j \leq n) \tag{27}$$

$$\nu_j(f_j - f_{max}) = 0\ (\forall\ j : 1 \leq j \leq n) \tag{28}$$

$$\mu_j \geq 0\ (\forall\ j : 1 \leq j \leq n) \tag{29}$$

$$\overline{\mu_j} \geq 0\ (\forall\ j : 1 \leq j \leq n) \tag{30}$$

$$\nu_j \geq 0\ (\forall\ j : 1 \leq j \leq n), \tag{31}$$

where μ_j, ν_j and $\overline{\mu_j}$ $(\forall j)$ are the Lagrange multipliers. Comparing the Kuhn-Tucker conditions for problems *SHR-DAG* and *SHR-DAG-L*, we notice that at least one ν_j should be definitely larger than zero, if the solution of *SHR-DAG-L* violates some upper bound constraints of *SHR-DAG*. This is because that when $\nu_j = 0$ $(\forall j)$, Equations (29) and (31) vanish and Equation (25) become identical to the Equation (20), which means that both sets of Kuhn-Tucker conditions (hence, the optimal solutions of two problems) coincide. But this contradicts the assumption that we gave. Similarly, if $\exists i$ $\nu_i > 0$, (which should be the case if the solution of *SHR-DAG-L* violates the constraints of *SHR-DAG*), then $\overline{\mu_j}$ should be zero: otherwise, from Equations (27) and (28) one would conclude that $f_i = f_{low} = f_{max}$, which is a contradiction.

In the rest of the proof we will show that, under the condition specified in the lemma, the Lagrange multipliers ν_i ($\forall i \in \Lambda$) are all nonzero, which will imply (by (28)) that $f_i = f_{max}$. Assume that $\exists m \in \Lambda$ such that $\nu_m = 0$ (implying $\overline{\mu_m} > 0$. From the previous discussion, we know that that must exist task T_j such that $\nu_j > 0$ (implying $\overline{\mu_j} = 0$) and $f_j = f_{max}$. Applying equation (25), we obtain: $\frac{f_m^2}{c_m \sum_{a=m}^{n} \mu_a} E'_m(f_m) - \overline{\mu_m} \frac{f_m^2}{c_m \sum_{a=m}^{n} \mu_a} = \frac{f_{max}^2}{c_j \sum_{a=j}^{n} \mu_a} E'_j(f_{max}) + \nu_j \frac{f_{max}^2}{c_j \sum_{a=j}^{n} \mu_a}$ giving $\frac{f_m^2}{c_m \sum_{a=m}^{n} \mu_a} E'_m(f_m) > \frac{f_{max}^2}{c_j \sum_{a=j}^{n} \mu_a} E'_j(f_{max})$. Since $f_m \le f_{max}$, then $\frac{f_{max}^2}{c_m \sum_{a=m}^{n} \mu_a} E'_m(f_{max}) \ge \frac{f_m^2}{c_m \sum_{a=m}^{n} \mu_a} E'_m(f_m)$. Finally we conclude that $\frac{f_{max}^2}{c_m \sum_{a=m}^{n} \mu_a} E'_m(f_{max}) > \frac{f_{max}^2}{c_j \sum_{a=j}^{n} \mu_a} E'_j(f_{max})$ which contradicts the assumption that $m \in \Lambda$. □

Before presenting the algorithm to find the optimal frequency assignments, we first give similar definitions and terms as used in Yao's algorithm [Yao et al. 1995].

Definition 3.6. The *intensity* of an interval $I = [z, z']$ is defined as

$$g(I) = \frac{\sum\limits_{T_i \in X_I} c_i}{z' - z} \quad \text{where} \quad X_I = \{T_j | z \le b_j \le z'\}, \tag{32}$$

which is the average frequency required to execute all the tasks whose b_i values satisfy $z \le b_i \le z'$ for a given interval $I = [z, z']$. Notice that the b_i value can be seen as the latest time point at which task T_i must complete in order to guuarantee the deadline and recovery task size constraints.

Algorithm 1 summarizes the basic steps to find the optimal frequency assignment for SHR-DAG. Here, we first compute the effective deadline and the value of b_j for each task T_j. The time complexity for this step with n tasks is $O(n^2)$. Steps from line 3 to 13 operate in a way similar to Yao's algorithm. Specifically, the new algorithm first computes the interval with maximum intensity g_{max}, and then sets the frequency of all the tasks whose b_i values are contained in that interval to g_{max}, and repeats the procedure for the remaining intervals. However, in order to satisfy the frequency bound constraints (f_{min} and f_{max}), the frequency obtained after invoking the Yao's algorithm must be adjusted in line 5. There are most n iterations before the task set Θ becomes empty. During each iteration, it takes at most n steps to compute the value of $g(I)$ (line 4). Therefore, the time complexity to perform from line 3 to 13 is $O(n^2)$ and the overall complexity of Algorithm 1 is $O(n^2)$.

3.4. Evaluations and Discussions

To evaluate the performance of the SHR-DAG scheme on both energy efficiency and system reliability, we designed and implemented a discrete-event simulator. For fair comparison, we also extended the GRE and SUEF algorithms [Zhu and Aydin 2006] for tasks with precedence constraints. These new extensions are called GRE-DAG and SUEF-DAG, respectively. We also implemented a static power management scheme in DAG settings, namely SPM-DAG, which focuses on minimizing energy consumption while not scheduling recovery tasks and hence ignoring reliability degradations. In fact, SPM-DAG is a special case of SHR-DAG that does not reserve any slack for recovery. In other words, SPM-DAG can be implemented by SHR-DAG algorithm by using D_j instead of b_j in Algorithm 1. Notice that SPM-DAG cannot preserve system reliability and is included here to show the maximum achievable energy savings. The complexity of these schemes is given in Table I.

Table I. The Complexity of the Schemes (n is the number of tasks)

Scheme	Complexity	Scheme	Complexity
GRE-DAG	$O(n)$	SUEF-DAG	$O(n \log n)$
SPM-DAG	$O(n^2)$	SHR-DAG	$O(n^2)$

ALGORITHM 1: Optimal Frequency Assignment for SHR-DAG

```
1: Under ALAP policy, compute the effective deadlines ($D_1^e, D_2^e, \ldots, D_n^e$) and
   $b_j = min\{D_k^e - \sum_{i=j}^{k} c_i (\forall\, k : \; j \le k \le n)\}$ values for all tasks.
2: Set $z = 0$, $\Theta = \{T_1, T_2, \ldots, T_n\}$;
3: while $\Theta$ is not empty do
4:    Identify $T_m \in \Theta$ such that $g(I = [z, z' = b_m]) = g_{max} \ge g(I' = [z, b_i]) \forall\; T_i \in \Theta$;
5:    Let $s = max\{f_{low}, min\{g_{max}, f_{max}\}\}$;
6:    if $s = f_{low}$ then
7:       set the frequency of tasks in $\Theta$ to $s$ and return;
8:    else
9:       set the frequency of tasks in $\Lambda = \{T_i | b_i \le b_m \wedge T_i \in \Theta\}$ to $s$;
10:      $\Theta = \Theta - \Lambda$;
11:      set $z = z + \sum \frac{c_i}{s} (\forall i : T_i \in \Lambda)$;
12:   end if
13: end while
```

In our simulations, we have $\lambda_0 = 10^{-6}$ for the lowest error rate, which is realistic according to historical data on transient faults [Hazucha and Svensson 2000; Ziegler 2004]. The processing frequency of the system can vary from $f_{min} = 0.1$ to $f_{max} = 1.0$. Furthermore, we assume $C_{ef} = 1$ and $m = 3$.The frequency-independent power P_{ind} for each task is normalized with respect to the maximum frequency-dependent power $P_d^{max} = 1$. All results are normalized with respect to those of the *no power management* (NPM) scheme, which executes all tasks at f_{max}.

The tasks are generated with the well-known TGFF tool [Dick et al. 1998]. Since task sets with different number of tasks yielded similar performance trends for all the schemes, we report only the results for cases with 10 tasks per task set. Within a task set, the worst-case execution time c_i for each task is randomly generated by the UUniFast algorithm [Bini and Buttazzo 2005] in the range of [10ms, 100ms]. Each result point in the figures is obtained by averaging the values obtained by three kinds of graph topologies: (1) *independent tasks*, without precedence constraints among any tasks; (2) *chain graph*, where the precedence constraints among tasks form a chain; and (3) *tree graph*. For each graph topology, the reported simulation results correspond to the values averaged over 1000 different task sets.

3.4.1. Evaluation of Energy Consumption. First,Figure 3 shows the impact of available slack on the system energy consumption. In these experiments, the exponent d for the error rate in Equation (3) is set to 2 and the frequency-independent power is $P_{ind} = 0.05$. The available slack is represented by $L = D - C$, where $C = \sum c_i$. Intuitively, the larger the slack, the more opportunities for DVS and energy savings. Therefore, all schemes can get lower energy consumption with increasing value of L. Moreover, compared to the existing RA-PM schemes (GRE-DAG and SUEF-DAG), SHR-DAG can obtain significantly more (up to 35%) energy savings. The reason is that, unlike SUEF-DAG and GRE-DAG that allocate separate recovery tasks statically, SHR-DAG reserves only a small amount of slack for the shared recovery and is able to allocate much more slack for DVS. Moreover, as more slack becomes available (e.g., when $L \ge 0.6 \cdot C$),

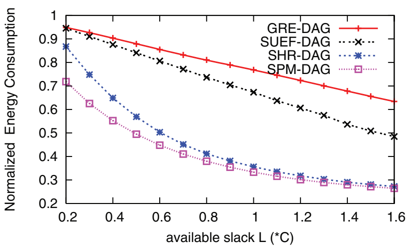


Fig. 3. The impact of system slack on system energy consumption.

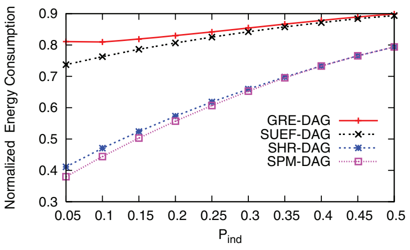
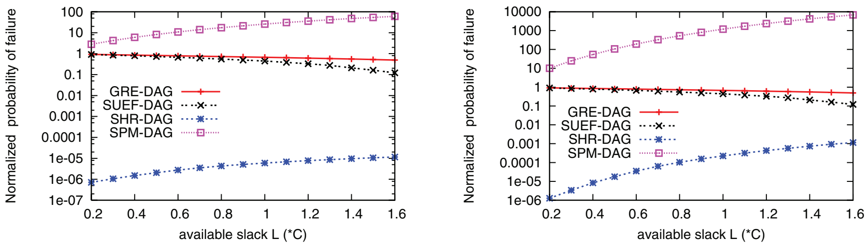


Fig. 4. The impact of P_{ind} on system energy consumption.



(a) normalized probability of failure for $d = 2$ (b) normalized probability of failure for $d = 5$

Fig. 5. The impact of slack on system reliability.

the performance of SHR-DAG becomes remarkably close to that of SPM-DAG, which stands for the upper bound for any DVS-based energy management algorithm.

Figure 4 shows the relationship between different frequency-independent active power P_{ind} and the system energy consumption when we set the available slack to $L = 0.8 \cdot C$. The minimum value of P_{ind} is set to 0.05. The energy consumption for all schemes increases with increased P_{ind}. This is because, with larger values of P_{ind}, the frequency-independent energy consumption increases and the threshold of energy-efficient frequency for tasks become higher, which limits the DVS opportunities (Section 2.2). Again, SHR-DAG outperforms GRE-DAG and SUEF-DAG. Further its energy performance is very close to the bound yielded by SPM-DAG.

3.4.2. Evaluation of System Reliability. Next, we evaluate the system reliability dimension for these schemes. For convenience, we present the *probability of failure (PoF)* (defined as $1-$*reliability*) achieved by all schemes. Again, all results are normalized with respect to NPM.Figures 5(a) and 5(b) show the reliability performance when the exponent d is 2 and 5, respectively. Here, P_{ind} is set to 0.05. Without special provision for reliability, SPM-DAG can lead to several orders of magnitude degradation for system reliability. As expected, all RA-PM schemes (SHR-DAG, GRE-DAG and SUEF-DAG) can preserve the system's original reliability.

More interestingly, the results point to a somewhat counterintuitive phenomena: SHR-DAG has also a clear advantage on the reliability side through the evaluated spectrum, despite the fact that it uses a shared recovery block for all tasks. The main reason comes from the fact that GRE-DAG and SUEF-DAG allocate separate recovery tasks statically. As a result, except for cases where the available slack is very large, only a subset of tasks can be managed by SUEF-DAG and GRE-DAG while the remaining tasks do not receive any recovery. On the other hand, SHR-DAG allocates a single recovery block that can be used by any task in case it fails. In essence, SHR-DAG provides a better protection for the single-fault scenarios that are typically much more likely than multiple-fault scenarios that SUEF-DAG and GRE-DAG provision for – these two schemes are unable to cover certain single-fault scenarios. When more slack is available, the *PoF* of SHR-DAG increases slightly as the system can use more aggressive voltage scaling. For the cases where the error rate is very sensitive to the voltage scaling (i.e., $d = 5$), the normalized *PoF* of SHR-DAG approaches (but never exceeds) 0.001.

4. ONLINE EXTENSION OF SHR-DAG

While the offline SHR-DAG scheme provides a powerful mechanism to save more energy while preserving system original reliability, some runtime optimizations can further improve its performance. Typically, real-time applications only consume a small fraction of their WCETs and abundant amount of dynamic slack can be available at runtime. In fact, many DVS frameworks [Aydin et al. 2004; Pillai and Shin 2001] have been proposed in the past to exploit the *dynamic slack* that arises from early completions of tasks for better energy savings. The same approach can be incorporated in SHR-DAG as well.

In this section, we consider an extension to our static SHR-DAG scheme. The on-line SHR-DAG algorithm, namely, DSHR-DAG, dynamically reclaims the slack at runtime to further scale down the processing frequency of tasks for additional energy savings. Specifically, DSHR-DAG operates as follows: initially, SHR-DAGis invoked to obtain the scaled frequency for the first running task. Then, at runtime, when a task completes early and/or without encountering errors, the frequency assignment for the next task will be updated by reinvoking SHR-DAG with the size of recovery block needed and the worst-case workload for the remaining tasks. The details of DSHR-DAG is given in Algorithm 2. At runtime, the system's original reliability can be guaranteed since we decide frequency assignments by iteratively invoking SHR-DAG, which can preserve system original reliability as shown in Lemma 3.2. With only the information about the worst-case workload in advance, applying SHR-DAG among remaining tasks for each step can maximize energy savings. In other words, at each scheduling point, we need to invoke the SHR-DAG algorithm. Hence the complexity of DSHR-DAG at each scheduling point is $O(j^2)$, where j is the total number of remaining tasks. A further observation is that, at runtime, we need to decide only the frequency assignment for the currently running task. That is, we can stop SHR-DAG algorithm when the frequency assignment for the current task is obtained (i.e., just a single iteration from line 4 to line 5 in Algorithm 1 with the complexity $O(j)$). Therefore, the complexity of DSHR-DAG at each scheduling point can be improved to $O(j)$. In the following simulations, we use the improved version of DSHR-DAG.

4.1. Evaluations and Discussions

In this section, we evaluate the performance DSHR-DAG. For comparison, we also implemented DGRE-DAG and DSUEF-DAG by extending GRE-DAG and DSUEF-DAG to dynamic settings, respectively. Moreover, in order to compute an absolute bound on achievable energy savings, we also implemented a *clairvoyant* scheme (denoted

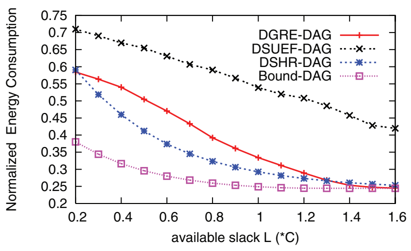


Fig. 6. The impact of slack on system energy consumption.

ALGORITHM 2: Online Frequency Assignment for DSHR-DAG

```
1: Under ALAP policy, obtain effective deadlines ($D_1, D_2, \ldots, D_n$) for all tasks and also
   compute $b_j = min\{D_k - \sum_{i=j}^{k} c_i (\forall\, k:\ j \le k \le n)\}$ for each task $T_j$.
2: for $i := 1$ to $n$ do
3:    Invoke SHR-DAG algorithm (Algorithm 1) for tasks from $T_i$ to $T_n$ to get the initial
      frequency assignments $\{f_i\}$;
4:    Execute $T_i$ at frequency $f_i$ and record its completion time $a_i$;
5:    if a fault is detected at $T_i$'s completion then
6:       Schedule a recovery of size $c_i$ at $f_{max}$ and update $a_i = a_i + c_i$;
7:    end if
8:    for $j := i + 1$ to $n$ do
9:       $b_j = b_j - a_i$; //update $b_j$ by considering the elapsed time
10:   end for
11: end for
```

as Bound-DAG) that uses the information about the actual workload of the tasks in advance.

The simulation settings are essentially the same as those in Section 3.4. In addition, to model the variations of the actual workload of tasks, we use the ratio $\frac{WCC}{BCC}$, which denotes the ratio of the worst-case execution time (WCC) to the best-case execution time (BCC). At runtime, the actual execution time of each task is determined in the range $[BCC, WCC]$ with the uniform probability distribution. Here, the higher ratio of $\frac{WCC}{BCC}$ indicates that the actual execution time of tasks deviates more from the worst-case and more dynamic slack can be expected at runtime. In the simulations, we set $d = 2$ and all results are normalized with respect to that of the NPM scheme.

4.1.1. Evaluation of Energy Consumption. Figure 6 first shows the effects of the available slack L on the system energy consumption, for $\frac{WCC}{BCC} = 3$ and $P_{ind} = 0.05$. As before, when more slack is available, more energy savings can be obtained. For most cases, DSHR-DAG has clear advantages over DGRE-DAG and DSUEF-DAG. However, when the available slack is limited (e.g., $L = 0.3 \cdot C$), the difference between the clairvoyant algorithm Bound-DAG and DSHR-DAG is more significant since Bound-DAG can utilize lower frequencies with the knowledge of tasks' actual execution time.

A trend to underline is the greatly enhanced performance of DGRE-DAG (compared to the worst-case settings). When $\frac{WCC}{BCC} = 3$, DGRE-DAG outperforms SUEF-DAG and its achieved energy savings are quite close to that of DSHR-DAG. The reason is that, many tasks complete much early at such high workload variation settings. As suggested by previous DVS research [Aydin et al. 2004; Pillai and Shin 2001], reusing the slack as early as possible typically pays off. However, DSUEF-DAG chooses to reallocate the slack according to slack usage efficiency factors, which can possibly exclude the

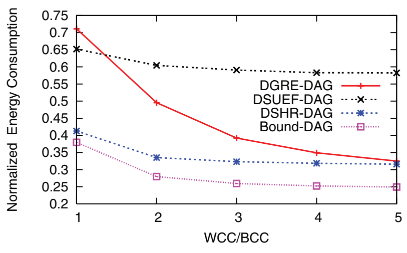


Fig. 7. The impact of $\frac{WCC}{BCC}$ ratio on system energy consumption.

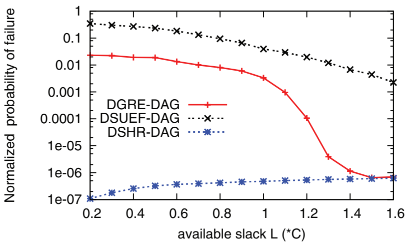


Fig. 8. The impact of the system slack on the probability of failure.

next tasks that would immediately benefit. When L is very large, DGRE-DAG, DSHR-DAG and Bound-DAG converge to almost the same level, which is determined by the energy-efficient frequency.

Figure 7 further illustrates how system energy consumptions changes for different variation of tasks' actual execution time workload (i.e., different ratios of $\frac{WCC}{BCC}$) with $L = 0.8 \cdot C$ and $P_{ind} = 0.05$. As expected, the system energy consumption generally decreases with decreasing actual workloads (i.e., large $\frac{WCC}{BCC}$ ratios). Among the three schemes, DSHR-DAG obtain the most energy savings, which is close to the yardstick algorithm Bound-DAG by at most a margin of 7%.

Interestingly, when $\frac{WCC}{BCC} = 1$, DGRE-DAG's performance quickly deteriorates. It is because, when the actual workload does not exhibit high variations, DSUEF-DAG assigns slack according to tasks' slack usage efficiency factors, which typically yields better results, compared to DGRE-DAG that make all available slack to the next task.

4.1.2. Evaluation of System Reliability. Next, Figure 8 shows the probability of failure as the function of the available slack with $P_{ind} = 0.05$ and $\frac{WCC}{BCC} = 5$. Although DSHR-DAG still performs the best, DGRE-DAG and DSUEF-DAG greatly benefit from dynamic slack reclamation, especially for large values of L. Essentially, DGRE-DAG and DSUEF-DAG can allocate more recovery tasks at runtime by recycling the slack of unused recoveries, which leads to smaller values of PoF. Another interesting observation is that, just like the energy dimension, DGRE-DAG appears to outperform DSUEF-DAG as it reassigns recovery tasks in a greedy fashion. Moreover, when $L \geq 1.5 \cdot C$, almost all tasks will be effectively executed at the energy-efficient frequencies by DGRE-DAG and DSHR-DAG, which gives similar PoF values.

Figure 9 illustrates how the system reliability changes for different variations of tasks' actual workload when $L = 0.8 \cdot C$ and $P_{ind} = 0.05$. In general, the probability of failure for DGRE-DAG and DSUEF-DAG decreases with lower actual workload (i.e., large $\frac{WCC}{BCC}$ ratios) since more slack becomes available due to early completions of tasks. Because of the greedy nature of DGRE-DAG (i.e., aggressively allocate all dynamic slack to the next task), the achieved system reliability under DGRE-DAG

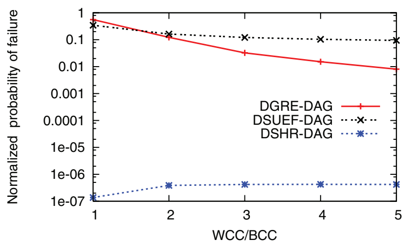


Fig. 9. The impact of $\frac{WCC}{BCC}$ ratio on the probability of failure.

tends to degrade with smaller values of $\frac{WCC}{BCC}$. When $\frac{WCC}{BCC} \geq 2$, DSHR-DAG can execute almost all tasks at the energy-efficient frequencies, which yields almost stable *PoF* values.

5. RELATED WORK

Simultaneous consideration of both energy consumption and fault tolerance has attracted researchers' attention in recent years. By deploying the primary/back-up model for independent periodic tasks, Unsal et al. proposed an energy-aware software-based fault tolerance scheme. The optimal number of checkpoints was investigated by Melhem et al. [2004] to tolerate faults and reduce energy consumption at runtime. Zhang et al. further adopted the checkpointing technique for soft periodic realtime tasks and developed an online fault tolerance algorithm in Zhang and Chakrabarty [2003]. 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 Wei et al. [2006], the authors proposed an efficient online scheme to minimize energy consumption through DVS policies, by considering the runtime behavior of tasks and fault occurrences. Focusing on tolerating a fixed number of faults, Liu et al. [2010] proposed a heuristic-based scheduling technique to minimize energy consumption for independent frame-based tasks. However, these existing research works [Melhem et al. 2004; Zhang and Chakrabarty 2003; Wei et al. 2006; Liu et al. 2010] focused on tolerating a fixed number of faults within the context of energy-aware real-time operation.

Despite the effectiveness of DVS to reduce energy consumption, it has been shown that DVS has a significant and negative effect on the system reliability [Zhu et al. 2004; Zhang and Chakrabarty 2003], primarily because of the significantly increased transient fault rates at low supply voltage and frequency levels. Hence, recently a number of research studies promoted the so-called Reliability-Aware Power Management (RA-PM) framework. The modeling of system reliability as a function of frequency/voltage assignment and a preliminary analysis of related trade-off dimensions has been conducted by Zhu et al. [2004]. Following the same line of research, RA-PM has been extended to multiple tasks with a common deadline [Zhu and Aydin 2006] and to periodic real-time tasks [Zhu and Aydin 2009]. To address the conservativeness of such an individual-recovery based approach, considering independent frame-based task model, a preliminary version of SHR technique for independent frame-based tasks was first proposed in Zhao et al. [2009]. Recently, SHR has been further extended to *generalized shared recovery (GSHR)* technique [Zhao et al. 2011] through which a small number of recovery tasks are shared by all the tasks while satisfying an arbitrary system-level target reliability.

Moreover, based on the fault models developed in [Zhu et al. 2004], Ejlali et al. [2001] studied a number of schemes that combine the information about hardware resources and temporal redundancy to save energy and to preserve system reliability. Considering dependent tasks represented by directed acyclic graphs (DAGs), Pop et al. proposed a novel framework by studying the energy and reliability trade-offs for distributed heterogeneous embedded systems [Pop et al. 2007]. Dabiri et al. [2008] studied the problem of assigning frequency/voltage to tasks for energy minimization subject to reliability and timing constraints. Recently, hardware redundancy technique [Ejlali et al. 2009] is deployed through an extra spare processor to preserve original system reliability where the catch-up scheme is developed to minimize system energy consumption. In order to achieve more energy savings in a standby sparing scheme.

For multiprocessor settings, the exact evaluation of the reliability may be quite complex due to the possibility of task replication and need for considering the schedule lengths on individual processors, even when voltage scaling is not used. A bicriteria scheduling heuristic based on Pareto optimization has been developed in Girault and Kalla [2009] in these settings. Benoit et al. showed later that computing the exact reliability of a given schedule with task replication on a parallel system is $\#P'-$Complete (hence at least as hard as NP-Complete problems).

6. CONCLUSIONS

The negative effects of DVS on system reliability due to increased transient faults at lower supply voltage and frequency promote the necessity of reliability-aware power management (RA-PM). However, the existing RA-PM schemes normally schedule a separate recovery task for any task whose execution is scaled down, which is rather conservative as the recovery tasks may not be needed simultaneously, especially for non-preemptive execution of frame-based real-time tasks.

In this article, focusing on a set of frame-based real-time tasks with precedence constraints that have individual deadlines and the common period, we proposed and evaluated a shared-recovery based RA-PM scheme, SHR-DAG. First, based on tasks' effective deadlines that are derived from precedence constraints of tasks following as late as possible (ALAP) policy, we show that EDF scheduling can always obtain a valid schedule provided that the task set is feasible. Based on the shared-recovery technique, where all scaled tasks can share a single recovery block, the SHR-DAG scheme is shown to be able to preserve the system reliability. For a given valid schedule with fixed execution order of tasks, the static optimal frequency assignment of SHR-DAG is presented and formally demonstrated. We further proposed an online extension of SHR-DAG aiming at better utilizing dynamic slack for more energy savings. Simulation results show that, compared to the existing individual-recovery based RA-PM schemes, SHR-DAG cannot only obtain more energy savings but also have a clear advantage on the reliability dimension. The energy savings under SHR-DAG are very close to the upper-bounds in both static and dynamic settings.

REFERENCES

Acharya, S. and Mahapatra, R. 2008. A dynamic slack management technique for real-time distributed embedded systems. *IEEE Trans. Comput. 57*, 2, 215–230.

Aydin, H., Devadas, V., and Zhu, D. 2006. System-level energy management for periodic real-time tasks. In *Proceedings of the IEEE Real-Time Systems Symposium*.

Aydin, H., Melhem, R., and Mossé, D. 1999. Incorporating error recovery into the imprecise computation model. In *Proceedings of the IEEE International Conference on Real-Time Computing Systems and Applications*.

Aydin, H., Melhem, R., Mossé, D., and Mejia-Alvarez, P. 2004. Power-aware scheduling for periodic real-time tasks. *IEEE Trans. Comput. 53*, 5, 584–600.

Baruah, S. K., Howell, R. R., and Rosier, L. 1990. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Syst. 2*, 301–324.

BENOIT, A., CANON, L.-C., JEANNOT, E., AND ROBERT, Y. 2011. Reliability of task graph schedules with transient and fail-stop failures: complexity and algorithms. *J. Scheduling*, 1–13.

BINI, E. AND BUTTAZZO, G. C. 2005. Measuring the performance of schedulability tests. *Real-Time Syst. 30*, 1–2.

BUTTAZZO, G. C. 2004. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Real-Time Systems Series. Springer

CACCAMO, M. AND BUTTAZZO, G. 1998. Optimal scheduling for fault-tolerant and firm real-time systems. In *Proceedings of the IEEE International Conference on Real-Time Computing Systems and Applications*.

CASTILLO, X., MCCONNEL, S., AND SIEWIOREK, D. 1982. Derivation and calibration of a transient error reliability model. *IEEE Trans. Comput. 31*, 7, 658–671.

CHANDRA, V. AND AITKEN, R. 2008. Impact of technology and voltage scaling on the soft error susceptibility in nanoscale cmos. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*.

DABIRI, F., AMINI, N., ROFOUEI, M., AND SARRAFZADEH, M. 2008. Reliability-aware optimization for DVS-enabled real-time embedded systems. In *Proceedings of the International Symposium on Quality of Electronic Design*.

DICK, R., RHODES, D. L., AND WOLF, W. 1998. Tgff: Task graphs for free. In *Proceedings of the 6th International Workshop on Hardware/Software Co-design*.

EJLALI, A., AL-HASHIMI, B. M., AND ELES, P. 2009. A standby-sparing technique with low energy- overhead for fault-tolerant hard real-time systems. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*.

EJLALI, A., SCHMITZ, M. T., AL-HASHIMI, B. M., MIREMADI, S. G., AND ROSINGER, P. 2005. Energy efficient SEU-tolerance in DVS-enabled real-time systems through information redundancy. In *Proceedings of the International Symposium on Low Power and Electronics and Design*.

ELNOZAHY, E. M., KISTLER, M., AND RAJAMONY, R. 2002. Energy-efficient server clusters. In *Proceedings of the Workshop on Power Aware Computer Systems*.

ERNST, D., DAS, S., LEE, S., BLAAUW, D., AUSTIN, T., MUDGE, T., KIM, N. S., AND FLAUTNER, K. 2004. Razor: Circuit-level correction of timing errors for low-power operation. *IEEE Micro 24*, 6, 10–20.

GIRAULT, A. AND KALLA, H. 2009. A novel bicriteria scheduling heuristics providing a guaranteed global system failure rate. *IEEE Trans. Dependable Secure Comput. 6*, 4, 241–254.

HAZUCHA, P. AND SVENSSON, C. 2000. Impact of cmos technology scaling on the atmospheric neutron soft error rate. *IEEE Trans. Nucl. Sci. 47*, 6, 2586–2594.

IYER, R., ROSSETTI, D., AND HSUEH, M. 1986. Measurement and modeling of computer reliability as affected by system activity. *ACM Trans. Comput. Syst. 4*, 3, 214–237.

IZOSIMOV, V., ELES, P., AND PENG, Z. 2010. Value-based scheduling of distributed fault-tolerant real-time systems with soft and hard timing constraints. In *Proceedings of the 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia*.

JEJURIKAR, R. AND GUPTA, R. 2004. Dynamic voltage scaling for systemwide energy minimization in real-time embedded systems. In *Proceedings of the International Symposium on Low Power Electronics and Design*.

KUNG, S., WHITEHOUSE, H., AND KAILATH, T. 1985. *VLSI and Modern Signal Processing*. Prentice-Hall, Inc., Englewood Cliffs, NJ.

LIU, Y., LIANG, H., AND WU, K. 2010. Scheduling for energy efficiency and fault tolerance in hard real-time systems. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*.

LUENBERGER, D. 1984. *Linear and Nonlinear Programming*. Addison-Wesley, Reading MA.

MEJIA-ALVAREZ, P., AYDIN, H., MOSSÉ, D., AND MELHEM, R. 2000. Scheduling optional computations in fault-tolerant real-time systems. In *Proceedings of the IEEE International Conference on Real-Time Computing Systems and Applications*.

MELHEM, R., MOSSÉ, D., AND ELNOZAHY, E. 2004. The interplay of power management and fault recovery in real-time systems. *IEEE Trans. Comput. 53*, 2, 217–231.

PILLAI, P. AND SHIN, K. G. 2001. Real-time dynamic voltage scaling for lowpower embedded operating systems. In *Proceedings of the ACM Symposium on Operating Systems Principles*.

POP, P., POULSEN, K., IZOSIMOV, V., AND ELES, P. 2007. Scheduling and voltage scaling for energy(reliability trade-offs in fault-tolerant time-triggered embedded systems. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*.

PRADHAN, D. K. 1996. *Fault-Tolerant Computer System Design*. Prentice-Hall, Inc., Upper Saddle River, NJ.

QUAN, G. AND HU, X. 2007. Energy efficient dvs schedule for fixed-priority real-time systems. *ACM Trans. Embed. Comput. Syst. 6*, 4, 1–30.

WANG, Y., LIU, D., WANG, M., QIN, Z., AND SHAO, Z. 2010. Optimal task scheduling by removing inter-core communication overhead for streaming applications on mpsocs. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*.

WEI, T., MISHRA, P., WU, K., AND LIANG, H. 2006. Online task-scheduling for fault-tolerant low-energy real-time systems. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*.

WEISER, M., WELCH, B., DEMERS, A., AND SHENKER, S. 1994. Scheduling for reduced cpu energy. In *Proceedings of the USENIX Conference on Operating Systems Design and Implementation*.

YAO, F., DEMERS, A., AND SHENKER, S. 1995. A scheduling model for reduced cpu energy. In *Proceedings of the Annual Symposium on Foundations of Computer Science*.

ZHANG, Y. AND CHAKRABARTY, K. 2003. Energy-aware adaptive checkpointing in embedded real-time systems. In *Proceedings of the Conference on Design, Automation and Test in Europe*.

ZHAO, B., AYDIN, H., AND ZHU, D. 2008. Reliability-aware dynamic voltage scaling for energy-constrained real-time embedded systems. In *Proceedings of the International Conference on Computer Design*.

ZHAO, B., AYDIN, H., AND ZHU, D. 2011. Generalized reliability-oriented energy management for real-time embedded applications. In *Proceedings of the Design Automation Conference*.

ZHAO, B., ZHU, D., AND AYDIN, H. 2009. Enhanced reliability-aware power management through shared recovery technique. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*.

ZHU, D. 2006. Reliability-aware dynamic energy management in dependable embedded real-time systems. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*.

ZHU, D. AND AYDIN, H. 2006. Energy management for real-time embedded systems with reliability requirements. In *Proceedings of the International Conference on Computer Aided Design*.

ZHU, D. AND AYDIN, H. 2009. Reliability-aware energy management for periodic real-time tasks. *IEEE Trans. Comput. 58*, 10, 1382–1397.

ZHU, D., MELHEM, R., AND MOSSÉ, D. 2004. The effects of energy management on reliability in real-time embedded systems. In *Proceedings of the International Conference on Computer Aided Design*.

ZHUO, J. AND CHAKRABARTI, C. 2005. System-level energy-efficient dynamic task scheduling. In *Proceedings of the Annual Conference on Design Automation*.

ZIEGLER, J. F. 2004. Trends in electronic reliability: Effects of terrestrial cosmic rays. http://www.srim.org/SER/SERTrends.htm.

Received December 2011; revised August 2012, November 2012; accepted November 2012